

An Applied Graph-Theoretic Approach for Two-way to One-way Road Network Conversion

Tan Kok Chuen

School of Mechanical & Aerospace Engineering

A Master thesis submitted to Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005



TE
175
T161
2005

Abstract

It is not uncommon in the field of traffic engineering for two-way roads to be converted to one-way roads and vice versa. In these situations the traffic engineer would be faced with the problem of selecting direction for each of the one-way roads. This problem is named the one-way road network problem in this thesis. By modelling the road networks as graphs, an algorithm (called 'conversion algorithm' in this thesis) is developed; this enables the search for a global optimal solution for the given road network (based on a theoretical efficiency criterion) by means of enumeration, or exhaustive listing of solutions. The said conversion algorithm is next coupled with the minimum cut graph model which allows further refinement of the theoretical solution(s) obtained based on actual existing traffic conditions. The conversion algorithm together with the minimum cut graph model thus form a model named 'one-way road network model' for solving the above-mentioned problem. A case study is selected and simulated in order to verify the usefulness of the one-way road network model.

In the one-way road network model, two-way road networks are modelled by undirected graphs and one-way road networks by digraphs. The problem then becomes one of converting an undirected graph to a digraph. The conversion algorithm is developed for obtaining feasible digraphs; this is done by using the device of binary number counting to model the directionality of digraph edges so as to enumerate all possible digraphs. For a road network, only strongly connected graphs are "feasible" hence the concept of strong connectivity is incorporated in the conversion algorithm to weed out the "infeasible" digraphs. This is done by inspecting the indegrees and outdegrees of each of the enumerated digraphs. Those with zero indegree or outdegree are not strongly connected and are removed by the conversion algorithm. To arrive at a more efficient one-way road network, the theoretical aspects of network efficiency including junction-traversal delay are explored. Dijkstra's algorithm is modified from calculating the shortest path to calculating the network shortest total travel time. The modified algorithm is incorporated as part of the conversion algorithm and used to compare the digraphs enumerated to give the digraph(s) representing the road network with shortest total travel time.

With the selected 'theoretically efficient' road networks, actual traffic conditions are then taken into consideration and the various one-way road directions on the network are reversed and tested using a minimum-cut graph model. This allows efficient consideration of not only each direction but combinations of directions from different roads as well. By employing a minimum cut algorithm onto the graph model, the desired directions at critical junctions of the one-way road system for the case study are decided. This gives an efficient one-way road network layout based on actual traffic conditions for the selected case study.

The one-way road network is tested against the original two-way road network by means of computer simulation. Results of the simulation runs showed that the one-way road network performed better than the original two-way road network under given traffic load of the case study. This thus verifies the usefulness of the one-way road network model developed in this thesis.

Recommendations for future work include improving the conversion algorithm by graph manipulation techniques. Using heuristic reasoning on the results obtained, patterns such as 'twirls' in the one-way roads can be observed; this suggests that graph manipulation techniques identifying digraphs with such patterns can be developed as shortcuts to obtaining theoretically efficient digraphs. Such techniques allow the huge number of infeasible graphs to be removed without having to enumerate all possible digraphs, saving on time and computational resources. A limitation of the minimum-cut graph model caused by the triangle graphs employed can also be further researched. In addition, complexity aspects of the one-way road network problem can also be looked into.

In the above study, it is assumed that all roads used are unsignalised, that is road junctions are assumed to have no traffic lights; traffic flow consists of only motorised vehicles excluding motorcycles, bicycles and pedestrians; road network is analysed at a macro level, that is to say concepts and variables at the micro level (like driver behaviour, fuel emission, car-following theory) will not be taken into consideration; a fixed traffic demand is used, that is the amount of traffic used for various simulation runs will remain unchanged.

Acknowledgements

The author would like to extend his sincere gratitude to the following people:

Supervisor, A/Prof Lew Sin Chye, for his guidance and contributions throughout this project.

A/Prof Wong Yiik Diew, A/Prof Lum Kit Meng and A/Prof Tan Yan Weng for advice and help in matters pertaining to traffic assignment, transportation simulation and statistics.

Technical and administrative staff from Centre for Advanced Numeric Engineering Simulations, School of Mechanical & Production Engineering and Transportation Laboratory, School of Civil & Environmental Engineering.

Fellow graduate students from Rapid Prototyping Research Laboratory and MAE Graduate Student Club for their support and friendship.

Fellow hostelites from Hall of Residence 1, Hall of Residence 2 and Hall of Residence 9 for their care and companionship.

Especially to Willi Ching Choon Yong, May Win Naing, Valren Wee Zhi Sheng, Gabriel Lee Yeong Yeow, Lawrence Lim Chee Leong and Andy Chua Loy Ngee for their help in making this journey possible.

Last but not least friends, family and loved ones for their love and encouragement.

Table of Contents

ABSTRACT.....I
ACKNOWLEDGEMENTS.....III
TABLE OF CONTENTS.....IV
LIST OF FIGURES.....IX
LIST OF TABLES.....XII

I INTRODUCTION..... 1
1.1 Background..... 1
1.2 Objectives..... 4
1.3 Scope..... 4
1.4 Organisation of the Thesis..... 5

II LITERATURE REVIEW..... 7
2.1 Overview..... 7
2.2 Background information..... 7
 2.2.1 Combinatorics..... 7
 2.2.2 Graph Theory..... 8
 2.2.2.1 Graph..... 9
 2.2.2.2 Adjacency matrix..... 9
 2.2.2.3 Directed graph..... 10
 2.2.2.4 Indegree and outdegree..... 11
 2.2.2.5 Weighted graph..... 12
 2.2.2.6 Minimum spanning tree (MST)..... 13
 2.2.2.7 Strong Connectivity..... 14
 2.2.2.8 Shortest path algorithm..... 14
 2.2.2.9 Shortest total path..... 15
 2.2.2.10 Minimum cut and Maximum flow..... 16

2.2.3 Application of graph theory.....	18
2.2.3.1 Tree building in traffic assignment.....	18
2.2.3.2 Minimum-cut graph model in Module Orientation Problem.....	19
2.2.4 Traffic assignment.....	24
2.2.4.1 O-D matrix.....	25
2.2.4.2 Cost-flow function.....	26
2.2.4.3 Vehicle arrival distribution.....	27
2.2.4.4 Measurement-of-effectiveness (MOE).....	27
2.2.4.5 Traffic assignment.....	28
2.2.5 Traffic simulation.....	30
2.2.5.1 Driver behaviour.....	30
2.2.5.2 Traffic flow.....	31
2.2.5.3 Traffic simulation.....	32
2.2.5.4 Fixed time-interval scanning approach.....	33
2.2.6 One-way/two-way road network.....	33
2.3 Review of road network design.....	34
2.3.1 Road network design methodology.....	34
2.3.2 Conversion of two-way roads to one-way roads.....	36
2.3.2.1 Conversion of two-way roads to one-way roads for comparison.....	36
2.3.2.2 Shared Streets.....	39
2.3.2.3 Conversion of two-way roads to one-way roads to increase safety and reduce congestion.....	40
2.4 One-way urban road network problem.....	41
 III METHODOLOGY.....	 43
3.1 Overview.....	43
3.2 Two-pronged approach.....	43
3.3 Theoretical investigation.....	44
3.3.1 Requirements of the model.....	45
3.3.2 Assessing the road network.....	45
3.3.3 Deciding one-way road directions.....	46

3.3.4 Initial layout of one-way roads.....	47
3.3.5 A suitable initial one-way road network layout.....	49
3.3.6 Overview of conversion algorithm.....	51
3.3.7 Overview of one-way urban road network model.....	52
3.4 Validation process.....	53
3.4.1 Requirements of the validation process.....	53
3.4.2 Analyser tool.....	53
3.4.2.1 Examining the analyser tool.....	53
3.4.2.2 Simulation as the analyser tool.....	55
3.5 Choosing the WIJ road network for case study.....	57
IV THE MECHANICS OF THE CONVERSION ALGORITHM.....	59
4.1 Overview.....	59
4.2 Modelling the conversion algorithm.....	59
4.3 The mechanics of the conversion algorithm.....	67
4.3.1 First part: Convert an undirected weighted graph G to all its possible digraphs.....	67
4.3.2 Second part: Ensuring strong connectivity.....	71
4.3.3 Third part: Calculating STTT.....	73
4.3.3.1 Time taken to cross a road intersection.....	73
4.3.3.2 Finding the time taken to cross an intersection.....	74
4.3.3.3 Shortest path algorithm.....	81
V THE DEVELOPMENT OF THE CONVERSION ALGORITHM.....	88
5.1 Overview.....	88
5.2 The conversion algorithm and its development.....	88
5.2.1 Using sub-algorithms.....	88
5.2.2 The algorithms.....	89
5.3 Verification.....	113
5.4 Case study.....	115

VI	MINIMUM-CUT GRAPH MODEL.....	119
6.1	Overview.....	119
6.2	Formulation.....	119
6.2.1	Super node T	122
6.2.1.1	Case Study.....	122
6.2.2	Edge capacity.....	125
6.2.2.1	Case study.....	128
6.2.3	Min-cut graph model.....	130
6.2.3.1	Min-cut graph model construction and minimum cut.....	131
6.2.3.2	The graph construction algorithm and min-cut graph model.....	131
6.2.3.3	Case study.....	136
VII	SIMULATION.....	138
7.1	Overview.....	138
7.2	Background to traffic simulation.....	138
7.3	Choosing Paramics.....	139
7.3.1	The Paramics model.....	139
7.3.2	Reasons for using Paramics.....	142
7.4	Using Paramics.....	145
7.4.1	Building the network with Paramics.....	145
7.4.2	Calibration of Paramics.....	146
7.4.2.1	Network.....	146
7.4.2.2	Vehicle Mean Top Speed.....	147
7.4.2.3	Overall Simulation Configuration.....	147
7.4.2.4	Driver Behaviour Factors.....	148
7.5	Traffic data for the simulation.....	149
7.6	Test cases and results.....	152
7.6.1	Test cases.....	152
7.6.2	Results and discussion.....	155
7.6.2.1	Results.....	155
7.6.2.2	Discussion.....	161

VIII CONCLUSION AND UNSOLVED PROBLEMS.....	168
8.1 Overview.....	168
8.2 Conclusion.....	168
8.3 Contributions to knowledge.....	169
8.4 Limitations and recommendations for further work.....	170
8.4.1 Conversion algorithm.....	171
8.4.1.1 Graph decomposition.....	171
8.4.1.2 Node condensation.....	172
8.4.1.3 Strong connectivity.....	173
8.4.1.4 Isomorphism.....	174
8.4.1.5 Twirling.....	175
8.4.2 Min-cut graph model.....	178
8.4.2.1 Triangle graph.....	178
8.4.3 Simulation.....	179
8.4.3.1 Evening peak period.....	179
8.4.4 NP-completeness.....	180
REFERENCES.....	181
APPENDIX A: List of Publications.....	192
APPENDIX B: Complexity of problems.....	193
APPENDIX C: Dijkstra Algorithm.....	196
APPENDIX D: Type of movement/turn present.....	198
APPENDIX E: Verifying the conversion algorithm.....	199

List of Figures

Figure 1.1: A simple two-way road network..... 2

Figure 2.1: A graph, G 9

Figure 2.2: Adjacency matrix of graph, G 10

Figure 2.3: Digraph, G_I 11

Figure 2.4: Adjacency matrix of G_I 11

Figure 2.5: A weighted graph..... 12

Figure 2.6: Spanning tree..... 13

Figure 2.7: Weighted spanning tree..... 14

Figure 2.8: Cut of a graph..... 16

Figure 2.9: A given road network..... 18

Figure 2.10: An example of PCB..... 19

Figure 2.11: Module placement on PCB..... 20

Figure 2.12: Flipping the modules..... 21

Figure 2.13: A five-module example..... 22

Figure 2.14: Triangle graph for module 2 and module 3..... 23

Figure 2.15: Min-cut graph model of the five-module example..... 24

Figure 2.16: Increased distance in one-way road..... 34

Figure 3.1: A simple strongly connected one-way road network..... 47

Figure 3.2: Obtaining MOE for assessment of the network..... 49

Figure 3.3: First-level flowchart for the analyser tool..... 54

Figure 3.4: WIJ road network..... 57

Figure 4.1: Two possible ways of changing an undirected edge to a directed edge... 60

Figure 4.2: A simple graph example..... 61

Figure 4.3: Representing directed edge with “0” and “1”..... 64

Figure 4.4: An adjacency matrix..... 68

Figure 4.5: 1 by 3 matrix to represent an edge of the digraph..... 69

Figure 4.6: 4 by 3 matrix to represent the digraph..... 70

Figure 4.7: 4 by 4 matrix to represent the digraph..... 71

Figure 4.8: Nodes with zero indegree and outdegree..... 72

Figure 4.9: Possible intersections for the experiment.....	75
Figure 4.10: A four-legged intersection.....	76
Figure 4.11: Speed of A against actual time.....	78
Figure 4.12: Comparing d_{st} with $(d_{sm} + d_{mt})$	84
Figure 4.13: Comparing the road network with the 3-D matrix.....	85
Figure 5.1: Part (ii) and part (iii) nested within part (i).....	90
Figure 5.2: Flowchart for Algorithm 5.1.....	91
Figure 5.3: Flowchart for Sub-Algorithm 5.2.....	93
Figure 5.4: A row in M.....	94
Figure 5.5: Flowchart for Sub-Algorithm 5.3.....	95
Figure 5.6: Flowchart for Sub-Algorithm 5.4.....	98
Figure 5.7: Flowchart for Sub-Algorithm 5.5.....	100
Figure 5.8: Flowchart for Sub-Algorithm 5.6.....	102
Figure 5.9: Path from node s (past node m) to node t.....	104
Figure 5.10: Flowchart for Sub-Algorithm 5.7.....	108
Figure 5.11: Flowchart for Sub-Algorithm 5.8.....	110
Figure 5.12: Flowchart for Sub-Algorithm 5.9.....	112
Figure 5.13: A simple graph example.....	113
Figure 5.14: All-to-all shortest time matrix.....	114
Figure 5.15: WIJ Network.....	115
Figure 5.16: WIJ network in the form of a weighted graph.....	116
Figure 5.17: Digraph of WIJ network obtained.....	118
Figure 6.1: Idealised WIJ network.....	124
Figure 6.2: Triangle graph.....	127
Figure 6.3: Triangle graph for roads H and J	129
Figure 6.4: Flowchart for Algorithm 6.1.....	134
Figure 6.5: Min-cut graph model of the WIJ network.....	135
Figure 6.6: Resultant one-way road network for WIJ.....	137
Figure 7.1: 3-D graphics visualisation of Paramics.....	141
Figure 7.2: 5 zones are defined.....	149
Figure 7.3: Graphical results for a typical simulation run for Cases A, B and C.....	154

Figure 7.4: Case A at 8:50:00am.....	157
Figure 7.5: Case B at 8:50:00am.....	159
Figure 7.6: Case C at 8:50:00am.....	160
Figure 7.7: Road networks of Case B and Case C.....	166
Figure 8.1: Graph manipulations using graph decomposition and node condensation	172
Figure 8.2: Graph manipulation for the Strong Connectivity of a Digraph.....	173
Figure 8.3: Isomorphic graphs.....	175
Figure 8.4: Twirls observed in the STTT graphs.....	176
Figure 8.5: Two ways of twirling.....	177
Figure 8.6: Considering 3 by 3 graph.....	178
Figure B.1: Asymptote behaviours of $O(n)$ and $O(\lg n)$	194
Figure C.1: All-to-all shortest path matrix.....	197
Figure D.1: From i to j	198
Figure D.2: From k to i to j	198

List of Tables

Table 2.1: O-D matrix..... 25

Table 4.1: Forming possible digraphs..... 61

Table 4.2: Incrementing a 3-digit binary number..... 64

Table 4.3: Speed of A against actual time..... 77

Table 4.4: Time taken to cross the intersection..... 79

Table 5.1: Weighted adjacency matrix of Figure 5.16..... 117

Table 5.2: Weighted adjacency matrix of the digraph with the minimum STTT..... 117

Table 6.1: Roads not to be flipped..... 125

Table 6.2: Additional delay for flipping the roads H & J 129

Table 6.3: Additional delay for “flipping” the edges H , J and M 130

Table 6.4: Edge capacities for the edges of the three roads..... 130

Table 7.1: Zones definition..... 150

Table 7.2: O-D matrix for WIJ network..... 151

Table 7.3: Tabular results for a typical simulation run for Cases A, B and C..... 153

Table 7.4: Overall delay for the whole network for Cases A, B and C..... 156

Table E.1: 128 possible digraphs..... 200

I INTRODUCTION

1.1 Background

Traffic congestions along stretches of roads are not uncommon, particularly during the peak morning hour or the evening rush hour. As stated by Ortúzar and Willumsen [1] and Raney *et al.* [2] congestions are serious issues in several major cities around the world (like New York in United States, London in England and Zurich in Switzerland just to name a few) where the road network is intricately dense and the buildings are closely packed.

Various measures have been implemented to ease such blockage of the arterial roads. According to Ben-Joseph [3], Appleyard [4], Branston [5], Beimborn [6] and Brilion and Blanke [7], such measures include flyovers to bypass congested areas, varying traffic signal phasing, addition of road lanes, improving subway transit service and even traffic calming.

As stated by Greenshield [8], a simplistic way of looking at traffic congestion is that the road has reached its capacity and can no longer accept any more vehicles to pass through. A queue will thus form and will propagate downstream, resulting in traffic congestion. Nevertheless as stated by TRB [9], the infrastructure of road network is somewhat fixed and this does not allow capacity of the road to be increased through road widening. According to Stemley [10], a suggested way to overcome this problem of increasing the road capacity with fixed road infrastructure is by changing certain two-way roads of the network into one-way thereby increasing the number of available lanes on particular roads.

According to Raney *et al.* [2] and Ben-Joseph [3], this conversion of two-way roads into one-way roads to curb congestion has been successful in certain areas like Zurich and San Diego (United States) but not in others like West Palm Beach, Florida (United States), as stated by Appleyard [4]. Due to the varying levels of success of the conversion (of two-

way road to one-way road), the decision to change roads from two-way to one-way or vice versa are subjects of debate in Vancouver, Canada [11].

According to Stemley [10], the choices of the directions of the one-way roads play a major factor in the success of the conversion (of two-way road to one-way road). This thus becomes an important issue to a transport engineer designing one-way roads for a city since he is now faced with the intricate decision of selecting the direction of the various one-way roads. This is illustrated in the following example. Figure 1.1 shows a two-way road network. Given a stretch of two-way road x , if it was changed to a one-way road it would give a problem of deciding the direction of x (from A to B or B to A). What should the decision be based upon?

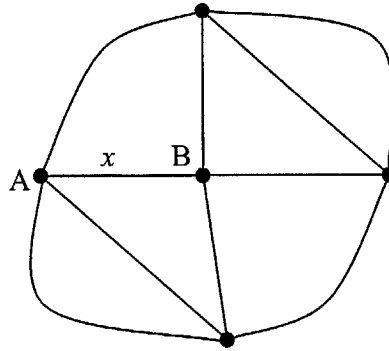


Figure 1.1: A simple two-way road network

In addition, all other roads in Figure 1.1 are to be changed to one-way as well. Some of these roads intersect one another along the way as shown. How does one decide which direction these one-way roads should be?

To select the best combination is not an easy task. It certainly cannot be done by looking at each road individually. As Ortúzar and Willumsen [1] has stated, changing any road by itself would create an impact on the whole road network. For example, if a new road is to be built, the portion of population that will use it is not the only group that will benefit from it. Others may benefit from the fact that other roads in the road network will now have less congestion since some vehicles will be diverted to the new road. The one-way urban road problem is therefore one in which the direction of each road needs to be decided and the best combination of directions of all roads selected.

As this is a complex problem, it is considered desirable to have a rational procedure for selecting the combination that would be the best economically from the standpoint of the road user population as a whole. While an economic criterion rarely is or should be the sole criterion for investment decisions on public works, the most economically efficient solution can provide a basis for planning deliberations.

By taking a graph-theoretic approach, the existing two-way road network is first modelled as an undirected graph. The problem simply stated then becomes one of finding a methodology to convert an undirected graph to a directed graph (or digraph) which is optimal for the objective of the conversion. This whole process is referred to as the Conversion Algorithm in Chapter 4.

Although the goal of research is that of solving a general problem of the same nature, the whole process is best explained using the Woodsville Interchange Junction (WIJ) road network as a case study. Based on the existing network, thousands (2^{21} or 2097152) of digraphs can be obtained. In larger problems, the numbers may even take years to compute, depending on the efficiency of the algorithm used. However not all of these solutions are feasible digraphs. For our purposes, only 'strongly connected' (refer to Section 2.2.2.7) digraphs are feasible digraphs (as explained in Section 4.2). In order to weed out the infeasible digraphs, the concept of indegree and outdegree of nodes is explored and utilised in a sub-algorithm. Of the feasible digraphs then obtained, the optimal digraph depends on the objective function (normally a cost function) used. In transportation theory, this cost function is referred to as the measurement-of-effectiveness, MOE (as explained in Section 2.2.4.4). A model based on the concept of universal accessibility (minimum total 'all-to-all' distance) is employed (as explained in Section 3.3.5). In order to compute this MOE, a number of known algorithms are explored, and Dijkstra's algorithm is eventually selected (as explained in Section 4.3.3.3)

Having obtained the notional optimal feasible digraph, the solution is further refined to take into consideration the cumulative delay experienced by all drivers at all junctions in all journeys. To address the sub-problem of delay at junctions which has a psychological

impact on drivers in their assessment of how good a road network is, the techniques and algorithms explored in the Module Orientation problem is adapted. This is referred to, for lack of a better terminology at this point in time, as 'minimum-cut graph model' and discussed in Section 3.3.3.

Results for the WIJ case study are then run on simulation in order to validate the model (refer to Section 7.6.2). A commercially proven simulation package, Paramics, is used to run the simulation, this is to justify for the change of the WIJ from a two-way road network to a one-way road network.

1.2 Objectives

This project aims to establish the following:

1. To develop a one-way urban road network model for use in the conversion of a two-way urban road network to a one-way urban road network by:
 - Investigating and selecting a suitable shortest path algorithm, from graph theory, for use in the conversion of two-way network to one-way network
 - Creating an algorithm for the conversion (named conversion algorithm) using graph theory
 - Integrating the shortest path algorithm and the conversion algorithm to form the one-way urban road network model to generate a one-way road network with shortest total travel time
 - Using minimum-cut (min-cut) graph model to further refine the resultant one-way road network (obtained from the conversion algorithm) based on actual traffic data
 - Proving that the change in delay in the one-way road network is equivalent to the minimum cut of the min-cut graph model.
2. Validation of the one-way urban road network model by means of simulation of the urban road network

1.3 Scope

The scope of this project is restricted as follows:

- i) Simulation of the road network using traffic flow from a fixed demand of traffic as given in the case study. This assumes that the traffic flow will remain unchanged throughout the simulation and the same traffic flow will be used for various simulation testing.
- ii) Simulation of the road network using only traffic flow from motorised vehicles (not inclusive of motorcycles). This assumes that non-motorised vehicles like motorcycles, bicycles and pedestrians will not be present in the road network.
- iii) Road network studies will only include unsignalised intersection, which are road junctions with no traffic signal or road signs. Signalised intersections will be assumed to be unsignalised.
- iv) Studies on the road network will be done at a macro level; this is due to the fact that the focus of this study is on the network as a whole. That is to say various concepts and variables described at a micro level (such as driver behaviour, car following theorem, fuel emission) will not be taken into consideration.
- v) Studies on the traffic flow and the road network will include only that as contained within the network of roads given in the case study. Surrounding roads and vehicular traffics will not be taken into the consideration.

The validation work would be carried out using Paramics, a commercially proven microscopic simulation modelling software [12].

1.4 Organisation of the Thesis

The thesis is organised as follows:

Chapter 1 gives an introduction to the one-way urban road design. A way to rationalise conversion of the two-way to one-way urban road network is proposed.

Chapter 2 gives some background knowledge on the research area, a brief survey of previous work done on road conversions and the problem statement of this thesis.

Chapter 3 gives an outline of the methods employed to solve the one-way urban road problem. Reasoning for the various methods used and the logical flow of each method in solving the problem is given.

Chapter 4 is an overview of the various graph theory theorems employed in this project and how they are used in the development of the conversion algorithm.

Chapter 5 describes the development of the mechanism of the conversion algorithm used for the “two-to-one-way” conversion and explain in detail the various considerations for the implementation of the conversion algorithm.

Chapter 6 describes the use of the minimum cut graph model developed by Cheng *et al.* [13] as a means to refine the one-way road network problem for the situation of a given set of actual traffic data and how this minimum cut graph model is incorporated into the one-way road network model.

Chapter 7 discusses the use of Paramics, the traffic simulation package employed in this project. It then moves on to a case study of the WIJ road network in Singapore.

Chapter 8 presents the conclusions of the project, the project’s contribution to knowledge as well as limitations and recommendations.

II LITERATURE REVIEW

2.1 Overview

The chapter begins with some background information (on combinatorics, graph theory, traffic assignment, simulation and one-way/two-way road network) needed for a better understanding of the research work done in this project. It moves on to review how road networks are planned and how the decisions to build them are made. The process of planning for “two-way to one-way” road conversion is reviewed next and the chapter ends by identifying a need for a model to perform such a conversion.

2.2 Background information

2.2.1 Combinatorics

Combinatorics is the mathematics of discretely structured problems. Although its boundaries are not easily defined, combinatorics includes graph theory, enumeration, and designs. It is a subject which in the past was studied principally for its aesthetic appeal. Today's modern technology with its vital concern for the discrete has given combinatorics new challenges and a new seriousness of purpose. In particular, since computers require discrete formulations of problems, combinatorics has become indispensable to modern computer science.

Loosely speaking, combinatorics is the science of counting. This is the area of mathematics in which we study families of sets (either finite or infinite) with certain characteristic arrangements of their elements or subsets, and ask what combinations are possible, and how many there are. More often than not combinatorics is used to select the element of a given set with a given characteristic or property (for example the route with the shortest distance) in an attempt to optimise the given set by means of a certain counting method. Such methods include numerous elementary topics, such as enumerating all possible permutations or combinations of a set (either finite or infinite).

For simple combinatorial optimisation problems, the solution may be found by going through all possibilities within the set (that is enumeration) and selecting the element that satisfies the given condition of the problem. However this poses a challenge when dealing with very large finite sets. Quite often, seemingly simple combinatorial problems required not only sophisticated counting methods but well drawn data structures as well as heavy duty computational power. A very good example is the deceptively simple travelling salesman problem (TSP). It can be stated very simply:

A salesman travels from one city to another cyclically; he only visits each city once and ends up where he started. In what order should he visit all the cities so as to minimise travelling distance?

This seemingly simple problem has intrigued both mathematicians as well as computer scientists into using various methods such as genetic algorithms [14], ant colony optimisation [15] and neural networking [16] just to name a few, to solve various instances of TSP. The difficulty of the TSP increases exponentially when the number of cities increases. A recent breakthrough by Applegate *et al.* [17] solved a 13,506 cities TSP by running a branch-and-cut algorithm on a network of 48 workstations, including Digital Alphas, Intel Pentium IIs, Pentium Pros and Sun UltraSparcs.

As shown, seemingly simple combinatorial problems should not be underrated since they can be a big challenge when the search space increases. More often than not, there is a need for sophisticated counting methods to eliminate elements in the set without even having to explicitly consider them; else the number of possibilities explodes exponentially when given a bigger search space.

2.2.2 Graph Theory

A body of network theory combining nodes and linkages evolved in the transportation field primarily for the economics of movement of people and goods [18, 19]. Nodes of different sizes, such as cities, towns and villages, are included. Nodes can be sources and/or destinations, and routes can have capacities. By combining these variables into a

network, which also can vary in many structural ways, complex systems can be created. Spatial models and equations describing them have been formulated and used for different purposes. Examples include network connectivity, network circuitry, linkages per node, flows in a capacitated network, networks as weighted graphs, graph-theory interpretation of hierarchies, nodal accessibility, and optimal transport pattern [19].

2.2.2.1 Graph

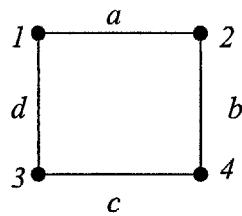


Figure 2.1: A graph, G

Figure 2.1 shows a graph, G , with nodes (or vertices) $\{1, 2, 3, 4\}$ and edges (also known as links or arcs) $\{a, b, c, d\}$. According to Ahuja *et al.* [20], formally the graph is represented as $G(V,E)$ which can be interpreted as a graph, G , with a node set of V and an edge set of E . Sometimes the edge can be represented by the nodes that it is attached to. For example edge a can be represented as $(1, 2)$ meaning the edge connecting nodes 1 and 2 .

Graph G depicted here is an undirected (or plain) graph. All of its edges have no directions. This implies that movement between node 1 and node 2 (through edge a) can be both ways: from node 1 to node 2 and from node 2 to node 1 . A directed graph will be discussed in Section 2.2.2.3.

2.2.2.2 Adjacency matrix

Besides being represented in graphic form (as shown in Figure 2.1), graphs may be represented in matrix form. This matrix form is known as the adjacency matrix of the graph.

Using the same graph, G , in Figure 2.1. Its adjacency matrix is:

$$\begin{array}{cc}
 \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} & \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{matrix} \\
 (a) & (b)
 \end{array}$$

Figure 2.2: Adjacency matrix of graph, G

Figure 2.2(a) shows the adjacency matrix of G . For easier understanding, the rows and columns of the matrix are numbered as shown in Figure 2.2(b). Reading across row 1 gives the nodes that are connected to node 1 by means of an edge.

For row 1:

- column 1 shows a “0” meaning node 1 is not connected to node 1
- column 2 shows a “1” meaning node 1 is connected to node 2
- column 3 shows a “1” meaning node 1 is connected to node 3
- column 4 shows a “0” meaning node 1 is not connected to node 4

The same interpretation can be applied to rows 2, 3 and 4 thereby portraying G in its adjacency matrix form. In an adjacency matrix both the rows and the columns represent the nodes of the graph. Hence the same information is repeated twice – in some schemes only the upper triangular matrix is employed.

2.2.2.3 Directed graph

Directed graphs or digraphs are graphs containing edges with directions. Figure 2.3 shows a directed graph, $G1$. It is similar to G ; however all its edges have a direction.

Edge a is directed from node 1 to node 2. This implies that movement from node 1 to node 2 through edge a is possible. However movement from node 2 to node 1 through edge a is not possible. To get to node 1 from node 2 one has to move through edge b to node 4, then to node 3 through edge c and finally to node 1 through edge d .

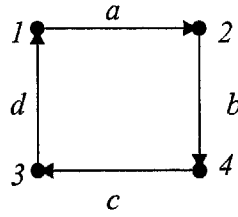
Figure 2.3: Digraph, $G1$

Figure 2.4 shows the adjacency matrix of $G1$. The means of generating the adjacency matrix of a digraph is the same as that of an undirected graph.

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2.4: Adjacency matrix of $G1$

2.2.2.4 Indegree and outdegree

As stated by Diestel [22], indegree of a node of a digraph refers to the number of edges pointing towards the node while outdegree of a node in a digraph refers to the number of edges pointing away from the node. Taking $G1$ as an example, node 1 has one edge (edge d) pointing towards it and one edge (edge a) pointing away from it. Therefore, the values of node 1's indegree and outdegree are both 1.

Indegree and outdegree of a digraph can be obtained from the digraph's adjacency matrix [22]. To illustrate this, the adjacency matrix in Figure 2.4 is used as an example. Row 1 of the matrix shows "0 1 0 0": this implies that node 1 has a directed edge to node 2 and none to all the rest of the nodes of $G1$. Hence, summing up all values across row 1 gives the number of edges directed away from node 1, which is the outdegree of node 1.

Looking down column 1, row 1 with value 0 means there is no edge connecting from node 1 to node 1. Row 2 with value of 0 means there is no edge connecting from node 2 to node 1. Row 3 with value of 1 means there is a directed edge from node 3 to node 1.

Row 4 with value of 0 means there is no edge connecting from node 4 to node 1. This implies that looking down column 1 would give all the nodes that have a directed edge to node 1. Hence, summing up all values down column 1 gives the number of edges directed towards node 1, that is the indegree of node 1.

2.2.2.5 Weighted graph

A weighted graph is a graph in which each of its edges has a numerical value attached to it. This value is called the arc capacity, edge capacity or weight of the edge. The weight of the graph is defined as the sum of all the edge capacities of the graph [21].

Figure 2.5(a) shows a figure of a weighted graph while Figure 2.5(b) shows its adjacency matrix. In the given graph the weight of edge (1,2) is 4. The weight of the graph is the sum of all the weights of all edges, giving a value of 23 in this case.

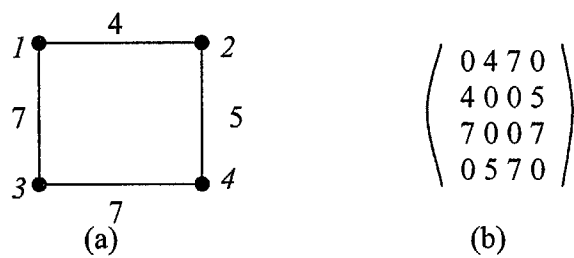


Figure 2.5: A weighted graph

In the adjacency matrix, should two nodes i and j be connected by an edge with a weight of n then a value of n will be placed in the i^{th} row and the j^{th} column of the matrix. For example, nodes 1 and 2 are connected by an edge of weight 4, hence a value of 4 will be placed on the element that occupies the 1st row and the 2nd column of the matrix. Zero value means that the two nodes are not adjacent and do not have an edge connecting them.

2.2.2.6 Minimum spanning tree (MST)

In order to understand the concept of minimum spanning tree (MST) the terms path, cycle and tree are first explained. According to Diestel [22]:

Path

A path from node i to node j refers to a series of edges that can connect node i to node j .

Cycle

A cycle refers to a path that leads away from a given node i and back to itself. In Figure 2.3, the path containing the edges $\{a, b, c, d\}$ forms a cycle from node l back to itself.

Tree

A tree is a graph in which any of its two nodes are connected by exactly one path. In addition to that, a tree does not contain any cycles.

Minimum spanning tree or MST

A spanning tree of a graph is a subset of the original graph [21]. It is a tree that contains all the nodes of the original graph but not necessarily all its edges. Figure 2.6(a) shows a graph while a possible spanning tree from the graph is depicted in Figure 2.6(b). Figure 2.6(c) shows another possible spanning tree from the same graph.

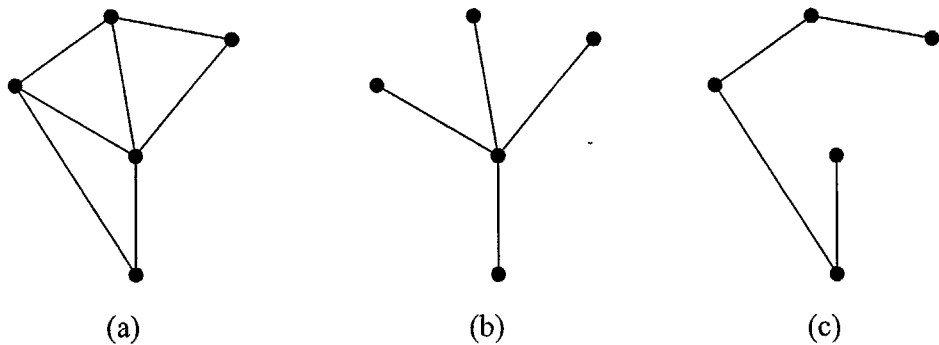


Figure 2.6: Spanning tree

For a given weighted graph as shown in Figure 2.7(a), each possible spanning tree may have a different weight. Spanning tree in Figure 2.7(b) has a weight of 10 while spanning

tree in Figure 2.7(c) has a weight of 4. The spanning tree whose weight is the smallest is deemed the MST of the weighted graph [21].

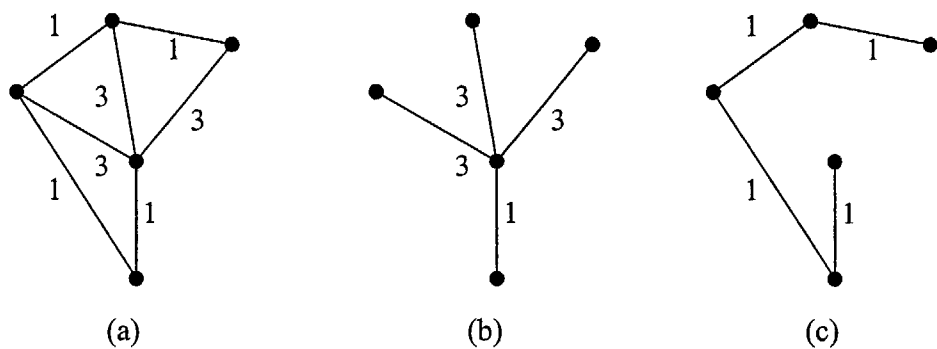


Figure 2.7: Weighted spanning tree

2.2.2.7 Strong Connectivity

Strong connectivity refers to the concept of strongly connected graphs [22]. A graph is strongly connected if, for all given nodes of the graph, there are paths connecting each node to all other nodes in the graph.

2.2.2.8 Shortest path algorithm

Given two nodes i and j of a graph, there may be more than one path (of varying lengths) leading from i to j . According to Gross and Yellen [21], the shortest path between i and j refers to the path that has the shortest length. The length of a path is the sum of the weights of all its edges.

As stated by Ortúzar and Willumsen [1], the shortest path algorithm is a method employed to find the shortest path between any given node to any other node. There are two types of shortest path algorithms [22]:

- i) One-to-all
- ii) All-to-all

One-to-all

Shortest path algorithms that are one-to-all require a node to act as an initiator [20]. After the initiator has been assigned the algorithm will give the shortest paths from the initiator to all other nodes in the graph.

In order to get the shortest paths from all nodes to all other nodes, every node within the graph will take a turn to act as the initiator [20]. This implies that for a graph consisting of 4 nodes the algorithm will be applied 4 times to get the shortest paths from all 4 nodes to all other nodes.

All-to-all

For shortest path algorithms that are all-to-all, these need to be applied only once onto the graph. The result of that single application will give the shortest paths from all nodes to all other nodes in the graph. This implies that for a graph with 4 nodes the algorithm needs to be applied only once to get the shortest paths from all 4 nodes to all other nodes.

2.2.2.9 Shortest total path

Shortest total path as defined by Wang [23] (for any graph – undirected or directed) refers to the sum of all shortest paths from every node to all other nodes in the graph. Given a graph, G , with n nodes: $1, 2, \dots, (n-1), n$, the shortest total path is defined as follows:

Sum of shortest paths from 1 to all other nodes

$$= \text{Shortest path from } 1 \text{ to } 2 + \text{Shortest path from } 1 \text{ to } 3 + \dots + \text{Shortest path from } 1 \text{ to } n$$

(2.1)

Shortest total path of G

$$\begin{aligned}
 &= (\text{Sum of shortest path from } 1 \text{ to all other nodes}) + \\
 &\quad (\text{Sum of shortest path from } 2 \text{ to all other nodes}) + \dots + \\
 &\quad (\text{Sum of shortest path from } n-1 \text{ to all other nodes}) + \\
 &\quad (\text{Sum of shortest path from } n \text{ to all other nodes}) \\
 &(2.2)
 \end{aligned}$$

2.2.2.10 Minimum cut and Maximum flow

Given a weighted graph, $G(V,E)$, such that s is a subset of V and t is $(V-s)$. Based on Gross and Yellen [21], a cut C on G between s and t is a set of edges such that every directed path from s to t passes through at least one edge in C . The capacity of the cut C is the sum of all its edge capacities. Figure 2.8 shows an example of a cut C . The capacity of C in this case would be the sum of all its edge capacities $(1+2+2+3)$, giving a total of 8.

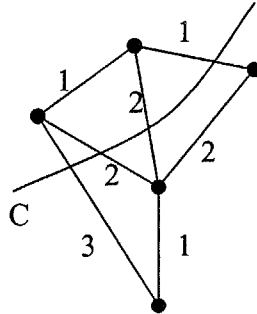


Figure 2.8: Cut of a graph

A minimum cut is a cut that separates the graph into two separate partitions such that the capacity of the cut is the lowest possible in the given graph.

The maximum flow on the other hand is defined as the largest amount of flow possible for any given weighted graph from a source, s , to a sink, t . The weights of the edges of the graph are assumed to be the capacity or some limiting factors to the flow of the graph.

According to Ford and Fulkerson [24], the maximum flow of a given graph equates to the capacity of the minimum cut of the graph. This can be explained by considering the graph as a series of linked water pipes, the maximum flow of this network equals to the maximum flow in the smallest pipe in the network, that is the flow is limited by bottleneck pipe. The minimum cut can be seen as a distributed bottleneck, that is a collective bottleneck for the whole network instead of a simple bottleneck.

As discussed and cited by Chalermsook *et al.* [25], there has been numerous algorithms developed for the maximum flow problem ever since the first minimum cut algorithm developed by Ford and Fulkerson to solve the problem. This includes the large neighbourhood automated timetabling by Ahuja *et al.* [20] and a deterministic near linear time algorithm for finding minimum cuts in planar graph by Chalermsook *et al.* [25]. As stated previously, the answer to the maximum flow problem gives the answer to the capacity of the minimum cut [24]. Thus such algorithms are applicable to the minimum cut problem as well. However, to find the minimum cut an additional step has to be taken after application of the maximum flow algorithm. All edges that contain a flow equivalent to their capacity are marked. A cut that partitions the network into two distinct subgraphs and consisting of only the marked edges forms the minimum cut. The explanation behind this is that since the minimum cut can be seen as a collective bottleneck for the whole network, therefore the group of bottlenecks (which are the marked edges with used up capacities) that will “cut” the graph into two is the minimum cut. It is noted that often, not all the marked edges are used and multiple cuts are possible.

Like the TSP, the maximum flow/minimum cut problem is known to be Non-deterministic Polynomial (NP) complete (see Appendix B). This means that it is one of those combinatorial problems that is so complex (especially when the set gets bigger) that there is no one general solution that will solve all cases of this problem. It is noted that solutions to specific instances of the maximum flow/minimum cut problem do exist, however that is not within the scope of discussion.

2.2.3 Application of graph theory

This section discusses two applications of graph theory in relevant research areas. One application is in traffic assignment and the other in the Module Orientation Problem.

2.2.3.1 Tree building in traffic assignment

Given a network of roads, traffic assignment is the process whereby the volumes of vehicles in various routes of the road network are decided. Tree building is but a step in the traffic assignment process [1].

The road network is first represented as a graph [as shown in Figure 2.9(a)]. The next step of the traffic assignment is the tree building process. Tree building proceeds to build the MST from the given road network [1] for a given source. This is done by using a shortest path algorithm and iteratively finding the shortest paths from the given source to the rest of the nodes [1]. Let the source be node A, then the MST built is shown in Figure 2.9(b). Traffic is then loaded onto the road network. Vehicles will choose only the paths available in the MST. It is noted however that the weights attached to the edges of the road network does not represent the distance of the edges but rather, the cost of traversing the given edge (this cost is discussed in Section 2.2.4.2). The role in which tree building plays in traffic assignment is explained in greater detail in Section 2.2.4.5.

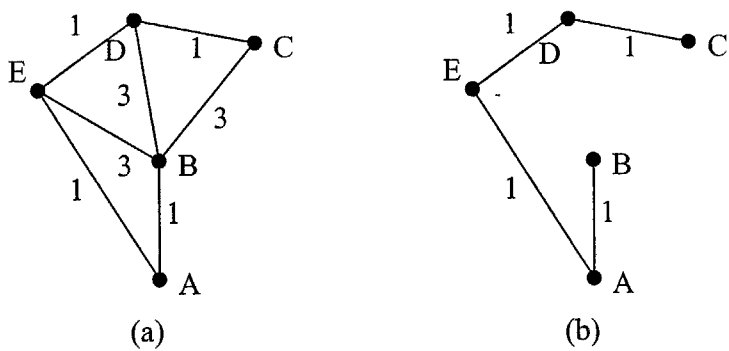


Figure 2.9: A given road network

2.2.3.2 Minimum-cut graph model in Module Orientation Problem

In this section, the Module Layout Problem [26] as well as the Module Orientation Problem [13] are discussed prior to the minimum-cut (or min-cut) graph model.

Module Layout Problem and Module Orientation Problem

Both Module Layout Problem and Module Orientation Problems are problems that occur during the design of printed circuit board (PCB) as shown in Figure 2.10. Various circuit components (or modules) like resistors, semi-conductors and even microprocessors need to be placed onto the circuit board. In the placement of these modules onto the PCBs, in addition to reducing net length, the physical area occupied by the modules can be reduced by the manner in which the modules are placed onto the PCBs. Net length refers to the total length of physical circuitry required. Figures 2.11(a) and (b) show two ways in which modules can be placed onto the PCB. As shown, the layout in Figure 2.11(b) would be a better choice in terms of the area that the modules occupy as well as the net length (depicted by the dotted lines) needed to link various modules together. The problem, as illustrated, of finding an optimum layout for the placement of modules is known as the Module Layout Problem.

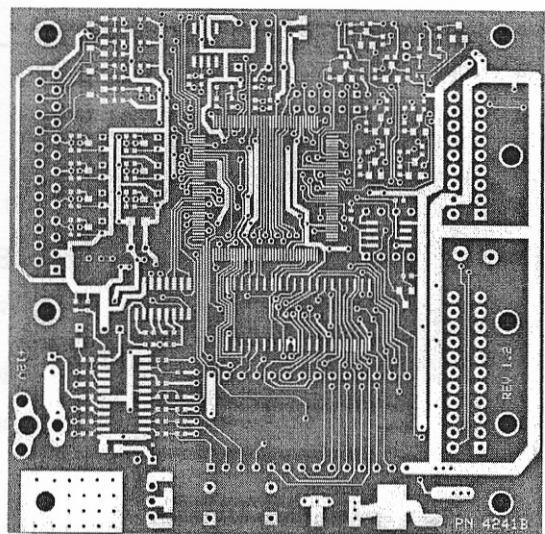


Figure 2.10: An example of PCB

After the modules are placed on the PCB, the net length can be further reduced by different orientations of modules. This is illustrated in Figure 2.12 that shows the same PCB depicted in Figure 2.11(b). In Figure 2.12, pins connecting the three modules (*a*, *b*, and *c*) are added in to depict where the net connects each of the modules. Pin *a* that is found on both Module 1 and Module 2 shows the positions where the net will connect Module 1 and Module 2 together. Pin *b* shows where the net will connect Module 2 and Module 3 together. Pin *c* shows where Module 1 and Module 3 connect to each other.

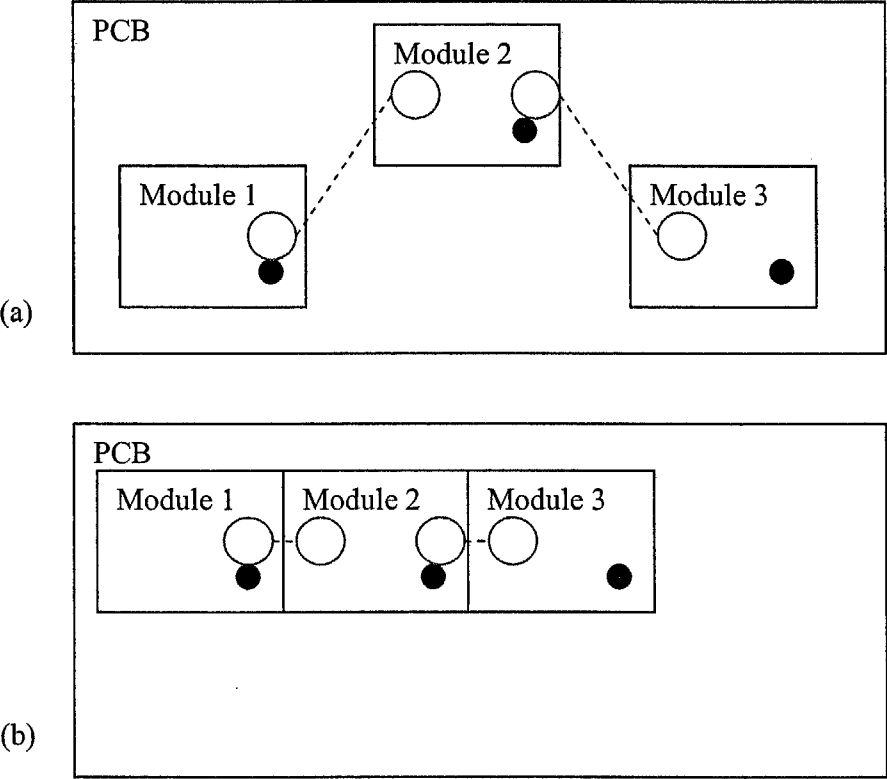


Figure 2.11: Module placement on PCB

Figure 2.12(a) shows the initial orientation of each of the modules on the PCB. Figure 2.12(b) shows the PCB after all three of the modules are flipped horizontally. Comparing positions of pin *a* in Figure 2.12(a) and (b), after flipping both Module 1 and Module 2 [in Figure 2.12(b)] length of the net needed to connect Module 1 and Module 2 using pin *a* is reduced. This is the same for pins *b* and *c*.

As shown by Preas and Karger [27] and Yao *et al.* [28], the overall net length can be further reduced by changing the orientation of the modules. The problem of finding the

optimum orientations of the modules is known as the Module Orientation Problem. Generally in solving the Module Orientation Problem, it is assumed that the modules are already in their optimum positions (that is the Module Layout Problem has been solved).

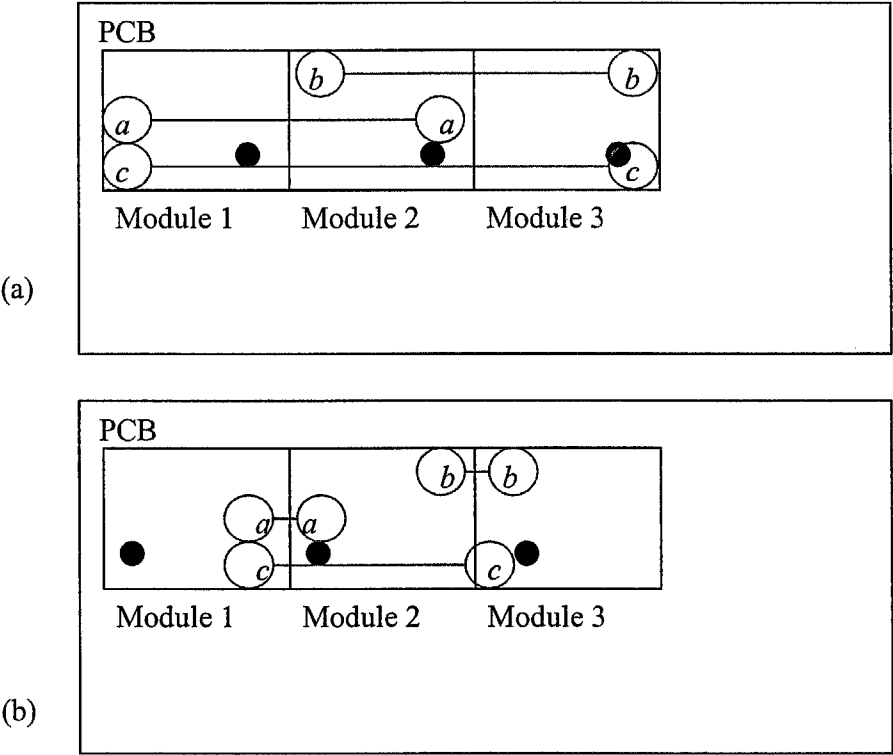


Figure 2.12: Flipping the modules

Min-cut graph model

Min-cut graph model as defined by Cheng *et al.* [13] is but one of the many solutions employed to solve the Module Orientation Problem [26, 29, 30, 31, 32, 33]. Min-cut graph model fashions each of the modules as nodes of a graph and the change in circuit net length as the edge capacities of the graph [13]. It will construct a graph $G(V, E)$ with a capacity c_{ij} associated with each edge (i, j) in E . The node set V contains $n+1$ nodes: a supernode T represents those modules not to be flipped, while n nodes represent the n modules which may be flipped. A minimum cut will be used to partition the graph into two sub-graphs. One contains k nodes which are the k modules to be flipped while the other contains $n-k+1$ nodes which represents all the modules not to be flipped. The min-cut graph model states that the capacity of the cut is equivalent to the change in the net

length of the PCB should the k nodes be flipped. Since this cut is a minimum cut hence this change in net length is minimised.

Consider the example given by Cheng *et al.* [13] as shown in Figure 2.13. The modules are numbered 1, 2, ..., 5 while the pins of the modules where the net connects each of the modules together are labelled a, b, ..., e. The pin of a module is on the module itself, near the edge of the module while the net consists of wiring linking the pins together (as shown by the lines joining the pins together).

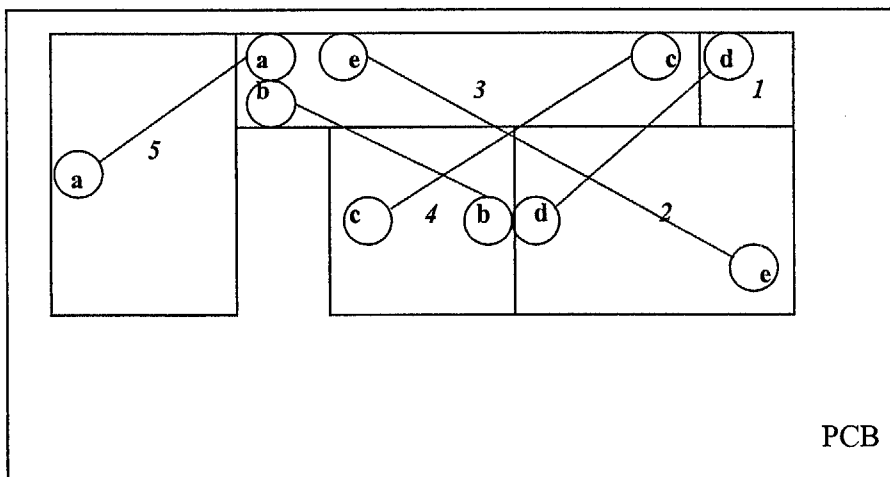


Figure 2.13: A five-module example (from Cheng *et al.* [13])

In order to construct a min-cut graph model for the given example, the edge capacities have to be found first. This can be done by considering one net at a time. Consider the net at pin e that connects module 2 and module 3 together. For easy reference the net is named net e as well to denote that it connects the modules together using pin e. The change in the length of net e can be calculated if one or both of the modules are flipped. Let

C1 be the additional length when only module 2 is flipped,

C2 be the additional length when only module 3 is flipped,

C3 be the additional length when both modules 2 and 3 are flipped.

A triangle graph will be devised (see Figure 2.14) containing three nodes, T , 2 and 3. Since we are only flipping module 2 and module 3, the rest of the modules are not flipped and are represented by supernode T . Nodes 2 and 3 represent modules 2 and 3 respectively. The triangle graph thus represents all the modules on the PCB.

For this triangle graph, four different cuts are possible. The first cut where all nodes will reside on one side of the cut, represents the non-flipping case. As proven by Cheng *et al.* [13], the other three cuts will represent the above-mentioned three different flipping cases. The edge capacities of the triangle graph are formulated to reflect the flipping cost C_1 , C_2 and C_3 as shown (see Figure 2.14, taken from Cheng *et al.* [13]):

$$c_{2T} + c_{23} = C_1$$

$$c_{3T} + c_{23} = C_2$$

$$c_{2T} + c_{3T} = C_3$$

(2.3)

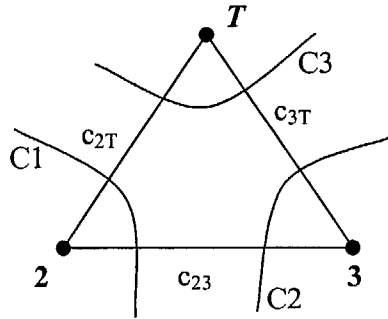


Figure 2.14: Triangle graph for module 2 and module 3 (from Cheng *et al.* [13])

Values of C_1 , C_2 and C_3 may be obtained by flipping the various modules and obtaining the change in total net length on the PCB. For example, when only module 2 is flipped, the overall change in net length is -5, hence C_1 will be -5. Let the values of C_1 be -5, C_2 be 2 and C_3 be -1 in this case. By substituting these values into Equation (2.3), values for c_{2T} , c_{3T} and c_{23} can be obtained by solving the three simultaneous equations and are given as:

$$c_{2T} = -4$$

$$c_{3T} = 3$$

$$c_{23} = -1$$

Using the same method, a triangle graph is formed for each of the nets. Four other triangle graphs are obtained for nets a , b , c and d . The edge capacities for the rest of the edges may be found. The min-cut graph model can thus be formed by using the graph construction algorithm given by Cheng *et al.* [13]. The answer given by Cheng *et al.* [13] is reproduced and shown in Figure 2.15.

A minimum cut, $C_{\min}$, is applied across the graph to partition the graph into two sub-graphs: $\{3, T\}$ and $\{1, 2, 4, 5\}$. This implies that modules 1, 2, 4 and 5 are to be flipped while the orientation of module 3 remains. Since capacity of $C_{\min}$ is -11, according to the min-cut graph model [13], the change of total net length would be equivalent to the capacity of $C_{\min}$ which is -11, denoting a decrease in net length by 11 units.

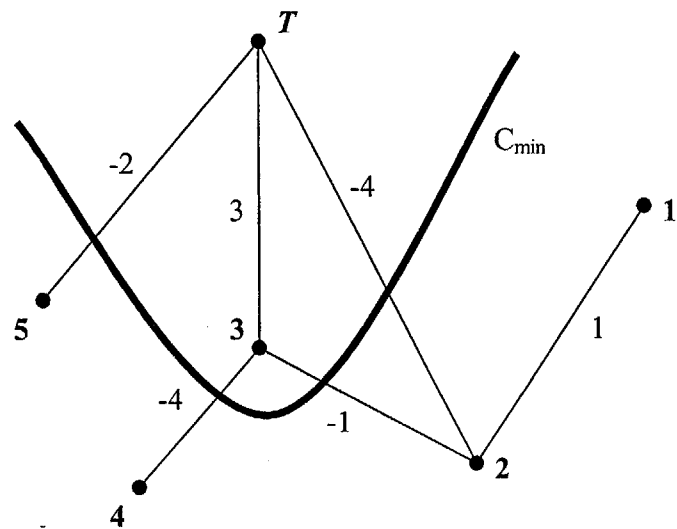


Figure 2.15: Min-cut graph model of the five-module example (from Cheng *et al.* [13])

2.2.4 Traffic assignment

In this section, the problem of traffic assignment is outlined. However, terms like O-D matrix, cost-flow functions, vehicle arrival distribution and measurement-of-effectiveness are discussed beforehand in order to make understanding of traffic assignment easier.

2.2.4.1 O-D matrix

O-D matrix as defined by Ortúzar and Willumsen [1], refers to the origin and destination matrix. Origin in this context refers to the location where vehicles originate. This can be seen as the location where the vehicle leaves its parking place to start its journey. Destination is the location where vehicles would end up. A table is formed showing the number of vehicles proceeding from the various origins going to their various destinations. This table is an O-D matrix.

Table 2.1 shows an example of an O-D matrix for 3 locations A, B and C. Looking across the first row (which is for location A), the O-D matrix shows that for every hourly period there are:

- i. 0 vehicles going to location A from location A.
- ii. 10 vehicles going to location B from location A.
- iii. 60 vehicles going to location C from location A.

The same interpretation can be made for the second and third rows, to represent movements of vehicles between these three locations. In some cases the term “zone” is used in place of “location”. When used in an O-D matrix both terms have the same meaning except perhaps a zone could refer to a bigger area of influence for example, from west zone to the east zone.

Table 2.1: O-D matrix

Locations	A	B	C
A	0 veh/h	10 veh/h	60 veh/h
B	50 veh/h	0 veh/h	25 veh/h
C	10 veh/h	30 veh/h	0 veh/h

2.2.4.2 Cost-flow function

Given a stretch of road as it gets more congested, the time taken for each vehicle to traverse the stretch of road may increase exponentially. This congestion continues until the point when the capacity of the road is reached and it can no longer allow any more vehicle to move into it. The travel time taken at this point would notionally be infinity [8].

Congestion in this case is measured in terms of the flow of vehicles, defined as the number of vehicles per hour. As stated by Ortúzar and Willumsen [1], as the time taken to make a journey is sometimes perceived by drivers as cost of the journey, models describing the relationship between congestion of roads and vehicle travel times are known as cost-flow functions [34, 35, 36, 37].

Ortúzar and Willumsen [1] gives a generalised cost-flow function as:

$$C(x) = a\alpha + b\beta + c \tag{2.4}$$

where

- x the route/road used,
- $C(x)$ the cost for using route/road x ,
- α distance of x ,
- β flow of vehicle in x ,
- a, b, c constants.

As shown in Equation (2.4), since distance of the route x , α , does play a part in the cost for using x hence, one of the ways to lower the value of $C(x)$ would be to reduce α . One possible way of doing that is to reduce the distance required for travelling.

Depending on the traffic assignment model (see Section 2.2.4.5) used, the cost-flow function may be used to find the cost of the road network as a system or the cost of traversing the road network to the drivers.

2.2.4.3 Vehicle arrival distribution

For a congested morning traffic peak hour, according to Huber [38], the frequency distribution for the number of vehicles arriving per 15-second interval follows a Generalised Poisson distribution [39]. This is so because the mean/variance ratio of the frequency distribution for vehicle arrival is greater than 1.0.

Erlang distribution [39] is the interval distribution that follows directly from Generalised Poisson distribution. It would give the time between vehicle arrivals instead of number of vehicles arriving in a given time interval. This thus makes Erlang distribution the continuous counterpart of the discrete Generalised Poisson distribution. Hence, if a continuous distribution is used for arrival rate of traffic for the morning peak hour traffic instead of a discrete distribution, the Erlang distribution would be used in place of the Generalised Poisson distribution

2.2.4.4 Measurement-of-effectiveness (MOE)

Measurement-of-effectiveness (MOE) [40] is a way to assess road networks. As road networks may be made up of more than one type of traffic facilities (examples of traffic facilities are pedestrian crossings, traffic signals, signalised traffic junctions, unsignalised traffic junctions, “Stop” signs, “Give way” signs and so on) each traffic facility is given a parameter to measure its effectiveness. For example, overall delay is used to determine the effectiveness of unsignalised traffic junctions.

For the parameters chosen (for example, delay) there are prescribed ranges of values; actual values of the parameters falling between certain intervals of these ranges may be designated a standard value or notation to indicate how effective that traffic facility is. These standard values are the MOEs of the particular traffic facilities. For example should delay fall between 0 second and 10 seconds, then the MOE of the unsignalised traffic junction takes the value of A (the best).

2.2.4.5 Traffic assignment

According to Ortúzar and Willumsen [1], once a transport engineer has finished the design of the road network of a new estate, he would have to assess the road network in order to ensure that various workings (for example the congestion level) of the road network are acceptable. As it is too impractical and costly to test the newly designed road network with real live traffic after it has been built, one of the ways this testing can be done is by means of a traffic assignment model (which can be in the form of a computer simulation).

First, the transport engineer would need traffic data in the form of an O-D matrix. He then needs to build the road network in the traffic assignment/simulation model. With that, the vehicles can now be loaded into the road network with an arrival rate that follows an acceptable distribution. Traffic assignment then takes care of the routing problem of the vehicle.

In traffic assignment, the effect of congestion, in the form of the number of vehicles present (or vehicle flow), is modelled with the aid of cost-flow functions [34, 35, 36, 37]. Depending on the traffic assignment algorithm used, different cost-flow functions are used. However these cost-flow functions can be generalised into the form given in Equation (2.4). Cost-flow functions are factored into the network to decide which route has the least delay.

With the O-D matrix given, a portion of the traffic volume (this amount depends on the traffic assignment algorithm used) will be allocated to various routes of the network. Using cost-flow functions, the cost of using various edges/links of the network is calculated. Using tree building, a MST (based on the cost of the edges of the network) is obtained for each possible source of the network. Vehicles for a given source will then take only the routes on the MST obtained. The number of vehicles using a specific route is decided by the traffic assignment algorithm used. Since all routes used are derived from the MST, they are the routes with the least cost (in terms of delay) at that point of time.

After a while, as congestion starts building up (due to large numbers of vehicles entering the network) this route will no longer have the least delay. Delay is once again calculated using cost-flow function; a new MST is obtained and a new route with the least delay is determined. Some of the vehicles will move away from the congested route to this new route. Like before, the traffic assignment algorithm will decide the number of vehicles that will move away from the congested routes.

The whole process is repeated and this will take several iterations until no drivers can reduce his travel cost any further by changing route. At this stage, the whole road network is said to have reached Wardrop's equilibrium, as stated by Wardrop [41]. The routes that various vehicles take are thus decided. In addition to that, by knowing the number of vehicles on each route, the traffic assignment model used will be able to identify the routes in which congestion is likely to occur.

With the vehicles' routes decided, the road network can be assessed for its MOE. Using the MOE, various alternate road networks can be compared in an attempt to find the better road network to be built.

In order to accurately assess a road network, it is important to do so on a typical day and not on days where special events such as New Year Eve's celebrations are being held.

In order to get the typical number of vehicles using each route in the road network on a typical day, one has to examine how a driver chooses his routes to his destination daily. A driver going to work first considers the distance needed for various routes (this constitutes the distance component in the cost-flow function). Then based on his experience in using the network, he considers the various routes based on the number of vehicles or level of congestion expected on each of the routes at the time he is going to travel (this constitutes the congestion effect component in the cost-flow function). The driver then selects the route A, which he feels will be the fastest (this corresponds to selecting the routes from the MST obtained from the tree building process). However

when the driver sets out to travel to his destination, he may find many other drivers selecting route A as well. Route A may not be the fastest after all.

The next day, the driver may reconsider his route again based on what happened the day before (this corresponds to the re-iteration process of the traffic assignment). This time round he may choose route B instead (this corresponds to the re-routing process of traffic assignment). Since other drivers may have chosen route B as well, route B may not be as fast as the driver first expects it to be. On the other hand, due to the group of drivers moving away from route A to route B, route A may be less congested now.

This whole process will then repeat itself again until eventually all drivers are satisfied with the route they have chosen and will continue using the same route every other day. At this point the number of vehicles using the network is said to be typical (that is the network has reached Wardrop's Equilibrium). This traffic volume can then be used accurately to assess the road network.

2.2.5 Traffic simulation

In this section, pertinent mathematical models from [40, 42, 43, 44, 45, 46, 47, 48] which are used in traffic simulation are briefly discussed. Following that, three levels of traffic simulation as described by Algiers [49] and Banks *et al.* [50] will be discussed. Lastly a time advancement procedure of traffic simulation as discussed in Highway Capacity Manual [40] and Banks *et al.* [50] will be briefly covered.

2.2.5.1 Driver behaviour

Various traffic simulation techniques by Smith *et al.* [12], Wang and Prevedouros [51] and Barcelo [52] use various driver behaviour theories to simulate the movement of vehicles in the simulation. These theories include:

- i. Car-following theory, as discussed by Andersen *et al.* [43] and Herman *et al.* [44]

- ii. Gap acceptance theory, as described by Highway Capacity Manual [40] and Hughes [45]
- iii. Lane-changing theory, as described by Highway Capacity Manual [40] and McShane and Roess [48]

Car-following theory

According to Andersen *et al.* [43] and Herman *et al.* [44], under certain circumstances, drivers are said to be following the vehicles being driven in front of them. The mathematical model that is fashioned from this theory is known as car-following theory [43, 44].

Gap acceptance theory

For a car making a right turn at a junction, the driver would have to find a suitable “space” in the face of opposing traffic. This “space” is referred to as a gap by Highway Capacity Manual [40]. The mathematical model that describes vehicles selecting suitable gaps between traffic streams to make a safe manoeuvre is known as gap acceptance theory [45].

Lane-changing theory

In various situations, drivers do switch lanes for different reasons like to make a turn or to overtake slow moving vehicles. The mathematical model that covers various driving situations that affect a driver’s lane selection decision is known as lane-changing theory [48].

2.2.5.2 Traffic flow

Movement of vehicles in a given road network has been described using flow theories by Kockelman [47] and Bloomberg and Dale [53]. As such several mathematical models describing traffic as a flow have been developed. These include:

- i. Traffic stream model by Highway Capacity Manual [40] and McShane and Roess [48]
- ii. Shock wave model by Abbas and Bullock [42]

Traffic stream model

According to McShane and Roess [48], vehicles are grouped together and seen as a platoon. Such platoons are regarded as streams of flow, named traffic streams. Mathematical models that analyse the behaviours, interactions and movements of a traffic stream are known as traffic stream models [40, 48].

Shock wave model

Back spill of traffic can occur due to queuing at junctions; this is referred to as traffic flow propagation. As stated by Abbas and Bullock [42], the mathematical model that describes this phenomenon is known as the shock wave model.

2.2.5.3 Traffic simulation

As described by Algers [49] and Banks *et al.* [50], traffic simulation can be carried out at three levels:

- i. Macroscopic
- ii. Microscopic
- iii. Mesoscopic

Macroscopic

Macroscopic traffic simulations as described in Banks *et al.* [50], Bloomberg and Dale [53], Bell [54] and Helbing and Treiber [55] are based on deterministic relationships between roads, intersection characteristics and traffic flow. Vehicles are simulated as one whole platoon using flow analogies as described by Kockelman [47] and Dale [53] and assumptions described in Bell [54] are used for details such as drivers' behaviours and vehicle types.

Microscopic

As stated by Cameron *et al.* [56], Cameron and Duncan [57] and Cetin *et al.* [58], by simulating interactions of individual vehicles, microscopic traffic simulation attempts to

represent the dynamic effects of traffic. It models the behaviours of drivers by use of driver behaviour theories like car-following theory as described by Highway Capacity Manual [40], gap acceptance theory as described by Hughes [45] and lane-changing theory as discussed in Karmeier [46].

Mesoscopic

According to McShane and Roess [48] and Pursula [59], mesoscopic is the hybrid of macroscopic and microscopic traffic simulations. It normally describes vehicles individually, a much detailed level than macroscopic simulation. On the other hand, activities and interactions between vehicles are described by road capacities, a less detailed level than microscopic.

2.2.5.4 Fixed time-interval scanning approach

Fixed time-interval scanning approach as described by Banks *et al.* [50] is a time advancement procedure that a traffic simulation can have. A traffic simulation using fixed time-interval scanning approach will advance the simulation time at a predefined amount. Any event that occurs within this prefixed period will be reflected in the simulation as the simulation time progresses.

2.2.6 One-way/two-way road network

According to Karagheuwfi [60] when a two-way road network is compared to a one-way road network, the two-way road network would be a faster network when traffic volume is low. This is due to the extra distances required in the one-way road network (see Figure 2.16).

Both Figures 2.16(a) and (b) show the same road network, where roads are two-way in Figure 2.16(a) and roads are one-way in Figure 2.16(b). For a vehicle to get from point 2 to point 1 in Figure 2.16(a) it could take route A. In Figure 2.16(b) however, the vehicle would have to take the longer route B.

On the other hand, when traffic volume is high the one-way road network would be a faster network. This is because the advantage of shorter distance is now removed in the two-way road due to the congestion effect caused by higher traffic volume. The longer distance in the one-way road would become an advantage since it would mean a larger capacity to hold traffic when traffic volume becomes higher.

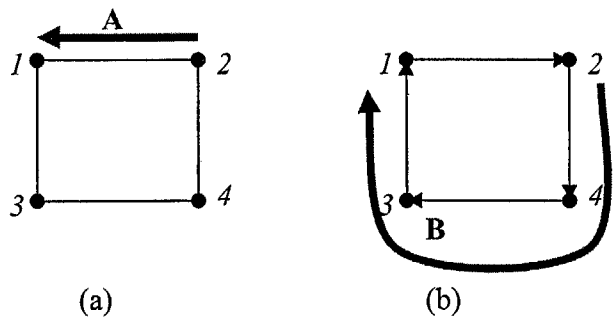


Figure 2.16: Increased distance in one-way road

2.3 Review of road network design

This section begins by reviewing the planning process of road network design before building the actual road network. It also touches on the planning process when there is a need for a change in the road network. The uses of simulation in these processes are brought up. This is followed by a discussion on the conversion to one-way roads from two-way roads.

2.3.1 Road network design methodology

Based on literature reviewed on the building of new road networks, plans (inclusive of layout of roads) are drawn with respect to:

- i. Buildings to be built, as stated by Karagheuwfi [60], Abdulhai [61] and Owen *et al.* [69].
- ii. Existing traffic facilities surrounding the road network, as stated by Raney *et al.* [2], Cameron and Duncan [57], Abdulhai [61] and Owen *et al.* [69].
- iii. Volumes of vehicles expected as discussed by Karagheuwfi [60], Krause [67] and Taplin [70].

- iv. Local economy as discussed by Iravani [65].
- v. Safety as discussed by Nobel and Jenks [68].

With each proposed layout of the roads, a simulation of the road network is conducted. Based on expected volumes of vehicles, vehicles are loaded into the simulation model before running the simulation. According to Ortúzar and Willumsen [1] and Highway Capacity Manual [40], as long as the road network is performing within a certain predefined level of acceptance or measurement of effectiveness, the road network is considered effective and will be built.

In cases where the existing road networks are not performing to satisfaction, for example congestions that are too heavy or have an unacceptably high number of accident occurrences, the networks are re-evaluated. Changes to the networks planned towards resolving the existing problems are then decided upon. Such decisions could be based on:

- i. social demands such as road users requesting for traffic signals at junctions as described by Janssen [72] and Taylor and Willoughby [75]
- ii. logical reasoning such as addition of extra lanes to cater for congestion as described by McShane and Roess [48] and Nagel [74]
- iii. proven solutions from other similar cases as discussed by Brilion and Blanke [7] and Kanazaki *et al.* [73]
- iv. elaborate long-term development planning as stated in Central Western Precinct [71]

Just as with new road network design, simulation is again used. Generally, two test cases will be set up, first of the original road network, followed by the newly proposed road network. Results of the simulation are then analysed. As long as the new network is performing better than the original road network (based on MOE for the road network), the new network would be considered effective. The actual road network would then be changed to the new network's design.

2.3.2 Conversion of two-way roads to one-way roads

Literatures abound on the various aspects of one-way streets; some discuss the effects of one-way streets (such as the advantages as well as the disadvantages). Some study the various transport facilities that go well with one-way streets, such as shared streets; others compare a road network before and after conversion from two-way streets to one-way streets. In this section, literature on one-way streets are reviewed and discussed in an attempt to account for the niche which this project attempts to fill up. Various literatures are grouped and discussed in accordance to the topic of their interest; the three groups thus obtained are as follows:

- i) Conversion of two-way roads to one-way roads for comparison
- ii) Shared streets
- iii) Conversion of two-way roads to one-way roads to increase safety and reduce congestion

2.3.2.1 Conversion of two-way roads to one-way roads for comparison

One-way street/road was first discussed in 1972 by Karaghewfi [60]. In his paper, Karaghewfi emphasised on comparing some of the effects of converting two-way streets to one-way streets; he also offered some possible reasons to explain why one-way streets are better than two-way streets.

According to Karaghewfi, there is an overall reduction in both vehicular and pedestrians-related accidents. The reason being, in the case cited, after the conversion, the order of approaching traffic (from different directions) at an intersection is reduced from 4 to 2 (for a 4-legged intersection). The number of conflict points for vehicles is reduced as well; this reduces the chance of accidents thereby resulting in a reduction in the number of accidents.

Karaghewfi also reported that due to the reduced number of approaching traffic directions, there is less delay at those intersections. This means less delay in the overall trip time. In addition, results of studies show that the number of stops drops by 66% after converting to one-way streets. This makes one-way streets a better choice since drivers are more accepting of slow but steady movement but less accepting (that is feel

impatient) with regards to more stop-and-go movements even if movements are at a higher speed. More stop-and-go movements are also environmentally unfriendly due to higher fuel consumption and greater engine wear.

In Karaghewfi's studies, the mechanics for the conversion of the two-way streets to one-way streets is absent; instead elaborate long-term town planning is cited. However, there is no detail on how to go about the conversion of two-way streets to one-way streets nor are there explicit explanations for the chosen directions for each of the one-way streets.

In 1982, Karmeier [46] also did a comparison of one-way streets to two-way streets. In it he highlighted the advantages of one-way streets over two-way streets, these advantages being safety, capacity and convenience. According to Karmeier, they are mostly the results of reduced number of opposing traffic flows at the intersection:

- With reduced incidence of opposing traffic, conflict points at intersections are reduced hence an increase in safety of one-way streets
- With removal of opposing traffic flow, vehicles can now occupy the lane previously used for the opposing traffic hence an increase in capacity in one-way streets
- With reduced occurrences of opposing traffic, there is also less delay at an intersection. This reduces the frequency of stop-and-go experienced by drivers and hence an increase in average speed

As Karmeier's emphasis is on the advantages of a one-way street over a two-way street, he did not discuss how the two-way streets are converted into one-way streets, but only briefly mentioned that the choice of direction of the one-way streets are based on expected volume of vehicles. No explicit explanation on the selection of the directions of the one-way streets is given.

After Karmeier, Stemley [10] also conducted another intensive study in comparing two-way streets and one-way streets. However, Stemley included not only the advantages of

using one-way streets over two-way streets, he discussed the disadvantages of one-way streets over two-way streets as well.

In his findings, the disadvantages include:

- Confusion by drivers in figuring out the one-way road pattern
- Extra travelling distance and hence the extra fuel consumed by vehicles, although this is debatable
- Longer walking distance for transit passengers to get to their destinations from transit stops or to get to the return service
- A higher chance of emergency vehicles being blocked at an the intersection

In his findings, the advantages include:

- Continuous movement for traffic, that is smoother traffic flow hence less stop-and-go movements for vehicles
- Fewer conflict points at intersection hence less accidents
- There will be less traffic as traffic comes now only from one direction instead of two, which is better for the elderly who has slower movements and reaction times
- Increase in capacity
- Less travel time due to reduced number of stoppages and hence increase in accessibility

Stemley emphasised on uncovering the advantages and disadvantages of one-way streets (over two-way streets) and giving explanations for his findings. He then related his findings to the positive implications the one-way streets have on the economy of the city. In Stemley's report, the reason for the conversion to one-way streets are economically driven; however there is no explicit explanation or discussion on how the two-way streets are converted to one-way streets as well as the choice of directions for each of the one-way streets.

The most recent study done on one-way streets is by Meredith and Prem [76] in 2001. The emphasis of their research is on comparing the efficiency, convenience and safety level of traffic operations between one-way and two-way streets in business districts.

They did so by looking at various intersections: one-way intersection, two-way intersections or intersections of both one-way and two-way streets.

Meredith and Prem studied 33 intersections in all; from there, baseline conditions for various intersections are established. These baseline conditions were then used to develop 8 different traffic scenarios of one-way and two-way streets. The 8 scenarios were then tested using a traffic simulation model developed for this study.

Findings of the model indicated, significantly that a two-way street network does not result in shorter distance travelled as compared to a one-way street network. However, converting one-way streets to two-way streets in the business district area does increase the time required to travel through the business district due to the increased number of stops (made by the vehicles) caused by congestion of various intersections.

In this case, the decisions on conversion of two-way streets to one-way streets are based on studying past usage data at similar intersections; with that 8 different networks are built for testing. However, Meredith and Prem did not look into how a new one-way network ought to be built should there exist no past data for it to base on nor any indication on its similarity to other intersections/networks as yet.

2.3.2.2 Shared Streets

Shared street was discussed in 1981 by Appleyard [4] who examined the conversion of one-way streets to shared streets. Shared streets are basically one-way streets that integrate pedestrian activity and vehicular movement on one shared surface (normally attained by the removal of the raised pavements/ curbs/ sidewalks).

Appleyard discussed the use of shared streets in the business districts of cities and recognised the use of the one-way streets, in shared street design, as a means to tame the detrimental effects of traffic in the business districts. The reason for conversion in Appleyard's research is given as based on social demand. Nevertheless he did not discuss how the one-way streets are converted from two-way streets in the first place (before the

one-way streets are changed to become shared streets) nor provide any detail on the choice of the directions of the various one-way streets.

Shared street is further reviewed by Ben-Joseph [3] who prescribed the one-way street as an indispensable component of a shared street. In his research, Ben-Joseph emphasised on various features that characterised a one-way street as a shared street; according to him, it is the psychological effect that a driver feel like an intruder in a shared street that helps reduce congestion and increase safety along shared streets. He based his findings on traffic data of reported cases of shared streets in Europe [7, 67, 68, 77], Japan [73] and Israel [78] where the number of accidents declined by more than 20% and the number of severe accidents by more than 50%, compared with traditional streets.

Unlike Appleyard (who concentrated on shared streets in the business districts), Ben-Joseph concentrated on shared streets in residential areas; however like Appleyard, his highlight is on studying the effects of shared streets. Ben-Joseph did not discuss how the one-way streets (in the cases he quoted) are converted from two-way streets in the first place nor did he give explicit reasons for the choice of directions of the one-way streets; the reason for the conversion is given as for the safety of road users.

2.3.2.3 Conversion of two-way roads to one-way roads to increase safety and reduce congestion

According to Veeraragavan and Robert [79], rapid urbanisation and growth in Bangalore City has led to an increase in the number of road accidents as well as the degree of congestion. Two-way roads were thus converted to one-way roads. Veeraragavan and Robert found that after the conversion both the number of accidents and overall delay are reduced. Even from an economic point of view, as explained by Veerargavan and Robert, one-way roads are effective in reducing travel costs around the city as well as reducing accident costs.

Veeraragavan and Robert emphasised on converting two-way roads to one-way roads as a means to reduce road accidents and congestion. The reason for this is to keep up with the

growth of the city. However Veeraragavan and Robert did not discuss on how exactly the directions of the one-way roads are decided upon. Instead they merely gave the reason for the conversion and the primary aim of the resultant one-way road network which is to reduce road accidents and congestions.

2.4 One-way urban road network problem

Consider a road network consisting of various road intersections at irregular intervals. To change all roads within the road network to one-way roads one would encounter the problem of deciding the directions for each and every of the one-way roads. Choice of direction of each of the one-way roads can be made for a variety of reasons. Nevertheless in the literature reviewed, none seems to address that explicitly (see Section 2.3.2). There are no fixed parameters to be used and all cases discussed do not seem to be consistent with each other. The new one-way road networks are generally compared against the original road network using a given set of criteria (which differs in the cases discussed). The new one-way road networks are commended so long as the results are better than the original road network. This only gives a better result than the previous road network; it might not be the best solution yet. In fact, there is no theory for what constitutes a best solution.

Hence, a model for deciding the directions of the one-way road network is to be developed in this thesis. This model should not only give a more efficient and consistent way of doing things but also provides a way to ensure accuracy and correctness of results. The model will prove extremely useful when the road network gets bigger and more complicated. While results from a model should hardly be the only consideration for investment decisions such as road building, these results may however be used as a foundation for further planning.

Furthermore, the model can be extended to other types of networks, especially to those in which flow is already in a single direction and these directions had been decided by the network planners. The model will then be able to help optimise such networks even further by taking into consideration a physical attribute such as direction. With that, the

one-way urban road network problem is defined as: deciding the directions of all one-way urban roads when converting the roads from two-way to one-way in a given urban road network such that the resultant one-way urban road network would be the most efficient, by some measure to be defined. The objective of this thesis is therefore to formulate a model to handle this conversion of two-way urban roads to one-way urban roads.

The word “urban” is added to describe the roads discussed in the one-way road network. According to Highway Capacity Manual [40], urban roads refer to roads with relatively high density of vehicle flow and junctions no further than 3.0 km apart. This is done to distinguish the roads from freeway roads (roads that form expressways) and rural roads (long stretches of roads with junctions further than 3.0 km apart). The terms “roads” and “urban roads” will be used interchangeably throughout the rest of this thesis unless otherwise qualified.

With the problem identified and the objective defined, the next step would be to devise a way to formulate and validate a model for the conversion. These will be discussed in the next chapter, Chapter 3: Methodology.

III METHODOLOGY

3.1 Overview

This chapter will first discuss the approach to unravel the one-way urban road network problem. A two-pronged approach consisting of a theoretical investigation and a validation process will be used. These two sub-approaches are discussed in detail. The theoretical investigation will examine the ways to formulate the model. The validation process will suggest using simulation to validate the model as well as to verify the efficiency of the network. The chapter ends by discussing the Woodsville Interchange Junction (WIJ) road network as a choice for case study.

3.2 Two-pronged approach

In order to create a functional model (to handle the conversion of two-way to one-way roads), various aspects of the one-way urban road problem have to be explored. This includes (and is not limited to):

i. Transportation: to find ways to assess the road network

With the various ways to assess the road network defined, the model can then be modelled in such a way that it is able to identify the area(s) of the road network that it should optimise.

ii. Traffic assignment, in deciding the routing of vehicles

This allows the model to utilise a road network such that vehicles will be routed in a manner such that the parameter(s) identified in part (i) can be optimised.

iii. Graph model, in representation of the road network and tree building in traffic assignment

Part of traffic assignment consists of tree building that uses graph theory. Hence a graph model is employed as the primary tool to help create a road network such that

vehicles can be routed in such a way that the parameter(s) identified in part (i) can be optimised.

iv. Assessing the traffic direction of roads with respect to the whole road network

Finding a methodology to more easily and efficiently assess the direction of traffic flow of roads so as to give the required optimised one-way road network which is a major objective of this study.

A theoretical investigation is thus needed to explore the nature of the problem so as to formulate the model.

There is also a need to validate the model after it has been formulated and applied. This can be done by testing the resultant one-way road network (obtained from the model) against its original two-way counterpart. Comparisons based on defined criteria can then be made between the new one-way road network and the original two-way road network.

A validation process is therefore needed to assess the effectiveness of the model. Hence as discussed above, a two-pronged approach is required for the one-way road network problem. First, a theoretical investigation is used to formulate the model. Secondly, a validation process is used to verify the model.

3.3 Theoretical investigation

In this section, various theoretical aspects of the one-way road network problem are examined. This would provide insight to the model's general components, thereby leading to the formulation of the model.

3.3.1 Requirements of the model

In order to formulate the model, its requirements are examined first. Based on the definition of the one-way urban road network problem, both traffic flow directions (as well as the combination of flow directions) of all roads in the network have to be considered in order to systematically arrive at the most efficient one-way road network to quantify efficiency. The model would need:

- i. Either a set of parameters to quantify efficiency, or a standard method to assess the road network's efficiency. This would allow quantitative comparison between each of the one-way road networks. A more efficient one-way road network can then be found systematically.
- ii. An effective methodology to determine the road direction for each of the road in the network to give a resultant, functional one-way road network that is the most efficient, based on part (i).

3.3.2 Assessing the road network

According to Ortúzar and Willumsen [1], for an urban road network (where intersections are less than 3.0 km apart), it would be inaccurate to assess the road network based on the measurement-of-effectiveness (MOE) [40] of the roads. This is because the roads are not long enough for the propagating effects of congestion/delay from the intersections to subside.

Based on the Highway Capacity Manual (HCM) [40], for the above-mentioned case, it would be more accurate to assess such a road network at its various intersections. As stated in the scope of this thesis (see Section 1.3), only unsignalised intersections are considered for the purposes of this study. In order to assess any urban road network with unsignalised intersections, delay at these intersections is commonly used as the MOE. From an engineer's point of view, delay is therefore an important parameter which can be used to assess the urban road network in this case. (It is noted that should a different road network with different traffic facilities be used, the parameters for assessing the road network may differ.)

According to Ortúzar and Willumsen [1], the cost-flow function (see Section 2.2.4.2) is generally a reflection of the road-user assessment of the road network in question. The generalised form of the cost-flow function [see Equation (2.4)] is made up of distance and time (delay). This means generally that a road user will consider the use of a road network by first looking at the distances of the various available routes. Then based on experience with the network, the delay incurred (due to congestion) and the time required to get to the destination will be considered. For any chosen route, the distance is fixed and cannot be changed. Time (delay) on the other hand is dynamic and changes with respect to interaction with other road users; for example, should there be a heavy traffic jam a vehicle may take up to half an hour to move a distance of 300m. Hence in considering the routes to take, from a driver's point of view, "travelling time" (including all delays) will be used over "distance" as the criterion to assess the effectiveness of route/network.

3.3.3 Deciding one-way road directions

A similar problem to the one-way urban road problem is the Module Orientation Problem [26]. The Module Orientation Problem (as explained in Section 2.2.3.2) is the search for the optimal flip of all the modules to give a minimum net length for the PCB. The minimum-cut (or min-cut) graph model by Cheng *et al.* [13] for the Module Orientation Problem solves the latter by modelling the modules of the given PCB as a graph. Using a minimum cut, it then decides which modules to flip (see Section 2.2.3.2).

The flip of a module refers to an actual physical change in the direction that the module is facing. This physical flip of the module is analogous to the change in direction of a one-way road. While the physical flip of the module would affect the net length for the PCB, the change in direction of the one-way road would affect the net delay of the whole network. Both deal with change in an overall parameter of a system when directions of system components are changed.

Using parameter mapping:

- i) The one-way roads may be mapped onto the modules.
- ii) The physical reversal of directions of the one-way roads may be mapped onto the flips of the modules.
- iii) The net delay of the one-way road network may be mapped onto the net length of the PCB.

The min-cut graph model could thus be used to obtain the best layout for the resulting one-way road network.

3.3.4 Initial layout of one-way roads

In the Module Orientation Problem, in the initial layout, the modules are assumed to be placed onto the PCB in an optimal manner before the flipping process starts. This problem of finding the optimal initial placement of the modules is known as the Module Layout Problem (see Section 2.2.3.2). This could play a vital part in the Module Orientation Problem since as shown by Cheng *et al.* [13], different initial module layouts could result in very different solutions for the Module Orientation Problem.

Similarly, an initial layout of the one-way roads should be determined before application of min-cut graph model onto the road network, as different results may be obtained with different initial road layouts. This is illustrated in the example shown in Figure 3.1.

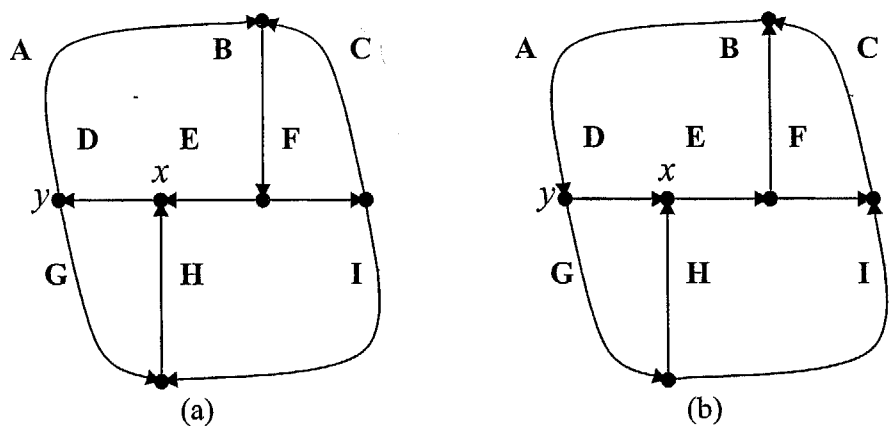


Figure 3.1: A simple strongly connected one-way road network

Both Figures 3.1(a) and (b) show two strongly connected configurations of the same road network, however the directions of the one-way roads in both road networks are different. A to H represent roads in the network while x and y represent specific road intersections used in the example.

Looking at Figure 3.1(a), should the direction of road D be reversed, all roads at intersection y would be leading away from the intersection and none towards it. This implies that no vehicle can reach y since there is no road leading to it; y becomes an isolated node. (In graph theory, the indegree of y is said to be zero.) On the other hand all roads at intersection x would be leading towards it, none away from it. No vehicle can leave x ; x also becomes isolated. (In graph theory terms the outdegree of intersection x is zero.)

Due to the isolation of nodes x and y , no vehicle will be able to get to y and no vehicles will be able to leave x . Hence, D will not be considered for “flipping” when the min-cut graph model is applied to this road network. In addition, B, F and H will also not be considered for “flipping” for Figure 3.1(a). (The terms “flip” and “flipping” when used with one-way roads would mean a reversal in direction of traffic flow of the road unless otherwise specified).

On the other hand, when D is flipped in Figure 3.1(b) both x and y do not become isolated. This means D will be considered for flipping. However x becomes isolated when E is flipped. Hence, in this case, A, C, E and G will not be considered for flipping.

As shown, with different initial road layouts, different solutions could be obtained (since the roads to be considered for flipping are not the same). Therefore, there is a need to find a suitable initial strongly connected one-way road layout. A suitable initial road layout is one where the efficiency of the network (as assessed by its MOE) can be increased in the eventual, optimised one-way road network. One that decreases the network’s efficiency will be considered unsuitable.

3.3.5 A suitable initial one-way road network layout

In order to obtain a suitable initial road layout that will heuristically play a part in increasing the network’s efficiency, the assessment process of the MOE of the eventual one-way road network is re-examined. This is provided in the flow chart shown in Figure 3.2. As seen in Figure 3.2, traffic assignment is used to decide the routes of the vehicles. With the routes decided the network will be loaded with vehicles. Once Wardrop’s equilibrium is reached, the MOE of the network is assessed. The traffic assignment procedure consists of two processes: cost-flow function computation and tree building algorithm.

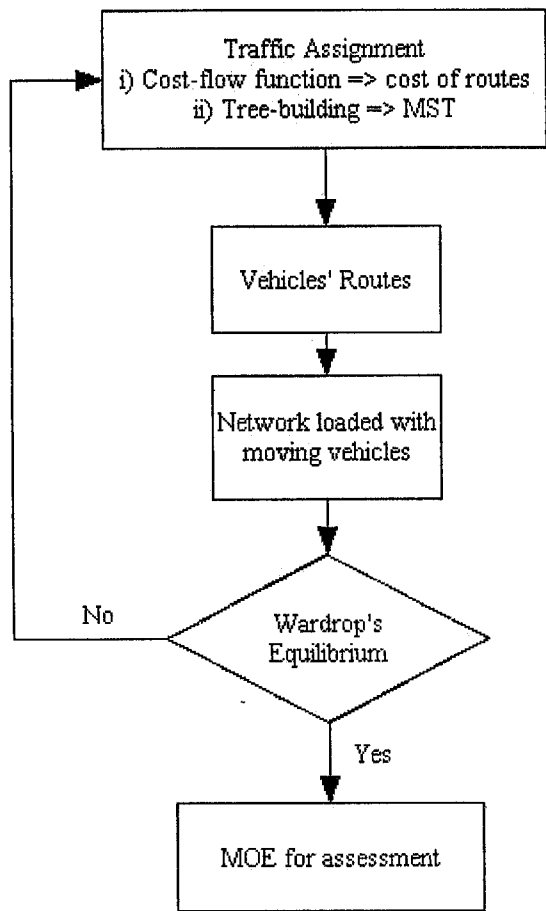


Figure 3.2: Obtaining MOE for assessment of the network

Cost-flow functions (see Section 2.2.4.2) take into consideration the traffic flow on each road (or edge when the road network is modelled as a graph) in calculating “travel cost” (or cost) for traversing each road/edge. With the cost of each road as the weight of the edge, a tree building algorithm then builds the MST for every given source (traffic origin) of the network. As the routes on the MST are such that they cost the least, all vehicles will only use the routes on their MST.

Heuristically, since MOE is dependent on the traffic assignment process, lowering the cost of the network will eventually lead to a more efficient MOE for the network. Since the initial network layout does not take traffic volume into account, we can say that it is a network which ignores the effect of congestion. Vehicles move freely along the road without being hampered by other vehicles. The cost of traversing such a network will now be confined to the time spent traversing the various roads and the intersections (without the effect of congestion). Hence, based on this heuristic, the lesser the time spent in traversing the network, the more efficient the MOE will be. A suitable initial one-way road network will be one that has the lowest total time spent by all road-users in traversing the network.

Making the assumption of universal accessibility, a one-way road network that has the lowest time spent in traversing the network is one in which the total time spent in traversing the network from all nodes (locations) to all other nodes (locations) is the shortest.

Assuming that all vehicles travel at an average speed of α km/h, the shortest time spent for traversing a one-way road network of n nodes from the first node (or node 1) to all other nodes becomes:

Shortest time spent for traversing the network with node 1 as the source

$$= [(\text{shortest path from 1 to 2}) \div \alpha + (\text{time spent at all intersections along the given path})]$$

$$[(\text{shortest path from 1 to 3}) \div \alpha + (\text{time spent at all intersections along the given path})] +$$

$$\dots +$$

$$\begin{aligned}
 &[(\text{shortest path from } 1 \text{ to } n-1) \div \alpha + (\text{time spent at all intersections along the given path})] \\
 &+ \\
 &[(\text{shortest path from } 1 \text{ to } n) \div \alpha + (\text{time spent at all intersections along the given path})] \\
 &\quad (3.1)
 \end{aligned}$$

Hence the shortest total travel time (STTT) spent in traversing the one-way road network from all nodes to all other nodes will be:

$$\begin{aligned}
 &\text{STTT} \\
 &= \text{Shortest time spent for traversing the network with node } 1 \text{ as the source} + \\
 &\text{Shortest time spent for traversing the network with node } 2 \text{ as the source} + \dots + \\
 &\text{Shortest time spent for traversing the network with node } (n-1) \text{ as the source} + \\
 &\text{Shortest time spent for traversing the network with node } n \text{ as the source} \\
 &\quad (3.2)
 \end{aligned}$$

As for any given two-way road network, there are many possible one-way strongly connected road networks, here a suitable initial one-way road network is defined as the one with the minimum STTT, that is with the minimum shortest total travel time spent in traversing the network. An algorithm is needed such that with a given two-way road network, a one-way road network with the minimum STTT (shortest total travel time spent in traversing the one-way road network from all nodes to all other nodes) for use as the initial one-way road network layout can be obtained. For easy reference, this algorithm will be referred to simply as the conversion algorithm.

3.3.6 Overview of conversion algorithm

In the tree building process, the road network is represented in the form of a graph; similarly, for the conversion algorithm, the road network will be represented by a graph. This allows the conversion algorithm to find the one-way road network with the minimum STTT.

In representing a given two-way road network as a graph, an undirected graph is used. This is because an undirected edge allows two-way “movement” within the graph. A digraph would therefore represent a one-way road network since directed edges, like one-way roads, only allow one-way “movement”.

The conversion algorithm will therefore convert any given undirected graph G into a digraph. The STTT for traversing this digraph will be the least when compared to all other possible digraphs obtainable from G and the one with the minimum STTT used as the suitable initial one-way road network. With the initial layout problem resolved, the model (for conversion of two-way road network to one-way road network) will now be defined.

3.3.7 Overview of one-way urban road network model

As the model would be used to solve the one-way urban road network problem, it will be referred to as the one-way urban road network model for easy reference.

The steps of the one-way urban road network model are as given:

1. Use the conversion algorithm to obtain a suitable initial layout of the one-way road network
2. Apply the min-cut graph model to the initial one-way road network layout to obtain a final one-way road network

The conversion algorithm will be discussed in detail in Chapters 4 and 5 while the min-cut graph model will be discussed in detail in Chapter 6.

As the one-way urban road network model has been defined, there is now a need to validate the model. A way of validating the model will now be explored. (Throughout the rest of this thesis, the term model will be used to refer to the one-way urban road network model, unless otherwise specified).

3.4 Validation process

In this section, the requirements of the validation process are studied and a way to validate the model is explored and defined.

3.4.1 Requirements of the validation process

In order to devise a way to validate the model, the requirements of the validation process have to be examined first. For the model to be effective, it would have to generate a more efficient one-way road network as a solution. (Efficiency means the one-way road network has the least overall delay in this case as explained in Section 3.3.2). Therefore, there is a need to evaluate the overall delay of the one-way road network obtained from the model. This can be done by accumulating the overall delay from the original two-way road network and comparing it with the overall delay of the new one-way road network. As mentioned before, it is impractical and costly to test the resultant one-way road network on actual road network. An analyser tool will be used to analyse and assess the overall delay of the road network instead.

3.4.2 Analyser tool

In this section, the requirements of the analyser tool are examined first. This is followed by the choice of using simulation as the analyser tool.

3.4.2.1 Examining the analyser tool

The inputs and outputs of the analyser tool will be examined first in order to provide insight into the components needed for the analyser tool. The relationship between the analyser tool, its inputs and its outputs is shown in Figure 3.3 as a first-level flowchart.

According to Brunner *et al.* [80] an analyser tool used to analyse efficiency of road network (so as to give the below mentioned outputs from the given inputs) would require the following analytical techniques:

- i. Demand (of road users) and supply (of the road network) analysis [1] for traffic assignment.
- ii. Road capacity analysis [40] for congestion effects needed for traffic assignment.
- iii. Traffic stream analysis [48] for traffic flow within the network.
- iv. Shock wave analysis [42] for flow propagation (that is back spill of traffic due to queuing).
- v. Queuing analysis [81] for traffic flow at various intersections.
- vi. Gap acceptance analysis for vehicular interactions [45] at intersections.
- vii. Car-following analysis for traffic flow [43, 44] along the roads.
- viii. Lane-changing analysis [40] for vehicle movements along the roads.

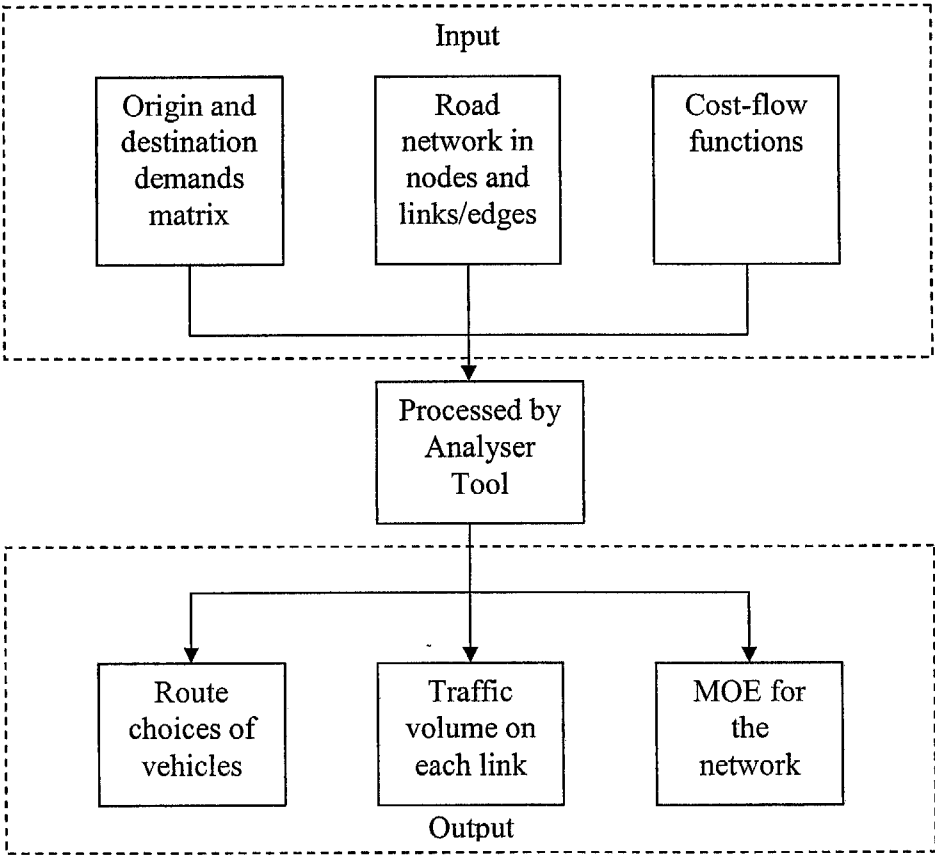


Figure 3.3: First-level flowchart for the analyser tool

In addition to the required analytical techniques the analyser tool also needs to fulfil the following:

- i. Ability to analyse the overall network and accumulate overall network delay since that is needed for assessment of the network.
- ii. Stochastic in nature to mimic the randomness of actual traffic.
- iii. A fixed time-interval scanning approach [40, 50, 75] to handle the irregularities and complexity of the random interactive traffic.
- iv. Ability to model the interaction of queuing process of vehicles especially in near congested and/or congested intersections [50].

3.4.2.2 Simulation as the analyser tool

It would be rather complex and tedious to integrate the various analyses together using an analytical approach. On the other hand, as stated by Highway Capacity Manual [40] a simulation is capable of doing just that and with less resources.

Furthermore, using random number generation, the stochastic behaviour of actual traffic and vehicular interactions can be modelled by simulation [40, 50]. Being a computer program, fixed time-interval scanning can be programmed to be the time advancement procedure of the simulation. In addition, interactive queuing of vehicles can be observed using simulation without too much difficulty [50].

Moreover, according to literature reviewed, the simulation is also capable of the following:

- i) Handle queued traffic that are not “served” by transferring them from one time period to the next; this is useful in the event of congestion in various parts of the network [12, 50].
- ii) Allow demand to be varied over time and space; this allows further “stress testing” of the road network [63, 65].

- iii) Model unusual arrival and service (at the various network intersections) patterns which do not follow traditional mathematical distributions [82].
- iv) Carry out experiments off-line without using real life trial-and-error approach. There are social and economic impacts to consider in physically changing traffic facilities to study the effects of alternatives. Simulation therefore allows such experiments to be carried out off-line [83].
- v) Provide time and space sequence information in addition to means and variances: this is rather important in the case of traffic situations since a simulation can depict traffic congestion (or the lack of it) over time and space. Whereas use of single estimators like means or variances can only provide information over a period (say 8:30am to 9:30am), exact situation at an exact time (for example congestion level at location A at exactly 8:55am) would require time and space sequence information [50].
- vi) Allow the system to be studied in real time, compressed time or expanded time. Should period of study start at 8:30am and end at 9:30am and there is not much traffic after 9:05am, simulation can accelerate time interval using compressed time to speed up the simulation run. Furthermore, simulation can be slowed down using expanded time for periods during which detailed studying is required [63, 65].
- vii) Allow the user to examine the effects of changes in the system operations, answering questions like “What if?” or “What happens when?” [63, 65].

As such computer simulation will be used as the analyser tool to analyse and assess the overall delay of the road network so as to validate the proposed model. The use of simulation to validate the model will be discussed in Chapter 7.

In addition to defining the approach for the validation process, an appropriate case study is chosen for the one-way urban road network model to be applied on. The resultant one-

way road network is then validated using computer simulation. Hence, a suitable case study will be discussed in the next section.

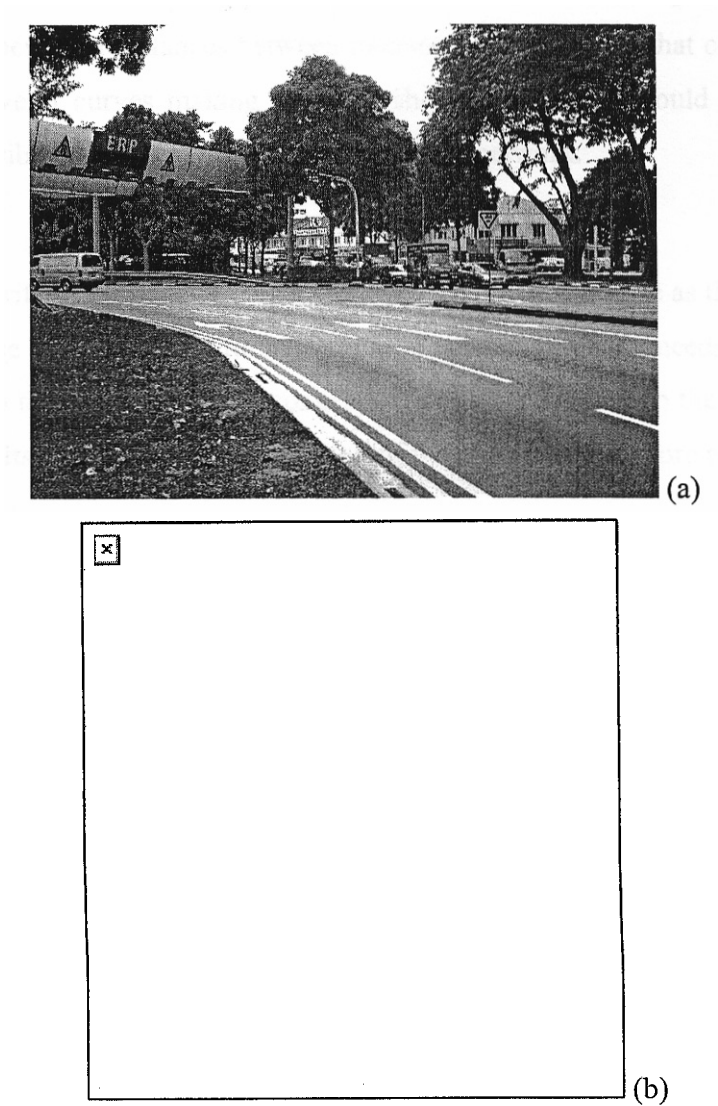


Figure 3.4: WIJ road network

3.5 Choosing the WIJ road network for case study

WIJ (as seen in Figure 3.4) is located slightly east off the centre of Singapore. The junction occupies an approximate area of 820m by 820m and is located below the Pan Island Expressway, PIE (one of the many freeways in Singapore). Two traffic signal lights and two other “Stop” signs to give the right-of-way in intersections control the

junction. WIJ, among many other busy junctions in Singapore, actually faces the problem of traffic congestion during the peak morning and evening hours. As shown in Figure 3.4(b), for a road network occupying the given area it has a fairly complicated road layout with rather short distances between intersections; adding to that complexity is the presence of several curves making up these short roads. These could be some of the reasons that attributed to the heavy traffic flow in the network.

Due to the severity of congestion the WIJ is actually a good example as the case study for this project since road network with no serious congestion basically needs no unnecessary improvement to the roads. With this congestion, any improvement to the road network as well as the results of the practical application of the model will be more apparent.

With the methodology described, the next step would be to look into the one-way urban road network model in detail. The desired conversion algorithm will be looked into first, followed by the min-cut graph model. For purposes of clarification, the WIJ road network will be used at each stage of the model's formulation to demonstrate the functioning of the model.

IV THE MECHANICS OF THE CONVERSION ALGORITHM

4.1 Overview

This chapter describes in detail the steps of the conversion algorithm from applying on a given two-way undirected network to obtaining an initial one-way road network layout. It first discusses the main intention and the mechanics of the algorithm, then moves on to examine the time taken (delay) by vehicles to cross intersections with various movement configurations (straight through, left turning or right turning). Consideration of delays at intersections is needed by the conversion algorithm to find the shortest total travel time (STTT) for the road network. Finally the choice for shortest path algorithm is chosen and incorporated into the conversion algorithm.

4.2 Modelling the conversion algorithm

The two-way road network is first represented as an undirected weighted graph $G = (V, E)$, with road intersections forming the nodes, V , and the bidirectional weighted roads between intersections forming the edges, E , of the graph. The times spent in traversing the roads give the weights of the edges. All the intersections are numbered for easy referencing; since no number is repeated, all intersections have unique identities.

The conversion algorithm takes G and produces a “feasible” weighted digraph, $G1$, such that $G1$ has the minimum STTT out of all possible digraphs derivable from G . This implies that a search for $G1$ out of all possible digraphs is required. In order to do this, all possible digraphs for G can be generated and compared. This entails three separate tasks:

- i. Converting the given undirected weighted graph G to a set of all its possible digraphs, $D1$.
- ii. $D1$ is trimmed further to a smaller set of “feasible” digraphs, $D2$, with a criterion to remove the “infeasible” digraphs.

- iii. Comparing the digraphs in $D2$ to get $G1$ (which may have more than one instance).
- iv.

Task (i) will be discussed first, followed by Task (ii) then Task (iii).

i. Converting the given undirected weighted graph G to all its possible digraphs

When an undirected edge is changed to a directed edge, there are two possibilities as shown in Figure 4.1.

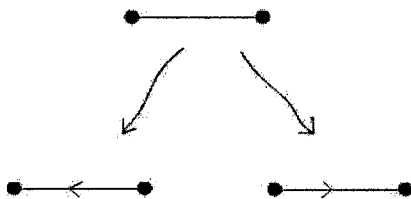


Figure 4.1: Two possible ways of changing an undirected edge to a directed edge

If G has n edges it means that there will be 2^n possible digraphs. This is a combinatorial problem as should the number n be big, there will be many digraphs to consider. In fact with present computing power, the time needed to search for $G1$ may take years for n greater than 32. To make the search more efficient, a systematic way to generate the digraphs can be used. Secondly, a “feasibility” criterion may be used as a constraint to get rid of the extraneous digraphs to reduce the search space. A systematic way to generate the digraphs will be explored next.

A systematic way to generate the digraphs

To form all digraphs from an undirected graph, both directional possibilities of each of the edges have to be considered in combination with both directional possibilities of all the other edges. This is a combinatorial problem. A systematic way to make the generation of digraphs exhaustive and easy to implement computationally is needed. One way to do this is described in the following example:

Figure 4.2 shows an undirected graph with three edges 1, 2 and 3 as well as three nodes A, B and C. First directions of all edges are fixed and the last edge, edge 3, is considered. Edge 3 will be flipped from one direction to the other as shown in row 1 and row 2 of

Table 4.1. This will give 2 possible digraphs. Edge 3 is flipped back to its original position.

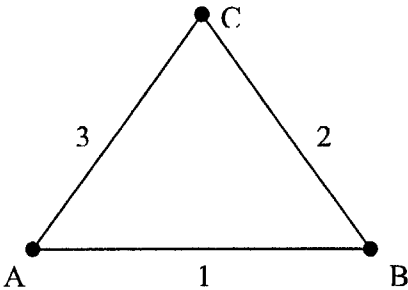
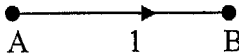
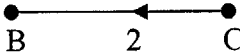
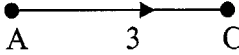
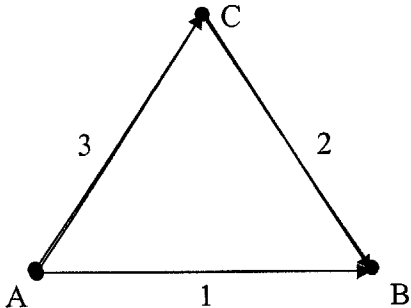
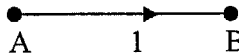
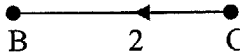
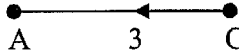
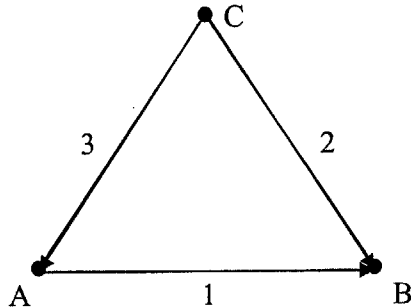
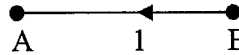
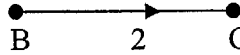
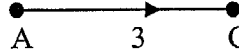
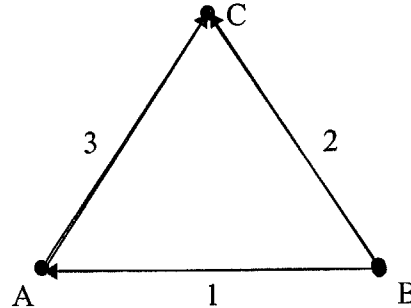
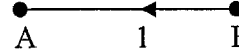
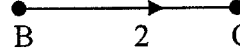
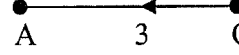
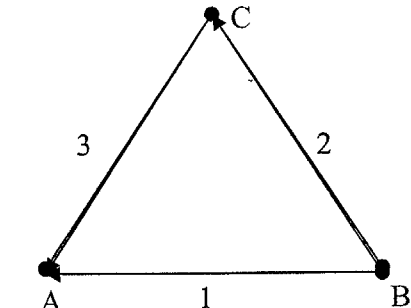


Figure 4.2: A simple graph example

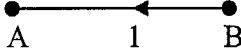
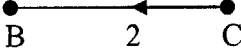
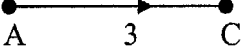
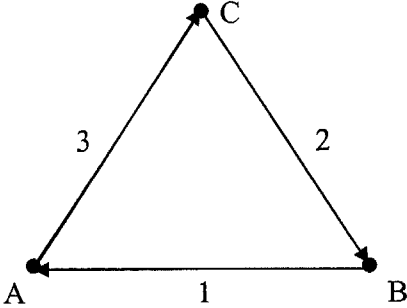
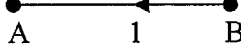
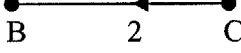
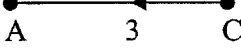
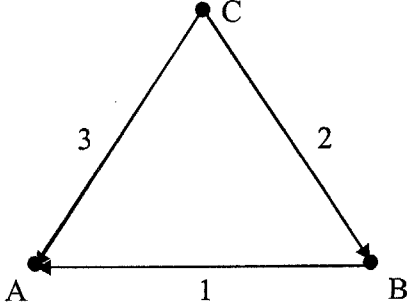
Table 4.1: Forming possible digraphs

No.	Edge 1	Edge 2	Edge 3	Digraph
2				

Cont'd

Cont'd

Next edge 2 is considered. The direction of edge 2 is changed and the edge 3 is flipped from one direction to the other as before (see row 3 and 4 of Table 4.1). This gives another 2 digraphs. Edge 2 and 3 are flipped back to its original position.

Lastly edge 1 is considered. The direction of edge 1 is changed and edge 3 is flipped from one direction to the other (as shown in row 5 and row 6 of Table 4.1). Subsequently edge 2 is flipped while edge 3 is flipped again from one direction to the other (as shown in row 7 and row 8 of Table 4.1).

This gives all the possible digraphs from the original undirected graph. The same technique can be used on any undirected graph in order to systematically obtain all possible digraphs.

Examining Table 4.1 row by row, it shows a systematic way of changing the directions of the edges from one to the other. This is not unlike the way a binary number increases in

its value (as shown in Table 4.2). Comparing Table 4.1 and Table 4.2, they are similar to each other except that directions of edges are used in Table 4.1 while a digit of “1” or “0” is used in Table 4.2. Letting “0” represents one direction of an edge and “1” represents another direction of an edge as shown in Figure 4.3, Table 4.2 can then be used to represent Table 4.1 instead.

Table 4.2: Incrementing a 3-digit binary number

Row No.	1 st digit	2 nd digit	3 rd digit	Binary No.
1	0	0	0	000
2	0	0	1	001
3	0	1	0	010
4	0	1	1	011
5	1	0	0	100
6	1	0	1	101
7	1	1	0	110
8	1	1	1	111

Incrementing an n-digit binary number can therefore be used to generate all possible digraphs from an undirected graph with n edges. Each binary number will correspond to a digraph. Furthermore, this would be an easy and systematic way to generate all digraphs (from a given undirected graph). This is because this will only involve adding a value of one to the binary number. From each binary number a digraph can then be modelled.



Figure 4.3: Representing directed edge with “0” and “1”

The first step of the conversion algorithm will therefore generate all the possible digraphs via the device of incrementing a binary number. For an undirected graph with n edges, an n-digit binary number is used. An algorithm increments the binary number by a value of one each time, starting from zero, until the binary number with value up to 2^{n-1} , i.e. a

binary number with all '1's, is reached. A digraph is obtained from each of the generated binary numbers. This whole process produces DI .

We shall now move on to discuss a criterion for ridding off the "infeasible" digraphs generated.

ii. Criterion to remove "infeasible" or extraneous digraphs

In order to devise a way to get rid of extraneous digraphs, possible logical constraints present may be explored. Since each generated digraph represents an actual road network configuration, the physical constraints of the road network form the criterion needed.

One obvious constraint to be imposed on the converted one-way road networks is that a vehicle should be able to get to any location of the resulting road network from all of the other locations. In graph theory terminology, each "feasible" digraph has to be **strongly connected**. Therefore, strong connectivity of the resulting one-way road network or digraph will be used as the criterion for getting rid of extraneous digraphs generated.

So by checking for strong connectivity of the set of digraphs, DI , generated in the first step, the next sub-algorithm of the conversion algorithm removes digraphs which are not strongly connected. As discussed in Section 4.3, the criterion of "strong connectivity" of a digraph has everything to do with the non-zero condition of the indegree and outdegree of any of its nodes. This step will greatly reduce the search space for $G1$.

iii. Comparing the digraphs to get $G1$

As discussed in Section 3.3.5, the STTT will be used as the criterion to select $G1$. All the digraphs that have both non-zero indegree and outdegree could then have their STTT calculated. Comparisons can then be made in order to determine the digraph with the minimum STTT.

In order to calculate the STTT (see Section 3.3.5), first the "standard" times spent in crossing intersections of all possible movement configuration are needed; secondly the time spent traversing each of the possible shortest route/path is needed:

- i) In order to determine a standard time for crossing an intersection (for all possible movement types to be classified), an analyser tool/method is required. This analyser tool/method will be able to model an experiment consisting of an intersection type and record the time required by each vehicle (and for each possible movement type) to cross the given intersection. The timings obtained can then be used in Equation (3.1) to find the STTT. (Details of the analyser tool together with the time spent in crossing each intersection type, are given in Section 4.3.3).
- ii) In order to compute the time spent traversing each possible shortest path, the speed of each vehicle and distance of each possible shortest path is needed. For the speed of each vehicle, the current speed limit of the road (50 km/h in this case) could be used. For the distance of each possible shortest path, a shortest path algorithm is required. This algorithm can then be used to find all the shortest paths from every node to every one of the other nodes. With the speed of the vehicle and distance needed, the time to get from one node to another can be found. (The choice of shortest path algorithm is discussed in Section 4.3.3). This combined with the timing (for crossing intersections) obtained can then be used to give the STTT.

With the STTT obtained for each of the “feasible” digraphs, comparisons can be made in order to find the feasible digraph with the minimum STTT.

To summarise, the conversion algorithm is made up of 3 parts.

- i) Converting the given undirected weighted graph to all its possible digraphs: this is accomplished through the device of incrementing a binary number.
- ii) Ensuring strong connectivity to reduce search space for G1: this gets rid of extraneous digraphs which are not strongly connected and hence infeasible for our purposes.

- iii) Computing the STTT: by first finding the times taken by which to cross each intersection type, and then by employing an efficient shortest path algorithm, the STTT for each of the feasible digraphs may be obtained. The systematic comparison of all STTTs then enables GI to be found.

The entire conversion algorithm, consisting of a series of sub-algorithms (described in Section 5.2), was tested on the UNIX platform with Silicon Graphics Inc. (SGI) Origin 3400 Server using C++ programming language. The source undirected weighted graph, G , is stored in its adjacency matrix form in a C++ program to be manipulated by the conversion algorithm.

4.3 The mechanics of the conversion algorithm

In this section, the mechanics and functioning of the conversion algorithm are explained. As detailed in the previous section, the conversion algorithm consists of three parts. The mechanics and functioning of each of the three parts will be explained in this section.

The first thing to consider before explaining each of the three parts however would be a way to represent the graph in the conversion algorithm. The graph could be represented either in its graphical form (with nodes and edges connecting each other to form a network) or in its adjacency matrix form. For an algorithm to run on a computer, should the graph be represented in its graphical form, the program would need to capture information in terms of coordinates of locations within a given 2-dimensional space. The alternative and more convenient representation is in the form of that graph's adjacency matrix. This matrix can be stored in a compact tabular form in the program and will hence be used.

4.3.1 First part: Convert an undirected weighted graph G to all its possible digraphs

The conversion algorithm would need to know the nodes that each edge is joined to. This is because when converting an undirected edge (A, B) to a directed edge, the conversion algorithm needs to be able to tell the direction of the directed edge using the nodes (that is

whether is it from node A to node B or from node B to node A). To do this the nodes which each edge is connected to must be known. The conversion algorithm will be able to capture this information by going through the elements of adjacency matrix. It scans every value in the adjacency matrix row by row; if the value being examined is not zero it records the row number and the column number where the entity resides. This corresponds to the number designations of the nodes that connect that edge.

In the adjacency matrix given in Figure 4.4 for example, the conversion algorithm will go through it row by row. In row 1 column 1, node 1 is not connected to node 1 (since it shows a value of 0). When the algorithm moves to row 1 column 2 it encounters a value of 1 showing that node 1 is connected to node 2. This will be captured by the algorithm as it takes down the value of (1, 2) to denote the edge connecting node 1 to node 2.

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Figure 4.4: An adjacency matrix

As discussed before, every undirected edge in an undirected graph can take up two possibilities when it is converted to a directed edge (see Figure 4.1). Thus, there will be 2^n number of possible digraphs to be generated for an undirected graph with n edges.

Let “0” denote one possibility and “1” denote the other possibility as shown in Figure 4.3. As discussed in the previous section, generating all possible digraphs from a given undirected graph with n edges can be seen as incrementing an n -digit binary number. In order to perform step 1 of the conversion algorithm, it will need to increment the n -digit binary number starting from the binary form of zero (for example 0000 for a 4-digit binary number). It will only stop at the binary form of $(2^n - 1)$ [for example 1111 or 15 ($2^4 - 1$) for a 4-digit binary number] instead of 2^n since the sequence starts at zero and not one. Increasing the binary number can be done by adding a value of 1 after each binary number is generated.

Next, the conversion algorithm will need to generate a digraph from a given binary number. In order for the conversion algorithm to do that, information on the node designation of each edge and the particular direction of the edge (“0” or “1”) need to be grouped together. From there, for any given graph (using the example given in Figure 4.2) and given any binary number (for example 010), the conversion algorithm will know that the first digit of “0” refers to directed edge (A, B), the second digit of “1” refers to directed edge (C, B) and the third digit of “0” refers to directed edge (A, C).

To do the above-mentioned, a matrix may be used. This is because a matrix would be easy to implement, manipulate and represent in the computer program. For every edge present, the conversion algorithm will allocate a 1 by 3 matrix. The first two columns of the matrix contain the numbers of the two nodes that the edge is connected to while the last column contains the possible direction of the edge, “0” or “1”. As shown in Figure 4.5(a), given an edge connected to node 1 and node 2, the first two columns of the matrix would take the values of “1” and “2”.

Having that, however, would be meaningless if we do not know which direction “0” refers to, or which direction “1” refers to. A rule can be implemented such that for any given edge (A, B), “0” always refer to from A to B and “1” always refer to from B to A. For the example given in Figure 4.5(a), the last column will take the value “0” to represent that the edge is directed from node 1 to 2. For Figure 4.5(b), the last column takes the value of “1” instead to represent the edge being directed from node 2 to node 1.

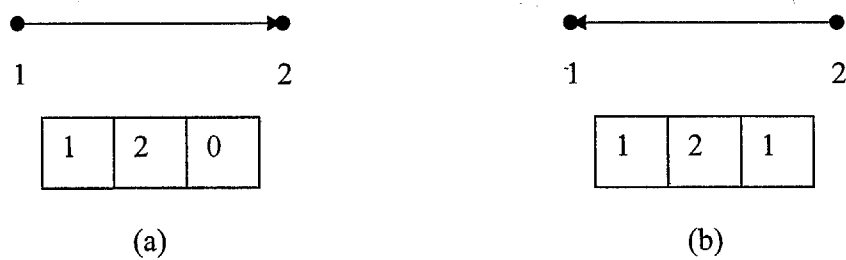


Figure 4.5: 1 by 3 matrix to represent an edge of the digraph

Based on the above reasoning, a graph with n edges will be allocated an $n \times 3$ matrix. For easy reference, this $n \times 3$ matrix will be referred to as the conversion matrix. The nodes

connecting each of the edges will be placed into the first two columns of the conversion matrix. The last column of the conversion matrix obtains its value successfully from the n -digit binary number. The first digit from the binary number will fill up the slot at the [first row, last column] of the conversion matrix, the second digit the slot at the [second row, last column] and so on. This is illustrated with the example shown in Figure 4.6.

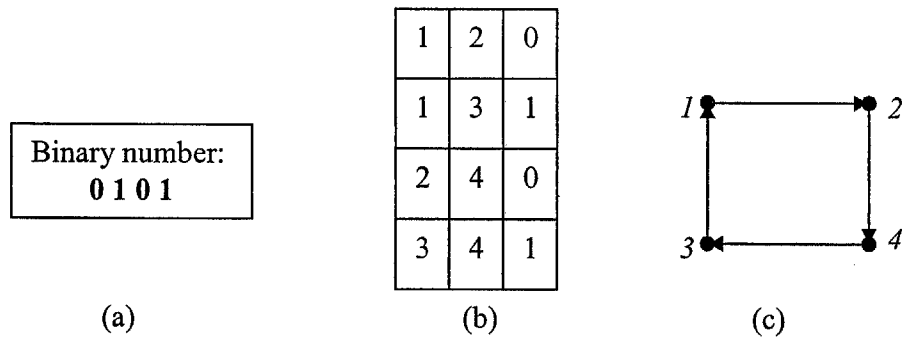


Figure 4.6: 4 by 3 matrix to represent the digraph

In Figure 4.6(a), the 4-bit binary number (as many bits as the number of edges are required) being represented has a value of 0101. This is mapped successively onto the last column of the 4 x 3 conversion matrix in Figure 4.6(b). From the first row of the conversion matrix that says “1, 2, 0”, an edge directed from node 1 to 2 will be formed [see Figure 4.6(c)]. Going down the conversion matrix row by row, the conversion algorithm will thus be able to form the digraph. Figure 4.6(c) shows the resultant digraph in this given case. The resultant digraph will then be stored in its adjacency matrix form.

In the discussion so far the graph being considered is an unweighted graph. However when the road network is modelled as an undirected graph, it would be weighted. The weights of the edges will be the lengths of the roads or the traversal times. Instead of using an $n \times 3$ matrix, an $n \times 4$ matrix could be used, with the fourth column containing the weights of the edges of the graph. With the weights present, generating the digraph in its adjacency matrix form would be easier. This is illustrated with the example given in Figure 4.7.

In Figure 4.7(a), the binary number generated at the time of the algorithm is 0101; this will be mapped onto the 3 columns of the conversion matrix, as shown in Figure 4.7(b).

Taking row 1 of the conversion matrix as example, the conversion algorithm will read “1 2 0 4”. The first two columns “1 2” only indicates it that it will make the insertion in either [row 1, column 2] or [row 2, column 1]. Having read the third column “0” the algorithm will make the choice for insertion to [row 1, column 2]. This is so since “0” means an edge leading from node 1 to node 2 and not an edge from node 2 to node 1. Reading the value found in the fourth column of the conversion matrix, it will insert the value “4” into the adjacency matrix, at the [row 1, column 2] location. The rest of the adjacency matrix entries are formed accordingly [see Figure 4.7(c)].

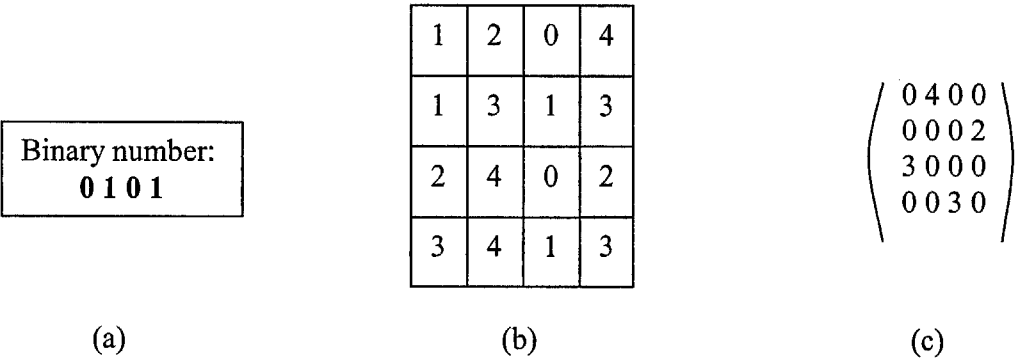


Figure 4.7: 4 by 4 matrix to represent the digraph

As can be seen, if an undirected graph with n edges is given, the conversion algorithm will first form an n -digit binary number. With the n -digit binary number, the conversion algorithm will increase it, step by step, from the binary form of zero to the binary form of (2^n-1) . At each step, with the binary number generated the conversion algorithm, as discussed above, will be able to produce the digraph the binary number represents. Hence, by advancing the binary number count, the conversion algorithm will be able to generate all possible digraphs.

4.3.2 Second part: Ensuring strong connectivity

To ensure strong connectivity of a digraph, the indegrees and outdegrees of all nodes of each digraph can be checked. A node with zero indegree implies that no edge leads towards it [as shown in Figure 4.8(a)]. Relating that in terms of road network, it means that the intersection has no road leading to it. Conversely, a node with zero outdegree

implies that no edge is leading away from it [as shown in Figure 4.8(b)]. This means that the intersection has no road leading away from it. Hence a digraph with zero indegree or zero outdegree for any of its nodes would not be strongly connected.

This part of the conversion algorithm rejects digraphs that are not strongly connected by ensuring that indegrees and outdegrees of every node in the digraph generated is not zero. It should be noted that while a digraph with zero indegree or outdegree for any of its nodes is not strongly connected, the converse is not necessarily true; a graph having no nodes with zero indegree or outdegree need not be strongly connected (see section 9.4.1.3).

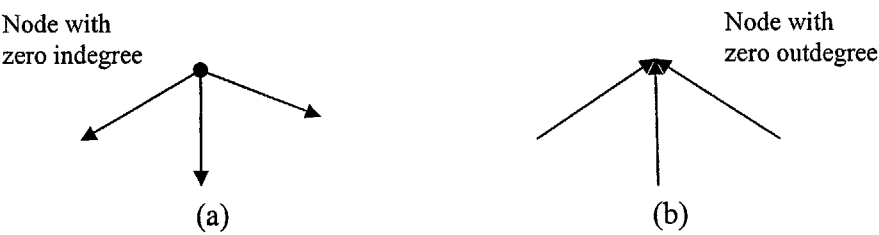


Figure 4.8: Nodes with zero indegree and outdegree

By inspecting the adjacency matrix of each digraph, this part of the conversion algorithm sums up all the row entries for each row of the adjacency matrix. This is to compute the value for the outdegree of each node in that digraph. Should any of the outdegrees give a value of zero, the conversion algorithm will discard the digraph at this stage (since the digraph is definitely not strongly connected).

Similarly, all entries for each column of the adjacency matrix are summed up to obtain the indegree of that node. Again, should any of the indegrees have a zero value the conversion algorithm (at that stage) will also discard that digraph. In fact a digraph is discarded as “infeasible” as soon as the first instance of a node having an indegree or outdegree of zero occurs.

With a large number of non strongly connected digraphs removed, the next step of the algorithm is to search among the remaining feasible digraphs for the digraph (or digraphs) with the minimum STTT.

4.3.3 Third part: Calculating STTT

As discussed in the previous section, the digraph (or digraphs) obtained from the final step of the conversion algorithm will have a minimum STTT. As such to calculate the STTT of the feasible digraphs generated the following is required:

- i. The standard times taken by a driver to cross a road intersection for various movement types (see Section 4.3.3.1).
- ii. A suitable shortest path algorithm to efficiently compute distance of the shortest path between locations. Hence by dividing the shortest path obtained by the average speed of the vehicles the shortest traversal time between locations needed to calculate STTT is obtained (see Section 4.3.3.3).

In the sections that follow, the standard times needed to cross a road intersection will first be developed. This is followed by a review of several shortest path algorithms for use in this project.

4.3.3.1 Time taken to cross a road intersection

In order to find the STTT for any given road network, the times spent at intersections (for all possible movements) have to be considered. This is so as according to Highway Capacity Manual [40], for urban roads the total travelling time is greatly affected by times spent crossing intersections. To find the times spent at intersection, an experiment can be conducted and empirical data collected for the time spent at intersections by vehicles for various movements, that is going straight through, turning right or turning left. This is so since different movements take different amount of time to cross an intersection.

In order to more accurately construct the experiment, its inputs and outputs are examined first. From the inputs and outputs of the experiment, a typical road intersection could be used and the times taken to cross the intersection by various vehicles for the various movement types can be recorded in order to gather the required output. Nevertheless, to carry out the experiment using an actual road intersection may be both tedious and resource consuming. Furthermore the road intersection used may not be generic enough to give a set of timing that could be used for all intersections. An analytical method or an analyser tool may be used instead to perform this experiment. For the reasons stated in Section 3.4.2.2, discrete simulation could be used as the analyser tool to conduct the experiment. Paramics is chosen as the software to be used to run the simulation. (The reasons for using Paramics are further discussed in Chapter 7).

In order to obtain an output of time taken for various vehicles to cross the intersection, an experiment can be conducted using the simulation. A generic intersection can be modelled and the standard times taken by vehicles crossing the intersection (for three movement types, to be further discussed in the next section) can be obtained.

4.3.3.2 Finding the time taken to cross an intersection

The goal of this simulation experiment is to find the standard times taken by a vehicle to cross an intersection. According to Ortúzar and Willumsen [1], the time taken by a vehicle to cross an intersection is defined as the time from when the vehicle starts to slow down upon nearing the intersection, crosses the intersection and then accelerates to a constant speed.

For a vehicle crossing an intersection, there are 3 possible movements. First, the through movement: the vehicle will drive straight past the intersection without making any turns. Secondly, the left turning movement: the vehicle makes a left turn at the intersection. Thirdly, the right turning movement: the vehicle makes a right turn at the intersection. The times taken for all three movements will be given in this report as the standard times taken to cross an intersection.

The experiment

A generic one-way intersection is needed in order to conduct the experiment. For the purpose of this project, either a three-legged or four-legged intersection may be used (as stated by Ortúzar and Willumsen [1], since any intersections with more than four different approaching traffic is better constructed as a round-about hence they are not considered as they are not within the scope of study). Nevertheless a four-legged intersection [see Figure 4.9(a)] is chosen instead of a three-legged intersection [see Figure 4.9(b)] because:

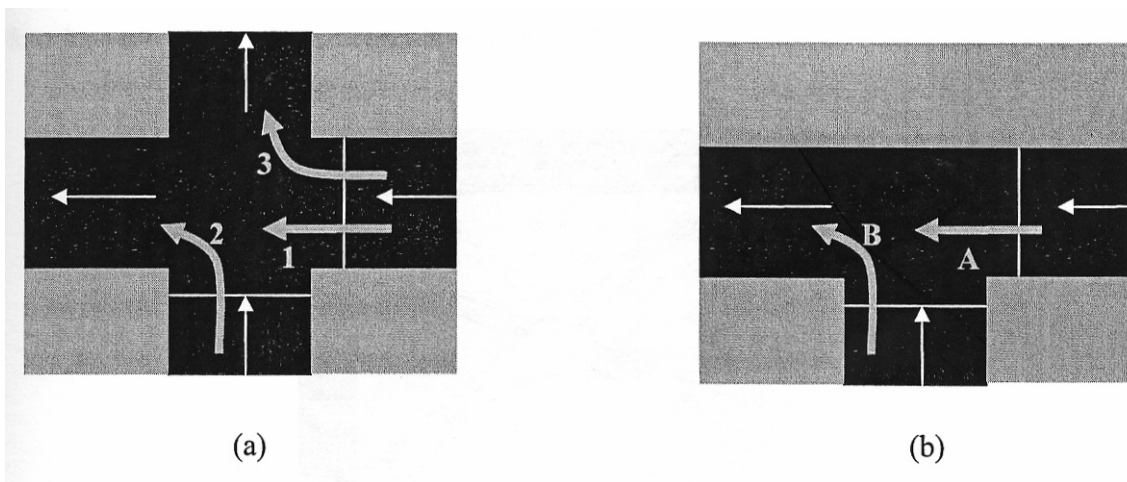


Figure 4.9: Possible intersections for the experiment

- i. Four-legged one-way intersection allows testing of all three movement types: “straight through” [movement 1 of Figure 4.9(a)], “left-turning” (movement 2) and “right-turning” (movement 3). While a three-legged one-way intersection only allows testing of two movement types: “straight through” [movement A of Figure 4.9(b)] and depending on the directions of the one-way road, either “left-turning” or “right-turning”. In this case of Figure 4.9(b), the movement type is “left-turning” (movement B).
- ii. Time obtained in a four-legged one-way intersection may be used for time spent in a three-legged one-way intersection. This is because movement types in the four-legged intersection face the same opposing traffic as the three-legged one-way intersection; that is they are the same. For example, movement 1 in Figure 4.9(a) only needs to look out for traffic from the left, this is the same as movement A in

Figure 4.9(b). Hence the time taken for both movements 1 and A are the same. On the other hand, movement 2 in Figure 4.9(a) needs to look out for traffic from the right which is the same as movement B in Figure 4.9(b). Hence the time taken for both movements 2 and B are the same.

Hence as described above, since a four-legged one-way intersection can test for more movement types than a three-legged one-way intersection and all movement types in the latter is the same as the movement types available in the former intersection, therefore a four-legged one-way intersection is used instead.

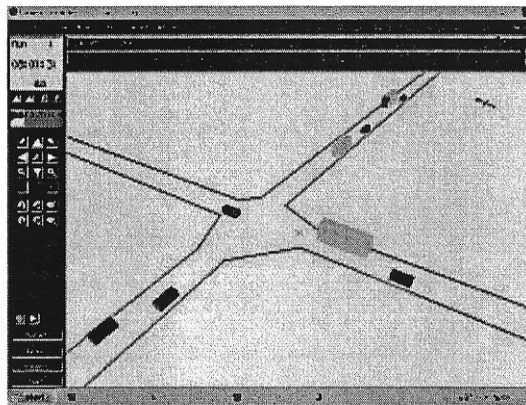


Figure 4.10: A four-legged intersection

A four-legged one-way intersection is designed for the experiment and shown in Figure 4.10. Vehicles are loaded into the intersection and tracked in order to record the speed of the vehicle at any one time. With the vehicle's speed tracked, the actual time when a vehicle starts to slow down as it approaches the intersection will be known. The actual time when the same vehicle reaches a constant speed again after crossing the intersection is known as well. From this, the total time taken for the vehicle to cross the intersection is obtained.

To illustrate this, an example is used. In this example, vehicle A is chosen and a tracker is placed onto it to enable the speed of the vehicle to be tracked throughout the duration of the simulation. A is then allowed to cross the intersection, making a left turn at the

intersection. A beacon (used by Paramics to track time) is placed at the location where A is expected to enter the intersection in order to note down the time when A just enters the intersection. The time when A just leaves the intersection is noted down by another beacon placed at the location where A is expected to leave the intersection. Results of this example are shown in tabular form in Table 4.3 and graphical form in Figure 4.11.

Table 4.3: Speed of A against actual time

Time (am)	Speed (km/h)
8:00:46	55.3
8:00:47	55.33
8:00:48	55.2
8:00:49	55.18
8:00:50	55.17
8:00:51	55.15
8:00:52	55.15
8:00:54	55.13
8:00:56	55.13
8:00:58	55.17
8:01:00	55.18
8:01:01	55.17
8:01:02	55.25
8:01:03	55.32
8:01:04	55.32

Region X in Figure 4.11 depicts the time before A crosses the intersection. Region Y is the time while A was in the intersection itself. Region Z shows the time when A had cross the intersection and was accelerating back to a constant speed.

As shown in Figure 4.11 at point s , A started to decelerate, slowing down for the approaching intersection. In region Z after A crossed the intersection, it started accelerating until point t when it reached a satisfactory speed and remained constant at it. The time for A to cross the intersection therefore would be the time from point s to point t . This value is 16 seconds, as recorded in the simulation experiment.

A set of 41 vehicles are tracked for each of the three movements for the intersection and the results are given in the next section. 41 vehicles are chosen for the sample size

because according to Walpole *et. al.* [85] one can then assume Normal Distribution for the mean time taken for the vehicles to cross the intersection for all 3 movement types as cited by the Central Limit Theorem and a 40 degrees-of-freedom can thus be used for the Student's t-distribution.

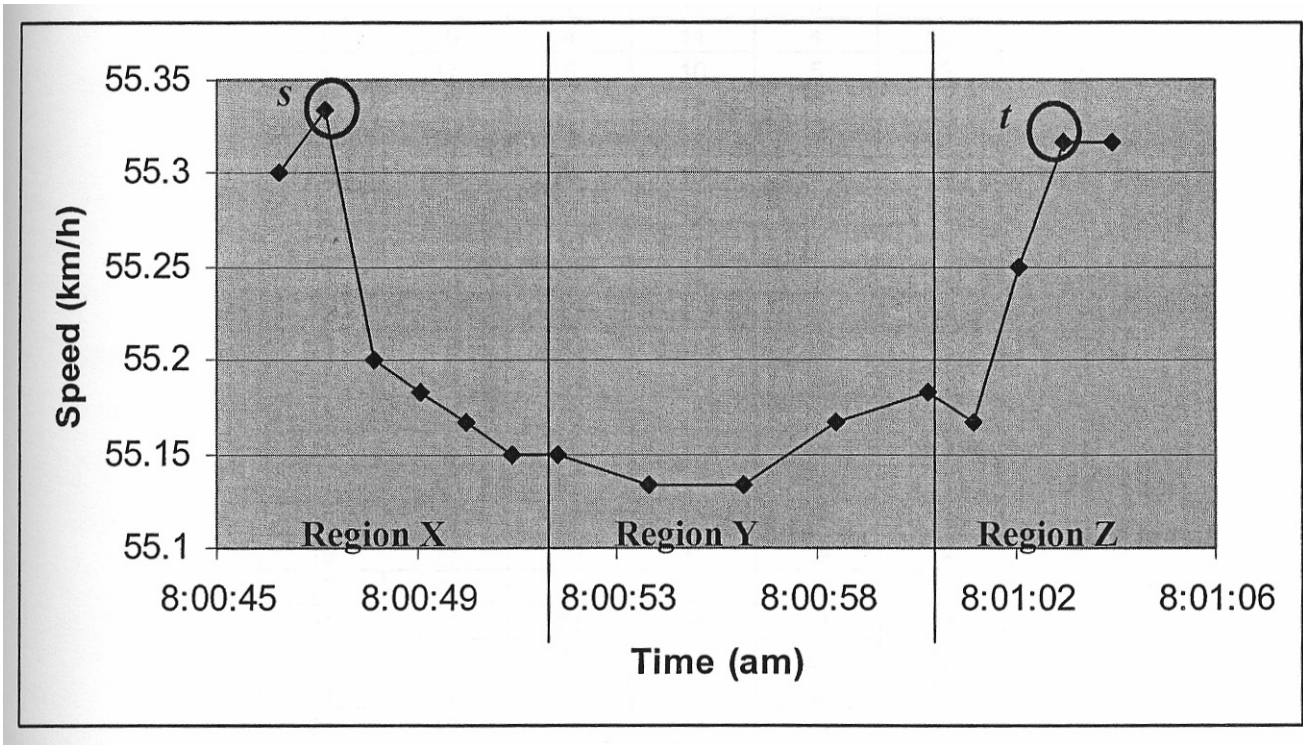


Figure 4.11: Speed of A against actual time

The results

Table 4.4 shows the time taken for each set of the 41 vehicles to cross the intersection. Letting T_1 , T_2 and T_3 be the time taken for any vehicle to cross an intersection with left turning, straight through and right turning movements respectively, we thus have:

For the “left turning” movement

The sample mean time taken to cross the intersection is 16.73 seconds while the sample variance is 0.7512 seconds. Since actual variance is unknown in this case, according to Walpole *et. al.* [85], the Student's t-distribution with 40 degrees-of-freedom is used to find the estimated time taken to cross the intersection.

Table 4.4: Time taken to cross the intersection

Left Turn		Straight Through		Right Turn	
Vehicle No.	Total Time Taken	Vehicle No.	Total Time Taken	Vehicle No.	Total Time Taken
1	18	1	11	1	13
2	17	2	11	2	12
3	17	3	11	3	13
4	16	4	11	4	12
5	17	5	10	5	13
6	17	6	11	6	14
7	16	7	11	7	13
8	17	8	11	8	13
9	17	9	11	9	14
10	18	10	11	10	13
11	18	11	12	11	14
12	17	12	11	12	13
13	16	13	11	13	13
14	16	14	11	14	12
15	16	15	11	15	13
16	17	16	11	16	14
17	16	17	11	17	14
18	16	18	11	18	14
19	17	19	11	19	14
20	17	20	11	20	13
21	17	21	11	21	13
22	17	22	11	22	13
23	17	23	11	23	13
24	17	24	11	24	13
25	16	25	11	25	13
26	17	26	10	26	13
27	17	27	11	27	14
28	16	28	11	28	13
29	17	29	11	29	14
30	17	30	11	30	14
31	18	31	11	31	13
32	18	32	12	32	13
33	17	33	11	33	14
34	16	34	11	34	13
35	17	35	11	35	12
36	17	36	11	36	13
37	17	37	11	37	13
38	13	38	11	38	13
39	16	39	11	39	12
40	16	40	11	40	12
41	17	41	11	41	13
Total	686	Total	451	Total	538
Mean	16.73	Mean	11	Mean	13.12

T_1 is estimated to be 16.73 seconds with a 95% confidence interval of (16.4548, 17.0087).

For the “straight through” movement

The sample mean time taken to cross the intersection is 11.0 seconds while the sample variance is 0.1000 seconds. Since population variance is unknown in this case, the Student's t-distribution with 40 degrees-of-freedom is used to find the estimated time taken to cross the intersection.

T_2 is estimated to be 11.0 seconds with a 95% confidence interval of (10.8990, 11.1011).

For the “right turning” movement

The sample mean time taken to cross the intersection is 13.12 seconds while the sample variance is 0.4098 seconds. Since population variance is unknown in this case, the Student's t-distribution with 40 degrees-of-freedom is used to find the estimated time taken to cross the intersection.

T_3 is estimated to be 13.12 seconds with a 95% confidence interval of (12.9174, 13.3265).

Time needed to cross the intersection

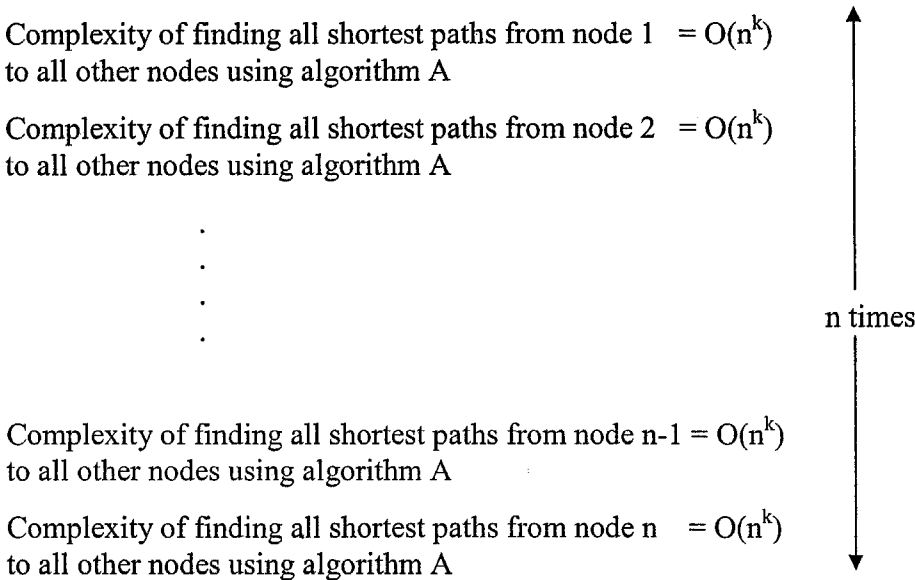
Value of T_1 is estimated to be 16.73 seconds, T_2 estimated to be 11.0 seconds and T_3 is estimated to be 13.12 seconds. The straight through movement, T_2 , is the least since vehicles need not make an additional manoeuvre (like making a left turn) in order to go pass the intersection. Both T_1 and T_3 are times for turning movements; however T_3 is less than T_1 . This is because in the simulation the vehicles are right hand drive. According to Cameron *et. al.* [56], the blind spot for right turning (in right hand drive) is less than blind spot for left turning in Paramics. The driver being seated further away from the left curb thus needs more reaction time when making a left turn as compared to making a right turn. Hence it is easier (and takes less time) for drivers to make a right turn rather than make a left turn.

4.3.3.3 Shortest path algorithm

This section will review several shortest path algorithms available to select a suitable shortest path algorithm for use in the conversion algorithm.

Complexity of algorithms as criteria to choose the shortest path algorithm

As the shortest total path requires the sum of all shortest paths from each node to all other nodes, it is viewed as an all-to-all shortest path problem. Hence if a one-to-all shortest path algorithm is used, the particular algorithm would have to be applied once for every one of the nodes present. According to Cherkassky *et al.* [87], given a graph G with n nodes, if an one-to-all shortest path algorithm A of complexity $O(n^k)$ (see Appendix B for complexity of problems and Big O notation) is applied once for every node present, this implies that its complexity would become $O(n^{k+1})$. This is shown by:



Therefore, the complexity of finding all shortest paths from all nodes to all nodes using algorithm A = $O(n^k) \times n$
= $O(n^{k+1})$

However, this does not mean that a one-to-all shortest path algorithm will not be considered for use in the conversion algorithm. Complexity of the algorithm (as described

by Big O notation) will be the more accurate criterion used to assess/compare suitability of any shortest path algorithm. This is shown in the example given. Should the complexity of an all-to-all shortest path algorithm B be $O(n^3)$ and that of an one-to-all algorithm C be $O(n)$ then:

Complexity of finding all shortest paths from all nodes to all nodes using algorithm B
 $= O(n^3)$

Complexity of finding all shortest paths from all nodes to all nodes using algorithm C
 $= O(n^{1+1}) = O(n^2)$

As algorithm C is the less complex algorithm, it would be chosen over algorithm B.

Choice for the shortest path algorithm

Although extensive empirical studies on the performance of shortest path algorithms have been reported in the literature [86, 87, 88, 89, 90, 91, 92, 93, 94], according to Yin *et al.* [95] there is no clear answer as to which algorithm, or set of algorithms performs best (based on the algorithm's complexity) on real road networks.

A study by Cherkassky *et al.* [87] presented one of the most comprehensive evaluations of shortest path algorithms. They evaluated a set of 17 shortest path algorithms. In their experiment, Cherkassky *et al.* coded the 17 algorithms using the C programming language and tested the C programs on a SUN Sparc-10 workstation. One-to-all shortest paths can be computed by these C programs. A number of simulated networks with various degrees of complexity were used for evaluating the algorithms. The results of their studies suggest that no single algorithm performs consistently well on all simulated networks.

More recently, Zhan and Noon [96] used 21 real road networks for evaluating the shortest path algorithms. These 21 networks included the U.S. National Highway Planning Network (NHPN) covering the continental U.S. and 20 state-level road networks generated from road networks in 10 states in the Midwest and Southeast of the United States.

Two shortest path algorithms that perform best on real road networks were identified by Zhan and Noon [96], these two algorithms are:

- 1) The Pallottino graph growth algorithm [90].
- 2) The Dijkstra algorithm [89].

Based on their evaluation, it was suggested that the best performing implementation for solving the one-to-all shortest path problem is Pallottino's graph growth algorithm with two queues (better known as TQQ) [90]. They further suggested that when the goal is to obtain a one-to-one shortest path or one-to-some shortest paths, the Dijkstra's algorithm offers some advantages because it can be terminated as soon as the shortest path distance to the destination node is obtained. Based on recommendations by Zhan and Noon [96], only the TQQ [90] and Dijkstra's algorithm [89] would be taken into consideration for the use in the conversion algorithm.

As Pallottino's algorithm is a one-to-all shortest path algorithm it is best suited for the one-to-all shortest path problem. This means that when it is used for a graph with n nodes it has to be performed n times (once for each node) to solve the all-to-all shortest path problem. According to Sakellariou [98], Pallottino's algorithm complexity thus becomes $O(n^4)$ for all-to-all shortest path problem.

The Dijkstra's algorithm is also a one-to-all shortest path solution. As with Pallottino's algorithm, when it is applied to a graph with n nodes it needs to be performed n times to solve the all-to-all shortest path problem. As stated by Cherkassky [87], Dijkstra's complexity thus become $O(n^3)$ for the all-to-all shortest path problem and according to Crauser *et al.* [99] this thus make Dijkstra's algorithm a better choice over Pallottino's algorithm.

Incorporating time spent at intersections into Dijkstra's algorithm

To explore how the times spent at intersections may be incorporated into the shortest path algorithm (Dijkstra's in this case), Dijkstra's algorithm is first discussed.

A formal description of Dijkstra's algorithm is given in Appendix C. Generally in Dijkstra's algorithm, the mechanism for finding the shortest path between node s and node t , d_{st} , is to compare the sum of d_{sm} (the shortest path between node s and node m) and d_{mt} (the shortest path between node m and node t) with the currently known d_{st} (which is initialised to infinity) as shown in Figure 4.12. Should d_{st} be longer, it will be replaced by $(d_{sm} + d_{mt})$ else d_{st} remains unchanged. The comparison continues in a similar fashion until all the shortest paths from s to the rest of the nodes in the graph are found.

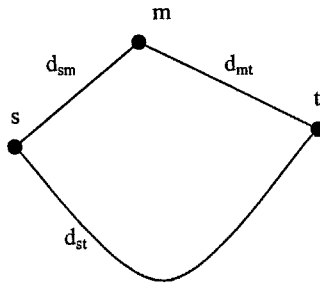


Figure 4.12: Comparing d_{st} with $(d_{sm} + d_{mt})$

As discussed in Section 4.3.3 the STTT is required to find the network/digraph with the minimum STTT. So in order to find the STTT spent in the road network, instead of finding the actual distance between two nodes Dijkstra's algorithm will be modified to find the time spent travelling between the two nodes (or simply time taken). This is done by dividing the actual distance by the average speed of the vehicle (50km/h is used in this case). Hence shortest path, d_{st} (between two nodes s and t), as given by Dijkstra's algorithm after being divided by the average speed, AveSpeed, will give the shortest time, t_{st} , spent travelling between the two nodes s and t . Thus by introducing AveSpeed into Dijkstra's algorithm and dividing all shortest paths obtained by AveSpeed, Dijkstra's algorithm is modified to give shortest traversal time instead of shortest path. (A detailed description of the modified Dijkstra's algorithm will be given in the later part of this Chapter).

Next the time spent at each intersection can be incorporated into Dijkstra's algorithm by considering the type of movement (whether "straight through", "left turning" or "right turning") made at the node m from node s while traversing to node t . Should the movement be "left turning" then T_1 (see Section 4.3.3.2) will be added into the time spent

in traversing from node s to node m then to node t . Should the movement be “straight through” then T_2 will be added; T_3 will be added when the movement is “right turning”.

The main mechanism of the Dijkstra’s algorithm is now modified to compare the shortest known traversal time, t_{st} , (between node s and node t) with $(t_{sm} + t_{mt} + T_i)$ instead of shortest known path, d_{st} , with $(d_{sm} + d_{mt})$. Where T_i is the time spent at the intersection and could take the value of either T_1 , T_2 or T_3 (depending on the type of movement required at node m to get to node t from node s).

However this suggests that the type of movement needed at node m to get to node t from node s should be known. This can be obtained from the user as the graph/road network is given (by the user) for the first time. This information can be stored in a three-dimensional matrix which allows easy retrieval as well as insertion as an array when run as a computer program on the SGI workstation. For easy reference, this three-dimensional matrix will be known as intersection time matrix, B . Furthermore, semantically it is an appropriate representation of the actual situation. For example, for the time (be it T_1 , T_2 or T_3) spent in crossing the intersection at node m from node s to node t [as shown in the road network given in Figure 4.13(a)] can be stored at the location $[s, m, t]$, read as [row s , column m , height t] [as shown in the three-dimensional matrix given in Figure 4.13(b)]. Comparing Figure 4.13(a) and Figure 4.13(b) they show a very close representation of the actual situation whereby the vehicle moves from node s , past node m to node t .

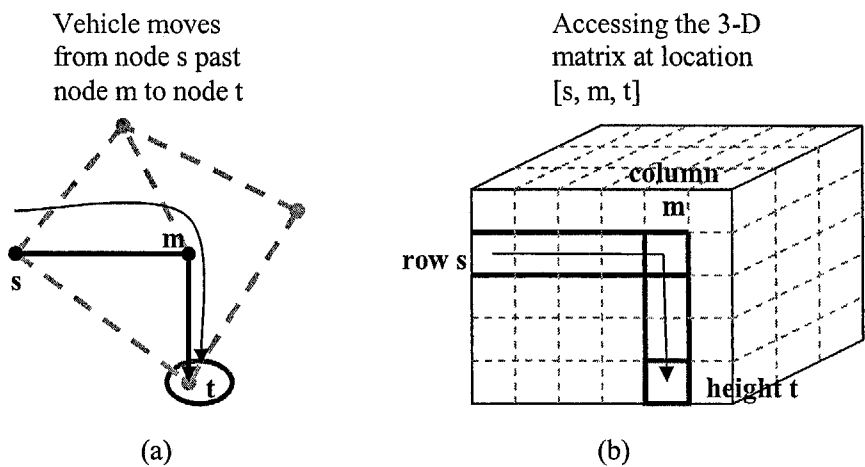


Figure 4.13: Comparing the road network with the 3-D matrix

The Dijkstra's algorithm is thus modified and is now suitable for calculating the STTT of a network instead of the shortest path. The formal description of the modified algorithm is as given:

Procedure Modified Dijkstra

Begin

 Initialise sorted list, L, and put s, the source node, into L.

 Initialise P, matrix for storing previous nodes of the shortest path.

 Initialise t, matrix for storing current traversal time for shortest paths, and set all entries

 to infinity.

 Initialise B, three-dimensional matrix for holding time spent at the various intersections

 in the graph

 Set $t[s]$ to 0.

 Set $P[s]$ to 0.

 While L is non-empty

 Set current node, X, to $L[1]$.

 For all nodes, Y, adjacent to X

 Set $T_i = B[P[X]][X][Y]$

 If $\{t[X] + [\text{weight of edge (X,Y)} \div \text{AveSpeed}] + T_i\} < t[Y]$

 Set $t[Y] = t[X] + [\text{weight of edge (X,Y)} \div \text{AveSpeed}] + T_i$.

 Set $P[Y] = X$.

 Add Y to L.

 Endfor.

 Remove X from L.

 Sort L in ascending order.

 Endwhile.

End.

As explained the main difference between the presented modified Dijkstra's algorithm and the original Dijkstra's algorithm is in the main mechanism of the algorithm. Here the

average speed of vehicle, AveSpeed, is used to divide the weight of edge (X,Y) (which in actual fact is the path length of the XY), this would give the time taken to traverse edge (X,Y) instead of distance/path length. T_i which is the time spent in the intersection is obtained from B, the three-dimensional matrix, and added into the main mechanism for comparison. This would allow the comparison of the total time for traversal between node X and node Y plus the time spent in the intersection (when the vehicle traverse from the previous node of X, P[X], through node X to node Y) with the current known shortest traversal time from node X to node Y, $t[Y]$. The one with the smaller value will be labelled as $t[Y]$. The process then continues until L is empty, that is all nodes have been used.

All values in the matrix, t, can then be added up to give the shortest time spent for traversing the network with node s as the source (see Equation 3.1). Another node can then be selected to be the source node and the process continues until all nodes in the graph have been used as the source node. From there STTT can then be obtained (see Equation 3.2).

The STTT of various digraphs enumerated by the conversion algorithm can then be obtained and compared. The digraph with the minimum STTT will then be given by the conversion algorithm as the suitable initial one-way road network layout.

With the workings and mechanism of the conversion algorithm explained, the conversion algorithm can now be developed and discussed in the next chapter.

V THE DEVELOPMENT OF THE CONVERSION ALGORITHM

5.1 Overview

Following the discussion on the mechanics of the conversion algorithm (in Chapter 4), the development process of the conversion algorithm, which is actually a sequence of sub-algorithms, is now detailed. Finally, the conversion algorithm is verified using a simple graph example and applied to the WIJ road network case study in Section 5.4.

5.2 The conversion algorithm and its development

This section will first discuss the use of sub-algorithms in the conversion algorithm. It will subsequently examine the development of each of the sub-algorithms and how each relates to the others in the conversion algorithm.

5.2.1 Using sub-algorithms

As discussed in Section 4.2, the algorithm has to perform 3 major tasks. This suggests that the algorithm can be fashioned as a main algorithm consisting of 3 sub-algorithms (or objects). The conversion algorithm is written within an object-oriented framework using the C++ programming language running on the SGI workstation. According to Stroustrup [107], by using an object-oriented framework, readability of the programs is enhanced since the codes are compartmentalised. As stated by Märtin [108], object-oriented programming will allow easier debugging and troubleshooting, due also to the compartmentalisation. As each of the tasks is encapsulated as an object, should a single task fail to perform as expected, the location of the bug can easily be traced. Furthermore objects can inherit from objects and can be reused (when needed) therefore reducing the need to rewrite.

5.2.2 The algorithms

Algorithm 5.1: Conversion

This is the main algorithm, the conversion algorithm itself. It will take in an undirected graph (in its adjacency matrix form) as input and produce the digraph with the minimum shortest total time (in its adjacency matrix form) as the output.

The main algorithm needs to be initialised before it is ready to convert an input undirected graph into the required digraph. The initialisation will consist of accepting the input from the user of the algorithm. To do this, a sub-algorithm for receiving the adjacency matrix of the undirected graph is required. This sub-algorithm, due to its functionality, is simply called *Sub-Algorithm 5.2: Initialisation*.

Having obtained the adjacency matrix of the undirected graph, there is a need to allocate a matrix, M , for representing the graph used as the conversion matrix. This calls for a sub-algorithm that will store the graph into the conversion matrix. This sub-algorithm will be called *Sub-Algorithm 5.3: Store the graph into conversion matrix M* .

As discussed in Section 4.2, the algorithm to produce the digraph will consist of 3 parts. These 3 parts or tasks are fashioned as 3 sub-algorithms to be called upon as and when needed by the main algorithm. The 3 tasks are:

- i. Converting the given undirected weighted graph G to all its possible digraphs
- ii. Ensuring indegree or outdegree is not zero
- iii. Calculating and comparing STTT

Upon closer inspection, part (ii) and part (iii) are actually nested within part (i). This is because part (i) will first generate a digraph, part (ii) will examine the digraph for the first occurrence of non-zero indegree or outdegree. Part (iii) then calculates the shortest total path of a feasible digraph and compares it to the minimum shortest total path obtained so far. Part (i) will then be called again to generate the next digraph. So instead of having 3 sub-algorithms for the 3 parts, one sub-algorithm will be used in which the other 2 sub-

algorithms are nested within (as shown in Figure 5.1). This sub-algorithm will be called *Sub-Algorithm 5.4: Producing digraph*.

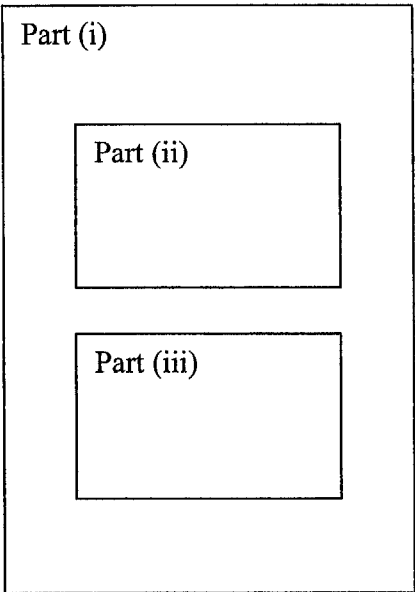


Figure 5.1: Part (ii) and part (iii) nested within part (i)

With that main algorithm, Algorithm 5.1, is formed. Figure 5.2 shows the flowchart for Algorithm 5.1. As shown in Figure 5.2, Algorithm 5.1 will consist of 3 sub-algorithms being carried out one after the other. First the input is obtained using Sub-Algorithm 5.2, next the graph is stored into its conversion matrix by Sub-Algorithm 5.3 and lastly the digraph is produced using Sub-Algorithm 5.4. The sub-algorithm structure is given as follows:

ALGORITHM 5.1: Conversion

1. Do Sub-Algorithm 5.2: Initialisation.
2. Do Sub-Algorithm 5.3: Store the graph into conversion matrix M.
3. Do Sub-Algorithm 5.4: Produce digraph.

Sub-Algorithm 5.2: Initialisation

The main objective of this sub-algorithm is to obtain the adjacency matrix of the undirected graph from the user as well as the inputs for the intersection time matrix, B. Obtaining these inputs from the user is viewed as the input of this sub-algorithm. The

adjacency matrix of this input graph and time matrix, B, are viewed as the outputs of this sub-algorithm.

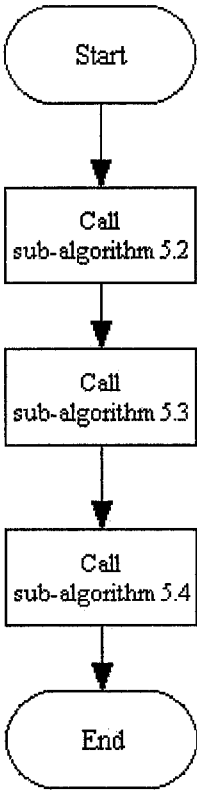


Figure 5.2: Flowchart for Algorithm 5.1

The size of the graph, in terms of the number of nodes present, needs to be known. This will allow a matrix of the right size to be allocated for use as the adjacency matrix. Letting n be the number of nodes present in the graph, this sub-algorithm will first ask user to give a value for n . With n obtained, the sub-algorithm will ask user to give the adjacency matrix of the graph which will be stored into a n by n matrix called A .

With matrix A provided by the user, matrix B will be created in the next step. In order to produce the intersection time matrix, this sub-algorithm will have to go through every row and column of the adjacency matrix. A systematic way to do this may be to examine every column of the first row before moving on to the next row. For example it will examine $A[1][1]$ (where $A[i][j]$ stands for entry of the matrix A at row i and column j) then $A[1][2]$ then $A[1][3]$ until $A[1][n]$ before starting again in $A[2][1]$.

Two indices of a matrix, i and j , may be used; i to represent each row while j to represent each column, as $A[i][j]$. Both i and j will be initialised to value of 1, j will be incremented to n . At the same time the sub-algorithm will be examining from $A[1][1]$ to $A[1][n]$. This can be done with a loop whereby j is used as the index and increased till the desired value (which is n in this case). This loop will be placed within another loop with i as the index; each time j reaches n , i will be incremented by 1 and j set back to 1. For example once $A[1][n]$ is reached, i will be increased to 2 with j set back to 1 giving $A[2][1]$; this allows the next row of A to be examined. By setting the limit of i to n as well this makes the two loops stop only when $A[n][n]$ is examined. This allows the whole adjacency matrix to be examined from $A[1][1]$ until $A[n][n]$.

In examining every entry $A[i][j]$, the sub-algorithm checks if $A[i][j]$ is zero. A zero denotes that there is no edge between node i and node j . A non-zero value, for example δ , denotes that there is an edge between node i and node j with a weight of δ . Should the value in $A[i][j]$ be non-zero (meaning there is an edge between node i and node j) then another index, h , will be used; the value of h will be incremented from 1 to n . Values in all possible $A[j][h]$ will be looked into to check for any edge existing from node j to node h . For any edges where $h \neq i$ (since we already know that there is an edge between node j and node i), the program will then ask the user to input the type of turn present when moving from node i (past node j) to node h . Depending on the user's input, T (T_1 , for left turning movement; T_2 , for straight through movement; T_3 for right turning movement), will be inserted into $B[i][j][h]$. The matrix, B , will thus be formed once every entry in $A[i][j]$ is examined.

With that, Sub-Algorithm 5.2 is formed, Figure 5.3 shows the flowchart for Sub-Algorithm 5.2 and the sub-algorithm is given as follows:

SUB-ALGORITHM 5.2: Initialisation

1. Input: n , number of nodes present in the graph.
2. Input: A , adjacency matrix of the graph.
3. Initialize i to 1.
4. If $i \in [1, n]$

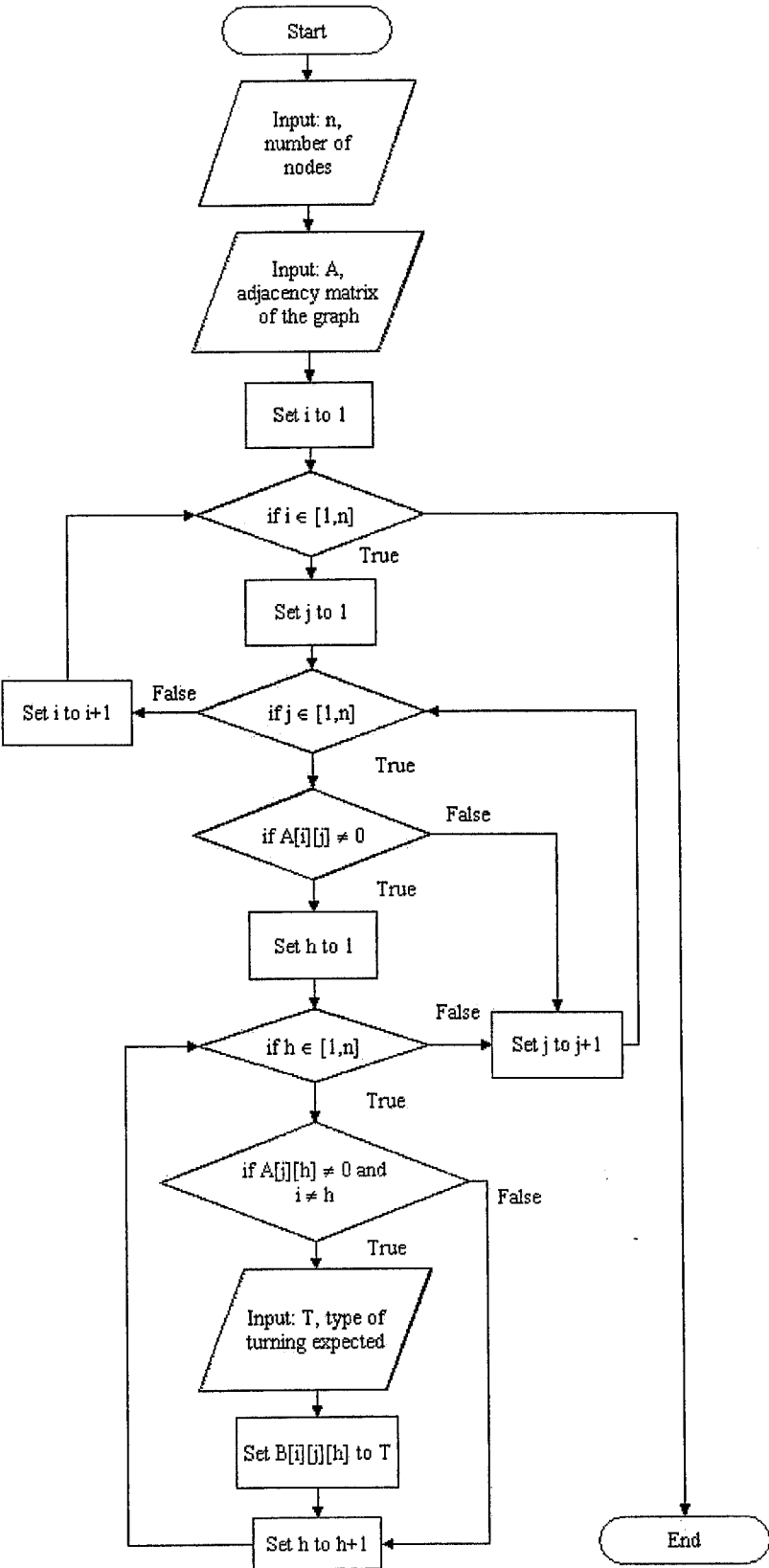


Figure 5.3: Flowchart for Sub-Algorithm 5.2

- 5. Set j to 1.
- 6. If $j \in [1,n]$
- 7. If $A[i][j] \neq 0$
- 8. Set h to 1.
- 9. If $h \in [1,n]$
- 10. If $A[j][h] \neq 0$ and $i \neq h$
- 11. Input: T, type of turning expected at intersection.
- 12. $B[i][j][h] = T$.
- 13. Set h to h+1.
- 14. Set j to j+1.
- 15. Set i to i+1.

Sub-Algorithm 5.3: Store the graph into conversion matrix M

This sub-algorithm will go through the adjacency matrix of the graph and store the graph into the conversion matrix, M. The input of this sub-algorithm is the adjacency matrix while the output is the conversion matrix.

In order to produce the conversion matrix, this sub-algorithm will have to go through every row and column of the adjacency matrix as described in Section 4.3. A systematic way to do this may be to examine every column of the first row before moving on to the next row as described in Sub-Algorithm 5.2. In examining every entry $A[i][j]$, the sub-algorithm checks if $A[i][j]$ is zero to determine if there is an edge between node i and node j. In this case the sub-algorithm will note down the nodes that the edge is connected to (that is node i and node j) as well as the weight of the edge, δ . Value of i, j and δ will then be placed into the first row of M (named max) as shown in Figure 5.4. However, do note that the third column is automatically set to “0”. This is so since this column contains the binary number which starts with a value of zero.

i	j	0	δ
---	---	---	----------

Figure 5.4: A row in M

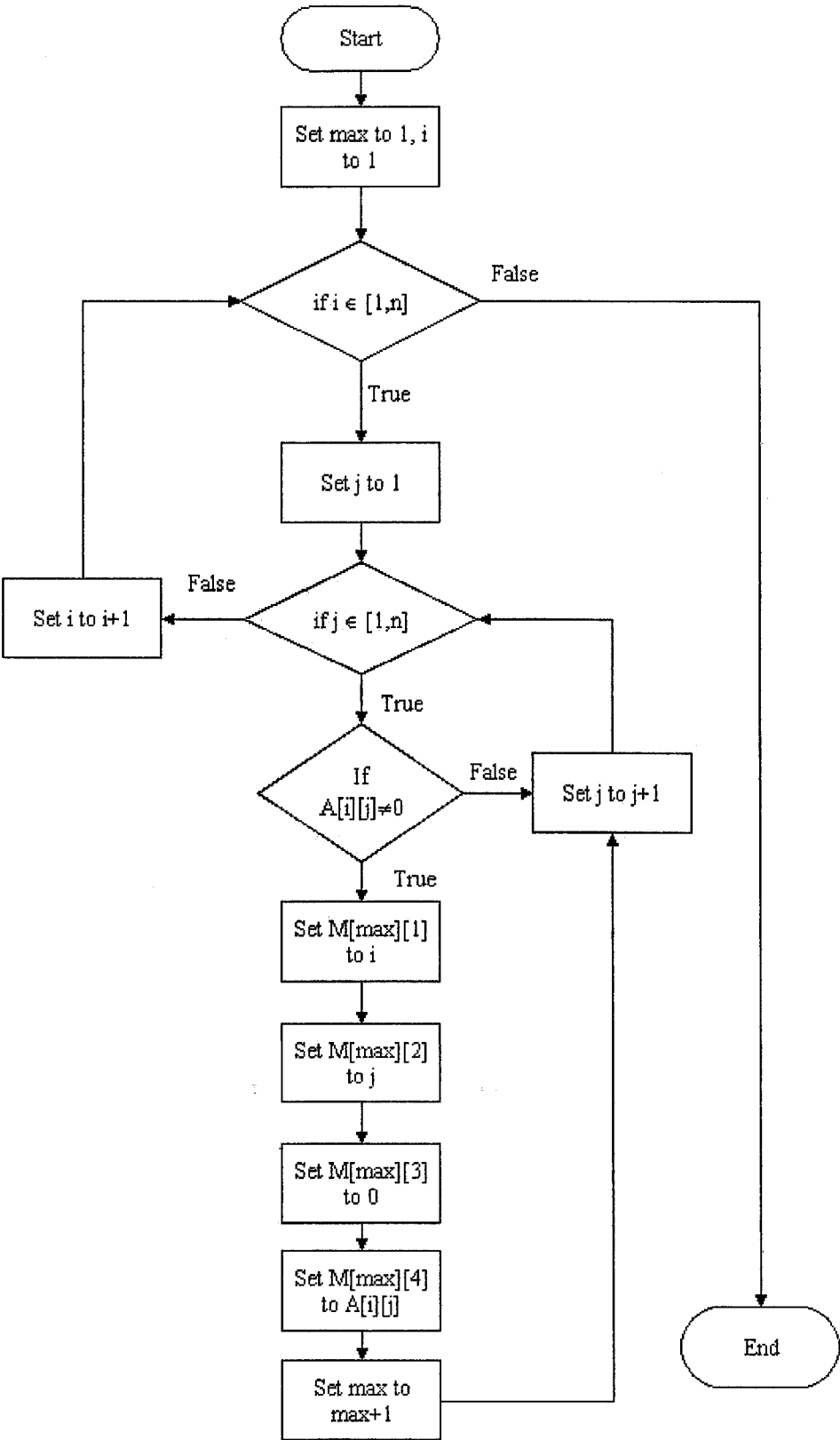


Figure 5.5: Flowchart for Sub-Algorithm 5.3

The value of max is then incremented by one. This is done so that the sub-algorithm need only insert values of i, j and δ into row max regardless of what the value of max is.

Figure 5.5 shows the flowchart for Sub-Algorithm 5.3. As shown, the algorithm will initialise max and i to 1. It will then use 2 nested loops (one with j as index within another with i as index) to examine every entry in A. Should $A[i][j]$ be non-zero, the algorithm will store the nodes to which the edge is connected and the weight of the edge into M.

SUB-ALGORITHM 5.3: Store the graph in conversion matrix M

1. Initialize both i and max to 1.
2. If $i \in [1, n]$
3. Set j to 1.
4. If $j \in [1, n]$
5. If $A[i][j] \neq 0$
6. $M[\text{max}][1] = i.$
7. $M[\text{max}][2] = j.$
8. $M[\text{max}][3] = 0.$
9. $M[\text{max}][4] = A[i][j], \text{ weight of edge } (i, j).$
10. $\text{max} = \text{max} + 1.$
11. Set j to j+1.
12. Set i to i+1.

Sub-Algorithm 5.4: Produce the minimum STTT digraph

This sub-algorithm takes M as input, generates all possible digraphs and removes those with zero indegree or outdegree. It will subsequently compute the shortest total time for the digraphs and obtain the digraph with the minimum shortest total time as output.

As an n-digit binary number is needed to generate all possible digraphs, this sub-algorithm first allocates a , an n-digit binary number, and set its value to zero. Since this binary number is to be incremented until the value $(2^n - 1)$ is reached (that is the binary number with n '1's), a program loop is used to increment it, while generating a digraph at each iteration.

The binary number is first inserted into the third column of M (see Section 4.3). Next, the sub-algorithm will produce a digraph from the given M . Since this by itself is a specific function, a sub-algorithm is written to carry out the task. The sub-algorithm is named *Sub-Algorithm 5.5: Forming the digraph using M* .

Each digraph formed from Sub-Algorithm 5.5 is checked for non-zero indegree and outdegree. This, again being a specific function by itself, is carried out using another sub-algorithm, named *Sub-Algorithm 5.6: Checking digraph indegree and outdegree*.

For each digraph with non-zero indegree and outdegree, the next step involves calculating the shortest total time of the digraph as defined in Section 4.3.3. To do this, the Dijkstra's algorithm is modified to produce an all-to-all shortest time matrix instead of all-to-all shortest path matrix. (General outline of the modification is given in Section 4.3.3.3). The modified Dijkstra's algorithm, again being a specific function by itself, is written as an independent sub-algorithm. This sub-algorithm is named *Sub-Algorithm 5.7: Modified Dijkstra's algorithm*.

With the all-to-all shortest time matrix obtained for each digraph, the STTT of the digraph may be computed and compared with the STTTs of other digraphs. These tasks are carried out using another sub-algorithm, *Sub-Algorithm 5.8: Finding the 'minimum STTT' digraph*.

Once the digraph with the minimum STTT is obtained, results are printed out as output for the user to analyse. This is done by a sub-algorithm called *Sub-Algorithm 5.9: Print results*.

With the foregoing, Sub-Algorithm 5.4 is formed. Figure 5.6 shows the complete flowchart for Sub-Algorithm 5.4. To summarise, Sub-Algorithm 5.4 consists of the following sub-tasks: First an n -digit binary number, a , is initialised to zero. A simple loop exhaustively generates all possible digraphs with each digraph generated by Sub-Algorithm 5.5. Sub-Algorithm 5.6 ensures non-zero indegree and outdegree. The all-to-all

shortest path matrix is then obtained using Sub-Algorithm 5.7 and the digraph with the minimum shortest total path then obtained using Sub-Algorithm 5.8. Results are finally printed using Sub-Algorithm 5.9.

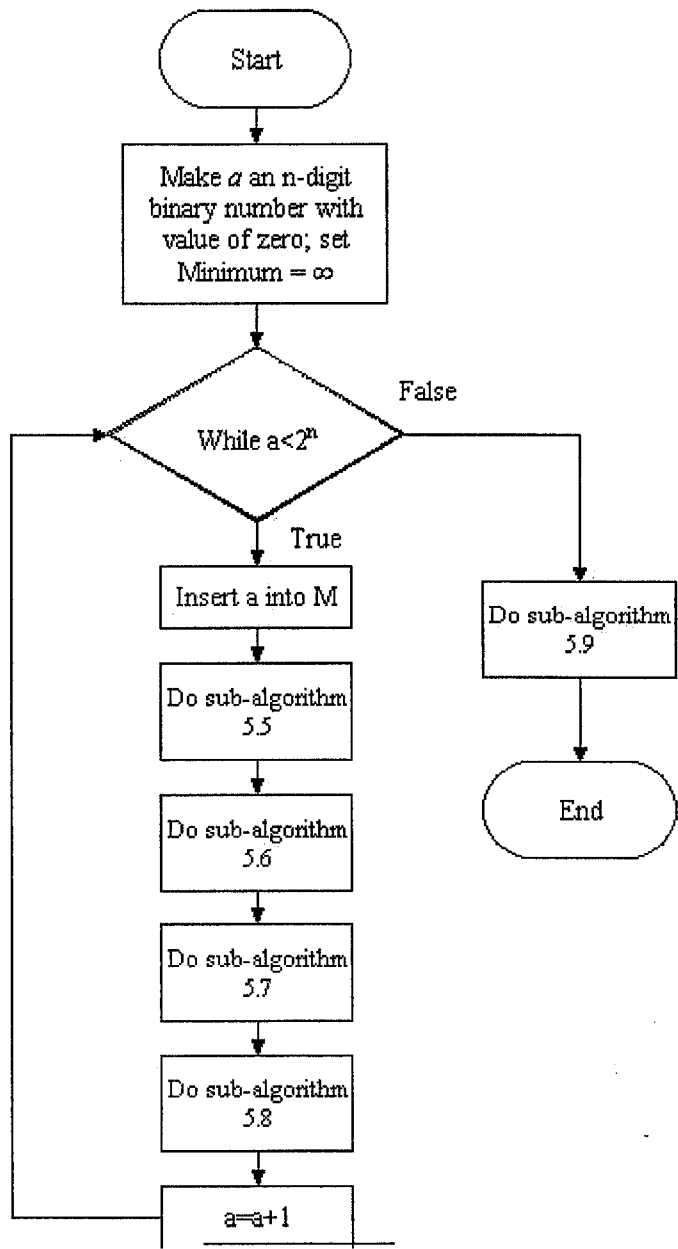


Figure 5.6: Flowchart for Sub-Algorithm 5.4

To recapitulate, Sub-Algorithm 5.4 performs the following:

SUB-ALGORITHM 5.4: Produce the minimum STTT digraph

1. Set the n-digit binary number, a, to zero.

2. Set Minimum to ∞ (The Minimum variable is discussed in Sub-Algorithm 5.8).
3. While $a < 2^n$
4. Insert a into matrix M .
5. Do Sub-Algorithm 5.5: Forming the digraph using M .
6. Do Sub-Algorithm 5.6: Checking indegree and outdegree of digraph.
7. Do Sub-Algorithm 5.7: modified Dijkstra's algorithm.
8. Do Sub-Algorithm 5.8: Finding the digraph with the minimum shortest total time.
9. Do Sub-Algorithm 5.9: Print results.

Sub-Algorithm 5.5: Forming the digraph using M

This sub-algorithm will generate a digraph with the conversion matrix, M , given. The input is M and its output is the adjacency matrix of the digraph that is represented by the binary number in M .

As an adjacency matrix of a digraph will be produced by this sub-algorithm, it will first allocate an $(n \times n)$ null matrix, q , to be "filled in" as the adjacency matrix of the digraph.

In order to generate the digraph, the sub-algorithm scans M . This is done by using a loop with index i . The value of i is first set to 1. As M has $(\text{max}-1)$ filled rows (this is so since max is always set to the next empty row), the loop will iterate from $i=1$ to $(\text{max}-1)$.

By going through M row by row, the sub-algorithm decides the direction for each of the edges represented by the rows (as explained in Section 4.3). Let x take the value in $M[i][1]$ and y take the value in $M[i][2]$. If the value in $M[i][3]=0$, the sub-algorithm would go to $q[x][y]$ and insert the weight of the edge (that is value of $M[i][4]$). If value in $M[i][3]=1$, the sub-algorithm would insert the value of $M[i][4]$ into $q[y][x]$ instead.

With that, Sub-Algorithm 5.5 is formed. Figure 5.7 shows the detailed flowchart for Sub-Algorithm 5.5. As shown in Figure 5.7, Sub-Algorithm 5.5 will first search through M , and depending on the value of $M[i][3]$, the direction of the edge is decided and the weight

of the edge placed into q , adjacency matrix of the digraph in the appropriate location. The pseudo-code of the sub-algorithm is given as follows:

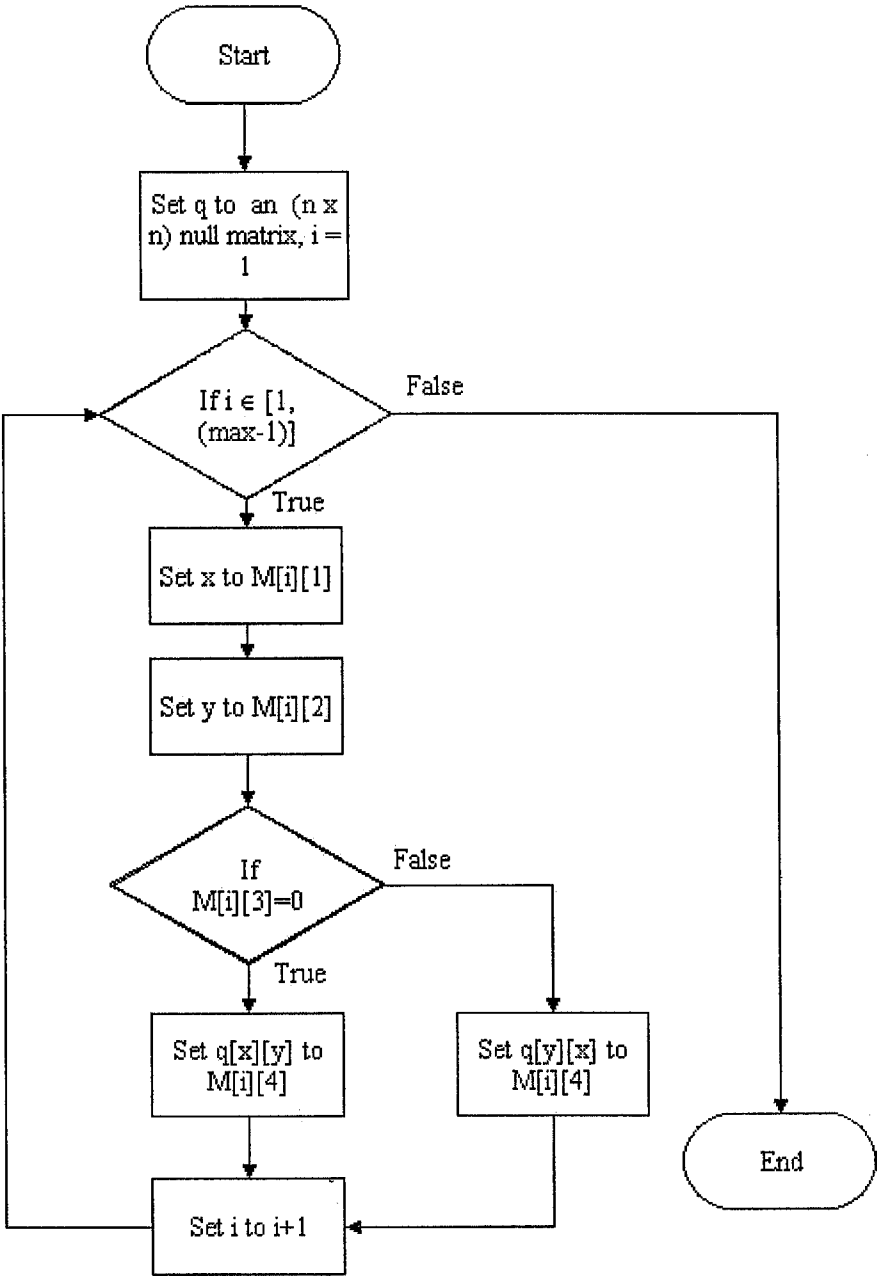


Figure 5.7: Flowchart for Sub-Algorithm 5.5

SUB-ALGORITHM 5.5: Forming the digraph using M

- 1. Initialize an $(n \times n)$ null matrix, q .
- 2. Set i to 1.
- 3. If $i \in [1, (\max-1)]$

4. Set x to $M[i][1]$.
5. Set y to $M[i][2]$.
6. If $M[i][3]$ equals 0
7. $q[x][y] = M[i][4]$.
8. If $M[i][3]$ equals 1
9. $q[y][x] = M[i][4]$.
10. Set i to $i+1$.

Sub-Algorithm 5.6: Checking indegree/outdegree of digraph nodes

The sub-algorithm checks the indegree and outdegree of the digraph produced by Sub-Algorithm 5.5. The input is q , the adjacency matrix of the digraph. Its output is a Boolean parameter, *Reject*, which indicates whether the digraph should be rejected (based on its indegree or outdegree).

Using the adjacency matrix, q , the value for the indegree of each node of the digraph can be obtained by summing up all values down each column of q (see Section 2.2.1.4). Similarly, the value for the outdegree of each node of the digraph can be obtained by summing up all values across each row of q (see Section 2.2.1.4). Let the sum in each case be z . If z (which denotes the indegree or outdegree of each node of the digraph) equals zero, then the digraph would be rejected. Boolean parameter, *Reject*, is used to indicate whether the digraph should be rejected. It will be set to False (meaning “Not to reject”) at the start of the algorithm.

By using two loops (as shown in Sub-Algorithm 5.6) q may be examined row by row and hence the outdegree of each node in q computed. Similarly by using two loops, q may be examined column by column and the indegrees of all the nodes in q also obtained.

At any point of the examination if indegree or outdegree of just one node is zero, *Reject* would be set to True. This should terminate the examination at the end of that loop, since it is pointless examining the rest of q . To do this, “*Reject* being False” can be set to be one of the criteria of each loop (in addition to incrementing the index i or j). Once this criterion is not met (that is once *Reject* is not False) Sub-Algorithm 5.6 terminates.

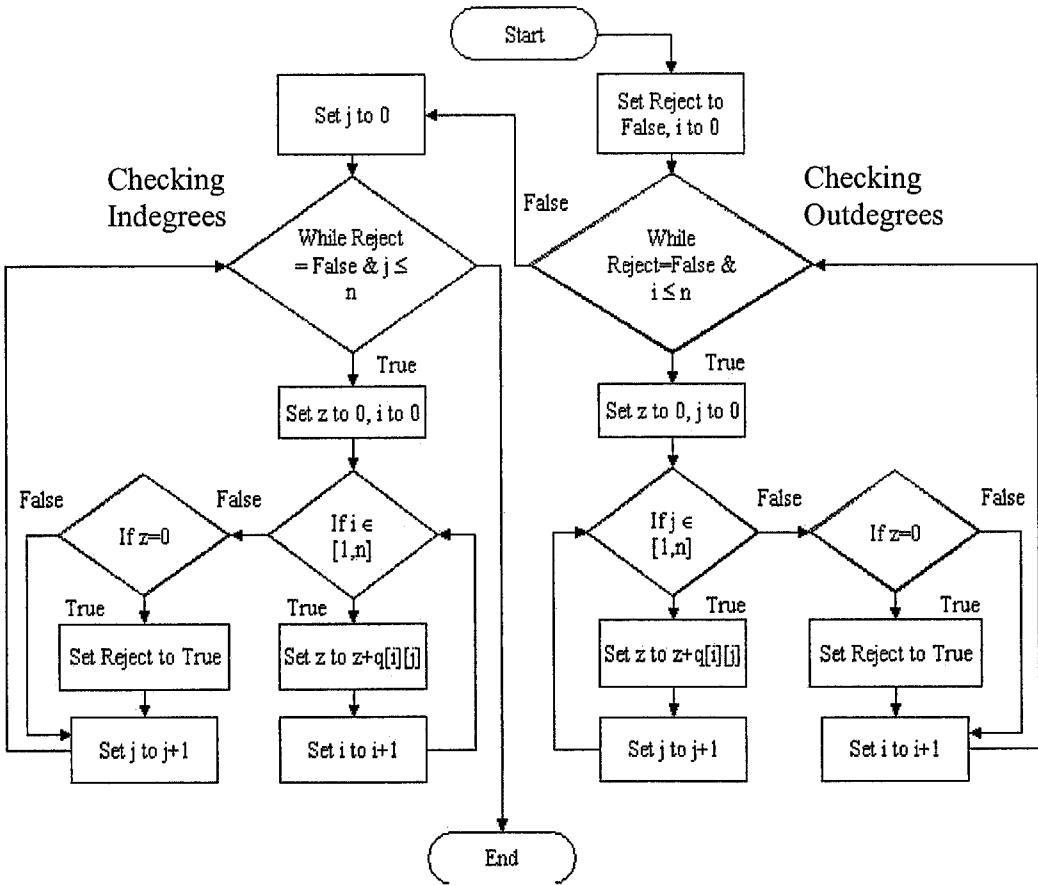


Figure 5.8: Flowchart for Sub-Algorithm 5.6

With that, Sub-Algorithm 5.6 is formed. Figure 5.8 shows the flowchart for Sub-Algorithm 5.6. As shown in Figure 5.8, Sub-Algorithm 5.6 will first create two loops to go through the rows of q . It will sum up all entries in each row and place this sum into z . Should z be zero this implies that the outdegree of that node is zero. Reject will be set to “True”, this will terminate the sub-algorithm. If none of the outdegrees is zero, the indegrees would be examined next. Sub-Algorithm 5.6 will create another two loops to go through the columns of q . It will sum up all entries in each column and place this sum into z . Should z be equal to zero, this implies that the indegree of that node is zero. Reject will be set to “True”, this will terminate the sub-algorithm. The pseudo-code of the sub-algorithm is given as follows:

SUB-ALGORITHM 5.6: Checking indegree/outdegree of digraph nodes

1. Set Reject to “False” and i to 1.

2. While Reject is “False” and $i \in [1,n]$
3. Set z to 0 and j to 1.
4. If $j \in [1,n]$
5. $z = z + q[i][j]$.
6. Set j to $j+1$.
7. If $z = 0$
8. Set Reject to “True”.
9. Set i to $i+1$.
10. Set j to 1.
11. While Reject is “False” and $j \in [1,n]$
12. Set z to 0 and i to 1.
13. If $i \in [1,n]$
14. $z = z + q[i][j]$.
15. Set i to $i+1$.
16. If $z = 0$
17. Set Reject to “True”.
18. Set j to $j+1$.

Sub-Algorithm 5.7: Modified Dijkstra’s algorithm

This sub-algorithm uses a modified Dijkstra’s algorithm (see Section 4.3.3.3) to calculate the all-to-all shortest time matrix from the adjacency matrix, q , of the digraph. Inputs are q and the Boolean parameter, Reject (from Sub-Algorithm 5.6), with the all-to-all shortest time matrix, s , as the output.

The sub-algorithm by having Reject as its input first determines if the digraph has a non-zero indegree or outdegree. If Reject has a value of “True” (that is the digraph has zero indegree or outdegree) the sub-algorithm terminates immediately, as this indicates that there is at least one node in the digraph with no path leading towards it or out of it. The modified Dijkstra’s algorithm will not be able to determine the shortest travel time from any node toward that node or the shortest travel time from that node to any of the other nodes. However, should Reject be “False”, the modified Dijkstra’s algorithm finds the shortest travel time from all nodes to all other nodes and produces the result in s . As

Dijkstra's algorithm is a single-to-all shortest path algorithm, a separate index, i , is used to track that every node in the graph takes a turn to be the source node.

A separate matrix, *time*, is used to store all the current shortest travel time from the source node to another node, for example $time[j]$ will give the current, known shortest travel time from the source node i to the node j . This matrix is initialised to take values of infinity, meaning that all shortest known travel times from the source node to the rest of the nodes of the graph are first set to infinity. The shortest travel time from the source node to the source node (that is $time[i]$) is initialised to zero. This is so since the best way to get from node i to node i is not to move at all, hence time taken is zero second.

A sorted list, L , which will be used to store all the nodes linked to the source node by a path is initialised to zero. L is initialised to store the source node, i .

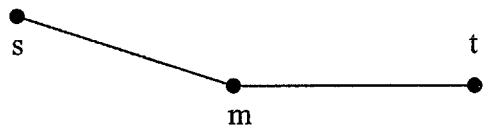


Figure 5.9: Path from node s (past node m) to node t

A separate matrix, *prev*, is used to store all the previous nodes leading from the source node to the final destination node. For example a given path as shown in Figure 5.9 with s as the source node and t as the destination node has m as the previous node of t leading from s to t ; $prev[t]$ will thus give m in this case. (This is used by the original Dijkstra's algorithm to enable the user to find, in addition to the value of the shortest path, the nodes making up the shortest path).

The first member of L is set to be the current node, m . Just like the original Dijkstra's algorithm (see Section 4.3.3.3), the modified Dijkstra's algorithm will compare the current shortest travel time, T_t , from the source node, i , to destination node, t , with the sum of the current shortest travel time from s to m and m to t ($T_m + T_{mt}$).

However the modified algorithm will compare T_t with $(T_m + T_{mt} + \textit{turn})$ instead, where *turn* is the time spent in negotiating the intersection at node m , which can be located by finding out what kind of turn exist from i (past m) to t . According to Crauser *et al.* [99], m will always be adjacent to t . In order to determine the type of turn made, the previous node of m (on the path from i to m) needs to be found. (The need for information on the previous node is explained in Appendix D). This previous node of m can be found in the matrix, *prev*, which records all the previous nodes of node m . According to Cherkassky *et al.* [87], all possible nodes to be used as m will already have an existing shortest path from i to m (which is recorded in matrix, *time*).

Values of *turn* can then be obtained by accessing the intersection time matrix, B , at location $B[i][m][t]$. This will cause *turn* to take the value of T_1 (= 16.73s), T_2 (= 11.0s) or T_3 (= 13.12s) (see Section 4.3.3.2 for more information).

Value of T_t and T_m may be obtained from the matrix, *time*. T_{mt} may be calculated by taking distance, d , from m to t from the adjacency matrix and dividing d by the average speed of the vehicle, AveSpeed (which is 50km/h in this case).

The modified Dijkstra's algorithm will then compare T_t with the sum of $(T_m + T_{mt} + \textit{turn})$. Should the latter be smaller than T_t then T_t will take the new value of $(T_m + T_{mt} + \textit{turn})$. Node m will be allocated as *prev*[t] to denote that the shortest time travelled from i to t now goes through the new path past m . Node t will also be added to the list, L ; node m will be removed from L . L will then be sorted in ascending order such that the node with the least shortest travel time from s will be placed in the first position (readers are encouraged to look up on relevant literature [87, 99] for more details on Dijkstra's algorithm).

The algorithm will then return to check for the first member of L and label it as the current node, m . The whole process will then be repeated until there comes a point when L is empty. At that point, all values of matrix, *time*, would have been placed into the i^{th} row of the all-to-all shortest time matrix, s .

The value of i is then incremented and the next node used as the source node. Sub-Algorithm 5.7 then iterates to calculate the shortest travel time from the new source to all other nodes of the graph. This will eventually fill s up, a row at a time, to provide all the STTTs from all nodes to all nodes.

With that, Sub-Algorithm 5.7 is formed. Figure 5.10 shows the flowchart of Sub-Algorithm 5.7. As shown in Figure 5.10, Sub-Algorithm 5.7 first examines the value of *Reject*: if *Reject* is “True” the sub-algorithm would terminate immediately. On the other hand, if *Reject* is “False”, the modified Dijkstra’s algorithm would be used. A loop (with index i) is used to incrementally select each node of the network as the source node. With the source node, i , the selected matrix, *time*, is initialised to take infinity for all values; i is placed into L and $time[i]$ is set to zero (to denote shortest time from node i to node i is zero second).

While L remains non-empty, the first element of L will be set to be current node, m . An index j is used to reference the adjacency matrix, q , at the m^{th} row; this gives the nodes that are adjacent to m . These adjacent nodes are consecutively set to be destination node, t , one after the other (by using index j). The value of *turn* is obtained by accessing B at $B[i][n][t]$. The sub-algorithm then compares T_t (which is $time[t]$) with the sum of T_m (which is $time[m]$), T_{mt} (which is the distance from m to t , $q[m][t]$, divided by *AveSpeed*) and *turn*. If T_t is larger, then it will be replaced by $(T_m + T_{mt} + turn)$, $prev[t]$ will be set to m and t will be added to L . The current node, m , is removed from L and L is sorted in ascending order.

When L eventually becomes empty, all values of matrix, *time*, are transposed to the i^{th} row of s ; i is then incremented and a new source node obtained. The process iterates itself until every node has been used as the source node and the shortest travel time from all nodes to all other nodes are obtained. The pseudo-code for the sub-algorithm is given as follows:

SUB-ALGORITHM 5.7: Modified Dijkstra’s algorithm

1. If *Reject* is “False”.

2. Set i to 1.
3. If $i \in [1, n]$
4. Set j to 1.
5. If $j \in [1, n]$
6. Set $time[j] = \infty$
7. Set j to $j+1$.
8. Set $time[i]$ to 0.
9. Set $prev[i]$ to 0.
10. Insert i into L .
11. While L is non-empty
12. Set m to value of $L[1]$.
13. Set j to 1.
14. If $j \in [1, n]$
15. If $q[m][j] \neq 0$
16. Set t to be j .
17. Set $turn$ to be $B[i][m][t]$.
18. If $time[t] > [time[m] + (q[m][t] \div AveSpeed) + turn]$
19. Set $time[t]$ to $[time[m] + (q[m][t] \div AveSpeed) + turn]$.
20. Set $prev[t]$ to m .
21. Insert t into L .
22. Set j to $j+1$.
23. Remove m from L .
24. Sort L in ascending order.
25. Set j to 1.
26. If $j \in [1, n]$
27. Set $s[i][j]$ to $time[j]$.
28. Set j to $j+1$.
29. Set i to $i+1$.

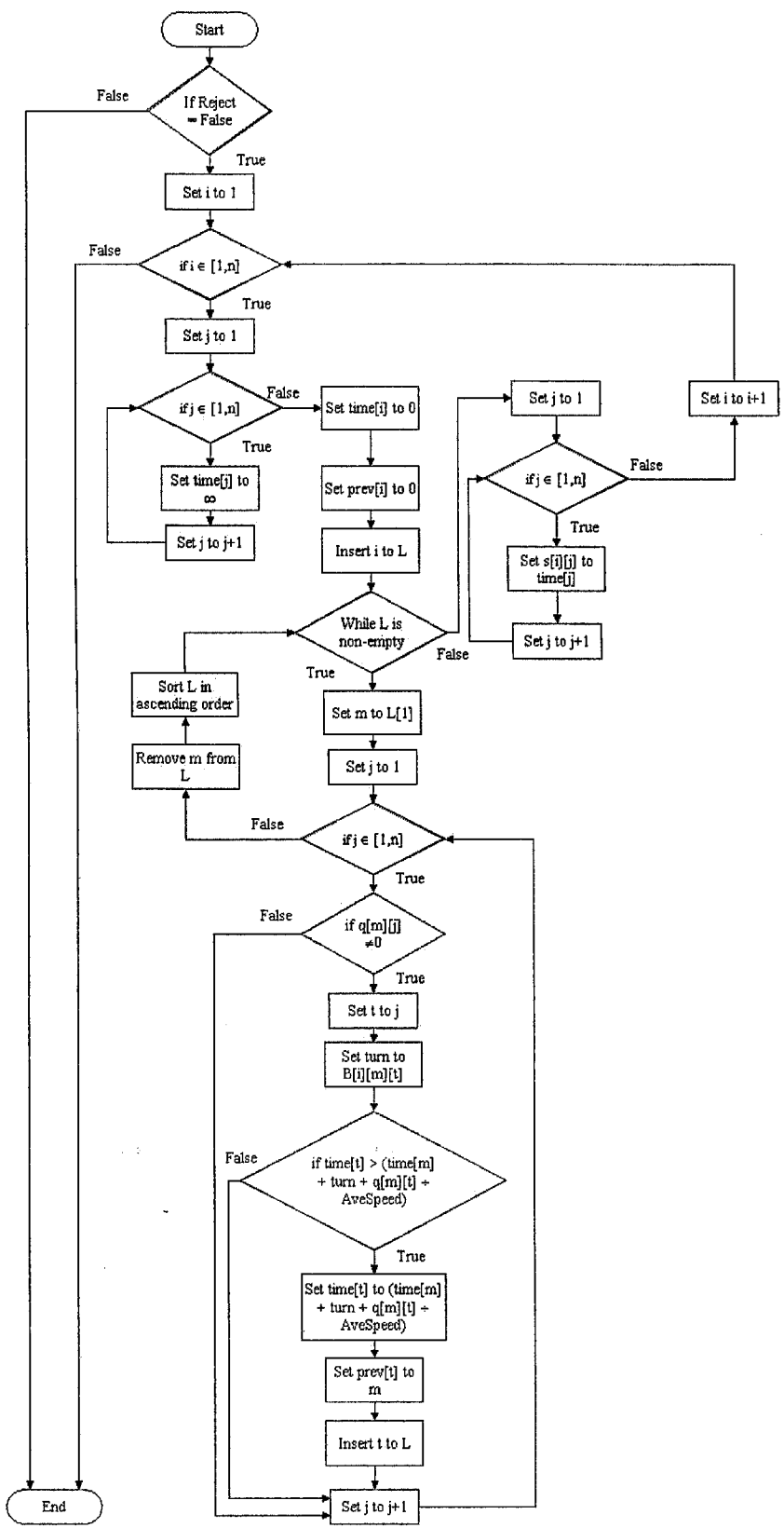


Figure 5.10: Flowchart for Sub-Algorithm 5.7

Sub-Algorithm 5.8: Finding the digraph with the minimum STTT

This sub-algorithm calculates the STTT of the digraph and compares it with the current minimum STTT, Minimum. The lesser of the two will take the place of Minimum. Inputs to this sub-algorithm are q (the adjacency matrix of the digraph), s (the all-to-all shortest path matrix obtained from Sub-Algorithm 5.7) and Minimum from Sub-Algorithm 5.4 (as explained in the later part of this section). Output from this sub-algorithm are r , the adjacency matrix of the digraph with the minimum STTT, as well as Minimum.

In order to compare the STTT of the current digraph with the best obtained so far, the STTT of each digraph is first calculated. This is done using two loops to go through s while summing up all values within s . Let the sum of these values be called *sum*. *Sum* will be compared with the current minimum STTT, Minimum, and the lesser of the two will replace Minimum. The adjacency matrix, q , of the “minimum” digraph will be stored in another $(n \times n)$ matrix r , that is r holds the adjacency matrix of the digraph currently known to have the minimum STTT.

Minimum is initially given a large numerical value (called infinity), large enough to ensure that it will always be replaced no matter how great the shortest total time value of the first digraph is. Minimum has to be initialised before this sub-algorithm is called. A good place to initialise it would be in the sub-algorithm, Sub-Algorithm 5.4, that calls on Sub-Algorithm 5.8. Hence Minimum is initialised to infinity in Sub-Algorithm 5.4 and inserted into this sub-algorithm as one of its inputs.

If both *sum* and Minimum have the same value then q would be stored into another matrix next to r , that is $r+k$ in this case. The value of k is set at 1 initially and will be incremented by a value of 1 each time *sum* equals to Minimum. However, each time Minimum is replaced by *sum*, k will be set back to 1 again. This is to ensure that digraph(s) with shortest total time equivalent to previous minimum shortest total time may be overwritten by alternative digraphs with the better minimum STTT.

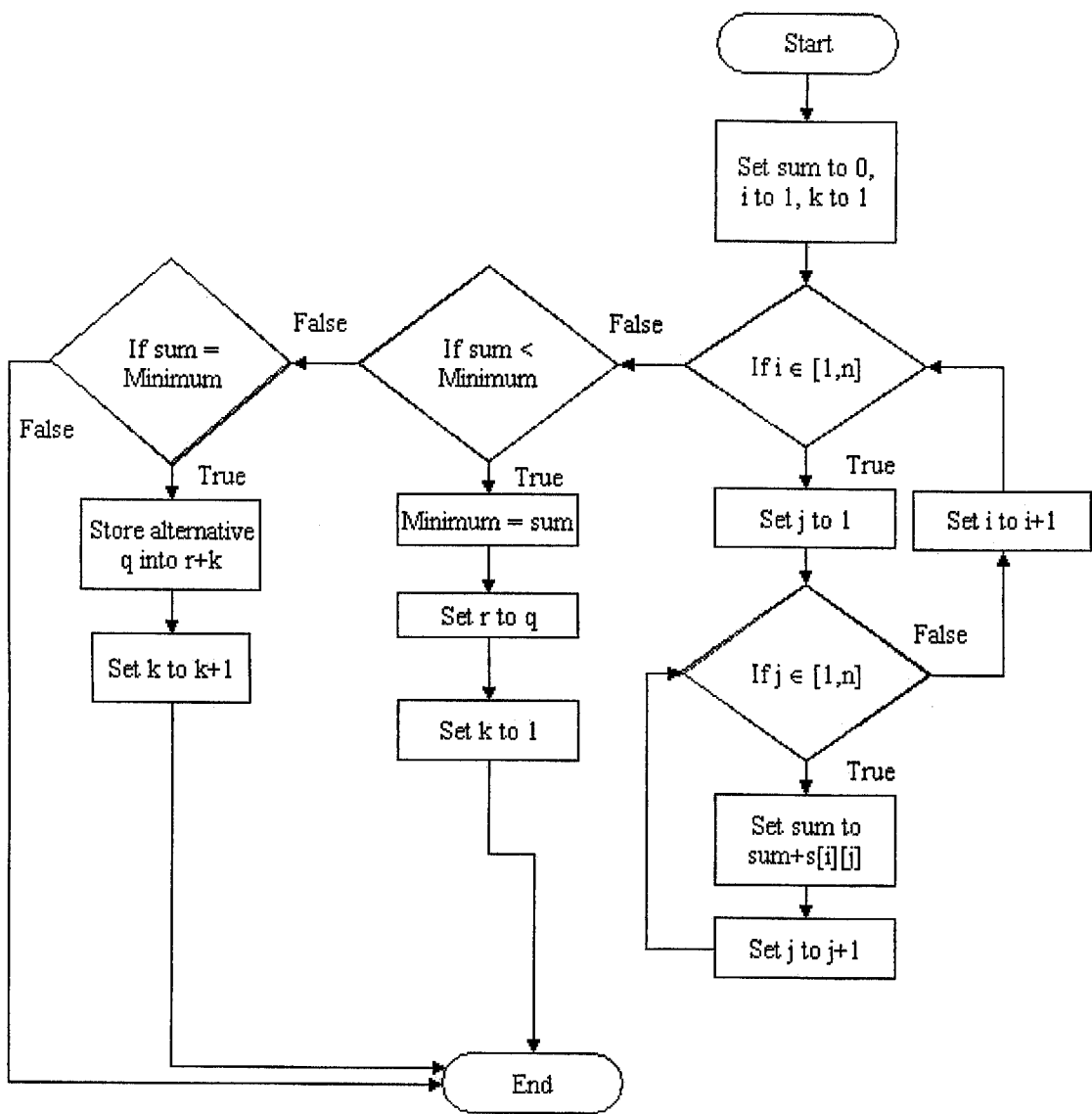


Figure 5.11: Flowchart for Sub-Algorithm 5.8

With that, Sub-Algorithm 5.8 is formed. Figure -5.11 shows the flowchart for Sub-Algorithm 5.8. As shown in Figure 5.11, Sub-Algorithm 5.8 first sets i and k to one and sum to zero. It then computes STTT from s and puts that value into sum . It subsequently compares sum with Minimum and may replace Minimum with a lesser value, in which case k will be reset to zero. The first digraph with the lower STTT is then stored into r . However if both sum and Minimum are equal it will store the alternative digraph into the matrix $r+k$ instead. The pseudo-code of the sub-algorithm is given as follows:

SUB-ALGORITHM 5.8: Finding the digraph with the minimum STTT

1. Set *sum* to 0, *i* to 1 and *k* to 1.
2. If $i \in [1, n]$
3. Set *j* to 1.
4. If $j \in [1, n]$
5. Set *sum* to $sum + q[i][j]$.
6. Set *j* to *j*+1.
7. Set *i* to *i*+1.
8. Compare *sum* with Minimum
9. If $sum < \text{Minimum}$
10. Set Minimum to *sum*.
11. Set Matrix *r* to be Matrix *q*.
12. Set *k* to 1.
13. Else
14. If $sum = \text{Minimum}$
15. Set Matrix *r*+*k* to be Matrix *q*.
16. Set *k* to *k*+1.

Sub-Algorithm 5.9: Print results

This sub-algorithm prints the adjacency matrix of the digraph with minimum STTT, as well as the value of the minimum STTT as output to the user. The inputs to this sub-algorithm are *r*, *k* and Minimum while the outputs are the printed results.

It first prints out *r* as output to the user, followed by printing the value of Minimum. It then checks for value of *k*. Should the value of *k* be more than 1, this indicates that there are more than one digraph with the minimum STTT. It will subsequently print out all digraphs present.

With that, Sub-Algorithm 5.9 is formed. Figure 5.12 shows the flowchart for Sub-Algorithm 5.9. The pseudo-code for the sub-algorithm is given as follows:

SUB-ALGORITHM 5.9: Print results

- 1. Print out Matrix r as the adjacency matrix of the digraph with the minimum STTT.
- 2. Print out Minimum as the value of the minimum STTT of the digraph.
- 3. If $k > 1$
- 4. Set i to 1.
- 5. If $i \in [1, (k-1)]$
- 6. Print $r+i$.
- 7. Set i to $i+1$.

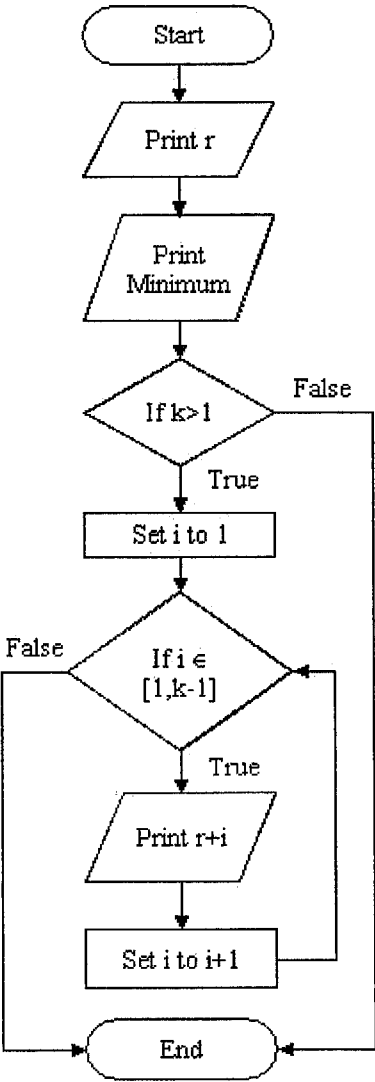


Figure 5.12: Flowchart for Sub-Algorithm 5.9

Having defined the conversion algorithm, the conversion algorithm was subsequently verified to ensure its correctness. This will be discussed next with the aid of a simple graph example.

5.3 Verification

The conversion algorithm will calculate and compare all STTT for feasible digraphs. It will then gives the digraph (or digraphs) with the minimum STTT. This can be verified with a very simple graph example, as shown in Figure 5.13(a). To recapitulate, the major steps of the conversion algorithm are as follows:

- i. Obtain all strongly connected digraphs by enumeration.
- ii. Calculate the STTT for all strongly connected digraphs.
- iii. Search for the digraph with the minimum STTT.
- iv. Verify that the digraph obtained in step (iii) indeed is the digraph with the minimum STTT.

Since verification requires enumeration of all strongly connected digraphs, only a simple graph will be used. The simple graph example with all edges of unit weight (of one metre), as shown in Figure 5.13(a) is used for this verification process.

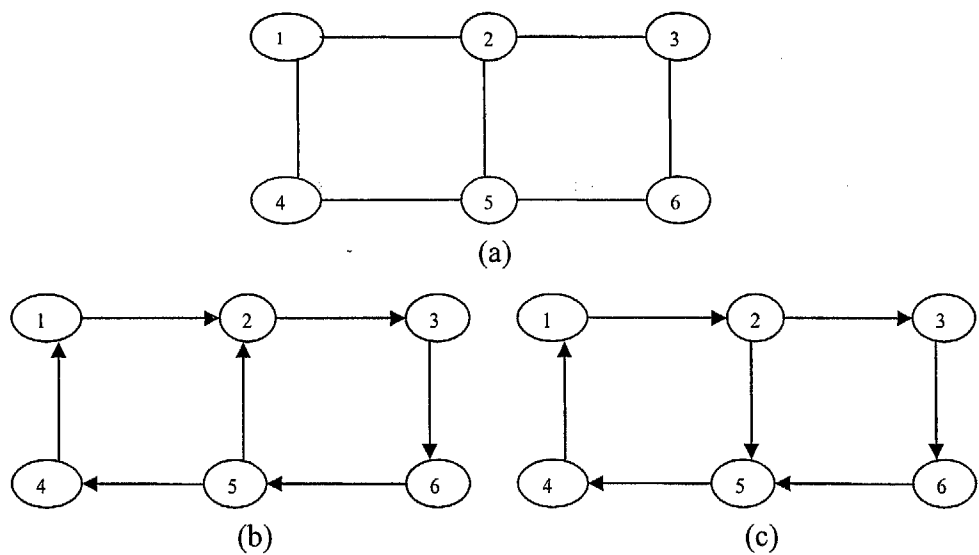


Figure 5.13: A simple graph example

With the given undirected graph, the conversion algorithm produces 2 digraphs as shown in Figures 5.13(b) and 5.13(c). (Actual results of the conversion algorithm are given in Appendix E). These digraphs obtained both have the same STTT value of 607.15 seconds. [Upon closer inspection however it is noted that both graphs are isomorphs of each other (see Section 9.4.1.4) - rotating Figure 5.13(b) clockwise by 180° will give Figure 5.13(c)].

Following that, all possible strongly connected digraphs are manually produced to calculate their STTTs for the purpose of comparison. As the undirected graph used has 7 edges, it generated 2⁷ possible digraphs. All the 128 digraphs produced (given in Appendix E) are checked for strong connectivity by means of simple calculation of indegrees and outdegrees. Only 6 out of the 128 digraphs were strongly connected (see Table E.1 in Appendix E).

00.00	00.07	11.14	50.72	37.53	24.34
50.72	00.00	00.07	37.53	26.24	13.26
37.53	26.24	00.00	24.34	13.26	00.07
00.07	13.26	24.34	00.00	50.72	37.53
13.26	00.07	13.26	00.07	00.00	26.24
24.34	13.26	26.24	11.14	00.07	00.00

(a)

00.00	00.07	11.14	26.46	13.26	24.34
26.24	00.00	00.07	13.26	00.07	13.26
37.53	50.72	00.00	24.34	13.26	00.07
00.07	13.26	24.34	00.00	26.24	37.53
13.26	26.24	37.53	00.07	00.00	50.72
24.34	37.53	50.72	11.14	00.07	00.00

(b)

Figure 5.14: All-to-all shortest time matrix

Applying the modified Dijkstra’s algorithm (see Sub-Algorithm 5.7) on these 6 strongly connected digraphs, their all-to-all shortest time matrices were produced. By summing up all the values in the all-to-all shortest path matrices, the STTTs were obtained and compared. The minimum STTT was given as 607.15 seconds and 2 digraphs were obtained and given in Figures 5.13(b) and 5.13(c) with their all-to-all shortest time matrix

given in Figures 5.14(a) and 5.14(b) respectively. This is the same as the result given by the conversion algorithm, thereby verifying the conversion algorithm.

With the verification of the conversion algorithm, the WIJ road network in Figure 5.15 (refer to Section 3.5 for detailed description of the problem) will be used to demonstrate its practical usefulness in a real world problem of a one-way road network problem.

5.4 Case study

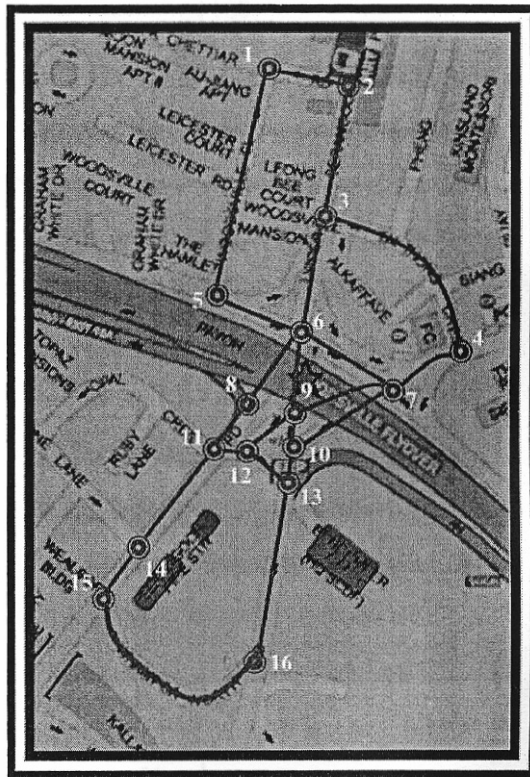


Figure 5.15: WIJ Network

In the WIJ network case study as in any network problem, the first step involves representing the given road network as a graph. The various intersections of the WIJ network are numbered as shown in Figure 5.15. These numbers will correspond to the numbered nodes in the graph to be obtained. Letting the intersections be the nodes of the graph, the roads be the edges of the graph and the distances of the roads (in metres) be the

weights of the edges, an undirected graph representing the WIJ network is obtained (as shown in Figure 5.16).

As the graph is used in the conversion algorithm in its adjacency matrix form, the adjacency matrix of the WIJ road network is formulated in Table 5.1.

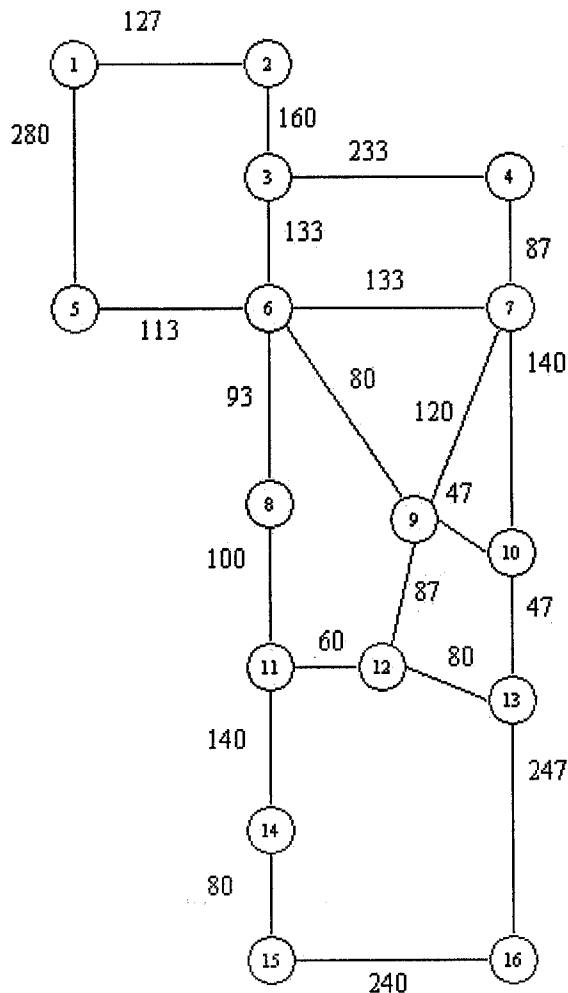


Figure 5.16: WIJ network in the form of a weighted graph

With the weighted adjacency matrix of the graph, the graph is ready to be manipulated by the conversion algorithm. Table 5.1 will be used as an input to the conversion algorithm. Applying the conversion algorithm on the undirected WIJ network, the digraph (see Table 5.2 for adjacency matrix) with the minimum STTT (which is 18823 seconds in this case)

is obtained. From the adjacency matrix given in Table 5.2, the digraph obtained is drawn and shown in Figure 5.17.

Table 5.1: Weighted adjacency matrix of Figure 5.16

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	127	0	0	280	0	0	0	0	0	0	0	0	0	0	0
2	127	0	160	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	160	0	233	0	133	0	0	0	0	0	0	0	0	0	0
4	0	0	233	0	0	0	86	0	0	0	0	0	0	0	0	0
5	280	0	0	0	0	113	0	0	0	0	0	0	0	0	0	0
6	0	0	133	0	113	0	133	93	80	0	0	0	0	0	0	0
7	0	0	0	87	0	133	0	0	120	140	0	0	0	0	0	0
8	0	0	0	0	0	93	0	0	0	0	100	0	0	0	0	0
9	0	0	0	0	0	80	120	0	0	47	0	87	0	0	0	0
10	0	0	0	0	0	0	140	0	47	0	0	0	47	0	0	0
11	0	0	0	0	0	0	0	100	0	0	0	60	0	140	0	0
12	0	0	0	0	0	0	0	0	87	0	60	0	80	0	0	0
13	0	0	0	0	0	0	0	0	0	47	0	80	0	0	0	247
14	0	0	0	0	0	0	0	0	0	0	140	0	0	0	80	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	240
16	0	0	0	0	0	0	0	0	0	0	0	0	247	0	240	0

Table 5.2: Weighted adjacency matrix of the digraph with the minimum STTT

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	280	0	0	0	0	0	0	0	0	0	0	0
2	127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	160	0	233	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	87	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	113	0	0	0	0	0	0	0	0	0	0
6	0	0	133	0	0	0	0	93	80	0	0	0	0	0	0	0
7	0	0	0	0	0	133	0	0	0	140	0	0	0	0	0	0
8	0	0	0	0	0	93	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	120	0	0	47	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0
11	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	87	0	60	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	247
14	0	0	0	0	0	0	0	0	0	0	140	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	240	0

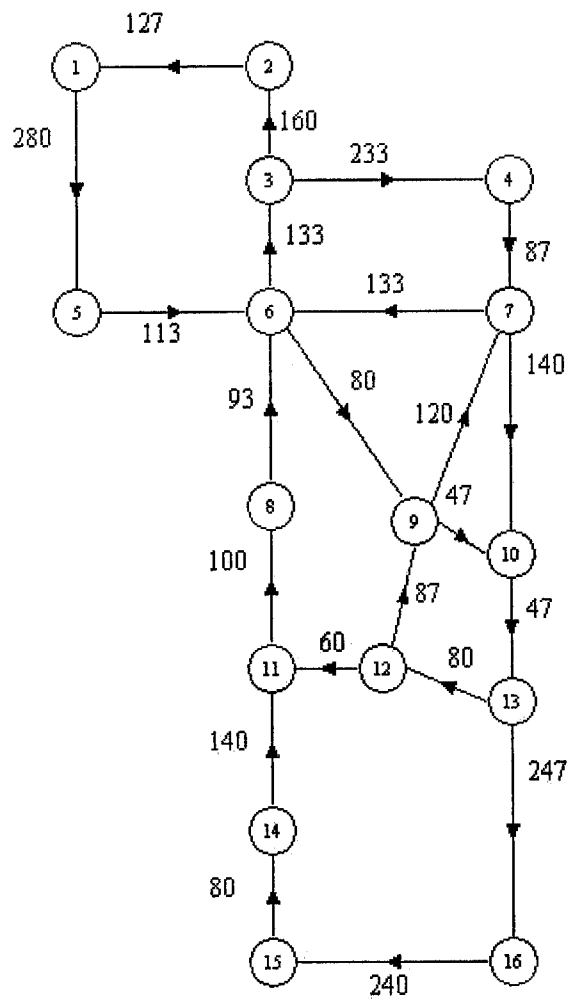


Figure 5.17: Digraph of WIJ network obtained

The initial layout of the one-way road network of WIJ is thus obtained. The case study is now ready to be tested in the next major step of the one-way urban road network model using the minimum-cut (min-cut) graph model. Therefore, the min-cut graph model will be discussed in detail in the next chapter.

VI MINIMUM-CUT GRAPH MODEL

6.1 Overview

In the event of traffic congestion, as stated by Ortúzar and Willumsen [1] due to the interactivity of vehicles the time taken to traverse the road network may be more than when the road network is not congested. The initial one-way road layout as obtained from the conversion algorithm in Chapter 5 has no traffic loaded. This implies that when actual traffic is loaded it may not be as efficient in reducing delay as it is when there is no traffic congestion. The minimum-cut (min-cut) graph model by Cheng *et al.* [13] is thus used to further refine the one-way road network layout obtained from the conversion algorithm when actual traffic is loaded into the road network.

This chapter describes in detail the use of a min-cut graph model to make further improvements on the initial one-way road network layout obtained using the conversion algorithm described in Chapter 5 when actual traffic load is taken into consideration. First, the formulation of the min-cut graph model for a road network is looked into. Following that an explanation of the min-cut graph model will be given. Finally, the chapter ends with a demonstration of the min-cut graph model, as applied to the Woodsville Interchange Junction (WIJ) road network.

6.2 Formulation

The min-cut graph model was first applied to the Module Orientation Problem by Cheng *et al.* [13] for the design of PCB (greater details at Section 2.2.3.2). In order to apply the min-cut graph model onto the one-way road network problem, the min-cut graph model needs to be further examined.

In using the min-cut graph model on the Module Orientation Problem, the PCB is represented by a single weighted graph $G(V, E)$. V is the node set, E is the edge set while weight of the edge (or edge capacity) is given by c_{ij} . V contains a total of $n+1$ nodes, where n nodes represent all the modules that can be flipped (that is reversed in

orientation) and the single other node represents a super node T . T is a collection of all the modules that, for varying reasons given by Libeskind-Hadas and Liu [26], cannot be flipped. These reasons include:

- i. The need for certain modules to be used as a common bridge between layers of PCB hence the module's positions need to be fixed.
- ii. The need for certain polarized parts (that is diodes and electrolytic caps) with positive leads to have the same orientation.
- iii. The need for certain modules (including connectors, switches, LEDs, mounting holes, heat sinks or any other item that mounts to an external location) to be in specific positions/orientations.

According to Cheng *et al.* [13], the min-cut graph model states that should a cut exist to partition the graph into two sub-graphs, the capacity of this cut would be equivalent to the change in total net length (of the PCB) caused by flipping the modules in one of the two sub-graphs. As one of the sub-graphs contains T (modules not to be flipped) the modules to be flipped are those in the sub-graph without T . Therefore, using a minimum cut ensures that the change in the total net length is minimised. The modules to be flipped are then chosen, giving rise to an optimum orientation of the modules.

Similarly in applying the min-cut graph model to the one-way road network problem, the road network could be represented by a single weighted graph $G(V, E)$. V is the set of nodes representing the roads of the network in G . E is the set of edges of G , edge capacity of each edge (i, j) is given by c_{ij} . (Note: in this model, nodes rather than edges, represent roads)

To prove that the min-cut graph model can be use for the one-way road network problem, is to prove that a cut across the graph will be such that the capacity of the cut would be equivalent to the net change in delay (of the one-way road network). A

minimum cut would be used to reduce the net delay in the one-way road network, thereby producing the one-way road network that is theoretically the most efficient.

We now attempt to prove that the capacity of the cut is equivalent to the change in net delay of the one-way road network, by proposing a hypothesis, which can be stated as follows:-

Hypothesis A:

Given a one-way urban road network represented by a graph $G(V,E)$ as described above, a cut $(U, V-U)$ that separates subset U from $V-U$, its complement (that is $\bar{U} = V-U$), with the super node T belonging to $V-U$, is such that capacity of the cut $(U, V-U)$ takes the value of C if and only if the change in overall net delay in the road network caused by reversing the flow direction of the one-way roads in U is equivalent to C .

(For easy reference, the above hypothesis shall be referred to as *Hypothesis A* throughout the rest of this thesis unless otherwise specified.)

To prove that *Hypothesis A* is correct this would mean that the cut $(U, V-U)$ needs to be proven to be the minimum cut of the graph, since this would mean that change in net delay caused by changing the direction of the roads in U would be at a minimum for the given min-cut graph model.

However, in order to form the min-cut graph model for the road network, first super node T has to be determined. Secondly, the edge capacities (which are needed for the capacity of the cut) of the road network needs to be obtained as well.

Super node T will be looked into first, followed by the edge capacities and eventually the min-cut graph model.

6.2.1 Super node T

T in the module orientation problem represents the collection of modules not to be flipped. Similarly, in the one-way road network problem it would represent the collection of roads not to be flipped (that is have their traffic flow directions reversed). Therefore a criterion for choosing the roads not to be flipped is required.

Given any one-way road network, a reason for not flipping a road would be to ensure strong connectivity (as explained in Section 3.3.4). Should flipping a road isolate a certain part of the road network, this road would not be considered for flipping and would be placed into T . Therefore, the roads are individually checked for loss of strong connectivity after flipping before being placed into T . To illustrate this, once again the case study of WIJ is used.

6.2.1.1 Case Study

The idealised network

Before choosing the roads for T , the WIJ network needs to be idealised. (An idealised road network refers to one in which only the major roads are taken into consideration while the minor roads are ignored.) This is because traffic of some intensity will be loaded into the network to determine the delay. According to Ortúzar and Willumsen [1], under such circumstances delay on minor roads are likely to be insignificant and need not be taken into consideration, since efficiency of the road network is mostly defined by the major roads.

The initial layout of the one-way road network of WIJ (see Figure 5.17) is used for idealisation. For comparison purposes, Figure 5.17 is reproduced in this section. Minor roads between nodes 6 and 9 as well as between nodes 9 and 10 are not considered and are shown only as dotted lines in the idealised network, as shown in Figure 6.1.

In addition to that, a series of connecting edges representing a whole stretch of continuous road are merged to form a single edge. For example, edge (7, 10) and edge (10, 13) are merged to form a single edge K (see Figure 6.1). This is justifiable, as in terms of the actual road network, edge (7, 10) and (10, 13) would be a continuous edge; in flipping roads to define T it would be impractical to flip just the edge (7, 10) and not edge (10, 13). The idealised network is thus formed and shown in Figure 6.1.

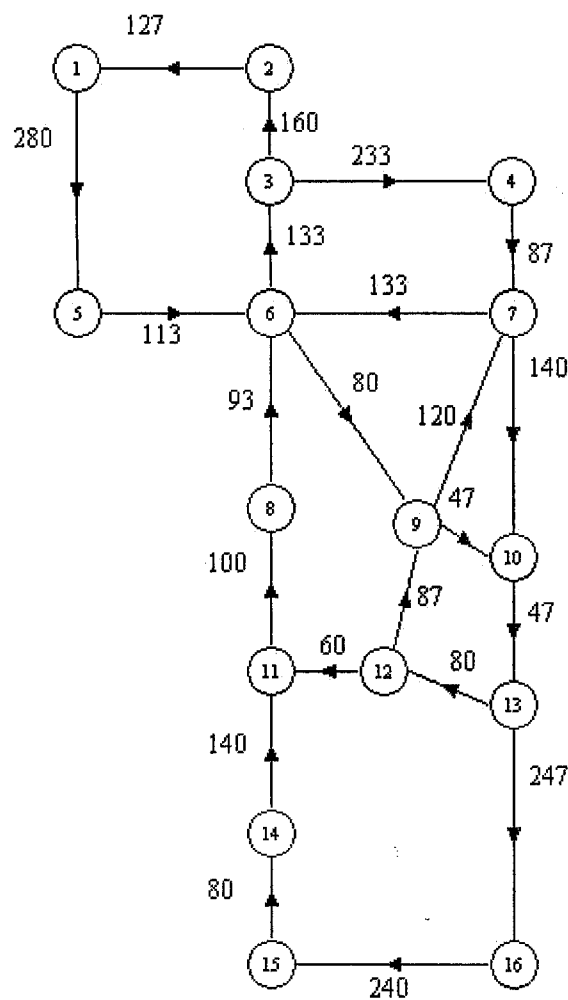


Figure 5.17: Digraph of WIJ network obtained (Reproduced)

In Figure 6.1, the edges are named A to Q while the numbers of the intersections/nodes remain unchanged. The arrows pointing to node 2, 5, 13, and 15 denote traffic entry points; the arrows pointing away from node 1, 8, 7 and 16 denote traffic exit points while the double edged arrow above node 4 denotes both an

entrance and exit point for the WIJ network. Continuous roads that stretch through nodes are shown: for example, *K* stretches from node 7 passes nodes 10 and finally to node 13. As for node 13, due to the presence of entrance point, this junction cannot be ignored; *K* is thus not merged with *P* or *N*. In addition, since the weights of the edges (length of the roads) do not play a significant role in the idealised network, they are omitted from Figure 6.1.

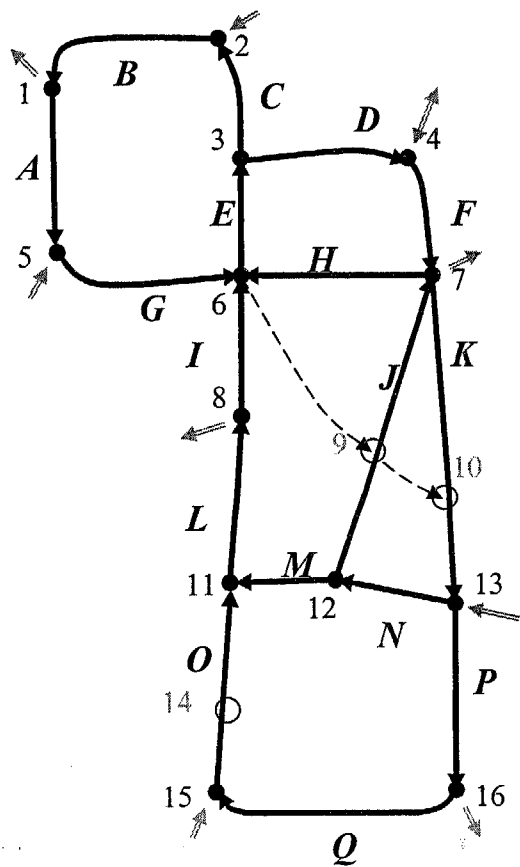


Figure 6.1: Idealised WIJ network

Checking for strong connectivity

With the network idealised it is then checked for existence of strong connectivity. All roads are flipped individually; the resultant networks obtained from the flippings of each road are checked to ensure that the network remains strongly connected. If strong connectivity of the network is removed after a flip, the flipped road is placed into *T*.

Taking A (from Figure 6.1) as an example, by flipping A , node 1 will become isolated, as there will be no road leading away from node 1, that is node 1's outdegree becomes zero. Thus, A will be placed in T . In the same way, the rest of the roads are individually flipped and checked for persistence of strong connectivity. Table 6.1 shows the results of individual flippings for each of the roads.

Table 6.1: Roads not to be flipped.

Road(s) flipped	Strongly Connected	Isolated node
A	No	1, 5
B	No	1, 2
C	No	2
D	No	4
E	No	3, 6
F	No	4
G	No	5
H	Yes	-
I	No	8
J	Yes	-
K	No	10
L	No	8, 11
M	Yes	-
N	No	12
O	No	14, 15
P	No	15, 16
Q	No	16

As shown in Table 6.1, only roads H , J and M are to be flipped. Hence, except for these three roads, all the rest of the roads will form T . It is noted that H and J are connected together at intersection 7 while J and M are connected at intersection 12. With T obtained, the edge capacity will be looked into next.

6.2.2 Edge capacity

To obtain all the edge capacities of $G(V,E)$, the triangle graphs developed by Cheng *et al.* [13] can be formulated and used just as they were in the Module Orientation Problem. In the Module Orientation Problem, each of the nets (which represents link lengths between each module) is used to formulate a triangle graph. In the one-way

road network problem the relevant links between the roads would be the delay at the intersections of the road network. Therefore, time delays at intersections will be used to formulate triangle graphs.

Using the same technique (as applied in the Module Orientation Problem), the triangle graph for each of the road intersections is formed as follows:

Consider an intersection s (within a one-way road network) which connects two one-way roads A and B . Let δ be the existing delay of the whole network. With the same traffic data, flipping one or both of the roads will cause a change in δ . By running the network on a simulation model, this change in δ can be obtained. We let these changes in delay be as follows:

- C1 be the additional delay when only A is flipped,
- C2 be the additional delay when only B is flipped,
- C3 be the additional delay when both A and B are flipped.

Since only A and B are considered in this case, the rest of the roads in the network are viewed as roads not to be flipped that is the rest of the roads will form the super node T . The whole road network can thus be represented as a graph with just three nodes: A , B and T . This graph is shown in Figure 6.2(a). Weights of the edges (or the edge capacities) are labelled as c_{AT} , c_{BT} and c_{AB} . (Due to the geometric shape of the graph, it is referred to as the triangle graph by Cheng *et. al.* [13]).

Recalling *Hypothesis A* (see Section 6.2) capacities of cuts of various parts of the graph would be the change in delay in the whole network when roads are flipped. Let:

- i. Capacity of the cut that partitions the triangle graph into $\{A\}$ and $\{B, T\}$ be C1.
- ii. Capacity of the cut that partitions the triangle graph into $\{B\}$ and $\{A, T\}$ be C2.
- iii. Capacity of the cut that partitions the triangle graph into $\{A, B\}$ and $\{T\}$ be C3.

This would give a graph as shown in Figure 6.2(b).

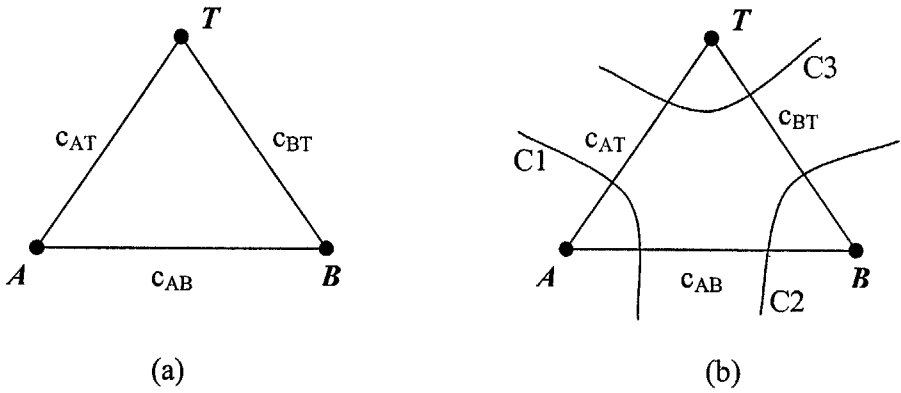


Figure 6.2: Triangle graph

Based on the definition of cuts (see Section 2.2.1.11) the following three equations are obtained:

$$\begin{aligned} c_{AT} + c_{AB} &= C1 \\ c_{BT} + c_{AB} &= C2 \\ c_{AT} \quad \quad \quad + \quad \quad \quad c_{BT} \quad \quad \quad &= \quad \quad \quad C3 \end{aligned} \tag{6.1}$$

Values for C1, C2 and C3 may be obtained by flipping each (or combinations) of the roads at this intersection and running the simulation model of the road network. By substituting the actual values of C1, C2 and C3 into Equation (6.1), the values for c_{AT} , c_{BT} and c_{AB} can be solved using simultaneous equations.

These edge capacities obtained will correspond to the edge capacities of the eventual min-cut graph model to be formed. However, consider an extension of this scenario whereby in addition to intersection s , road A is linked to intersection u . A similar triangle graph can be formed using intersection u and another set of value can be obtained for c_{AT} . Using the same reasoning from Cheng *et al.* [13], the edge capacity pertains to the road itself and not to the intersection. This means that flipping A , regardless of intersection s or intersection u , causes the change in delay in the whole

road network. Thus, edge capacity of A would be the arithmetic sum of both c_{AT}^S and c_{AT}^U [read as capacity of edge (A, T) at intersection u].

The WIJ network case study is given next to demonstrate the formulation of the edge capacities.

6.2.2.1 Case study

Edge capacities can be obtained by using cuts on the triangle graphs. In order for the triangle graphs to be formulated, changes in delay when each road is flipped are needed. This implies a need to first know the delay for the initial one-way road network. Delay refers to the cumulative time spent waiting at all intersections of the network by all vehicles, based on a benchmark traffic loading. By running simulation (details of simulation is discussed in Chapter 7) on the initial WIJ one-way road network (obtained from Figure 6.1) this initial delay is obtained and found to be 577.6 seconds when actual traffic demand of the WIJ network is used (given in Section 7.6.2).

As a demonstration of the procedure for obtaining the edge capacities, roads H and J at intersection 7 (see Figure 6.1) are used.

In order to find values for $C1$, $C2$ and $C3$, three test cases are run on the simulation model: first when road H is flipped, secondly when road J is flipped and lastly when both road H and J are flipped. Overall delay of the road network is obtained for all three cases and the changes in delay (as compared to the initial delay of the initial road network) have been tabulated and given in Table 6.2.

The triangle graph for intersection 7 with just edges H and J is formulated and given in Figure 6.3.

Table 6.2: Additional delay for flipping the roads H & J

Flipped Road(s)	Delay (sec)	Changes in delay (sec)
H	515.6	-62.0 (= $C1$)
J	517.6	-60.0 (= $C2$)
H & J	686.6	109.0 (= $C3$)

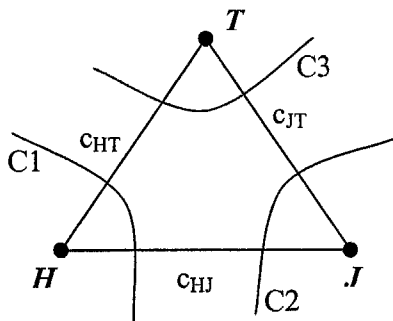


Figure 6.3: Triangle graph for roads H and J

With the triangle graph at hand and the values for $C1$, $C2$ and $C3$, three equations are formed [see equation (6.1)] as follows:

$$\begin{aligned} c_{HT} + c_{HJ} &= C1 \quad (= -62) \\ c_{JT} + c_{HJ} &= C2 \quad (= -60) \\ c_{HT} + c_{JT} &= C3 \quad (= 109) \end{aligned}$$

Solving the three simultaneous equations, the edge capacities c_{HT} , c_{JT} and c_{HJ} are obtained:

$$\begin{aligned} c_{HJ} &= -115.5 \\ c_{HT} &= 53.5 \\ c_{JT} &= 55.5 \end{aligned}$$

However, as discussed in Section 6.2.1.1, three roads need to be flipped (these are H , J and M); hence, in addition to those described above, more edge capacities are needed. Using the above method, changes in delay for flipping the roads at intersection 12 are found and given in Table 6.3. With that, similar triangle graph is formed for edges J and M and the edge capacities obtained. The edge capacity of a

road that occurs in more than one triangle graph (edge J in this case) is obtained when the values from different triangle graphs are arithmetically summed to get a final edge capacity for the road (see Section 6.2.2). The results for the final edge capacities are given in Table 6.4.

Table 6.3: Additional delay for “flipping” the edges H, J and M

Flipped Road(s)	Delay (sec)	Changes in delay (sec)
H	515.6	-62.0
J	517.6	-60.0
M	580.6	3.0
$H \text{ \& } J$	686.6	109.0
$J \text{ \& } M$	510.6	-67.0

Table 6.4: Edge capacities for the edges of the three roads

Edge (X, Y)	Edge capacity summation	Total edge capacity, c_{XY}
(H, J)	-115.5	-115.5
(J, M)	-67.0	-67.0
(H, T)	53.5	53.5
(J, T)	55.5 – 41.0	14.5
(M, T)	22.0	22.0

With the super node T as well as the edge capacities obtained, the min-cut graph model, $G(V,E)$, can be constructed. Therefore, construction of the min-cut graph model will be discussed next.

6.2.3 Min-cut graph model

In this section an informal description of the construction of the min-cut graph model and execution of the minimum cut on the graph is given; this is followed by a formal description of the graph construction algorithm and the min-cut graph model. Finally, the case study of WIJ is used to demonstrate the min-cut graph model construction algorithm as well as how the min-cut graph model can be used to solve the one-way road network problem.

6.2.3.1 Min-cut graph model construction and minimum cut

To construct the min-cut graph model, super node T is first defined by flipping various roads in turn in the network. Roads that remove strong connectivity from the network are placed into T . With T defined, n , the number of roads that can be flipped, will be known. Inclusive of T , there will thus be $n+1$ nodes in the min-cut graph model, $G(V,E)$.

Secondly, by using triangle graphs (one for every two roads at an intersection), edge capacities are found and summed up (if necessary).

Thirdly, edges are drawn linking each of the nodes together according to the edge capacities obtained; for example should an edge capacity, c_{XY} , of value 22.0 is obtained then an edge with a weight of 22 will be drawn linking node X and Y together. This will give the min-cut graph model of the given one-way road network.

Finally, a cut is used to partition the min-cut graph model into two sub-graphs. The roads in the sub-graph without T will be flipped. According to *Hypothesis A*, the change in delay (in flipping the above-mentioned roads) will be equivalent to the capacity of the cut. As such, in order to minimise delay, a minimum cut would be used.

6.2.3.2 The min-cut graph model

Using similar reasoning as the min-cut graph model given by Cheng *et al.* [13] for Module Orientation Problem, the min-cut graph model will be described in this section. It is noted that despite the min-cut graph model by Cheng *et al.* [13] is being adapted there still exist differences between the min-cut graph model by Cheng *et al.* [13] and the model used in this project. These differences are:

- 1) The graph used in Cheng *et al.* [13] is a representation of the PCB while the graph used here is a representation of a road network.

- 2) Cheng *et al.* [13] considered each net while flipping the modules involved to find the edge capacities based on the change in net length; the min-cut graph model given here considered each intersection while flipping the roads involved in order to find edge capacities based on the change in overall delay.

Algorithm 6.1: Min-cut graph model

First the graph needs to be constructed. This will involve calculating edge capacities and summing them up.

V , the node set of the graph, is initialised to contain n nodes (each node will represent a road that can be flipped) and the super node T . E , the edge set of the graph, will be initialised to a null set since the edge capacities of the graph are not found as yet.

For every intersection, s , with two roads (labelled A and B) that are not in T , a triangle graph will be formed and the edge capacities: c_{AT}^s , c_{BT}^s and c_{AB}^s calculated. The edge capacities found are added to E . To do this, first the sub-algorithm will check if the edge (i, j) exist in E . Should it be already present in E , the sub-algorithm will arithmetically add the edge capacity of (i, j) at the current intersection s , c_{ij}^s , to the total edge capacity, c_{ij} . This is so because capacities belonging to different intersections but pertaining to the same edge, c_{ij} , are to be added together (see Section 6.2.2). However should edge (i, j) not exist in E , it will be added to E and c_{ij} set to take the value of c_{ij}^s .

The graph is next constructed based on the value of the edge capacities. Edges joining nodes are drawn based on the edge capacities, which form the weight of these edges. (It is noted that no edge will be formed between two nodes should the edge capacity between the two nodes is zero).

With the graph constructed, in accordance to the min-cut graph model from Cheng *et al.* [13], a minimum cut is used to partition the graph into two sub-graphs. The graph

can then be stored in a list while the algorithm then searches exhaustively for the minimum cut of the graph.

With that, Algorithm 6.1 is formed. Figure 6.4 shows the flowchart for Algorithm 6.1. The pseudo-code of algorithm is given as follows:

ALGORITHM 6.1: Min-cut graph model

1. Initialize V with n nodes representing n roads and one super node T and initialize $E = \emptyset$.
2. For each intersection s and its two connected roads A and B :
3. Calculate c_{AT}^s , c_{BT}^s and c_{AB}^s at intersection s with triangle graph.
4. For each $c_{ij}^s \neq 0$ ($i \neq j$ and $i, j \in \{A, B, T\}$):
5. If edge $(i, j) \in E$
6. Set c_{ij} to $c_{ij} + c_{ij}^s$.
7. Else
8. Add edge (i, j) to E .
9. Set c_{ij} to c_{ij}^s .
10. Draw the graph based on the edge capacities obtained.
11. Initialize list A and append $G(V, E)$ to list A.
12. Exhaustive search for the minimum cut for the graph in list A.

The WIJ case study is presented next to demonstrate firstly the construction of the min-cut graph model, and secondly the use of the minimum cut to partition the min-cut graph model in order to obtain an optimal solution.

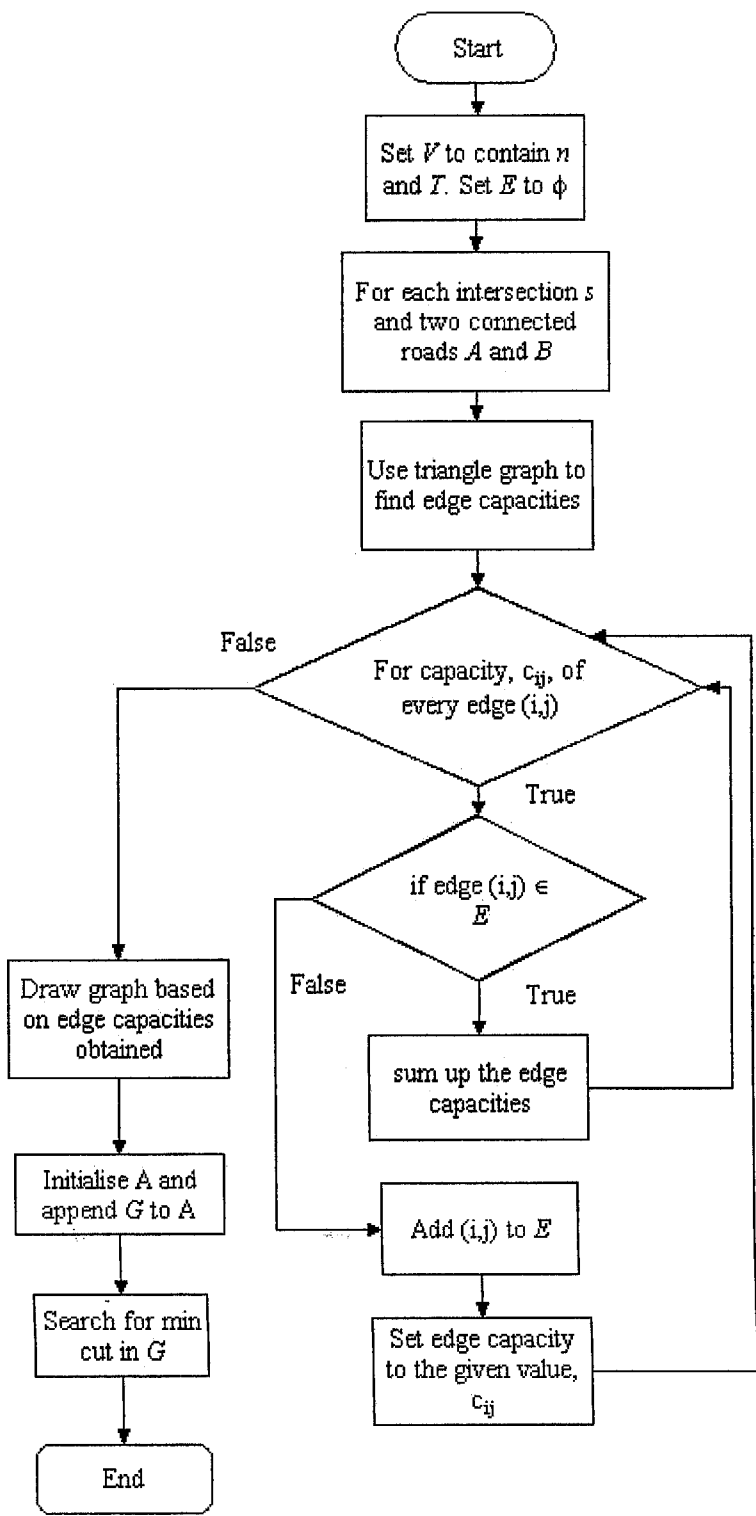


Figure 6.4: Flowchart for Algorithm 6.1

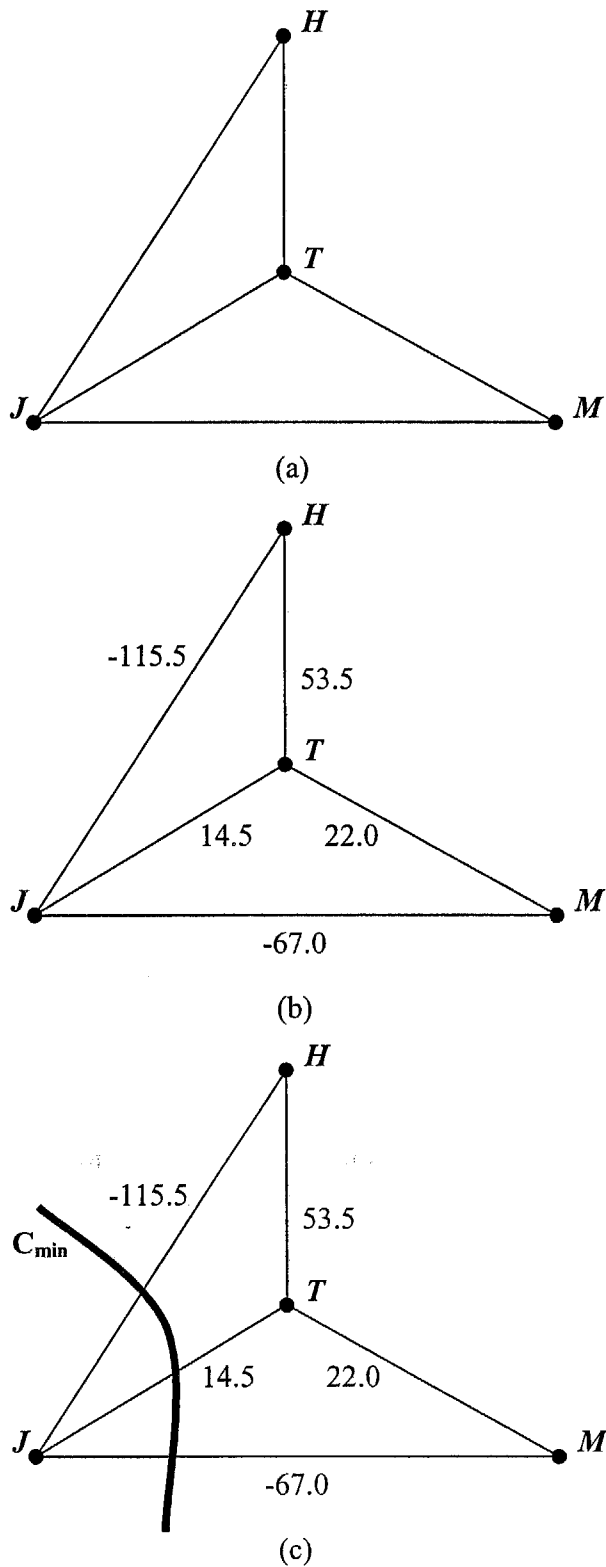


Figure 6.5: Min-cut graph model of the WIJ network

Since $C_{\min}$ partitioned the graph into $\{H, M, T\}$ and $\{J\}$, this means that road J is to be flipped. Flipping this road will thus give a new one-way road network for WIJ. This resultant one-way road network is as shown in Figure 6.6.

Capacity of $C_{\min}$ will be the sum of c_{JH} , c_{JM} and c_{JT} . This adds up to give -168 sec. To prove that *Hypothesis A* is correct is to show that when road J is flipped the delay of the whole network would be reduced by 168 sec. To do this, actual delay of the resultant one-way road network can be obtained. This can be done by running a computer simulation of the resultant one-way road network. The delay obtained can be compared to the delay of the initial one-way road network. This would give a more realistic value of the change in delay.

6.2.3.3 Case study

Based on results from Table 6.1, except for roads H , J and M , the rest of the roads of the WIJ network will form super node T . An unweighted graph consisting of 4 nodes $\{H, J, M, T\}$ is thus formed and shown in Figure 6.5(a).

Edge capacities obtained from Table 6.4 are added to the graph to give a weighted graph. The resultant graph is shown in Figure 6.5(b).

Finally a minimum cut, $C_{\min}$, is applied across the graph, this partitions the graph into two sub-graphs $\{H, M, T\}$ and $\{J\}$ as shown in Figure 6.5(c).

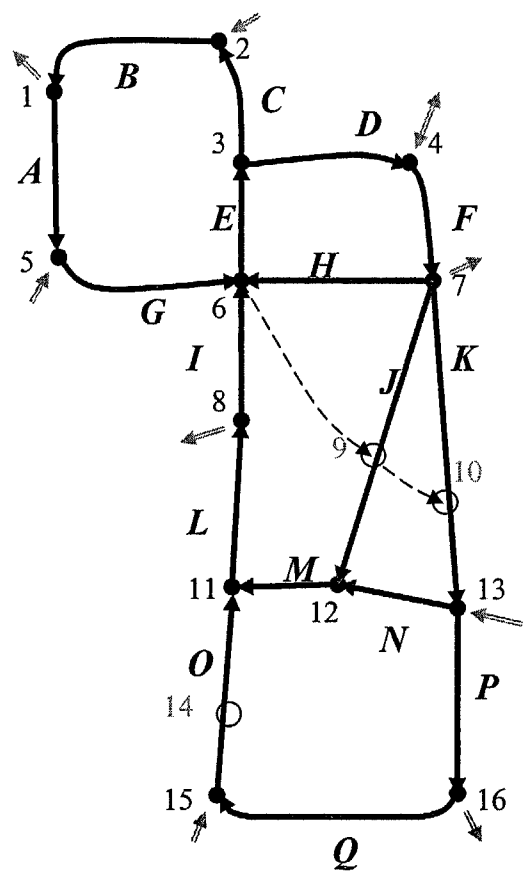


Figure 6.6: Resultant one-way road network for WIJ

Furthermore, to show that the resultant one-way road network is optimal, the delay for the resultant one-way road network may be compared against the delay of the original two-way road network. The next chapter discusses the use of computer simulation as a means to collect data on the road network’s delay, so as to validate the one-way road network model.

VII SIMULATION

7.1 Overview

This chapter begins with a brief discussion on the background to traffic simulation. This is followed by a justification of Paramics as the simulation package to be used in this project. Subsequently, the building and calibration of the WIJ road network model using Paramics is discussed. Traffic data to be used for input to the simulation is reviewed. Next, test cases are formulated for the validation of the one-way road network model. Finally, results of the test cases are given and discussed.

7.2 Background to traffic simulation

Traffic simulation has been in use since the early days of development of traffic theory [14, 34, 49, 59, 62, 100, 101]. The ever-increasing power of personal computers and the search for Intelligent Transport System (ITS) solutions to growing urban transport problems have led to the emergence of a number of microscopic simulation models as practical traffic analysis tools [49, 59]. Traffic flow simulation has been effective in assessing traffic conditions for designing various alternative transportation facilities [69].

The long history of traffic simulation is also evidenced by the numerous transportation simulation papers published in the Winter Simulation Conference Proceedings (WSC) and other publications over the years. The number of transportation simulation applications presented at WSC has increased to the point that in WSC'98 there was an entire track – 24 presentations organised into eight sessions – devoted to transportation simulation applications [80].

Transportation simulation models are built using certain mathematical models with some basic assumptions of traffic flow [2]. Therefore, the ability to recreate all real world cases is limited, yet simulation allows the solution of complex transportation related problems prior to actual construction.

Transportation simulation models can also help alleviate public concern on the impact of proposed traffic management or mass transit projects. An animated illustration of the strategy can be a key to winning acceptance by both the public and decision-makers. In addition, simulation models are capable of providing an ideal medium for conveying pros and cons of planning, design and operation options to non-technical audience. An overall simulation approach can be applied to analyse not only peak hour traffic conditions but can also be effectively used for planning transit services and other low-tech transportation solutions [102].

The demand in using traffic simulation has increased, and several software packages such as VISSIM [53], WATsim [52, 103], CORSIM [53, 64, 104], TRANSYT [105] and Paramics [12, 56, 64, 106] have been developed. A suitable simulation package from the many available options is thus needed to model the road network and to validate the one-way road network model. The next section therefore talks about pertinent issues in selecting Paramics as the simulation package for this project.

7.3 Choosing Paramics

This section first discusses the Paramics model. This is followed by reasons for choosing Paramics as the simulation package for this project.

7.3.1 The Paramics model

The Paramics model [12] is a microscopic stochastic simulation modelling software capable of modelling both urban road network situations as well as a wide set of freeway types. It is capable of simulating individual vehicles for the duration of their trip, providing travel time, traffic flow and congestion information as well as enabling the modelling of the interface between drivers and the ITS [56].

It can model each road type restriction (speed limit, reserved lanes, height and weight restrictions) as well as the type of vehicle and the driver's degree of aggressiveness. However, unlike other microscopic simulation programs Paramics is UNIX-based.

Nevertheless, using a Microsoft Windows-based software called Exceed by Hummingbird, the X-Windows platform is emulated on a standard workstation computer [64].

The Paramics model consists of five software modules [106]:

- i) **Modeller:** the core tool for simulation and visualisation.
- ii) **Processor:** the simulation configuration tool that allows multiple network simulations to be run in batch mode.
- iii) **Analyser:** the post simulation statistical viewing tool; it is primarily used to compile and display reports on statistical data output from the Modeller.
- iv) **Programmer:** the Application Programming Interface (API) to Modeller that allows users to overwrite default settings.
- v) **Monitor:** the module which helps calculate the levels of traffic pollution emission on the road.

The Modeller is the modelling tool that provides the three fundamental operations of Paramics, mainly model build, traffic simulation with 3-D visualisation [64] as shown in Figure 7.1 and powerful statistical output [107] available through a graphical user interface. Many aspects of a traffic network can be investigated including integrated urban and freeway networks, advanced signal control, roundabouts, public transportation, car parking, incidents, truck lanes, or special lane usage.

The Programmer allows users to customise various parts of the core models by using an API. Users can override the default behavioural models (such as car following, gap acceptance, lane changing or route choice) to match more closely the local driver and vehicle characteristics. Users can even implement their own complementary traffic

control strategies such as adaptive ramp metering, signal optimisation and incident detection [106].

Input requirements for the model include:

- i) Network characteristics: link description, lane restrictions, signposting, forced lane changes, geometry.
- ii) Demand data: O-D zone areas, level of O-D demand, vehicle type and proportion, breakdown by time period (if any).
- iii) Assignment: link cost factors, coefficients of generalised cost equations, assignment techniques.
- iv) General configuration: time step duration, speed memory, mean target headway, mean reaction time, right-hand drive/left-hand drive.



Figure 7.1: 3-D graphics visualisation of Paramics

Paramics's outputs include:

- i. Statistics at the overall road network level like total travel distance, overall travel time, total delay.
- ii. Link-based results using the Analyser that reports statistics such as traffic flow, queue lengths, delays, speeds and densities.
- iii. Location specific results using detector to get instantaneous information.

As Paramics is UNIX-based it is capable of running in parallel [57, 58, 108] with several servers for parallel processing of huge networks.

With the Paramics model explored, the reasons for using Paramics for this project are discussed next.

7.3.2 Reasons for using Paramics

Paramics is chosen as the traffic simulation software for the validation of the designed one-way road network model. The reasons for using Paramics are as follows:

Uneven terrain

Paramics can accurately represent the physical characteristics of the network in the modelled network geometry. The use of a CAD drawing as an overlay map in Paramics proved to be a very efficient and valuable tool in the process of building the network geometry [106]. This is an important aspect of the project since the WIJ network consists of various curved roads and gradient changes (slopes and hills). Furthermore, the use of 3-D animation in Paramics allows such gradient changes to be viewed in a clearer manner, giving users a better visualisation of the simulation.

Data collection

Paramics has very strong abilities [109] in statistical data collection and generates simulation results that better match field observed conditions [64]; besides collecting

a few general network performance data, Paramics can output link-based, intersection-based, detector-based and trip-based data. As the collection of overall intersection-based delays in the network (needed for this project) is possible, this makes Paramics a good choice as the simulation package.

Modelling congestion

In loading the WIJ road network with vehicles in this project, congestion may be expected. Congestion is a phenomenon [110] associated with chaotic and non-linear road traffic and as such is best modelled at microscopic level. Paramics being a microscopic simulation and capable of simulating each vehicle individually [56] is best suited for studies of congestion (as in this project).

Furthermore, Paramics uses a scheme of notional free queue. This allows vehicles which cannot be released into the network (due to congestion) to be kept in a separate array and released only when there is available capacity or reallocated to other available links of the same O-D pair [12]. This aspect of Paramics makes its simulation closer to reality since in actual situations drivers would move onto other available roads if their intended routes become too congested.

Curved roads

In network design for Paramics, each link (road) has an associated type that gives information on attributes such as number of lanes, one-way or two-way and especially the curvature of the road. Due to this, Paramics is well suited for designing and studying curved roads as well as roundabouts [111]. This feature is best seen in the use of Paramics in studies of the Britannia Roundabout in Adelaide, Australia [71]. This kind of refinement is absent in other simulation packages which use link-based trip assignments and a specified 50-foot minimum spacing requirement between each node [64]. As the WIJ road network does have a number of curved roads, Paramics makes a good choice for an accurate network design as well as simulation study of this case.

Zones and car parks

In Paramics, traffic data for the network are input in the form of O-D matrix. However, zones are used instead of locations in the O-D matrix. The definition of zones in Paramics refers to the final destination of drivers regardless of where the vehicles are parked. This aspect of Paramics makes it closer to reality since this means that a driver going to Building A could park his vehicle in the car park across the street instead if the car park in Building A is full. Car parks are thus defined separately from zones and the capacities of car parks are required in Paramics. Depending on the physical distances of the car parks to the various zones (this is calculated automatically by Paramics) a “walking time for drivers” is given to each car park with respect to each zone. Drivers will choose the car park with the least “walking time” to reach their destinations/zones unless the car park is full. In such case, drivers will choose the next car park with the least “walking time”.

Turning percentage versus O-D matrix

The WIJ network case study can be viewed as a study of an alternative network to an original live network. Simulation packages that use percentage of turning movements at intersections to simulate traffic movements are highly inappropriate in the studies of transportation system alternatives (be it an addition of lane or change in traffic signalling). This is because the turning percentages change immediately in the alternative network. As the alternative network has yet to exist, there is no way to predict the new turning percentages [74]. Paramics would hence make a better choice because the software (with the O-D matrix) uses traffic assignment to route the vehicles so they know at which intersections to make “what kind of” turns in order to reach their destination [70].

With Paramics chosen as the traffic simulation package, the WIJ network can thus be built and calibrated. The next section will discuss building and calibrating the WIJ network with Paramics.

7.4 Using Paramics

7.4.1 Building the network with Paramics

The following assumptions have been made in the modelling of the WIJ road network:

- i) Buses arrive at regular intervals.
In reality buses do not arrive at expected times nor regular intervals all of the time. However delays caused by the above-mentioned are not consistent (it may not even happen at all) and their impact are not great enough to affect the whole road network.
- ii) Delay to traffic caused by pedestrians, cyclists and motorcycles are negligible.
In reality there are too many reasons for a car to stop for pedestrians, cyclists and motorcycles. Nevertheless delay caused by the above-mentioned are not consistent (it may not even happen at all) and their impact are not great enough to affect the whole road network.
- iii) Vehicles do not stop at any point of any road other than due to congestion or upon reaching their destination.
In reality, vehicles may stop along the road for various reasons, for example to ask pedestrians for directions, to drop a letter into the mailbox, to pick up/drop off passengers just to name a few. Nevertheless delays caused by the above-mentioned are not consistent (it may not even happen at all) and their impact are not great enough to affect the whole road network.
- iv) Vehicle arrivals follow an Erlang distribution (see Section 2.2.3.3).

The base road network used for modelling is developed by graphically constructing the road network within Paramics. A CAD file of the network (as seen in Figure 7.1) is used as a layout from which kerb lines are drawn directly onto the scaled CAD file to ensure the model network represents the actual road network as close as possible. The uneven terrain that WIJ lays on is sculpted by using the contour map of WIJ; heights of various intersections and gradient of various roads are input into the simulation to achieve the uneven terrain of the actual network.

Link lengths and lane characteristics such as number of lanes, speed limits and lane widths together with bus routes are added into the network. As all intersections are unsignalised in this case study there is no need for addition of signal timings and phasing.

In addition, zones, representing origins and destinations, are defined alongside car parks and their associations with respective zones.

In the road network for WIJ, there are 5 zones (see Section 7.5) modelled for the simulation. Traffic data are given in zones in the form of O-D matrix so that the pilot run of the simulation can be made and the simulation thus calibrated.

7.4.2 Calibration of Paramics

7.4.2.1 Network

To start the calibration of the network, it is important that the physical characteristics of the network are accurately represented in the simulated network geometry. The use of a CAD drawing as overlay map in Paramics proved to be very efficient and valuable in ensuring the accuracy of the network's physical characteristics [106].

Network geometry is a major factor influencing vehicle behaviour; if modelled incorrectly vehicles may be forced to make sharp turns at the ends of the links causing a drop in vehicle speed. Vehicle speeds are reduced so that the vehicles can

make the turn safely; this could in turn result in more vehicle braking, thereby causing shockwaves to occur. This may lead to disruption in traffic flow and possibly also disruption [112] in the generation of vehicles in the nearby zones.

Badly aligned nodes and stoplines are avoided by using Paramics's default kerb positions. As long as nodes of the network are correctly placed onto the CAD file Paramics will usually draw a good representation [106] of the default kerb points and stoplines. The result of this is due to Paramic's link categories files that will dictate the width of the travelled way and position the kerb points and stop-lines accordingly once the lane width in question is set accurately to the actual width.

7.4.2.2 Vehicle Mean Top Speed

According to Gardes *et al.* [106], vehicles tend to exceed the speed limit by 10% in Paramics. In addition to that Lee *et al.* [113] verified the findings and shown that (for the roads in Singapore) by coding the link speeds of vehicles in Paramics 10% lesser than the top speed (mean) of vehicles allowed the simulation is able to realistically represent the actual road situation in Singapore. Hence the link speeds of vehicles are coded 10% less than the top speed (mean) of vehicles allowed so as to ensure a more realistic representation of the actual situation by the simulation.

7.4.2.3 Overall Simulation Configuration

The simulation time steps of Paramics will determine how many calculations are carried out for each second of the simulation. The default time step is set at 2 which means that there will be a calculation every 0.5 seconds of the simulation. If the time step is changed to 5, for example, then the calculations will be performed once every 0.2 seconds. A number of parameters such as vehicle speed and acceleration have some randomisation associated with them, hence the results of the simulation will be different for different time steps used. The time step was thus increased from 2 to 5 steps per seconds due to the fact that according to Gardes *et al.* [106], high density

flows mostly require more time steps per second and high density flow in terms of congestion is highly expected in this case study.

Furthermore, in association with time step change, the speed memory (the number of time steps for which a vehicle remembers its speed) was changed from 3 (the default value) to 8 steps. This is because changing the size of the speed memory allows the modelling of the same reaction time with smaller time steps.

7.4.2.4 Driver Behaviour Factors

According to Wylie *et al.* [111] three basic models are used in Paramics for the control of movements of individual vehicles in the network, namely:

- i) Vehicle Following model.
- ii) Gap Acceptance model.
- iii) Lane Changing model.

These models are effectively controlled by two key user specified parameters namely the mean headway (H) and the mean reaction time (R), as stated by Smith *et al.* [12]. By increasing or decreasing these two parameters the results of the simulation would change considerably.

The default values for mean headway and mean reaction time are both 1 second in Paramics. According to Abdulhai *et al.* [61], vehicles tend to have lower reaction times and accept smaller gaps than the default values of 1 second. However using genetic algorithm Lee *et al.* [113] calibrated the values for mean headway and mean reaction time as $H=0.615$ seconds and $R=0.415$ seconds for the roads of Singapore.

According to Gardes *et al.* [106], an acceptable range of simulated average network speed is between 48 km/h to 64 km/h. Hence the simulation for this project is calibrated at $H=0.615$ seconds, $R=0.415$ seconds and an average network speed of

50km/h is used; this is in conjunction with the speed limit of 50km/h for the WIJ road network.

With the network built and calibrated, the next step in the simulation would be to load the simulation with traffic data. The next section will talk about the traffic data needed for the simulation.

7.5 Traffic data for the simulation

Actual traffic data (of the WIJ road network) was given to the Transportation Laboratory of the School of Civil and Environmental Engineering in Nanyang Technology University by CPG Consultants Pte Ltd for the purpose of case study in one of the subjects (Network Analysis and Simulation) taught in the Masters programme by the school and is used for the simulation.

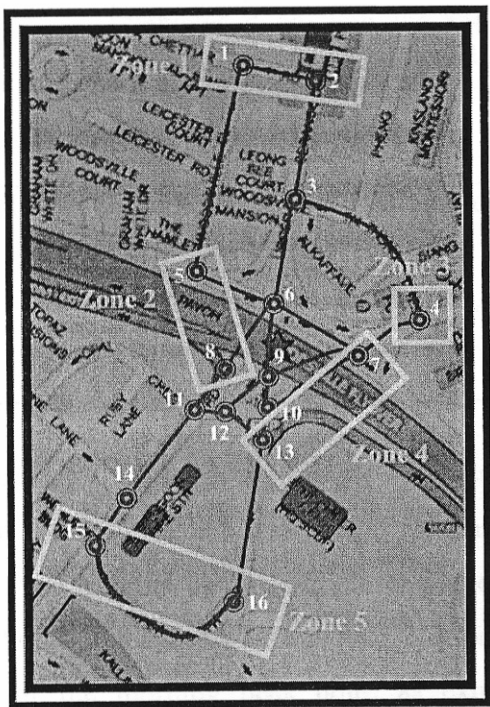


Figure 7.2: 5 zones are defined

Table 7.1: Zones definition

Zone	Nodes defining the Zone	Reasons
1	1 & 2	Node 1 brings vehicles out of the road network towards Upper Serangoon Road while Node 2 brings vehicles into the road network from Upper Serangoon Road. As both nodes have a similar function of serving vehicles in and out of Upper Serangoon Road, hence they form Zone 1.
2	5 & 8	Node 8 brings vehicles out, onto PIE ramp (towards Jurong) while Node 5 brings vehicles down into the network from PIE ramp (from Jurong). As both nodes have a similar function of serving vehicles in and out of PIE (to-and-fro Jurong), hence they form Zone 2.
3	4	Node 4 brings vehicles both in and out of the road network via Macpherson Road, hence node 4 by itself forms Zone 3.
4	7 & 13	Node 7 brings vehicles out of the road network up onto PIE ramp (towards Changi) while Node 13 brings vehicles down into the road network from PIE ramp (from Changi). As both nodes have a similar function of serving vehicles in and out of PIE (to-and-fro Changi), hence they form Zone 4.
5	15 & 16	Node 16 brings vehicles out of the road network to Kallang River while Node 15 brings vehicles into the road network from Kallang River. As both nodes have a similar function of serving vehicles in and out of Kallang River area, hence they form Zone 5.

The traffic data for the simulation is given in terms of zones in the form of an O-D matrix. The purpose of the original data was used for a feasible study for reducing traffic flow in the WIJ road network by an addition road lane. The collection of O-D matrix follows the method as outlined by Meng *et al.* [39] for the roads of Singapore. This O-D matrix is for the morning peak hour; as explained by Ortúzar and Willumsen [1], either the morning or evening peak hour are the best times to do a study on network flow. This is so since these two periods are the times when the traffic of the road network is the busiest and whatever study on road networks ought to be carried out when traffic is heaviest so as to understand the problem of

congestion caused by such heavy traffic. As such this O-D Matrix of the WIJ network given is the most suitable for this project to be used in simulation for the purpose of comparing the traffic congestions in alternate designs of the WIJ network.

There are 5 zones altogether (see Figure 7.2), since zones are basically areas whereby vehicles are expected to enter and leave the road network they are made up of exit/entrance points (which are nodes 1, 2, 4, 5, 8, 7, 13, 15 and 16). In this case study, zones are defined based on the functionality as well as physical proximity of the roads. Table 7.1 gives the zones, their corresponding nodes and the reasons for grouping various nodes together to form the various zones.

The O-D matrix for the WIJ network during the morning peak hour period is given in Table 7.2, figures given are in volume per hour (vol/h). As highlighted by Ortuzar and Willumsen [1], even though traffic flow is given in terms of hour (as in volume per hour) they are only representative of the actual traffic for about fifteen minutes. This is especially true for peak period traffic since traffic volume increases tremendously near the nine o'clock start work / five o'clock knock-off time and decays rapidly after that. The O-D matrix given in Table 7.2 is only representative of the traffic of WIJ during the peak period hour from 8:40am to 8:55am. Nevertheless the simulation is run from the period of 8:30am to 9:00am using a constant O-D matrix; the reason for this is to allow the initial loading of vehicles into the road network so as to build up the traffic in the first ten simulation minutes to the eventual peak hour starting at 8:40am.

Table 7.2: O-D matrix for WIJ network

	To Zone 1	To Zone 2	To Zone 3	To Zone 4	To Zone 5
From Zone 1	0	80	140	29	750
From Zone 2	83	0	702	57	879
From Zone 3	51	867	0	33	771
From Zone 4	69	40	630	0	51
From Zone 5	504	66	88	10	0

With the traffic data defined as above, the next step is to test the WIJ network with the given traffic data.

7.6 Test cases and results

7.6.1 Test cases

In validating the one-way road network model three test cases are used:

- i. Case A: Original two-way road network
- ii. Case B: Initial one-way road network layout
- iii. Case C: Final one-way road network

Case A: Original two-way road network

The original two-way road network of WIJ is used in the simulation with the traffic data described in the previous section. The reason for using this is that there is a need to compare the final one-way road network with the original two-way road network. If the one-way road network model is effective, it will produce a one-way road network that has a lesser overall delay than the original two-way road network. Thus comparing the two networks would allow assessment of the effectiveness of the one-way road network model.

Case B: Initial one-way road network layout

As discussed in Chapter 6, in the min-cut graph model, when various roads are flipped, the additional delay in the whole network caused by the flipping would be recorded. This additional delay refers to the change of the network delay from the delay of the initial one-way road network layout (obtained from a conversion algorithm discussed in Chapters 4 and 5). This implies that the network delay of the initial one-way road network layout is required.

Table 7.3: Tabular results for a typical simulation run for Cases A, B and C

Time (am)	Delay (sec)		
	Case A	Case B	Case C
8:30:00	0	0	0
8:30:30	0	0	0
8:31:00	0	0	0
8:31:30	0	0	0
8:32:00	0	0	0
8:32:30	0	0	0
8:33:00	0	0	0
8:33:30	0	0	0
8:34:00	0	0	0
8:34:30	0	0	0
8:35:00	0	0	0
8:35:30	0	0	0
8:36:00	0	0	0
8:36:30	0	0	0
8:37:00	0	16	0
8:37:30	0	22	12
8:38:00	0	23	18
8:38:30	0	47	25
8:39:00	60	53	39
8:39:30	78	60	47
8:40:00	81	67	52
8:40:30	116	79	63
8:41:00	268	135	80
8:41:30	340	197	94
8:42:00	347	234	147
8:42:30	481	279	195
8:43:00	489	312	244
8:43:30	494	385	268
8:44:00	512	436	289
8:44:30	534	477	296
8:45:00	541	506	312
8:45:30	560	519	323
8:46:00	611	568	375
8:46:30	633	589	384
8:47:00	678	613	390
8:47:30	688	645	398
8:48:00	754	658	401
8:48:30	798	661	411
8:49:00	799	670	426
8:49:30	813	672	461
8:50:00	814	679	474
8:50:30	828	682	476
8:51:00	841	684	477
8:51:30	856	671	481
8:52:00	894	665	472
8:52:30	893	662	468
8:53:00	886	657	410
8:53:30	885	646	383
8:54:00	885	621	299
8:54:30	742	536	214
8:55:00	671	509	143
8:55:30	554	325	89
8:56:00	340	268	21
8:56:30	301	193	0
8:57:00	239	116	0
8:57:30	210	67	0
8:58:00	189	0	0
8:58:30	0	0	0
8:59:00	0	0	0
8:59:30	0	0	0
9:00:00	0	0	0

Case C: Final one-way road network

As discussed above, in order to assess the effectiveness of the one-way road network model, one would have to compare the delay of the original two-way road network to that of the final one-way road network. Hence there is a need for Case C, the final one-way road network.

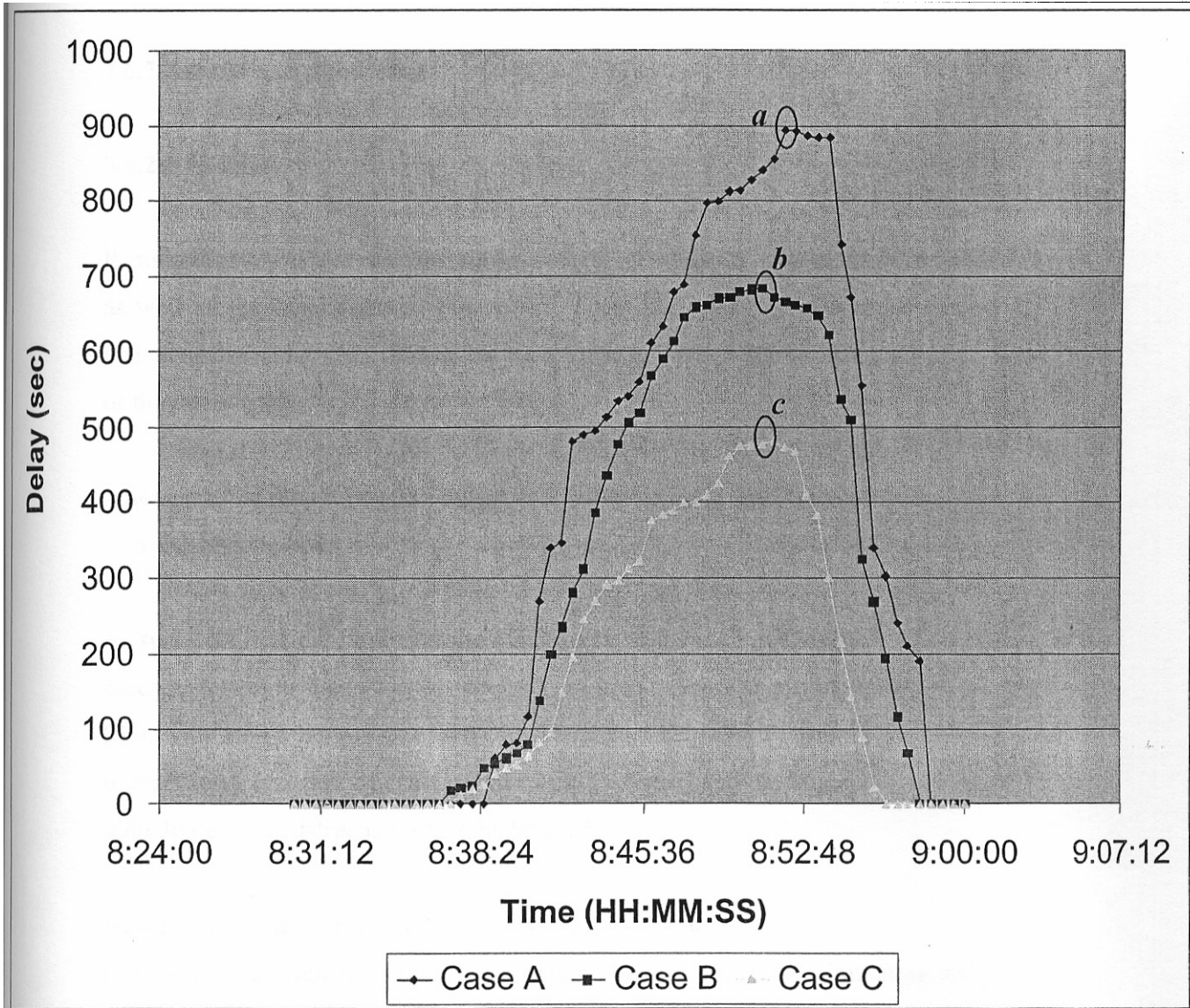


Figure 7.3: Graphical results for a typical simulation run for Cases A, B and C

All three cases are tested with the actual traffic data as given in Section 7.5. The simulation is allowed to run for 30 simulation minutes, from 8:30am to 9:00am. This is to simulate the morning peak period when traffic is at its heaviest. At the end of each test run, Paramics gives the overall delay required. This overall delay gives the average delay (in seconds) expected by a vehicle in traversing the network within the given simulation time. Delays for the test cases are collected and given in the subsequent section.

7.6.2 Results and discussion

7.6.2.1 Results

Results for typical test runs for all three cases are given in tabular form in Table 7.3 as well as graphical form in Figure 7.3. These serve to illustrate and compare the instantaneous delay experienced by each vehicle throughout the simulation for each of the three cases.

After each test run, the overall delays (see below) for the road network for each of the 3 cases are collected.

41 such test runs are done and the overall delays for each of these test runs for all 3 cases are given in Table 7.4. 41 test runs are used because according to Walpole *et al.* [85], with such a sample size, one can then assume Normal Distribution for the mean overall delay (as cited by the Central Limit Theorem) and the Student's t-distribution with 40 degrees-of-freedom can thus be used.

Results for Case A: Original two-way road network

In Case A, the original two-way WIJ road network is used. The simulation is run for 30 simulation minutes (during morning peak period from 8:30:00am to 9:00:00am) for each of the 41 test runs. At the end of each test run, Paramics gives the overall delay required. This overall delay gives the average delay (in seconds) expected by a vehicle in traversing the network within the given simulation time.

Table 7.4: Overall delay for the whole network for Cases A, B and C

Test run	Overall Delay		
	Case A	Case B	Case C
1	743	589	421
2	797	574	423
3	814	569	409
4	758	581	403
5	770	597	412
6	821	578	405
7	811	565	408
8	798	579	413
9	794	584	411
10	815	581	401
11	809	576	394
12	763	589	407
13	799	591	409
14	804	575	418
15	786	584	403
16	775	563	404
17	797	582	410
18	809	577	412
19	812	569	416
20	788	572	417
21	803	591	412
22	764	564	411
23	807	576	409
24	796	579	401
25	793	582	402
26	817	547	397
27	825	576	405
28	800	569	407
29	789	598	411
30	793	592	398
31	804	586	410
32	798	574	407
33	779	577	406
34	814	582	420
35	810	569	418
36	799	571	413
37	801	570	409
38	792	586	399
39	796	565	414
40	801	578	408
41	794	573	429
Total	32638	23680	16782
Mean	796.0488	577.561	409.3171
Variance	308.7476	101.7524	60.19074

Generally, for each of the test run, vehicles do not experience any delay in the first few minutes (see Figure 7.3). This is due to the initial loading of vehicles into a non-congested road network hence resulting in no delay.

As time passes to about 8:39:00am, vehicles start to experience delay. This is due to congestion as the road network is beginning to fill up with vehicles in the morning peak period. Figure 7.4 shows the simulation at about 8:50:00am when delay is 814.0 seconds. As shown the whole network is congested with vehicles. The delay then increases to 894.0 seconds (point *a* in Figure 7.3) at 8:52:00am for the same reason.

However, delay decreases for the next 6 minutes up to 8:58:30am where delay becomes zero. This is due to vehicles being cleared off the network as the morning peak period ends and less vehicles enter the network. This reduces congestion and hence the delay decreases.

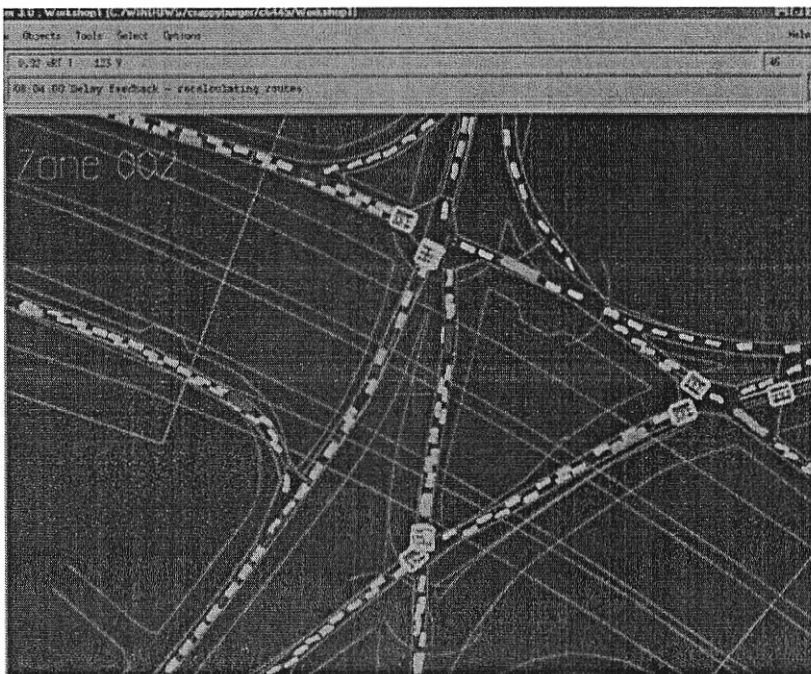


Figure 7.4: Case A at 8:50:00am

The overall delay for the test run is obtained from Paramics after the simulation is run. Table 7.4 shows the overall delay obtained for all 41 test runs. The sample mean overall delay for Case A is 796.0 seconds (this means that on the average a vehicle is delayed for about 13 minutes and 16 seconds when traversing the original two-way road network during morning peak period) while the sample variance is 308.8 seconds. Since actual variance is unknown in this case, according to Walpole *et al.* [85], the Student's t-distribution with 40 degrees of freedom is used to find the estimated overall delay for Case A.

Hence, the mean overall delay for Case A is estimated to be 796.0 seconds with a 95% confidence interval of (790.5, 801.6).

Results for Case B: Initial one-way road network layout

In Case B, the initial one-way WIJ road network layout obtained from the conversion algorithm is used. The simulation is run for 30 simulation minutes (during morning peak period) for each of the 41 test runs. Like Case A, at the end of each test run Paramics will give the results.

Like Case A, vehicles do not experience any delay in the first few minutes (see Figure 7.3) for all of the test runs. This is due to the initial loading of vehicles into a road network with no congestion (hence resulting in no delay).

As time passes, at about 8:37:00am, vehicles start to experience delay. This is due to congestion as vehicles begin to fill up the network. However, compared to Case A (which only experiences delay from 8:39:00am), Case B experiences delay 2 minutes earlier (at 8:37:00am) than Case A. This is because, using a similar explanation by Karagheuwfi [60], Case A being a two-way network is a faster ("faster" as in "clearing traffic out of the network more efficiently") road network than Case B, a one-way network, when traffic volume is low (see Section 2.2.6).

As time passes, congestion begins to build up. Figure 7.5 shows the road network at 8:50:00am; as shown the network is not as congested as that shown in Figure 7.4.

Delay increases to 684.0 seconds (point *b* in Figure 7.3) at 8:51:00am after which delay starts decreasing for the next 6 minutes (until 8:57:30am) where delay becomes zero as the morning peak period comes to an end. Comparing this to Case A however, Case B's delay only peaks at 684.0 seconds (lower than Case A's 894.0 seconds), starts decreasing 1 minute earlier and drops to zero 30 seconds earlier. This shows that Case B is a more efficient road network as compared to Case A therefore it is able to clear the vehicles for the morning peak period faster than Case A.

The overall delays for the 41 test runs are shown in Table 7.4. The sample mean overall delay for Case B is 577.6 seconds (this means that on the average a vehicle is delayed for about 9 minutes and 37 seconds when traversing the initial one-way road network during morning peak period) while the sample variance is 101.8 seconds. Since actual variance is unknown in this case, according to Walpole *et al.* [85], the Student's t-distribution with 40 degrees-of-freedom is used to find the estimated overall delay for Case B.

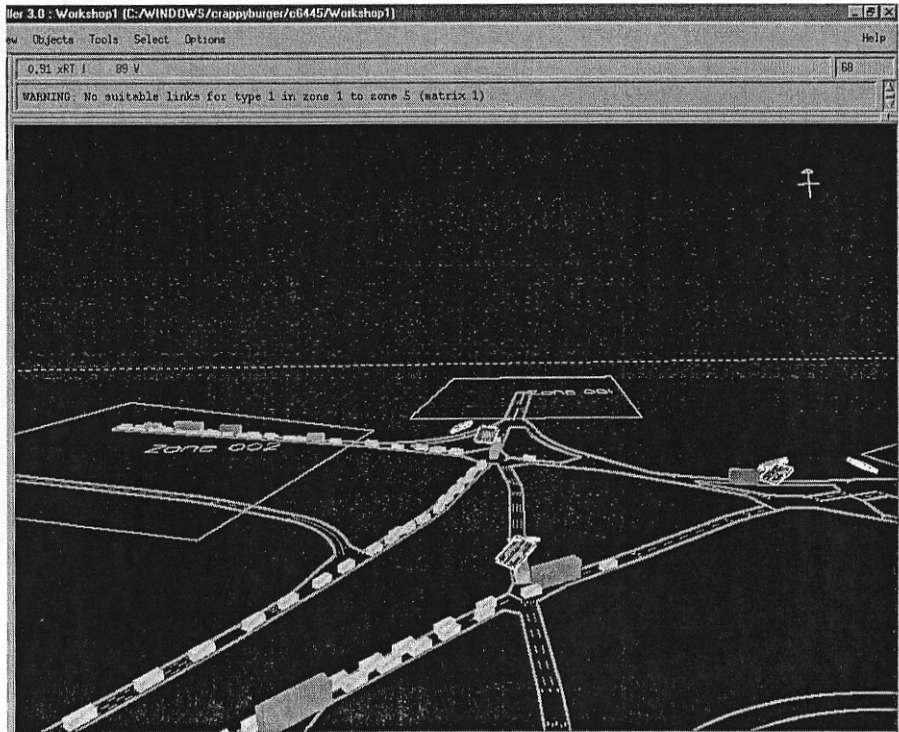


Figure 7.5: Case B at 8:50:00am

Hence, the mean overall delay for Case B is estimated to be 577.6 seconds with a 95% confidence interval of (574.4, 580.7).

Results for Case C: Final one-way road network

In Case C, the final one-way WIJ road network obtained from the min-cut graph model is used. The simulation runs for 30 simulation minutes (during morning peak period) for each of the 41 test runs. Like Cases A and B, at the end of each test run Paramics will give the overall delay obtained.

Like Cases A and B and for the reasons mentioned, vehicles do not experience any delay in the first few minutes (see Figure 7.3) for all of the test runs.

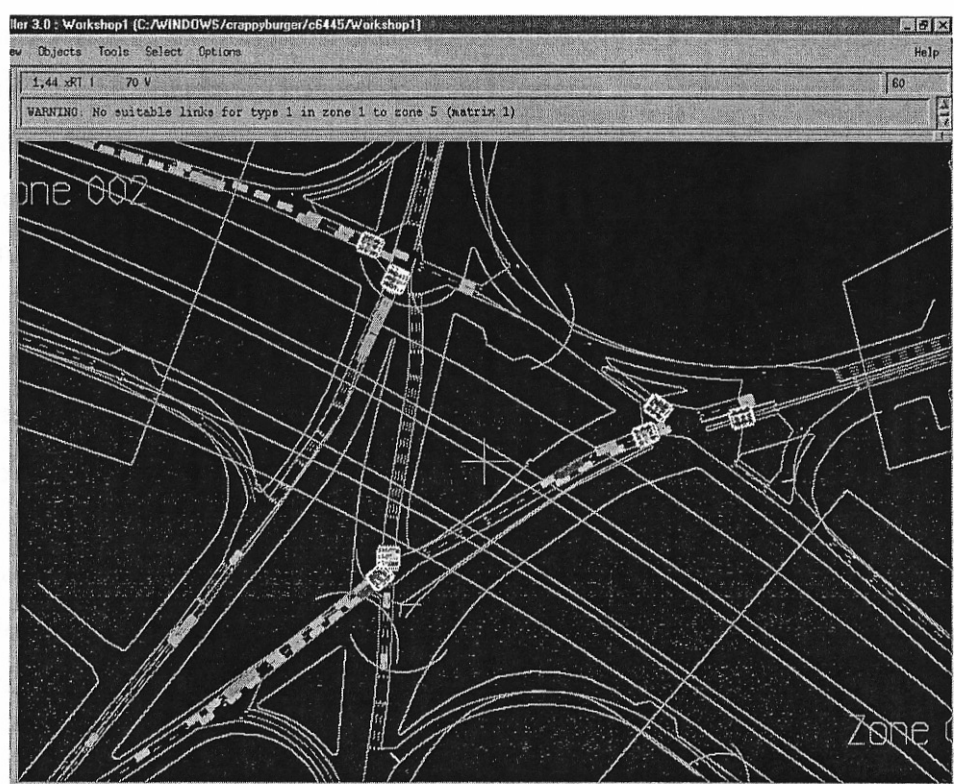


Figure 7.6: Case C at 8:50:00am

At about 8:37:30am, vehicles start to experience delay. Like A and B, this is due to congestion when vehicles start to fill up the network in the morning peak period. However comparing this with Case A, Case C experiences delay 1½ minutes earlier

than Case A. Like Case B according to Karagheuwfi [60], when traffic volume is low Case A being a two-way network is a faster road network than Case C, a one-way network (see Section 2.2.6).

On the other hand comparing the initial start-up delay of Case C with Case B, Case C experiences delay $\frac{1}{2}$ minute later than Case B; this indicates that as a one-way road network, Case C is a more efficient network than Case B.

As time passes, congestion begins to build up. Figure 7.6 shows the road network at 8:50:00am. As seen, the network is the least congested as compared to the networks in Figure 7.4 and Figure 7.5. Delay of the network increases to 481.0 seconds (point *c* in Figure 7.3) at 8:51:30am after which delay starts decreasing for the next $5\frac{1}{2}$ minutes (until 8:56:00am when delay becomes zero). Like Cases A and B, the reason for this is the reduction in the vehicle numbers as the morning peak period ends. Comparing this to Cases A and B however, Case C's delay only takes $5\frac{1}{2}$ minutes to reduce to zero. This is much shorter than both Cases A and B. This shows that Case C is a more efficient road network as compared to Cases A and B.

The overall delays for the 41 test runs are shown in Table 7.4. The sample mean overall delay for Case C is 409.3 seconds (this means that on the average a vehicle is delayed for about 6 minutes and 46 seconds when traversing the final one-way road network during morning peak period) while the sample variance is 60.2 seconds. Since actual variance is unknown in this case, according to Walpole *et al.* [109], the Student's t-distribution with 40 degrees-of-freedom is used to find the estimated overall delay for Case C.

Hence, the mean overall delay for Case C is estimated to be 409.3 seconds with a 95% confidence interval of (406.7, 411.8).

7.6.2.2 Discussion

Discussion of the results is given in this section. The discussion is grouped into two segments:

- i. *Hypothesis A*
- ii. One-way road network model

Hypothesis A

Referring back to Section 6.2.2.1 where the delay of the initial one-way network layout (Case B) is used to calculate additional delay caused by flipping various roads, the estimated overall delay of 577.6 seconds for Case B is used. This is because when flipping each of the roads, the delay caused should be delay towards the whole network and not just to the roads flipped. Hence the overall delay is used instead.

Comparing the overall delay of Case B with that of Case C, overall delay in Case C is 409.3 seconds while that in Case B is 577.6 seconds. This implies that additional delay in changing the one-way network from Case B to Case C is -168.2 seconds. Capacity of the cut, $C_{\min}$, (see Section 6.2.3.2) obtained from the min-cut graph model on the other hand is -168.0 seconds.

Recall *Hypothesis A*:

Given a one-way urban road network represented by a graph $G(V,E)$ as described above, a cut $(U, V-U)$ that separates subset U from $V-U$, its complement (that is $\bar{U} = V-U$), with the super node T belonging to $V-U$, is such that capacity of the cut $(U, V-U)$ takes the value of C if and only if the change in overall delay in the road network caused by reversing the flow direction of the one-way roads in U is equivalent to C .

To prove *Hypothesis A* is to show that the capacity of $C_{\min}$ is equivalent to the change in delay when road J is flipped. Statistically speaking, $C_{\min}$ ($= -168$ sec) may be equivalent to the change in delay ($= -168.2$ sec) when Case B is changed to Case C (that is when road J is flipped). This is so, as it is probably not coincidental that data values collected from experiments are equivalent to the expected values obtained from models. Furthermore, a change in delay at -168.0 seconds is very close to the

expected calculated value of -168.2 seconds. However, we can only show that the result is statistically significant.

As a sample size of 41 is used, according to the Central Limit Theorem [109] the distribution is Normal. As population variances are unknown, and having no proofs that they are equal, according to Walpole *et al.* [85], the Student's t-distribution is used.

Let

$C_{\min}$	capacity of minimum cut from the min-cut graph model
C_{delay}	change in overall delay when Case B is changed to Case C
μ_1	population mean delay for final one-way road network, Case C
μ_2	population mean delay for initial one-way road network, Case B
d_0	difference between μ_1 and μ_2
$\bar{x}_1$	sample mean delay for final one-way road network, Case C
$\bar{x}_2$	sample mean delay for initial one-way road network, Case B
s_1	sample standard deviation for final one-way road network, Case C
s_2	sample standard deviation for initial one-way road network, Case B
n_1	sample size for final one-way road network, Case C
n_2	sample size for initial one-way road network, Case B
ν	degrees of freedom
α	level of significance

Since

$$C_{\min} = -168.0 \text{ seconds}$$

$$C_{\text{delay}} = \mu_1 - \mu_2 = d_0$$

therefore to prove that

$$C_{\min} = C_{\text{delay}}$$

is to prove that

$$\mu_1 - \mu_2 = -168.0 \text{ seconds}$$

There is therefore a need to test the null hypothesis (H_0) that $\mu_1 - \mu_2 = -168.0$ seconds against the alternative hypothesis (H_1) $\mu_1 - \mu_2 \neq -168.0$ seconds. Following the hypothesis testing given by Walpole *et al.* [109] with $\alpha = 0.001$, this will give:

1. $H_0: \mu_1 - \mu_2 = -168.0$ seconds
2. $H_1: \mu_1 - \mu_2 \neq -168.0$ seconds
3. Level of significance: $\alpha = 0.001$
4. Critical region:

Given that

$$\begin{aligned} \bar{x}_1 &= 409.3171, & s_1^2 &= 60.19074, & n_1 &= 41, \\ \bar{x}_2 &= 577.561, & s_2^2 &= 101.7524, & n_2 &= 41. \end{aligned}$$

$$\text{As degrees-of-freedom, } \nu = \frac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{[(\frac{s_1^2}{n_1})^2 / (n_1 - 1)] + [(\frac{s_2^2}{n_2})^2 / (n_2 - 1)]}, \text{ thus } \nu = 75.$$

Range of the critical region is therefore given by $t < -3.438$ and $t > 3.438$.

5. Computations:

$$\text{As } t = \frac{(\bar{x}_1 - \bar{x}_2) - d_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \text{ therefore } t = -0.12272.$$

6. Decision:

Since value of t is within the critical region hence H_0 is not rejected and there is not enough evidence to support and establish H_1 .

As H_0 is not rejected thus it **cannot** be proven that the change of delay from Case B to Case C is **not** -168.0 seconds. This therefore gives sufficient ground to state that *Hypothesis A* is correct.

Let $C(U, V-U) = \sum_{A \in U, B \in V-U} c_{AB}$ denote the total capacity of the arcs connecting sets U and $V-U$. With the min-cut graph model (see Section 6.2.3.2), the value of $C(U, V-U)$ is obtained and is shown by the previous section to be equal to the total change of overall delay by flipping the one-way roads in U . Hence the cut capacity $C(U, V-U)$, for a given traffic volume and network layout, is at its minimal if and only if the change of direction in roads in U results in the minimum total travel cost, that is *Hypothesis A* is correct.

One-way road network model

Both of the one-way road networks (Case B and Case C) have lower delay than the two-way road network (Case A) when congestion sets in (see Figure 7.3). There are two reasons for this: first, the one-way road network model is more effective under congestion; secondly, as reported in literature [60], under conditions of high traffic demand, one-way roads are faster than two-way roads (see Section 2.2.5), as the longer distances traversed in a one-way network are more than compensated for by higher average vehicular speeds.

The second reason given could play a part in showing that one-way road networks can decrease overall delay in two-way road networks. On the other hand, comparing delays of Case B with Case C, Case C has a much lower delay than Case B.

Re-examining the two networks, the only difference between Case B [see Figure 7.7(a)] and Case C [see Figure 7.7(b)] is the direction of road J . It is noticed that during simulation minutes, heavy traffic congestion occurs at road H and road K for Case B; this could be due to the heavy vehicle flow (see Table 7.2) from Zone 3 to Zone 2 as well as from Zones 1, 2 and 3 to Zone 5 (since road K is the only road from these 3 zones to Zone 5).

In addition, it is noticed that road *J* is barely used. Examining the road network for Case B, vehicles are linked to road *J* from nodes 13, 10 and 7. However node 7 is a sink hence no vehicle will be coming in from node 7 except from node 4; there is no reason for any vehicle to travel from node 4 down road *K* then up node *J* again to go to road *H*; vehicles can go directly to road *H* from node 7. On the other hand there is little reason for vehicles from source node 13 to head towards sink node 7 since node 13 and node 7 both belong to zone 4 (PIE); in fact Table 7.2 shows that no vehicle from zone 4 to zone 4. There is also little reason for vehicles to go to road *H* from node 13 via node 7 since road *H* is already congested. This therefore gives a case whereby road *H* and *K* are heavily congested while road *J* is underutilised.

Case C on the other hand having road *J* going from node 7 to node 12 gives an alternative route (besides road *K*) for vehicles travelling away from node 7 towards Zones 1, 2 and 5. This therefore reduced the congestion in both roads *H* and *K* and therefore produces a road network with a lower delay as compared to Case B. This means that Case C is a more efficient road network as compared to Case B.

Since Case C is the resultant road network of choice from the one-way road network model this thus shows that the one-way road network model is effective, thereby validating the one-way road network model.

With the results presented and discussed, the next chapter will move on to discuss some conclusions to be drawn as well as recommendations for this project.

VIII CONCLUSION AND UNSOLVED PROBLEMS

8.1 Overview

This chapter consists of three segments: the first segment states the conclusion of this thesis; the second segment lists the contributions of this thesis to knowledge; lastly, limitations of this project and recommendations for further work are discussed.

8.2 Conclusion

Based on the findings of this project, the conclusions drawn are given in the rest of this section.

In this thesis, the one-way road network problem is identified (see Chapter 2). A one-way road network model is then developed in Chapters 4, 5 and 6 of this thesis in order to solve the one-way road network problem. Generally speaking, this one-way road network model consists of two parts. First an algorithm, called the conversion algorithm, is developed in detail. Secondly a sub-model named minimum-cut graph model is used to refine the model for real world situations. The conversion algorithm is applied first to the WIJ case study, followed by the min-cut graph model, to illustrate how they work in succession.

The conversion algorithm makes use of the concepts of graph theory in order to convert any existing two-way road network into a viable one-way road network. By viewing the road network as a graph, it then attempts to find the digraph with the STTT, out of the numerous possibilities. The concepts of indegree and outdegree are used by the conversion algorithm to add a criterion of strong connectivity. This then helps reduce search space thus speeding up the search. Dijkstra's shortest path algorithm is next modified to find the shortest time taken to get from one node to another (instead of the shortest path between two nodes) and incorporated into the conversion algorithm. This helps calculate the total travel times of the feasible digraphs generated such that the eventual "optimal" one (with STTT) may be found.

With that, the conversion algorithm is developed for the one-way road network model, and it is capable of giving at least one initial (theoretically optimal) one-way road network from a given two-way road network.

The min-cut graph model is next employed to further refine the initial one-way road network obtained (from the conversion algorithm). The min-cut graph model was first developed by Cheng *et al.* [13] to solve specific cases of the NP-complete problem called the Module Orientation Problem. Being a very similar problem to the Module Orientation Problem, the min-cut graph model is applied to the one-way road network problem. Triangle graphs are formed based on the changes in delay times caused by changing the directions of various one-way roads. This gives a final graph model. A minimum cut is then applied to the final graph model. This thus gives a final one-way road network with a much reduced delay. The min-cut graph model is hence successfully applied to the one-way road network problem and used as the main mechanism for tailoring the one-way road network model to suit known traffic flow conditions.

The refined one-way road network model is next validated by means of simulation of the WIJ network. The case of the original two-way road network is compared against the final one-way road network. Results obtained are consistent with those obtained from the min-cut graph model. The refined one-way road network is shown to have a lower total delay time as compared to the original two-way road network. The one-way road network model is therefore validated by means of simulation.

8.3 Contributions to knowledge

In addition to the practical and usable conclusions of this thesis, the contributions to knowledge are as follows:

The one-way road network model

Prior to this investigation there are no clearly documented methodologies which provide a rational procedure for giving a suitable solution to the one-way road

network problem. The one-way road network model developed is thus able to address this issue and provide a logical method to arrive at possible one-way road networks.

Min-cut graph model

As mentioned in the previous section, the min-cut graph model was developed for the Module Orientation Problem; as such, its use was solely to solve problems of optimally orientating the various modules found on a PCB. In this project, the min-cut graph model is successfully adapted for use in the one-way road network model. Another novel element is the use of the min-cut graph model in conjunction with data obtained via computer simulation of the network.

Conversion algorithm

In order to apply the min-cut graph model to the one-way road network problem, an initial one-way road network layout is needed. The so-called “conversion algorithm” is developed for this purpose. It is capable of converting undirected graphs into feasible digraphs, the STTT digraph is then selected as the initial one-way road network model.

8.4 Limitations and recommendations for further work

In this section, the limitations of this project as well as the recommendations for further research are discussed. The discussion is categorised under the following subject areas:

1. Conversion algorithm
 - i. Graph decomposition
 - ii. Node condensation
 - iii. Strong connectivity
 - iv. Isomorphism
 - v. Twirling
2. Min-cut graph model

- i. Triangle graph
3. Simulation
- i. Evening peak hour
4. NP-completeness

8.4.1 Conversion algorithm

As explained in Chapter 4, the conversion algorithm will go through all possible digraphs from a given n -edged undirected graph as if it is an n -digit binary number. This means that should n be a very big number, the number of digraphs to be generated (which is 2^n) will be enormous. This will take up too much computer resources to compute and for very big values of n it will take an unacceptably long time (such as years) to arrive at an answer. Hence, other ways could be explored to make the algorithm even more efficient and capable of taking in graphs with large values of n which are considered in the following sections.

8.4.1.1 Graph decomposition

Suppose there is a node u of $G(V,E)$ such that E can be partitioned into two non-empty subsets of $E1$ and $E2$. The induced subgraphs of $G(E1)$ and $G(E2)$ have just one node u in common which is called an articulation point as stated by Bondy and Murty [114] and Tucker [115]. According to Tucker [115], the depth-first-search spanning tree algorithm can be used to identify such articulation points and decompose the graph. For example, node 4 in Figure 8.1(a) is an articulation point that will decompose the graph into two sub-graphs as seen in Figure 8.1(b).

Graph decomposition can therefore be applied on the given network/graph so as to obtain a smaller graph [as shown in Figure 8.1(a) and (b)]. With a smaller graph, the number of edges, n , is reduced. This means that the number of digraphs to be

generated is reduced as well. Hence, the resources and time needed to get a solution can be reduced.

8.4.1.2 Node condensation

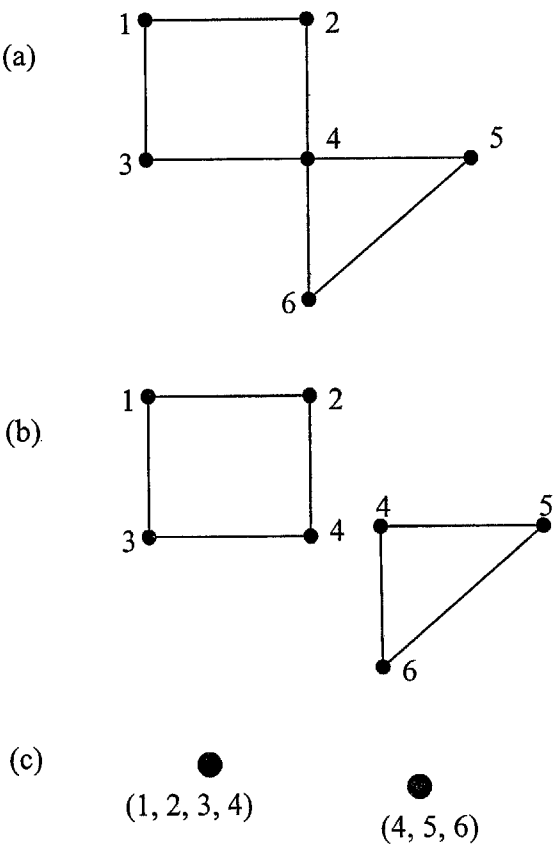


Figure 8.1: Graph manipulations using graph decomposition and node condensation

Given a set of nodes, the nodes can be condensed into one single node if there exists strong interconnection among the nodes of the set as shown by Cheng *et al.* [13]. Letting node u be a seed (where u is any generic node) of the cluster and P be a set of nodes such that $u \in P$. According to Liu [116], node condensation theorem states that if P satisfies the condition that there exists a minimum cut which places all nodes in P on the same side of the cut, then P can be condensed into a single node. The minimum cut of the condensed graph is also the minimum cut of the original graph.

For example, the graph given in Figure 8.1(a) is condensed into the graph as seen in Figure 8.1(c).

Hence, node condensation may be used to reduce the graph into a smaller graph. As explained previously, a smaller graph would be better for use by the conversion algorithm.

8.4.1.3 Strong connectivity

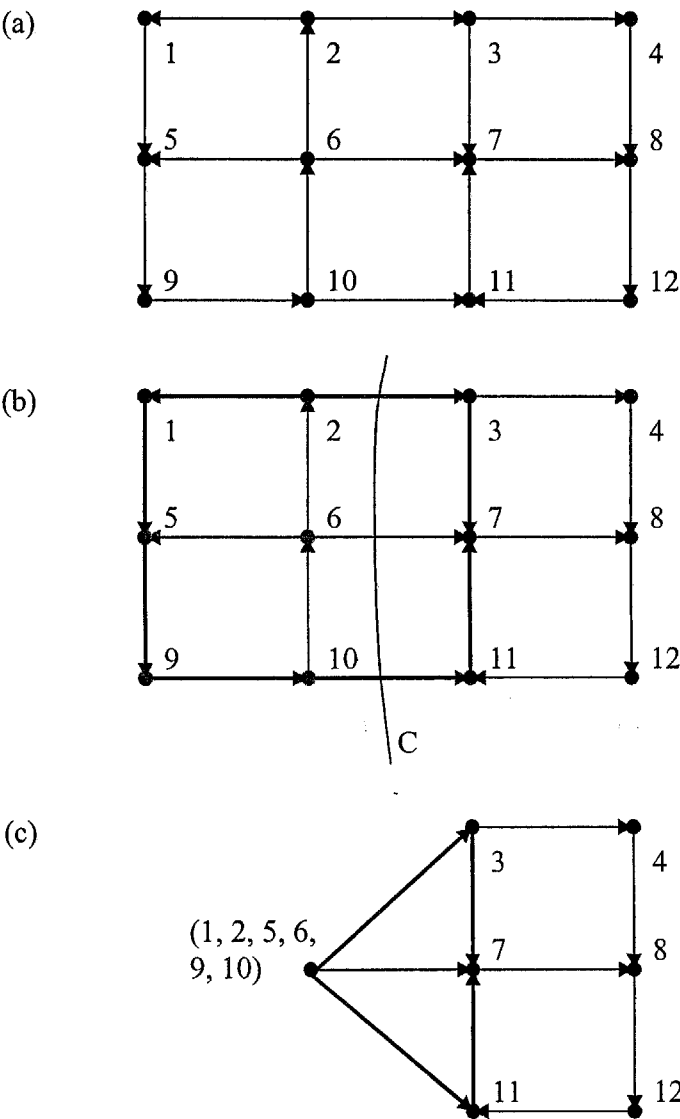


Figure 8.2: Graph manipulation for the Strong Connectivity of a Digraph

As discussed in Chapter 4, a digraph with nonzero indegree or outdegree is not strongly connected. This is due to the fact that some parts of the graph get “isolated” when nodes have nonzero indegree or outdegree. This is illustrated in Figure 8.2(a), all nodes in the given graph have edges directed away as well as towards them, that is all nodes have nonzero indegree and outdegree. However there is no single path from node 12 to node 1, neither is there one from node 4 to 5 nor one from node 7 to 8. As shown in Figure 8.2(b), the path $\{1, 5, 9, 10, 6, 2, 1\}$ forms a cycle within the digraph and there exist a cut, C , from the directed edges as shown in Figure 8.2(b). According to Frank [117] with the existence of such a cut in a digraph, the digraph will no longer be strongly connected.

On the other hand the existence of such a cut suggests the possibility of node condensation as discussed in Section 8.4.1.2 giving a super node $\{1, 2, 5, 6, 9, 10\}$ as shown in Figure 8.2(c). It can be seen that the super node has zero outdegree which is consistent with what was discussed previously in Chapter 4, which stated that should any node having zero indegree or outdegree the digraph would not be strongly connected.

The nature of strong connectivity is that there are many cycles therein. Therefore any part of the graph that consists of a circumscribing cycle can be condensed into a super node. This can be incorporated into the conversion algorithm and these super nodes can be checked to ensure that they do not have zero indegree or outdegree. This would more effectively test for strong connectivity of the digraphs generated by the conversion algorithm. This would then increase the efficiency of the algorithm since it would help rid off a larger number of non-strongly connected digraphs resulting in fewer digraphs for testing.

8.4.1.4 Isomorphism

Two graphs that contain the same number of nodes topologically connected in the same way are said to be isomorphs. The definition of isomorphism as given by Chartrand [118] and Robbins [119] is such that two graphs G and H with the same

number of nodes are said to be isomorphic if there is a permutation p of such that is in the set of graph edges $E(G)$ if and only if it is in the set of graph edges $E(H)$. Figure 8.3 shows two digraphs that are isomorphs.



Figure 8.3: Isomorphic graphs

In the conversion algorithm, isomorphs are generated and tested as if they are distinct graphs. Should isomorphs be identified, redundant testing would be removed and hence increase the efficiency of the algorithm. As stated by Luks [120], a way to identify isomorphs could be formulated and incorporated into the conversion algorithm. One simple way, in symmetrical configurations, is to fix the direction of one edge, say the edge at the top left hand corner of the graph. But it is worth investigating whether this will always work. Also a real road network is probably never isomorphic even if the solution is reversed, that is all road directions changed, in a symmetric network.

8.4.1.5 Twirling

In conducting experiments with the conversion algorithm in this project, it was found that the STTT digraphs for the 1 by 1 [see Figure 8.4(a)], 2 by 2 [see Figure 8.4(b)] and 3 by 3 graphs [see Figure 8.4(c)] contain twirls within each of the squares. A “twirl” in this case is defined as a cycle that does not circumscribe other nodes not forming the cycle, that is continuously directed back to its originating node. For example, in Figure 8.4(a) the edge from node 1 points to node 2 then to node 3 followed by node 4 and finally back to node 1 again, this forms a continuous cycle with no internal node and is therefore a twirl. The twirls in STTT digraphs seem to alternate, for example in Figure 8.4(c) the 1st square contains a twirl, followed by the

3rd square and then the 5th square and so on. It is noticed that the outer ring itself forms a “fill in” twirl as well [see Figure 8.4(b) and (c)].

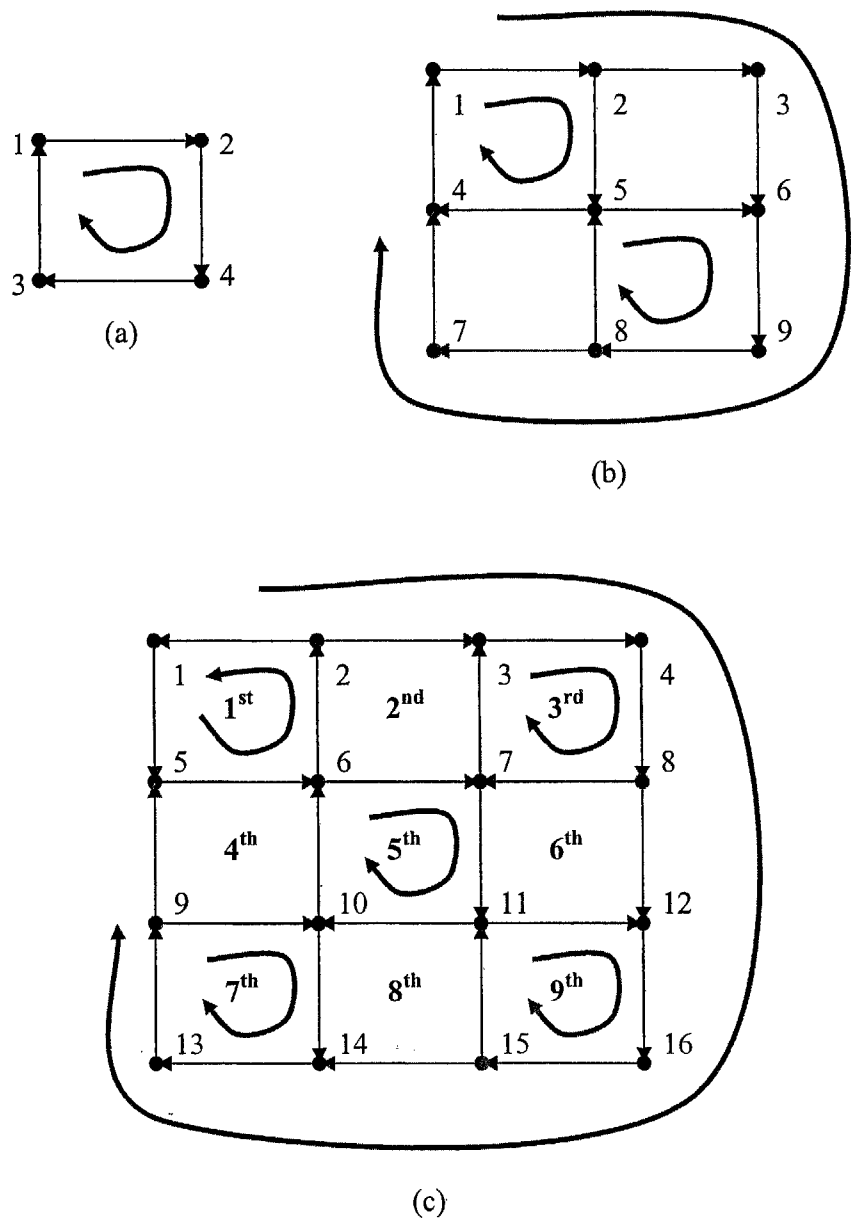


Figure 8.4: Twirls observed in the STTT graphs

By using heuristic reasoning, the STTT graphs generally contain twirls within alternate squares in the graph. Therefore the conversion algorithm may be modified such that instead of considering the original two possible directions of edges [see Figure 8.5(a)], it will consider the two possible twirlings of alternate squares [see

Figure 8.5(b)]. This will greatly reduce the computation time for the conversion algorithm as the number of possible digraphs to be considered is greatly reduced.

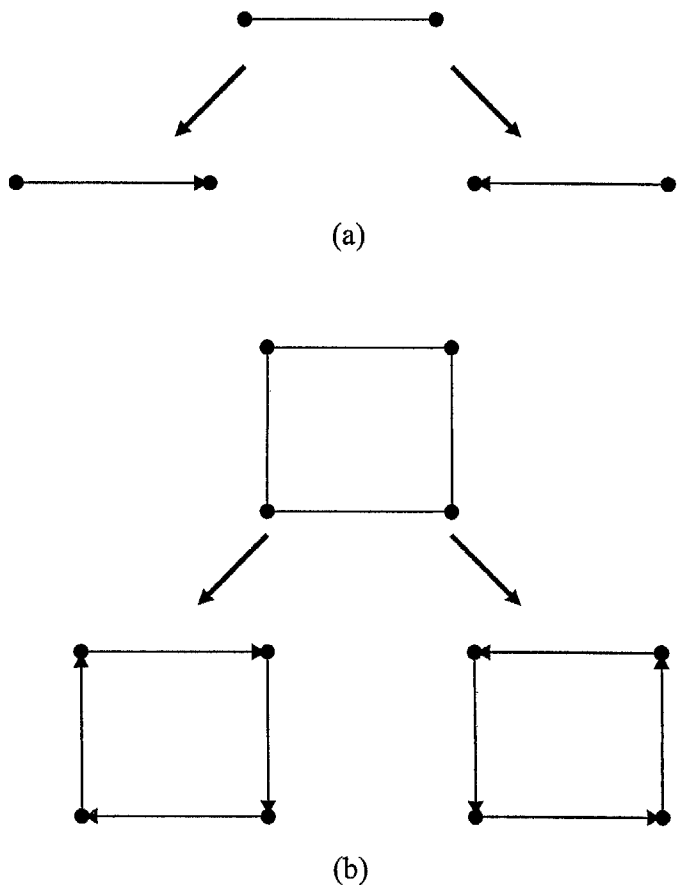


Figure 8.5: Two ways of twirling

For example, given an undirected 3 by 3 graph [see Figure 8.6(a)], the original conversion algorithm will consider all possible directions each of the edges can give. Since there are 24 edges present hence there will be 2^{24} possible digraphs to consider. Should the twirling of squares be considered, instead of edges, then there are only 6 (including the “outer fill-in square”) twirling to consider [see Figure 8.6(b)]. This means there are only 2^6 possible digraphs to consider instead of the original 2^{24} possible digraphs. Logically this will greatly reduce the computation time needed for the conversion algorithm to get the STTT graph, hence enabling the conversion algorithm to handle bigger graphs in a shorter time.

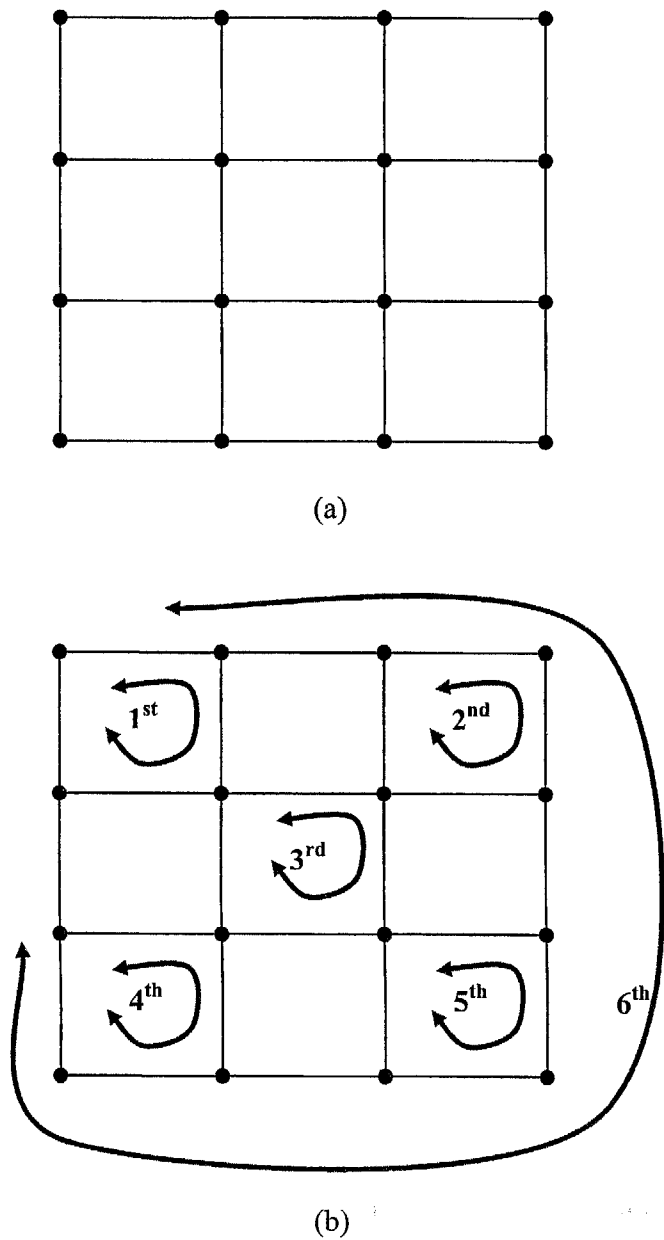


Figure 8.6: Considering 3 by 3 graph

8.4.2 Min-cut graph model

8.4.2.1 Triangle graph

For every intersection, a triangle graph is formed (see Section 6.2.2) from the two roads joining the intersection. However this would mean that should the intersection

contain more than two roads, for example three roads (A, B and C) then three triangle graphs: A and B, A and C, B and C, would have to be considered. The number of triangle graphs to be considered would be even more should more roads meet at that intersection.

Instead of using triangle graph, other graphs may be developed to incorporate all roads at the intersection and not just two roads at a time.

8.4.3 Simulation

8.4.3.1 Evening peak period

As with any traffic assignment problem, the solution for this one-way road network problem changes with a different set of input (in the form of an O-D matrix). In the case study for the WIJ network, the morning peak period has been chosen and used; in view of the evening peak period, the flows will be reversed and to complicate matters this reversed flows may not be a direct equivalent reversal (i.e. morning flow from B to A may not be the same amount as the evening flow from A to B). This implies that a different O-D matrix may be required for the evening peak period. Adding to that the network is one-way, the implication of this means that the shortest path from A to B is not the same as the shortest path from B to A. Drivers travelling from A to B in the morning using path b will not be able to take path b to travel from B to A in the evening since path b is one-way. Due to this if the evening O-D matrix differs greatly from the morning O-D matrix, the resultant road network obtained using the morning O-D matrix may not be the optimal road network for use for the evening period.

As such the one-way road network model could be expanded further such that it is capable of optimising the network for both morning and evening traffic flow.

8.4.4 NP-completeness

Both the Module Orientation problem [13] as well as the search for the minimum cut problem (employed in the min-cut graph model) are known to be NP-complete problems, as stated by Garey and Johnson [121]. As the conversion of a two-way road network to a one-way road network could be fashioned as a Module Orientation problem therefore it could be a NP-complete problem as well. Proofs however have to be developed to show that.

REFERENCES

- [1] Ortúzar, J.D.S. and Willumsen, L.G. (2001). *Modelling Transport*, 3rd edition. Wiley: Chichester, New York.
- [2] Raney, B., Voellmy, A., Vrtic, M. and Nagel, K. (2002). Towards a Microscopic Traffic Simulation of All of Switzerland. *P.M.A., Computational Science – ICCS 2002 Part I, Lecture Notes in Computer Science 2329*, Springer Heidelberg, pp. 371-380.
- [3] Ben-Joseph, E. (1995). Changing the residential street scene: adapting the shared street (Woonerf) concept to the suburban environment. *Journal of the American Planning Association*, Vol. 61, pp. 55-67.
- [4] Appleyard, D. (1981). *Livable Streets*. Berkeley, CA.: University of California.
- [5] Branston, D. (1976). Link capacity functions: a review. *Transportation Research* , Vol. 10, pp. 223-236.
- [6] Beimborn, E.A. (1995). *Inside the Blackbox, Making Transportation Models Work for Livable Communities*. Citizens for a Better Environment and the Environmental Defense Fund.
- [7] Brilion, W. and Blanke, H. (1990). Traffic Safety Effects from Traffic Calming. *Proceedings of Conference on Road Safety and Traffic Environment in Europe*. Gothenburg, Sweden.
- [8] Greenshield, B.D. (1935). A Study of Traffic Capacity. *Proceedings of Highway Research Board*, Vol. 14, pp. 448-477. (cited in Ortuzar and Willumsen [1], 2001)
- [9] TRB. (1997). *Quantifying Congestion: Final Report and User's Guide*, NCHRP Project 7-13, Transportation Research Board.
- [10] Stemley, J. (1998). One-way streets provide superior safety and convenience. *ITE Journal*, Vol. 68, No. 8, pp. 47-51.
- [11] *Downtown Transportation Plan* (2003). Administrative Report of Transportation and Traffic Committee, Vancouver, Canada, CC #5560-1.
- [12] Smith, M., White, M., Wylie, B.J.N., Cameron, G. and McArthur, D. (1994). The PARAMICS Parallel Traffic Simulator, *UK IT Forum Conference Digest*.

- [13] Cheng, C.K., Yao S.Z. and Hu, T.C. (1991). The Orientation of Modules Based on Graph Decomposition. *IEEE Transactions on Computers*, Vol. 40, No. 6, pp. 774-780.
- [14] Whitley, D. and Mathias, K. (1992). Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem, *Parallel Problem Solving from Nature-PPSN*, pp. 219-228. North Holland-Elsevier.
- [15] Dorigo, M. and Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, 1, 1.
- [16] Miglino, O., Menczer, D. and Bovet, P. (1994). A Neuro-Ethological Approach for the TSP: Changing Methaphors in Connectionist Models. *Journal of Biological Systems*, Vol. 2, Number 3, pp. 357-366.
- [17] Applegate, D., Bixby, R., Chvatal, V. and Cook, W. (1998). On the solution of traveling salesman problems. *Documenta Mathematica*, Extra Volume ICM.
- [18] Lowe, J.C. and Moryadas, S. (1975). *Special interaction: The geography of movement*. NY: Houghton Mifflin. (cited in Highway Capacity Manual [40], 2000)
- [19] Taaffe, E.J. and Gauthier, H.L.Jr. (1973). *Geography of transportation*. NJ: Prentice-Hall. (cited in Highway Capacity Manual [40], 2000)
- [20] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993). *Network Flows: Theory, Algorithms and Applications*. NJ: Prentice Hall.
- [21] Gross, J.L. and Yellen, J. (2003). *Handbook of Graph Theory*. CRC Press.
- [22] Diestel, R. (2000). *Graph Theory*. Springer-Verlag, NewYork.
- [23] Wang, C. (1997). *Graph Theory*. The Publishing House of Beijing Polytechnic University, Beijing.
- [24] Ford, L.R.J. and Fulkerson, D.R. (1962). *Flows in Networks*. Princeton University Press. (cited in Diestel [22], 2000)
- [25] Chalermsook, P., Fakcharoenphol, J., Nanongkai, D. (2004). A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, Louisiana

- [26] Libeskind-Hadas, R. and Liu, C.L. (1989). Solution to Module Orientation and Rotation Problems by Neural Computation Networks. *26th ACM/IEEE Design Automation Conference*, pp. 400-405.
- [27] Preas, B.T. and Karger, P.G. (1986). Automatic placement a review of current techniques. *Proceedings of 23rd Design Automat. Conf.*, pp. 622-629.
- [28] Yao, X., Yamada, M. and Liu, C.L. (1989). A new approach to the pin assignment problem. *IEEE Trans. Computer-Aided Design*, pp. 999-1006.
- [29] Hopfield, J.J. and Tank, D.W. Neural. (1985). Computation of Decisions in Optimization Problems. *Biology Cybernetics*, Vol. 52, pp 141-152.
- [30] Nobuo, F. and Junji, K. (1998). An Evolutionary Neural Network Approach for Module Orientation Problems. *IEEE Transactions on Systems, Man & Cybernetics: Part B*, Vol. 28, Issue 6, p849, 7p, Dec 1998.
- [31] Takefuji, Y. and Lee, K.C. (1992). A generalized maximum neural network for the module orientation problem. *Int. J. Electronics*, Vol. 72, No. 3, pp. 331-335.
- [32] Takefuji, Y., Lee, K.C. and Aiso, H. (1992). An Artificial Maximum Neural Network a Winner-Take-All Neuron Model forcing the State of the System in a Solution Domain, *Biology Cybernetics*, Vol. 67, pp. 243-251.
- [33] Yamada, M. and Liu, C.L. (1988). An Analytical Method for Optimal Module Orientation. *Proceedings of 1988 International Symp. On Circuits and Systems*, pp. 1679-1682.
- [34] Akçelik R. (1991). Travel Time Function for Transport Planning Purposes: Davidson's Function, its Time-Dependent Form and an Alternative Travel Time Function. *Australian Road Research*, Vol. 21 No. 3.
- [35] Bureau of Public Roads (1964). *Traffic Assignment Manual*. U.S. Dept. of Commerce, Urban Planning Division, Washington, D.C. (cited in Ortuzar and Willumsen [1], 2001)
- [36] Dowling, R., Singh, R.P. and Cheng, W.W. (1998). The Accuracy and Performance of Improved Speed-Flow Curves. *Annual Meeting, Transportation Research Board*, Washington, D.C.
- [37] Spiess, H. (1990). Conical Volume-Delay Functions. *Transportation Science*, Vol 24., No. 2.

- [38] Huber, M.J. (1982). *Traffic Flow Theory*. Transportation and Traffic Engineering Handbook, 2nd Ed., Prentice Hall, New Jersey.
- [39] Meng, Q, Lee, D-H. and Cheu, R. L. (2004). Simultaneously estimating O-D matrix and calibrating link travel cost functions from traffic counts. In *Proceedings of 8th International Conference on Applications of Advanced Technologies in Transportation Engineering, ASCE*, Beijing, China, 56-60.
- [40] *Highway Capacity Manual*. (2000). Washington, D.C. : Transportation Research Board, National Research Council, 2000.
- [41] Wardrop, J.G. (1952). Some theoretical aspects of road traffic research, *Proceedings of the Institute of Civil Engineers*, Part II, pp. 325-378. (cited in Ortuzar and Willumsen [1], 2001)
- [42] Abbas, M.M. and Bullock, D. (2003). An On-line Measure of Shockwaves for ITS Applications. *ASCE Journal of Transport Engineering*.
- [43] Andersen, G.J., Sauer, C. and Saidpour, A. (2002). *Analysis of Real World Driving Performance: A Model of Car Following*. PATH Project MOU 4220. Department of Psychology, University of California, Riverside.
- [44] Herman, R., Montroll, E.W., Potts, R.B. and Rothery, R.W. (1959). Traffic dynamics: Analysis of stability in car following. *Operation Research*, Vol. 7, pp. 86-106. (cited in Ortuzar and Willumsen [1], 2001)
- [45] Hughes, B.P. (1989). So You Think You Understand Gap Acceptance! *Australian Road Research Board*, Vol. 19, No. 3, pp. 195-204.
- [46] Karmeier, D.E. (1982). *Transportation and Traffic Engineering Handbook*, 2nd edition. Washington, D.C., USA: Institute of Transportation Engineers.
- [47] Kockelman, K.M. (2000). Modeling Traffic Flow-Density Relation: Accommodation of Multiple Flow Regimes and Traveler Types. *Transportation*.
- [48] McShane, W.R. and Roess, R.P. (1990). *Traffic Engineering*. Prentice Hall , Inc., Englewood Cliffs, New Jersey.
- [49] Algers, S. (2000). *Review of Micro-Simulation Models*. SMARTTEST (Simulation Modelling Applied to Road Transport European Scheme Tests) Project Report. Institute of Transport Studies, University of Leeds, UK.
- [50] Banks, J., Carson, H., Nelson, B. and Nicol, D. (2001). *Discrete-Event System Simulation*, 3rd ed. Prentice Hall, Upper Saddle River, New Jersey.

- [51] Wang, Y. and Prevedouros, P.D. (1998). Comparison of INTEGRATION, TSIS/CORSIM and WATsim in Replicating Volumes and Speeds on Three Small Networks. *Proceedings of 1998 Annual Meeting of Transportation Research Board*.
- [52] Barcelo, A. (1996). The parallelization of AIMSUN2 Microscopic Traffic Simulator for ITS applications. *Proceedings of 3rd World Conference on Intelligent Transport Systems*, pp. 14-18, Orlando.
- [53] Bloomberg, L. and Dale, J. (2000). A Comparison of the VISSIM and CORSIM Traffic Simulation Models On A Congested Network. *Transportation Research Record*, March 2000.
- [54] Bell, M. (1999). *Transportation Networks: Recent Methodological Advances*. London: Pergamon.
- [55] Helbing, D. and Treiber, M. (1999). Numerical Simulation of Macroscopic Traffic Equations. *Computing in Science and Engineering*, IEEE Computer Society.
- [56] Cameron, G.D.B., Wylie, B. J. N. and McArthur, D. (1994). PARAMICS – moving vehicles on a connection machine, *Proceedings of the 1994 conference on Supercomputing*, pp. 291-300. Washington, D.C., United States.
- [57] Cameron, G.D.B. and Duncan, C.I.D. (1996). PARAMICS – Parallel Microscopic Simulation of road traffic. *J. Supercomputing*, 10 (1):25.
- [58] Cetin, N., Burri, A. and Nagel, K. (2003). Parallel Queue Model Approach to Traffic Simulation. *Presented at TRB'03*.
- [59] Pursula, M. (1999). Simulation of traffic systems – an overview. *Journal of Geographic Information and Decision Analysis*, Vol. 3, No. 1, pp. 1-8.
- [60] Karagheuwfi, T. (1972). Traffic engineering succeeds in New York City. *ITE Traffic Engineering*, pp. 18-27.
- [61] Abdulhai, B., Sheu, J-B. and Recker, W. (1999). *Simulation of ITS on the Irvine FOT Area Using PARAMICS 1.5 Scalable Microscopic Simulator Phase 1: Model Calibration and Validation*. University of California at Irvine. California PATH Research Report UCB-ITS-PRR-99-12.

- [62] Akçelik, R. and Maher, M.J. (1977). Route control of traffic in urban road networks (Review and principles; Simulation experiments). *Transportation Science*, Vol. 11, No. 1, pp. 15-31.
- [63] Choa, F., Ronald, T. and Milam, R. (2000). HCM & CORSIM: Resolving the Differences. *Proceedings of ITE District Conference*.
- [64] Choa, F., Milam R. and Stanek, D. (2002). CORSIM, PARAMICS, and VISSIM: What the Manuals Never Told You. *Proceedings of the 2002 ITE Conference*.
- [65] Iravani, H. (2000). Integration of EMME/2 and CORSIM. *Proceedings of 15th International EMME/2 Users' Group Conference*, Vancouver, B.C.
- [66] Johnson, D.N. (1995). Economic analysis of the one-way grid in Roseburg, Oregon. *ITE Traffic Engineering*, November 1995, pp. 67-70.
- [67] Krause, J. (1986). Experience Problems and Strategies with Area-Wide 'Verkehrsberuhigung'. Planning and Transport Research and Computation. *Proceedings of Road Safety Meeting*, Vol. P284. Sussex, England. (cited in Ben-Joseph [3], 1995)
- [68] Nobel, J. and Jenks, M. (1989). *Safety and Security in Private Sector Housing Schemes: A Study of Layout Considerations*. London: Housing Research Foundation. (cited in Ben-Joseph [3], 1995)
- [69] Owen, L.E., Zhang, Y., Rao, L. and McHale, G. (2000). Traffic Flow Simulation Using CORSIM. *Proceedings of the 2000 Winter Simulation Conference*, pp. 1143-1147.
- [70] Taplin, J.H.E. (1999). Simulation Models of Traffic Flow. *Proceedings of Operational Research Society of New Zealand, 34th Annual Conference, ORSNZ'99. "OR in the New Millennium"* Hamilton NZ Operational Research Society of New Zealand (Inc.), pp. 175-184.
- [71] Central Western Precinct (2002). *Victoria Square Unification Transport Management Plan*. Final Report, August 2002. Adelaide, Australia.
- [72] Janssen, S.T. (1991). Road Safety in Urban Districts. Final Results of Accident Studies in the Dutch Demonstration Projects of the 1970s. *Traffic Engineering and Control*, Vol. 32, No. 6, pp. 292-296.
- [73] Kanazaki, N., Ohomori, Y. and Ishimura, S. (1984). The Use of Interlocking Block Pavements for the Reduction of Traffic Accidents. *Proceedings of Second*

- International Conference on Concrete Block Paving*. Delft, Holland. (cited in Ben-Joseph [3], 1995)
- [74] Nagel, K. (1996). Individual adaptation in a path-based simulation of the freeway network of North-Rhine Westfalia. *International Journal of Modern Physics C*, No. 7(6), pp. 883-892.
- [75] Taylor, I. and Willoughby, P. (1998). Junction Modelling in EMME/2. *Proceedings of 7th European EMME/2 Users Conference*, Oscar Faber, United Kingdom.
- [76] Meredith, J. and Prem, C. (2001). Central business district traffic circulation study: Kansas City Missouri. Institute of Transportation Engineers, *ITE Journal*, Vol. 71, No. 2, pp. 26-32.
- [77] Kraay, J.H. (1986). Woonerf and other Experiments in the Netherlands. *Built Environment*, Vol. 12, No. ½, pp.20-29. (cited in Ben-Joseph [3], 1995)
- [78] Polus, A. (1985). *Evaluation of the Characteristics of Shared Streets*. Haifa, Israel: Transportation Research Institute. (cited in Ben-Joseph [3], 1995)
- [79] Veeraragavan, A. and Robert, V.R. (2000). Evaluation of traffic management measures in accident reduction under mixed traffic. *Proceedings of the 16th International Cooperation on Theories and Concepts in Traffic Safety Workshop*, Soesterberg, Netherlands.
- [80] Brunner, D., Cross, G., McGhee, C., Levis, J. and Whitney, D. (1998). Toward Increased Use of Simulation in Transportation. *Proceedings of the 1998 Winter Simulation Conference*, pp. 1169-1175.
- [81] Ribeiro, V., Riedi, R., Crouse, M.S. and Baraniuk, R.G. (2000). Multiscale queuing analysis of long-range-dependent network traffic. *Proceedings of IEEE INFOCOM 2000*.
- [82] Choa, F., Ronald, T. and Milam, R. (2001). Recommended Guidelines for the Calibration and Validation of Traffic Simulation Models. *Proceedings of 2001 TRB Planning Applications Conference*.
- [83] Walters, G., El-Assar, H. and Zlotkowski, T. (1993). Clearing Bottlenecks Through Computer Simulation. *Proceedings of ITE District 6 Annual Meeting*, Las Vegas, Nevada.

- [84] Dix, A., Finlay, J., Abowd, G., Beale, R. (1998). *Human-Computer Interaction*, 2nd ed., Prentice Hall, Europe.
- [85] Walpole, R.E., Myers, R.H., Myers, S.L. and Ye, K. (2002). *Probability & Statistics For Engineers & Scientists*. Prentice Hall, New Jersey.
- [86] Ahuja, R.K., Mehlhorn, K., Orlin, J.B., and Tarjan, R.E. (1990). Faster algorithms for the Shortest Path Problem. *Journal of Association of Computing Machinery*, Vol. 37, pp. 213-223.
- [87] Cherkassky, B.V., Goldberg, A.V., and Radzik, T. (1993). *Shortest Paths Algorithms: Theory and Experimental Evaluation*. Technical Report 93-1480, Computer Science Department, Stanford University.
- [88] Dial, R.B., Glover, F., Karney, D., and Klingman, D. (1979) A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees. *Networks*, Vol. 9, pp. 215-248. (cited in Cherkassky *et al.* [87], 1993)
- [89] Dijkstra, E.W. (1959). A Note on Two Problems in Connection with Graphs. *Numeriche Mathematik*, Vol. 1, pp. 269-271. (cited in Cherkassky *et al.* [87], 1993)
- [90] Gallo, G. and Pallottino, S. (1988). Shortest Path Algorithm. *Annals of Operation Research*. Vol. 13, pp. 3-79.
- [91] Glover, F., Klingman, D., and Philips, N. (1985). A New Polynomially Bounded Shortest Paths Algorithm. *Operations Research*, Vol. 33, pp. 65-73.
- [92] Goldberg, A.V. and Radzik, T. (1993). A Heuristic Improvement of the Bellman-Ford Algorithm. *Applied Mathematics Letter*, Vol. 6, pp. 3-6.
- [93] Hung, M.H. and Divoky, J.J. (1988). A Computational Study of Efficient Shortest Path Algorithms. *Computers & Operations Research*, Vol. 15, pp. 567-576.
- [94] Mondou, J.F., Crainic, T.G., and Nguyen, S. (1991). Shortest Path Algorithms: A Computational Study with the C Programming Language. *Computers & Operations Research*, Vol. 18, pp. 767-786.
- [95] Yin, R., Tao, Y., Xie, R. and Sheng, X. (1999). *Data structure – illustrated using O-O method and C++*. The Publishing House of Tsinghua University, Beijing.
- [96] Zhan, F.B. and Noon, C.E. (1996). Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, Vol. 32, No. 1, pp. 65-73.

- [97] Zhan, F.B. (1997). Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. *Journal of Geographic Information and Decision Analysis*. Vol 1, No. 1.
- [98] Sakellariou, R. (1994). A Computational Study of Parallel Algorithms for the All-Pairs Shortest Path Problem. In E. A. Lipitakis (ed.) *Proceedings of the 2nd Hellenic-European Conference on Mathematics and Informatics*. Athens, Sep. 1994. Hellenic Mathematical Society, pp. 839-846.
- [99] Crauser, A., Mehlhorn, K., Meyer, U., and Sanders, P. (1998). A Parallelization of Dijkstra's Shortest Path Algorithm. *Lecture Notes in Computer Science*. Vol. 1450, pp. 722.
- [100] Drew, D.R. (1968). *Traffic Flow Theory and Control*. McGraw-Hill, New York. (cited in Highway Capacity Manual [40], 2000)
- [101] FHWA. (1982). Application of Traffic Simulation Models. *Proceedings of Conference on the Application of Traffic Simulation Models, June 1981*. Federal Highway Administration, US Department of Transport. Techshare Report FHWA-TS-82-207.
- [102] Klein, U., Schulze, T. and Straßburger, S. (1998). Traffic Simulation Based on the High Level Architecture. *Proceedings of the 1998 Winter Simulation Conference*, pp. 1095-1103.
- [104] Payne, H.J., Thompson, S. and Chang, G-L. (1997). Calibration of FRESIM to Achieve Desired Capabilities. *Proceedings of 76th Annual Meeting of Transportation Research Board*, Washington, D.C.
- [105] Rakha, H.A. and Van Aerde, M.W. (1996). Comparison of simulation modules of TRANSYT and INTEGRATION models. *Transportation Research Record*, Vol. 1566, pp. 1-7.
- [106] Gardes, Y., May, A.D., Dahlgren, J. and Skabardonis, A. (2002). Freeway Calibration and Application of the PARAMICS Model. *Proceedings of 81st Annual Meeting of TRB*, Washington, D.C.
- [107] Stroustrup, B. (2003). *The C++ Programming Language*, 3rd edition. Pearson Education, Indianapolis.
- [108] Martin, R.C. (2002). *Designing Object Oriented C++ Applications Using The Booch Method*. Addison-Wesley, Chicago.

- [109] Liu, H.X., Lianyu, C. and Recker, W. (2003). Development of the Capability-Enhanced PARAMICS Simulation Environment. *Proceedings of Transportation Research Board Annual Meeting*.
- [110] Toint, P. and Dehoux, P. (1991). Some Comments on Dynamic Traffic Modeling in the presence of Advanced Driver Information System. *Proceedings of the DRIVE Conference*, pp. 964-980, Brussels.
- [111] Wylie, B. J. N., Cameron, G., White, M., Smith, M. and McArthur, D. (1993). PARAMICS: Parallel Microscopic Traffic Simulator. *Proceedings of EuroCM User Group Meeting*, Paris, France.
- [112] Sacks, J., Rouphail, N.M., Park, B.B. and Thakuriah, P.V. (2000). Statistically-Based Validation of Computer Simulation Models in Traffic Operations and Management. *Journal of Transportation and Statistics*, Vol. 5, No. 1.
- [113] Lee, D.-H., Yang, X. and Chandrasekar, P. (2001) Parameter calibration for Paramics using genetic algorithm. *In Proceedings of the 80th Annual Meeting of TRB* (CD-ROM), Washington, D.C., USA.
- [114] Bondy, J.A. and Murty, U.S.R. (1979). *Graph Theory with Applications*. New York: North Holland. (cited in Cheng *et al.* [13], 1991)
- [115] Tucker, A. (1980). *Applied Combinatorics*. New York: Wiley, pp. 109-112. (cited in Cheng *et al.* [13], 1991)
- [116] Liu, C.L. (1968). *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill. (cited in Zhan [97], 1997)
- [117] Frank, A. (1981). How to make a Digraph Strongly Connected. *Combinatorica* 1(2), pp. 145-153. (cited in Diestel [22], 2000)
- [118] Chartrand, G. (1985). Isomorphic Graphs. *Introductory Graph Theory*. New York: Dover, pp. 32-40.
- [119] Robbins, H.E. (1939). A Theorem on Graphs, with an Application to a Problem of Traffic Control. *The American Mathematical Monthly*, Vol. 46, No. 5, pp. 281-283. (cited in Diestel [22], 2000)
- [120] Luks, E.M. (1980). Isomorphism of Bounded Valence can be Tested in Polynomial Time. *Proceedings of 21st Annual Symposium on Foundations of Computing*. IEEE Press, pp. 42-49. (cited in Wong [23], 1997)

- [121] Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979. (cited in Diestel [22], 2000)

APPENDIX A: List of Publications

1. Tan, K.C. & Lew, S.C. (2004). Fashioning One-Way Urban Road Network Design as Module Orientation Problem. *Proceedings of 10th International Conference on Urban Transport and The Environment in the 21st Century, Dresden, Germany.*
2. Tan, K.C. & Lew, S.C. (2005). Simulation and Graph Model in the One-way Road Network Problem. *Proceedings of 16th IASTED (International Association of Science and Technology for Development) International Conference on Modelling and Simulation, Cancun, Mexico.*
3. Lew, S.C. & Tan, K.C. (2005). An Algorithm for converting undirected graph to feasible digraphs. *Proceedings of 16th IASTED (International Association of Science and Technology for Development) International Conference on Modelling and Simulation, Cancun, Mexico.*

APPENDIX B: Complexity of problems

B.1 Complexity of problems

According to Garey and Johnson [121], a hierarchy for complexity of problems on finite set of inputs has been defined. This section will talk about the class NP-complete from this hierarchy. This section also touches on the Big O notation that describes the complexity of a solution to a given problem. The Big O notation enables comparison of various solutions to the same problem.

B.1.1 NP-complete

NP refers to non-deterministic polynomial time. To understand NP-complete better the terms non-deterministic machines and polynomial time are explained.

Non-deterministic machine

A non-deterministic machine is a machine that when given a particular input will map it to its output as a relation, as opposed to a function. As such, a non-deterministic machine may give more than one output with just a given input.

Polynomial time

Polynomial time refers to the time period taken for a machine to find a solution to the problem. This period is of a polynomial function of the size of the problem. The size of the problem normally refers to number of inputs of the problem. For example for a problem with n size, solving it in polynomial time means it can be solved in n^k units of time, where k is a constant.

NP-complete

A problem is NP if it can be solved using a non-deterministic machine in polynomial time. Within the class of NP, a subclass of problems that are hardest to solve is formed. This subclass is known as NP-complete.

B.1.2 Big O notation (or Order of complexity notation)

Big O notation also known as the Landau's symbol is a symbolism used to describe the asymptotic behaviour of a function/solution. More exactly, it is used to describe the asymptotic upper bound behaviour of a function/solution; in other words the worst case of the function/solution. It is represented by the symbol "O" and hence the name Big O notation.

Big O notation is used to describe the order of complexity of a function/solution. However, in a more practical sense, by putting order of complexity of function/solution into a standard notation it allows different solutions to the same problem to be compared thereby allowing a more efficient solution to be obtained. This is illustrated using the following example. Given two algorithms which are used to solve a same problem, *A* with complexity of $O(n)$ while *B* has a complexity of $O(\lg n)$. The asymptote behaviours of both algorithms would be as given in Figure B.1.

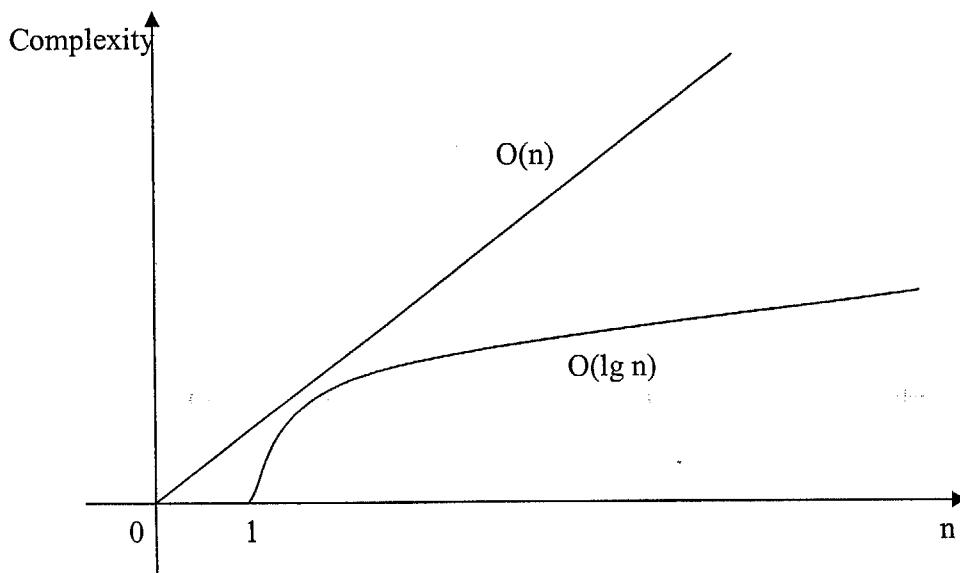


Figure B.1: Asymptote behaviours of $O(n)$ and $O(\lg n)$

A having a linear complexity of $O(n)$ would result in a straight line. *B* having a logarithmic complexity of $O(\lg n)$ would result in the curve as shown. It can be seen that from the graph, since the graph for algorithm *B* is always lower than that of *A*,

order of complexity for B is always lower than that of A . B would be a more efficient (in terms of complexity) algorithm to use.

APPENDIX C: Dijkstra Algorithm

A formal description of Dijkstra's shortest path algorithm as given by Dijkstra [89] is given below:

Procedure Dijkstra

Begin

 Initialise sorted list, L , and put s , the source node, into L .

 Initialise P , matrix for storing previous nodes of the shortest path.

 Initialise d , matrix for storing current shortest paths, and set all entries to infinity.

 Set $d[s]$ to 0.

 Set $P[s]$ to 0.

 While L is non-empty

 Set current node, X , to $L[1]$.

 For all nodes, Y , adjacent to X

 If $d[X] + \text{weight of edge } (X,Y) < d[Y]$

 Set $d[Y] = d[X] + \text{weight of edge } (X,Y)$.

 Set $P[Y] = X$.

 Add Y to L .

 Endfor.

 Remove X from L .

 Sort L in ascending order.

 Endwhile.

End.

In applying this algorithm on any graph with n nodes, it has to be applied n times with a different node acting as the source node. According to Cherkassky *et al.* [87], an all-to-all shortest path matrix will be obtained (see Figure C.1). This matrix will be n by n in dimension. The matrix's interpretation is similar to that of the adjacency matrix, however the values in the matrix will represent the shortest path instead of adjacency.

That is the value in row i column j will state the length of the shortest path from node i to node j . For example in Figure C.1, [row 1, column 2] gives a value of 1 to indicate that the shortest path from node 1 to node 2 is of length 1.

$$\begin{pmatrix} 0 & 1 & 4 & 2 \\ 1 & 0 & 2 & 4 \\ 3 & 6 & 0 & 2 \\ 1 & 1 & 2 & 0 \end{pmatrix}$$

Figure C.1: All-to-all shortest path matrix

For [row n , column n] it refers to the shortest path from node n to node n . Dijkstra's algorithm however is able to identify that it is actually referring to the same node and thus will place a value of zero at that particular location in the matrix.

APPENDIX D: Type of movement/turn present

D.1 Need for the previous node

The time spent at the intersection is determined by the type of turn present. In order to do so the type of turn the vehicles make must be known. By knowing the current node and the next node however is not enough to know what the kind of turn the vehicle is making. This is shown by the example given below:



Figure D.1: From i to j

Figure D.1 shows a vehicle movement from node i to j, however it does not tell what sort of movement it is making (whether straight through, right or left turn).

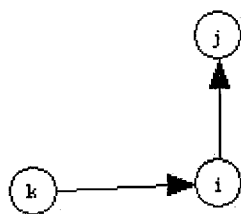


Figure D.2: From k to i to j

Figure D.2 shows the previous node from which the vehicle came from. It is now apparent that a left turn is made here. Thus in order to know the type of movement the vehicle is making, it is essential to know the current node, next node as well as the previous node.

APPENDIX E: Verifying the conversion algorithm

E.1 Results of the conversion algorithm

Results of the conversion algorithm on the simple graph example are as given:

0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 0 0 1
1 0 0 0 0 0
0 1 0 1 0 0
0 0 0 0 1 0
605.70

0 1 0 0 0 0
0 0 1 0 1 0
0 0 0 0 0 1
1 0 0 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
605.70

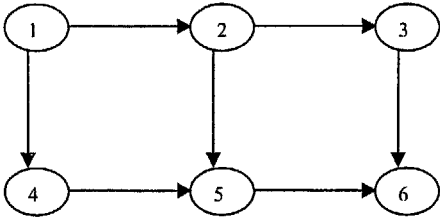
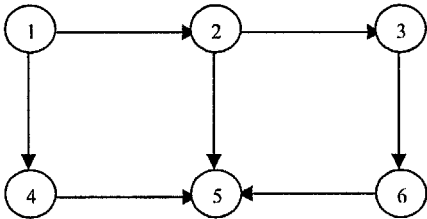
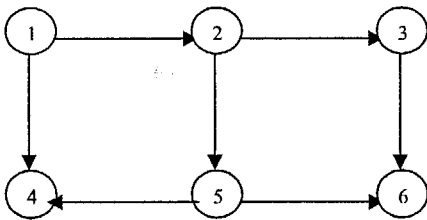
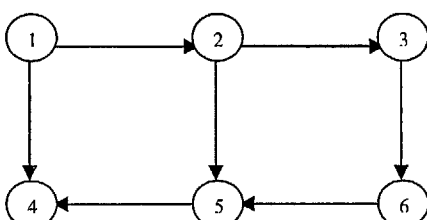
The adjacency matrix of the digraph obtained is shown. The STTT for each digraph is given at the bottom of their adjacency matrix which is 605.70 seconds in both cases. The graphical representation of the digraphs can be found in Figures 5.13(b) and 5.13(c).

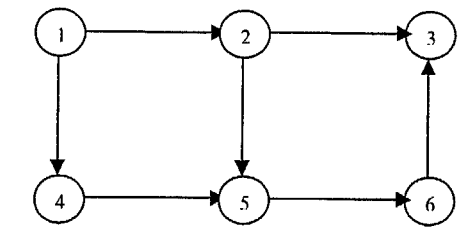
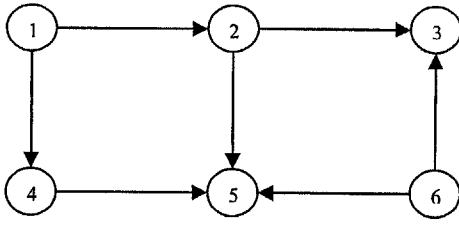
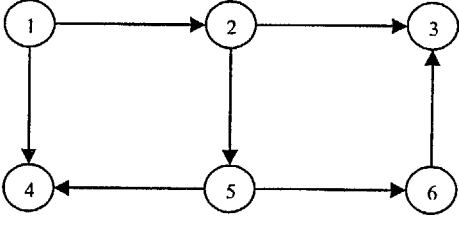
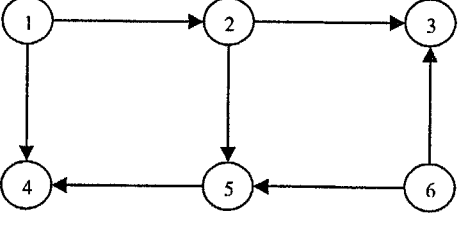
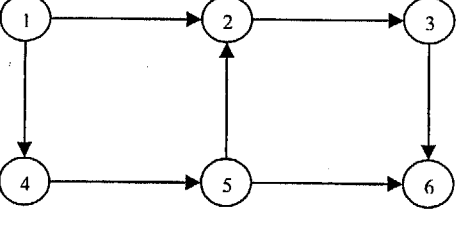
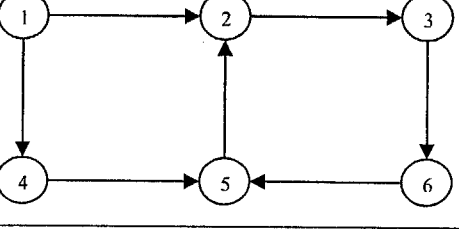
E.2 128 possible digraphs from the simple graph example

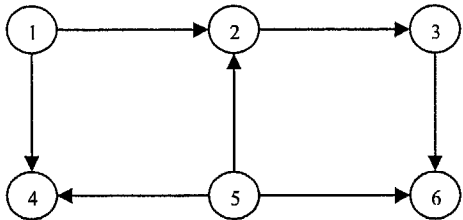
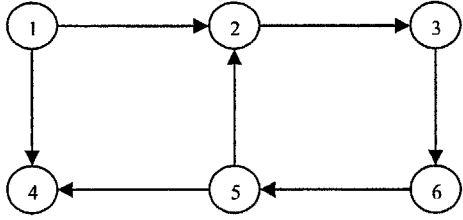
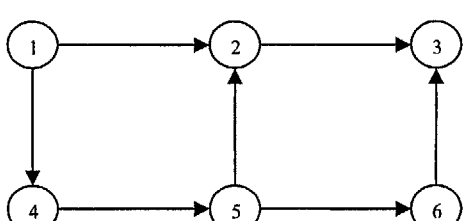
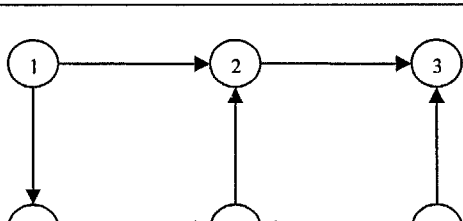
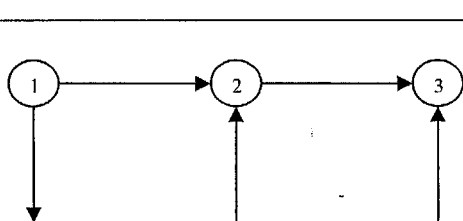
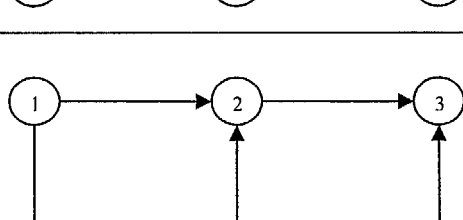
All the 128 possible digraphs from the given simple graph example are shown in the Table E.1 given below. Each of the binary representation of the digraph is given,

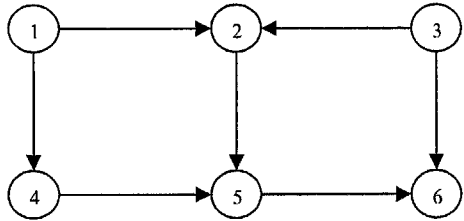
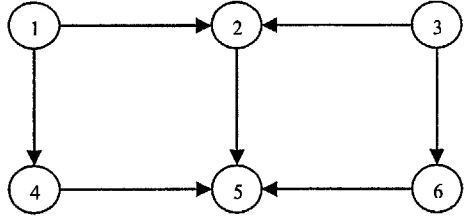
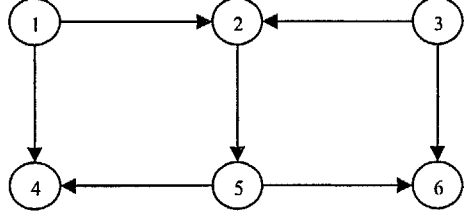
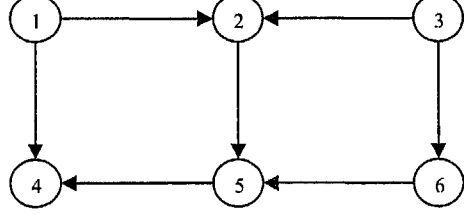
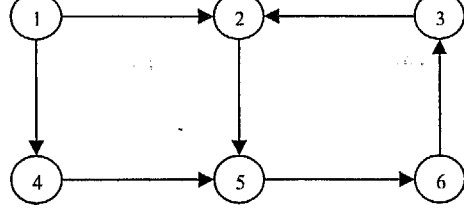
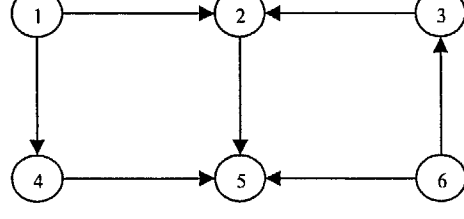
together with its graphical form as well as a column for indicating the strong connectivity of the digraph. The STTTs for all strongly connected graphs are given the table as well. As shown in the table, only 6 of the 128 digraphs are strongly connected and hence only these 6 are compared for the digraph with the minimum STTT. Results of comparison show 2 digraphs with the minimum STTT, both with the value of 605.70 seconds.

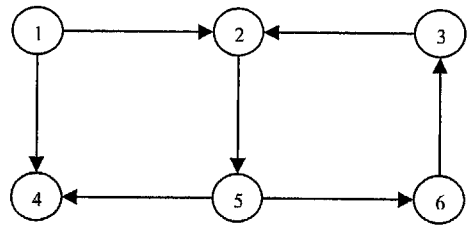
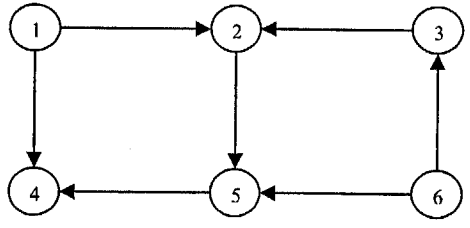
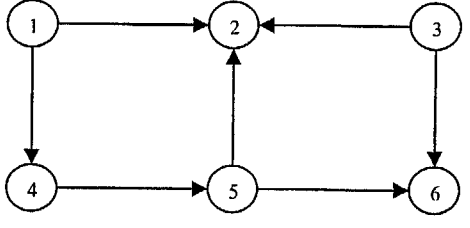
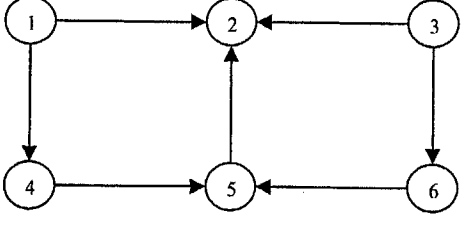
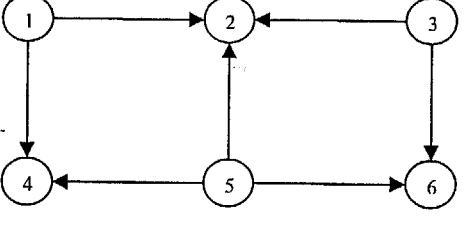
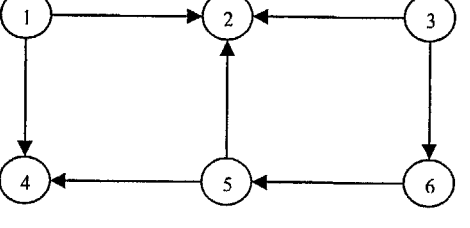
Table E.1: 128 possible digraphs

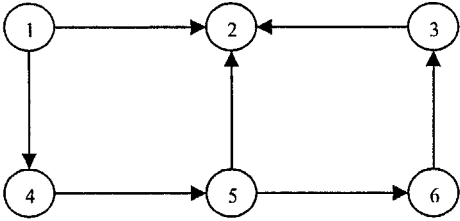
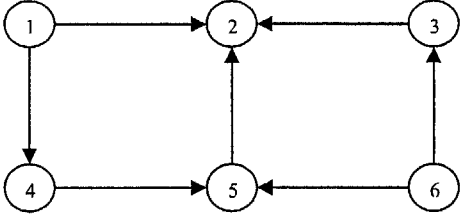
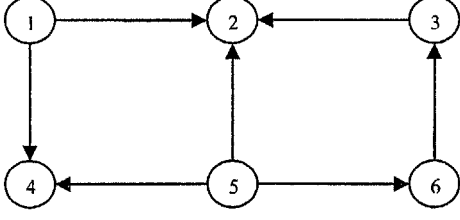
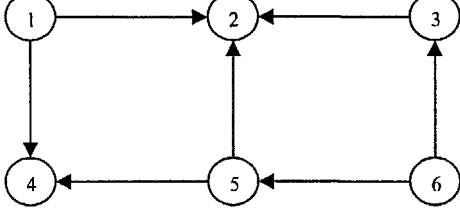
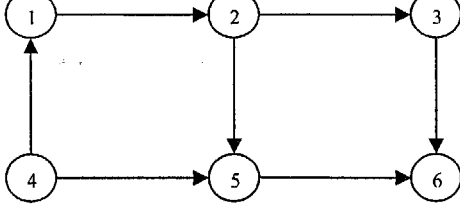
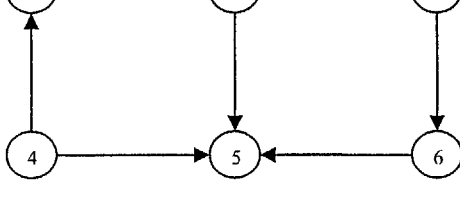
Binary representation	Digraph	Strong connectivity	Value of STTT
0000000		No	N.A.
0000001		No	N.A.
0000010		No	N.A.
0000011		No	N.A.

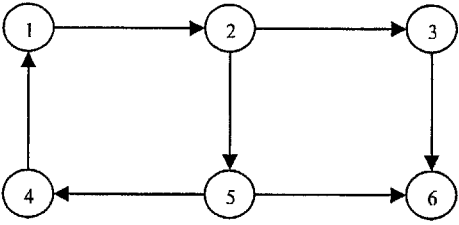
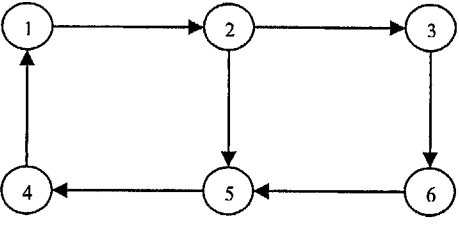
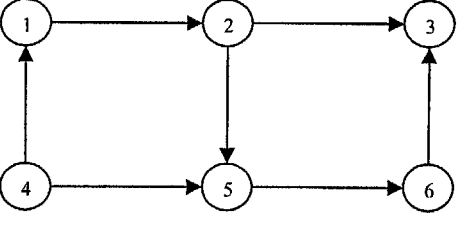
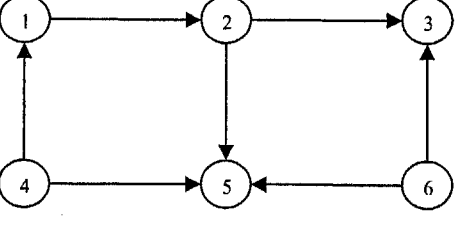
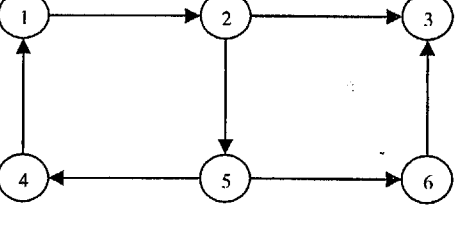
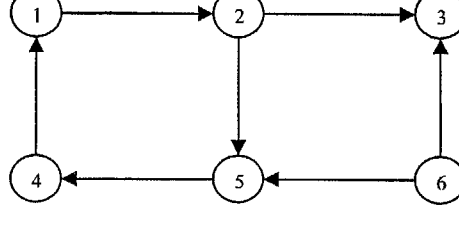
0000100		No	N.A.
0000101		No	N.A.
0000110		No	N.A.
0000111		No	N.A.
0001000		No	N.A.
0001001		No	N.A.

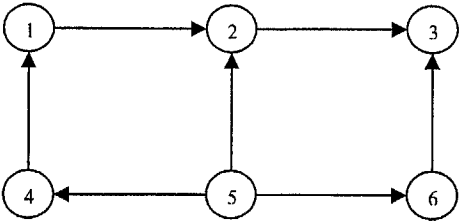
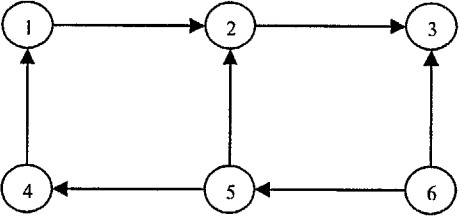
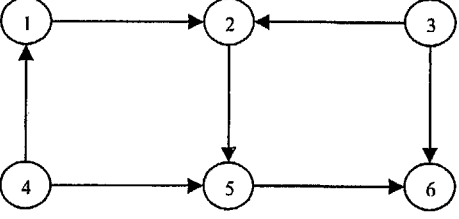
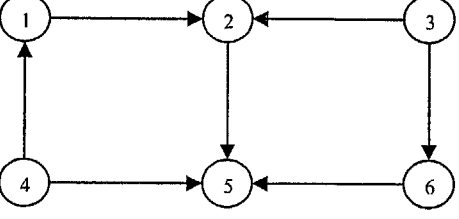
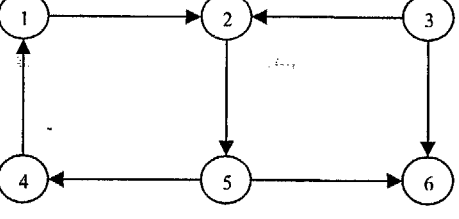
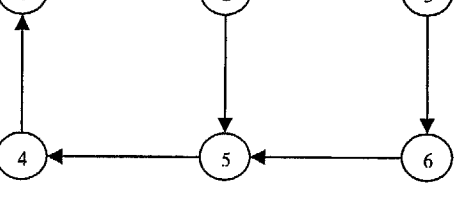
0001010		No	N.A.
0001011		No	N.A.
0001100		No	N.A.
0001101		No	N.A.
0001110		No	N.A.
0001111		No	N.A.

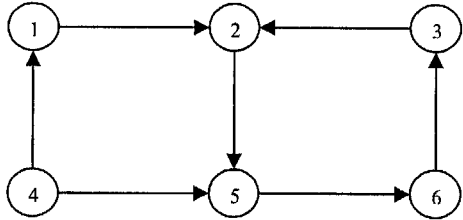
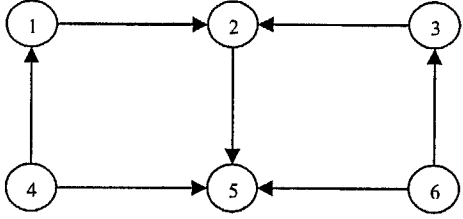
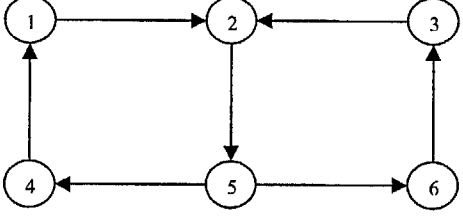
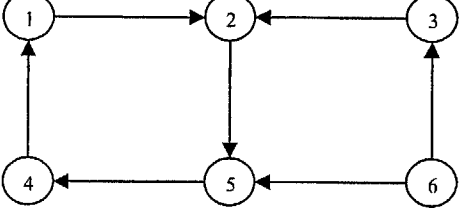
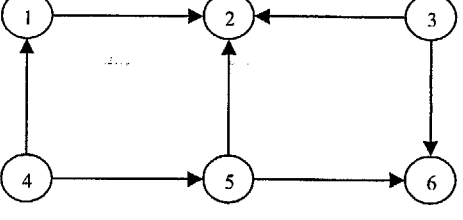
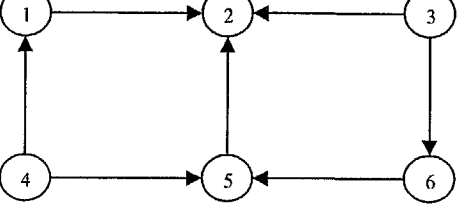
0010000		No	N.A.
0010001		No	N.A.
0010010		No	N.A.
0010011		No	N.A.
0010100		No	N.A.
0010101		No	N.A.

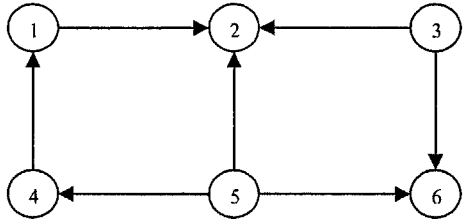
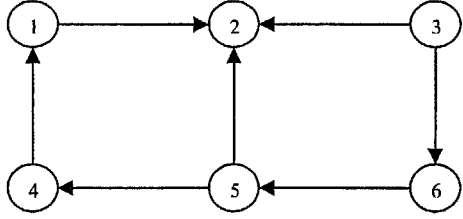
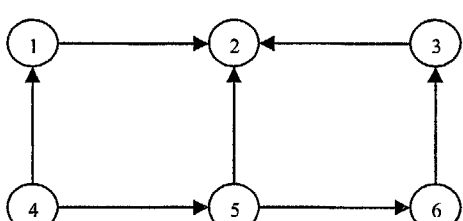
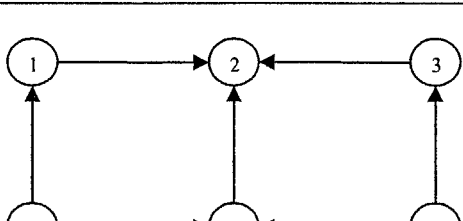
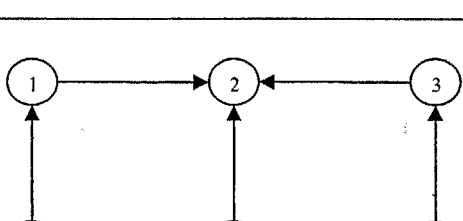
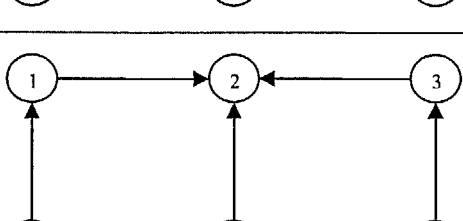
0010110		No	N.A.
0010111		No	N.A.
0011000		No	N.A.
0011001		No	N.A.
0011010		No	N.A.
0011011		No	N.A.

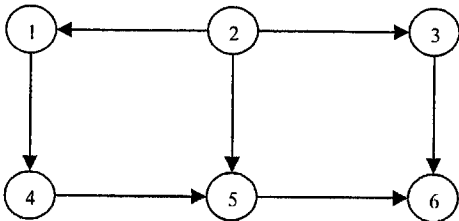
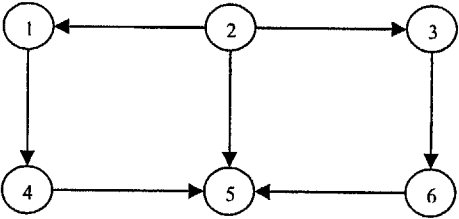
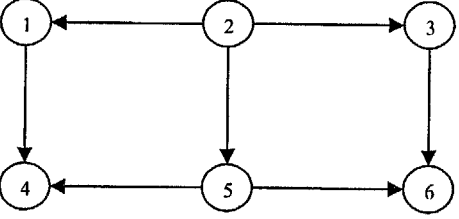
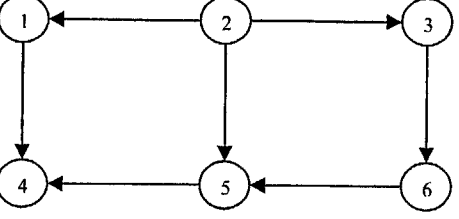
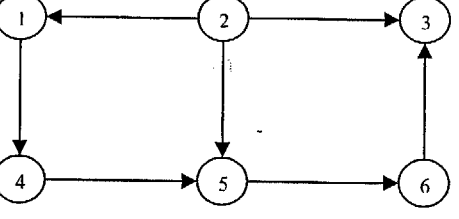
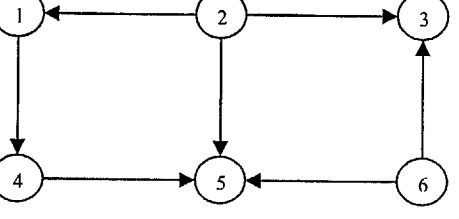
0011100		No	N.A.
0011101		No	N.A.
0011110		No	N.A.
0011111		No	N.A.
0100000		No	N.A.
0100001		No	N.A.

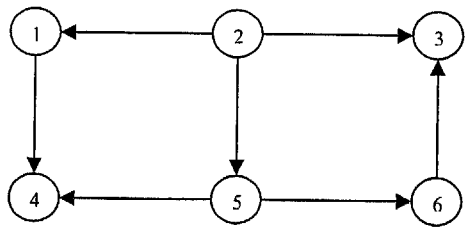
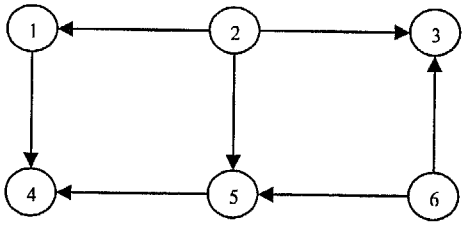
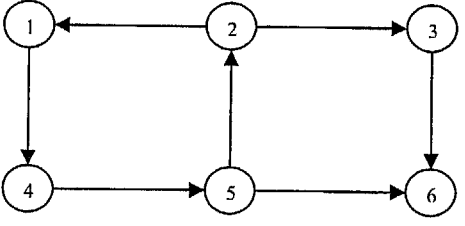
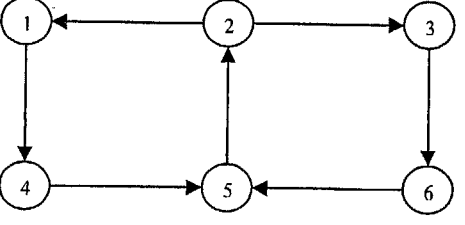
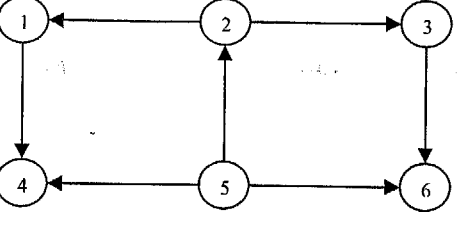
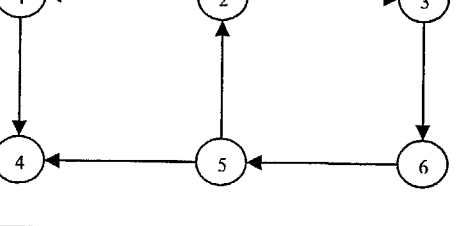
0100010		No	N.A.
0100011		Yes	607.15 sec
0100100		No	N.A.
0100101		No	N.A.
0100110		No	N.A.
0100111		No	N.A.

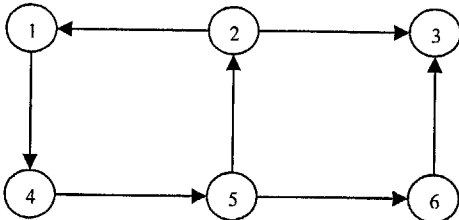
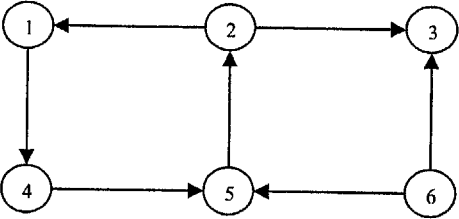
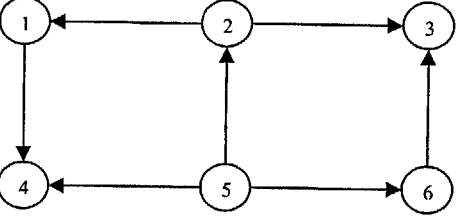
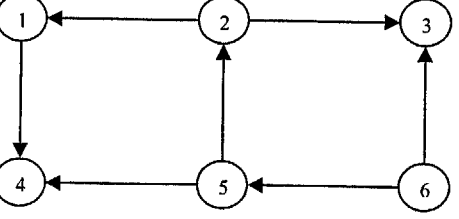
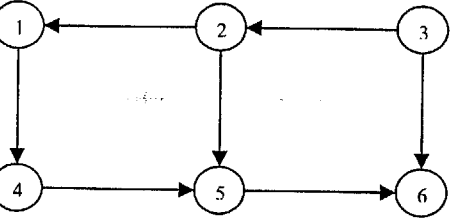
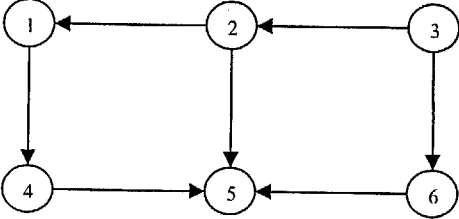
0101110		No	N.A.
0101111		No	N.A.
0110000		No	N.A.
0110001		No	N.A.
0110010		No	N.A.
0110011		No	N.A.

0110100		No	N.A.
0110101		No	N.A.
0110110		Yes	688.41 sec
0110111		No	N.A.
0111000		No	N.A.
0111001		No	N.A.

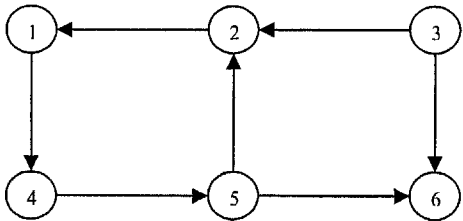
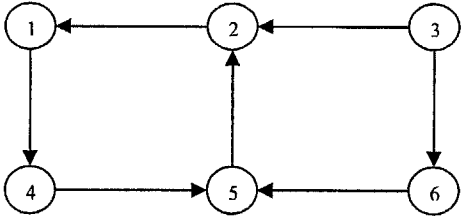
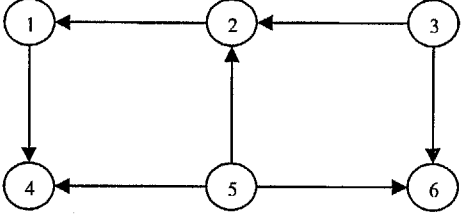
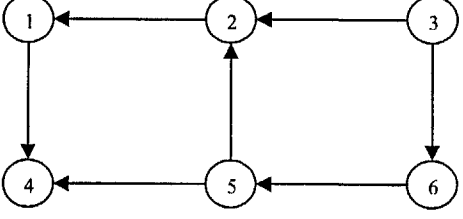
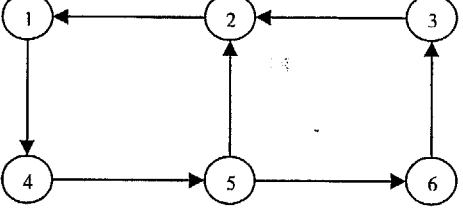
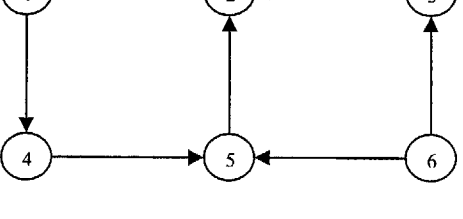
0111010		No	N.A.
0111011		No	N.A.
0111100		No	N.A.
0111101		No	N.A.
0111110		No	N.A.
0111111		No	N.A.

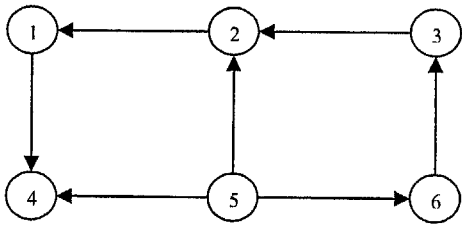
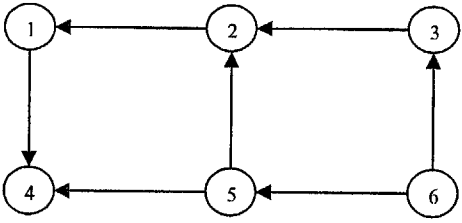
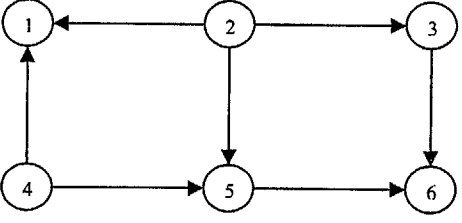
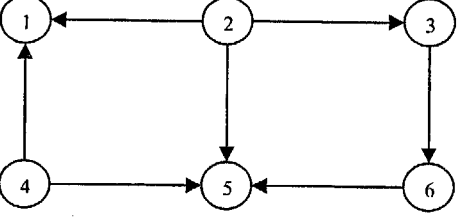
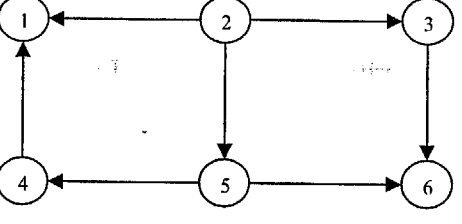
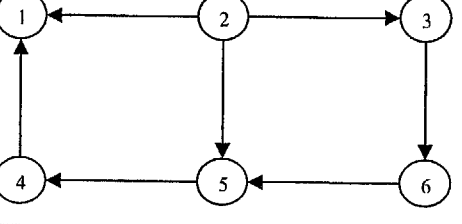
1000000		No	N.A.
1000001		No	N.A.
1000010		No	N.A.
1000011		No	N.A.
1000100		No	N.A.
1000101		No	N.A.

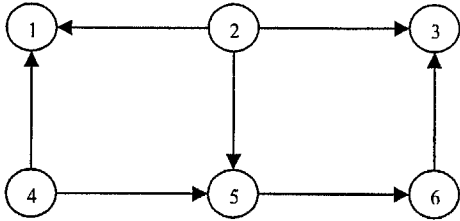
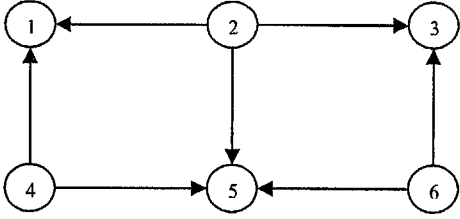
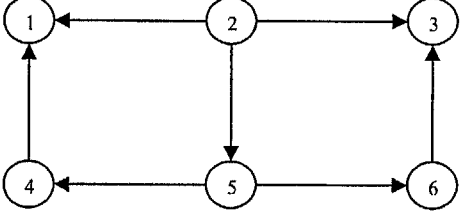
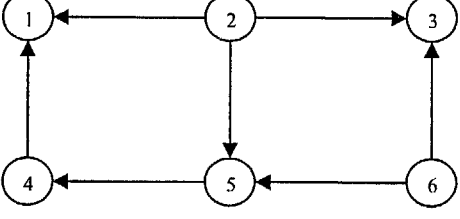
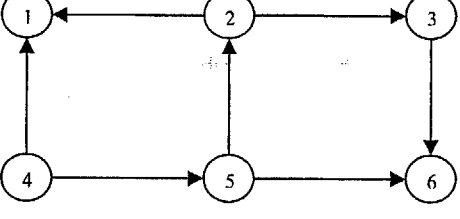
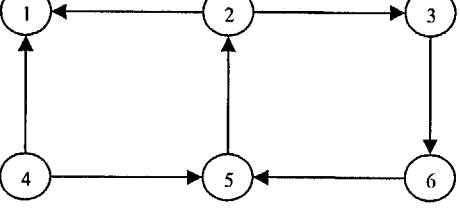
1000110		No	N.A.
1000111		No	N.A.
1001000		No	N.A.
1001001		Yes	688.41 sec
1001010		No	N.A.
1001011		No	N.A.

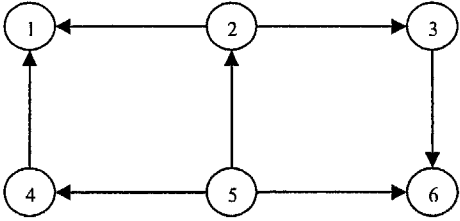
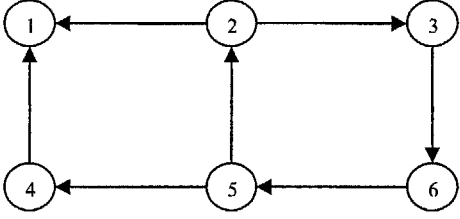
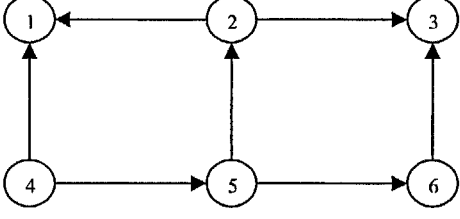
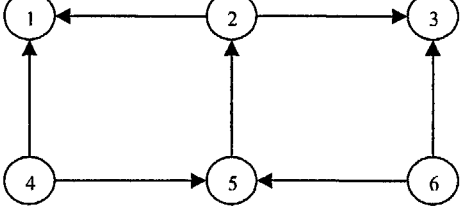
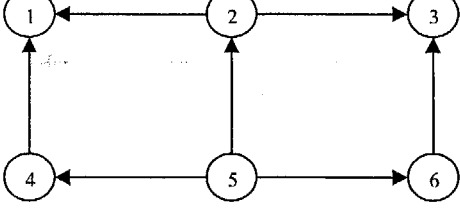
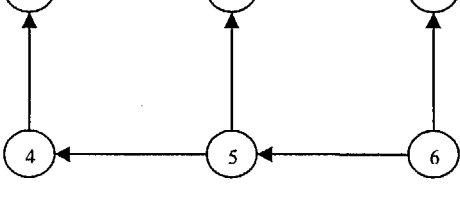
1001100		No	N.A.
1001101		No	N.A.
1001110		No	N.A.
1001111		No	N.A.
1010000		No	N.A.
1010001		No	N.A.

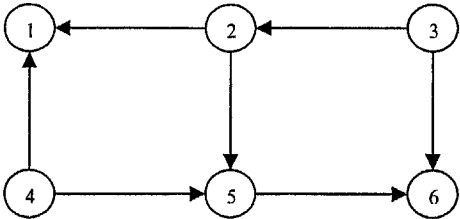
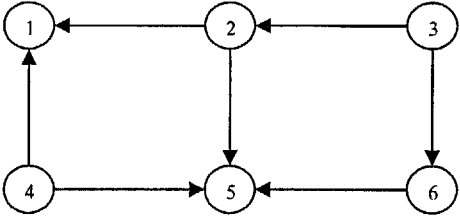
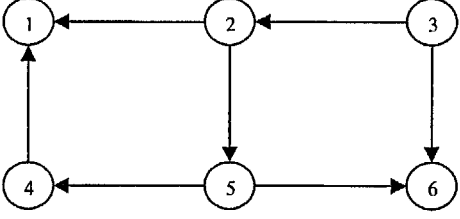
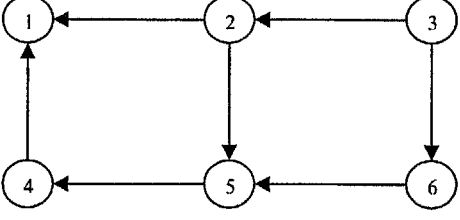
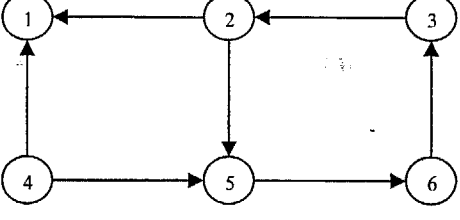
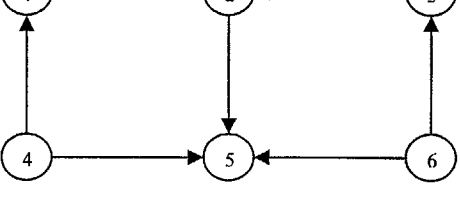
1010010		No	N.A.
1010011		No	N.A.
1010100		Yes	728.44 sec
1010101		No	N.A.
1010110		No	N.A.
1010111		No	N.A.

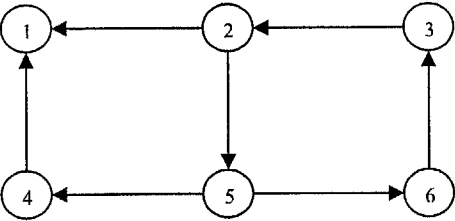
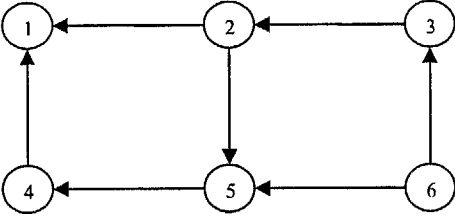
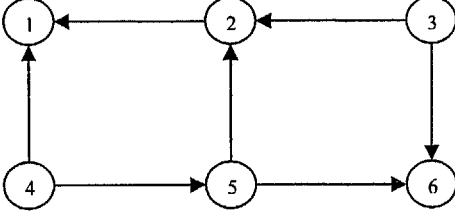
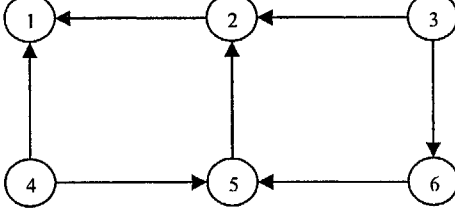
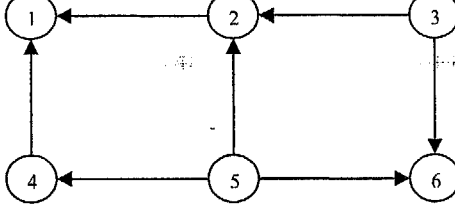
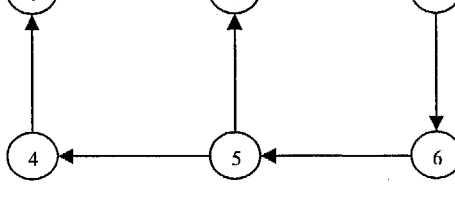
1011000		No	N.A.
1011001		No	N.A.
1011010		No	N.A.
1011011		No	N.A.
1011100		Yes	728.44 sec
1011101		No	N.A.

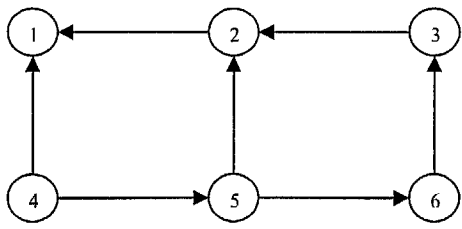
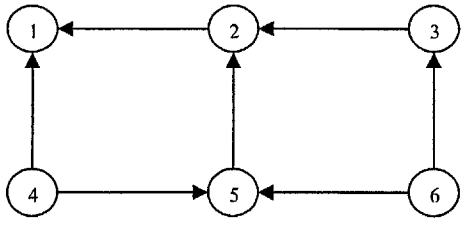
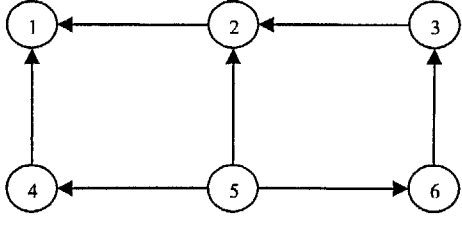
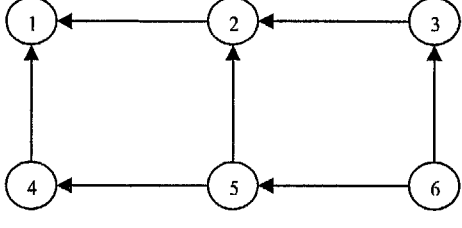
1011110		No	N.A.
1011111		No	N.A.
1100000		No	N.A.
1100001		No	N.A.
1100010		No	N.A.
1100011		No	N.A.

1100100		No	N.A.
1100101		No	N.A.
1100110		No	N.A.
1100111		No	N.A.
1101000		No	N.A.
1101001		No	N.A.

1101010		No	N.A.
1101011		No	N.A.
1101100		No	N.A.
1101101		No	N.A.
1101110		No	N.A.
1101111		No	N.A.

1110000		No	N.A.
1110001		No	N.A.
1110010		No	N.A.
1110011		No	N.A.
1110100		No	N.A.
1110101		No	N.A.

1110110		No	N.A.
1110111		No	N.A.
1111000		No	N.A.
1111001		No	N.A.
1111010		No	N.A.
1111011		No	N.A.

1111100		No	N.A.
1111101		No	N.A.
1111110		No	N.A.
1111111		No	N.A.