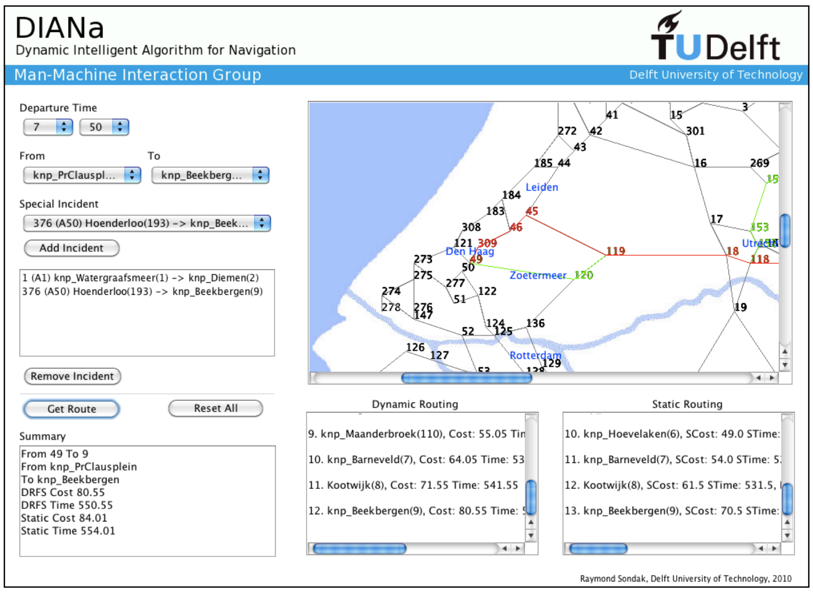
# Dynamic Intelligent Algorithm for Navigation (DIANa)

A Route Finding System using Historical Traffic Data


DIANa
Dynamic Intelligent Algorithm for Navigation
TU Delft
Man-Machine Interaction Group
Delft University of Technology
Add Incident
Remove Incident
Get Route
Reset All
Leiden
Den Haag
Zoetermeer
Rotterdam
Utrecht

Departure Time: 7 50

From: knp_PrClauspl... To: knp_Beekberg...

Special Incident: 376 (A50) Hoenderloo(193) -> knp_Beek...

1 (A1) knp_Watergraafsmeer(1) -> knp_Diemen(2)
376 (A50) Hoenderloo(193) -> knp_Beekbergen(9)

Summary

```
From 49 To 9
From knp_PrClausplein
To knp_Beekbergen
DRFS Cost 80.55
DRFS Time 550.55
Static Cost 84.01
Static Time 554.01
```

Dynamic Routing

```
9. knp_Maanderbroek(110), Cost: 55.05 Tin
10. knp_Barneveld(7), Cost: 64.05 Time: 53
11. Kootwijk(8), Cost: 71.55 Time: 541.55
12. knp_Beekbergen(9), Cost: 80.55 Time: 5
```

Static Routing

```
10. knp_Hoevelaken(6), SCost: 49.0 STime:
11. knp_Barneveld(7), SCost: 54.0 STime: 5
12. Kootwijk(8), SCost: 61.5 STime: 531.5, l
13. knp_Beekbergen(9), SCost: 70.5 STime:
```

Raymond Sondak, Delft University of Technology, 2010

by

Raymond Christopher Sondak

A thesis submitted in partial satisfaction
Of the requirements for the degree of

Master of Science

Presented at

Delft University of Technology,
Faculty of Electrical Engineering,
Mathematics and Computer Science,
Man-Machine Interaction Group.

October 2010

# Dynamic Intelligent Algorithm for Navigation (DIANa)
A Route Finding System using Historical Traffic Data

Raymond Christopher Sondak
Student No. 1059009
October 2010

**Delft University of Technology**
Faculty of Electrical Engineering, Mathematics, and Computer Science
Department of Mediamatics
Man-Machine Interaction Group
Mekelweg 4, 2628 CD, Delft
The Netherlands

## Member of the Supervising Committee

prof. drs. dr. L.J.M. Rothkrantz
dr. ir. P. Wiggers
ir. H. Geers

## Abstract

Traffic congestion has been one of the major problems in our everyday life. It has caused billions and billions of dollars including loss of productivity in our everyday life. Many institutions, governmental organizations, and academics have tried to come up with many initiatives and solutions to minimize congestion.

Our research on dynamic routing is meant to help minimizing this problem from individual perspective since we believe that traffic congestion is also directly related and can be influenced by individual's decision making behavior.

We focused on finding a solution that can be integrated to an existing system by using a routing algorithm that can forecast traffic congestion and finding the fastest route to get to the end point. The forecast will be based on the historical traffic information. Deriving on the notion that most traffic congestions happen almost at the same time during the day and the same hours daily, we have combined the historical real traffic data to be used in an algorithm to help an effective Dynamic Intelligent Algorithm for Navigation (DIANa) to avoid congestion or other road blockage.

# Acknowledgement

The completion of this thesis is one of the most important milestones in my life. It took a lot of strength and persistence and it was one of the biggest struggles for me. I would not be able to finish this thesis without the help and support from many people.

The first person I would like to thank is my supervisor, prof. drs. dr. L.J.M. Rothkrantz, who guided me through this project to the completion of this thesis. He is not only a man with bright ideas but also have a great personality. I really enjoyed my time working with him. He has shown a lot of patience and understanding regarding my full-time job with Capgemini while finishing this thesis. Without his coaching and guidance, I will not be able to finish this thesis.

Secondly I would like to thank Bart Vastenhouw for his support for setting up an iMac that I have used as a server for 3 months continuously to collects traffic data from ANWB. I would also like to express my gratitude to all of my friends in The Netherlands that always supporting me and pushing me to finish this thesis as soon as possible. Also a lot of thanks to everyone at MMI Group TU Delft for the positive attitudes and willingness to offer any kind of help when needed.

Special thanks for the member of supervising committee, dr. ir. P. Wiggers and ir. H. Geers.

I feel a deep sense of gratitude to my whole family, especially my mother and my four sisters for always supporting me in all my journeys. Words are not enough to describe how grateful I am to have you as my family, they are the people that I always looked up to and they are an important part of my life and I hope that I can share many more successful journeys with them in the coming years.

A very special thank to my lovely wife that has been by my side all this time. Her encouragement was the greatest support. There had been many moments when I felt like this thesis would never be done and without her presence I would never be able to get through those moments. She has been the light through my many ups and downs. She has shown the biggest affections and understanding not only as my other half but as well as a friend, a coach, a mentor, and a partner with whom I will share many more moments in my life. Her family has also been a big support to me. I have felt great encouragement from them, directly and indirectly in many ways, for which I also would like to express my gratitude.

At last, I would like to dedicate this thesis to my father who passed away 10 years ago. His memory has been the greatest inspiration to me. Although he is not here, I could feel his presence guiding me through this journey. His faith in me has given me strength and I hope I have become the person he has always hoped for.

# Content

**Acknowledgement** .......... **iv**

**Content** .......... **vi**

**List of Figures** .......... **x**

**List of Tables** .......... **xiv**

**List of Equations** .......... **xvi**

**1 Introduction** .......... **2**
1.1 Dynamic Route Finding Systems .......... 2
1.2 Background Study .......... 5
1.2.1 Source of Congestion .......... 5
1.2.2 How We Deal with Congestion Currently .......... 9
1.2.3 Satellite Navigation Systems in dealing with Congestion .......... 11
1.3 Thesis Problem Definition and Research Objectives .......... 13
1.4 Research Challenges .......... 14
1.5 Thesis Approach .......... 14
1.6 Thesis Overview .......... 16

**2 Dynamic Routing System** .......... **18**
2.1 Introduction .......... 18
2.2 Shortest Path Problem .......... 19
2.3 Concept of DIANa .......... 20
2.4 Static Road Routing .......... 21
2.5 Approach .......... 21
2.6 Development Process .......... 22

**3 Dynamic Routing** .......... **24**
3.1 The Theory .......... 24
3.2 Pseudo Code .......... 26
3.3 Flow Diagram of Dynamic Route Finding System .......... 28
3.4 Route Calculation of Dynamic Route Finding System .......... 29

**4 Data Collection** .......... **30**
4.1 Research Data .......... 30

4.2 Creating the Historical Data .......... 31
4.3 How to Calculate the Delays in the Database .......... 33
4.3.1 Traffic Information with Length and Delay Duration .......... 33
4.3.2 Traffic Information with Length and No Delay Duration .......... 36
4.4 Data Processing .......... 36
4.4.1 Linear Interpolation .......... 36
4.4.2 Averaging .......... 37

**5 Conceptual Design .......... 38**
5.1 Research Goal .......... 38
5.2 System Architecture .......... 39
5.3 Modeling Traffic Data .......... 40
5.3.1 Extract Data .......... 41
5.3.2 Data Transformation .......... 42
5.3.3 Load Data - Using Traffic Database in Dynamic Routing Algorithm . 43
5.4 Special Incident .......... 44

**6 Detailed Design .......... 46**
6.1 Proof of Concept .......... 46
6.2 The Implemented Algorithm .......... 46
6.2.1 Static Routing .......... 47
6.2.2 Dynamic Routing using Historical Data .......... 47
6.3 Software's used .......... 47
6.3.1 NetBeans IDE .......... 47
6.3.2 Matlab .......... 48
6.4 Requirements .......... 48
6.4.1 Functional Requirements .......... 48
6.4.2 Non-Functional Requirements .......... 49
6.4.3 Pseudo Requirements .......... 50
6.4.4 System Requirements .......... 50
6.5 Use Case .......... 51
6.6 Class Diagram .......... 53
6.7 State Chart Diagram .......... 54
6.8 Sequence Diagram .......... 55
6.9 Class Descriptions .......... 56
6.9.1 Class: Data Reader .......... 56
6.9.2 Class: RoutingGUI .......... 56
6.9.3 Class: RoutingEngine .......... 57
6.9.4 Class: NodeResult .......... 58
6.9.5 Class: Time .......... 58
6.9.6 Class: Node .......... 59
6.10 Detail Methods Description .......... 60
6.10.1 findShortest() Method of RoutingEngine.class .......... 60
6.10.2 getMin() Method of RoutingEngine.class .......... 61
6.10.3 relaxDynamic () Method of RoutingEngine.class .......... 61
6.11 GUI Design .......... 66

**7 Experiment and Results .......... 68**

7.1 Experiment ........ 68
Release 1: Routing using static data and Dijkstra's algorithm ........ 68
Release 2: Routing using dynamic data and dynamic algorithm ........ 68
7.2 Hypothesis Formulation ........ 68
7.3 Computational Consideration ........ 69
7.4 Sample Selection ........ 69
7.4.1 City Selection ........ 70
7.4.2 Departure Time Selection ........ 70
7.5 Methodology ........ 70
7.6 Results ........ 73
7.7 Conclusions ........ 74

**8 Data Validation ........ 76**
8.1 Data Reliability and Consistency ........ 76
8.1.1 Sample Selection ........ 76
8.1.2 Methodology ........ 76
8.1.3 Results ........ 79
8.2 Data Stability and Robustness ........ 83
8.2.1 Sample Selection ........ 84
8.2.2 Methodology ........ 84
8.3 Results ........ 87
8.4 Conclusion ........ 88

**9 Conclusion and Recommendation ........ 90**
9.1 Extended Summary ........ 90
9.1.1 Study Literature, Research Objectives, and Problem Definition ........ 90
9.1.2 Dynamic Routing Algorithm ........ 91
9.1.3 Data Collection ........ 91
9.1.4 Define Historical Traffic Database, Design a Dynamic Routing System, and Implementation ........ 91
9.1.5 Testing and Analysis ........ 92
9.2 General Conclusions ........ 92
9.3 Research Limitations and Challenges ........ 93

**Bibliography ........ 96**

**A Dijkstra's Algorithm ........ 100**
A.1 Introduction ........ 100
A.2 The Algorithm ........ 100
A.2.1 Graph Definitions ........ 100
A.2.2 How Dijkstra's Algorithm Works ........ 101
A.2.3 Pseudo Code ........ 102
A.2.4 Flow diagram of Dijkstra's algorithm ........ 104
A.2.5 Detail Example: Step by step operation of Dijkstra's algorithm ........ 105

**B Experiment Results ........ 112**

**C Data Reliability and Consistency Results with Dataset #1 ........ 118**

**D Data Reliability and Consistency Results with Dataset #2 .............................. 124**

**E Data Reliability and Consistency Results with Dataset #3 .............................. 130**

**F Data Reliability and Consistency Results with Dataset #4 .............................. 136**

# List of Figures

Figure 1: Traffic congestion illustration .......... 2
Figure 2: Three dimensional graph .......... 4
Figure 3: Traffic congestion in peak hours .......... 5
Figure 4: Traffic congestion anatomy [3] .......... 7
Figure 5: Traffic congestion anatomy [3] .......... 8
Figure 6: How to deal with traffic congestion .......... 11
Figure 7: Satellite Navigation System (TomTom) .......... 12
Figure 8: Individual decision on traffic congestions .......... 13
Figure 9: Routing System Hierarchy .......... 15
Figure 10: Dynamic Routing that can provide optimal solution .......... 19
Figure 11: Traffic jam illustration .......... 21
Figure 12: Traffic data with time dimension .......... 25
Figure 13: Function Dijkstra(G, s, t) pseudo code .......... 26
Figure 14: Function relaxNeighbors(u) pseudo-code .......... 27
Figure 15: Function extractMinimum(Q) pseudo-code .......... 27
Figure 16: Function timeData(u, v, d(u)) pseudo-code .......... 27
Figure 17: Flow Diagram Dynamic Route Finding System .......... 28
Figure 18: Route calculation in time dimension .......... 29
Figure 19: Traffic congestion from VID .......... 31
Figure 20: ANWB Data Polling .......... 31
Figure 21: ANWB traffic information .......... 32
Figure 22: Traffic information from junction to junction .......... 33
Figure 23: Traffic information between two junctions .......... 33
Figure 24: Traffic information with several exits between two junctions .......... 34
Figure 25: Traffic information between several junctions .......... 34
Figure 26: Overlapping traffic information .......... 35
Figure 27: Linear interpolation over NaN values .......... 36
Figure 28: Plot of traffic data from node Watergraafsmeer to node Diemen .......... 37
Figure 29: DIANa Infrastructure .......... 38
Figure 30: DIANa System architecture .......... 39
Figure 31: Shortest path tree .......... 40
Figure 32: Modeling traffic data .......... 40
Figure 33: Directed graph .......... 41
Figure 34: DIANa Flow Diagram .......... 44
Figure 35: Use case diagram .......... 51

Figure 36: Class diagram ........ 53
Figure 37: DIANa State chart diagram ........ 54
Figure 38: Sequence diagram ........ 55
Figure 39: Method findShortest() pseudo-code ........ 60
Figure 40: Method getMin() pseudo-code ........ 61
Figure 41: Method relaxDynamic() pseudo-code ........ 62
Figure 42: Method relaxStatic() pseudo-code ........ 64
Figure 43: Method inSet() pseudo-code ........ 65
Figure 44: Method inUnset() pseudo-code ........ 65
Figure 45: DIANa Graphical User Interface ........ 66
Figure 46: Traffic congestion statistic ........ 70
Figure 47: Color matrix of the difference between DIANa and static road routing using historical traffic data – More is better. ........ 72
Figure 48: Performance comparison(Appendix B) ........ 73
Figure 49: Color matrix of the difference between historical traffic data and individual traffic dataset – Less is better. ........ 78
Figure 50: Result of first dataset(Appendix C) ........ 79
Figure 51: Result of second dataset(Appendix D) ........ 80
Figure 52: Result of third dataset(Appendix E) ........ 81
Figure 53: Result of fourth dataset(Appendix F) ........ 82
Figure 54: Uniform Distribution in the interval (-MIN(h(r)), MIN(h(r))) ........ 85
Figure 55: Road r=1 from 12:00 - 23:50 with error percentage 0% - 100% ........ 86
Figure 56: Directed graph ........ 101
Figure 57: function Dijkstra(G, s) ........ 103
Figure 58: function relax-neighbors(u) ........ 103
Figure 59: function extract-minimum(Q) ........ 103
Figure 60: Flow diagram Dijkstra algorithm ........ 104
Figure 61: Directed graph ........ 105
Figure 62: Directed graph after step 1 ........ 106
Figure 63: Directed graph after step 2 ........ 107
Figure 64: Directed graph after step 3 ........ 108
Figure 65: Directed graph after step 4 ........ 109
Figure 66: Directed graph after step 5 ........ 110
Figure 67: Set Q and set S ........ 110
Figure 68: Experiment result with $t_0$=15:20 and $t_1$=15:40 ........ 113
Figure 69: Experiment result with $t_2$=16:00 and $t_3$=16:20 ........ 114
Figure 70: Experiment result with $t_4$=16:40 and $t_5$=17:00 ........ 115
Figure 71: Experiment result with $t_6$=17:20 and $t_7$=17:40 ........ 116
Figure 72: Experiment result with $t_8$=18:00 and $t_9$=18:20 ........ 117
Figure 73: R(t,1) with $t_0$=15:20 and $t_1$=15:40 ........ 119
Figure 74: R(t,1) with $t_2$=16:00 and $t_3$=16:20 ........ 120
Figure 75: R(t,1) with $t_4$=16:40 and $t_5$=17:00 ........ 121
Figure 76: R(t,1) with $t_6$=17:20 and $t_7$=17:40 ........ 122
Figure 77: R(t,1) with $t_8$=18:00 and $t_9$=18:20 ........ 123
Figure 78: R(t,2) with $t_0$=15:20 and $t_1$=15:40 ........ 125
Figure 79: R(t,2) with $t_2$=16:00 and $t_3$=16:20 ........ 126
Figure 80: R(t,2) with $t_4$=16:40 and $t_5$=17:00 ........ 127
Figure 81: R(t,2) with $t_6$=17:20 and $t_7$=17:40 ........ 128

**Figure 82: R(t,2) with $t_8$=18:00 and $t_9$=18:20 ........ 129**
**Figure 83: R(t,3) with $t_0$=15:20 and $t_1$=15:40 ........ 131**
**Figure 84: R(t,3) with $t_2$=16:00 and $t_3$=16:20 ........ 132**
**Figure 85: R(t,3) with $t_4$=16:40 and $t_5$=17:00 ........ 133**
**Figure 86: R(t,3) with $t_6$=17:20 and $t_7$=17:40 ........ 134**
**Figure 87: R(t,3) with $t_8$=18:00 and $t_9$=18:20 ........ 135**
**Figure 88: R(t,4) with $t_0$=15:20 and $t_1$=15:40 ........ 137**
**Figure 89: R(t,4) with $t_2$=16:00 and $t_3$=16:20 ........ 138**
**Figure 90: R(t,4) with $t_4$=16:40 and $t_5$=17:00 ........ 139**
**Figure 91: R(t,4) with $t_6$=17:20 and $t_7$=17:40 ........ 140**
**Figure 92: R(t,4) with $t_8$=18:00 and $t_9$=18:20 ........ 141**

# List of Tables

Table 1: Artificial Intelligence Matrix ........ 18
Table 2: DIANa algorithm variables description ........ 26
Table 3: Traffic database ........ 41
Table 4: Traffic database fields ........ 42
Table 5: Historical traffic data ........ 42
Table 6: Network data ........ 42
Table 7: Intersection data ........ 43
Table 8: Network weight data ........ 43
Table 9: Historical traffic data ........ 43
Table 10: Use case CalculateRoute ........ 52
Table 11: Use case EnterJourneyData ........ 52
Table 12: Lookup on traffic data ........ 63
Table 13: City selection based on total number of population ........ 69
Table 14: Top 5 - Most congested road in evening rush hour ........ 84
Table 15: Route chosen for analysis ........ 84
Table 16: Experiment with error percentage between 10% to 100% ........ 87
Table 17: Experiment with error percentage between 10% to 20% ........ 87

# List of Equations

Equation 1: Shortest path function ..... 20
Equation 2: Travel time between nodes ..... 24
Equation 3: Total travel time ..... 25
Equation 4: Calculating delay between knp. Oudenrijn and knp. Lunetten ..... 35
Equation 5: Calculating delay between knp. Lunetten and Driebergen ..... 35
Equation 6: Calculating delay between Nootdorp and Zevenhuizen ..... 35
Equation 7: Historical traffic data calculation ..... 37
Equation 8: Conversion to minutes ..... 63
Equation 9: Conversion to numeric time dimension of DIANa ..... 63
Equation 10: Rounding to integer ..... 63
Equation 11: Routing using static data ..... 68
Equation 12: Routing using historical data ..... 68
Equation 13: Dynamic traveling time matrix of 21 cities ..... 70
Equation 14: Static traveling time between 21 cities using traffic data ..... 71
Equation 15: DIANa performance matrix between 21 cities ..... 71
Equation 16: Performance gain at time t ..... 71
Equation 17: Time sample ..... 71
Equation 18: Elements of time sample ..... 71
Equation 19: DIANa traveling time matrix between 21 cities ..... 76
Equation 20: DIANa traveling time matrix between 21 cities using traffic data ..... 77
Equation 21: Sample time range ..... 77
Equation 22: Traffic data ..... 77
Equation 23: Performance difference between DIANa and each traffic data ..... 77
Equation 24: Time sample ..... 77
Equation 25: Elements of time sample ..... 77
Equation 26: Random noise generator ..... 85
Equation 27: Noisy road calculation ..... 85
Equation 28: Noisy road calculation ..... 85
Equation 29: Number of roads ..... 86
Equation 30: Error percentage ..... 86
Equation 31: Noisy road 1 calculation ..... 86
Equation 32: Cost function for weighted graph ..... 101
Equation 33: Unsettled set Q ..... 105
Equation 34: Minimum cost to reach vertex a ..... 105
Equation 35: Previous vertex of a ..... 105
Equation 36: Unsettled and settled vertices ..... 105

Equation 37: Function extractMinimum(Q) ........ 106
Equation 38: Remove vertex a from Q ........ 106
Equation 39: Add vertex a to S ........ 106
Equation 40: Calculate minimum costs to vertex b and vertex c from vertex a .. 106
Equation 41: Previous vertex of vertex b ........ 106
Equation 42: Add vertex b and vertex c to Q ........ 106
Equation 43: Set Q and set S ........ 107
Equation 44: Extract vertex with minimum cost from Q ........ 107
Equation 45: Remove vertex c from Q ........ 107
Equation 46: Add vertex c to S ........ 107
Equation 47: Minimum cost to vertex b ........ 107
Equation 48: Previous vertex of vertex b ........ 108
Equation 49: Minimum cost and previous vertex of vertex d ........ 108
Equation 50: Add b and d to set Q ........ 108
Equation 51: Set Q and set S ........ 108
Equation 52: Remove vertex b from Q ........ 108
Equation 53: Add vertex b to S ........ 108
Equation 54: Calculate minimum cost to vertex d ........ 109
Equation 55: Previous vertex of vertex d ........ 109
Equation 56: Add d to set Q ........ 109
Equation 57: Set Q and set S ........ 109

# Chapter 1

# Introduction

*Preview*

This chapter introduces the reader to the problem statements, research objectives, and background study of dynamic routing systems, which is the main topic of this thesis.

## 1.1 Dynamic Route Finding Systems

Commuting from one place to another has become one of the most important parts of our life. We commute to go to school, to work, to visit friends and relatives, and even for vacation. Although there are many ways for commuting, due to mostly convenience reason, most of us just get in a car and drive. This has resulted in a crowded road over the years and forms what we all know as congestion or traffic jam.

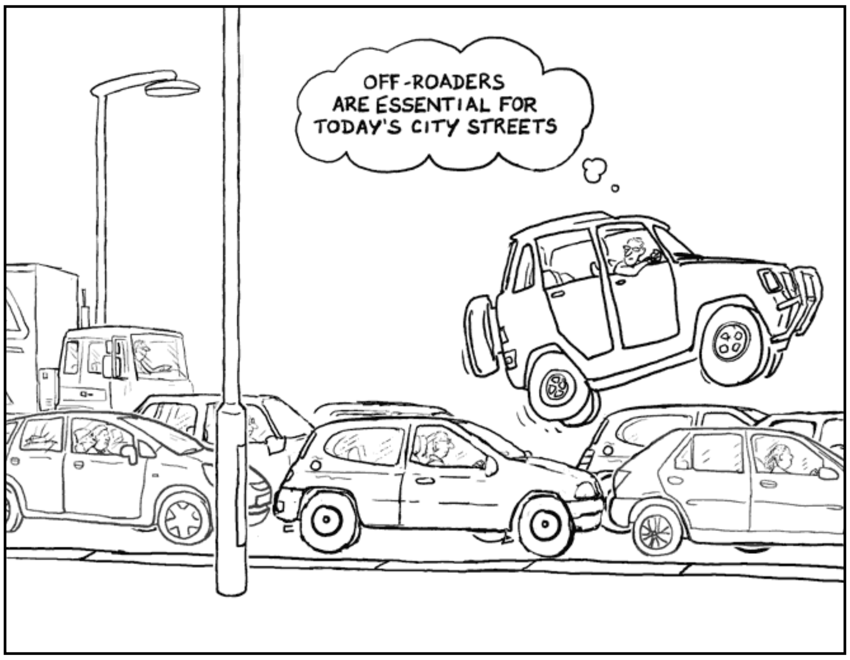


Figure 1: Traffic congestion illustration

Traffic jams cost billions in lost productivity, wasted fuel, and no matter how long the traffic jam is, there is nothing more frustrating than getting stuck in a traffic jam that stretches into the distance [1], [2].

Why did we always have to take the same road even though we know that the road we are taking is always congested on daily basis? Why doesn't our satellite navigation system direct us to a different route? Why are we still stuck in a traffic jam even though we have first listened to the radio or read traffic news on the internet before we stepped into our car?

We usually take the same route from home to work for example and vice versa because that is the fastest route we know --the fastest, if and only if there is no congestion--, or that is the fastest route our automotive navigation system (i.e. TomTom) gives us. An automotive navigation system should be able to provide us with an alternative route. The problem is that we can be stubborn at a time and take another road or exit than advised instead, irrationally, with a hope that it would lead to a faster trip, relying on the ability of the navigation system on recalculating a new route. However, often it could be even worse and our chosen route will probably lead us to a heavier congested road. We can also listen to the radio on the traffic news on the way and try to avoid congested road by taking another route, but it could be unreliable because the real-time information can no longer be valid at the time we get to that point.

Many digital maps use only signposted speeds to calculate journey time and do not take into account the way historical traffic flows during the week at different times of the day. As traffic jam or congestion tends to happen with high probability on the same locations and at the same time horizon, we might be able to use the historical traffic data as one way to predict traffic behavior and use our automotive navigation system to direct us accordingly. This is called dynamic routing.

If we want to travel from one point to another, we usually take the shortest route. The shortest route in a graph with given distances between nodes can be computed using Dijkstra's Algorithm. However, in case of congestion we would look for the shortest route in time, not in distance. The traveling time along the edges in the graph is a dynamic parameter that changes in time.

Computing the shortest route in time in a graph with dynamic changing traveling time along edges is called dynamic routing.

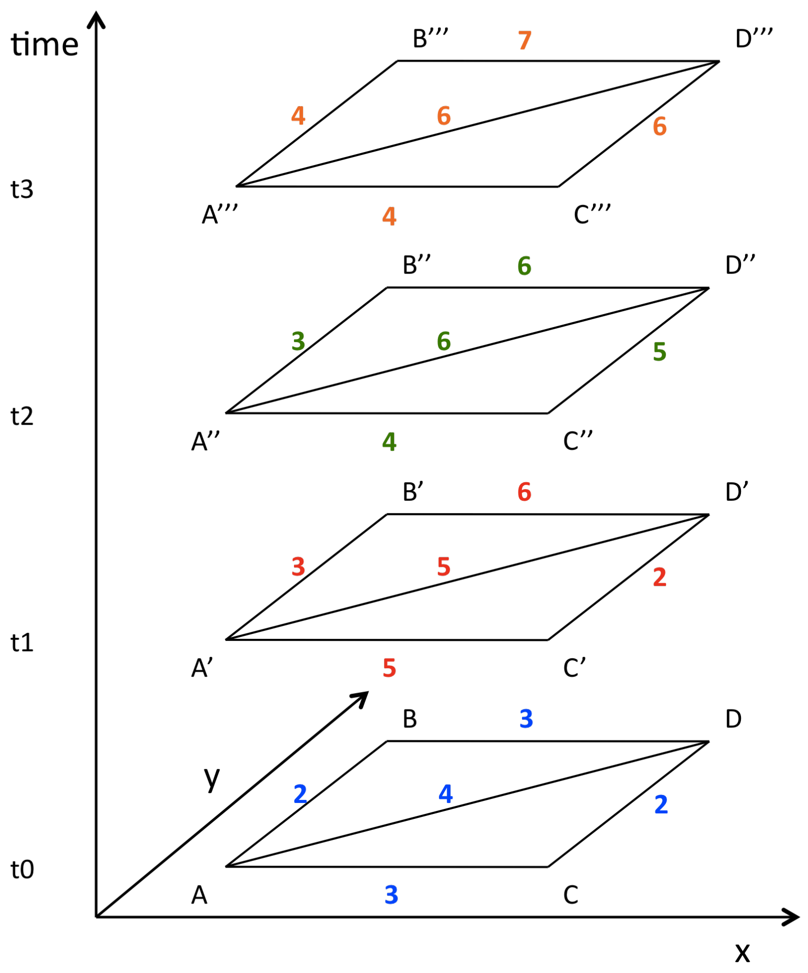


Figure 2: Three dimensional graph

Traveling time along the edges in a graph can be predicted on the basis of historical data, measured or computed in real time by tracking cars or predicted using advanced statistical.

The question is, whether dynamic routing systems are really able to reduce traveling time in case of no congestion or in the case of congestion due to extraordinary events. Consider the case of an accident on a road; it would really help to take an alternative route. Hence, our research question, how helpful is dynamic routings in general.

As automotive navigation devices become more widely used, we can incorporate a dynamic routing technique using historical data that automatically calculate the fastest route depending on the departure time in the automotive navigation devices.

## 1.2 Background Study

There have been many studies focusing on congestion and how to deal with this problem. As mentioned earlier, congestion has taken a high toll in our society, physically, mentally, and even economically. Why and how do congestions happen? The background study will focus on this question, the impact of congestion, and how we currently deal with this problem.

### 1.2.1 Source of Congestion

Traffic congestion has an increasingly negative impact on our quality of life. Across different places, people, business, and industry, the economy, and the environment pay a higher and higher price for increased congestions through delays, lost opportunities, increased accident, reduced competitiveness, pollution, frustration, and much more. It affects us not only physically but also mentally.

Figure 3: Traffic congestion in peak hours

Traffic congestion can be classified into three main categories. **One** is due to some incidents that can aggravate congestion like accidents, construction work, severe weather condition, etc. **Second** is simply due to the fact that there is more demand for the road space than the road capacity itself because of the low occupancy of the vehicle (Example: one person one vehicle going from one point to another, instead of sharing vehicle). **Third** is due to the infrastructure of physical highway features such as the number of lanes, width of each lanes, etc.

Previous work has shown that traffic congestion is a result of seven root causes [3] deriving from the main three categories mentioned above:

*Category 1 – Traffic – Influencing Events*

**1. Traffic Incidents**

Traffic incidents are events that disrupt the normal flow of traffic, usually by physical impedance in the travel lanes. Events such as vehicular crashes, breakdowns, and debris in travel lanes are the most common form of incidents.

**2. Work Zones**

Work zones are construction activities on the roadway that result in physical changes to the highway environment. These changes may include a reduction in the number or width of travel lanes, lane "shifts," lane diversions, reduction, or elimination of shoulders, and even temporary roadway closures.

**3. Weather**

Environmental conditions can lead to changes in driver behavior that affect traffic flow or traffic modalities. For example, we usually notice that there are more cars and more traffic jams when it rains.

*Category 2 – Traffic Demand*

**4. Fluctuations in Normal Traffic**

Day-to-day variability in demand leads to some days with higher traffic volumes than others. Varying demand volumes superimposed on a system with fixed capacity also results in variable (i.e., unreliable) travel times.

**5. Special Events**

Special events are a special case of demand fluctuations whereby traffic flow in the vicinity of the event will be radically different from "typical" patterns. Special events occasionally cause "surges" in traffic demand that overwhelm the system.

*Category 3 – Physical Highway Features*

**6. Traffic Control Devices**

Intermittent disruption of traffic flow by control devices such as railroad grade crossings and poorly timed signals also contribute to congestion and travel time variability.

**7. Physical Bottlenecks ("Capacity")**

Capacity is the maximum amount of traffic capable of being handled by a given highway section. Capacity is determined by a number of factors: the number and width of lanes and shoulders; merge areas at interchanges; and roadway alignment (grades and curves).

Congestion can result from one or interaction from several abovementioned causes.

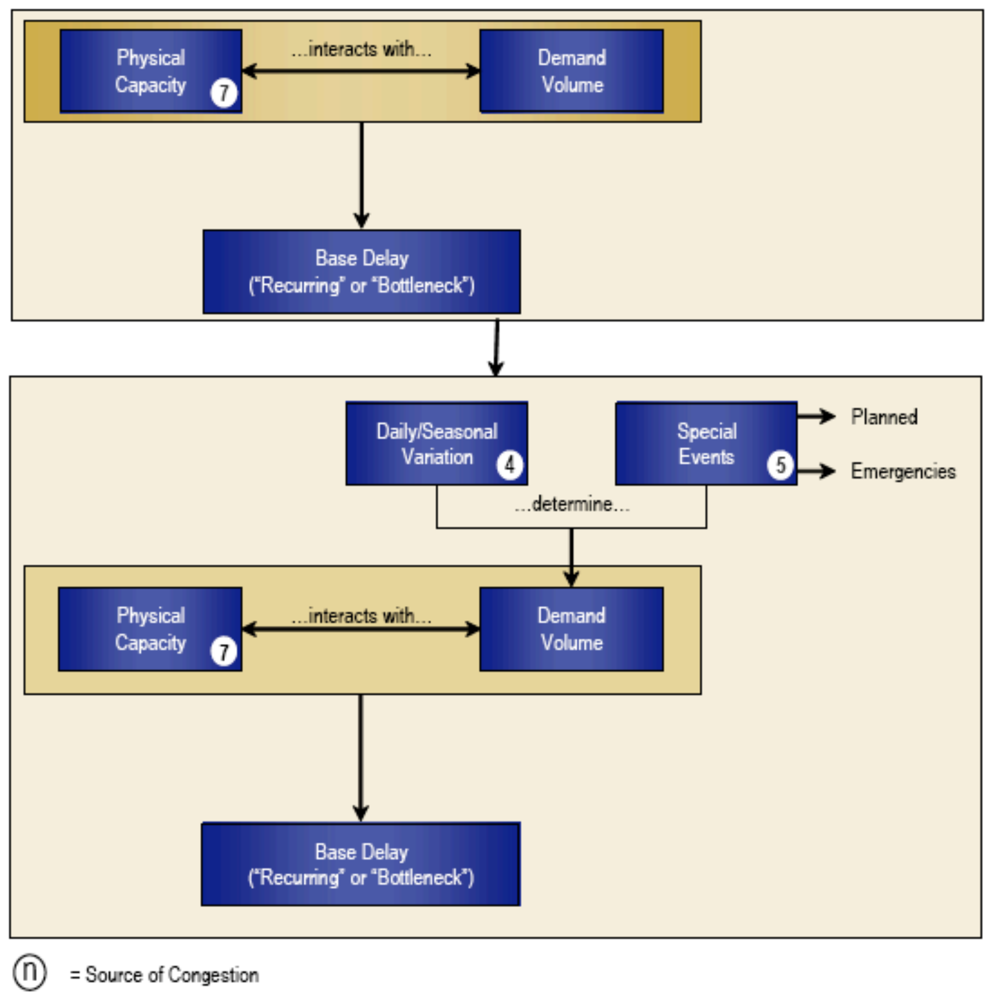


**Figure 4: Traffic congestion anatomy [3]**

In any given circumstances, the starting point for congestion on most days is the amount of traffic or vehicles on the road and the physical restrictions on the highway (bottlenecks). Traffic varies from day-to-day throughout the year and special events may cause surges in traffic at unexpected times. See traffic jam anatomy in figure 4 as an example of how much traffic varies even over as short a period as a month.

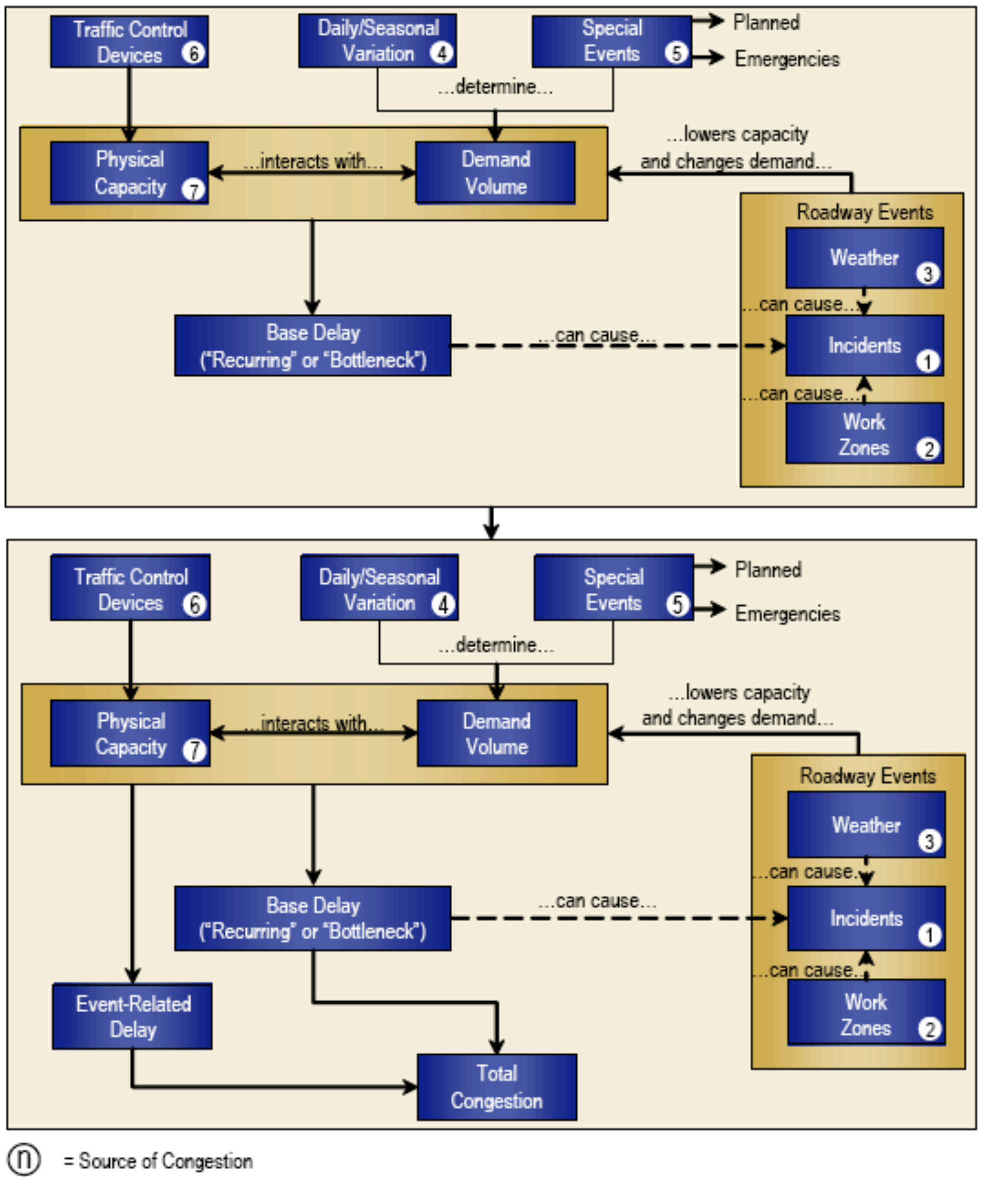


**Figure 5: Traffic congestion anatomy [3]**

Just as traffic varies across time periods, so does physical capacity. The operation of traffic signals changes capacity, often minute-to-minute. When roadway events occur, they also cause the physical capacity of the roadway to be lowered. In many cases, traffic incidents and work zones can reduce lanes, and bad weather causes drivers to space themselves out more. Base-level congestion caused by bottlenecks can lead to increased traffic incidents due to tighter vehicle spacing and vehicles overheating in summer. Finally, the existence of extreme congestion can cause some drivers to change their routes or to forego trips altogether.

Despite of knowing the sources and what causes congestion, it is still subject to continuous research *how to reduce and minimize congestion*. In the past years, congestion has become one of the major issues in the urban areas in most countries. Concentration of population in cities appears as one of the most important features in current modern civilization. As some people said, our way of life has become a major liabilities and it is a price that people pay for various benefits derived from agglomeration of population and economic activity [4].

In many countries, big cities and metropolitan areas are the central of business and economic activities. These areas by any means would draw people in satisfying both their social and economic needs. Culturally and economically vibrant cities are well known to have the worst congestion problems, while declining and depressed cities do not have much traffic. While people complaining about this problem, traffic congestion is actually a sign of a growing and healthy city.

In the Netherlands, during the rush hours, traffic congestion can add up to about three to four hundred kilometers across the different cities, usually the major cities area, or mostly known as the Randstad areas (like Rotterdam, Amsterdam, Utrecht, and Den Haag) are the most severe ones. In a normal situation, people usually can predict where the severe congestion would happen, in which highways, which junctions, and around which time. It is widely known that in the Netherlands, the Randstad areas are the most pact highways – due to the reasons mentioned above.

### 1.2.2 How We Deal with Congestion Currently

A few perspectives can be taken when analyzing how we currently deal with congestions. For this thesis, we define two different approaches on dealing with congestion. First is what we call *external factors*, which means those that are related to forces outside the influence of individual's motive and behavior. Second is what we call *internal factors*, which are related to the individual's motive and behavior, which at many times are irrational.

Example of external factors in dealing with congestion problem currently is government regulations. In many countries, the government uses some best practices to alleviate congestion and some of them are incorporated into law and legislation, for example speed limit and taxing of fuel and road capacity use.

Transportation engineers and planners have developed a variety of strategies to deal with congestion, which fall into three major categories:

- Adding more capacities by increasing the number and size of highways and providing more transit and freight rail service.
- Operating existing capacity more efficiently.
- Encouraging travel and land use patterns that use the system in less congestion producing ways, for example Travel Demand Management, non-automotives travel modes, and land use management.

External forces definitely lay a path to a broader approach in reducing congestion in a longer term. However, another approach that is often overlooked when dealing with congestion problem is the one related to the human behavior itself.

The second factor is what we call internal factors, which are directly related and can be influenced by individual's decision making and behavior, which is more pertinent to this thesis and our dynamic routing solution.

Transportation researchers have long struggled to find satisfactory ways of describing and analyzing congestion, as evident from the large number of often competing approaches and models that have been developed. Early researchers hoped to develop models based on fluid dynamics that would not only be accurate, but also universally applicable. But congestion, unlike fluid flow, *is not a purely physical phenomenon but rather the result of peoples' trip-making decisions and minute-by-minute driving behavior*. One should therefore expect the quantitative, if not also the qualitative, characteristics of congestion to vary with automobile and road design, rules of the road, pace of life, and other factors[4].

Therefore, attempts to reducing or minimizing congestion are being sought from many different perspectives. For example, for many years, economists have been advocating pricing and valuation as the way to deal with urban traffic congestion.

Personal preferences in driving behavior leads to multi faceted or multi criteria problems as seen below:

- Some people prefer shortest route in time or distance
- Some people prefer highways even if it is a longer route
- Some people prefer to avoid traffic lights
- Some people prefer well known routes
- Etc.

Thus, the dynamic routing challenge in a multiple criteria problem is that some people may prefer shortest route but others may not, which could be attributed foremost on the irrational human behavior in making decision.

Despite all these attempts, as mentioned previously, *congestion is not only a manifestation of physical phenomenon but as well as a result of individual trip decision making and driving behavior*, which is not always rational and can be related either to sentiment and emotional decisions or to the lack of knowledge and therefore the lack of capability to make rational decisions.

Assuming that an individual is able to act rationally when he/she has the knowledge to do so, it is *utterly important to equip individual with the right tool in order to help them making the right decision that will result in shortening their travel time and at the same time reducing congestion*, for example a navigation system that is equipped with capability to recalculate and reroute journey to minimize travel delay.

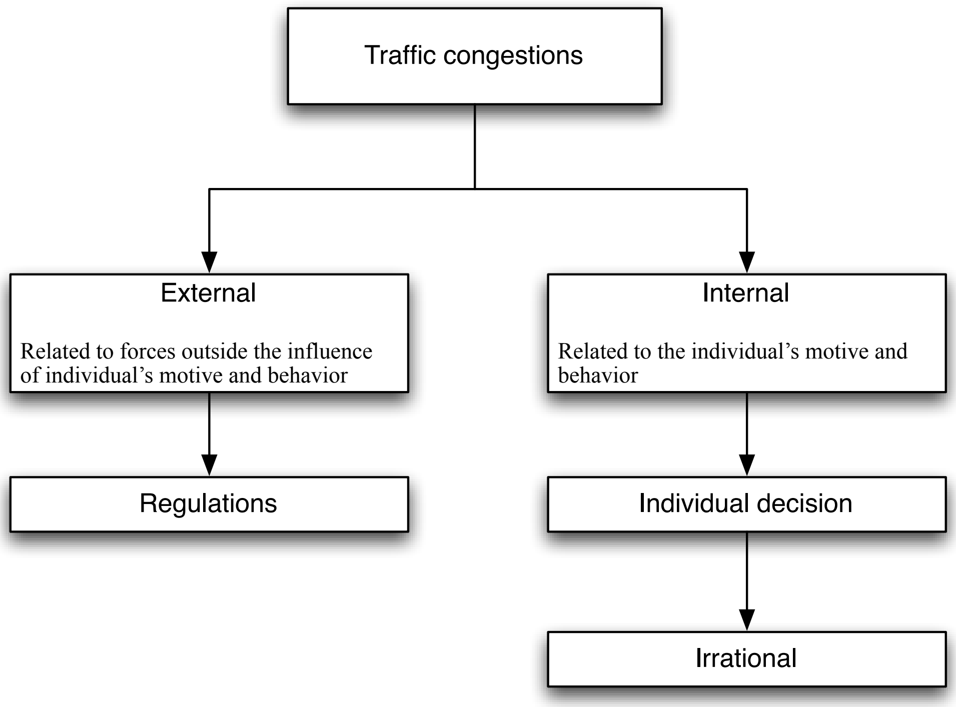

Traffic congestions
External
Related to forces outside the influence of individual's motive and behavior
Internal
Related to the individual's motive and behavior
Regulations
Individual decision
Irrational

Figure 6: How to deal with traffic congestion

### 1.2.3 Satellite Navigation Systems in dealing with Congestion

As mentioned before, it is important to equip individuals with the right tool in order to help them making the right decision that will result in shortening their travel time and at the same time reducing congestion. Such equipment currently exists in a form of satellite navigation. In today's world, GPS is widely used as a travel manager for individuals, to determine routing, and lately even enhanced with the capability to minimize delay and therefore should help minimizing traffic congestion. However, current navigation systems are operated based on a static routing method, meaning it is only based on the shortest route possible and not taking into account historical pattern on traffic phenomenon that might be used to generate an alternative route at certain time of day.

There is no doubt that satellite navigation systems can save time and money. In the recent years, the situation of positioning and navigation has changed dramatically. The dramatic drop in equipment prices over the past ten years, from approximately 20,000 Euro for a civilian receiver to about 80 Euro for the least expensive hand-held receivers today, have led to an enormous growth in the number of GPS users. New applications emerged; for example, car navigation systems, fleet management, aircraft approach and landing, bridge deformation monitoring, offshore drilling research, and the navigation of agricultural field machinery.

Figure 7: Satellite Navigation System (TomTom)

NAVSTAR GPS is the most used satellite navigation system today. The name GPS is often used as a collective term for satellite navigation systems in general. A GPS receiver calculates its position by carefully timing the signals sent by the constellation of GPS satellites high above the Earth. Each satellite continually transmits messages containing the time the message was sent, a precise orbit for the satellite sending the message (the ephemeris), and the general system health and rough orbits of all GPS satellites (the almanac). These signals travel at the speed of light through outer space, and slightly slower through the atmosphere. The receiver uses the arrival time of each message to measure the distance to each satellite, from which it determines the position of the receiver. The resulting coordinates are converted to more user-friendly forms such as latitude and longitude, or location on a map, and then displayed to the user [5].

Considering that such technology already existed and widely known and used, we intent to leverage on this technology to incorporate our dynamic routing system to enhance its capability and expand its use as an intelligent system. As said before, GPS is already widely used as a travel manager for individual, to determine routing, and when this system is equipped with an intelligent system that is able to provide a rational and optimized decision to individuals, it could become a very handy solution to minimize congestion and assuming that an individual is capable to act rationally this should be pertinent to compensate the lack of knowledge when making traffic decision based on the available information.

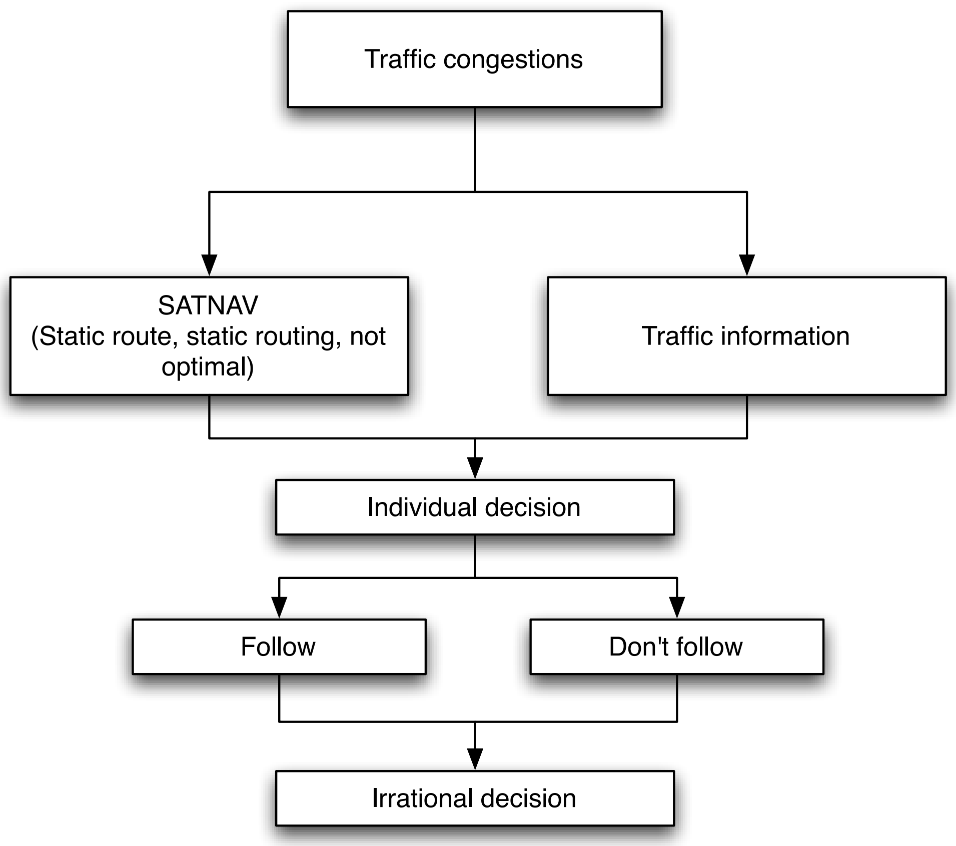


**Figure 8: Individual decision on traffic congestions**

## 1.3 Thesis Problem Definition and Research Objectives

One way of trying to tackle the issue with congestion is using a routing algorithm that can forecast traffic congestion and finding the fastest route to get to the end point. The forecast will be based on the historical traffic information. This will be the main point of the research for this thesis, how to build a system that can manage an effective and efficient route planning using historical traffic information.

The problem in computing a shortest route in time is that we need the speed or traveling time along roads from node to node. The current speed needs to be measured by devices or sensors along those roads. It means that to make prediction assessment of traveling time in future we need prediction models and one way to do that is based on historical data.

Of course there are already service providers that can inform you where the traffic is during the day, how many kilometers, and how long the delay will be, such as ANWB. This information is available through Internet, SMS, radio, etc. Sometimes, if the traffic is so severe, for example due to accident, and the road is completely blocked, they also inform road users of the alternative route.  Nowadays, most navigation system has also had already a device built in to track traffic and provide alternative route at real time. However, this is not always efficient and effective because most of the time real time means that you will probably have been already on the point of “no re-routing’ when the device signaling the traffic and hence you can no longer use the alternative route provided.

Therefore, due to the fact that most traffic congestion happens almost at the same time during the day and the same hours daily, we would like to explore the possibility of combining historical real traffic data to be used in an algorithm to help an effective Dynamic Intelligent Algorithm for Navigation (DIANa) to avoid congestion or other road blockage.

## 1.4 Research Challenges

This thesis however is not without issues and difficulties. There are some challenges we encounter in this research:

- How to model traffic speed along highways?
- How to adapt static route traveling system to be expected dynamic changes in traffic speed based on historical data?
- How to compute static and dynamic routing?

## 1.5 Thesis Approach

As any other research, this thesis follow the usual approach of scientific research and was built upon previous studies and theories on the precedence works related to Dijkstra and dynamic routing, as well as any traffic information. Thereafter, further steps include looking at closely on the algorithm, collecting data, designing the database and the dynamic routing system up to the implementation, testing, and analysis, as can be seen below:

**1. Study Literature**

This is the knowledge collection stage. During the literature study, many research articles and previous works are read in order to gain knowledge pertinent to this thesis. Literature study forms one of the most important pillar on building solid assumptions and hypothesis on the traffic congestion phenomena and how to deal currently with this problem. Further on, we also explore which algorithm could be used in the dynamic routing system, which at the end lead us to the Dijkstra’s algorithm, how it works and what kind of works related to Dijkstra have been done previously. One key input necessary to ensure that our dynamic routing system works properly is the data input. Therefore, we also try to find out from current literatures how to deal and collect the traffic data, which source to use, and any kind work relating to dynamic routing.

**2. Dynamic Routing Algorithm based on Dijkstra's Algorithm**

This stage is looking into the possibility to extend Dijkstra's algorithm to be used as a basis for DIANa and how to develop this algorithm into a dynamic routing algorithm using historical traffic data.

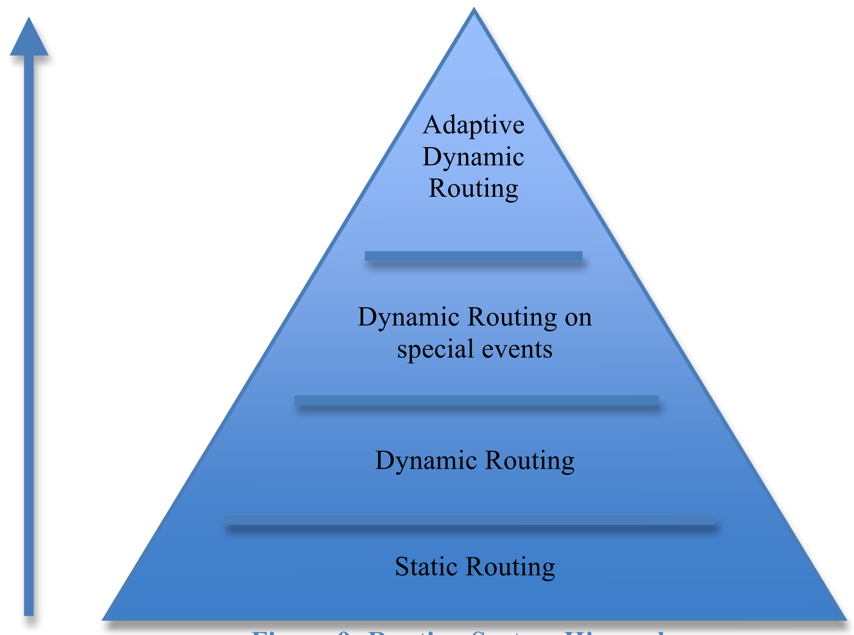


**Figure 9: Routing System Hierarchy**

Dynamic Routing may have several case-based reasons for different scenarios as seen in figure 9, namely:

1. Static routing algorithm will calculate shortest route based on a graph described by fixed paths.
2. Dynamic routing by using historical data where duration along edges is a function of time.
3. Dynamic routing by using historical data and special events or accidents to calculate the shortest route.
4. Adaptive dynamic that calculate shortest route based on multiple variable that keeps changing over time.

**3. Data Collection**

Data collection is crucial in this thesis. As in any other research, collecting the right and relevant data is essential. This step deals with the actual relevant data collection in order to build the historical information, which is an important input in the dynamic routing. This step includes collecting and analyzing data from the source system.

**4. Identify & Define historical traffic database**
Based on the data collected before, we need to identify and design the historical traffic information accordingly in order to be used in DIANa. We need to ensure that the data model is actually valid and ready to be used as the historical traffic database, which will be compatible with the Dijkstra's algorithm to be used in the dynamic routing.

**5. Design a Dynamic Routing System**
This step is the actual design stage where a high level DIANa design based on the Dijkstra's algorithm and the data model as previously mentioned is created.

**6. Implementation**
During the implementation stage, the dynamic routing system is developed according to the design determined earlier.

**7. Testing and Analysis**
Testing and analysis phase involves analyzing the result with possible limitations and elaborate explanations of the variables and checking the result for validity, consistency, and reliability.

**8. Reporting and writing document**
This is the final phase to report and write the document on the research and the conclusion as well as recommendation for future research.

## 1.6 Thesis Overview

This thesis is organized starting with introduction in **chapter 1**. This chapter presents the current development and challenges with traffic congestions, current available approaches to solve congestion and further on framing the problem statement, objectives, setup, as well as challenges to this thesis. **Chapter 2** deals with presenting and introducing the concept of dynamic routing system, the approach and development as well as comparison to the static routing. We will then continue with exploring the algorithm to be used in this dynamic routing in **chapter 3**. More specifically, this chapter explores and provides understanding to Dijkstra's algorithm as well as presenting the reader with the possibility to extend Dijkstra's algorithm into a dynamic routing algorithm using historical traffic data. **Chapter 4** focuses on the data collection method used in constructing the historical database, which is one of the crucial steps in this thesis. **Chapter 5** will then present the conceptual design of DIANa. This chapter focuses further on the system architecture, modeling the traffic data, and designing the database itself including how to use the traffic database on dynamic routing algorithm. **Chapter 6** will further detail this conceptual design as explained in the previous chapter including further explanation in the implemented algorithm, the requirements, and the use case and diagram. The next step in this thesis is conducting an experimental study based on DIANa concept and algorithm mentioned in the previous chapters and the focus of **chapter 7**. This experimental study is necessary to evaluate our system. This chapter will also describe those experiments and their results.

**Chapter 8** will then evaluate the validity and reliability of our data and result of our experimental study. Valid and reliable data is the basic and essential for a successful research, of which without may lead to different or even wrong conclusions. This chapter will show how we measure and test the data validity and reliability used for the database. Finally, the last chapter, **chapter 9** concludes.

# Chapter 2

# Dynamic Routing System

## *Preview*

Probably everybody has heard about Artificial Intelligence (AI), but relatively few people have a really good idea of what the term really means. For many people AI could have been a philosophy or even a technology utopia far beyond this century. What is AI? This chapter will discuss AI in a closer look and what is the possibility of implementing elements of AI using an algorithm for our congestion problem. This chapter will mainly discuss the concept of dynamic road routing system.

## 2.1 Introduction

Artificial Intelligence is the intelligence of machines and the branch of computer science that aims to create it. There are a lot of definitions of Artificial Intelligence in major AI textbooks, but no one know exactly a strict definition of Artificial Intelligence. According to Russell and Norvig [6] AI can be defined into two main dimensions (Table 1). The first dimension, the one on top are concerned with *thought processes* and *reasoning*, and the one on the bottom are concerned with *behavior*. The definition on the left measure success in terms of fidelity to human performance, and the ones on the right measure against an ideal concept of intelligence, which we will call rationality. If a system does the right thing given what it knows, we will call the system rational.

**Table 1: Artificial Intelligence Matrix**

| | **Human performance** | **Ideal concept of intelligence** |
|---|---|---|
| **Thought processes** | Systems that think like humans | Systems that think rationally |
| **Reasoning** | Systems that act like humans | Systems that act rationally |

In order to achieve our goal, we have to create a system that acts rationally given the situation. In our case, the situation is the traffic congestions. So given the traffic congestions data, our system should be able to act so as to achieve the best outcome which is finding the fastest route or fastest time, or where there is uncertainty, the best expected outcome. One way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals and then to act on that conclusion [6].

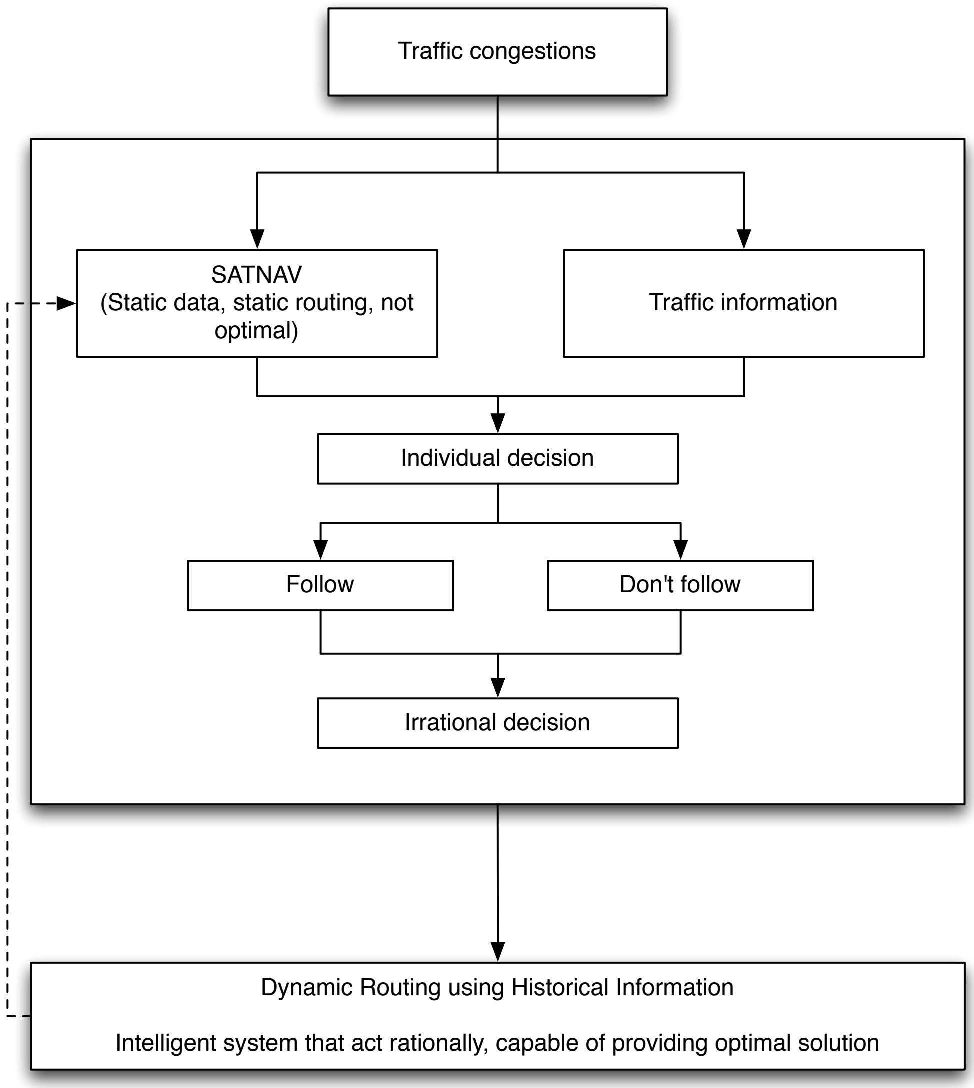

Traffic congestions
SATNAV
(Static data, static routing, not optimal)
Traffic information
Individual decision
Follow
Don't follow
Irrational decision
Dynamic Routing using Historical Information
Intelligent system that act rationally, capable of providing optimal solution

Figure 10: Dynamic Routing that can provide optimal solution

## 2.2 Shortest Path Problem

Shortest path problem is the problem of finding the shortest path of two connecting vertices from a weighted directed or undirected graph, such that the sum of the weights is minimized.

An example is finding the quickest way to get from one location to another on a road map; in this case, the vertices represent locations and the edges represent segments of road and are weighted by the time needed to travel that segment.

In mathematical terms, given a weighted graph with set of vertices $v \in V$, a set of edges $e \in E$ and a real-valued weight function $f : E \rightarrow R$. We have to find a path $P$ from source $v$ to each $v'$ of $V$ with the following constraints.

$$\min \sum_{p \in P} f(p)$$

**Equation 1: Shortest path function**

So that the sum is minimal among all paths connecting $v$ to each $v'$ of $V$.
The problem is also called the single-source shortest path problem, in which we have to find the shortest paths from a source vertex v to all other vertices in the graph.

The most important algorithms for solving this problem are Dijkstra's algorithm and Bellman-Ford algorithm. Both algorithms can solve single-source shortest path problem, the main difference is that Bellman-Ford algorithm can handle negative arcs weight which Dijkstra's algorithm fail to handle as noted by [7]. This is because the costs of previously visited vertices may decrease once we come across a negative arcs weight. It must be remembered that if there is a negative cycle there is no shortest path possible. In the implementation level, Bellman-Ford algorithm is a much slower algorithm compared to the Dijkstra's algorithm [8]. For problems with nonnegative arcs weight it has been proven that Dijkstra's algorithm is robust and the implementation of this algorithm is fairly simple [9].

## 2.3 Concept of DIANa

The main goal of this research is to explore the issue of static road routing in traffic and research the dynamic road routing algorithm. There are many possible ways of dynamic routing as suggested by [10], [11], [12], [13]. In this research we will use historical traffic information as the input data. Based on this idea, we will create a proof of concept by designing the historical data and implementing a dynamic road routing algorithm as well as analyzing the result of our research.

Before DIANa can be built, we should know what problems we could encounter with the static road routing, what the advantages and disadvantages are of static road routing and why we need it.

## 2.4 Static Road Routing

Everyone that drives a vehicle needs to go from one place to another. People tend to look for the fastest way to get from one place to the other. The most common way is by using an automotive navigation system. If you do not have an automotive navigation system then you can go to one of the many websites who can give you the fastest route to reach your destination, you can then print the route and take it with you.

Actually there is no problem with static road routing until we get in the middle of traffic jam. There are also a lot of information available on the internet or radio which can give us information about the traffic jam so that we can predict whether there are traffic jam in the route that we are going to take.

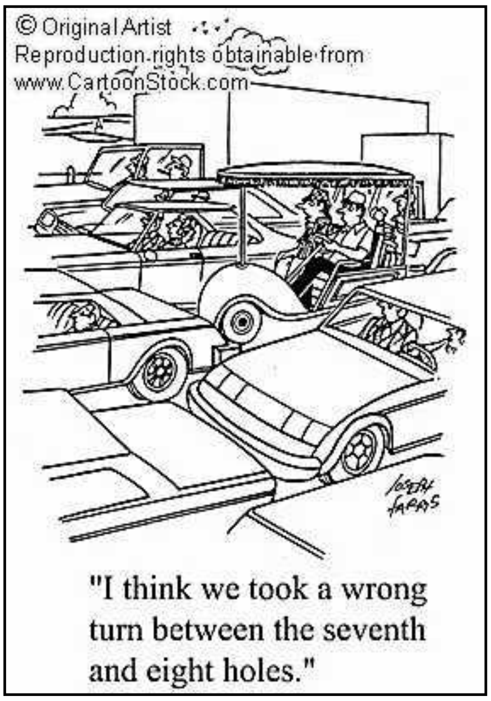

© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com
"I think we took a wrong turn between the seventh and eight holes."

Figure 11: Traffic jam illustration

## 2.5 Approach

As mentioned before, congestion is also a result of individual trip decision-making and driving behavior, which could be irrational at a time. Therefore, providing an individual with a tool that is capable of providing the right information with capability to induce the individual to act rationally, like providing recalculation and alternative route to minimize travel delay is essential. One of the solutions provided in this thesis is to create a system that acts rationally in the midst of traffic congestion, hence DIANa system.

This section deals with the approach and steps or input necessary to create such a system and whether such system would create partial solution to the traffic congestion caused by individual's decision making. First, we need to create a model of real world situation, this model is based on the historical traffic information available from a ready source. Second, this data model will be used as an input to create an intelligent system that is able to provide an optimal result given the data model.

**1. Modeling Real World Situation**

In order to prove our concept of DIANa using historical traffic information we need to be able to create a model of real world traffic information to compute travel time along roads or sections as a function of time of the day.

**2. Creating Intelligence System**

After establishing a model of real world traffic information, we then need to create an intelligent system that is able to provide a rational decision based on the data model. This system should be able to provide an optimal result when there is traffic congestion. The data model should provide an input to this system as serve as a basis to determine the fastest route or journey time, so that an individual using this system could decide, hopefully rationally based on the outcome provided, whether to stay with the shortest route given (static routing) or the alternative route with fastest route based on the historical information (dynamic routing).

## 2.6 Development Process

We implemented the system using the Incremental Development Process. Incremental development is an enhancement of the waterfall model as the prototyping is integrated. The application development phases of requirement analysis and global design are similar as in the waterfall model.

Waterfall model is actually a conventional software engineering paradigm and a sequential model in one direction to final stages. This means that it is a highly structured life cycle model and the result must be built and verified correctly at each phase.

However, with incremental development, the waterfall model is refined in such a way that the model is now characterized by the ability to add new requirements during the developing process. Successive incremental approach splits the phases concerning detailed design and implementation. It should enable dynamic development of system specifications.

When developing an AI system like a knowledge-based system, the expert is usually closely involved in all stages of the development that will result in a dynamic change of specifications and requirements. Due to the nature of this dynamic change of specification, the engineers will often need to revisit prior phase, which is not possible if the engineer uses the waterfall model. Therefore, using a conventional software engineering where specification and requirements are essentially static may not be suitable for an AI application development.

Our development process is as follows:

**1. Creating Network**

We need to create a model of traffic network for the whole Netherlands, including highways and provincial ways. We create the model by making selection of points on a traffic network that we can use as nodes in our model and by connecting two nodes with an edge we can model the traffic network.

**2. Collecting traffic information**

This step involves collecting a traffic database from a known source. One of the issues in this step is that some further action is necessary (for example cleaning up the data, filtering, etc) such that the data is reliable to be used in DIANa.

**3. Creating the database**

Once the data is setup and ready, we need to design historical traffic data model that can be used with our dynamic routing system that should result in the expected outcome.

**4. Developing Dynamic Route Finding Algorithm**

This step involves integrating the historical database by creating a dynamic route finding system that eventually can provide the optimal path for the given departure time also departure and arrival point.

**5. Developing Graphical User Interface**

Developing a Graphical User Interface is the next step after creating the dynamic route finding algorithm. Developing a user friendly interface is key since the system should be easily and readily used by the user to aid individual in making decision in their driving behavior and the different outcomes should be clearly presented.

**6. Testing**

The final step in development process includes testing and debugging DIANa to ensure that the system works and functions properly.

# Chapter 3

# Dynamic Routing

*Preview*

In this chapter we will explain how we could extend Dijkstra's algorithm into a dynamic routing algorithm using historical traffic data.

Our goal is to have an algorithm that can provide us with a result of a route with the shortest traveling time dependent on the historical traveling data of the road.

## 3.1 The Theory

The idea behind our dynamic routing algorithm is to use the historical data of the road networks and incorporate these data to find the fastest route based on departure time.

To compute the travel time from the starting point to destination point, we split up the route in a path along the nodes $n_0, n_1, \ldots, n_n$. In case of static routing we assume that the car drives at maximum speed $v(t_{i,j})$ between two nodes $[n_i, n_j]$ so we can calculate the travel time as $t_{i,j}$.

$$t_{i,j} = \frac{distance(n_i, n_j)}{Max(v(t_{i,j}))}$$

Equation 2: Travel time between nodes

But in time of rush hours or incidents, cars cannot drive with maximum speed. The speed $v(t_{i,j})$ depends on the time of the day, day of the week, and weather condition. If we know the speed along a link as a function of time, we are able to compute the travel time as a sum of travel time along the composing links.

$$TravelTime = \sum t$$

**Equation 3: Total travel time**

Based on the maximum speed, the distance, and the time delay of a specific road section on a specific time and day, we create a historical travel time data model with an assumption that this data is a good approximation for the whole year.

Our dynamic routing algorithm is basically the same principle as Dijkstra's algorithm, except for the manner of how it calculates the journey time between two nodes after relaxing its neighbors. Instead of looking from two-dimensional static data, our dynamic routing algorithm will read the travel time between nodes from three-dimensional data time from the database for that particular time. In this case, time is the third dimension.

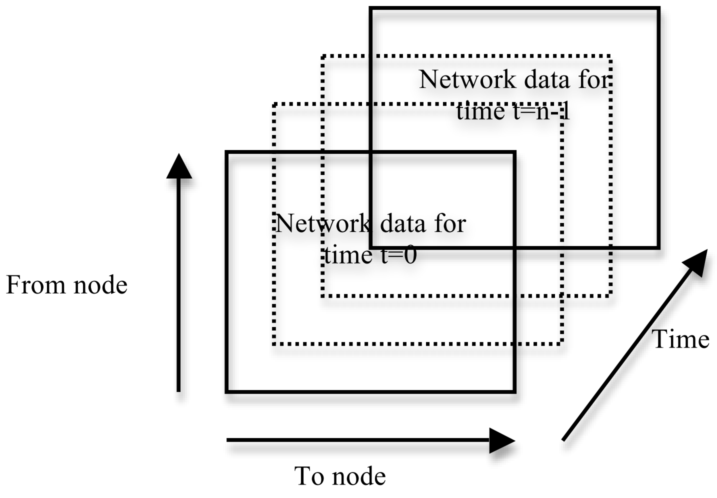


**Figure 12: Traffic data with time dimension**

## 3.2 Pseudo Code

The following variables will be used in order to explain the algorithm:

Table 2: DIANa algorithm variables description

| Variables | Description |
|---|---|
| $G$ | Weighted directed graph |
| $s$ | Source vertex |
| $start\_time$ | Departure time from source node |
| $d(v)$ | Stores the best estimate lowest cost from source $s$ to vertex $v$ |
| $\pi(v)$ | Stores the predecessor of each vertex on the shortest path from the source |
| $S$ | Set of settled vertices, the vertices whose lowest cost from the source has been computed |
| $Q$ | Set of unsettled vertices, the vertices whose cost from the source has to be computed |

**1. function Dijkstra(G, s, t):**

```
function Dijkstra(G, s, t)
        d := infinity // initialize d to infinity, π, S, and Q to empty
        S := empty set
        Q := empty set
        start_time := t

        Q := Q union {s} //first step add s to Q and d(s) to 0
        d(s) := 0
        π := s
        while Q is not an empty set           //the algorithm
                u := extractMinimum(Q)
                S := S union {u}              //add u to S
                relaxNeighbors(u)
```

Figure 13: Function Dijkstra(G, s, t) pseudo code

### 2. function relaxNeighbors(u):

```
function relaxNeighbors(u)
    for each vertex v adjacent to u

        if v not in set S & v not in set Q
            [u,v] = timeData(u,v,d(u))]
            d(v) := d(u) + [u,v] //calculate the distance
            π(v) := u         // set previous node of v as u
            Q := Q union {v}         //add v to Q

        if v not in set S & v in set Q
            [u,v] = timeData(u,v,d(u))]

            if d(v) > d(u) + [u,v]// a shorter distance exists
                d(v) = d(u) + [u,v]//recalculate the distance
                π(v) = u//set previous node of v as u
```

Figure 14: Function relaxNeighbors(u) pseudo-code

### 3. function extractMinimum(Q):

```
function extractMinimum(Q)
        find  vertex v with smallest distance d[v] from the source in Q
        remove v from Q and return v
```

Figure 15: Function extractMinimum(Q) pseudo-code

### 4. function timeData(u,v,d(u)):

```
function timeData(u,v,d(u))
        for time at startTime + d(u) look up the distance cost for
        [u,v] return the value
```

Figure 16: Function timeData(u, v, d(u)) pseudo-code

## 3.3 Flow Diagram of Dynamic Route Finding System

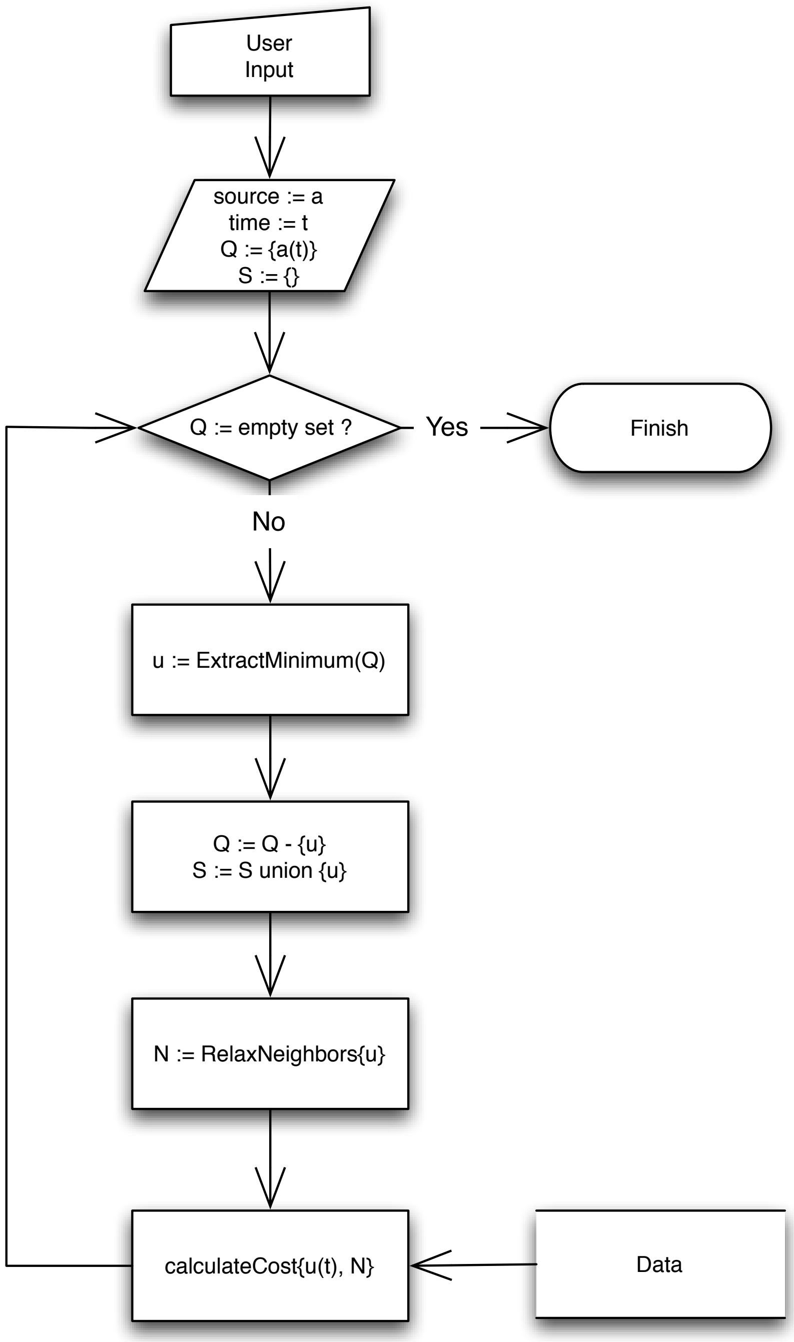


Figure 17: Flow Diagram Dynamic Route Finding System

## 3.4 Route Calculation of Dynamic Route Finding System

To illustrate the concept of DIANa, we use the following example below. The route consists of four nodes and five edges that serve as a connection between nodes. The starting point of the journey is node a with node d as our end point. The weight of each node may change in time.

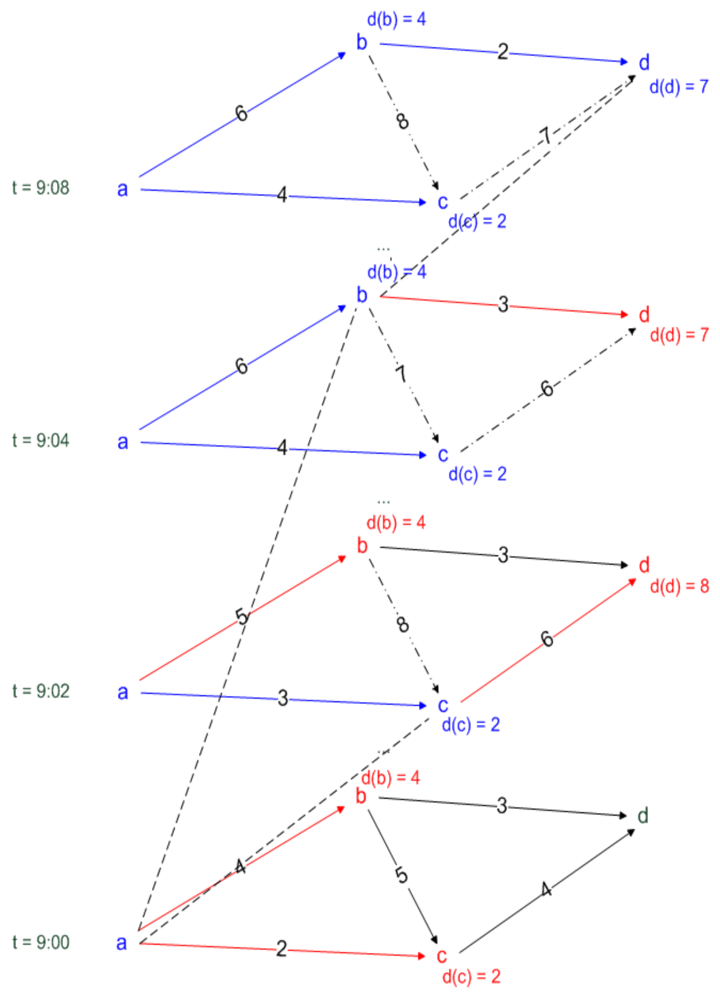


Figure 18: Route calculation in time dimension

With our DIANa algorithm using historical information of the edge, the resulting journey from node a to node d will be a-b-d with a total journey of seven. If we follow a static shortest path route, it will follow route a-c-d with a total journey of eight. From this simple illustration we have a performance wins of 1 point, which is satisfactory for a small traffic network.

Chapter 4

# Data Collection

*Preview*

Having reliable data in a research project is crucial. Our experiment and analysis should be based on a reliable and consistent data in order to be able to draw the right conclusions. This chapter focuses on the data collection method of the historical traffic data used for DIANa, where data are taken, and how they are calculated and processed.

## 4.1 Research Data

The research sample is based on collection of historical information of traffic condition on a certain day. The source of data is ANWB [14]. ANWB is one of the most reliable sources of traffic and road information in the Netherlands.

The sample used in this thesis is any four random Thursdays within a month. We realize that traffic intensity depends on the day of the week and also other events as already explained in Chapter 1. However, one of our goals is to show that dynamic routing really matters, so we chose a day with the most expected congestion in the week, which is Thursday as can be seen from figure 19.

The sample was checked properly to ensure there is no random distortion in the data, such as weather condition. Weather condition can severely impact the road condition and thus the traffic situation. During rain and snow, traffic congestion can double or even triple up than in usual days. Traffic congestion due to accident and roadwork are also eliminated from the data collection to get a normalize traffic data, which is only based on the greater demand than capacity on the road. Taking out outliers is important since the objective of this thesis is to use the historical data at different places to build an algorithm that can forecast congestion and thus calculating the fastest route to get to the end point.

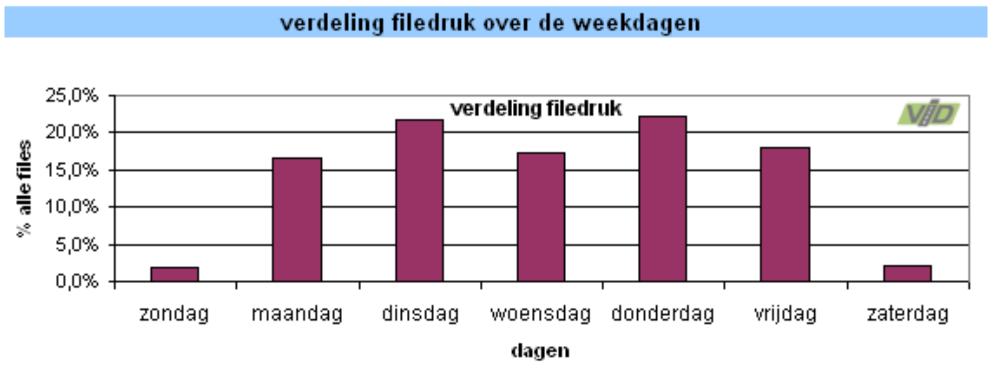


Figure 19: Traffic congestion from VID

## 4.2 Creating the Historical Data

As mentioned in the previous section, ANWB is the most reliable source of traffic information and hence is used as the source to collect the data for this thesis.

The data on the ANWB site is only real time data, hence there is no archive available publicly. ANWB was not willing to offer the database for free of charge for academic research. So we decided to poll the data from ANWB site.

The data collection is done firstly by getting traffic information from ANWB website[14] every two minutes. A script is made and scheduled to run every two minutes on our server twenty four hours long for two months to get this data. In total, per day (from 00.00 until 23.58) there are approximately 718 traffic information. From these 718 traffic information, a script is made to filter out all the duplications and unnecessary data resulting in approximately 170 traffic information data per day.

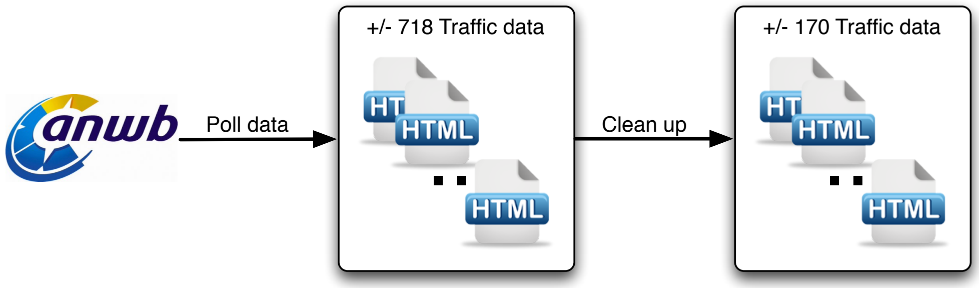
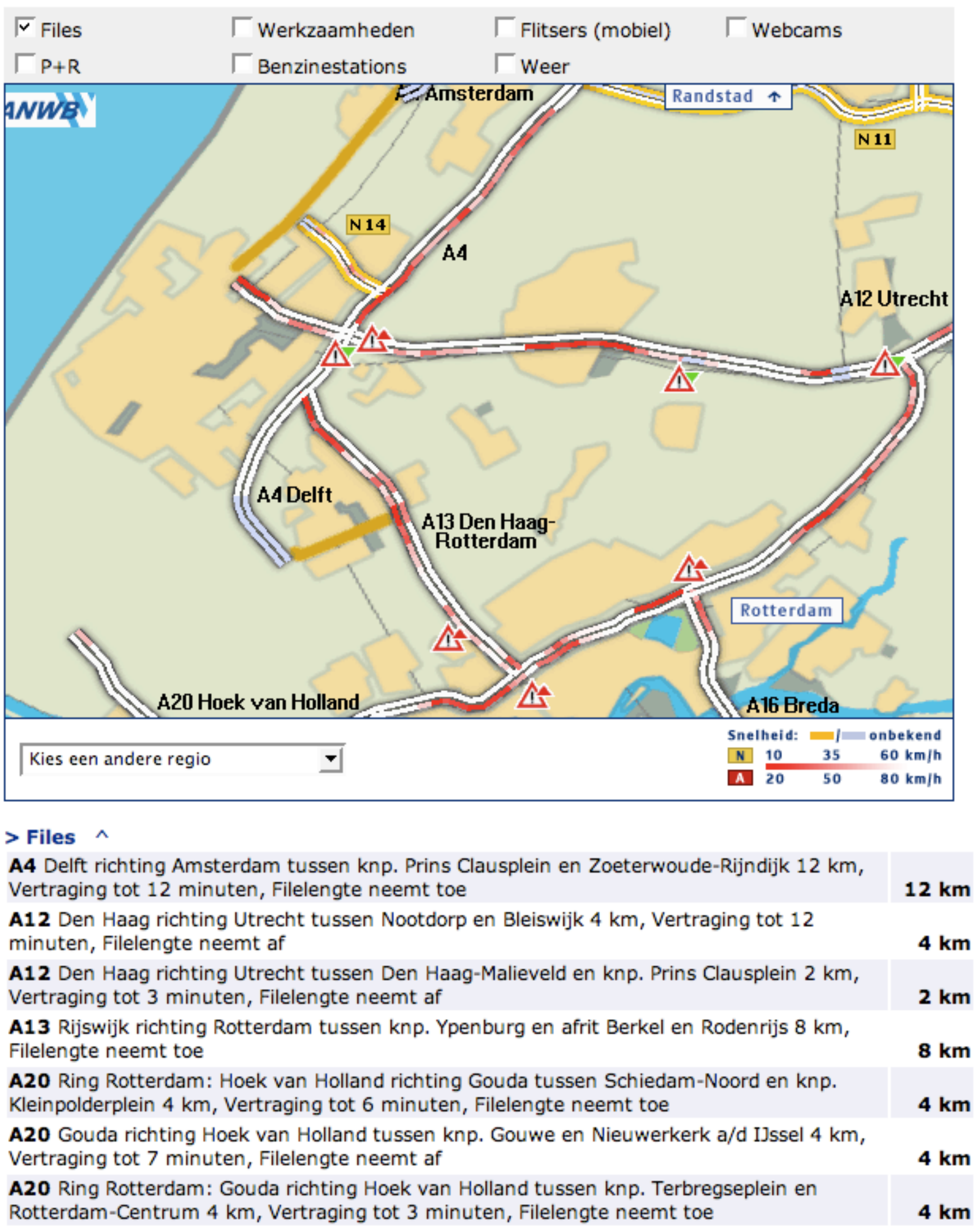


Figure 20: ANWB Data Polling

This traffic data gives information on highways that have delays or congestion - how many kilometers and how long the delay is. The delays are then put in the database to calculate the extra time needed to travel from one point to another. If the information contains only length and without duration, then the data is put in as missing values (NaN), which at the end will be linearly interpolated with the neighboring values (refer to chapter 4.4.1 for further explanation). The sampling frequency taken to record the traffic information is 10 minutes.



**> Files ^**

| | |
|---|---|
| **A4** Delft richting Amsterdam tussen knp. Prins Clausplein en Zoeterwoude-Rijndijk 12 km, Vertraging tot 12 minuten, Filelengte neemt toe | **12 km** |
| **A12** Den Haag richting Utrecht tussen Nootdorp en Bleiswijk 4 km, Vertraging tot 12 minuten, Filelengte neemt af | **4 km** |
| **A12** Den Haag richting Utrecht tussen Den Haag-Malieveld en knp. Prins Clausplein 2 km, Vertraging tot 3 minuten, Filelengte neemt af | **2 km** |
| **A13** Rijswijk richting Rotterdam tussen knp. Ypenburg en afrit Berkel en Rodenrijs 8 km, Filelengte neemt toe | **8 km** |
| **A20** Ring Rotterdam: Hoek van Holland richting Gouda tussen Schiedam-Noord en knp. Kleinpolderplein 4 km, Vertraging tot 6 minuten, Filelengte neemt toe | **4 km** |
| **A20** Gouda richting Hoek van Holland tussen knp. Gouwe en Nieuwerkerk a/d IJssel 4 km, Vertraging tot 7 minuten, Filelengte neemt af | **4 km** |
| **A20** Ring Rotterdam: Gouda richting Hoek van Holland tussen knp. Terbregseplein en Rotterdam-Centrum 4 km, Vertraging tot 3 minuten, Filelengte neemt toe | **4 km** |

Figure 21: ANWB traffic information

## 4.3 How to Calculate the Delays in the Database

As mentioned previously, there are two types of data coming from the traffic data – information with traffic delay in length and duration and information with length but without duration.

### 4.3.1 Traffic Information with Length and Delay Duration

There are a couple of ways to calculate the delay between nodes where there is length and duration:

**1. From junction to junction (Point A to B)**

To calculate the delay from junction to junction as illustrated in figure 18, we first need to know the name of the two nodes of the junction and the delay time between the two nodes (junctions) then add the delay time accordingly to the database, as explained below.

*Example:*

A4 Delft richting Amsterdam tussen knp. Badhoevedorp en knp. De Nieuwe Meer 3 km, Vertraging tot 1 minuut, Filelengte neemt toe 3 km

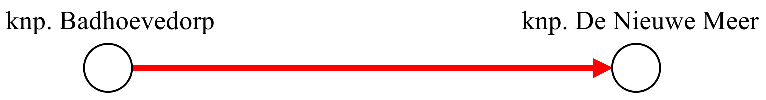


Figure 22: Traffic information from junction to junction

This is the simplest one. The database contains data from one node to another with name and unique ID. After checking at which highway the congestion happen, we then match it with the ID in the database. The extra time is then added to the normal time needed to travel from the one node to the other (Point A and B).

**2. From exit to exit between 2 adjacent junctions (Point A and B)**

When the traffic information is mentioned from exit to exit between two adjacent junctions, the same principal follows as the traffic information from junction to junction as in point 1. The delay time in between the two exits should be added to the junctions according to their ID in the database. Below are the examples and illustrations how the calculation should be done for various cases.

*a. Example:*

A2 Utrecht richting 's-Hertogenbosch tussen Waardenburg en afrit Kerkdriel 4 km, Vertraging tot 10 minuten, Filelengte neemt toe 4 km

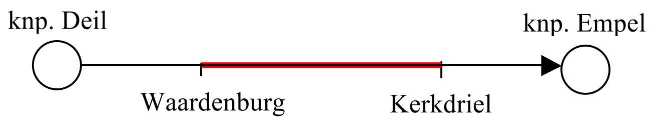


Figure 23: Traffic information between two junctions

The calculation is done the same way as in point 1, since the delays happen between the two nodes.

*b. Example:*

(1) A28 Utrecht richting Amersfoort tussen Soesterberg en afrit Maarn 2 km, Vertraging tot 2 minuten, Filelengte neemt toe 2 km

(2) A28 Utrecht richting Zwolle tussen Leusden-Zuid en Leusden 2 km, Vertraging tot 2 minuten 2 km

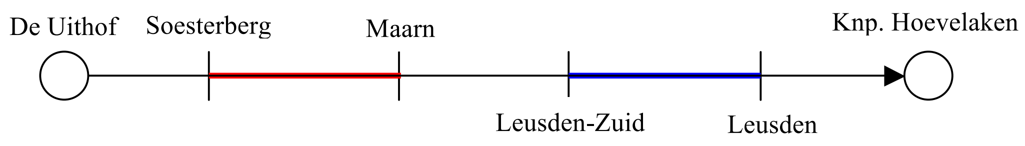


**Figure 24: Traffic information with several exits between two junctions**

To calculate the later example, we simply add up the delay time from exit 1 (Soesterberg) to exit 2 (Maarn) and delay time from exit 3 (Leusden-Zuid) to exit 4 (Leusden) since these congestions happens in the same road between two adjacent junctions as coded in the database.

**3. From exit/junction to exit/junction where there is/are another junction(s) in between (Point A to C)**

Calculating delay time from exit/junction to exit/junction where there is/are another junction(s) in between is slightly more complicated than the first two situations. Usually, the traffic information is given between point A and point C, as illustrated in figure 21. However, for the purpose of our database modelling, we need to know the time delay between point A to point B and point B to point C. We allocate the time delay proportionately according to the distance from point A to B and point B to point C from the total distance of point A to C, as can be seen below in the examples.

*Example:*

A12 Den Haag richting Arnhem tussen knp. Oudenrijn en Driebergen 9 km, Vertraging tot 23 minuten 9 km

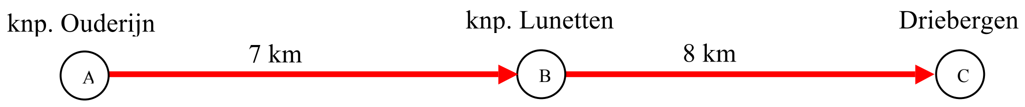


**Figure 25: Traffic information between several junctions**

$$Delay_{AB} = \frac{Length_{AB}}{Length_{AC}} \times Delay_{AC} = \frac{7}{15} \times 23\min \approx 11\min$$

Equation 4: Calculating delay between knp. Oudenrijn and knp. Lunetten

$$Delay_{AB} = \frac{Length_{BC}}{Length_{AC}} \times Delay_{AC} = \frac{8}{15} \times 23\min \approx 12\min$$

Equation 5: Calculating delay between knp. Lunetten and Driebergen

In this case the delay between junction A and junction B is calculated proportionally with the distance between A to B and A to C. This is also applicable to calculate the delay between junction B and junction C

**4. From two traffic information where the exits/junctions overlap each other**

Sometimes the traffic information was given between exits or junctions that overlap each other. In this case, we need to be careful not to double count the time delays between the junctions. Below examples and figures show how we should calculate the time delay when overlap exists.

*Example*:

(1) A12 Den Haag richting Utrecht tussen Nootdorp en Zoetermeer 2 km, Vertraging tot 7 minuten, Filelengte neemt af 2 km

(2) A12 Den Haag richting Utrecht tussen Zoetermeer-Centrum en Zevenhuizen 6 km, Vertraging tot 9 minuten, Filelengte neemt af 6 km

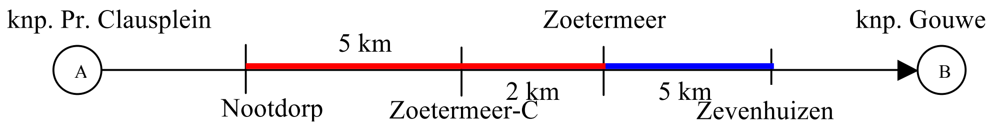


Figure 26: Overlapping traffic information

$$Delay_{AB} = Delay_{Nootdorp-Zoetermeer} + Delay_{Zoetermeer-Zevenhuizen}$$

$$Delay_{AB} = 7\min + \left(\frac{5}{7} \times 9\min\right) \approx 13\min$$

Equation 6: Calculating delay between Nootdorp and Zevenhuizen

The calculation is basically similar to point 2.b, which is to add up the two traffic delay information. However the overlap point should be taken out to avoid double counting by proportionally calculate the distance as in point 3 above.

### 4.3.2 Traffic Information with Length and No Delay Duration

Traffic information with length but without delay duration, then this information is regarded as a missing value. This data is then put in the database as "NaN". The objective of inputting this data as a missing value is to get interpolation from the available data before this missing value and the next available data after.

*Example*:
A12 Utrecht richting Den Haag tussen knp. Lunetten en knp. Oudenrijn 3 km, Filelengte neemt af

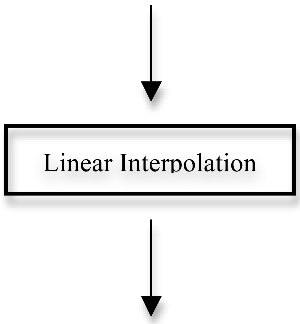
## 4.4 Data Processing

After collecting data from ANWB, the next step is to deal with these data in such a way that it can be meaningful and can be interpreted in the right way. There are two steps taken in the data processing: linear interpolation to deal with all the missing values and then averaging the data from the four Thursday to increase accuracy and validity.

### 4.4.1 Linear Interpolation

The next thing we need to do is to calculate all missing values at a point of time. We calculate all missing values by simply linearly interpolating the missing values.

| From | To | … | 17:50 | 18:00 | 18:10 | 18:20 | 18:30 | … |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Knp_Holendrecht_2 | Vinkeveen | | 9.00 | NaN | NaN | NaN | 5.00 | … |
| | | | | | | | | |



| From | To | … | 17:50 | 18:00 | 18:10 | 18:20 | 18:30 | … |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Knp_Holendrecht_2 | Vinkeveen | | 9.00 | 8.00 | 7.00 | 6.00 | 5.00 | … |
| | | | | | | | | |

Figure 27: Linear interpolation over NaN values

### 4.4.2 Averaging

As we can see in figure 27, the horizontal line is the minimum traveling time to travel from knp. Watergraafsmeer to knp. Diemen if there is no known delay.

We can see from the figure that traffic jam in this section of the road usually happen between 15:00 and 20:00 with a peak delay around 17:30 which can be up to 16 minutes delay.

To create a representative traffic data we calculate the average of each road sections from four sample data, which we recorded on Thursday.

$$D(t) = \frac{TH(t,1) + TH(t,2) + TH(t,3) + TH(t,4)}{4}$$

Equation 7: Historical traffic data calculation

With $D(t)$ is historical traffic data and $TH(t,a)$ is Thursday traffic data at time $t$ for day $a$.

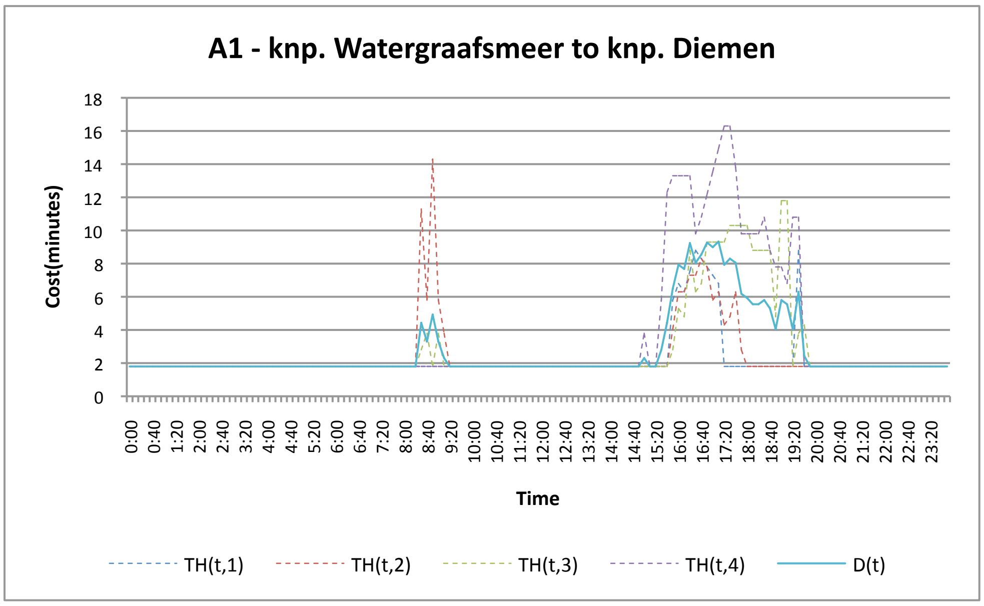


Figure 28: Plot of traffic data from node Watergraafsmeer to node Diemen

# Chapter 5

# Conceptual Design

*Preview*

This chapter focuses on global conceptual design of this research. Conceptual design involves the system architecture, modeling the traffic data, and designing the database itself. Furthermore, there will be an emphasis on how to use the traffic database on dynamic routing algorithm.

## 5.1 Research Goal

The main goals in the development of DIANa presented in this thesis are the enhancement of the nowadays Static Routing, by creating a proof of concept of DIANa using historical traffic data and to show that dynamic routing has a significant impact on the traveling time.

We still have to create a working prototype that should be user accessible online or offline with PC or portable computer.

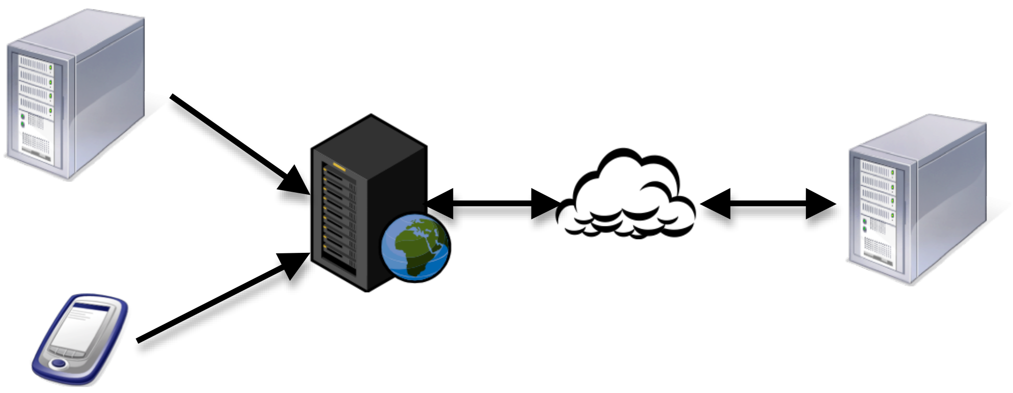


Figure 29: DIANa Infrastructure

## 5.2 System Architecture

DIANa is a complete system with user input and graphical user interface. The main components of the system's architecture of our DIANa can be seen below.

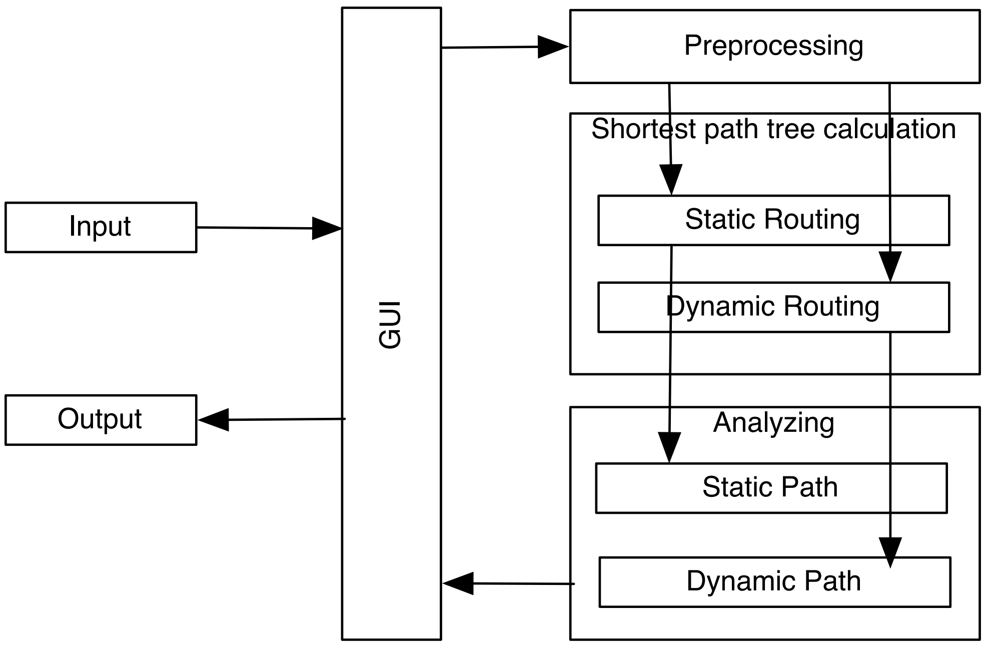


Figure 30: DIANa System architecture

First, the user can put in his or her preferences like departure time as well as departure and destination point. The preprocessing is used to convert user input to be used in DIANa and also to load the historical traffic data to DIANa.

Second is the shortest path tree calculation section. A single-source shortest path tree will be calculated for both dynamic and static routing algorithm, which will result in a shortest path tree from departure point to all possible arrival points.

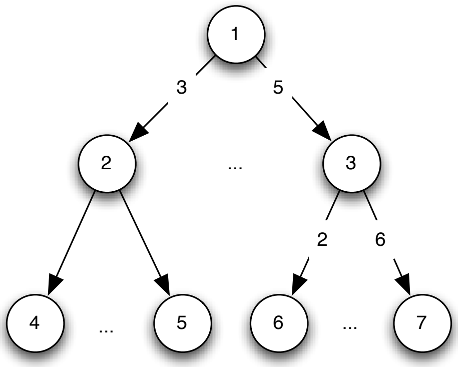


Figure 31: Shortest path tree

Finally, in the analyzer section, the shortest path and the traveling time will be calculated by back-tracing from the destination point to the departure point for both static and dynamic routing.

## 5.3 Modeling Traffic Data

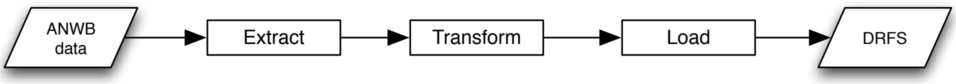


Figure 32: Modeling traffic data

We create a model of the historical traffic data in the following steps:

1. **Extract data**

First step is to extract the data from the source, which is in our case ANWB traffic information.

2. **Transform data**

We need to transform the extracted data to design a relational database model of traffic information.

3. **Load data**

Finally we load the relational database model to our dynamic route finding system.

### 5.3.1 Extract Data

As a start, in order to be able to build a traffic database, a directed graph, which consists of junctions on highways and regional roads, needs to be created. These are only junctions that connect one highway to another, from highway to regional road and vice versa, or regional road to regional road.

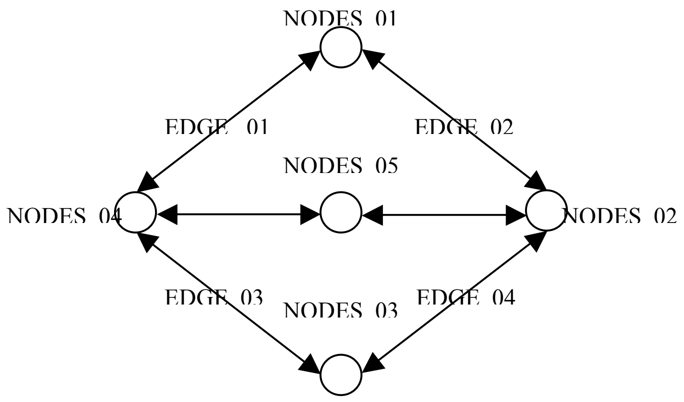


**Figure 33: Directed graph**

**Table 3: Traffic database**

| RD_ID | RD_NAME | ND_NAME_FR | ND_ID_FR | ND_NAME_TO | ND_ID_TO | DUR | … | 17:10 | … |
|---|---|---|---|---|---|---|---|---|---|
| 2 | A1 | knp_Watergraafsmeer | 1 | knp_Diemen | 2 | 1.80 | … | 9.30 | … |
| 3 | A1 | knp_Diemen | 2 | knp_Muiderberg | 3 | 4.20 | … | 11.70 | … |
| 4 | A1 | knp_Muiderberg | 3 | Laren | 4 | 6.60 | … | 6.60 | … |
| … | … | … | … | … | … | … | … | … | … |

The nodes in the directed graph are junctions and the edges that connect pairs of nodes are the roads direction between junctions. Every nodes and edges will have a name and unique ID. Each road has length and maximum speed (taken from http://www.maximumsnelheden.info). For this research, the most relevant factor is time. The time is calculated based on the road length and the maximum speed of that road. If there is delay due to congestion, then the extra time delay is added in the column for that specific time (refer to Table 3).

Table 4: Traffic database fields

| Column | Field | Description |
|---|---|---|
| 1 | RD_ID | ID of road |
| 2 | RD_NAME | Road name |
| 3 | ND_NAME_FR | From node name of link |
| 4 | ND_ID_FR | From node ID of link |
| 5 | ND_NAME_TO | To node name of link |
| 6 | ND_ID_TO | To node ID of link |
| 7 | DUR | Initial duration in minutes |
| 8 to 151 | SAMPLE_TIME | Duration with sample time of 10 minutes |

### 5.3.2 Data Transformation

In this section we will explain how we convert the collected data into data model to be used in DIANa.

#### 5.3.2.1 Historical Data

The historical data is a matrix with a dimension of 808 x 145, which consists of 808 road connections, road id column and 144 timestamp data. Timestamp 0 is 00:00 and timestamp 143 is 23:50.

Table 5: Historical traffic data

| ROAD_ID | 0 | 1 | … | 142 | 143 |
|---|---|---|---|---|---|
| 0 | 1.8 | 1.8 | … | 1.8 | 1.8 |
| … | … | … | … | … | … |

#### 5.3.2.2 Network Data

The network data has a dimension of 808 x 3 records, which consists of 808 road connections between two nodes.

Table 6: Network data

| ROAD_ID | FROM_NODE_ID | TO_NODE_ID |
|---|---|---|
| 0 | 1 | 2 |
| 1 | 2 | 3 |
| … | … | … |

#### 5.3.2.3 Intersection Data

Intersection data includes the intersection ID’s and name as well as the intersection x- and y-coordinate.

Table 7: Intersection data

| NODE_ID | NODE_NAME | X_COORDINATE | Y_COORDINATE |
|---|---|---|---|
| 1 | knp_Watergraafsmeer | 840 | 900 |
| 2 | knp_Diemen | 860 | 920 |
| … | … | | … |

#### 5.3.2.4 Static Data

Static data contains the shortest traveling time of a road, which is table that consists of 808 records with 2 columns.

Table 8: Network weight data

| ROAD_ID | TRAVEL_TIME |
|---|---|
| 0 | 1.8 |
| 1 | 4.2 |
| … | … |

#### 5.3.2.5 Nodes Data

Nodes data is a matrix of 808 x 4, which consists of 808 road connections each with its id, name, length, and maximum speed allowed.

Table 9: Historical traffic data

| ROAD_ID | NAME | LENGTH | MAX_SPEED |
|---|---|---|---|
| 0 | A1 | 3 | 100 |
| 1 | A1 | 7 | 100 |
| … | … | … | … |

### 5.3.3 Load Data - Using Traffic Database in Dynamic Routing Algorithm

DIANa chooses the best route based on the departure time and location. Instead of using a static data to calculate the shortest traveling path, our system will be using a historical traffic data depending on the departure time. Every time our DIANa system needs to calculate the traveling time between two nodes, it will look into our historical traffic data for that particular time. As we have already explained in the previous chapter on how the algorithm works, DIANa will keep looping until every other node is in a settled set.

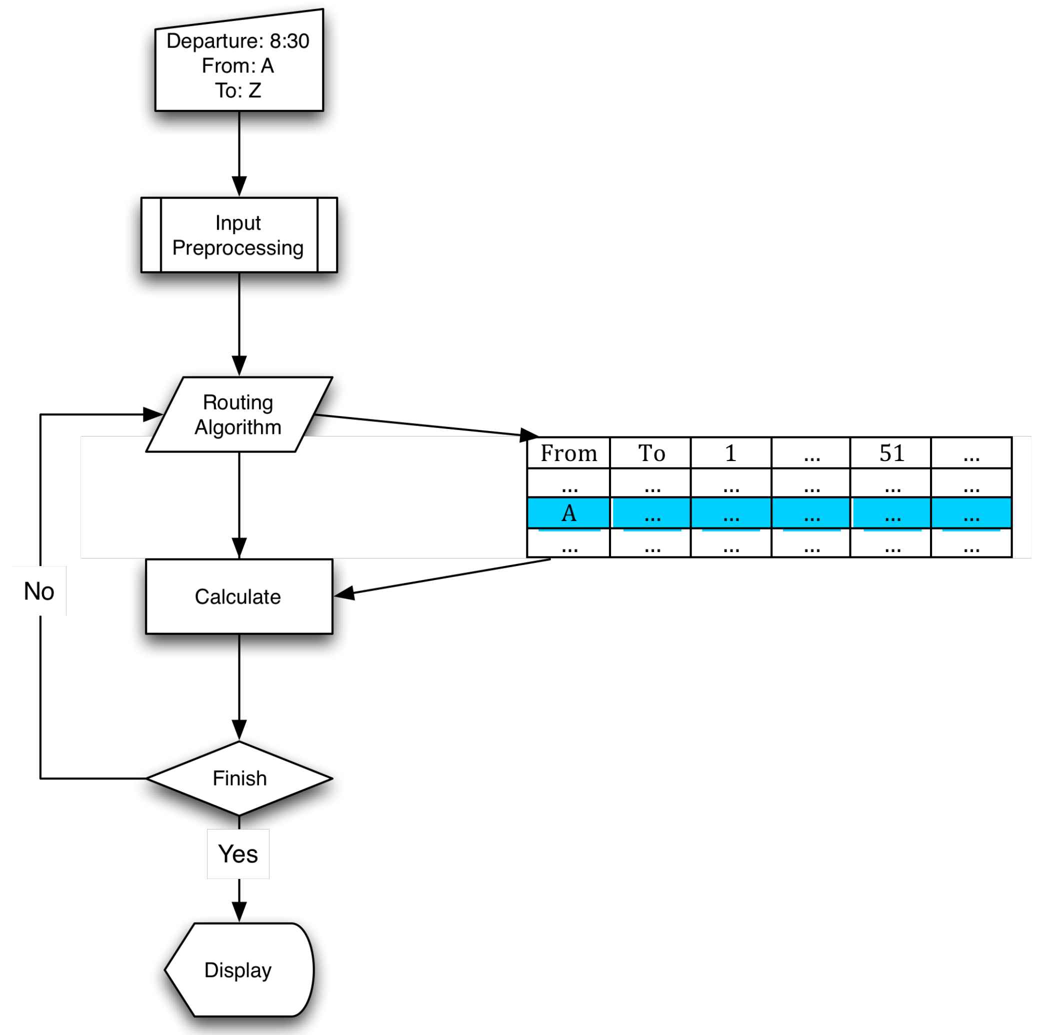
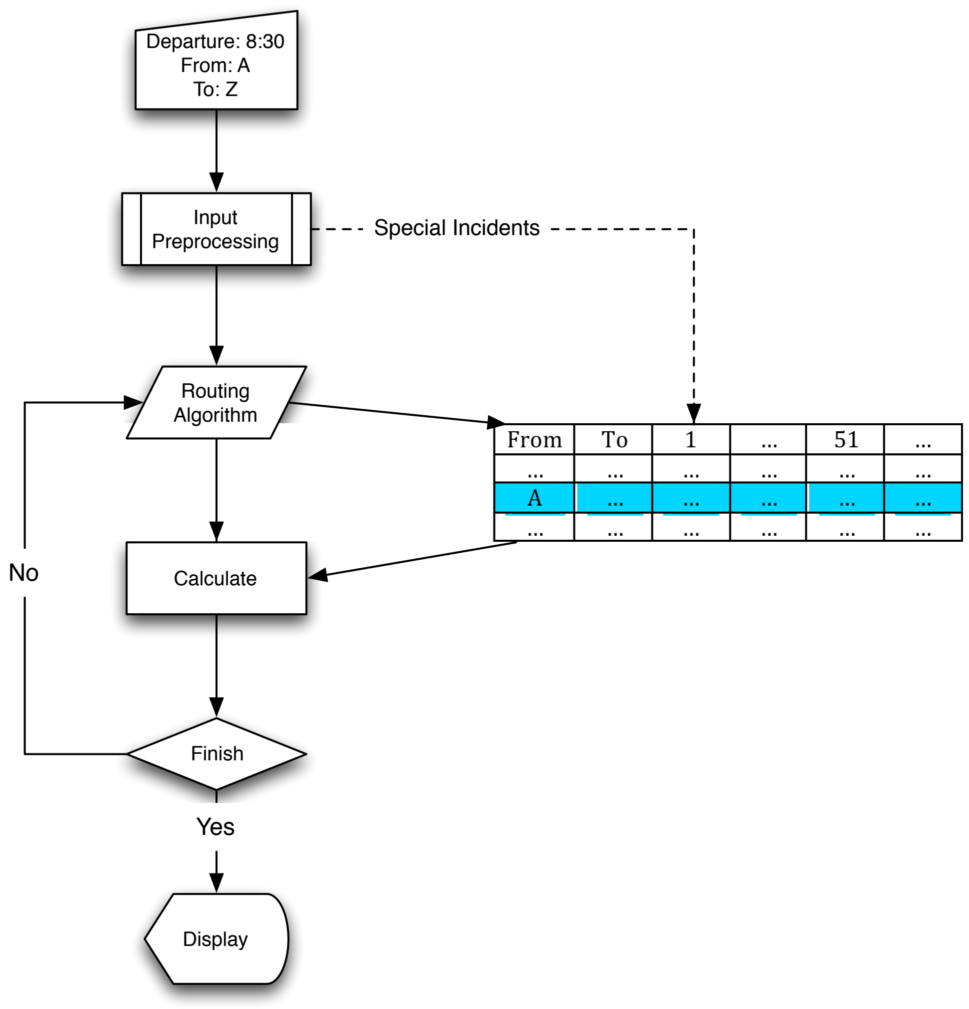


Figure 34: DIANa Flow Diagram

## 5.4 Special Incident

There is a possibility for user to provide extra information for the historical traffic data, for example if there is a road blockage, etc. If user provides a special incident to DIANa, then that particular road will not be used in our calculation of shortest path.


Departure: 8:30
From: A
To: Z
Input Preprocessing
Special Incidents
Routing Algorithm
Calculate
Finish
No
Yes
Display
From
To
1
...
51
A

Chapter 6

# Detailed Design

*Preview*

This chapter focuses on the detailed design of the implemented algorithm, which further explains the implemented algorithm, the requirements, and the use case and diagram.

## 6.1 Proof of Concept

This research is meant as a proof of concept for further research. Proof of concept is a short and/or incomplete realization of a certain method or ideas to demonstrate its feasibility, or a demonstration in principle, whose purpose is to verify that some concepts or theories are probably capable of exploitation in a useful manner. The proof of concept is usually considered as a milestone on the way to a fully functioning prototype.

Hence, there is a need to create a real world model of traffic information to be used here in order to recreate or simulate the real situation. There might be some limitations or discrepancies found in this research. However, the overall result of this study after doing some validity and reliability test should be considered as the start of a fully functioning application of dynamic routing system.

## 6.2 The Implemented Algorithm

This section will further elaborate on the three implemented algorithm used in this research:

1. Shortest path using static data
2. Static routing using historical travel time
3. Dynamic routing using historical data.

### 6.2.1 Static Routing

In this section, we will explain the shortest path algorithm using static data. The shortest path algorithm that we used is based on the Dijkstra's algorithm. After we calculate the shortest path, we then split the manner we calculate the traveling time in two ways. First, we calculate the traveling time with the assumption that there is no traffic and then we calculate the traveling time using our historical traffic data for that particular shortest path.

#### 6.2.1.1 Shortest Path using Static Data

The static routing we implement here is a Dijkstra's based on the shortest path algorithm. This algorithm calculates the shortest path using static data assuming that there is no traffic congestion at any given time.

#### 6.2.1.2 Static Routing using Historical Travel Time

We implement another static routing with a different way of calculating the travel time. Instead of assuming there is no traffic congestion, we assume that the resulting shortest path contains of traffic congestion and the resulting traffic data containing delay from the congestion is used as the travel time for our historical traffic data.

### 6.2.2 Dynamic Routing using Historical Data

This section deals with the implementation of the dynamic route algorithm, taking traffic congestions into consideration. This implies that this algorithm should function as to finding whichever routes with the shortest travel time regardless the distance.

## 6.3 Software's used

### 6.3.1 NetBeans IDE

NetBeans refers to both a platform for the development of applications for the network (using Java, JavaScript, PHP, Python, Ruby, Groovy, C, and C++), and an integrated development environment (IDE) developed using the NetBeans Platform.

The NetBeans Platform allows applications to be developed from a set of modular software components called modules. A module is a Java archive file that contains Java classes written to interact with the NetBeans Open APIs and a manifest file that identifies it as a module. Applications built on modules can be extended by adding new modules. Since modules can be developed independently, applications based on the NetBeans platform can be extended by third party developers.

### 6.3.2 Matlab

MATLAB is a numerical computing environment and fourth generation programming language. Developed by The MathWorks, MATLAB allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it is numeric only, an optional toolbox uses the MuPAD symbolic engine, allowing access to computer algebra capabilities. An additional package, Simulink, adds graphical multidomain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MathWorks claimed that MATLAB was used by more than one million people across industry and the academic world[15].

## 6.4 Requirements

In this section we describe the main requirements of the concept we use in this research. Requirements are the basis of all software applications as they focus on the ability of the applications. Requirements can be described in many ways. The most common division is by describing requirements in functional, non-functional, and pseudo requirements.

### 6.4.1 Functional Requirements

Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts. This section will describe the functional requirements related to the DIANa system.

The functional requirements of the DIANa system are the following:

**1. Read the historical traffic data and network data**

In order to calculate the fastest route the historical traffic data and network data should be loaded in to DIANa. After this process, the historical data and network data will be saved in an array list that later will be used by the system for calculation.

**2. Visualize the network data**

After loading the network data, the system will visualize the data, in this case map of the Netherlands. Every junction will be number-coded and every big city area will be text-coded so that the visualization is clear for the user.

**3. Accept user input**

The system should be able to accept user input. The user inputs in this case are:

- Departure and arrival node
- Departure time
- Road blockage
- Command to calculate the route.

**4. Button and drop down menu**

Drop down menus to select departure node, arrival node, departure time, and road blockages should be implemented in the system. This is to ensure that user can easily access the button without the need to type input text.

Button with the function to add and delete road blockages as well as to command the system to calculate the route is also implemented in the system.

**5. Show resulting route**

There are two resulting routes, dynamic route and static route. The user will be able to see the resulting routes displayed on the screen. There will be three display windows shown. The top window will show the map of the Netherlands with two different routes each with different color. The other two windows at the bottom will show more detailed text, route, cost, and arrival time.

**6. World Wide Web accessibility**

One of the advantages of the system is that it should be user accessible from the World Wide Web. The formatting layout of the system should also be PC, operating system, and browser friendly. A server that will be online for twenty four hours should be provided and managed by server administrator.

**7. Offline accessibility**

The system should be able to run on the user PC without the need of an Internet connection.

### 6.4.2 Non-Functional Requirements

In this section we will describe non-functional requirements. Non-functional requirements describe user-visible aspects of the systems that are not directly related with the functional behavior of the system.

The following are the non-functional requirements for DIANa:

**1. User interface**

The system will have a user-friendly graphical user interface. The graphical user interface will be designed for simplicity such that only the functionalities that are relevant for the user are shown and any other functionalities will be disabled.

**2. Performance**

The system should be able to respond fast even if there is a lot of data that needs to be processed. This should not result in a delay of more than 3 seconds.

**3. Hardware considerations**

The system will require a computer with any operating system and any browser. For online access of the system, users need to have an Internet connection that is able to make a connection with the server.

**4. Error handling**

Every user or system error will be made visible for the interaction between user and the system.

**5. Modifiability**

The system should be user modifiable for future research. This is important so that future researcher can add extra functionalities and enhance the overall usability of the system.

### 6.4.3 Pseudo Requirements

Due to the system requirements that the system should be operating system independent, online accessible, and offline accessible, the system is written using Java. NetBeans IDE is used for development environment. To evaluate the result, Matlab and Microsoft Excel will be used. As for documentation, Microsoft Word, OmniGraffle, Microsoft Visio, and Papers for Mac will be used.

### 6.4.4 System Requirements

In order to get most of DIANa, you need a system with the following recommended requirements:

- Java version: Version 6.0
- Operating system: All Platforms that support Java 6.0
- Memory: Minimum 128 mb

## 6.5 Use Case

Use case is used to represent the functionality of the system. Use case focuses on the behavior of the system from an external point of view. A use case describes a function provided by the system that yields a visible result for an actor. An actor describes any external entity that interacts with the system. Actors initiate a use case to access the system functionality. The actors are outside the boundary of the system, whereas the use case is inside the boundary of the system.

The use case diagram of DIANa can be seen in figure 35 ad we will also explain two use cases from the use case diagram in table 10 and table 11.

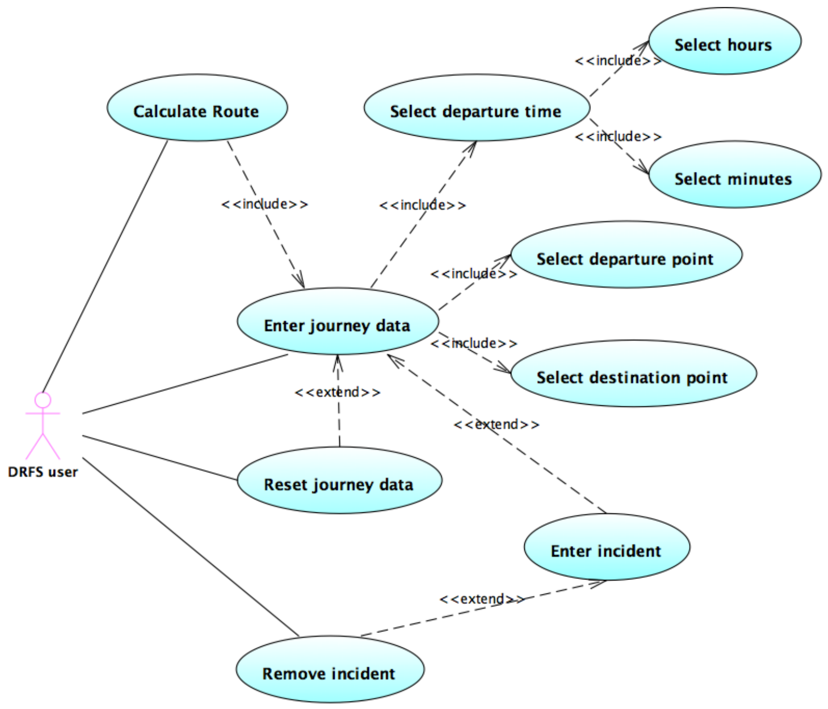


Figure 35: Use case diagram

**Table 10: Use case CalculateRoute**

| | |
|---|---|
| Use case name | CalculateRoute |
| Participating actor | Invoked by DIANaUser |
| Entry condition | 1. The DIANaUser fill up the journey data by invoking EnterJourneyData use case |
| Flow of events | 2. After DIANaUser reviews the journey information, DIANaUser select Get Route button |
| Exit condition | 3. Shortest route for the journey will be displayed on the screen |
| Special requirements | Shortest path must be calculated and displayed within 5 seconds |

**Table 11: Use case EnterJourneyData**

| | |
|---|---|
| Use case name | EnterJourneyData |
| Participating actor | Invoked by CalculateRoute use case |
| Entry condition | 1. The DIANaUser starts the DIANa |
| Flow of events | Step 2 to 4 can be done in random order:<br>2. DIANaUser entering the departure time by invoking SelectDepartureTime use case<br>3. DIANaUser entering the departure point by invoking SelectDeparturePoint use case<br>4. DIANaUser entering the destination point by invoking SelectDestination point use case<br><br>Step 5 is optional:<br>5. DIANaUser entering special incidents by invoking Incident use case |
| Exit condition | 6. DIANaUser have already completed the journey data |
| Special requirements | - |

## 6.6 Class Diagram

Class diagrams describe the structure of the system in terms of classes and objects. Classes are abstractions that specify the attributes and behavior of a set of objects. Objects are entities that encapsulate state and behavior. Each object has an identity that can be referred individually and is distinguishable from other objects.

The following is the class diagram of the system:

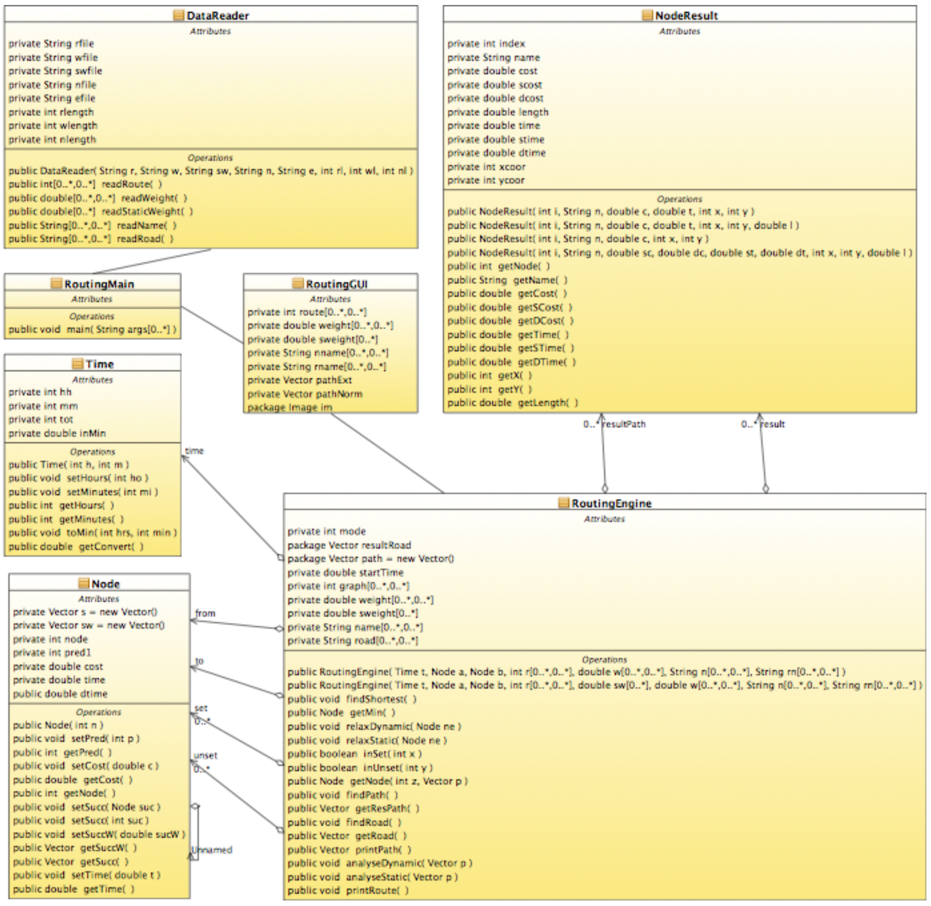

DataReader
Attributes
private String rfile
private String wfile
private String swfile
private String nfile
private String efile
private int rlength
private int wlength
private int nlength
Operations
public DataReader( String r, String w, String sw, String n, String e, int rl, int wl, int nl )
public int[0..*,0..*] readRoute( )
public double[0..*,0..*] readWeight( )
public double[0..*] readStaticWeight( )
public String[0..*,0..*] readName( )
public String[0..*,0..*] readRoad( )
NodeResult
Attributes
private int index
private String name
private double cost
private double scost
private double dcost
private double length
private double time
private double stime
private double dtime
private int xcoor
private int ycoor
Operations
public NodeResult( int i, String n, double c, double t, int x, int y )
public NodeResult( int i, String n, double c, double t, int x, int y, double l )
public NodeResult( int i, String n, double c, int x, int y )
public NodeResult( int i, String n, double sc, double dc, double st, double dt, int x, int y, double l )
public int getNode( )
public String getName( )
public double getCost( )
public double getSCost( )
public double getDCost( )
public double getTime( )
public double getSTime( )
public double getDTime( )
public int getX( )
public int getY( )
public double getLength( )
RoutingMain
Attributes
Operations
public void main( String args[0..*] )
RoutingGUI
Attributes
private int route[0..*,0..*]
private double weight[0..*,0..*]
private double sweight[0..*]
private String nname[0..*,0..*]
private String rname[0..*,0..*]
private Vector pathExt
private Vector pathNorm
package Image im
Time
Attributes
private int hh
private int mm
private int tot
private double inMin
Operations
public Time( int h, int m )
public void setHours( int ho )
public void setMinutes( int mi )
public int getHours( )
public int getMinutes( )
public void toMin( int hrs, int min )
public double getConvert( )
time
0..* resultPath
0..* result
RoutingEngine
Attributes
private int mode
package Vector resultRoad
package Vector path = new Vector()
private double startTime
private int graph[0..*,0..*]
private double weight[0..*,0..*]
private double sweight[0..*]
private String name[0..*,0..*]
private String road[0..*,0..*]
Operations
public RoutingEngine( Time t, Node a, Node b, int r[0..*,0..*], double w[0..*,0..*], String n[0..*,0..*], String rn[0..*,0..*] )
public RoutingEngine( Time t, Node a, Node b, int r[0..*,0..*], double sw[0..*], double w[0..*,0..*], String n[0..*,0..*], String rn[0..*,0..*] )
public void findShortest( )
public Node getMin( )
public void relaxDynamic( Node ne )
public void relaxStatic( Node ne )
public boolean inSet( int x )
public boolean inUnset( int y )
public Node getNode( int z, Vector p )
public void findPath( )
public Vector getResPath( )
public void findRoad( )
public Vector getRoad( )
public Vector printPath( )
public void analyseDynamic( Vector p )
public void analyseStatic( Vector p )
public void printRoute( )
Node
Attributes
private Vector s = new Vector()
private Vector sw = new Vector()
private int node
private int pred1
private double cost
private double time
public double dtime
Operations
public Node( int n )
public void setPred( int p )
public int getPred( )
public void setCost( double c )
public double getCost( )
public int getNode( )
public void setSucc( Node suc )
public void setSucc( int suc )
public void setSuccW( double sucW )
public Vector getSuccW( )
public Vector getSucc( )
public void setTime( double t )
public double getTime( )
from
to
set
0..*
unset
0..*
Unnamed

Figure 36: Class diagram

## 6.7 State Chart Diagram

A notation for describing sequence of states of an object goes through in response to external events.

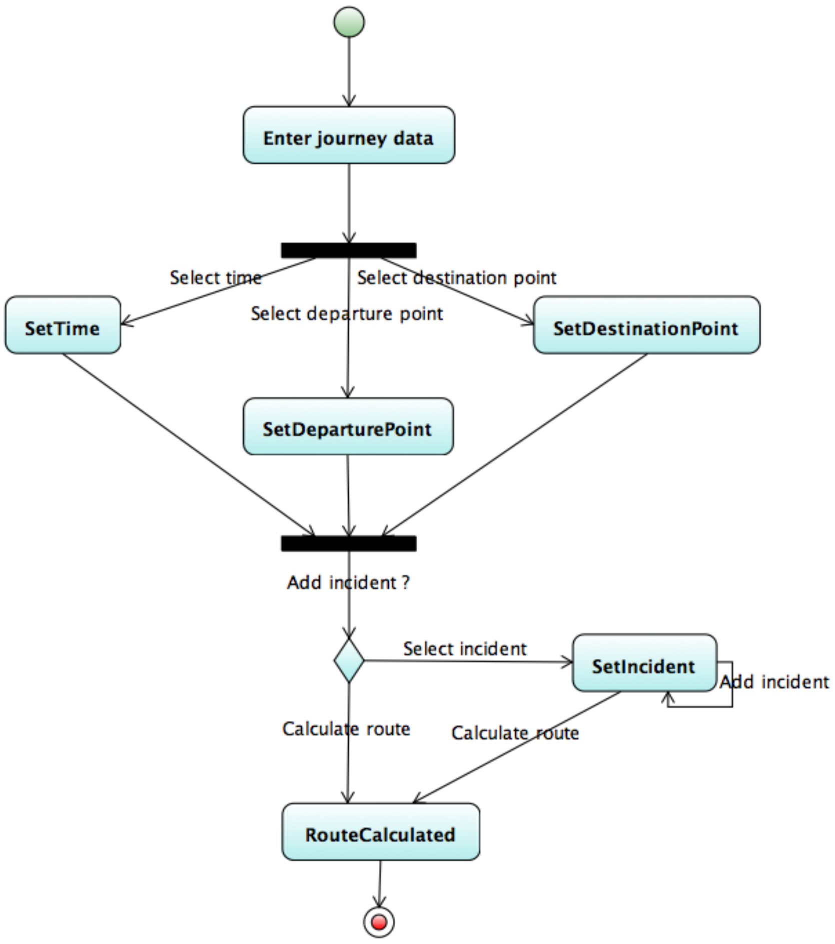


Figure 37: DIANa State chart diagram

## 6.8 Sequence Diagram

Sequence diagram describes patterns of communication among a set of interacting objects. An object interacts with another object by sending messages. The reception message by an object triggers the execution of an operation, which in turn may send messages to other objects. Arguments may also be passed along with a message and are bound to the parameters of the executing operation in the receiving object. In the figure below, a sequence diagram is given when a user uses the system to calculate the shortest path given the user inputs.

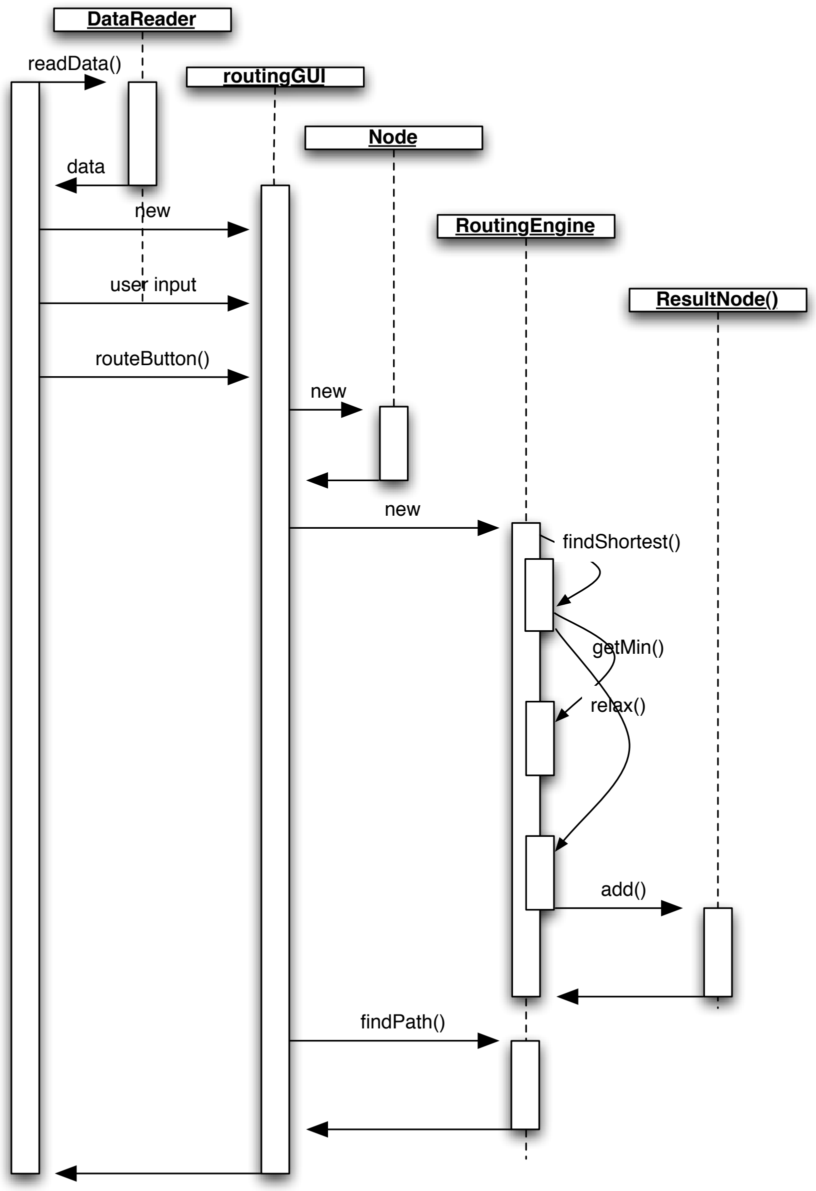


Figure 38: Sequence diagram

## 6.9 Class Descriptions

This section deals with the description of the most important Java Classes with its methods and attributes.

### 6.9.1 Class: Data Reader

| | |
|---|---|
| **Class** | DataReader<br>Read the historical traffic data |
| **Methods** | public int[][] readRoute()<br>read network route<br><br>public double[][] readWeight()<br>read historical traffic data<br><br>public double[] readStaticWeight()<br>read static traffic data<br><br>public String[][] readRoad()<br>read the nodes data and its properties<br><br>public String[][] readName()<br>read the String name for each nodes |

### 6.9.2 Class: RoutingGUI

| | |
|---|---|
| **Class** | RoutingGUI<br>Interface of DIANa |
| **Methods** | private void initComponents(ActionEvent evt)<br>initialization of GUI interface<br><br>private void  routeButton(ActionEvent evt)<br>shortest path calculation<br><br>private void addIncidentButton(ActionEvent evt)<br>blocks part of network<br><br>private void removeIncidentButton(ActionEvent evt)<br>remove the selected incident from the list<br><br>private void resetButton(ActionEvent evt)<br>rest all to default |

### 6.9.3 Class: RoutingEngine

| | |
|---|---|
| **Class** | RoutingEngine<br>Dynamic and static routing algorithm |
| **Methods** | RoutingEngine()<br>Constructor<br><br>public void findShortest()<br>Methods to find the shortest path. Will loop until all nodes are in settled set.<br><br>public Node getMin()<br>get nodes with minimum travel time from unsettled set.<br><br>public boolean inSet(int x)<br>Check if node x is in settled set<br><br>public boolean inUnset(int y)<br>Check if node y is in unsettled set<br><br>public Node getNode(int z, Vector p)<br>get the Node object with number attributes z from Vector p.<br><br>public void relaxDynamic(Node ne)<br>relax neighbouring nodes of node ne<br><br>public void relaxStatic(Node ne)<br>relax neighbouring nodes of node ne<br><br>public void findPath()<br>find the shortest path from the shortest tree node.<br><br>public Vector getRespath()<br>get the resulting resulting shortest path. |

### 6.9.4 Class: NodeResult

| **Class** | NodeResult |
|---|---|
| **Attributes** | private int index<br>node index<br><br>private String name<br>node name<br><br>private double cost, scost, dcost, length<br>node static/dynamic cost, and length<br><br>private double time, stime, dtime<br>node travel time from departure node<br><br>private int xcoor<br>x-coordinate<br><br>private int ycoor<br>y-coordinate |

### 6.9.5 Class: Time

| **Class** | Time |
|---|---|
| **Methods** | public void setHours(int ho)<br>set the hours<br><br>public int getHours()<br>return the hours<br><br>public void setMinutes(int mi)<br>set minutes<br><br>public int getMinutes()<br>return minutes<br><br>public void toMin(int hrs, int min)<br>convert the given time to minutes<br><br>public double getConvert()<br>return the conversion from time to minutes |

### 6.9.6 Class: Node

| **Class** | Node |
|---|---|
| **Methods** | public Node(int n)<br>Constructor<br><br>public void setPred(int p)<br>set the predecessor node as p<br><br>public int getPred()<br>return the index of predecessor node<br><br>public void setCost(double c)<br>set cost to reach the node as c<br><br>public double getCost()<br>return the cost to reach node<br><br>public void setSucc(int succ)<br>set node successor as succ<br><br>public Vector getSucc()<br>get node successor<br><br>public void setTime(double t)<br>set the time to reach node as t<br><br>getTime()<br>get the time to reach node t<br><br>public int getNode()<br>get the index of node |

## 6.10 Detail Methods Description

In order to create DIANa, there are several steps involved and this section will focus on the most important methods of DIANa in details and which necessary steps to be done to ensure that the system will function properly.

### 6.10.1 findShortest() Method of RoutingEngine.class

There are two types of shortest path algorithm, dynamic (mode 2) and static (mode 1). For each mode there is a loop which checks whether not the unsettled set is empty. If it is not empty then it will get the node with minimum traveling time from unsettled set, put that node in settled set, then relax its neighbors.

```
public void findShortest(){
   …
   if(mode == 2){
      while(!unset.isEmpty()){
         u = getMin();
         set.addElement(u);
         relaxDynamic(u);
      }
   }
   else if(mode ==1){
      while(!unset.isEmpty()){
         u = getMin();
         set.addElement(u);
         relaxStatic(u);
      }
   }
   …
}
```

**Figure 39: Method findShortest() pseudo-code**

### 6.10.2 getMin() Method of RoutingEngine.class

Return node object from unsettled set with the lowest traveling time.

```
public Node getMin(){
    double c = (unset.elementAt(0)).getCost();
    …
    for(int i=1; i<unset.size();i++){
        ctemp=(unset.elementAt(i)).getCost();
        if(ctemp < c){
            c = ctemp;
            cmin = i;
        }
        …
        Node setnode = (Node)unset.elementAt(cmin);
        unset.removeElementAt(cmin);
        return setnode
}
```

**Figure 40: Method getMin() pseudo-code**

### 6.10.3 relaxDynamic () Method of RoutingEngine.class

Node relaxation algorithm is part of the method that will relax its neighboring nodes and calculate its traveling time. To get the correct traveling time between two nodes, we need two variables - the predecessor node and the time.

The algorithm will then search for road number, which have a connection between the two nodes and it will use this number as the row number of our historical data and convert the time to column number.

```
public void relaxDynamic(Node ne){
    for(int i = 0;i<graph.length;i++){
        if(graph[i][0] == ne.getNode()){
            if(!inSet(graph[i][1]) && !inUnset(graph[i][1])){
                …
                ne.setSucc(adj);
                double tmp = (double)(Math.round(ne.getTime()));
                double tmp2 = tmp/10;
                int j = (int)(Math.round(tmp2));
                …
                adj.setCost(((ne.getCost()+weight[i][j])));
                adj.setPred(ne.getNode());
                adj.setTime((ne.getTime()+weight[i][j]));
                unset.addElement(adj);
            }
            else if(!inSet(graph[i][1]) && inUnset(graph[i][1])){
                Node adj = getNode(graph[i][1], unset);
                double tmp = (double)(Math.round(ne.getTime()));
                double tmp2 = tmp/10;
                int j = (int)(Math.round(tmp2));
                double we = weight[i][j];
                …
                if(adj.getCost() > ne.getCost() + weight[i][j]){
                    adj.setCost(((ne.getCost()+weight[i][j])));
                    adj.setPred(ne.getNode());
                    adj.setTime(ne.getTime()+weight[i][j]);
                }
            }
        }
        …
    }
}
```

**Figure 41: Method relaxDynamic() pseudo-code**

#### 6.10.3.1 Detail example

From: Knp_Holendrecht
To: Vinkeveen
Time: 18:07
Our historical data is a network matrix data of 808 x 144 as can be seen below.

**First step**, the algorithm will look for index of road connection between Knp_Holendrecht_2 and Vinkeveen, which is a record with index number *27*.

**Second step**, it will convert the time to column number in the following manner:

1. Convert time to minutes

$$18:07 \rightarrow (18 \times 60) + 7 = 1087$$

**Equation 8: Conversion to minutes**

2. Divide the conversion by 10 to match the numeric dimension of our historical data.

$$\frac{1087}{10} = 108.7$$

**Equation 9: Conversion to numeric time dimension of DIANa**

3. Round the result to integer

$$Round(108.7) = 109$$

**Equation 10: Rounding to integer**

The integer number *109* is the index number of the column in our historical traffic database.

**Finally** we look in our historical traffic database at row number 27 and column number 109 for the traveling time between the two nodes which is 7.00 minutes

**Table 12: Lookup on traffic data**

| | | | … | 107 | 108 | 109 | 110 | 111 | … |
|---|---|---|---|---|---|---|---|---|---|
| ID | From | To | … | 17:50 | 18:00 | 18:10 | 18:20 | 18:30 | … |
| … | … | … | … | … | … | … | … | … | … |
| 27 | Knp_Holendrecht_2 | Vinkeveen | … | 9.00 | 8.00 | 7.00 | 6.00 | 5.00 | … |
| | | | | | | | | | |

808

144

### 6.10.3.2 relaxStatic() method of RoutingEngine.class

Node relaxation method for static routing algorithm.

```
public void relaxStatic(Node ne){
    for(int i = 0;i<graph.length;i++){
        if(graph[i][0] == ne.getNode()){
            if(!inSet(graph[i][1]) && !inUnset(graph[i][1])){
                Node adj = new Node(graph[i][1]);
                        ne.setSucc(adj);
                ne.setSuccW(sweight[i]);
                adj.setCost(((ne.getCost()+sweight[i])));
                adj.setPred(ne.getNode());
                unset.addElement(adj);
            }

            else if(!inSet(graph[i][1]) && inUnset(graph[i][1])){
                Node adj = getNode(graph[i][1], unset);
                if(adj.getCost() > ne.getCost() + sweight[i]){
                    adj.setCost(((ne.getCost()+sweight[i])));
                    adj.setPred(ne.getNode());
                }
            }
        }
        …
    }
}
```

**Figure 42: Method relaxStatic() pseudo-code**

#### 6.10.3.3 inSet() method of RoutingEngine.class

Search the settled set for a particular node, if the node is present then it will return true otherwise false.

```
inset() method of RoutingEngine.class
//check element is settled nodes
public boolean inSet(int x){
  …
  while(i<set.size()){
    if((set.elementAt(i)).getNode() == comp){
      val = true;
    }
    …
  }
  return val;
}
```

**Figure 43: Method inSet() pseudo-code**

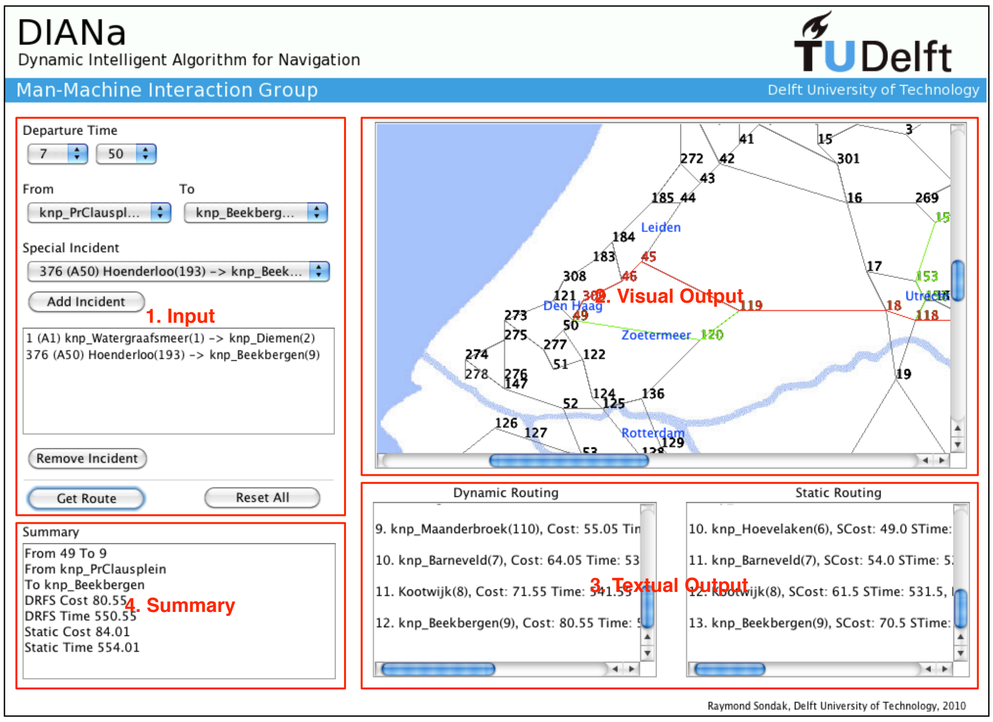
#### 6.10.3.4 inUnset() method of RoutingEngine.class

Search the unsettled set for a particular node, if the node is present then it will return true otherwise false.

```
//check elements in unsettled node
public boolean inUnset(int y){
  …
  while(i<unset.size()){
    if((unset.elementAt(i)).getNode() == comp){
      val = true;
    }
    …
  }
    return val;
}
```

**Figure 44: Method inUnset() pseudo-code**

## 6.11 GUI Design



Departure Time: 7 50

From: knp_PrClauspl…

To: knp_Beekberg…

Special Incident: 376 (A50) Hoenderloo(193) -> knp_Beek…

```
1 (A1) knp_Watergraafsmeer(1) -> knp_Diemen(2)
376 (A50) Hoenderloo(193) -> knp_Beekbergen(9)
```

Summary

```
From 49 To 9
From knp_PrClausplein
To knp_Beekbergen
DRFS Cost 80.55
DRFS Time 550.55
Static Cost 84.01
Static Time 554.01
```

Dynamic Routing

```
9. knp_Maanderbroek(110), Cost: 55.05 Tin
10. knp_Barneveld(7), Cost: 64.05 Time: 53
11. Kootwijk(8), Cost: 71.55 Time: 541.55
12. knp_Beekbergen(9), Cost: 80.55 Time:
```

Static Routing

```
10. knp_Hoevelaken(6), SCost: 49.0 STime:
11. knp_Barneveld(7), SCost: 54.0 STime: 5
12. Kootwijk(8), SCost: 61.5 STime: 531.5,
13. knp_Beekbergen(9), SCost: 70.5 STime:
```

Figure 45: DIANa Graphical User Interface

The Graphical User Interface contains of 4 parts:

1. **Input**
   - *Departure time*
     A departure time can be entered from the drop down menu
   - *Departure/Arrival point*
     User can select one out of 274 departure and arrival points
   - *Special Incident*
     User can select roads that need to be avoided because of accidents or special incidents.
   - *Get Route*
     Execute DIANa to calculate the shortest route based on user input
   - *Reset All*
     Reset all user input

2. **Visual Output**
   The fastest route by using dynamic- and static data is shown on a map with the intersection ID as identification. The fastest route by using historical traffic data shown as a green line and fastest route by using static data is shown as a red line.

3. **Textual output**
   The fastest route by using dynamic- and static data is shown as a textual output with the intersection name and intersection ID as identification.

4. **Summary**
   Summary of the advantages of fastest route by using historical traffic data compare with static data.

Chapter 7

# Experiment and Results

*Preview*

This chapter focuses on the actual experiment of the research. An experimental study is performed to evaluate our system. We will describe those experiments and their results.

## 7.1 Experiment

### Release 1: Routing using static data and Dijkstra's algorithm

$$TravellingTime_{A_n \to A_{n+1}} = \frac{Distance_{A_n \to A_{n+1}}}{MaxSpeed_{A_n \to A_{n+1}}}$$

Equation 11: Routing using static data

### Release 2: Routing using dynamic data and dynamic algorithm

$$TravellingTime_{A_n \to A_{n+1}} = HistoricalData_{A_n \to A_{n+1}}$$

Equation 12: Routing using historical data

## 7.2 Hypothesis Formulation

Our assumption is that DIANa will provide a route with a shorter traveling time than static road routing, however if there is no better route than its static road routing counterparts, DIANa will provide the same route as the static road routing with the same total traveling time. Hence, the successful result and experiment should not deviate from this assumption.

The hypothesis to be tested is the following:

H1: *In the presence of congestion, dynamic routing should result in a shorter traveling time or if there is no better route the same traveling time compared to the static routing.*

## 7.3 Computational Consideration

There are two computational considerations need to be taken into account for this hypothesis:

- Inspecting graph of travel time
- Compute variance of prediction

We show that by using historical data, dynamic routing helps if we have reliable prediction of traveling time along the road segments.

- Historical data are used to read current and predicted traveling time.

## 7.4 Sample Selection

In order to be able to perform our experiment, we need to define departure city, arrival city, and departure time. Our expectation is that during the rush hours with high traffic streams between big cities, dynamic routing using historical traffic data should provide a better solution than static routing.

Table 13: City selection based on total number of population

| | **City** | **Province** | **Populations** |
|---|---|---|---|
| **1** | Amsterdam | Noord-Holland | 739.290 |
| **2** | Rotterdam | Zuid-Holland | 539.650 |
| **3** | Den Haag | Zuid-Holland | 473.940 |
| **4** | Utrecht | Utrecht | 255.200 |
| **5** | Eindhoven | Noord-Brabant | 209.700 |
| **6** | Tilburg | Noord-Brabant | 182.150 |
| **7** | Almere | Flevoland | 180.920 |
| **8** | Groningen | Groningen | 166.120 |
| **9** | Nijmegen | Gelderland | 151.030 |
| **10** | Haarlem | Noord-Holland | 146.960 |
| **11** | Arnhem | Gelderland | 140.780 |
| **12** | Breda | Noord-Brabant | 139.810 |
| **13** | Apeldoorn | Gelderland | 136.510 |
| **14** | Enschede | Overijssel | 132.560 |
| **15** | Amersfoort | Utrecht | 126.750 |
| **16** | Zoetermeer | Zuid-Holland | 120.000 |
| **17** | Dordrecht | Zuid-Holland | 118.540 |
| **18** | Leiden | Zuid-Holland | 117.480 |
| **19** | Maastricht | Limburg | 116.200 |
| **20** | Zwolle | Overijssel | 112.250 |
| **21** | ‘s-Hertogenbosch | Noord-Brabant | 102.220 |

### 7.4.1 City Selection

To perform our experiments, we select a number of departure and arrival cities. To ensure reliability of our samples, we have selected cities in The Netherlands with number of populations of more than 100.000. The list of cities with number of populations of more than 100.000 as per 1 January 2007 can be found in table 13 [16].

### 7.4.2 Departure Time Selection

In order to get a significant result for the experiment, we select departure time that has the most traffic congestion. The most congestion usually happens in the morning and in the evening when people goes to work and come back home on working days. We decided to take the evening rush time between 15.20 until 18.20 with 20 minutes interval as can be seen in figure 46 below.

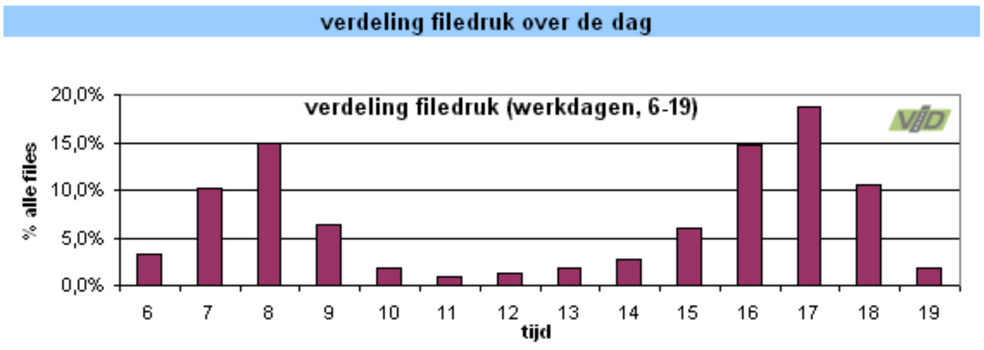


Figure 46: Traffic congestion statistic

## 7.5 Methodology

In order to show that Dynamic Route Finding using Historical Data can perform better than Static Route Finding System we will do the following:

1. Calculate the travel time of every combination of departure cities as mentioned in table 13 and departure time from 15.20 until 18.20 with 20 minutes interval. The result will be in Matrix form of traveling time

$$D(t) := \left(d(t)_{i,j}\right)_{21\times 21}$$

Equation 13: Dynamic traveling time matrix of 21 cities

With $t$ is the departure time and $i,j \in \{1,2,\ldots,21\}$ are the departure and arrival cities.

2. We compute the shortest path by using Dijkstra's algorithm and calculate the total traveling time using historical traveling data as we mentioned previously in section

6.2.1.2 *Static Routing with Historical Travel Time* for the combination of all cities in table 13 and departure time as in step 1.

$$SD(t) := \left(sd(t)_{i,j}\right)_{21\times 21}$$

Equation 14: Static traveling time between 21 cities using traffic data

With $t$ is the departure time and $i, j \in \{1,2,\ldots,21\}$ are the departure and arrival cities.

3. We calculate the performance of our Dynamic Route Finding System by taking the difference between 2 and 1

$$P(t) := \left(p(t)_{i,j}\right)_{21\times 21}$$

Equation 15: DIANa performance matrix between 21 cities

With

$$P(t) := SD(t)_{i,j} - D(t)_{i,j}$$

Equation 16: Performance gain at time t

4. Result of $P(t) := \left(p(t)_{i,j}\right)_{21\times 21}$
will be plotted in matrix form of color to help us visualize the result more easily.

With time sample

$$t \in \{t_0, t_1, \ldots, t_9\}$$

Equation 17: Time sample

and

$$t \in \{15:20, 15:40, 16:00, 16:20, 16:40, 17:00, 17:20, 17:40, 18:00, 18:20\}$$

Equation 18: Elements of time sample

The color matrix shows the difference between DIANa and static road routing on congested road.

*The more significant the travel time difference between the dynamic and the static road routing on a congested road, the more we can leverage on the functionalities of DIANa to be used to calculate the fastest route (Figure 47).*

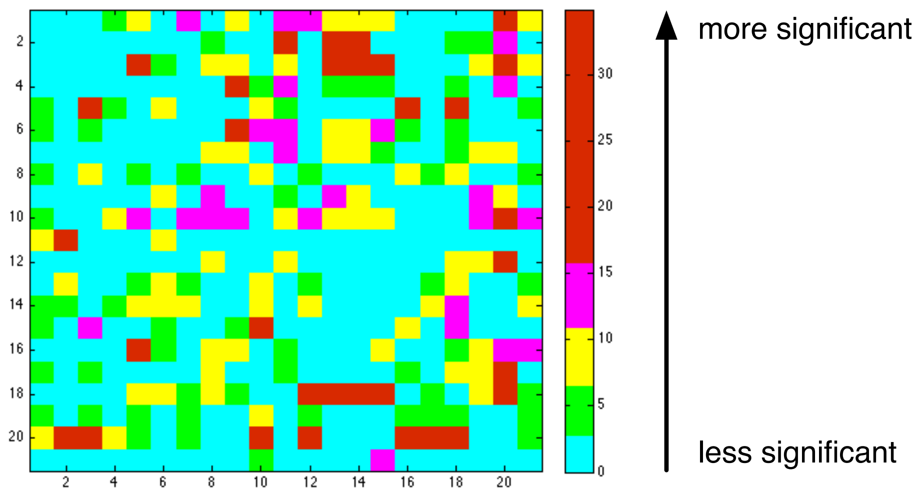


**Figure 47: Color matrix of the difference between DIANa and static road routing using historical traffic data – More is better.**

## 7.6 Results

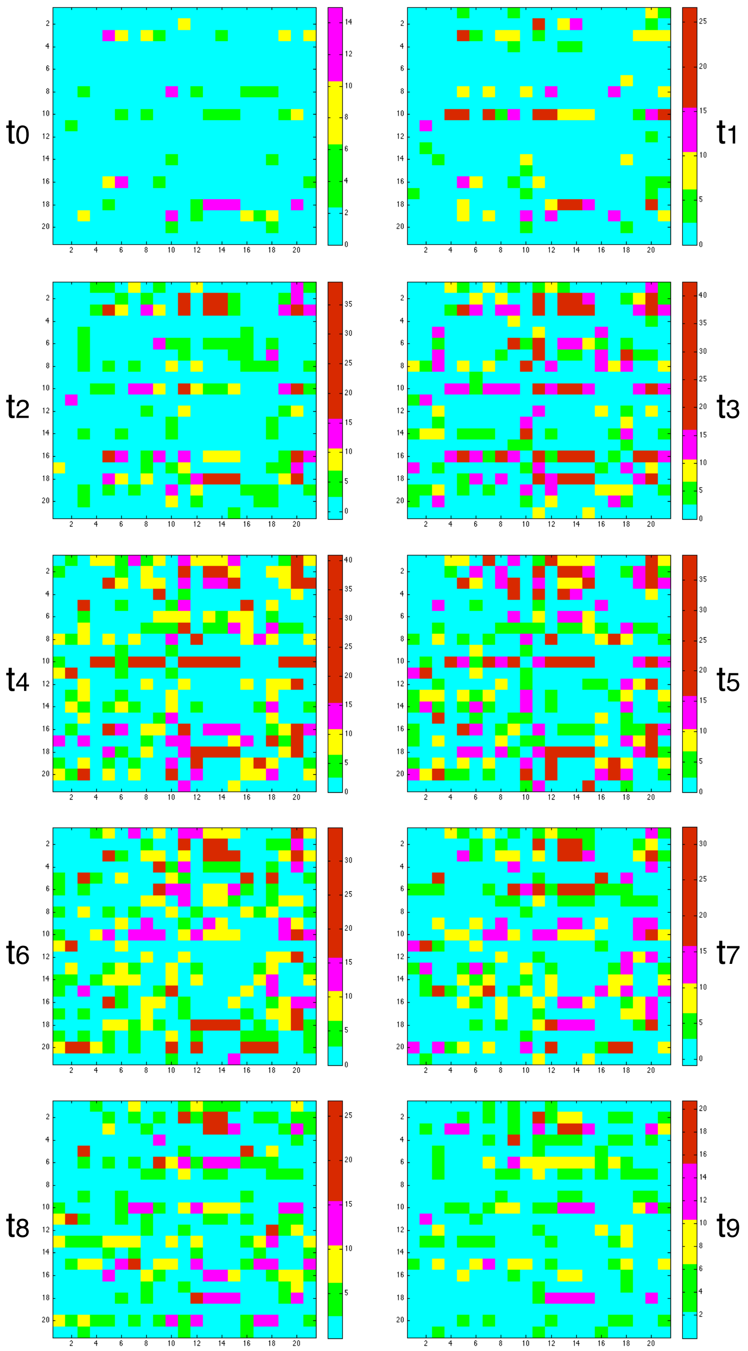


**Figure 48: Performance comparison(Appendix B)**

There are 10 matrices depicted in this section. Each matrix represents the performance of Dynamic Intelligent Algorithm for Navigation at a different point in time as mentioned in section 7.4 (between 15:20 and 18:20 at 20 minutes interval).

Matrix $t_1$ is the result of DIANa at 15:20. It can be seen that blue is the dominant color, which means that the difference between dynamic and static routing is marginal or non-existent. This is due to the fact that around this time congestion in highway is also minimal. Thus, at most routes, there is no difference in dynamic and static. This is as expected since in the absence of congestion, both dynamic and static routing should provide the fastest route or path with the least distance and travel time. This reasoning is also valid for Matrix $t_2$ and $t_3$.

Matrix $t_4$ to Matrix $t_8$ is the result of DIANa in between 16:20 and 17:40. These 5 matrices contain the most yellow, pink, and red color, which means that the difference between dynamic and static routing is considerable. The peak of traffic in highway occurs during these hours (please refer to figure 46). This shows that DIANa performs well during traffic time; it provides alternative routes with shorter travel time compared to the static routing. Certain nodes even show more than 30 minutes time difference between dynamic and static routing (indicated with red color).

Matrix $t_9$ and Matrix $t_{10}$ is the result of DIANa at 18:00 and 18:20. Matrix $t_9$ and $t_{10}$, as expected, showing more blue and green color compared to matrix $t_4$ to $t_8$ considering that traffic congestion at these hours is also less than traffic congestion in between 16:20 to 17:40.

## 7.7 Conclusions

Overall, we can conclude that Dynamic Intelligent Algorithm for Navigation performs as expected. The experiment is done by comparing the difference of dynamic route finding using historical data and static routing at different point in time reflecting different traffic condition. The result is as expected in which the difference between dynamic and static in the absence of congestion is marginal or even non-existent. In this situation, both routing alternatives should provide the shortest path and travel time and there should not be any different routes from the two systems. On the other hand, when there is congestion, it is expected the Dynamic Routing Finding System should provide an alternative route at the shorter travel time compared to the static route finding. The only exception is if there is no such alternative route at shorter travel time. From the result matrices, it can be seen that *with dynamic route finding system, the average time saving can be around 13-17 minutes, with some cases even above 30 minutes*. Thus, dynamic routing finding system is a *viable solution to congestion*. As mention in section 1.2, traffic congestion is not a physical manifestation but also depends on individual decision making, so it is utterly important to equip individual with the right tool in order to help them making the right decision that will result in shortening their travel time and at the same time reducing congestion.

Chapter 8

# Data Validation

*Preview*

Valid and reliable data is the basis and essential for a successful research. We may take different or even wrong conclusions without valid data. This chapter will show how we measure and test the data validity and reliability used for the database.

## 8.1 Data Reliability and Consistency

The purpose of this section is to evaluate the reliability and internal consistency of the data used for our database. As previously mentioned, our data consists of four different random Thursday's traffic data and derives from a ready to use source. Reliability and consistency test are often neglected, however, without reliable and consistent data, the outcome and analysis of this thesis will also be doubtful. The following subsections will deal with the tests we perform to ensure data reliability and validity as well as the result thereof.

### 8.1.1 Sample Selection

The sample used to test data reliability and consistencies are the same sample of city selection as used in section 7 as well as the departure time selection. This is to ensure that the result of this test could be generalized to the data we use in our database and will also ensure internal consistency.

### 8.1.2 Methodology

1. Calculating the travel time from every combination of departure cities as mentioned in table 13 and departure time from 15.20 until 18.20 with 20 minutes interval. The result will be in Matrix form of traveling time.

$$D(t) := \left(d(t)_{i,j}\right)_{21\times 21}$$

**Equation 19: DIANa traveling time matrix between 21 cities**

With $t$ is the departure time and $i,j \in \{1,2,...,21\}$ are the departure and arrival cities.

2. Calculate dynamic routing using the traffic data of each Thursday traffic data.

$$TH(t,a) := \left(th(t,a)_{i,j}\right)_{21\times 21}$$

**Equation 20: DIANa traveling time matrix between 21 cities using traffic data**

With the following variables:

$$t \in \{15:20,15:40,16:00,16:20,16:40,17:00,17:20,17:40,18:00,18:20\}$$

**Equation 21: Sample time range**

$t$ is the departure time and

$$a \in \{1,2,3,4\}$$

**Equation 22: Traffic data**

and $a$ is the Thursday traffic data.

3. Calculate the difference between Dynamic Routing using our System and Dynamic Routing using the traffic data of every Thursday.

$$R(t,a) := \left(r(t,a)_{i,j}\right)_{21\times 21} = D(t) - TH(t,a)$$

**Equation 23: Performance difference between DIANa and each traffic data**

With $t$ is the departure time and $a \in \{1,2,3,4\}$ are the departure and arrival cities.

4. Visualize the resulting difference $R(t,a)$

With sample time

$$t \in \{t_0, t_1, \ldots, t_{10}\}$$

**Equation 24: Time sample**

and

$$t \in \{15:20,15:40,16:00,16:20,16:40,17:00,17:20,17:40,18:00,18:20\}$$

**Equation 25: Elements of time sample**

The color matrix shows the difference on DIANa using historical traffic data and individual traffic dataset. *The less significant the difference between the historical traffic data and the individual traffic dataset, the more reliable and robust the historical dataset we used for DIANa*.

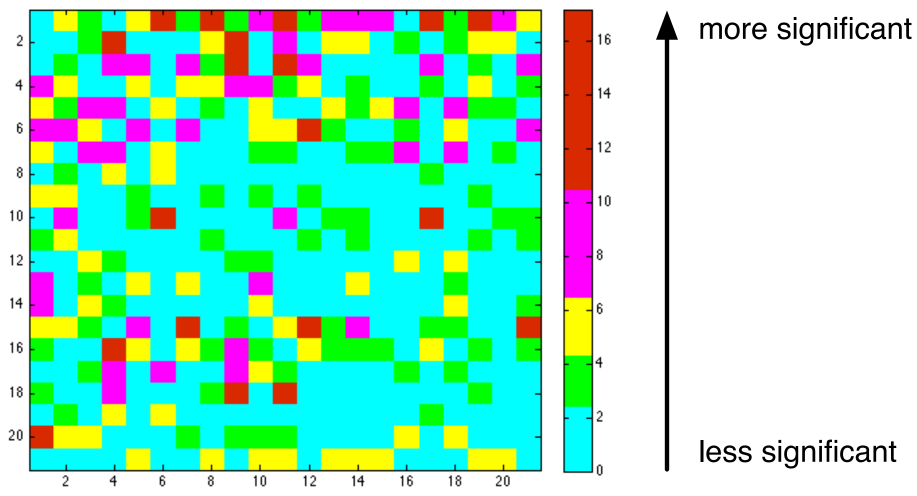


**Figure 49: Color matrix of the difference between historical traffic data and individual traffic dataset – Less is better.**

### 8.1.3 Results

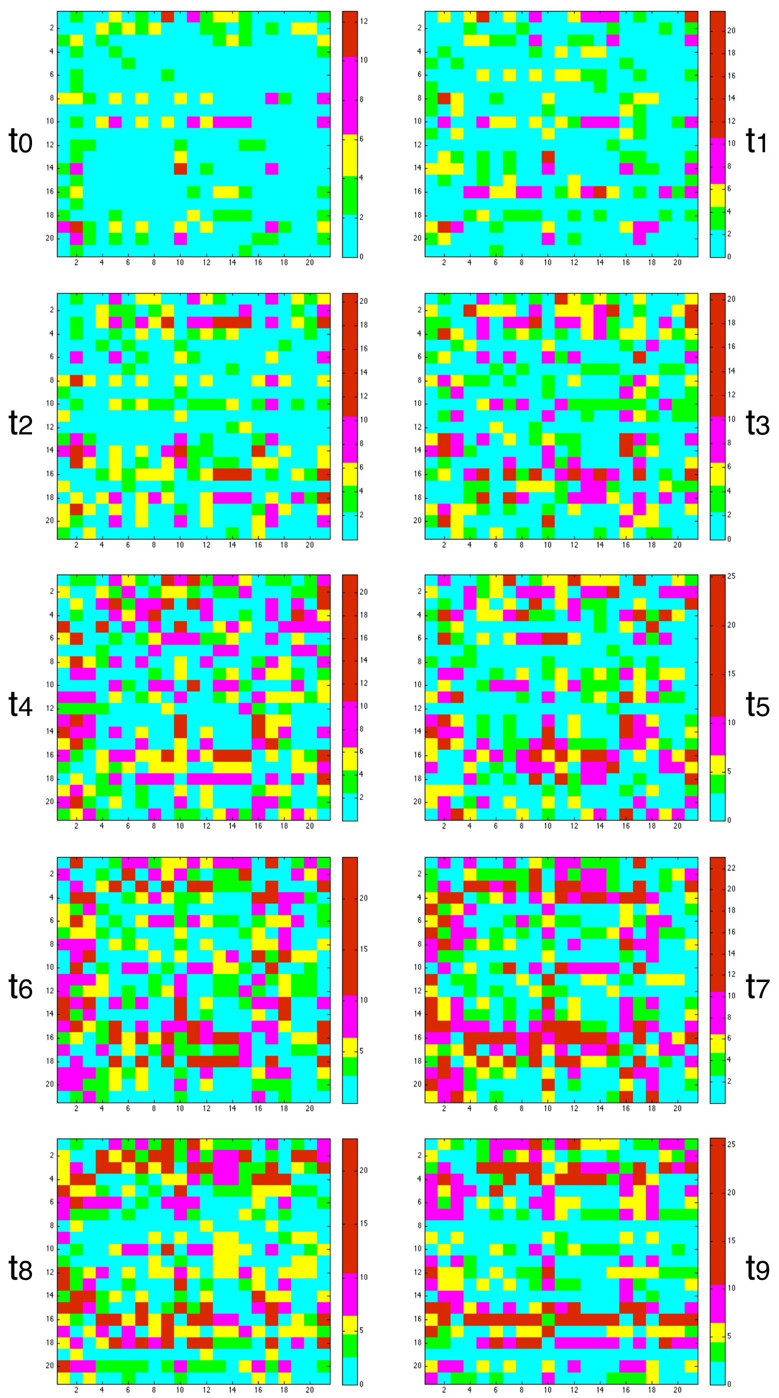


Figure 50: Result of first dataset(Appendix C)

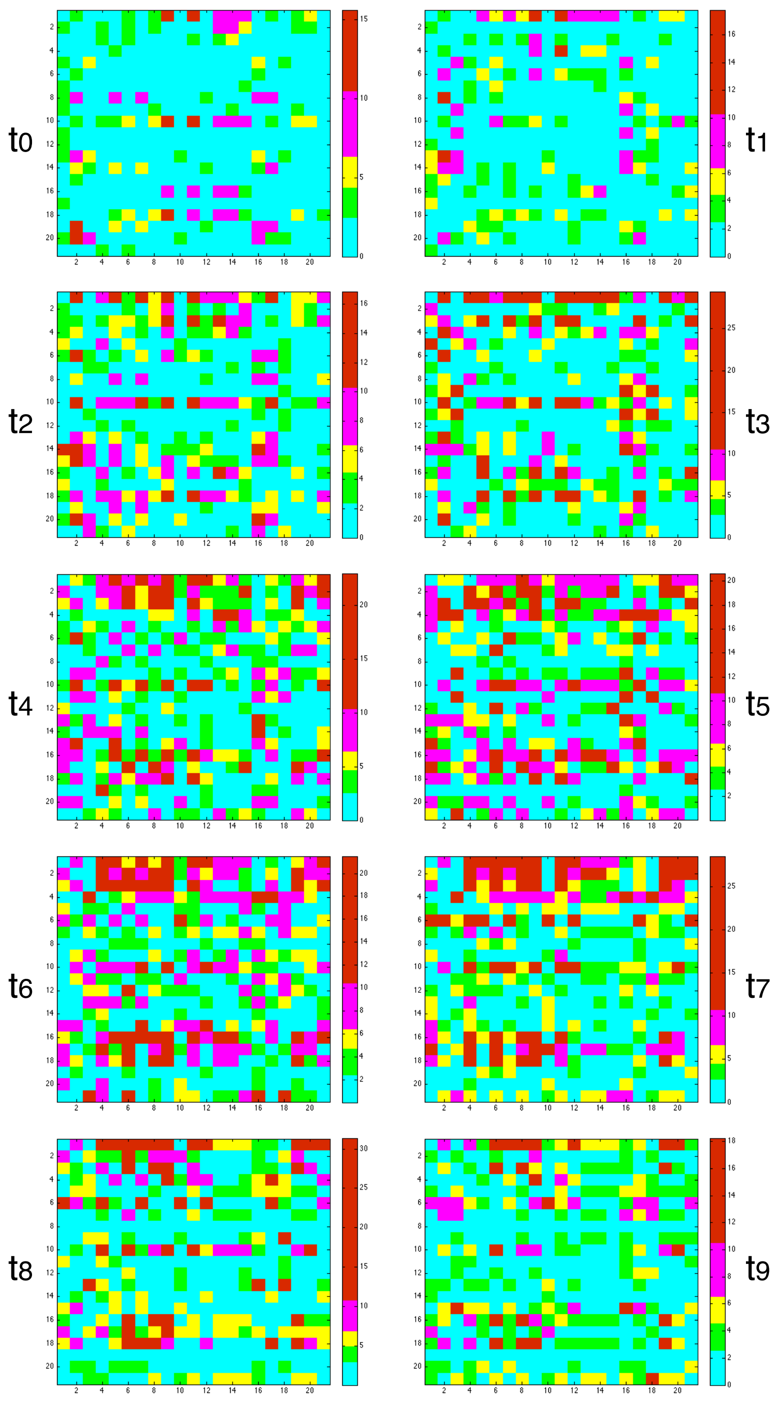


Figure 51: Result of second dataset(Appendix D)

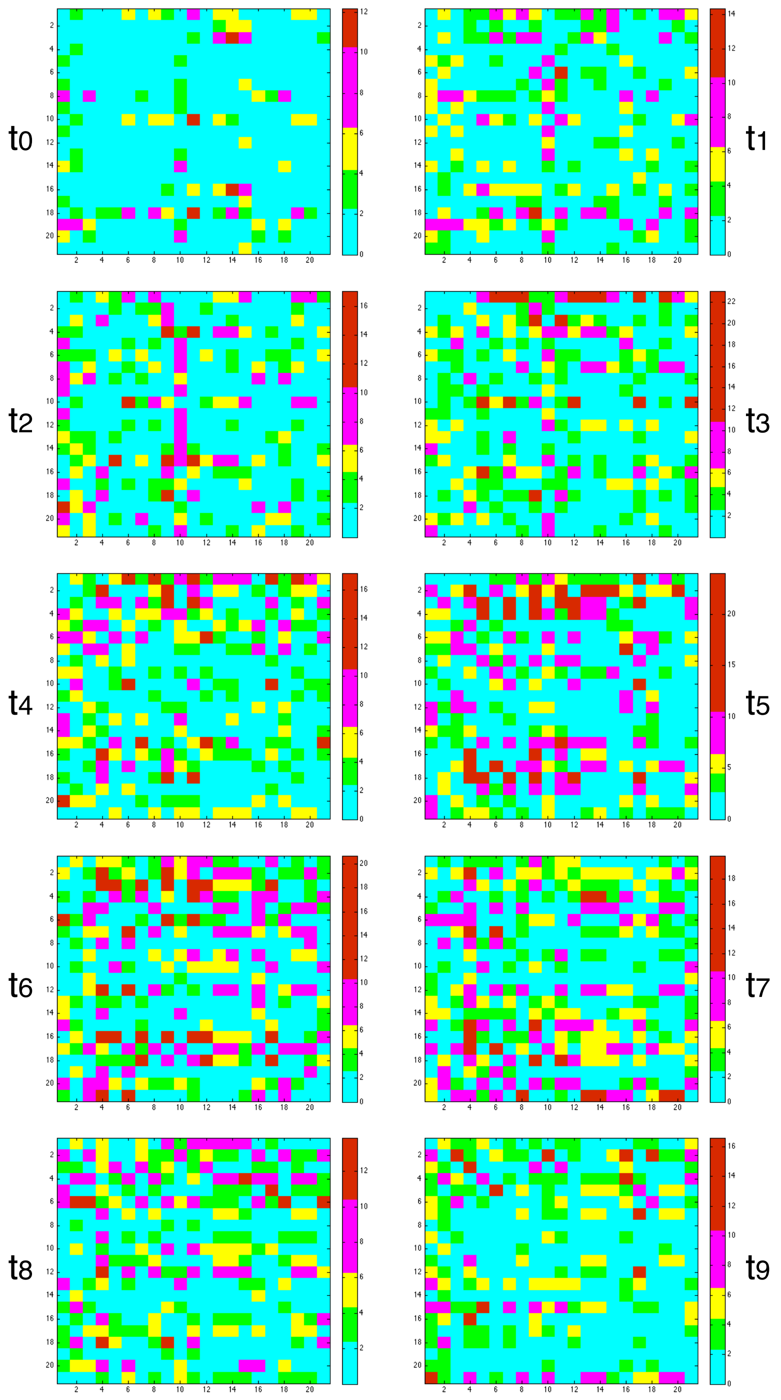


**Figure 52: Result of third dataset(Appendix E)**

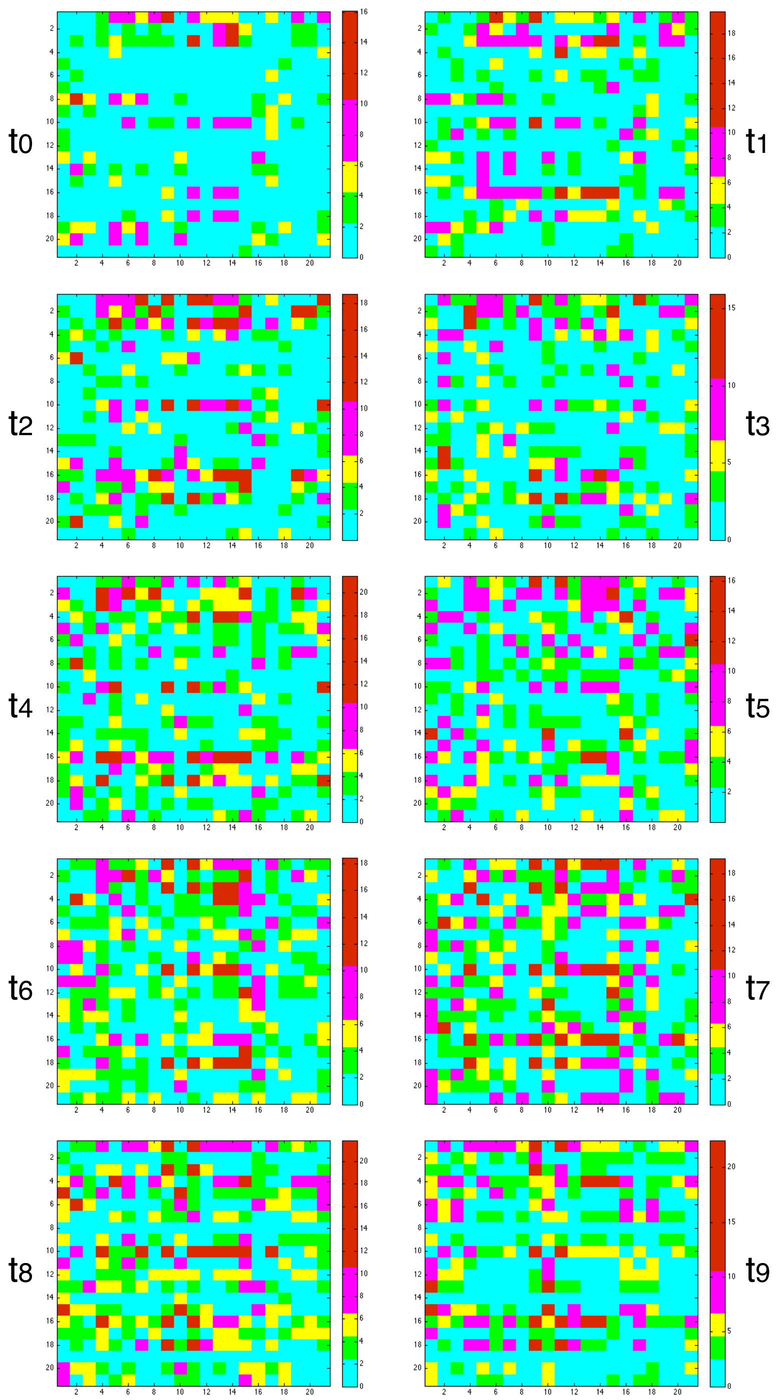


Figure 53: Result of fourth dataset(Appendix F)

The visualized matrices as depicted in this section is the result of difference of average dynamic routing data of the four datasets with dynamic routing data of each dataset at different point in time.

The lesser the difference between the average data and each of the datasets data means the more reliable and consistent the data used for this experiment. However, it needs to be taken into account that the method of data gathering for this purpose derived from an already existing data from ANWB Netherlands with no direct control of the situation of that particular day. It could mean that uncontrollable variables like incident or weather condition could influence the result. As mentioned before, actually these kinds of variables have been excluded, however, the after effect is inarguably still presence. For example, after an accident has been cleared up, it would take some time to normalize the traffic. In a perfect world, the data should only contain normal day-to-day congestion factors, which means that only controllable variables are taken into account.

As can be seen from most matrices, there is quite considerable blue and green color in each matrix, which indicate marginal or small time differences. Interestingly, as can be shown from the figures, at the hours when the congestion is minimal (between 15:20 to 16:00), the difference is smaller than those at peak hours. This might be explained that during low congestion period, the data in between the four datasets should be fairly stable unless there is particular incident or event that happened at certain highways. On the other hand, during the peak period, more variance might happen due to things like length of traffic that might be different at different time on different day, also when there is incidents, then the effect during these hours would be much larger than during low period.

In general, we are quite satisfied with the reliability and consistency test, taking into account of data availability and the fact that we need to create from ready-to-use to construct the data for this experiment. As mentioned before, although it seems like there are some considerable differences present in the matrices, we can confidently say that there is enough blue and green area that shows the difference between the four datasets are marginal and by averaging them, the data credibility and accuracy should improve and hence the data used for this experiment is reliable and consistent.

## 8.2 Data Stability and Robustness

From the section 8.1, we can conclude that the data used in this thesis is reliable and valid, which means we can be sure that the conclusion of this thesis is also reliable and valid. However, valid and reliable data is not enough, we also need to test whether the data used is stable and robust. This section will deal with testing the data stability. We will test at which level the maximum percentage of error DIANa can handle and still function properly. The following subsections explain in details the methodology of the test and the result.

### 8.2.1 Sample Selection

This section deals with the steps and approaches used to arrive at consistent and reliable sample used in the research. Sample selection is important since this is the basis to create a usable and valid model.

#### 8.2.1.1 City Selection

To test our data for stability and robustness, we choose routes that have the most traffic congestion where we expect maximum differences between dynamic routing and static routing. According to "online Dutch library" infonu.nl [17]. The most traffic congestion in 2007 happens with the following roads (Table 14).

**Table 14: Top 5 - Most congested road in evening rush hour**

| | Road | Traffic congestion direction |
|---|---|---|
| **1** | A27 | Noordeloos richting Breda |
| **2** | A2 | Hedel richting 's-Hertogenbosch |
| **3** | A50 | Rijnbrug bij Renkum richting Oss |
| **4** | A1 | Knooppunt Diemen richting Amersfoort |
| **5** | A12 | Knooppunt Lunetten richting Arnhem |

For the experiment we also need define departure and arrival points that pass through the most traffic congestion Table 15.

**Table 15: Route chosen for analysis**

| | From(node number) | To(node number) | Route Cost(no traffic) |
|---|---|---|---|
| **1** | Knp. Eemnes(5) | Knp. Galder(142) | 49 Minutes |
| **2** | Knp. Oudenrijn(18) | Goirle(199) | 43 Minutes |
| **3** | Knp. Waterberg(106) | Knp.Hintham(22) | 34 Minutes |
| **4** | Knp. Coenplein(87) | Knp. Beekbergen(9) | 50 Minutes |
| **5** | Knp. Pr. Clausplein(49) | Knp. Velperbroek(105) | 61 Minutes |
| **6** | Knp. Pr. Clausplein(49) | Knp. Beekbergen(9) | 60 Minutes |

#### 8.2.1.2 Time Selection

To ensure our test reliability we choose the peak of evening rush hour, which occurs around 17.50 as departure time.

### 8.2.2 Methodology

The following method is used to test our data stability and robustness.

1. Calculate dynamic routing of table 15 using historical traffic data $H$ with departure time at 17.50.
2. Calculate dynamic routing of table 15 using historical traffic data $H$ with addition

of noise $Q$ and use the parameter $e$ as the error percentage.

We generate noise for each road $h(r)$ by using $rand()$ function, which returns random numbers whose elements are uniformly distributed in the interval $(0,1)$.

Our intention is to generate a random number whose numbers are uniformly distributed between $(-MIN(h(r)), MIN(h(r)))$, we choose $MIN(h(r))$ as the limit because we do not want that our noise generator to generate numbers greater than the minimum value of the road, because we would like to prevent the noisy road to result in negative numbers.

The following is the noise function that generates a random numbers whose elements are uniformly distributed in the interval $(-MIN(h(r)), MIN(h(r)))$ with $\mu = 0$.

$$Q(h(r)) = MIN(h(r)) \times ((2 \times rand()) - 1)$$

**Equation 26: Random noise generator**

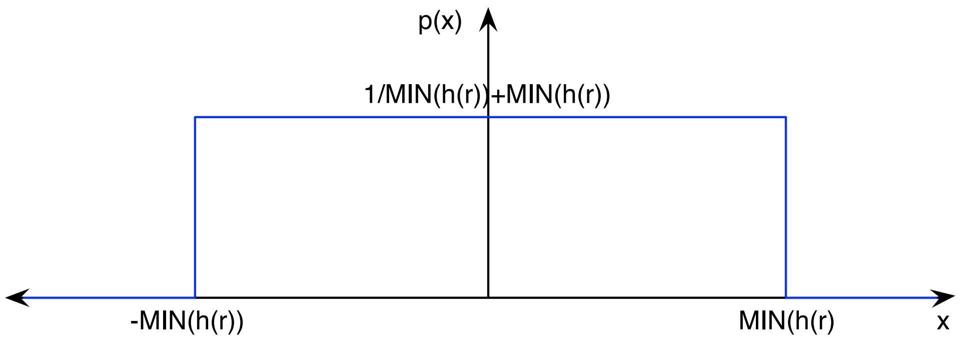


**Figure 54: Uniform Distribution in the interval (-MIN(h(r)), MIN(h(r)))**

To calculate a noisy data we need to add our historical traffic data with random generated noise, but before we add the randomly generated noise we first multiply the noise with error percentage $e$ so that we can easily incorporate the amount of noise to the data to be able to test the stability and robustness of our historical traffic data.

$$N(h(r), e) = h(r) + (e \times Q(h(r)))$$

**Equation 27: Noisy road calculation**

$$N(h(r), e) = h(r) + (e \times (MIN(h(r)) \times 2) \times (rand() - 0.5))$$

**Equation 28: Noisy road calculation**

With $r$ as road traffic data and $e$ as the error percentage.

$$r \in \{1,2,\ldots,808\}$$

Equation 29: Number of roads

$$e \in \{0\%,\ldots,100\%\}$$

Equation 30: Error percentage

Example:
To calculate the first noisy road of our historical data for road number 1 ($r = 1$), we do the following.

$$N(h(1),e) = h(1) + \left(e \times \left(MIN(h(1)) \times 2\right) \times \left(rand() - 0.5\right)\right)$$

Equation 31: Noisy road 1 calculation

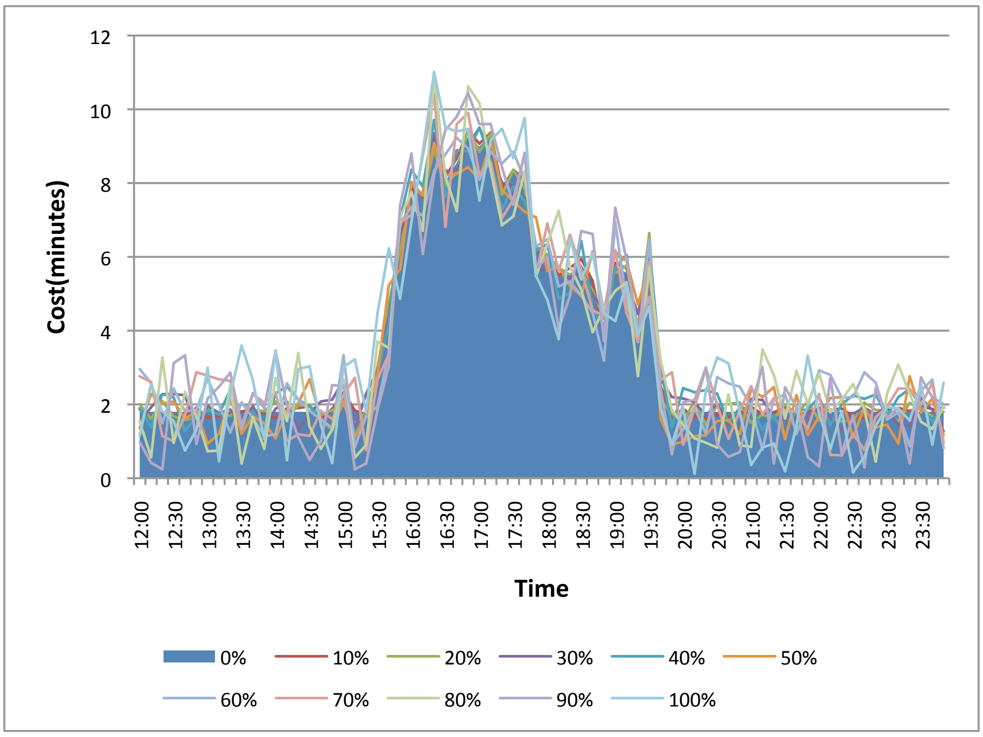


Figure 55: Road r=1 from 12:00 - 23:50 with error percentage 0% - 100%

3. The final step is to make a comparison between the resulting route by using our historical traffic data and historical noisy data.

If the noisy data resulting the exact same route as the historical traffic data, then it means that our historical traffic data is stable for some error percentage $e$.

## 8.3 Results

There are two tests done to check the data stability and robustness. This is an important test to show that the data gathered and used here are valid and can be used for the purpose of this research. The first table showed that data is test with percentage error between 10%-100%. Green indicates that the result is valid and as expected whereas red indicates that data is unstable.

It can be seen from the first table that it shows much red colors in the 20% range although it still showing a lot of green up to 30%. In order to be more accurate, second test is done and with smaller step of error percentage between 10%-20%. Here can be seen clearly that data remain stable up to 14% and the limitation stays on the 15%.

**Table 16: Experiment with error percentage between 10% to 100%**

| | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 |
| 2 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 |
| 3 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟩 |
| 4 | 🟩 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 5 | 🟩 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 6 | 🟩 | 🟥 | 🟩 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |

**Table 17: Experiment with error percentage between 10% to 20%**

| | 11% | 12% | 13% | 14% | 15% | 16% | 17% | 18% | 19% | 20% |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 2 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 |
| 3 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 4 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 |
| 5 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 |
| 6 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 |

In general, 15% limitation is very satisfying considering the method of data gathering and data collection. This is also representative of the most congested highways in Netherlands and in the peak hour. We then can conclude that data used for this research in stable and robust.

## 8.4 Conclusion

This chapter deals with testing on data reliability and validity as well as data stability and robustness and we can confidently said that this thesis passes all tests and therefore the result of this thesis should be considered as valid and reliable.

The result from section 8.2 has shown that in general, we are quite satisfied with the reliability and consistency test, taking into account of data availability and the fact that we need to create from ready-to-use information to construct the data for this experiment. There are some considerable differences present in the result presented, however the difference between the 4 Thursdays are marginal and by averaging them, the data credibility and accuracy should improve and hence we have enough confident to state that the data used for this experiment is reliable and consistent.

Also from section 8.3, we can conclude that in general, the result of the stability and robustness test was within the expected boundaries/limitations. The sample used for this test is also representative of the most congested highways in Netherlands and in the peak hour, which then brings us to believe that the data used for this research in indeed stable and robust.

# Chapter 9

# Conclusion and Recommendation

## 9.1 Extended Summary

### 9.1.1 Study Literature, Research Objectives, and Problem Definition

#### 9.1.1.1 Research Background

During the study literature, we learn that traffic congestion has caused billions and billions of dollars including loss of productivity in our everyday life. Many institutions, governmental organizations, and academics have tried to come up with many initiatives and solutions to minimize congestion.

In general, two different factors can be distinguished in dealing with congestion, external and internal factors.  External factors are related to forces outside the influence of individual's motive and behavior, for example government regulation. External factors are aimed to a broader approach in minimizing congestion and will only produce result in a long term.

The second factor introduced in this thesis is internal factors, which are directly related and can be influenced by individual's decision making and behavior, which is more pertinent and the focus of this thesis and our dynamic routing solution.

A very important concept that has become the basis of this thesis is that *congestion is not only a manifestation of physical phenomenon but as well as a result of individual trip decision making and driving behavior*, which is not always rational and can be related either to *sentiment and emotional decisions* or to the *lack of knowledge*, which lead to the lack of capability in making rational decisions. Addressing particularly this lack of knowledge, it is *utterly important to equip individual with the right tool in order to help them making the right decision that will result in shortening their travel time and at the same time reducing congestion*, for example a navigation system that is equipped with capability to recalculate and reroute journey to minimize travel delay.

#### 9.1.1.2 Research Objectives and Problem Definition

As mentioned previously, we need a system that is capable to provide an alternative route in case of congestion to minimize travel delay. However, currently such system is based only on real time data, which means that by the time the user receive the traffic information, it is already too late to use the alternative route provided.

Therefore, this thesis is meant to find a solution that can be integrated into an existing device, using a routing algorithm that can forecast traffic congestion and finding the fastest route to get to the end point before the actual congestion so that the user could make a decision beforehand depending on the time and route he/she is planning to take. The forecast will be based on the historical traffic information deriving on the notion that most traffic congestions happen almost at the same time during the day and the same hours daily. We use this historical real traffic data to create an effective Dynamic Intelligent Algorithm for Navigation (DIANa) to avoid congestion or other road blockage.

### 9.1.2 Dynamic Routing Algorithm

The algorithm used in this thesis is basically similar to the Dijkstra's Algorithm. However, instead of looking from two-dimensional static data, our dynamic routing algorithm incorporates time as the third dimension. By taking time into account for the calculation of shortest route between starting point and destination point we can gain the optimal performance in case of congestion compared to shortest route algorithm that only use two-dimensional data.

### 9.1.3 Data Collection

The model in this thesis is based on the real data collected from historical traffic information from ANWB, one of the most reliable traffic sources in the Netherlands. This traffic data provides us with information on which highways and roads that have delays or congestions, at which time, for how long, and for how many kilometers.

The main steps during this data collection is first of all running a script and scheduled it to pick up data every 2 minutes from a sample of four random Thursday resulting in 170 traffic information data per day. Secondly, we then entered the traffic data into a database and calculating traffic delay between road junctions with 10 minutes interval. Thereafter, we did linear interpolation to deal with missing values and then averaging the data from the sample taken to increase accuracy.

### 9.1.4 Define Historical Traffic Database, Design a Dynamic Routing System, and Implementation

Once we have determined the algorithm and data model, we then created the high-level conceptual design base. This stage involved building the system architecture, modeling the traffic data, and designing the database itself. During this process, we found out that with a well-defined dynamic routing algorithm, we can easily incorporate the historical traffic database we collected as the source data. Further, it is also easy to update the historical data in the future when needed.

Based on this high-level conceptual design, we created a proof of concept, DIANa, based on the Dijkstra's algorithm and the data model we created before. This proof of concept is based on the notion that DIANa should help alleviate traffic congestion from individual perspective and to show that using a dynamic routing would indeed have a significant impact on the traveling time compared to static routing. This stage involved the design of the implemented algorithm, how to make sure the output shows the intended results (differences in time between static and dynamic routing), the requirements, the use case, and the diagram.

### 9.1.5 Testing and Analysis

Once we have the algorithm and database at hand, we then conducted an experiment to see whether our dynamic routing system works and indeed create a significant difference is time compared with static routing when there is congestion and alternative routes to avoid the congestion are available. This experiment is based on selected cities in The Netherlands with number of populations of more than 100.000 and done by comparing the difference of dynamic route finding using historical data and static routing at different point in time reflecting different traffic condition. On the other hand, when there is congestion, as expected, our Dynamic Intelligent Algorithm for Navigation could provide an alternative route at a shorter travel time compared to the static route finding.

We plotted the result of the experiment in color coded matrices and we could see that *with dynamic route finding system, the average time saving can be around 13-17 minutes, with some cases even above 30 minutes*. Thus, based on this, we could conclude that dynamic routing finding system is a *viable solution to congestion*.

We went one step further to ensure that our experiment and our database is reliable, hence the result presented in this thesis is relevant. We performed a test to the reliability and validity of the data and the algorithm and we could conclude that in general, the result of the stability and robustness test was within the expected boundaries/limitations. The sample used for this test is also representative of the most congested highways in Netherlands and in the peak hour, which then brings us to believe that the data used for this research in indeed stable and robust and therefore we could be confident that DIANa would work.

## 9.2 General Conclusions

This thesis is based on the idea that people tend to look for the fastest way to get from one place to the other and one of the most common ways is by using an automotive navigation system or go to one of the many websites who can give you the fastest route to reach your destination. This is called static routing. The real issue with this method is that we will not be able to avoid congestion until we get in the middle of traffic jam and by that time it is already too late to take an alternative route.

This is where our research on dynamic routing plays a role. We should be able to choose an alternative route even before we start our journey based on traffic prediction based on historical data on the assumption that traffic flows and congestions contain a certain pattern. Our idea is that dynamic routing should be able to provide an optimal result when there is traffic congestion. The data model should provide an input to this system and serve as a basis to determine the fastest route or journey time, so that an individual using this system could decide, hopefully rationally based on the outcome provided, whether to stay with the shortest route given (static routing) or the alternative route with fastest route based on the historical information (dynamic routing).

Deriving from this concept, we created DIANa, a dynamic intelligent algorithm for navigation that will provide and alternative route based on historical information. In this thesis, the author presents a proof of concept that hopefully can be used and developed in the future for further implementation.

After a few tests and analysis, DIANa is proved to be working. There is indeed a significant difference in time by using static routing or DIANa when there is congestion. As we have said before, traffic congestion is not a physical manifestation but also depends on individual decision making, so it is utterly important to equip individual with the right tool and we hope that DIANa is the answer to this need.

## 9.3 Research Limitations and Challenges

One of the challenges present with creating our intelligent system is to have the right algorithm. The idea behind our dynamic routing algorithm is to use the historical data of the road networks and incorporate these data to find the fastest route based on departure time. Our goal is to have a "correct" shortest route and this algorithm is basically the same as Dijkstra's algorithm, except for how it calculates the journey time between two nodes after relaxing its neighbors. Instead of looking from two-dimensional static data, our dynamic routing algorithm would be able to read the travel time between nodes from three-dimensional data time from the database for that particular time with time being the third dimension.

Another challenge in our research is creating the historical database itself with reliable data. We have decided to base our data on collection of historical information of traffic condition on a certain day and we have picked any four random Thursdays within a month since Thursday is the most expected day with congestion in the week. We then collected the traffic information from those Thursdays from ANWB that gives information on highways that have delays or congestion - how many kilometers and how long the delay is. The database is constructed by noting the delays, which are then inputted in the database to calculate the extra time needed to travel from one point to another. This process has taken most of the time as this is one of the critical steps in our research and producing a reliable database is crucial.

Although this thesis evidently present the superiority of dynamic routing compared to static routing and when integrated with current existing navigation system undoubtedly would help reduced congestion at hand of individuals, this research is not without its flaws. One of the limitations in this research is the data used to construct the historical database. Although reliable and valid, this is still a “ready-to-use” data and not a tailor made data specifically used for this research.

One important thing to note is that this research is meant as a proof of concept for further research. Proof of concept is a short and/or incomplete realization of a certain method or ideas to demonstrate its feasibility, or a demonstration in principle, whose purpose is to verify that some concept or theory is probably capable of exploitation in a useful manner. This research should be considered as a milestone on the way to a fully functioning prototype. We hope this thesis will pave the way to further research in incorporating this DIANa algorithm into a real life tools that would help alleviate traffic congestion as opposed to the current static navigation system available in the market.

# Bibliography

[1] J I Daniel and K Bekka, "The Environmental Impact of Highway Congestion Pricing," *Journal of Urban Economics*, Jan 2000. [Online]. http://linkinghub.elsevier.com/retrieve/pii/S0094119099921356

[2] A C McKinnon, "The Impact of Traffic Congestion on Logistical Efficiency," *sml.hw.ac.uk*, Jan 1998. [Online]. http://www.sml.hw.ac.uk/logistics/pdf/ioltsum.pdf

[3] CambridgeSystematics, "Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestio Mitigation," pp. 1-140, Sep 2005.

[4] C R Lindsey and E T Verhoef, "Traffic Congestion and Congestion Pricing," *dare.ubvu.vu.nl*, Jan 2000. [Online]. https://dare.ubvu.vu.nl/handle/1871/9420

[5] (2008, Jan.) Wikipedia, the free encylopedia. [Online]. http://en.wikipedia.org/wiki/Global_Positioning_System

[6] S Russel and P Norvig, "Artificial Intelligence - A Modern Approach," pp. 1-1112, May 2006.

[7] D B Johnson, "A Note on Dijkstra's Shortest Path Algorithm," *Journal of the ACM (JACM)*, Jan 1973. [Online]. http://portal.acm.org/citation.cfm?id=321768

[8] E B Zhan, "Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures," *Journal of Geographic Information and Decision Analysis*, Jan 1997. [Online]. http://teams.gemstone.umd.edu/classof2009/fastr/Zahn%20Article.pdf

[9] B V Cherkassy, A V Goldberg, and T Radzik, "Shortest paths algorithms: Theory and experimental evaluation," *Mathematical Programming*, Jan 1996. [Online]. http://www.springerlink.com/index/H764865329543501.pdf

[10] H Dibowski, "Hierarchical Routing System using Ant Based Control," p. 93, Aug 2003.

[11] G Eggenkamp, "Dynamic Multi Modal Route Planning," p. 123, Jun 2001.

[12] T N Lam and C O Tong, "A Dynamic Route Guidance System Based on Historical and Current Traffic Pattern," *Intelligent Vehicles' 92 Symposium*. [Online]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=252287

[13] P W Eklund, S Kirkby, and S Pollitt, "A Dynamic Multi-source Dijkstra's Algorithm for Vehicle Routing," *Intelligent Information Systems*, Jan 1996. [Online]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=573976

[14] ANWB. (2008) ANWB verkeersinformatie. [Online]. http://www.anwb.nl/verkeer

[15] A Guide to the Edsger W. Dijkstra Papers, 1948-2002. [Online]. http://www.lib.utexas.edu/taro/utcah/00378/cah-00378.html#a0

[16] (2008, Jan.) Wikipedia, De vrije encyclopedie. [Online]. http://nl.wikipedia.org/wiki/Lijst_van_grote_Nederlandse_steden

[17] (2008, Jan.) InfoNu.nl. [Online]. http://auto-en-vervoer.infonu.nl/verkeer/17326-file-top-10-knelpunten-ochtend-avondspits.html

[18] Wikipedia. (2009) Wikipedia, the free encyclopedia. [Online]. http://en.wikipedia.org/wiki/Edsger_Dijkstra

[19] Edsger Wybe Dijkstra. (2009) University of Texas. [Online]. http://www.lib.utexas.edu/taro/utcah/00378/cah-00378.html#a0

[20] (2008, Jan.) National Institute of Standards and Technology. [Online]. http://www.nist.gov/dads/HTML/directedGraph.html

[21] GISWiki. (2009) GIS | Dijkstra's algorithm. [Online]. http://en.giswiki.net/wiki/Dijkstra's_algorithm

[22] M Wachs, "Fighting Traffic Congestion with Information Technology," *uctc.net*, 2002. [Online]. http://www.uctc.net/papers/556.pdf

[23] E T Verhoef, "Time, speeds, flows and densities in static models of road traffic congestion and congestion pricing," *Regional Science and Urban Economics*, Jan 1999. [Online]. http://linkinghub.elsevier.com/retrieve/pii/S0166046298000325

[24] J L Träff and C D Zaroliagis, "A Simple Parallel Algorithm for the Single-Source Shortest Path Problem on Planar Digraphs," *Irregular*. [Online]. http://www.springerlink.com/index/57643706m2128725.pdf

[25] B Storey and R Holtom, "The use of historic GPS data in transport and traffic monitoring.," *Traffic Engineering+ Control*, Jan 2003. [Online]. http://www.trafficchannel.com/pdf/congest_mon.pdf

[26] R Raman, "Recent results on the single-source shortest paths problem," *ACM SIGACT News*, Jan 1997. [Online]. http://portal.acm.org/citation.cfm?id=261352

[27] I Parry, "Comparing the efficiency of alternative policies for reducing traffic congestion," *Journal of Public Economics*, Jan 2002. [Online]. http://linkinghub.elsevier.com/retrieve/pii/S0047272700001638

[28] S Ishak and H Al-Deek, "Performance Evaluation of Short-Term Time-Series Traffic Prediction Model," *Journal of Transportation Engineering*, Jan 2002. [Online]. http://link.aip.org/link/?JTPEDI/128/490/1

[29] R V Helgason, J L Kennington, and B D Stewart, "The one-to-one shortest-path problem: An empirical analysis with the two-tree Dijkstra algorithm," *Computational Optimization and Applications*, Jan 1993. [Online]. http://www.springerlink.com/index/V2M9P7648T814878.pdf

[30] G Fleck and R Mertens, "Method and system for determining dynamic traffic information," *US Patent 6,012,012*, Jan 2000. [Online]. http://www.google.com/patents?hl=en&lr=&ie=UTF-8&vid=USPAT6012012&id=hgABAAAAEBAJ&oi=fnd&dq=historical+road+traffic+vehicle+congestion+historical

[31] A Creak, "Edsger W. Dijkstra," *ACM SIGPLAN Notices*, vol. 37, no. 12, Dec 2002. [Online]. http://portal.acm.org/citation.cfm?id=636517.636521

[32] M Ben-Akiva et al., "Real-time Prediction Of Traffic Congestion," *Vehicle Navigation and Information Systems*, Jan 1992. [Online]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=640246

[33] Renaud Waldura. (2007) Renaud Waldura. [Online]. http://renaud.waldura.com/doc/java/dijkstra/

[34] (2008, Jan.) Maximum snelheden. [Online]. http://www.maximumsnelheden.info

# Appendix A

# Dijkstra's Algorithm

*Preview*

This chapter focuses on Dijkstra's algorithm - Why we need to understand the algorithm and what the importance of Dijkstra's algorithm in this research. This chapter will answer most of these questions since Dijkstra's algorithm is the essential basis for developing our dynamic route system.

## A.1 Introduction

Edsger Dijkstra (May 11, 1930 – August 6, 2002) was a Dutch computer scientist and one of the most influential member of founding generation. He received many awards during his life as a scientist including A. M. Turing Award, he also made a number of remarkable contributions to computer science according to [18][19], one of his most famous contribution is shortest path algorithm which is named after him, known as Dijkstra's algorithm, which is the essence of many other variant of shortest path algorithm.

Dijkstra's algorithm is a greedy search algorithm which always find the optimal solution that solves single-source shortest path problem (SSSP) on a weighted directed graph provided all edges are non-negative. This algorithm is probably the best-known shortest path algorithm and also most implemented because of its simplicity but yet very efficient. It will construct an optimal shortest path tree from the source to all destinations with respect to cost, which is known as the single-source shortest paths problem (SSSP).

## A.2 The Algorithm

This section will further explain the Dijkstra algorithm, its definition and how it works.

### A.2.1 Graph Definitions

The shortest path problem consist of directed graph $G$ which is an ordered pair that is $G := (V, E)$ subject of the following conditions:

- $V$ is a set whose elements are called vertices or nodes;
- $E$ is a set of ordered pairs of distinct vertices that is a subset of $V \times V$, called directed edges, arcs, or arrows.

$e = (x,y)$ is a directed arc is the direction of the edge being from $x$ to $y$, with the following property:

- $y$ is called the head of the edge;
- $x$ is called the tail of the edge;
- $y$ is said to be a direct successor of $x$;
- $x$ is said to be a direct predecessor of $y$;
- if a path leads from $x$ to $y$, then $y$ is said to be a successor of $x$, and $x$ is said to be a predecessor of $y$;
- The arc $(y,x)$ is called the arc $(x,y)$ inverted.

**Formal Definition:** A graph $G$ is a pair $(V,E)$, where $V$ is a set of vertices, and $E$ is a set of edges between the vertices $E \subseteq \{(u,v) \mid u,v \in V\}$. If the graph does not allow *self-loops*, adjacency is *irreflexive*, that is $E \subseteq \{(u,v) \mid u,v \in V \wedge u \neq v\}$ [20].

A weighted directed graph also has a cost or weight function $w$ for each edge $e$.

$$w : E \rightarrow \{r \in R \mid r \geq 0\} \cup \{\infty\}$$

Equation 32: Cost function for weighted graph

Equation above is a function of the cost or weight of edge $e \in E$ in the graph, and represent the lack of each edge by an edge with cost $\infty$.

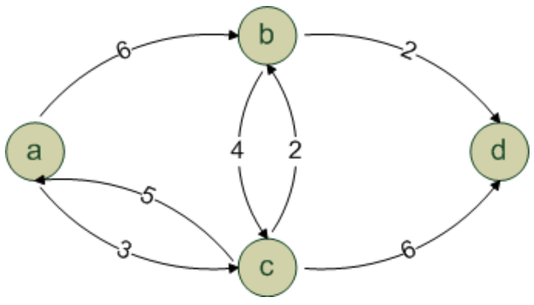


Figure 56: Directed graph

### A.2.2 How Dijkstra's Algorithm Works

The input of the algorithm consists of a weighted directed graph $G := (V,E)$ and a source vertex $s \in V$. We will denote $V$ the set of all vertices, $E$ the set of all nodes with each edge as an ordered pair of vertices in graph $G$, and $w$ the non-negative cost of the edge. The algorithm works by keeping for every vertex $v$ the cost $d(v)$ of the shortest path found so far between $s$ and $v$.

Dijkstra's algorithm partitions vertices in two distinct sets, the set of unsettled vertices $Q$ and the set of settled vertices $S$. Unsettled set $Q$ is the to-be-examined vertices and settled set $S$ are already-examined vertices. For every vertex, if a vertex is considered settled by the algorithm, it will be move from the unsettled set $Q$ to settled set $S$. The algorithm begin by adding source vertex $s \in V$ in unsettled set $Q$ and will keep running as long as there are still vertices that need to be examined and ends if all vertices are in the settled set $S$ .

The key of the algorithm is the vertex relaxation method: if there is an edge from vertex $u$ to vertex $v$, then the shortest known path from $s$ to $u$, $d(u)$ can be extended to a path from $s$ to $v$ by adding the cost of edge $[u,v]$ at the end. This path will have length $d(u)+[u,v]$. If this is less than the current $d(v)$, we can replace the current value of $d(v)$ with the new value.

After each relaxation procedure the algorithm will add the neighboring vertex or vertices of vertex $u$ to the unsettled set $Q$ and move vertex $u$ to settled set $S$. The vertex in the unsettled set $Q$ that will be relaxed by the algorithm is the vertex that has the lowest cost from the source vertex $s$. [21].

### A.2.3 Pseudo Code

The following variables will be used in order to explain the algorithm:

| **Variables** | **Description** |
|---|---|
| $G$ | Weighted directed graph |
| $s$ | Source vertex |
| $d(v)$ | Stores the best estimate lowest cost from source $s$ to vertex $v$ |
| $\pi(v)$ | Stores the predecessor of each vertex on the shortest path from the source |
| $S$ | Set of settled vertices, the vertices whose lowest cost from the source has been computed |
| $Q$ | Set of unsettled vertices, the vertices whose cost from the source has to be computed |

1. function Dijkstra(G, s):

```
function Dijkstra(G, s)
        d := infinity // initialize d to infinity, π, S, and Q to empty
        S := empty set
        Q := empty set

        Q := Q union {s} //first step add s to Q and d(s) to 0
        d(s) := 0
        π(s) := s
        while Q is not an empty set                //the algorithm
                u := extract_minimum(Q)
                S := S union {u}                   //add u to S
                relax-neighbors(u)
```

Figure 57: function Dijkstra(G, s)

2. function relax-neighbors(u):

```
function relax-neighbors(u)
    for each vertex v adjacent to u
        if v not in set S & v not in set Q
            d(v) := d(u) + [u,v]       //calculate the cost
            π(v) := u         // set previous node of v as u
            Q := Q union {v}        //add v to Q
        if v not in set S & v in set Q
            if d(v) > d(u) + [u,v]      // a lower cost exists
                d(v) = d(u) + [u,v] //recalculate the cost
                π(v) = u      //set previous node of v as u
```

Figure 58: function relax-neighbors(u)

3. function extract-minimum(Q):

```
function extract-minimum(Q)
        find  vertex v with lowest cost d[v] from the source in Q
        remove v from Q and return v
```

Figure 59: function extract-minimum(Q)

### A.2.4 Flow diagram of Dijkstra's algorithm

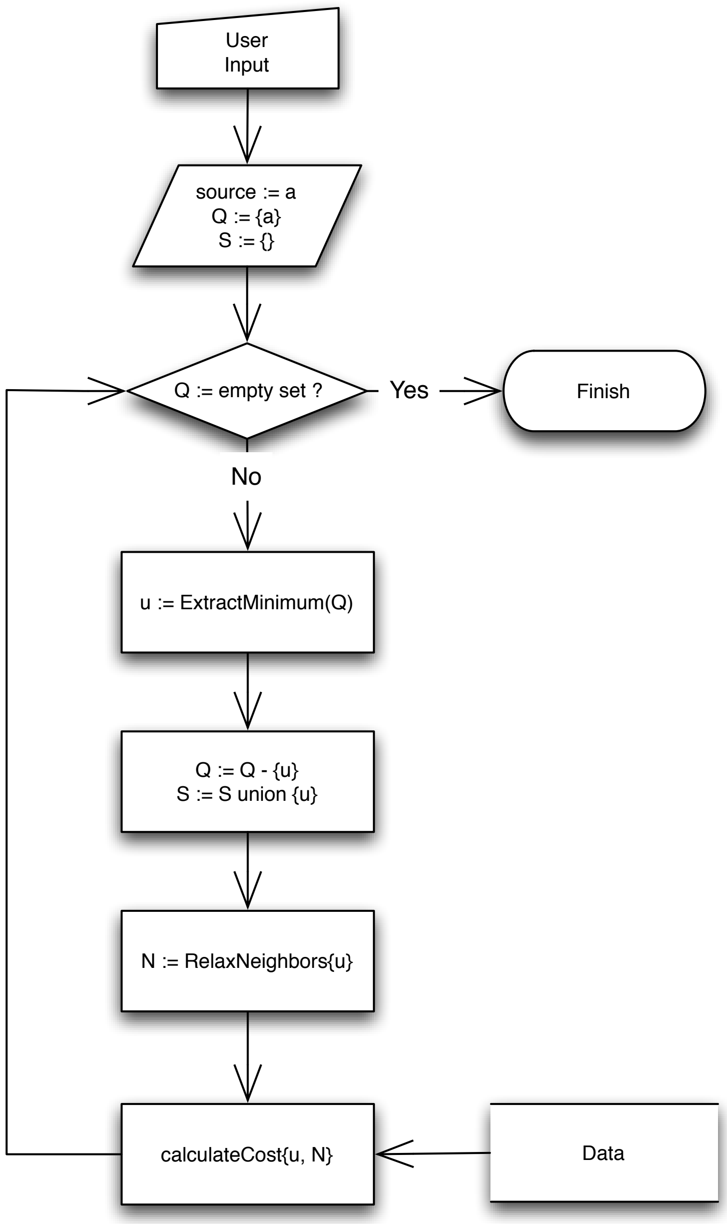


Figure 60: Flow diagram Dijkstra algorithm

### A.2.5 Detail Example: Step by step operation of Dijkstra's algorithm

In this section we will give a detail example of how the algorithm exactly works. We will start off by taking an example of a simple weighted directed graph $G := (V, E)$ that consists of 4 vertices and 7 directed edges which links vertices together. The edges of weighted directed graph has a weight, or numeric positive value associated with each edge. The weight between vertex $u$ and $v$ can be noted by $[u,v]$.

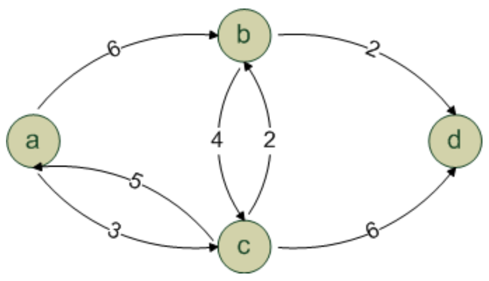


Figure 61: Directed graph

The Dijkstra's algorithm will search the shortest path from the source vertex, which is in this case vertex a to every other vertex in the graph. The initial value of unsettled set $Q$ and settled set $S$ are:

$Q := \{\ \}$ and $S := \{\ \}$

**Step 1:**

Given initial weighted directed graph $G := (V, E)$. The algorithm will start by adding source vertex $a$ to the unsettled set $Q$:

$$Q := Q \cup \{a\}$$

Equation 33: Unsettled set Q

calculate the cost to reach vertex $a$ from the source vertex:

$$d(a) = 0$$

Equation 34: Minimum cost to reach vertex a

and assign previous vertex of vertex $a$ as a:

$$\pi(a) = a$$

Equation 35: Previous vertex of a

After this step the value of unsettled set $Q$ and settled set $S$ will be:

$$Q := \{a(0,a)\}$$

$$S := \{\ \}$$

Equation 36: Unsettled and settled vertices

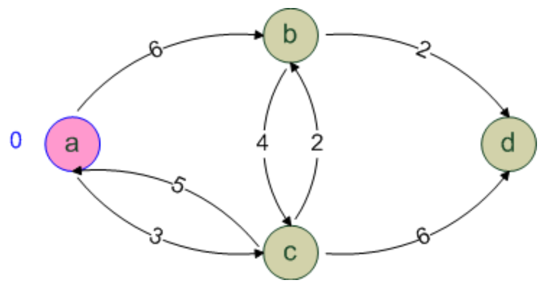


Figure 62: Directed graph after step 1

**Step 2:**
After the first step, unsettled set $Q$ is not empty, then the algorithm will extract a vertex in unsettled set $Q$ that have a minimum cost from source vertex $a$, which is again vertex $a$:

$$extractMinimum(Q) = a$$

Equation 37: Function extractMinimum(Q)

Remove vertex $a$ from unsettled set $Q$:

$$Q := Q - \{a\}$$

Equation 38: Remove vertex a from Q

Then add vertex $a$ to settled set $S$:

$$S := S \cup \{a\}$$

Equation 39: Add vertex a to S

and relax vertex $a$ neighbors, which are vertex $b$ and vertex $c$. That has the following cost from source vertex $a$:

$$d(b) = d(a) + [a,b] = 0 + 6 = 6$$
$$d(c) = d(a) + [a,c] = 0 + 3 = 3$$

Equation 40: Calculate minimum costs to vertex b and vertex c from vertex a

The algorithm will assign previous vertices of vertex $b$ and vertex $c$ as vertex $a$:

$$\pi(b) = a$$

Equation 41: Previous vertex of vertex b

Then add vertex $b$ and vertex $c$ to unsettled set $Q$:

$$Q := Q \cup \{b\} \cup \{c\}$$

Equation 42: Add vertex b and vertex c to Q

After this step the value of unsettled set $Q$ and settled set $S$ will be:

$$Q := \{b(6,a), c(3,a)\}$$
$$s := \{a(0,a)\}$$

Equation 43: Set Q and set S

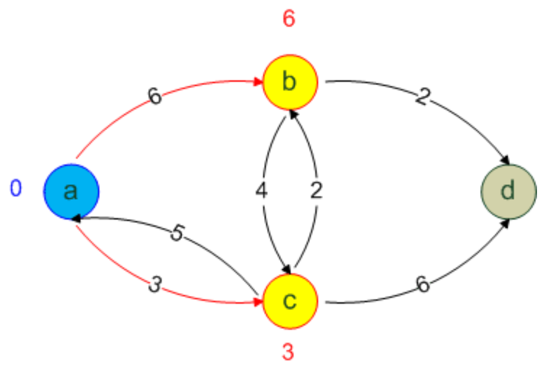


Figure 63: Directed graph after step 2

**Step 3:**
Just like the previous steps, again the algorithm will extract a vertex in set $Q$ that have minimum cost from source vertex $a$, which is vertex $c$:

$$extractMinimum(Q) = c$$

Equation 44: Extract vertex with minimum cost from Q

Then remove vertex $c$ from unsettled set $Q$:

$$Q := Q - \{c\}$$

Equation 45: Remove vertex c from Q

and add vertex $c$ to settled set $S$:

$$S := S \cup \{c\}$$

Equation 46: Add vertex c to S

Then relax vertex $c$ neighbors, which are vertex $a$, $b$, and $d$. The algorithm will ignore vertex $a$ because vertex $a$ is already in settled set $S$.

Again it will calculate the distance from source vertex $a$ to vertex $b$ and vertex $d$, which are the neighboring vertices of vertex $c$. But because the cost from source vertex $a$ to vertex $b$, $d(b)$ is already calculated before at step 2, the algorithm will compare the already calculated cost with a new cost if we pass through other node, vertex $c$ before getting to $b$:

$$d(b) = \min\{d(b), d(c) + [c,b]\} = \min\{6,5\} = 5$$

Equation 47: Minimum cost to vertex b

Because of the algorithm found a shortest route, the previous vertex of vertex $b$ will also be updated:

$$\pi(b) = c$$

Equation 48: Previous vertex of vertex b

The cost and previous vertex of vertex $d$ from the source vertex are:

$$d(d) = d(c) + [c,d] = 9$$

$$\pi(d) = c$$

Equation 49: Minimum cost and previous vertex of vertex d

After that the algorithm will add vertex $b$ and vertex $d$ to unsettled set $Q$:

$$Q := Q \cup \{b\} \cup \{d\}$$

Equation 50: Add b and d to set Q

After this step the value of unsettled set $Q$ and settled set $S$ will be:

$$Q := \{b(5,c), d(9,c)\}$$

$$S := \{a(0,a), c(3,a)\}$$

Equation 51: Set Q and set S

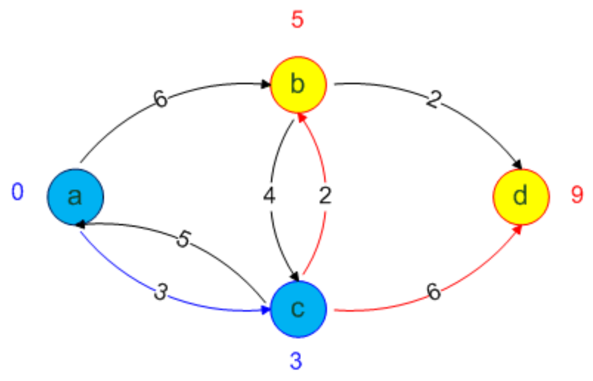


Figure 64: Directed graph after step 3

**Step 4:**

The next vertex in unsettled set $Q$ with minimum cost from source vertex $a$ is vertex $b$. The algorithm will remove vertex $b$ from unsettled set $Q$:

$$Q := Q - \{b\}$$

Equation 52: Remove vertex b from Q

add vertex $b$ to settled set $S$ and relax its neighbors:

$$S := S \cup \{b\}$$

Equation 53: Add vertex b to S

The adjacent vertices of vertex $b$ are: vertex $c$ and vertex $d$. Vertex $c$ will be ignored because it's already in settled set $S$. Just like in step 3, because the cost of vertex $d$ from source vertex $a$ is already been calculated before, the algorithm will compare the already calculated cost with the new cost when it go through vertex $b$:

$$d(d) = \min\{d(d), d(b) + [b,d]\} = \min\{9,7\} = 7$$

Equation 54: Calculate minimum cost to vertex d

The path to $d$ through $b$ is have a lower cost then through $c$, so the algorithm will also update the previous vertex of vertex $d$:

$$\pi(d) = b$$

Equation 55: Previous vertex of vertex d

and vertex $d$ will be added to unsettled set $Q$:

$$Q := Q \cup \{d\}$$

Equation 56: Add d to set Q

After this step the value of unsettled set $Q$ and settled set $S$ will be:

$$Q := \{d(7,b)\}$$

$$S := \{a(0,a), c(3,a), b(5,c)\}$$

Equation 57: Set Q and set S

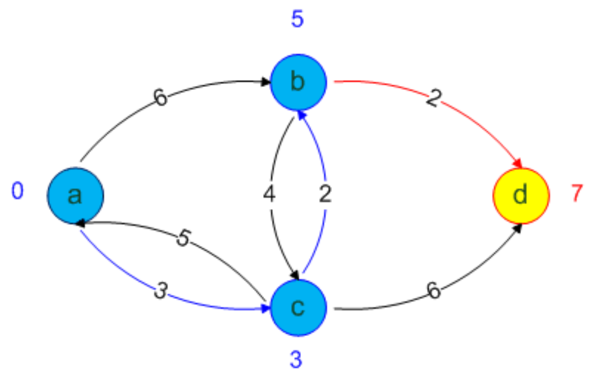


Figure 65: Directed graph after step 4

**Step 5:**

Extracting the minimum of unsettled set $Q$ resulting in vertex $d$, vertex $d$ will then be removed from unsettled set $Q$, add to settled set $S$, and relax vertex $d$ neighbors.

At this point all vertex $d$ neighbors are settled and unsettled set $Q$ is empty, so the algorithm will end. The following figure is the resulting shortest path from a to every other nodes in graph $G$.

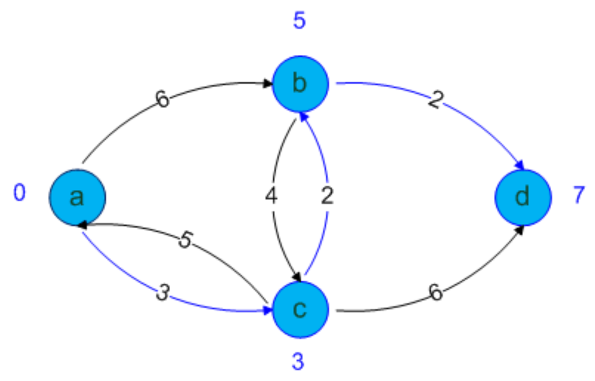


Figure 66: Directed graph after step 5

The final value of unsettled set $Q$ and settled set $S$ are:

$$Q := \{\ \}$$

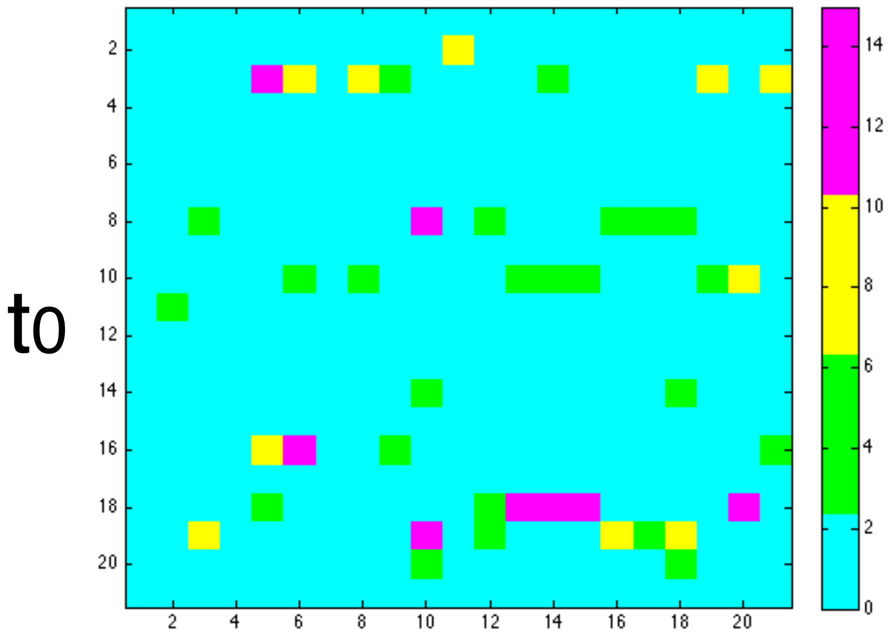
$$S := \{a(0,a), c(3,a), b(5,c), d(7,b)\}$$

Figure 67: Set Q and set S

After the algorithm ends the final result is the settled set $S$ consisting of an optimal shortest path tree from the source vertex to every other vertices.

# Appendix B

# Experiment Results



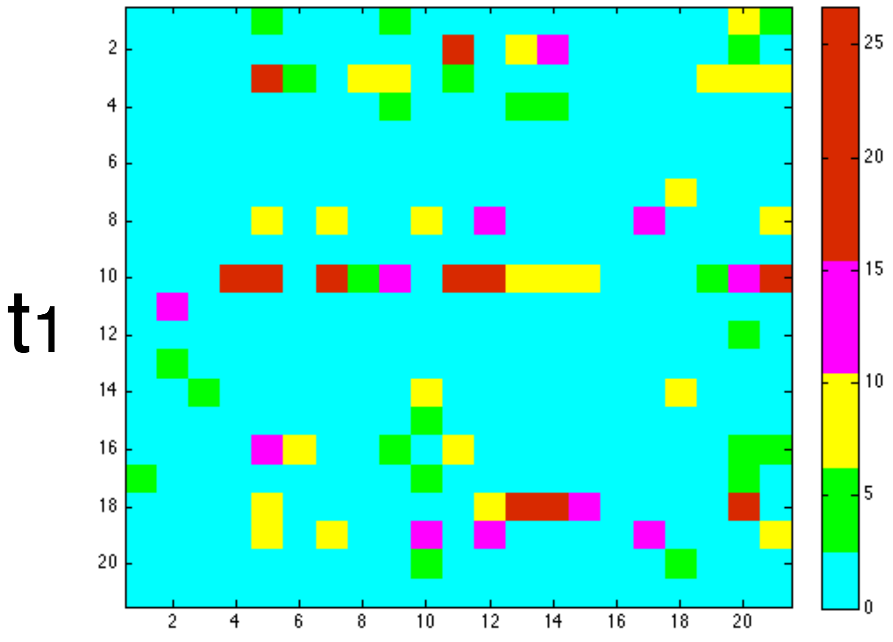
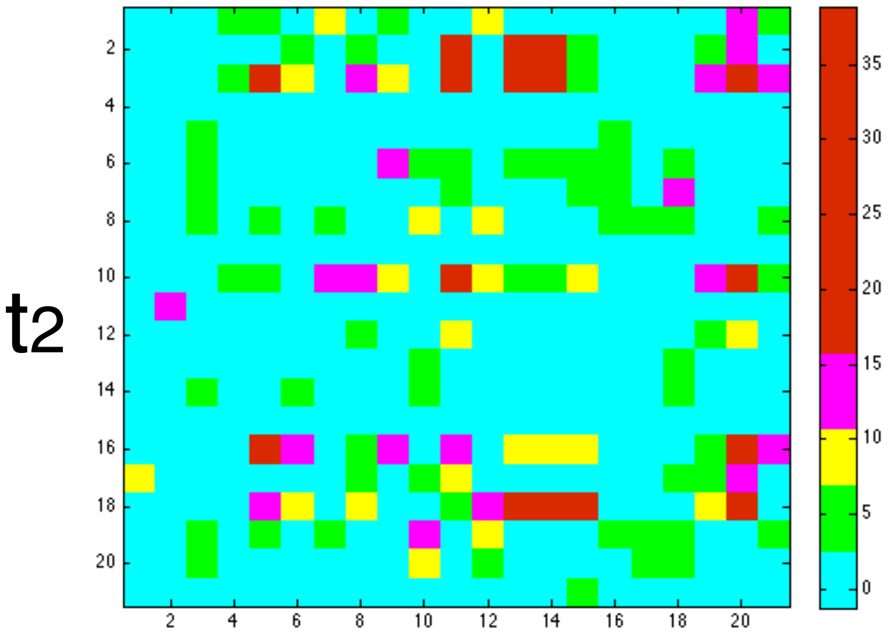


Figure 68: Experiment result with $t_0$=15:20 and $t_1$=15:40



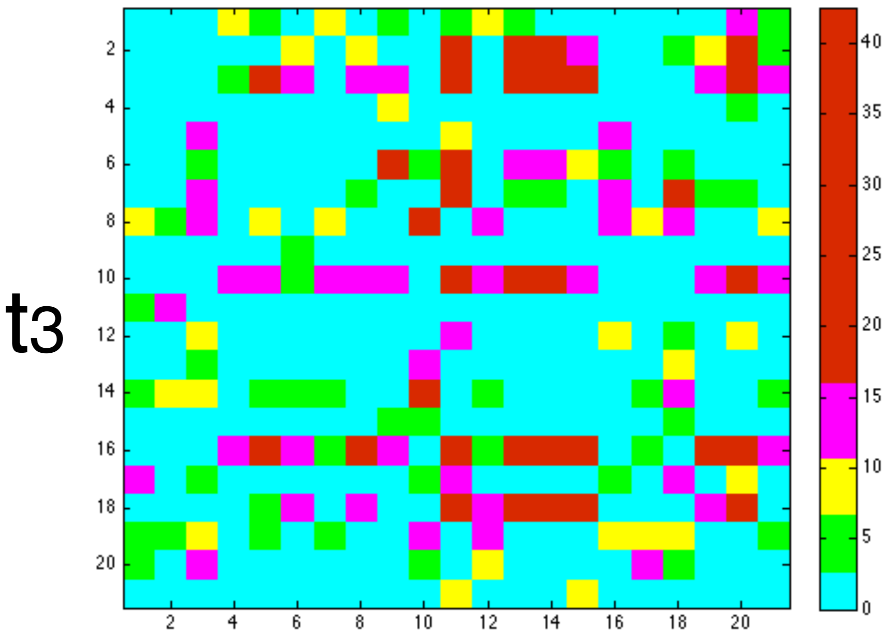
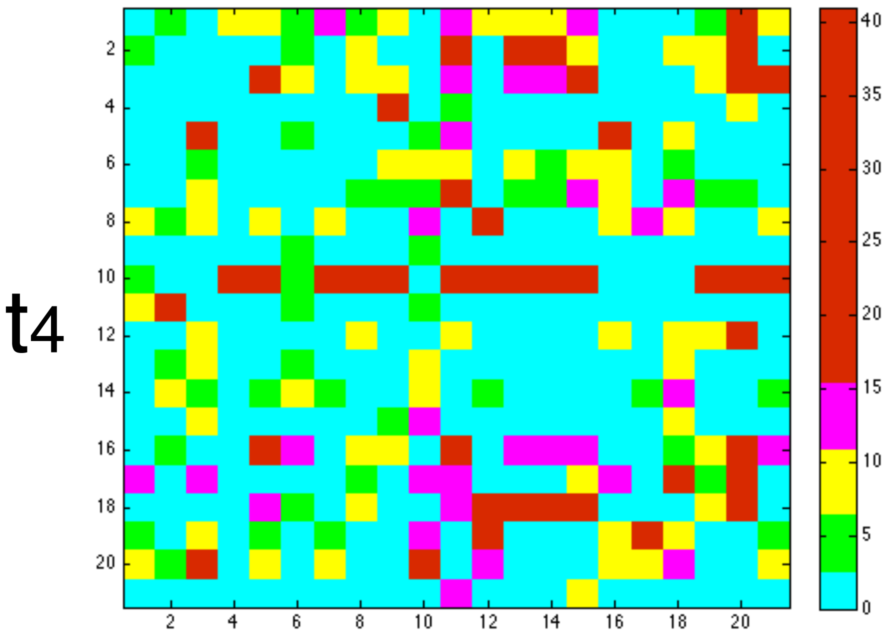


Figure 69: Experiment result with $t_2$=16:00 and $t_3$=16:20



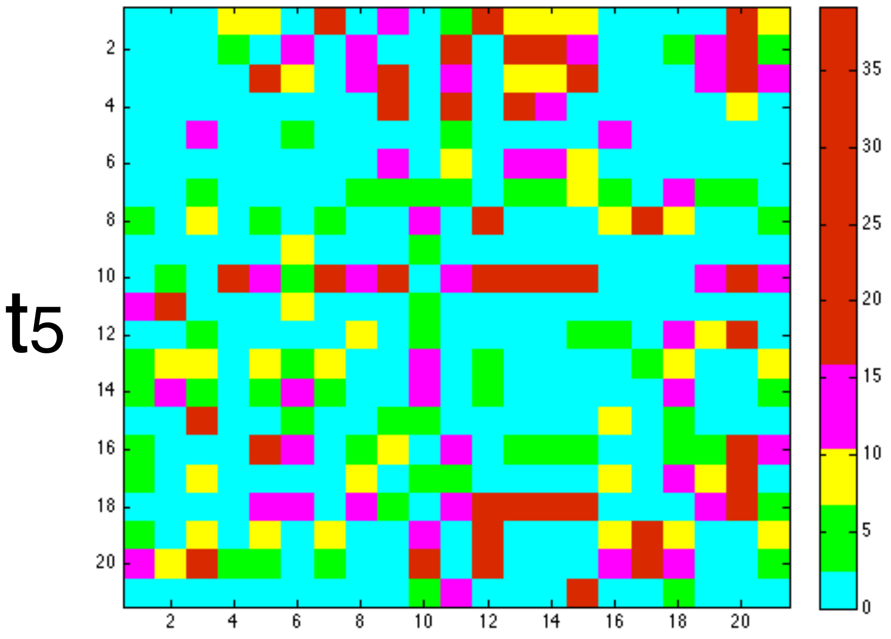
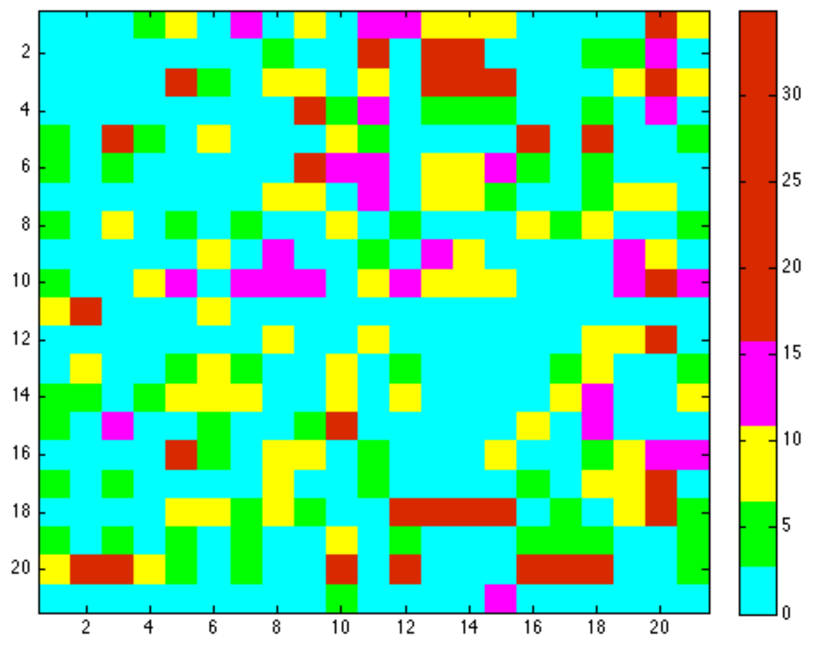


Figure 70: Experiment result with $t_4$=16:40 and $t_5$=17:00

t6



t7

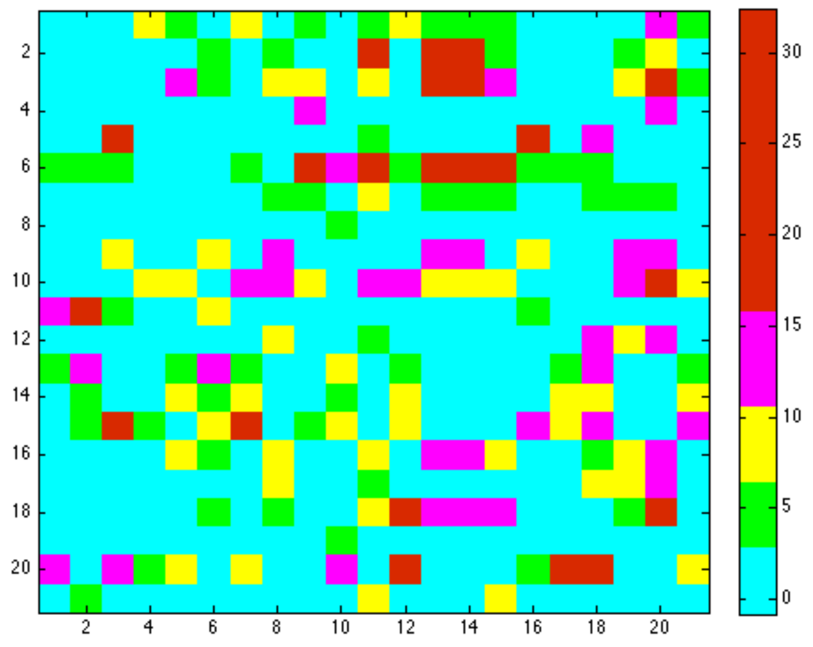
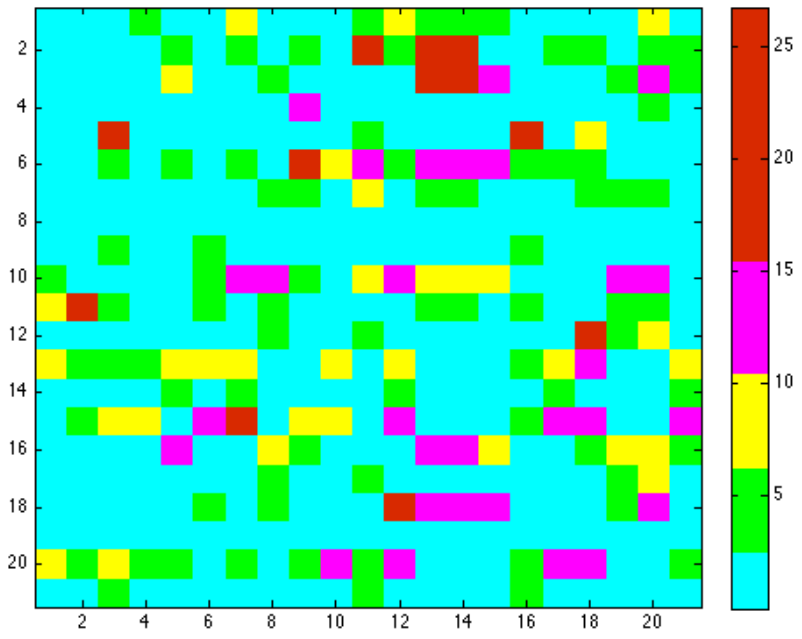


Figure 71: Experiment result with $t_6$=17:20 and $t_7$=17:40

t8



t9

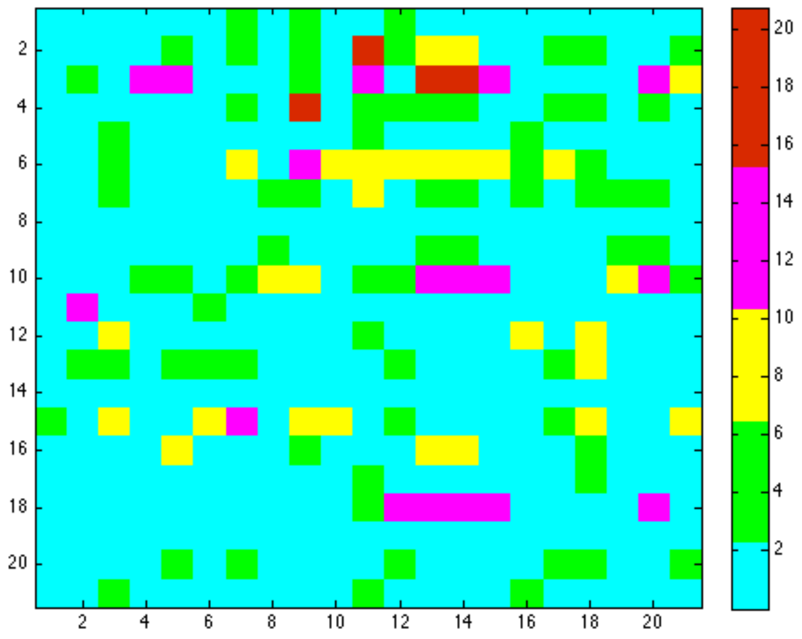
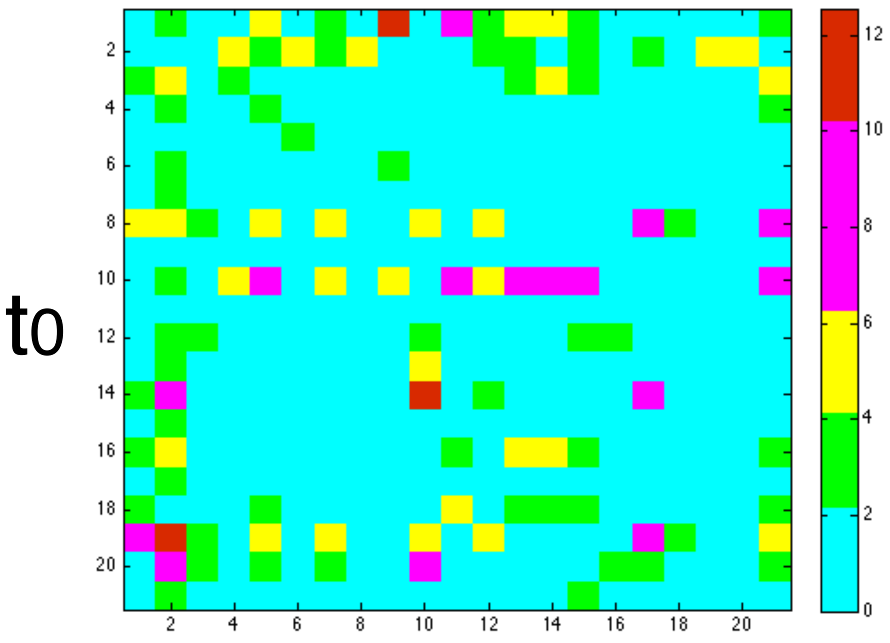


Figure 72: Experiment result with $t_8$=18:00 and $t_9$=18:20

# Appendix C

# Data Reliability and Consistency Results with Dataset #1



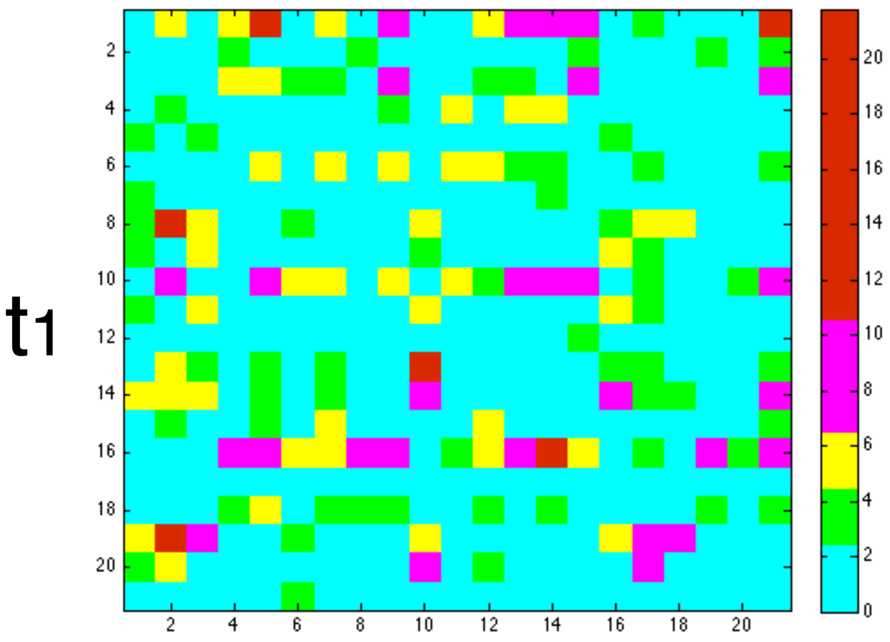
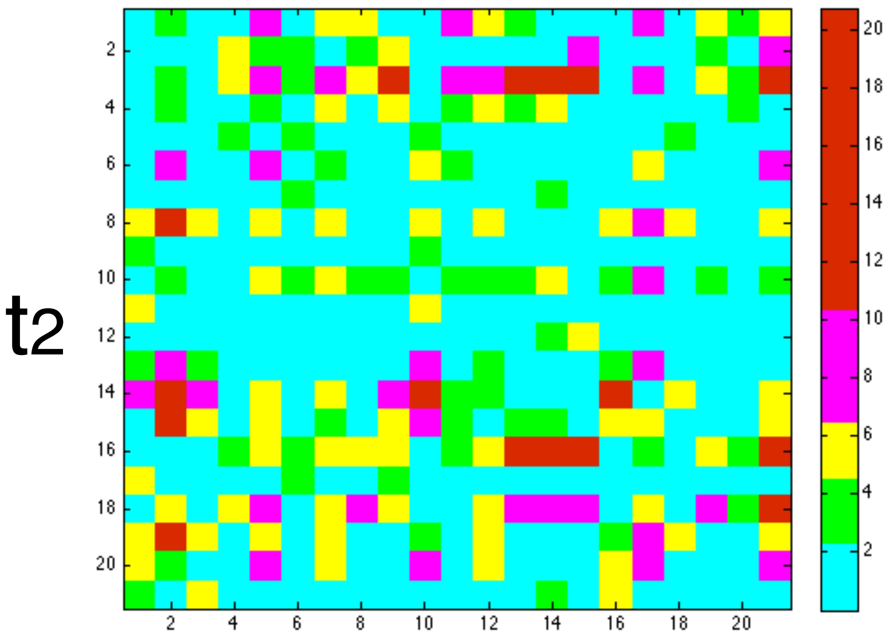


Figure 73: R(t,1) with $t_0$=15:20 and $t_1$=15:40



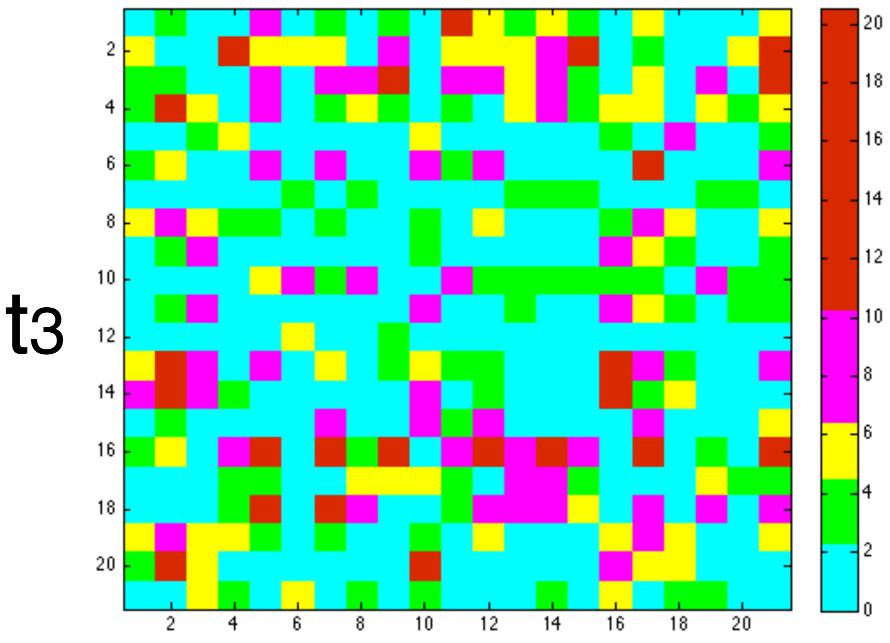
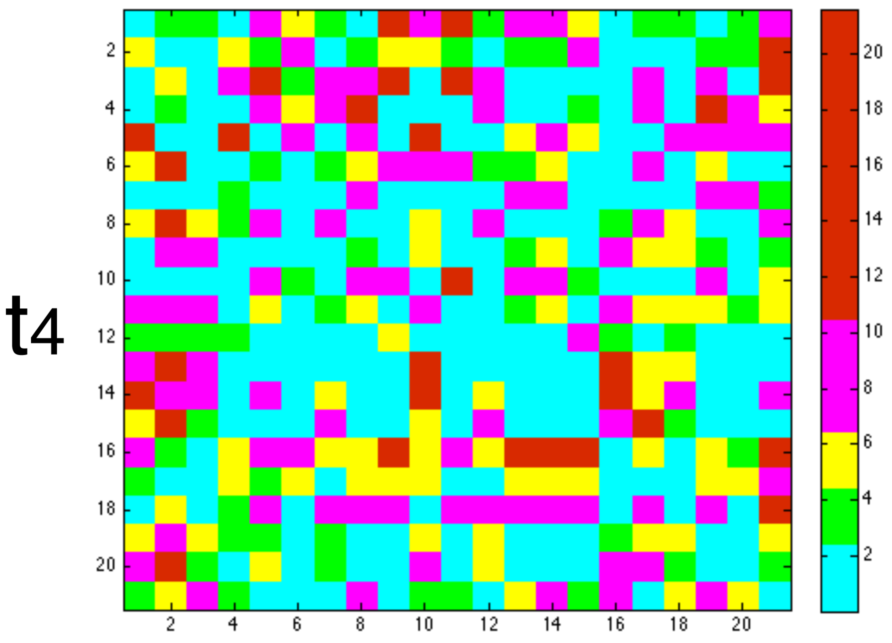


Figure 74: R(t,1) with $t_2$=16:00 and $t_3$=16:20



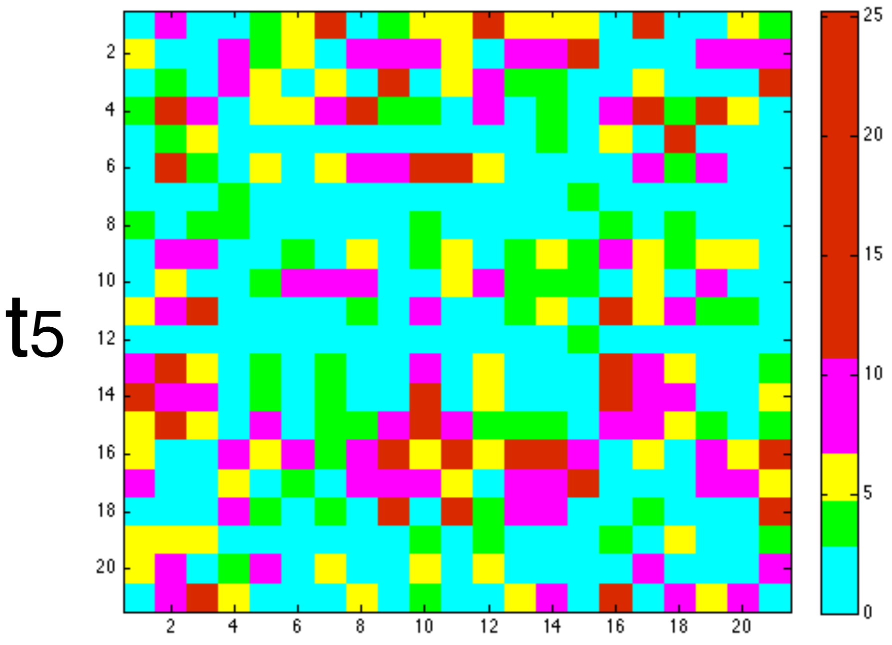
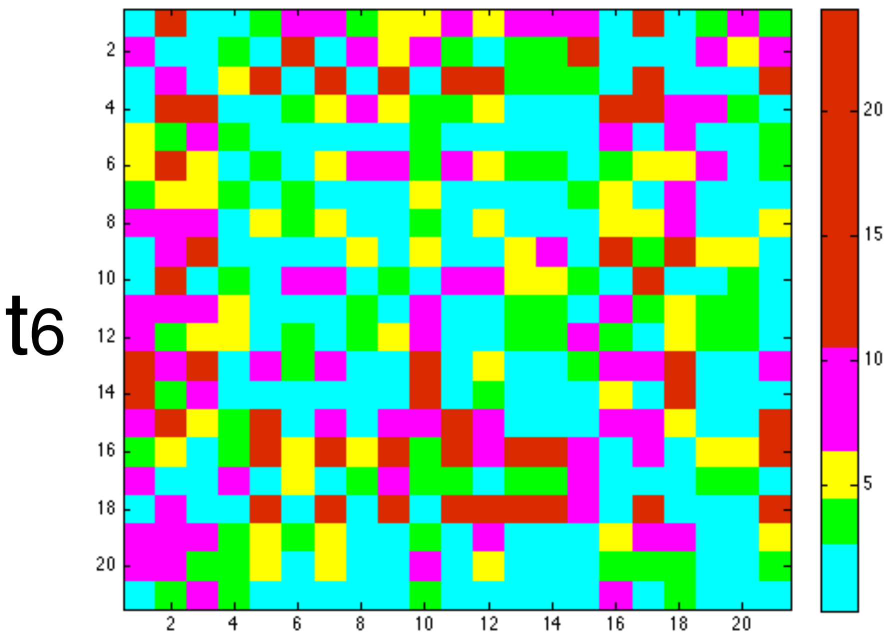


Figure 75: R(t,1) with $t_4$=16:40 and $t_5$=17:00



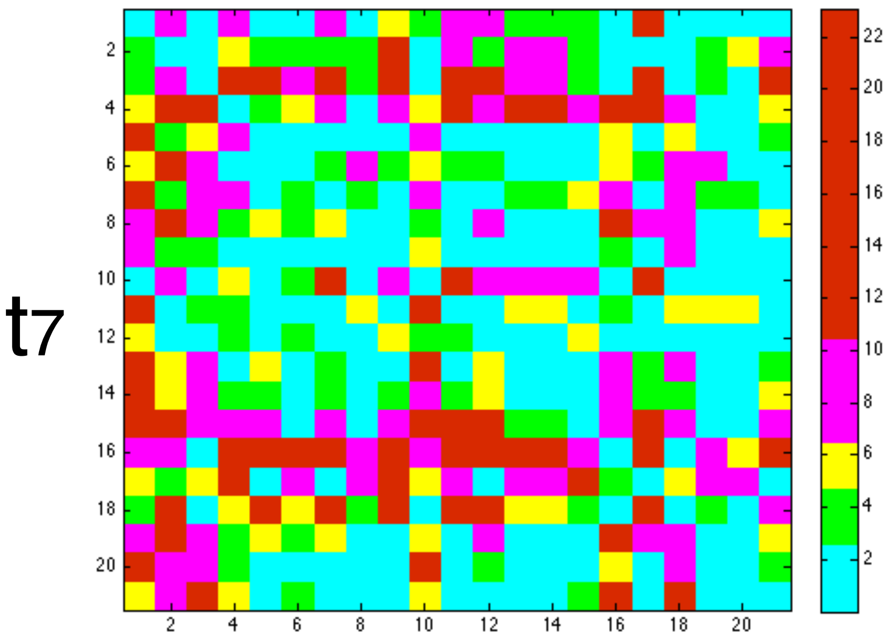
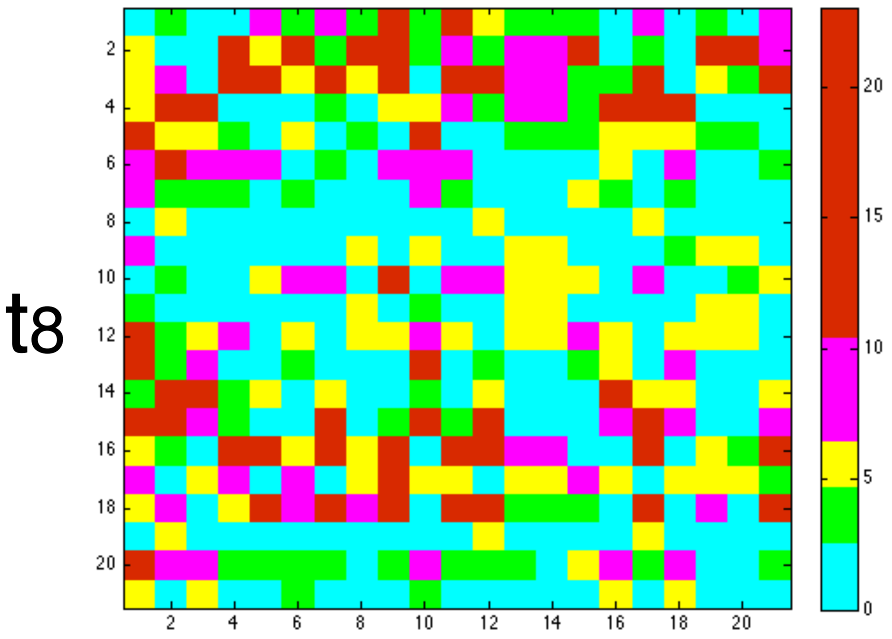


Figure 76: R(t,1) with $t_6$=17:20 and $t_7$=17:40



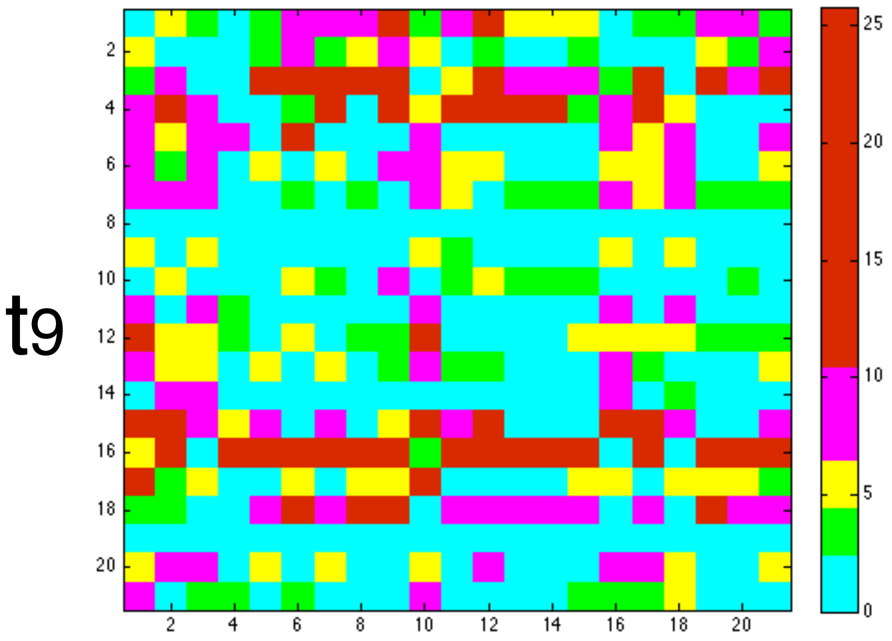
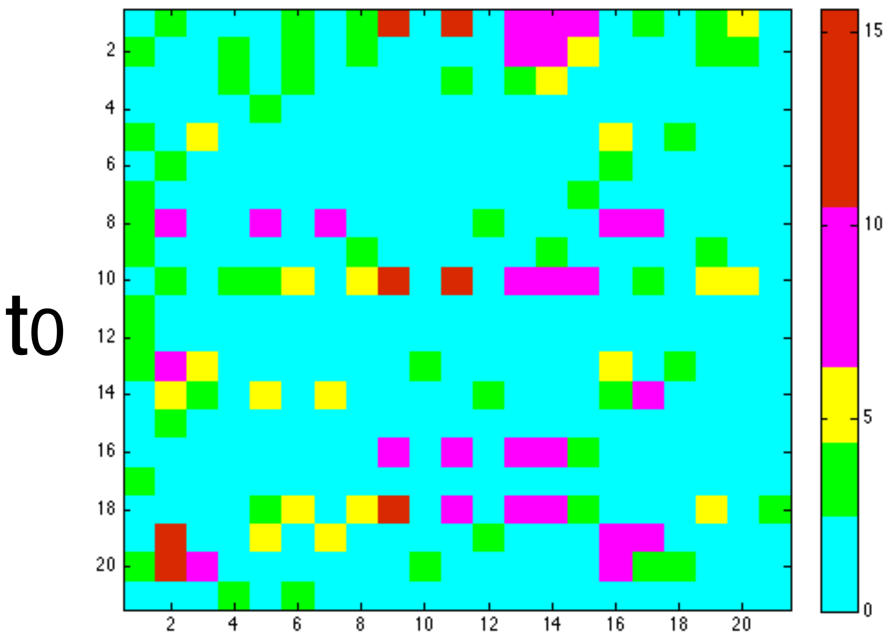


Figure 77: R(t,1) with $t_8$=18:00 and $t_9$=18:20

# Appendix D

# Data Reliability and Consistency Results with Dataset #2



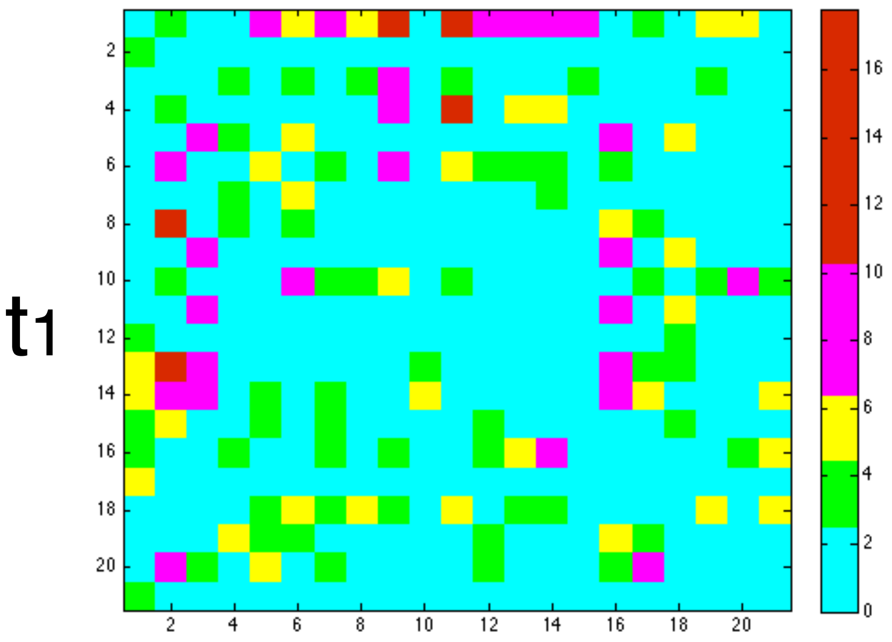
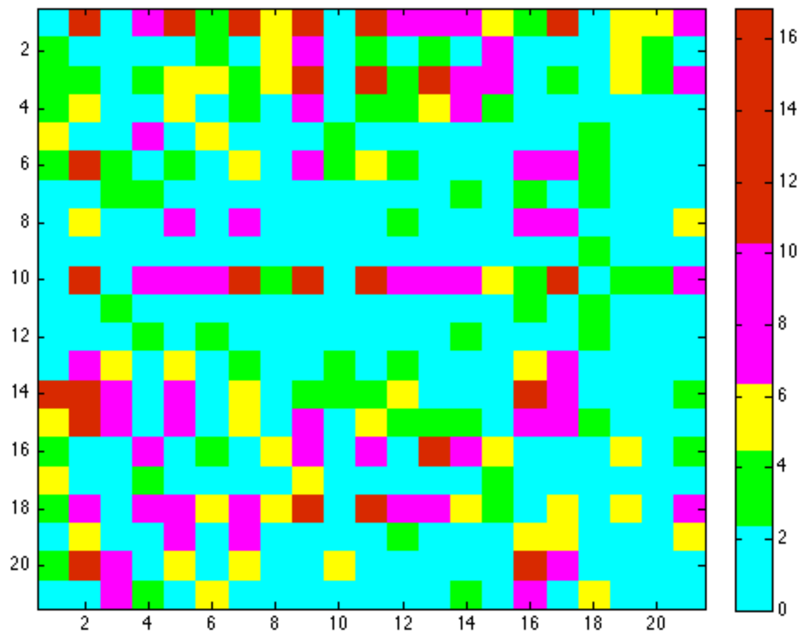


Figure 78: R(t,2) with $t_0$=15:20 and $t_1$=15:40

t2



t3

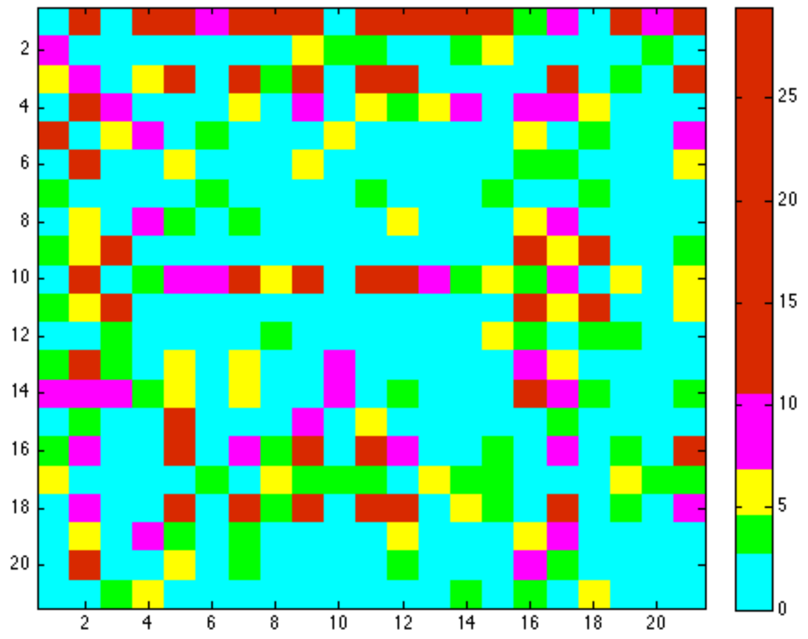
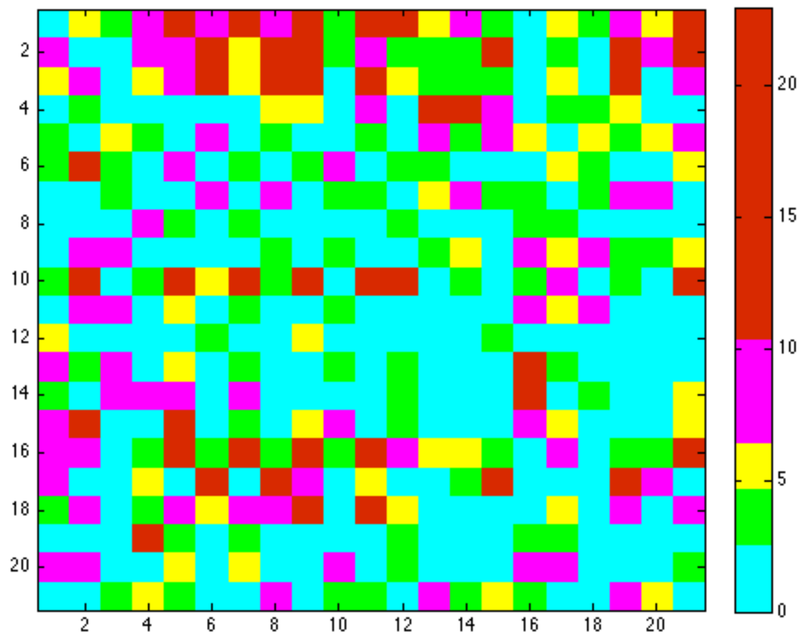


Figure 79: R(t,2) with $t_2$=16:00 and $t_3$=16:20

t4



t5

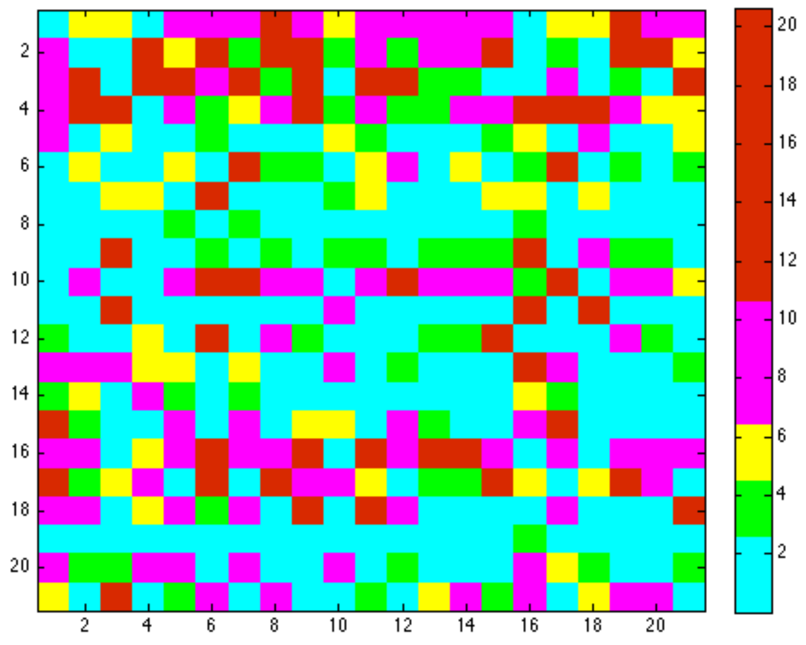
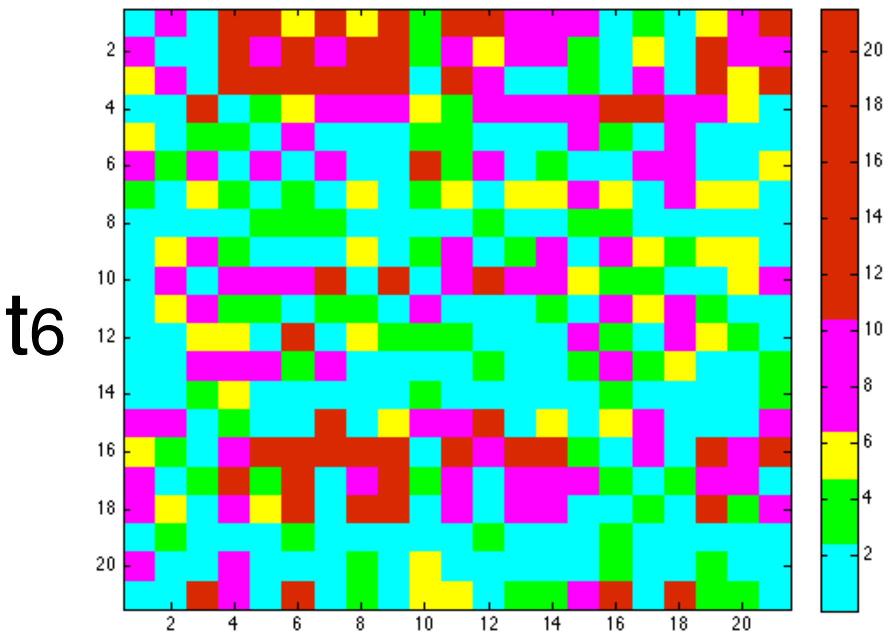


Figure 80: R(t,2) with $t_4$=16:40 and $t_5$=17:00



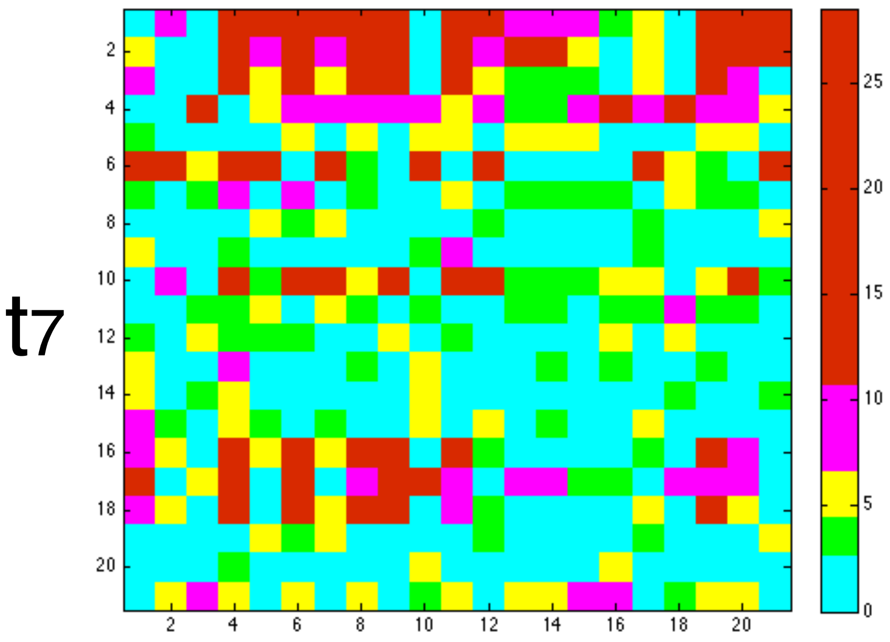
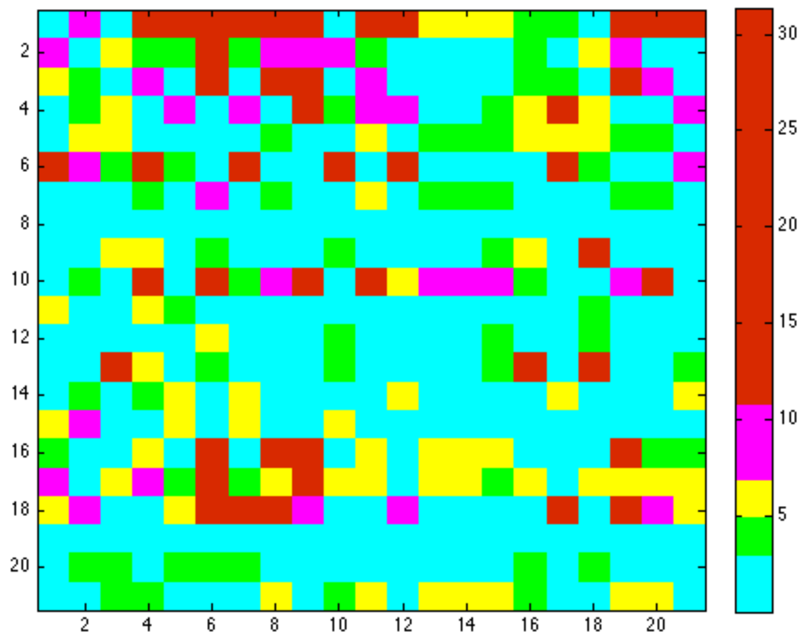


Figure 81: R(t,2) with $t_6$=17:20 and $t_7$=17:40

t8



t9

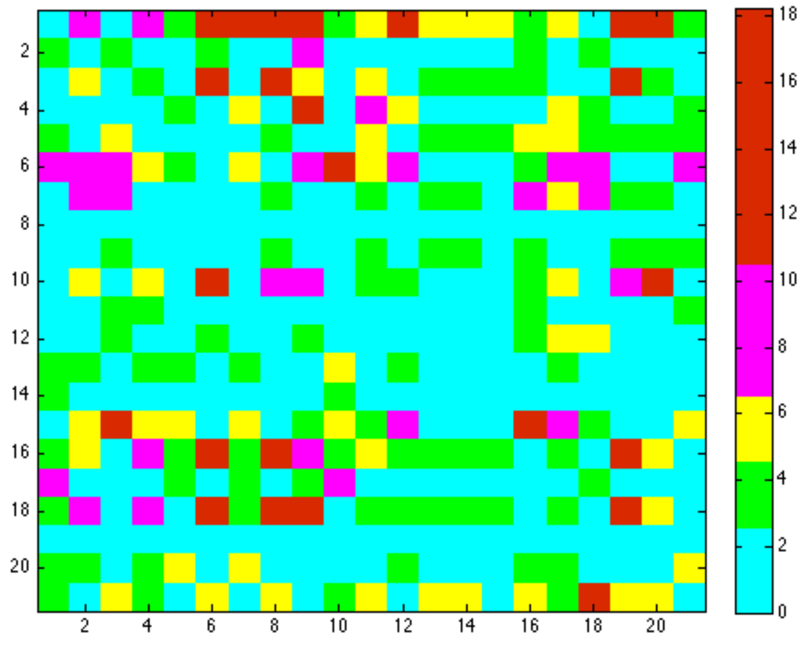
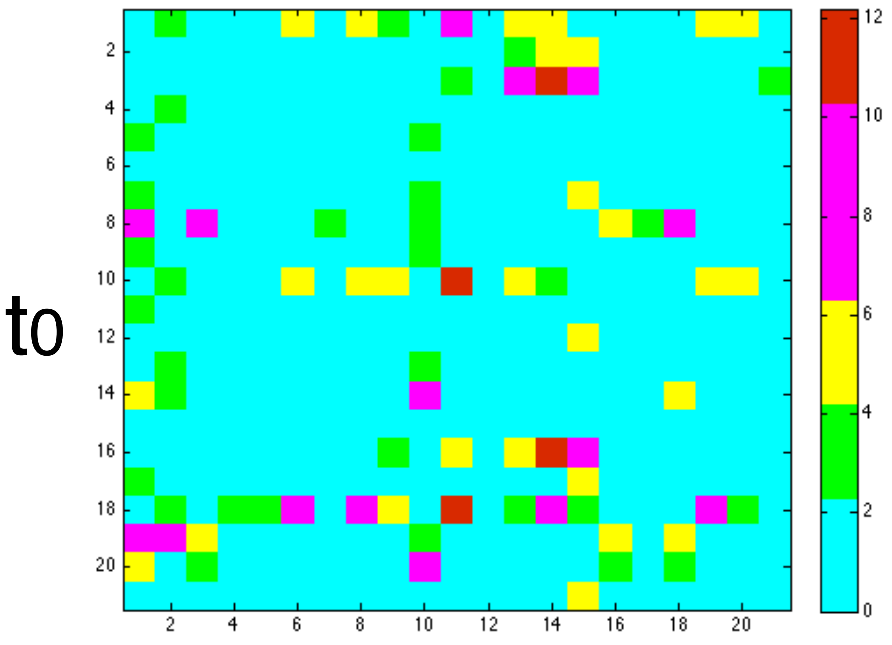


Figure 82: R(t,2) with $t_8$=18:00 and $t_9$=18:20

# Appendix E

# Data Reliability and Consistency Results with Dataset #3



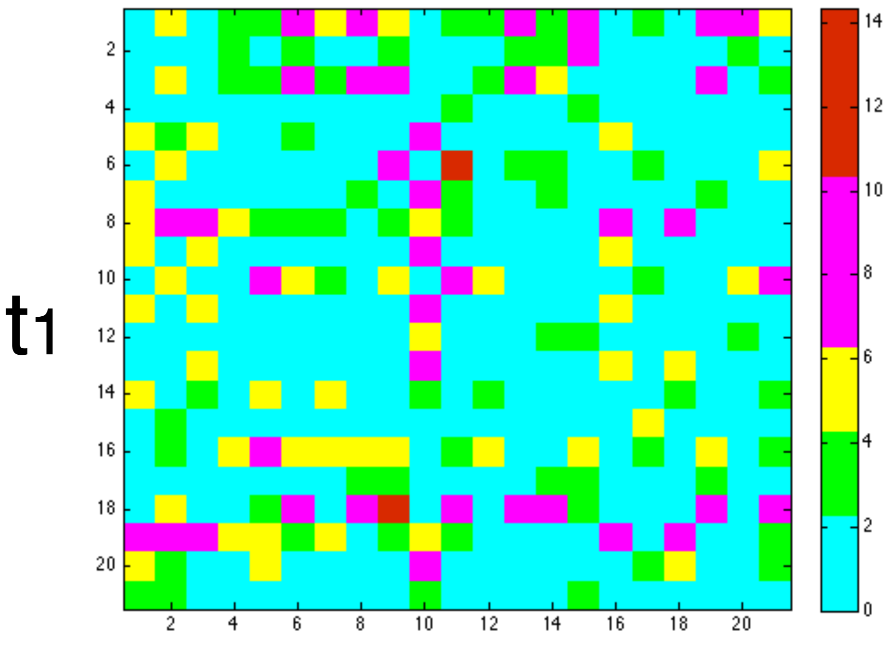
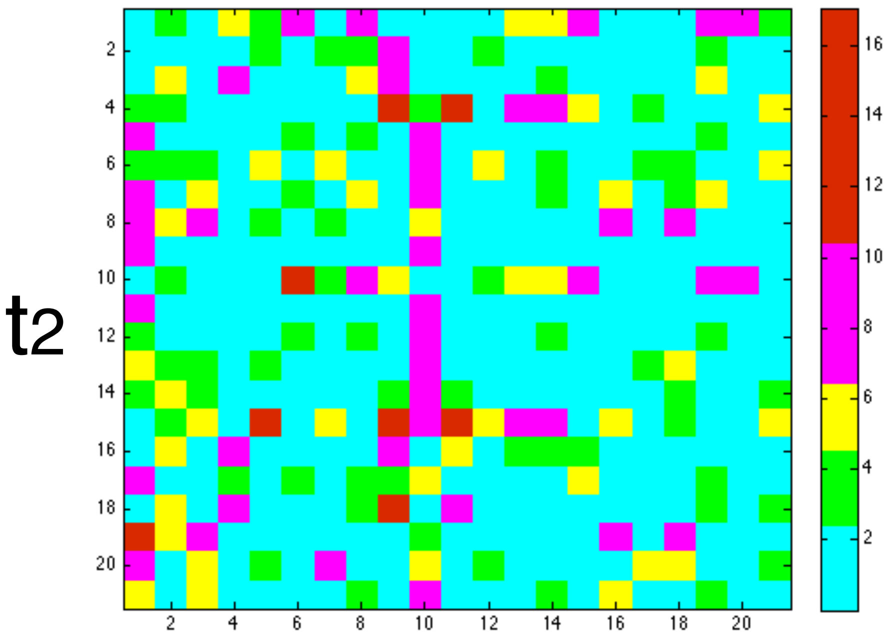


Figure 83: R(t,3) with $t_0$=15:20 and $t_1$=15:40



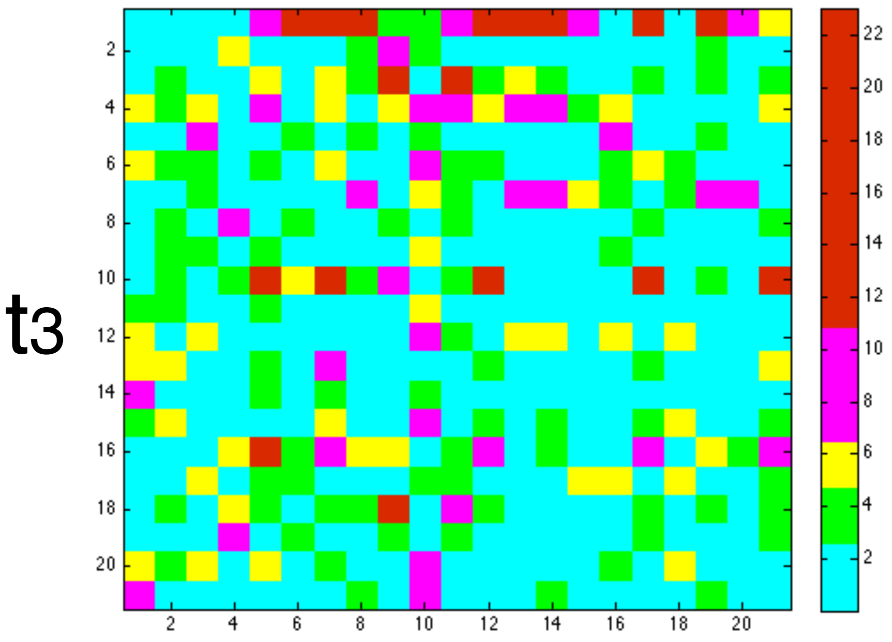
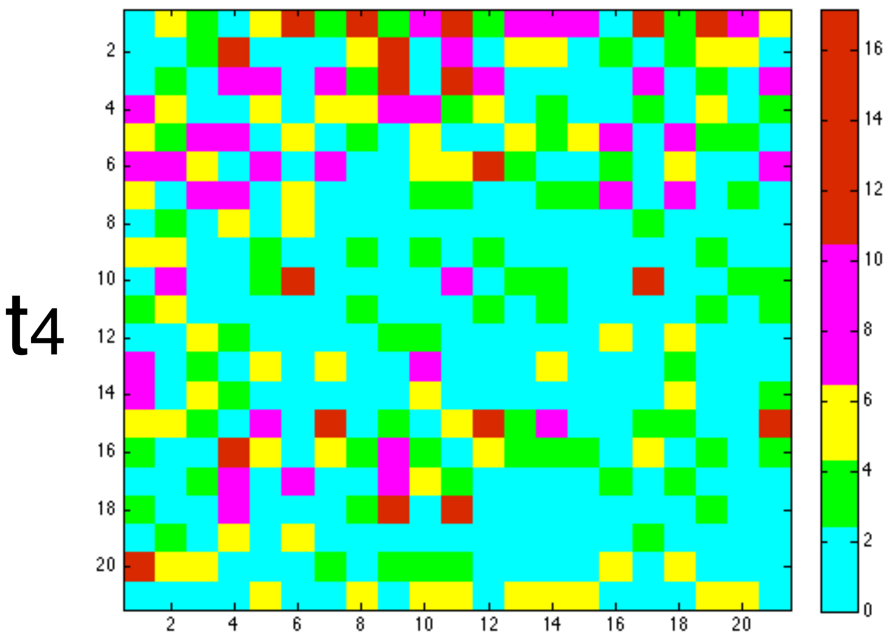


Figure 84: R(t,3) with $t_2$=16:00 and $t_3$=16:20



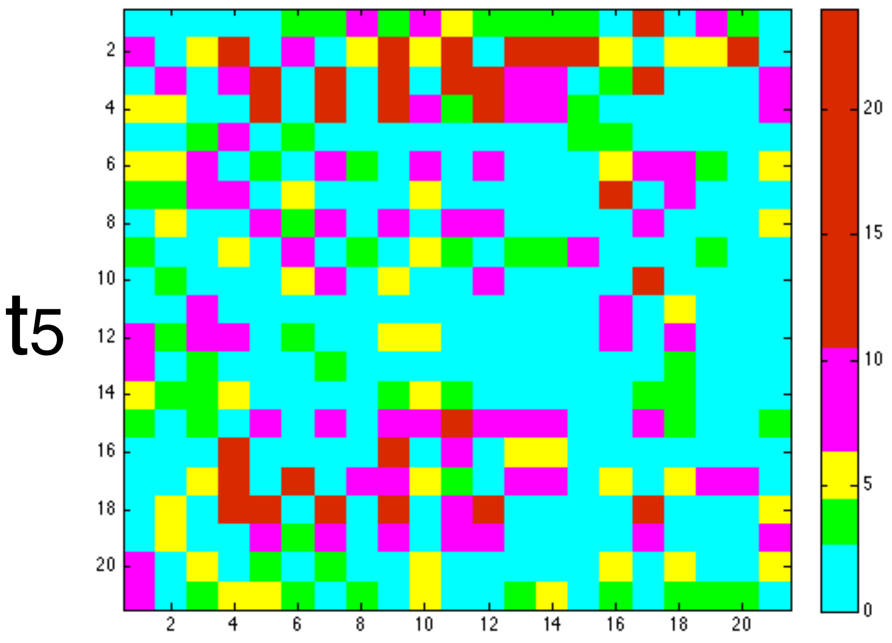
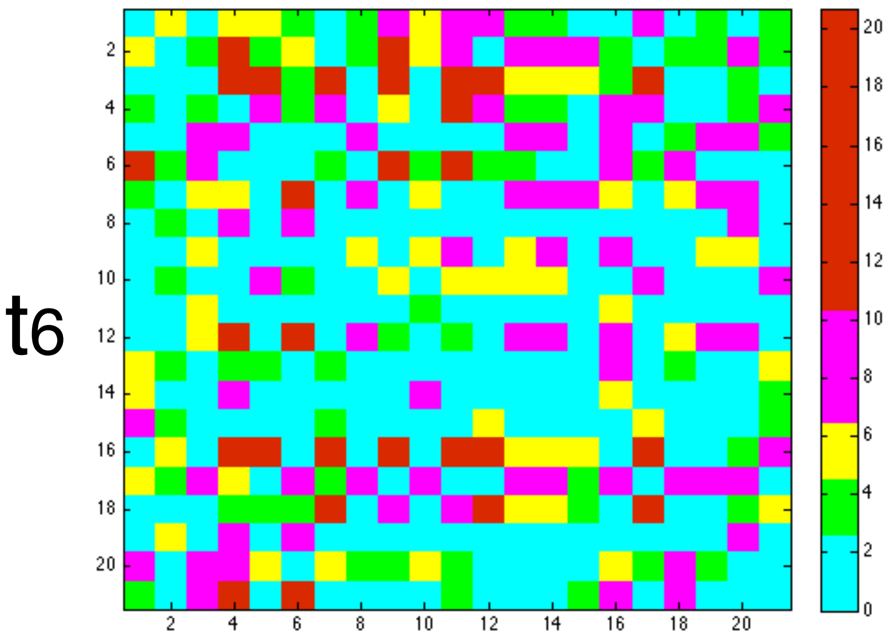


Figure 85: R(t,3) with $t_4$=16:40 and $t_5$=17:00



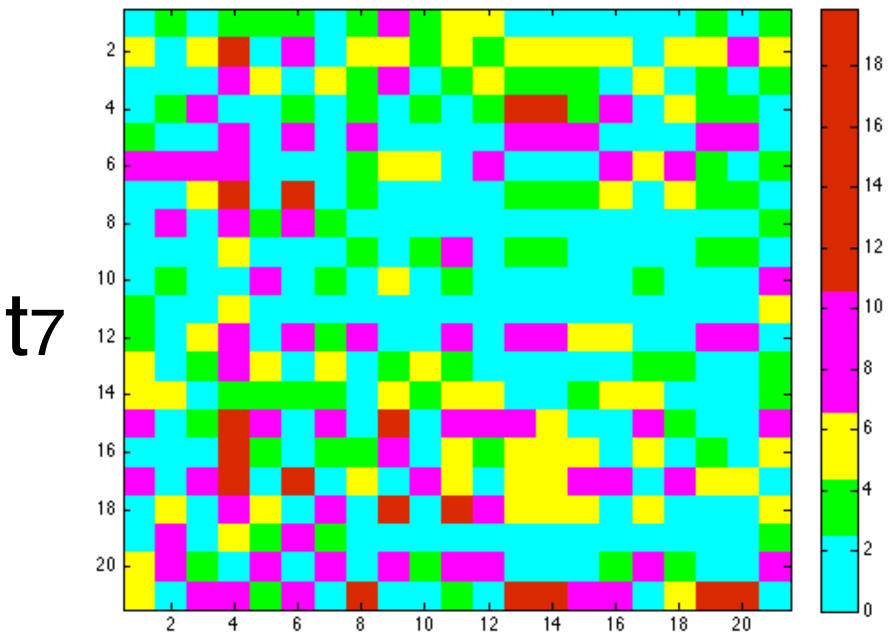
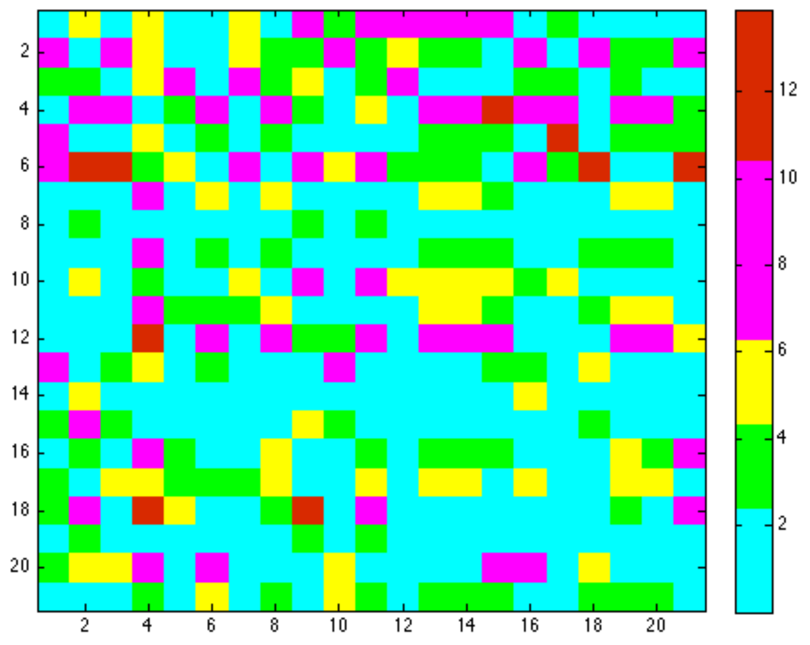


Figure 86: R(t,3) with $t_6$=17:20 and $t_7$=17:40

t8



t9

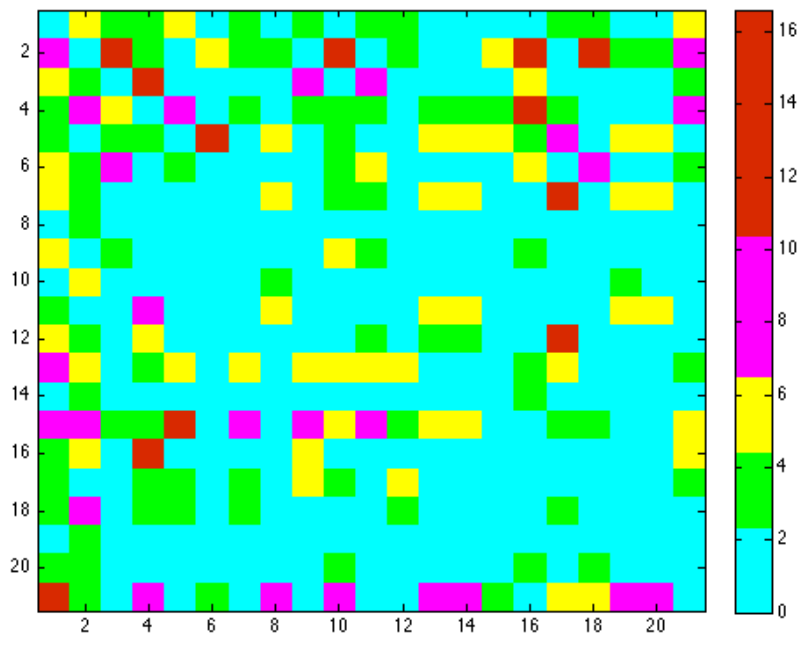


Figure 87: R(t,3) with $t_8$=18:00 and $t_9$=18:20

Appendix F

# Data Reliability and Consistency Results with Dataset #4

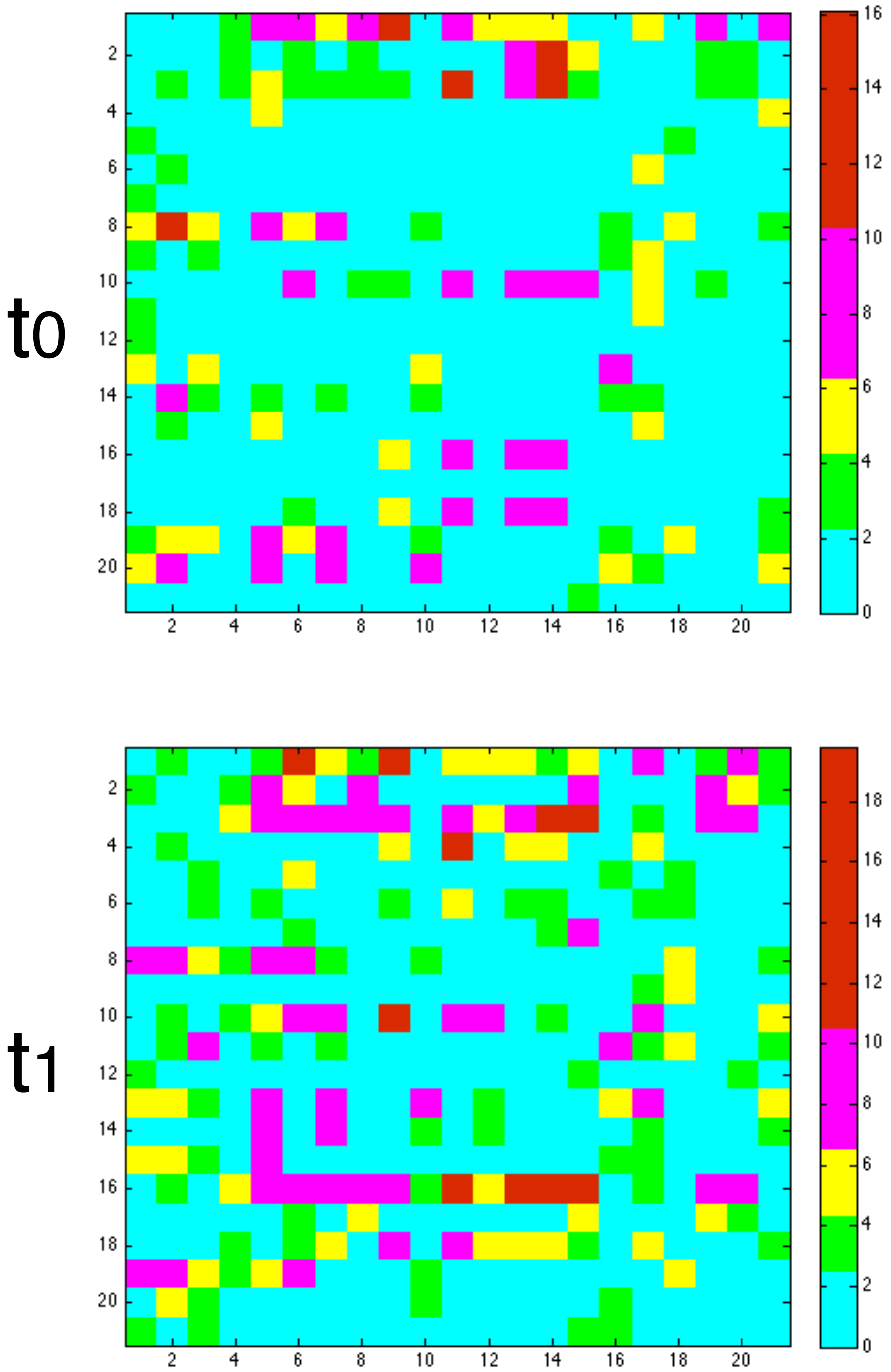


Figure 88: R(t,4) with $t_0$=15:20 and $t_1$=15:40

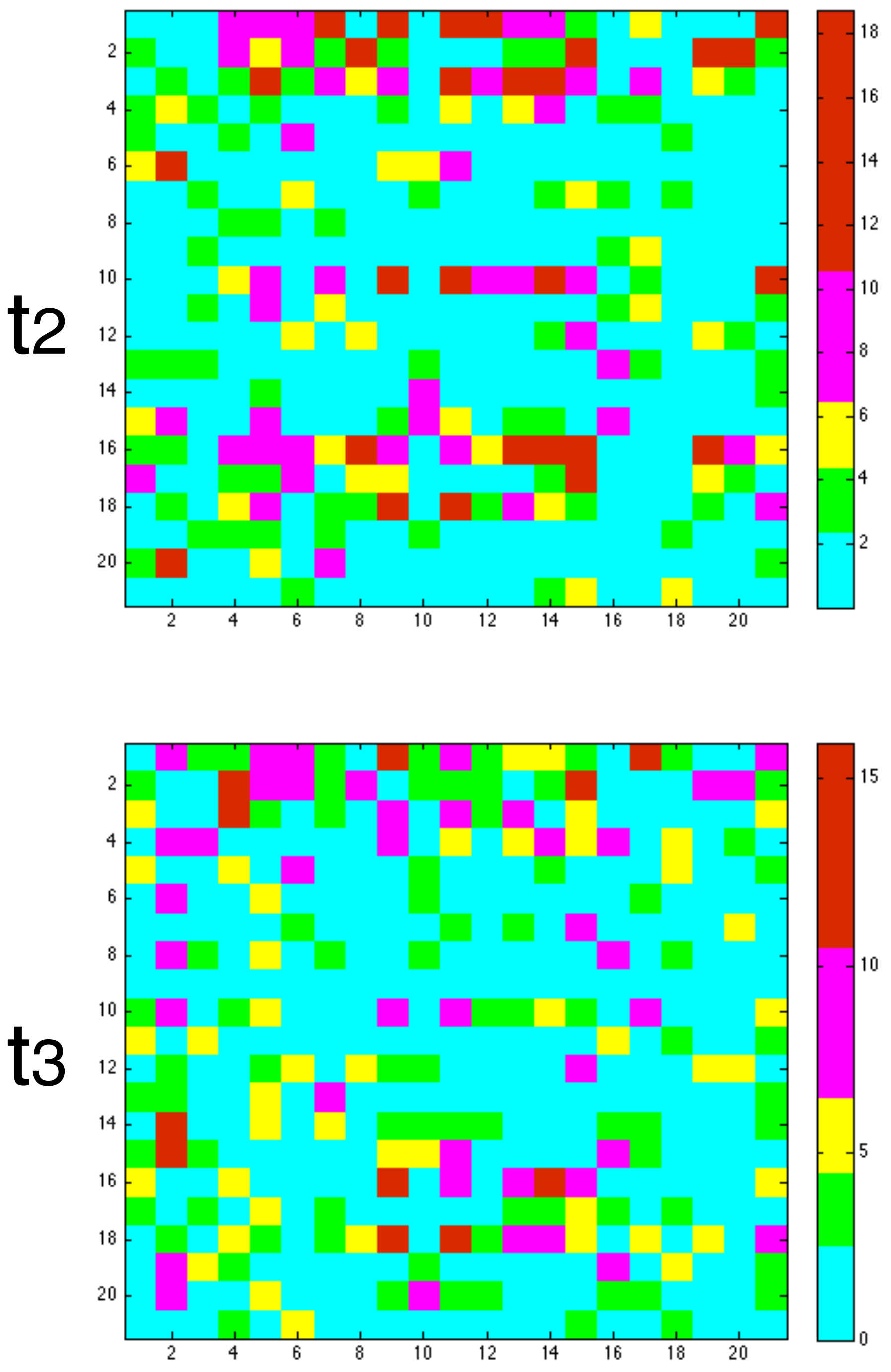
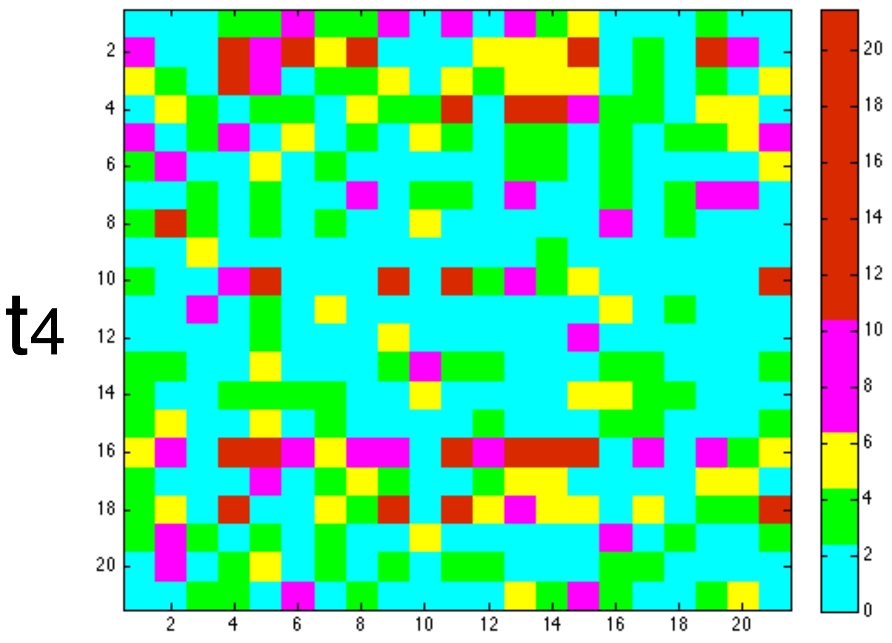


Figure 89: R(t,4) with $t_2$=16:00 and $t_3$=16:20



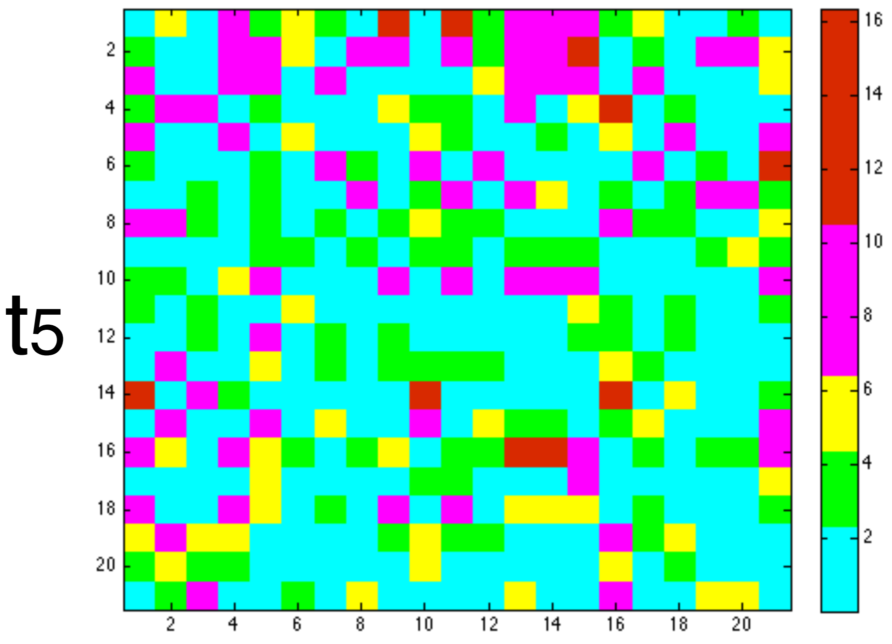
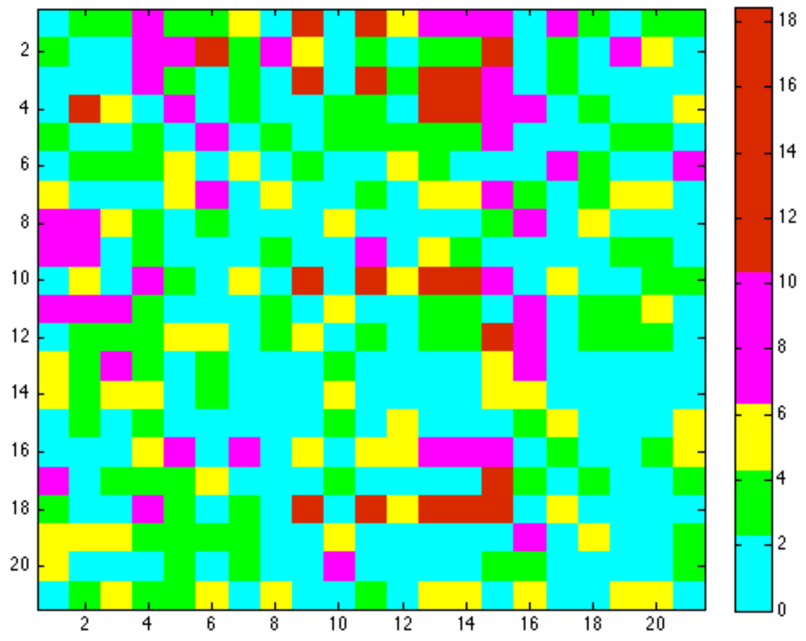


Figure 90: R(t,4) with $t_4$=16:40 and $t_5$=17:00

t6



t7

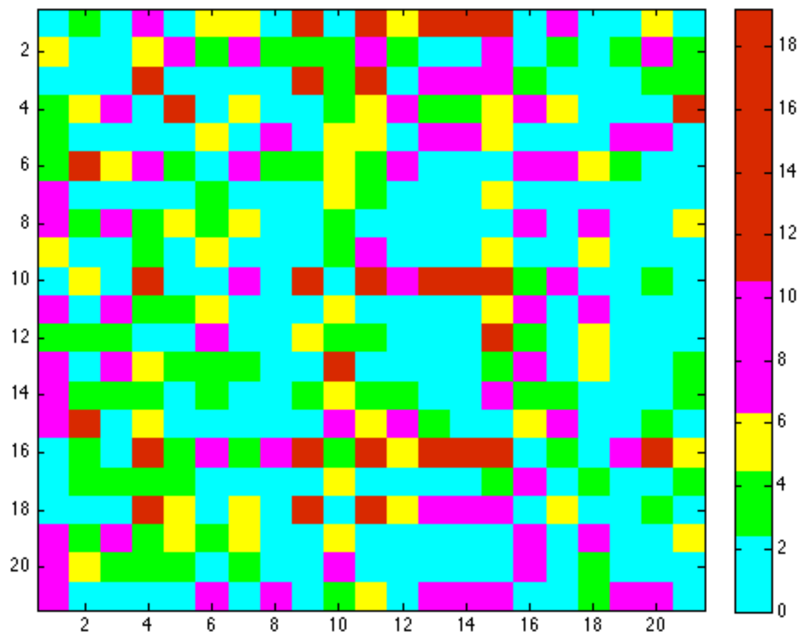
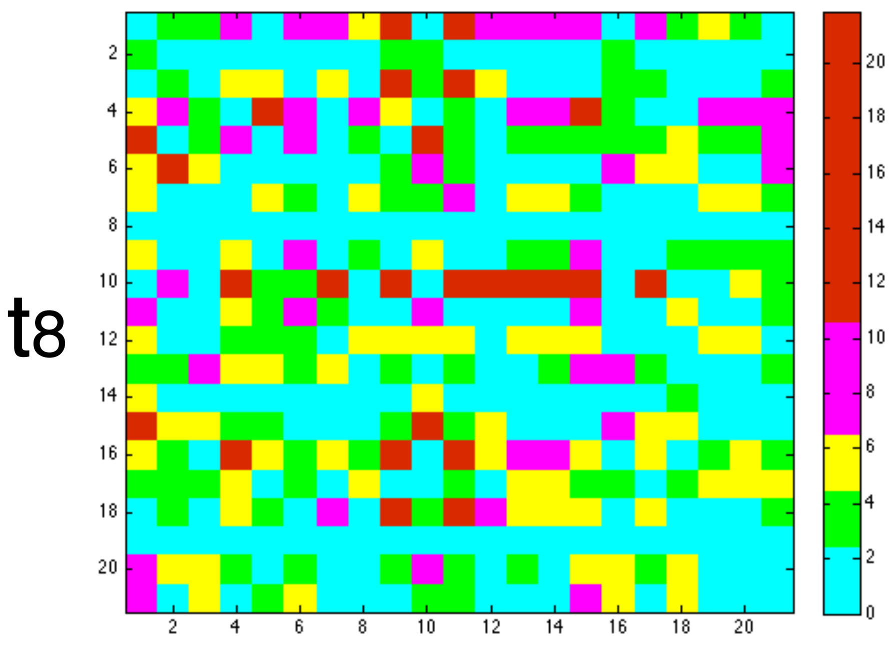


Figure 91: R(t,4) with $t_6$=17:20 and $t_7$=17:40



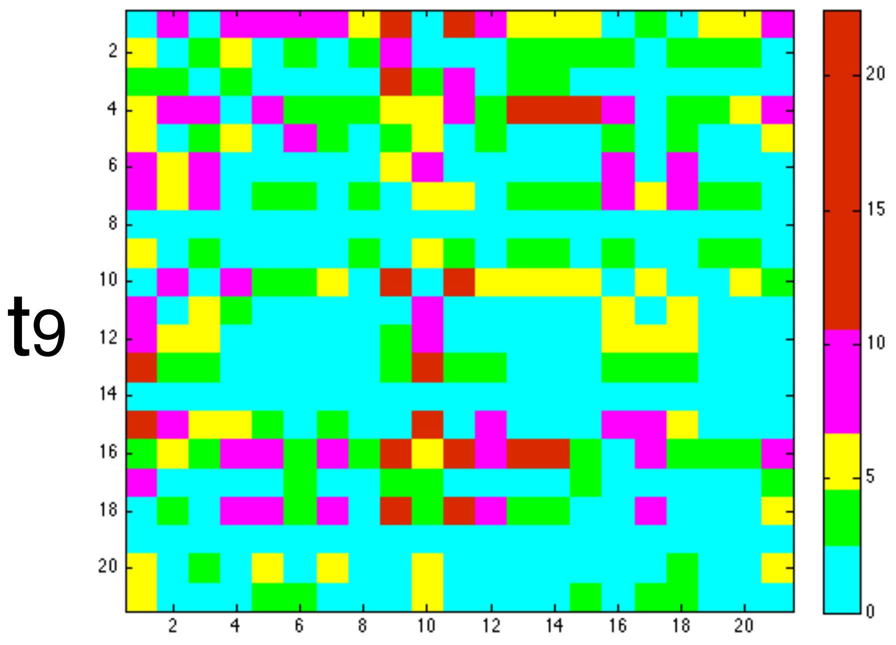


Figure 92: R(t,4) with $t_8$=18:00 and $t_9$=18:20