Intelligent Driver Agent Model for Autonomous Navigation in a Computer Simulated Vehicular Traffic Network

A thesis presented in total
fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

at Massey University,
Albany (Auckland), New Zealand.

Gligor Kotushevski

September 2010

Abstract

The purpose of this study was to investigate the possibilities of automating vehicular traffic and decrease traffic congestion by developing an intelligent driver agent model that autonomously navigates through a computer simulated traffic network. The aim was to examine various pathfinding algorithms and cost evaluation functions through different traffic conditions so that a basic intelligent driver agent model is designed using the best combination of algorithms and cost functions found.

A computer simulation of vehicular traffic has been implemented to study different agent models. The intelligent driver agents developed act as independent entities with their own emergent properties and individual behaviours. Each simulated vehicle was navigated through the traffic network to its destination using a user defined algorithm and cost function. The case studies conducted focused on measuring the travel times of each driver agent from the starting to the destination point.

The results indicated that the agents traveled at higher average speeds under low density traffic conditions, while lowering their average speed as the traffic density increased. It was also discovered that hybrid cost evaluation functions (designed by combining two or more basic cost functions) perform better in low and medium density traffic, while basic cost functions perform better under high density traffic conditions. Finally, the results revealed that Dijkstra pathfinding using a hybrid combination of time and length cost functions should be used under low and medium density traffic conditions and D* pathfinding using congestion cost evaluation function under high density traffic conditions.

The conclusion was that the intelligent driver agent model implemented is suitable to be used as a navigation model for self-driving vehicles in traffic simulation software, but also given the right technology and social acceptance it is suitable to be implemented as a navigation model for robot vehicles and deployed in real world traffic situations.

Acknowledgements

Firstly, I wish to thank my parents Nikola and Lidija for encouraging and supporting me to achieve my dreams. Thank you for always believing in me and my abilities, and for rasing me become the person I am today. I wish to thank my sister Ljubica and my brother in law Bojan for always being there for me and helping me adjust to this new environment. Thank you for opening your home to me and welcoming my sometimes burdensome presence. Without you I would not have been here. I truly love the four of you and will always owe you my deepest gratitude.

I would like to thank my grandparents Todor, Vlado, Ljubica and Marija for their love and support through my life. You are always in my heart even when I am far away from you.

I also wish to thank my uncle and aunt Gjoko and Biljana, and my two cousins Marija and Vladimir whose help brought me here in New Zealand and without who my life would not feel complete. I love all of you dearly and I am blessed to have such an amazing family.

I want to thank Professor Ken Hawick for his continuous guidance through this project, without whose supervision and support this thesis would not have been possible. Thank you for the great ideas and feedback which helped me persevere through this process and complete this degree.

I wish to thank all my colleagues and friends Teo Susnjak, Arno Leist, Daniel Playne for their support and encouragement, and a special thanks to Guy Kloss for sharing his unlimited knowledge and helping me with Latex, Gnuplot, Linux, Python and a million other things. It is an honor for me to be part of a great bunch like you guys.

I also owe my gratitude to Massey University for supporting my research and providing a scholarship to fund part of the student fees during my studies.

Last but not least, I would like to thank all my friends who helped me focus and keep my mind on the important tasks even though they did not understand a bit of what I was doing. Special thanks to my buddies Martin and Mile for the countless coffee breaks and hours spent warming up for thesis writing, as well as to Nazley and Amy who both took time from their busy lives to help me spend more time on my research. I would like to thank my mate Viktor for always making my mornings more enjoyable by sharing amusing stories of our lives and remembering the good old times, as well as my colleagues Jocki and Verdi for their moral support both while working together and throughout my research. You guys are all truly amazing and I am honored to know each and every one of you.

- Gligor Kotusevski, Auckland, 2010

Contents

List of Figures

List of Tables

Twenty years from now you will be more disappointed by the things that you didn't do than by the ones you did do. So throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover.

Mark Twain

1 Introduction

1.1 Motivation

In today's busy world, traffic has become part of our life. While in the past people lived in smaller towns and used to walk from one place to another, today in the big cities it is nearly impossible to reach a destination by walking. That is why we have adopted driving in our everyday lives. However, cities have become so overpopulated that even though their traffic networks consist of multiple lane roadways, we experience traffic congestions on a regular basis. A traffic network gets overwhelmed with vehicles at a certain time period of the day and we, the people in the vehicles, are the ones who feel the impact. I have always seen room for improvement in this field of our lives, even more so today when our traditional solutions are no longer feasible.

In the past, our traditional solution to traffic congestions was to just build larger and more roadways [3]. Today however, this is no longer a viable solution. We already have cities with huge roads consisting of five or more lanes in one direction, and building or adding more to them has a high financial and environmental cost. Therefore people have started asking themselves what we can do to make a better use of our current traffic system.

I feel that automating our traffic system is one of the best solutions to our everyday traffic challenges. A perfect automated vehicular traffic system would provide solution to the following:

- Traffic congestion;

- Traffic accidents;
- Decrease pollution caused by traffic;
- Decrease time spent in traffic;
- Decrease productivity loss.

People have done extensive research on the impact traffic congestions have on productivity in our world. Ashley [3] and Templeton [47] are just two who have quoted numbers of values in the hundreds of billions of dollars nationwide for the US only in terms of productivity loss because of traffic congestions. Therefore, if an automated traffic system could solve our traffic congestion problem this will inherently increase our quality of life in different areas such as productivity and pollution. Idle running cars account for 12-15% of the CO_2 emissions in the US only according to Templeton [47], this is also equal to 50 billion dollars worth of petrol, as well as the always increasing number of collective hours of delay and productivity.

Furthermore, preventing traffic accidents is probably one of the dominant cases in favor of automated vehicular traffic. If an automated traffic system can prevent traffic accidents this would save around 1.2 million lives and around 48 million injuries annually around the world, as well as around 2.3 trillion dollars of accident costs [2, 5, 47].

Imagine a world where traffic safety would be priority number one, there would be no need for traffic police, no driving under influence and by doing so endangering the lives of people, but at the same time a world with more productivity because of less time spent in traffic, less greenhouse gas emissions and no traffic congestions. This world is something I believe is possible to achieve with the technology available today and a smart automated traffic design.

This dissertation is a quest to find an answer to a small part of the problems we face in coming up with an automated vehicular traffic system. In the next sections I give a summary on the methodologies I have employed to come up with ideas and experiments on the subject, as well as a thesis overview of what each chapter presents.

1.2 Methodology

In order to gather extensive knowledge in traffic theory, metrics and variables I took into consideration different traffic engineering literature. Basic information about traffic flow,

density, space-mean speed was found in [22, 31, 46].

Furthermore, I looked into existing ideas and research behind technologies used today to create driverless vehicles such as the ones used in [10, 11, 12, 16, 18, 51, 52]. I also did a review and analysis of existing traffic simulation software, both scientific and commercial [7, 33, 36, 48, 49, 50]. By doing so I found what the main features of traffic simulation software are and compared them to the features I wanted to implement in my traffic simulation application.

Taking into account all the gathered information I was able to design a vehicular traffic simulation application and implement it. I also designed and implemented intelligent agents to use as drivers of vehicles that drive within a simulated traffic network following specific rules.

Additionally, I used my traffic simulation software to do experimental work doing different case studies on various pathfinding algorithms and cost evaluation functions [14, 38, 43]. I ran multiple traffic simulations under different traffic conditions and studied each algorithm and cost function using my intelligent driver agents. I then used the obtained results to analyze the performance of each of them under the different traffic conditions in order to come up with a proposal for a "smart" driver agent that can be used in an automated vehicular traffic system.

1.3 Thesis Overview

The main goal I had set before starting research in this area was to answer the following question, "Is there a viable way to automate vehicular traffic, and by doing so, decrease traffic congestion?" It has been a trend in the last few decades to automate different processes people used to operate before. As already discussed I feel that automating traffic would greatly increase road safety and our quality of life in general. The goal of automatically controlled traffic would be to create a perfect system with no traffic congestion and no traffic accidents. The existing automatic vehicle models propose self-driving vehicles using different sensors that control various vehicle actions [3, 8, 16, 26, 40]. I feel that by combining the technology we have already used to create driverless vehicles with a central control system that has the knowledge of all the vehicles in a traffic network the actual vehicular traffic behaviour within this network could be run much more efficiently.

This dissertation presents a proposal for an intelligent driver agent for autonomous navigation through traffic which has been extensively investigated through study cases by simulating various traffic conditions in an artificial roadway network and studying different path finding algorithms and cost evaluation functions. In Chapter 2 the existing research and existing traffic simulation software is reviewed and the different metrics and ideas are discussed. Chapter 3 presents a general overview of traffic simulation, as well as gives a definition of the various pathfinding algorithms and cost evaluation functions that are studied in my research. Chapter 4 presents a formal definition of the driver agent, the two distinct types of driving agents used in my traffic simulation software, as well as a case study using the different path finding algorithms and basic cost evaluation functions. In Chapter 5 the basic cost evaluation functions used previously are combined into hybrid cost evaluation functions in order to get a more accurate driving force for the pathfinding algorithms to chose the best possible route to the destination point, these hybrid cost evaluation functions are then examined in another case study and the results are presented and analyzed here as well. Chapter 6 presents an analysis of the basic traffic parameters recorded during the conducted case studies, as well as a discussion of how I plan to use these quantifiable metrics in my future work. Chapter 7 concludes by reviewing and summarizing my findings and presenting the future research directions.

It is the theory that decides what can be observed.

Albert Einstein

2 Theory

2.1 Overview

This chapter includes three major components I researched so that I gather enough information to come up with ideas on how to implement my traffic simulation application, as well as how to implement, study and analyze the results of my intelligent driver agent traffic simulations.

In Sect. 2.2 I discuss in details the existing research on autonomous vehicles, various driver assistance systems, as well as different ideas and concepts that are popular today. In Sect. 2.3 I present a review of various scientific and commercial traffic simulation software packages and discuss their strengths and weaknesses. In Sect. 2.4 I present an introduction to the basic traffic system theory including the different metrics and parameters important to traffic engineering experts which will be discussed in more detail in Chapt. 6.

2.2 Existing Research

Existing research and development in the field of autonomous vehicles can be divided in two categories:

- Fully autonomous
- Partially autonomous

The system I worked on is concentrating on fully autonomous vehicles, however studying the literature on these I found that the two categories coincide with each other. They do so because the technology developed for the partially autonomous vehicles is also used in the fully autonomous ones. In the following subsections I present the major ideas and concepts I found compelling, as well as the technology and methods being developed to make these concepts a reality.

2.2.1 Partially Autonomous Vehicles

Partially autonomous vehicles are essentially the vehicles we drive today. Most of the vehicles include numerous assistance systems and technologies such as Anti-lock breaking system (ABS) or Cruise Control system. These systems and technologies make up the so called smart or intelligent car for which the European Commission has launched the Intelligent Car Initiative [16] in February 2006 that sets out Europe's future strategy for development of vehicles that are smarter, safer and cleaner [15].

European Commission's Information Society agree that we have become dependent on transportation in our everyday lives, as well as the fact that increasing road traffic generates serious social problems beginning with congestion of the roadways to damage of the environment and above all traffic accidents. Statistics show that around 40,000 people die every year in traffic accidents around Europe and many more are injured [16].

The Intelligent Car Initiative aims to speed up the development and deployment of smart vehicles in Europe so that the transportation system becomes more convenient for drivers. They seek to introduce numerous technologies in the vehicles to assist the driver in the driving functions, thus preventing, avoiding or mitigating accidents [15]. The driver assistance technologies listed on the Intelligent Car Initiative Website [16] include:

- Anti-lock breaking system (ABS) - a system that prevents the wheels from locking up by automatically modulating the brake pressure when the driver brakes hard.
- Adaptive Cruise Control (ACC) - a system that improves the function of standard cruise control by automatically adjusting the vehicle speed and distance to the vehicle ahead.
- Adaptive headlights - a system that can direct the beams by moving each headlamp left, right, up or down in reaction to steering wheel angle, speed and movement of the vehicle.

- Blind Spot Detection System - a system that continuously monitors the rear blind spots on both sides of the vehicle and warns the driver if changing lanes is not safe at a moment.

- Driver Drowsiness Monitoring - a system that can detect the driver's drowsiness in several ways: by tracking the driver's facial features, movements of hands and feet, by analysing eye-closures and head pose or even changes in heartbeat. The driver is alerted if drowsiness or distraction is detected.

- Electronic Brake Assist System (EBS) - a system that is very efficient aid in emergency braking situations when the driver wants the vehicle to stop as quickly as possible.

- Electronic Stability Control (ESC) - a system that detects the deviation between the vehicle's trajectory and the intended direction. Without any action on the part of the driver, small amounts of braking are applied separately to each wheel and this can bring the vehicle back to the intended course. The driver maintains control of the vehicle and often does not even notice that the Stability Control system has intervened.

- Gear Shift Indicator (GSI) - a system that assists the defensive motoring style of the driver helping him to save fuel and to drive more smoothly and economically. Its role is to give a visual warning when a gear change is necessary.

- Lane Departure Warning System (LDWS) - a system that monitors the position of the vehicle within its lane and warns the driver if the vehicle deviates or is about to deviate from the lane.

- Night Vision - a system that provides increased night-time visibility by far- or near-infrared sensors with a range equivalent to upper beam headlights.

- Obstacle and Collision Warning System - a system that helps the driver prevent or mitigate accidents by detecting vehicles or other obstacles on the road ahead and by warning the driver if a collision becomes imminent.

- Vulnerable Road User Protection System - a system that helps prevent collisions and protect vulnerable road users such as pedestrians and cyclists. (not yet commercially available)

- Tyre Pressure Monitoring System (TPMS) - a system that monitors the air pressure inside a pneumatic tyre.

Undoubtedly having these driver assistance systems installed in every vehicle would make traffic a much safer environment than what it is today. And knowing that all of this technology, with the exception of the Vulnerable Road User Protection System, is already commercially available leads me to believe that we can see them become more and more integrated in every new vehicle we purchase.

Another partial automating vehicle technology was developed in the early 1990s when the U.S. Department of Transportation launched the National Automated Highway System Consortium (NAHSC). The consortium aimed to push the development and deployment of fully automated highway systems (AHS) [8]. These automated highway systems were developed to reduce the following distances between the vehicles by grouping them in so called platoons, thus allowing more vehicles to occupy a given stretch of the road [3, 8].

The foremost plan for the AH system was to build separate highway lanes that will carry the necessary technology to automate the driving of the vehicles in them. When a driver wanted to enter the automated highway lane, the system would interact with the sensors and technology incorporated in the said vehicle and would take over the control. Each vehicle in the automated highway lane would share information among the other vehicles and the AHS. Using this method, researchers working on the proposed system said that it would increase the capacity of a typical highway lane by three times [3].

A prototype of the AHS was tested in August 1997 on a stretch of Interstate 15 near San Diego, California. The experiment was conducted by the NAHSC and it proved to be a success. Ashley [3] reported that the augmented cars ran a total of about 8,000 miles, carried 4,000 passengers, and had no safety incidents.

This sound idea however seems to have been abandoned today. Bishop [6] discusses the different social and political forces that have led to the deteriorating of this project. The main concern was people safety in a fully automated AH system, and therefore a halt has been introduced in the development of this system so that a less futuristic and more marketable solution is found [6].

2.2.2 Fully Autonomous Vehicles

If our technology development keeps progressing with the same pace as in the last few decades, we can say that fully autonomous vehicles are fundamentally the vehicles of the future. A future that we can expect very soon according to car manufacturers [26, 40].

Research in robotics has led to ideas and concepts about self-driving vehicles, so called Robot Cars or shortly Robocars [47]. One of the key development offices for robot vehicles is the Defense Advanced Research Projects Agency (DARPA) established in 1958 by the U.S. Department of Defense [13].

Development in the field of self-driving vehicles has been pushed by organizing experiments and challenges for these robocars so that scientists can compete and earn funding for their future research and development. One such challenge has been organized by DARPA on three occasions in 2004 [10], 2005 [11], and 2007 [12]. The so called DARPA Grand Challenges in 2004 and 2005 were organized in unpopulated areas. Robot vehicles were driving through a course of various obstacles in the Mojave Desert, and while none of the robot vehicles was able to complete the route in 2004, there were five vehicles that did complete the route in 2005, the winner being Stanley developed by Stanford University [41]. The DARPA 2007 Challenge, also known as the DARPA Urban Challenge, was held in an urban environment in Victorville, California. Each robot vehicle had to obey all traffic rules and at the same time adapt and react to other traffic and obstacles located in the environment as well as merge into traffic.

Figure 2.1: DARPA Urban Challenge 2007 robot vehicles. On the left side Tartan Racing Team's robot vehicle called "Boss" first place winner at the challenge. On the right side Stanford Racing Team's robot vehicle called "Junior" second place winner at the challenge. [42, 45]

The challenge was a complete success, with the 60 mile urban area course being completed by six teams, where the winner was a team comprised of Carnegie Mellon University and General Motors called the Tartan Racing [45] team. Fig. 2.1 includes photos of the first and the second place winners robot vehicles of the DARPA Urban Challenge 2007. Interesting facts about the event are discussed in [23, 32], where it says that each robot vehicle showed different personality during the challenge. Some of the vehicles were more aggressive, while some were being safe. Henderson [23] sums it up really well by saying "SUV with mind of

its own wins robot car race".

Another key research lab in this field is The Artificial Vision and Intelligent Systems Laboratory (VisLab) of Parma University in Italy [53]. They have developed automotive technologies and algorithms for fully autonomous vehicle driving. One such vehicle is called BRAiVE (short for BRAin-drIVE) which is presented at their BRAiVE Website [51]. This prototype vehicle is used to test end experiment the technology they are developing at the lab. One of the most important challenges organized by VisLab is the VisLab Intercontinental Autonomous Challenge also known as VIAC [52]. This challenge was launched in July 2010 and is expected to be completed at the end of October 2010. The experiment consists of two self-driving electronic vehicles autonomously driving from Parma, Italy to Shanghai, China. That is a route of 13,000 km to be completed over the course of three months. There is also live video coverage of the event streamed online, where people can view the progress of the robot vehicles.

Additionally, Frog Navigation Systems in the Netherlands has also developed an autonomous vehicle system called FROG (Free Ranging On Grid) [18]. Their technology is comprised of multiple autonomous vehicles and a supervisory central system. Each vehicle is equipped with long range and short range sensors so that it can avoid collisions with other vehicles and obstacles located in the environment, while the supervisory central system is there to direct the vehicles to their destinations. This technology has been originally developed to control machines and robots on the factory floor, however it has also seen use as part of a public transport system operated by 2getthere [1].

Most of the technologies used in these challenges and experiments are available to the general public even today. Therefore, a world with self-driving vehicles is very well within our reach in the next few decades. Johnson [26] reported in 2005, that Vauxhall was planning to include the technology for a partially autonomous Vectra to go on sale after 2010, while in 2008, Squatriglia [40] reported that General Motors had announced to begin testing driverless cars in 2015 and have them on the road by 2018.

2.2.3 Ideas and Concepts

Urban Designer Michael E. Arth proposes a key concept for our future living environments. His paper on New Urbanism and New Pedestrianism [2] presents his vision and idea on how to solve many of our traffic problems today.

Arth proposes a pedestrian-oriented urban design which could eliminate the need for privately owned cars which spend most of their time being parked. He says that if we trade private cars for efficient, zero emission, self-driving public taxis, we can have any type of vehicle we want, when we want it, for the fraction of the cost of owning a private car. This would solve a wide range of problems, starting from global warming to traffic congestions and traffic accidents. He goes further to explain that doubters need to look no further than the existing car sharing programs and GM's 2008 Opel Vectra whose traffic assistance system is reportedly capable of driving itself on the highway. He also says that all this is possible within 20 years [5, 26].

Figure 2.2: Traffic-Busting Network - The idea behind "traffic radar" is simple: Other cars report where they are and how fast theyre going; your car takes that data, figures out where the traffic is, and routes you around it. Imagine youre in the blue car on the left (1). Many of the cars nearby are broadcasting their position and speed to you wirelessly. Slow-moving cars (2) clog the most direct route to your destination (3). Cars on nearby roads are moving a bit faster (4), while others move even more quickly (5). Your navigation computer uses this information to reroute you onto the fastest path (6). [24]

The idea for traffic-busting networks with cars using traffic radars is what motivated my research the most. Ralf Herrtwich, director of vehicle IT research at Daimler Chrysler says that using traffic radars and cars connected through a wireless network will ensure that our vehicles are informed about the road conditions in the traffic network and can relay this information to us. If the vehicles were also able to take control and drive by themselves, then they can use the gathered information to autonomously advance to our destination [24].

If we could combine wireless vehicle communication through traffic radars with the technology used in driverless vehicles in the VIAC or DARPA Challenges and a supervisory

central system such as the one used by FROG AVG Systems, we could very well design a state of the art self sustainable traffic network.

2.3 Existing Simulation Software

In order to investigate the different traffic simulation strategies which will be discussed in further detail in Chapt. 3, as well as study the requirements for a traffic simulation software application, I have reviewed and compared various traffic simulation packages. In this section I present a review of the investigated software applications, their features, differences, positives and negatives, while also discussing some ideas that can be applied to such software applications.

I reviewed the following traffic simulation packages:

1. SUMO - Simulation of Urban Mobility, version 0.10.3

   "**S**imulation of **U**rban **MO**bility" (SUMO) is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks. [7]

2. Quadstone Paramics Modeller, version 6.4.1

   Quadstone Paramics is a modular suite of microscopic simulation tools providing a powerful, integrated platform for modelling a complete range of real world traffic and transportation problems. [36]

3. Treiber's Microsimulation of Road Traffic [49]

   Treiber's Microsimulation is a personal software project created by Martin Treiber and used in his research of traffic dynamics and traffic modelling.

4. Aimsun, version 6.0.4

   Aimsun is a simulation package that integrates three types of transport models: static traffic assignment tools; a mesoscopic simulator; and a microsimulator. [50]

5. Trafficware SimTraffic, version 6 (build 614)

   SimTraffic is a simulation application part of Trafficware's Synchro Studio package. It serves as a traffic simulator for Trafficware's Studio, which also includes traffic lights synchronization application. [48]

6. CORSIM TRAFVU, version 6.1

   CORSIM TRAFVU serves as a traffic simulation viewer and is part of the TSIS CORSIM software package. It provides animation and static graphics of traffic networks, using the CORSIM input and output files created by a licensed user of TSIS CORSIM. [33]

I compared their features with respect to the following criteria:

1. Open Source and Free Use
2. Operating System Portability
3. Package Documentation and User Interface
4. Creating traffic networks and associated vehicle patterns
5. GUI Simulation and quality of graphic representation
6. Simulation output (data and files)
7. Ability to simulate very large traffic networks
8. Additional capabilities
9. CPU and Memory performance

These are explained in more details in the following subsections.

2.3.1 Open Source and Free Use

Out of the six software packages I considered, only two are open source and free to use (SUMO and Treiber's Microsimulation of road traffic), while the other 4 are paid (Paramics Modeller, Aimsun, SimTraffic and CORSIM). While the free packages could be investigated to their full potential, I could only study the demo versions of the four commercial packages. The demo versions had 30 days usage restrictions and/or feature restrictions.

The greatest positive of open source projects is that their source codes are freely available for download and use. The two free traffic simulation packages I considered had their source codes available online with the difference between the two packages being that SUMO is actually an open source project that is being developed by two different institutions, while

Treiber's Microsimulation is a personal software project whose source code has been made available.

The source code is not available for study, modification or research for users of the four commercial software packages (Paramics Modeller, Aimsun, SimTraffic and CORSIM).

One of the most popular features of open source projects is that they can be further modified by other programmers, as already mentioned previously. This feature supports the potential for parallelizing simulation models and packages to explore high end computer systems.

2.3.2 Operating System Portability

Operating System portability is another feature of a software package that is becoming more and more popular with the development of new and improved operating systems. This is also the biggest issue that most of the traffic simulation packages I reviewed face. I have found that only two packages (SUMO and Treiber's Microsimulation of road traffic) have the ability to work under multiple operating systems, including the most commonly used today Microsoft Windows, Linux and Mac OS. While the other four packages (Paramics Modeller, Aimsun, SimTraffic and CORSIM), which are also the commercial ones were only able to work under the Microsoft Windows operating system.

It is interesting to mention that Treiber's Microsimulation avoids the operating system portability problem due to the fact that it is a java applet and it is run within a web browser, therefore no restriction based on operating systems exists.

Additionally concerning the commercial simulation packages it is notable to add that the inability to use these software packages on Linux and Mac OS operating systems reduces and restricts their audience and user pool; more and more so everyday these two operating systems become more popular and used in the world.

2.3.3 Package Documentation and User Interface

In complex software systems such as traffic simulation packages, it is of utmost importance for users to have an easy to understand and self explanatory graphical user interface and user manual.

I was not able to locate a user manual for only one (Aimsun) of the reviewed software packages. However, all the other simulation systems had a well written and easily understandable user manual. The user manuals I compared all included their respective software's specifications, features, and usage explained, which can be very helpful for any new user of the software package.

Concerning the Graphical User Interface (GUI) of the simulation packages, I have found most of them to be really easy to understand and use, especially after following the user manual and the GUI explanations in it. Only one package (Aimsun) had a more challenging GUI to use, but that is mainly due to the fact of not having a manual to run the user through its operation.

It needs to be noted at this point that the GUI of the packages was only reviewed for its easy to follow set up for an early user of the software. Understandably, for an experienced user of a particular software application (no matter how complex its GUI), the software would be easy to use.

2.3.4 Creating Traffic Networks and Associated Vehicle Patterns

While reviewing these traffic simulation applications, I have found different approaches as to means of creation of traffic networks and vehicle movement patterns.

I was able to find a graphical editor for traffic networks for four (SUMO, Paramics Modeller, Aimsun and SimTraffic) of the reviewed applications, while two of them did not have this feature. It is important to mention that the SUMO package did not include its own graphical editor for traffic network, but there is a SUMO Traffic Modeler software application available as an open source application. Additionally, the SimTraffic package purports to include a graphical editor in its full version, while this was not the case for its demo version I reviewed.

Concerning the approaches and ways to create the traffic networks, none were available in the demo versions of SimTraffic and CORSIM TRAFVU, as well as in Treiber's Microsimulation which only simulates predefined scenarios. However, I found the following techniques available under the mentioned software package:

1. SUMO:
    - Manual (by hand) write up of a traffic network in an XML file;

- Import networks created in other traffic simulation applications;
- Using an automatic network generator creating three different types of networks:
  - Grid network
  - Spider network
  - Random network

2. Quadstone Paramics Modeller:

   - Using an automatic network generation wizard (that generated only a simple two zone network in the demo version) - apparently in the full version of the application a user could automatically create a traffic network using the wizard, and then further customize it using the graphical network editor available. However, I was not able to confirm this using only the demo version.

3. Aimsun:

   - Manually drawing of the traffic network using the available graphical network editor.

I also found various techniques to define vehicle movement patterns within the simulation applications, some of which were more user friendly than others. Difference exists between the packages because they use different simulation models (micro or macro model), which I will explain in more detail in Chapt. 3. As an example the SUMO software package is very different in this sense from the other applications because it is the only application where each vehicle (agent) knows its own destination and list of edges it needs to pass until this destination is reached. However, the other software applications included very similar techniques of vehicle movement patterns generation. The following list presents the approaches I acquired under the certain software package:

1. SUMO:

   - Manual (by hand) vehicle route definition using four different types of definitions:
     - Flow definitions;
     - Flow and turning ratios;
     - Origin-Destination (OD) matrices;
     - As well as, the usual for this application, manual edge by edge, vehicle by vehicle XML route definition.

- Import vehicle routes created in other traffic simulation applications;
- Using a random vehicle route generator.

2. Quadstone Paramics Modeller:
   - Applying an OD matrix;
3. Aimsun:
   - Applying OD matrices using zone supply and demand of traffic.
   - The package also included a vehicle route randomization.

I was not able to review any means of creating vehicle routes in the demo versions of SimTraffic and CORSIM TRAFVU, but they should be available in the full versions of the software packages. Additionally, Treiber's Microsimulator includes some kind of statistical distribution of vehicles where the user can basically define the number of vehicles emitted per hour from a certain intersection as well as the ratio between cars and trucks in certain scenarios.

Comparing the results I acquired from investigating the different applications, we can conclude that most of the packages include graphical editor for traffic networks, and also most of the packages use a statistical method for distributing vehicular traffic through their traffic network. While application of OD matrices is the most popular approach to define vehicular routes in these applications, I found this feature to be very difficult to use in some and very easy to use in other software packages.

To further explain the issues I encountered with OD matrix application I will take examples from Paramics Modeller and SUMO packages. In the Paramics Modeller package, OD matrix application is created to be more visual and user friendly, while in the SUMO package OD matrix application is more XML writing and using command line. OD Matrix application in Paramics Modeller is done by editing a simple table. This table, while being edited, has a feature to preview the zone we are editing in the viewing panel, coloring it with specific color and pointing an arrow. This way, a user can very clearly see what area he or she is defining the traffic supply and demand to and from. While on the other hand, the SUMO package OD matrix application was more writing intensive. A user has to create an XML file using a specific pattern, then using command line map the OD matrix to create a trip list which can then be used in the simulation to navigate the vehicles through the traffic network. I found the second approach to be more difficult to use than the one Paramics

Modeller incorporates, mostly because users tend to understand better something they can see like in the Paramics Modeller software rather than something they have to imagine like in the SUMO software package. While an XML can also be very easily readable and understandable it still lacks the visual effect Paramics Modeller has and having to write down each district id, and each source and destination id's without making a mistake can also prove to be a difficult task by itself. Therefore, I found the approach incorporated in the SUMO software package to be more difficult to use and less user friendly.

2.3.5 Graphical Simulation and Quality of Graphical Representation

One of the best ways for a non-technical user to see results of a traffic simulation is to actually view the simulation running in real time. Graphical representation of the traffic simulation is a good method to examine what exactly and at what periods in time something important happens. This method can also help determine if there is a positive or negative effect on traffic flow if something changes during the simulation or there is some kind of disruption in the traffic pattern.

Different simulation packages included different graphical representations of the running simulation, one of the packages (SUMO) also included a non-graphical simulation that only produces output files and statistics by which one could measure and determine what has happened within the traffic network. While all of the software packages included some kind of graphical representation of the traffic simulation there still are major differences between the quality of these representations. The key distinction consisted of the graphics being two or three dimensional. Three (SUMO, CORSIM and Treiber's Microsimulation or road traffic) of the reviewed packages had only two dimensional representation, while the other three packages (Paramics Modeler, Aimsun and SimTraffic) had also an option for a three dimensional simulation.

Concerning the two dimensional preview of the simulation packages I found it to be a regular bird-view of a traffic network with easily identifiable representation of the various vehicles. This is visible in Fig. 2.6 where we can see long and short vehicles simulated within the traffic network, and especially in Fig. 2.7 where we see red rectangles representing the simulated cars and larger black rectangles representing the simulated trucks in the traffic network.

However, there were a few differences between the three dimensional representations of the software packages that included this feature. I found that the Quadstone Paramics Modeller had the best graphics representation of all packages. It included libraries of textures to use for city buildings, vehicles, pedestrians etc. and thus made the simulation look much more realistic than the other packages, as pictured in Fig. 2.3. Paramics Modeller also had the feature for a user to add as many cameras as needed viewing the simulation from different points of reference. The user could actually have views of all points of interest within the traffic network open at the same time while the simulation was running and could very easily examine the pattern of traffic flow through them.

Figure 2.3: Paramics Modeller traffic simulation running, showing three dimensional representation using the available graphics libraries to preview buildings around the traffic network, as well as traffic members such as cars, trucks, public transport and pedestrians [36]

The Aimsun package included a basic three dimensional preview of the simulation. It also had the feature to add as many cameras as needed to capture all the points of interest of the traffic network, as pictured in Fig. 2.4, but did not include as many different textures as the Paramics Modeller. Aimsun's simulation also dropped in frames per second rapidly when we used a three dimensional preview with multiple view points. Further details on CPU and Memory performance and the platform used to review these packages is given in Subsect. 2.3.9.

Trafficware's SimTraffic only included a two dimensional view, but it was mentioned

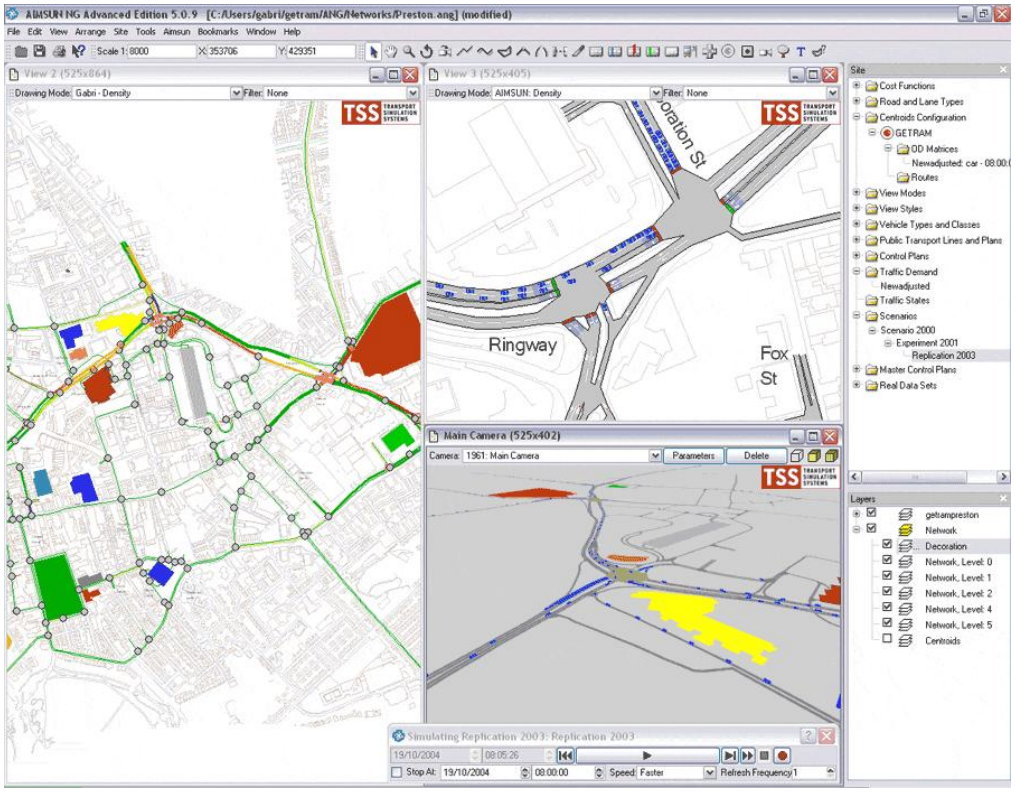

Figure 2.4: Aimsun traffic simulation package, illustrating how multiple views are used to capture different points of interest in a traffic network. Each view point with its own angle, representation and graphics quality settings [50]

in its user manual that the application supports three dimensional view. The explanation given in the manual clarified that the software has the option to save graphics data into an s3d file. This file is said to contain the information needed to display SimTraffic's three dimensional representation in a 3D graphics package. The file also includes vehicle and pedestrian locations at each time slice and geometric data for the simulated traffic network. However, at the present time there is no viewer that supports this file format available.

2.3.6 Simulation Output (Data and Files)

Although following a graphical representation of a traffic simulation can be one of the best methods of determining the traffic flow through a specific traffic network, statistical output always gives additional information that can escape the human eye when viewing a real time execution of a simulation. This is the reason why most of the reviewed software packages included some kind of output files or data. However, I was only able to investigate the output of the SUMO package to its full extent while the demo versions of the other applications

did not include the statistical output, which was said to only be included in the full version of the products. The following list includes the approaches I have found under the different packages:

1. SUMO includes simulation output through generating output files. The output files contain the following:
   - Network state dump of a certain point of time in the simulation;
   - Lane/Edge dump of a certain point of time in the simulation;
   - Detectors dump - detectors are used to locate a point of interest within this simulation package. Using this output file creates a dump of the specific area where the detector is located;
   - Net-wide vehicle emission states and travel times - an output file concerned with the emission states of the vehicles running through the simulated traffic network and their travel times;
   - Vehicle-oriented trip information - file generated to output the route information of the vehicles within the simulated traffic network;
   - Output coupled to traffic lights - output file generated to dump certain actions connected to the traffic lights and how their pattern affected the traffic flow.
2. Quadstone Paramics Modeller includes tools to statistically represent what is happening in the simulated traffic network. The measurements included in the software applications include queuing patterns, peaks and troughs in demand, speed, density, line of sight, journey time, etc. Additionally, all these tools could be used during run time by the user, so he or she could calibrate the model "on the run". [36]
3. Aimsun includes more than 20 different view styles for graphical representation of statistical information about the traffic and the events occurring in the ongoing simulation of the traffic network. [50] However, I was not able to investigate them in the demo version of the software application.
4. Trafficware's SimTraffic seems to have different types of reports available as an output from a traffic network simulation, however none of them is included in the demo version of the application.
5. CORSIM TRAFVU, as already mentioned, is only a graphical player for previewing the simulation generated by the CORSIM package, therefore it did not include any

type of output files. However, it was noted that the full version of TSIS CORSIM package did include some ways of statistical representation and output files of the outcomes from a simulation of a certain traffic network. [33]

I was really surprised to find out that there is no defined file or output standard that all simulation applications use. Looking from a user point of view, it would be very helpful for such a standard to exist because users of different packages can share their results and can compare their simulations using different packages with a consistent output. This is certainly an issue that producers of traffic simulation software need to work and agree on in the future.

2.3.7 Ability to Simulate Very Large Traffic Networks

Traffic simulation software packages are mostly used to simulate and plan changes within a traffic network. Sometimes the traffic network that is being examined may be of a very large scale, like a downtown area of a large city or in some cases even a whole city, as pictured in Fig. 2.5. Therefore, it is important for a package to have the option of simulating very large traffic networks with thousands of roads and millions of vehicles running on them.

Most of the software applications I reviewed had the ability to cope with very large traffic networks. Using the demo versions of the commercial software packages however, I was only able to study the examples that were included with the installed packages. All of the examples were fairly small networks, with a few more complex ones. As I expected, the larger the road network was the slower the traffic simulation ran. Additionally, I expected a simulation to run slower when using more complex graphic models, but instead I just had a lower frame rate while the simulation was running at the certain set speed.

I also studied the SUMO free software package with a large traffic network and I received a similar result. However, it was noted in the user manual of this application that the package could simulate traffic networks of up to 10,000 edges (roads).

2.3.8 Additional Capabilities

Additional features give software packages more useful procedures and components for the user to enjoy. In these applications I have found a good amount of additional capabilities. Starting with examples included with the installation of the software up to the means to

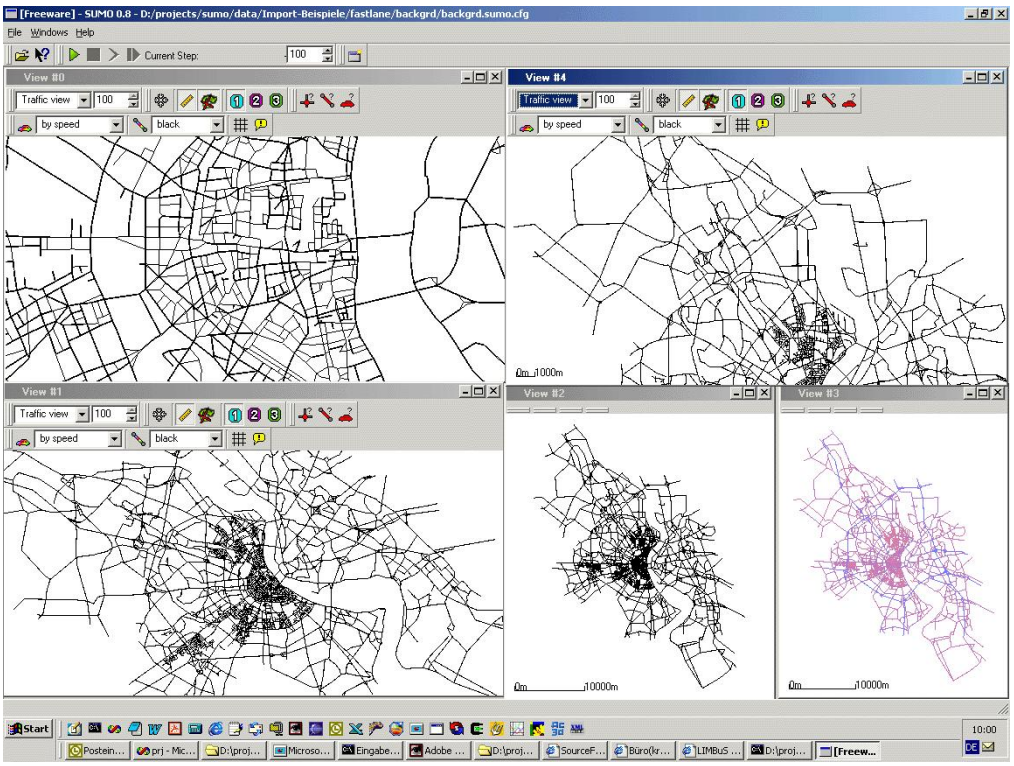

Figure 2.5: A traffic simulation of the city of Cologne conducted in the SUMO software package, illustrating a large traffic network simulation with several thousands of roads and intersections [7]

record a video of a running simulation, these additional features are noticeable enough to provide better quality of the software.

I have found the following additional capabilities in the software applications:

- **Examples** included with installation - this feature was part of all packages with the exception to the Aimsun package. This came to our surprise because Aimsun proved to be the most difficult package to start working, therefore I expected to at least have some kind of example traffic networks to simulate and gather results from. Additionally, examples were not included with the CORSIM TRAFVU package but they were available for download on the website of the software, one example is pictured in Fig. 2.6.

- Means to **convert** other **traffic simulator formats** - I found this feature in two (SUMO and Aimsun) of the reviewed packages, and they could convert traffic networks from most of the other software applications. While the other four packages did not include this feature because of different reasons, we can predict that Treiber's Microsimulator

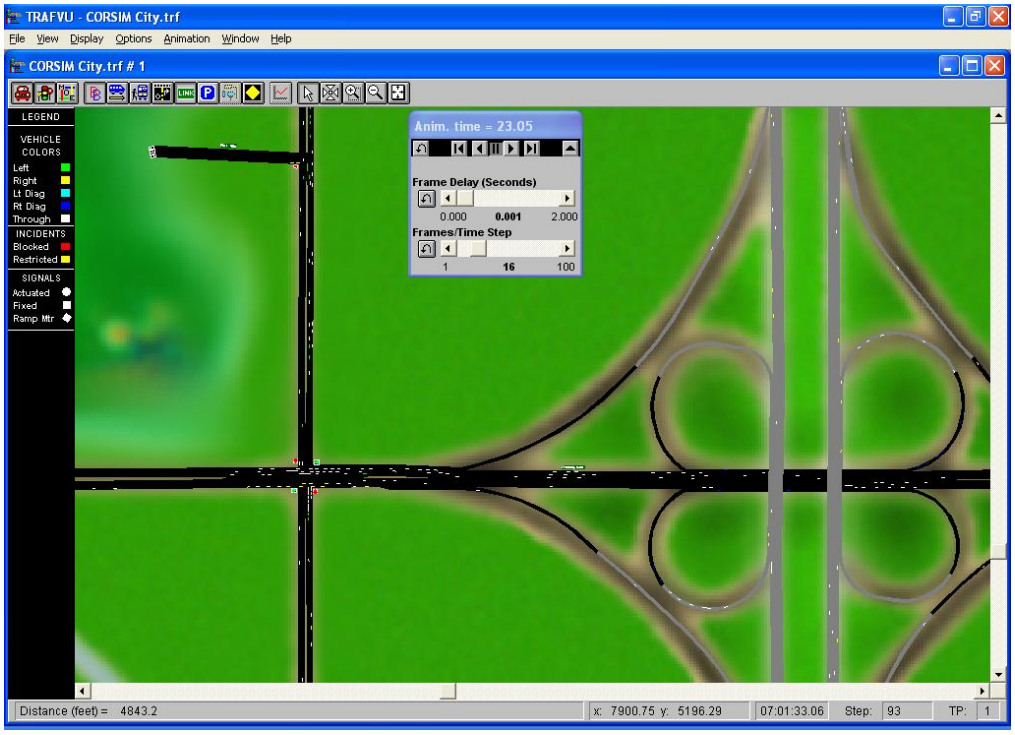

Figure 2.6: CORSIM TRAFVU software package - traffic network simulation of one of the example traffic networks available at the TSIS CORSIM website, the particular one simulating a spaghetti junction example [33]

does not include this feature because its scalability is not near any of the other traffic simulation packages. Also I expected CORSIM TRAFVU to not have this feature because it is only a viewing environment used to preview and graphically represent the simulation already created with the CORSIM software application. But it was a surprise that Paramics Modeller and SimTraffic packages did not include this feature.

- **Different unit systems** and changing between them - maybe not as important to the average user, but still a viable feature for a traffic simulation package which was present in three of the reviewed packages but in different forms. For example the SUMO package only included means to convert the default m/s unit to km/h; Quadstone Paramics and Aimsun on the other hand include three different systems in its package: UK, US and the Metric system. However, the other three applications did not have this feature. I suspect that for SimTraffic and CORSIM this is because I studied only their demo version, while for Treiber's Microsimulation of road traffic this feature is not available due to the simplicity of this software project.

- Simulation of **different vehicle types** - all of the reviewed packages included this

feature, some of them in more complex version while some in a rather simplistic version. To give an example, using the Paramics Modeller I could use a library of different vehicles, but one could also define their own vehicles while inputting the type, size, weight, color and other characteristics of the user-defined vehicle. The simple version of different vehicle types is used in Treiber's Microsimulation which only includes two types of vehicles: cars and trucks, a preview of them seen in Fig. 2.7.

Figure 2.7: Treiber's Microsimulation of Road Traffic - illustrating the use of different types of vehicles used in the simulation: cars and trucks. The cars are represented with smaller rectangles in red color, while the trucks are larger rectangles in black color [49]

- Means to **configure traffic lights** - while all of the packages include some version of configurable traffic lights I found some to be difficult to set up. As an example for difficultly configurable traffic lights I mention the SUMO package because composing the traffic light pattern in SUMO is done manually (by hand) writing into an XML file (as most features of this package). Expectedly, I was not able to study any of the means to configure traffic lights in the demo versions of the commercial software packages, while Treiber's Microsimulator included a traffic lights scenario in its predefined scenarios, but it was not possible to configure the pattern of the lights.

- Means of adding **public transport and public transport stops** - a key factor in city traffic, traffic jams and congestions is undoubtedly public city transport. As such, the feature to add public transport is of great importance for traffic simulation applications. By adding buses, trains, trams etc. to the traffic network a user can see

how they affect the flow through the network and what can be done to improve this flow if it is not at the desired level.

I found four (SUMO, Aimsun, Paramics Modeler and CORSIM) of the reviewed applications to have this feature available. However, I was only able to study it in SUMO and Aimsun, while this feature was not part of the demo versions of Paramics Modeller and CORSIM TRAFVU. Additionally, I was not able to find presence of this feature in the demo version of SimTraffic, while Treiber's Microsimulation did not include a public transport scenario to be studied.

- Means of **vehicle rerouting or closing a street** - a specific feature that can see a lot of use in simulating traffic during road works. Using vehicle rerouting a user can simulate and determine the flow of traffic in specific conditions that require rerouting of vehicles by closing a street.

  Three of the demo versions I reviewed (Paramics Modeler, SimTraffic and CORSIM) did mention to include this feature in the full version of the software package, but I was not able to investigate it. However, I was able to study it in SUMO and Aimsun. The SUMO software application included vehicle rerouting using the manual configuration of rerouting in an XML file; while the Aimsun package included this feature using specific scenario settings.

- Means to **restrict road access depending on vehicle type/class** - very similar to the rerouting feature, but dependent on the vehicle type. Again a feature that can be used to plan future road additions to a traffic network and determine the impact it would have on the traffic flow if different types of vehicles were restricted on not using the new road.

  Comparing the different applications for this feature I found exactly the same result as in the vehicle rerouting feature. Three (Paramics Modeler, SimTraffic and CORSIM) of the demo applications user manuals again noted that this feature was included in their full versions, while the SUMO and Aimsun package had the usual XML configuration and specific scenario configuration of road access for different vehicle types respectively.

- Support for **right and left hand driving** - this feature is very important for broadening the potential user pool of one traffic simulation application. Support for both right and left hand driving ensures the package to be usable by a user from any country in the world, while support for only one side restricts this pool to only part of the countries of the world. That being mentioned, I found only two (Paramics Mod-

eler and Aimsun) of the reviewed packages to include this feature even in their demo versions, while I was not able to locate this feature within SimTraffic and CORSIM. Additionally, I found this feature to be one of the very few that were not available in SUMO, while Treiber's Microsimulation did not include this feature because all the scenarios included in this application contained only one way streets.

- Means to **record a video** of the running simulation - a very useful feature for computer simulation applications when the user needs to have a video as an output of a particular traffic network simulation. However, I only found this feature in two of the reviewed software packages (Paramics Modeller and SimTraffic). While in Paramics Modeller I was able to record a video and encode it using any of the installed codecs on the computer, SimTraffic's video recording was only mentioned as part of the application's full version. The other four software applications do not include means to record a video of the running simulation.

- Means to **print and screenshot** the current frame of the simulation - similar to recording a video feature, but instead just printing or screenshoting a particular frame of the running simulation. This feature was however included in more packages than the video recording. I found it available in four (Paramics Modeller, Aimsun, SimTraffic and CORSIM) of the reviewed packages (even in their demo versions); while this feature was not available in SUMO and Treiber's Microsimulation.

- Simulation of **pedestrians** - this feature is certainly very important in simulating city traffic, especially in areas where there is high density of pedestrians that move either on the sidewalks or cross the streets. I found this feature available in three (Paramics Modeller, Aimsun and SimTraffic) of the reviewed simulation packages, while CORSIM, SUMO and Treiber's Microsimulator packages did not include this feature.

2.3.9 CPU and Memory Performance

I reviewed these traffic simulation software applications using a laptop with Intel Core 2 Duo Extreme processor running at 2.8GHz with 6MB of cache memory and 2GB of RAM memory.

Tab. 2.1 provides information about the CPU and Memory performance of the software applications while they were actively simulating a traffic network.

	SUMO	Paramics Modeller	Treiber's Microsimulation of road traffic	Aimsun	SimTraffic	CORSIM TRAFVU
CPU usage	Between 5-17%, depending on the number of vehicles currently running on the traffic network	Constant 50%	N/A	Between 25-40%, depending on the number of vehicles and the scenario currently simulated	Constant 50%	Constant 50%
Memory usage	Between 12-16 MB, depending on the traffic network	Between 40-140 MB, depending on the traffic network and the graphic models used	N/A	Between 30-40 MB, depending on the traffic network	Around 35 MB, does not depend much on the traffic network simulated	Between 28-32 MB, depending on the traffic network

Table 2.1: CPU and Memory performance of the different traffic simulation software applications while actively simulating a traffic network

I expected the simulations with the best graphics representation to use more memory than the other simulations that used only 2D or poorer 3D graphics. The results present in Tab. 2.1 do confirm my expectations. Concerning the CPU usage, I expected most of the simulators to have 50% CPU usage which would mean that one of the CPU cores was used to its 100% potential, but I found out that only three of the 6 applications were using the CPU to its full extent.

2.4 Traffic Systems Theory

The study and research of traffic engineers greatly helps in determining traffic parameters and measurements that are used for examining the outcome of a traffic simulation. Traffic system experts introduce variety of traffic parameters and quantifiable metrics commonly used in traffic analysis and traffic modelling today. Studying various literature on traffic systems theory I managed to find the needed definitions, equations and relationships between

traffic parameters, which I then investigated in my traffic simulation software.

Basic traffic system parameters and their corresponding definitions were found in [22, 31, 46]. These are:

- Traffic Flow or Volume - Number of vehicles per unit time
- Speed - Time per unit length
- Concentration or Density - Vehicles per unit length

Using these basic traffic measurements I was able to simulate different traffic conditions and by doing so I could study the travel times of my intelligent driver agents from one point to another under the various conditions, as well as their overall behaviour. I have conducted studies for three different scenarios by controlling the traffic density within the simulated traffic network:

- Low Density Traffic - number of simulated vehicles equal to 10% of the total capacity of the traffic network (see Sect. 6.2)
- Medium Density Traffic - number of simulated vehicles equal to 29% of the total capacity of the traffic network
- High Density Traffic (Rush Hour) - number of vehicles equal to 68% of the total capacity of the traffic network

The case studies, including results about travel times and analysis, are explained in more details in Chapt. 4 and Chapt. 5, while the definitions, equations and relationships of the various traffic measurements and parameters, as well as a study and analysis of results based on my study cases are presented in Chapt. 6.

Life is too short for traffic.

Dan Bellack

3

Traffic

3.1 Simulating Traffic

Computer simulation is more and more popular discipline in the field of science in general. People tackle different scientific problems using computer power through simulating artificial experimenting environments. They use these simulated environments to test scientific models in order to prove or disprove their feasibility and correctness.

With the increase of processing capabilities of personal computers, researchers have gained interest in modelling and simulating environments and areas that a few years ago we believed they could only be simulated in the science fiction movies. Today people use the high machine power to simulate environments at a rate much faster than any real environment, thus any experiment conducted in the simulated medium would provide results minutes, hours, days or sometimes even weeks, months and years ahead of what the same experiment would provide conducted in the real world.

One of the systems that is best studied using a computer simulation is a vehicular traffic network system, where we have many factors coordinating and affecting the conditions within the traffic network as well as the behaviour of the drivers. It is more common to experiment with traffic in a computer simulated environment because experimenting with it in the real world is just not practical.

Traffic systems in terms of a collection of dynamic vehicles arranged on a network of roadways are highly complex systems with a rich range of emergent properties. As I previ-

ously mentioned in Chapt. 2, computer scientists have come up with different strategies to simulate traffic systems. They fall under the following two categories:

1. Macroscopic simulation including: statistical dispersion models [19], freeway traffic model [29] etc.
2. Microscopic simulation including: cellular automata [9], multi-agent simulation [4, 17, 20, 21, 35, 37], particle system simulation [39] etc.

as well as strategies combining these two approaches, such as the ones discussed in [25, 34, 54].

Microscopic simulation involves treating each individual vehicle in the system as an autonomous agent, while macroscopic simulation involves vehicles that are statistically distributed by some kind of emitter object at a certain predefined or dynamic rate. Each model has its own advantages and disadvantages. From the reviewed traffic simulation applications presented in Chapt. 2, I have found that microsimulation models for example can have issues with the use of computer memory, while macrosimulation models did not provide any means to examine impacts a single vehicle had on the traffic network and the flow of other vehicles through the network.

3.2 Vehicle Navigation

Driving in our vehicles everyday, we evaluate different path options to get to our destination. We take all known factors into consideration to find our best route to get from point A to point B spending the least time possible driving. We may know the length and the speed limit of the roads we are planning to drive on, we may even estimate the traffic situation on them using the day and time as a reference, but we can never know the exact properties of every single road option we can take to reach our destination. What if there was a way we could know all the factors of every road we could possibly consider? Different pathfinding algorithms may use this information to control our vehicles and get us to a destination safely and in a timely manner.

Throughout my research, I considered three different pathfinding algorithms:

1. A* [38]

2. D* [28, 43]
3. Dijkstra [14]

and three basic cost evaluation functions:

1. Time Cost Function
2. Length Cost Function
3. Congestion Cost Function

I also went further to combine these cost evaluation functions into hybrid functions, which will be discussed in more detail in Subsect. 3.2.4.

3.2.1 A* Pathfinding

A* algorithm is one of the most used pathfinding algorithms for known environments. It uses two cost functions so that it determines the order in which the search visits the different vertices in a graph (or in our case the intersections within the traffic network).

The two cost functions are denoted:

- $g(x)$ - which is equal to the total cost from the starting vertex to the current vertex.
- $h(x)$ - which is an estimate of the cost from the current vertex to the goal vertex. $h(x)$ always underestimates the cost from the current vertex to the goal vertex so that the algorithm visits the least amount of vertices until it finds the best possible path.

$$f(x) = g(x) + h(x) \tag{3.1}$$

Furthermore, Eq. 3.1 shows the total cost $f(x)$ in terms of $g(x)$ and $h(x)$. A* pathfinding starts with the initial vertex and maintains a list of vertices to be traversed. This list is known as the open list and the vertices in it are prioritized by their total cost $f(x)$ value. The algorithm also maintains a so called closed list where it stores the vertices that have already been evaluated. At every iteration, the algorithm removes the vertex with the lowest total cost from the open list, evaluates it and its neighbours and puts it on the closed list. When this vertex is equal to the goal vertex then the algorithm has successfully found the shortest path. Alg. 1 contains the pseudocode on which my A* pathfinding implementation is based.

Algorithm 1 A* Pathfinding

```
function PathCost(node_start, node_end)
  return Path_(node_start, node_end).GetCost()
function HeuristicCost(node_start, node_end)
  return ImaginaryPath_(node_start, node_end).UnderestimateCost()
function AStar(node_start, node_goal)
  declare OPEN_list[ ]
  declare CLOSED_list[ ]
  OPEN_list.Add(node_start)
  while (OPEN_list ≠ ∅) do
    node_current = OPEN_list.LowestFValueNode()
    CLOSED_list.Add(node_current)
    OPEN_list.Remove(node_current)
    if (node_current = node_goal) then
      break
    end if
    for (node_n ∈ node_current.GetNeighbours()) do
      if (node_n ∉ CLOSED_list) then
        if (node_n ∉ OPEN_list) then
          node_n.Parent = node_current
          g(node_n) = g(node_current) + PathCost(node_current, node_n)
          h(node_n) = HeuristicCost(node_n, node_goal)
          f(node_n) = g(node_n) + h(node_n)
          OPEN_list.Add(node_n)
        else
          newG = g(node_current) + PathCost(node_current, node_n)
          if (g(node_n) >= newG) then
            node_n.Parent = node_current
            g(node_n) = newG
            f(node_n) = g(node_n) + h(node_n)
            OPEN_list.Add(node_n)
          end if
        end if
      end if
    end for
  end while
  declare DestinationPath_list[ ]
  declare PathNode ← CLOSED_list.getLast()
  while (PathNode ≠ node_start) do
    DestinationPath_list.AddRoad(PathNode, PathNode.GetParent())
    PathNode ← PathNode.GetParent()
  end while
  return DestinationPath_list
```

3.2.2 D* Pathfinding

The D* algorithm is a pathfinding algorithm designed for searching an unknown environment. Its name comes from "Dynamic A*" because it has been designed to act like the A* algorithm except that the costs of the edges can change dynamically while the algorithm is still searching. D* was introduced by Anthony Stentz in [43] and it is mainly used as a pathfinding algorithm for robot vehicles searching an unknown environment. Furthermore, Sven Koenig and Maxim Likhachev have come up with an algorithm known as D* Lite, which implements the same navigation strategy as D*, but it is algorithmically different. In their paper [28], they present the D* Lite on which much of their and Stentz's research is based today [27, 30, 44].

Even though in my traffic simulation software every agent knows the environment, as we'll see in Chapt. 4, this environment is always changing and therefore the D* algorithm fit perfectly into this dynamic pathfinding role. I have based my D* implementation on the D* Lite pseudocode in Alg. 2 found in [28].

3.2.3 Dijkstra Pathfinding

Dijkstra algorithm is one of the oldest pathfinding algorithms designed by Edsger Dijkstra and presented in [14]. Dijkstra algorithm is designed to find the path with lowest cost between a given vertex and all the other vertices in a graph. Therefore, it can be very expensive to run on graphs of a large scale both memory and time wise. However, the scale of the graph (traffic network) I simulated in my traffic simulation software was not very large, so Dijkstra fit very well as one of the options to examine in it. However, even if the traffic network was larger and included more intersections and roads, it is possible to implement an extension to the algorithm which will confine the space in which the algorithm can search smaller graphs and then combine the results of the few smaller searches.

The A* pathfinding algorithm, already explained in Subsect. 3.2.1, is essentially based on the Dijkstra algorithm with an added complexity of a heuristic function. What this means is that if we take the pseudocode presented in Alg. 1 and edit the function HeuristicCost($node_{start}$, $node_{end}$) to return 0, we will be running the Dijkstra pathfinding algorithm. This is given in Alg. 3.

Algorithm 2 D* Pathfinding

function CalculateKey(s)
  return $[min(g(s),\ rhs(s)) + h(s_{start},\ s) + k_m;\ min(g(s),\ rhs(s))]$

function Initialize()
  $U \leftarrow 0$
  $k_m \leftarrow 0$
  for $(s \in S)$ **do** $rhs(s) \leftarrow g(s) \leftarrow \infty$
  $rhs(s_{goal}) \leftarrow 0$
  $U.Insert(s_{goal},\ CalculateKey(s_{goal}))$

function UpdateVertex(u)
  if $(u \neq s_{goal})$ **then** $rhs(u) \leftarrow min_{s \in Succ(u)}(c(u,\ s) + g(s))$
  if $(u \in U)$ **then** $U.Remove(u)$
  if $(g(u) \neq rhs(u))$ **then** $U.Insert(u,\ CalculateKey(u))$

function ComputeShortestPath()
  while $(U.TopKey < CalculateKey(s_{start})$ *or* $rhs(s_{start}) \neq g(s_{start}))$ **do**
    $k_{old} \leftarrow U.TopKey()$
    $u \leftarrow U.Pop()$
    if $(k_{old} < CalculateKey(u))$ **then**
      $U.Insert(u,\ CalculateKey(u))$
    else
      if $(g(u) > rhs(u))$ **then**
        $g(u) \leftarrow rhs(u)$
        for $(s \in Pred(u))$**do** $UpdateVertex(s)$
      else
        $g(u) \leftarrow \infty$
        for $(s \in Pred(u) \cup u)$**do** $UpdateVertex(s)$
      end if
    end if
  end while

function Main()
  $s_{last} = s_{start}$
  $Initialize()$
  $ComputeShortestPath()$
  while $(s_{start} \neq s_{goal})$ **do**
    $s_{start} \leftarrow arg\ min_{s \in Succ(s_{start})}(c(s_{start},\ s) + g(s))$
    Move to s_{start}
    Scan the graph for changed edge costs
    if (*any edge costs changed*) **then**
      $k_m = k_m + h(s_{last},\ s_{start})$
      $s_{last} = s_{start}$
      for (*all directed edges* $(u,\ v)$ *with changed costs*) **do**
        Update the edge cost $c(u,\ v)$
        $UpdateVertex(u)$
      end for
      $ComputeShortestPath()$
    end if
  end while

Algorithm 3 Dijkstra Pathfinding - If we edit the HeuristicCost function in Alg. 1, we will be running Dijkstra Pathfinding Algorithm. This is the edited HeuristicCost function.

function HeuristicCost($node_{start}$, $node_{end}$)
  return 0

3.2.4 Cost Evaluation Functions

Cost evaluation functions are essentially the functions that determine the cost of taking a certain path in a graph. In my case, the cost evaluation functions I considered give the pathfinding algorithm means to control the path my intelligent driver agents take from their starting to their destination point.

As already mentioned, I have considered two types of cost functions:

- Basic Cost Functions, and
- Hybrid Cost Functions.

I have designed a set of rules for each cost function to calculate the cost of taking a particular road in the traffic network. There are two types of costs each function has to calculate, one is the actual cost and the other one is the heuristic cost. While the actual cost of taking a road is very straight forward, as described in the previous subsections the heuristic cost is always an underestimation of the future actual cost, so in accordance with that notion I have defined the following set of rules that apply to the basic cost evaluation functions:

1. Time cost evaluation function:
   - Actual: the time needed to travel the length of the road defined in Eq. 3.2;

$$\text{time cost} = \frac{\text{length of the road}}{\text{speed limit of the road}} \tag{3.2}$$

   - Heuristic: the time needed to travel the Euclidean distance from the current node to the destination node using the speed limit of the fastest driving road in the traffic network.
2. Length cost evaluation function:
   - Actual: the length of the road;

- Heuristic: the Euclidean distance from the current node to the destination node.

3. Congestion cost evaluation function:

   - Actual: the congestion of the road defined in Eq. 3.3;

$$\text{congestion cost} = \frac{\text{number of vehicles on the road} \times \text{length of a vehicle}}{\text{length of the road}} \tag{3.3}$$

   - Heuristic: the congestion of the least congested road in the traffic network.

It is important to note at this point, that taking the Euclidean distance between the current node and the goal node is the best way to underestimate the length cost between the two points. We basically "imagine" that there exists a road connecting these two nodes and take its length as the heuristic cost. If there is in fact a road that connects those two nodes than the heuristic cost will be equal to the actual cost. Accordingly, to find an admissible heuristic for the time cost function, I had to find the speed limit of the fastest driving road within the traffic network together with the Euclidean distance between the current and the goal node and plug them into Eq. 3.2 to calculate the time heuristic cost from the current node to the goal node. Lastly, to underestimate the heuristic cost for the congestion function between the current node and the goal node I just took the value of the least congested road within the traffic network.

Furthermore, I considered all possible combinations of these basic cost evaluation functions to come up with the four hybrid cost functions. These will be discussed in more detail in Chapt. 5:

1. Time and Length hybrid cost evaluation function. Referred to as "TL" in data tables.
2. Time and Congestion hybrid cost evaluation function. Referred to as "TC" in data tables.
3. Length and Congestion hybrid cost evaluation function. Referred to as "LC" in data tables.
4. Time, Length and Congestion hybrid cost evaluation function. Referred to as "All" in data tables.

In the following chapters I will present a formal definition of my intelligent driver agent, and I present the results of the case studies from the traffic simulations using the pathfinding

algorithms and cost evaluation functions described in this chapter. In Chapt. 4 the case study is conducted using the basic cost functions, while in Chapt. 5 the hybrid cost functions are defined, then they are used to conduct a case study and finally a comparison and analysis between the two case studies is presented.

Everything in life is somewhere else, and you get there in a car.

Elwyn Brooks White

4

The Agent

4.1 Traffic Network

Taking into consideration the existing research and traffic simulation software I discussed in Chapt. 2, as well as the simulation models discussed in Chapt. 3, I designed a traffic simulation application and used it to study the intelligent driver agents behaviour. In order to investigate the behaviour of each driver agent exclusively depending on the pathfinding algorithm and cost evaluation function combination it was necessary to use a microsimulation model and treat each agent as a separate entity. In this chapter I present a formal definition of my intelligent driver agent, the set of rules each agent follows, as well as the case study conducted using the basic cost evaluation functions.

Using the tools available in my traffic simulation software I created a grid traffic network shown in Fig. 4.1. Each road in the network has 4 lanes in one direction which is better visible in Fig. 4.15. Additionally, each road is defined with a specific speed limit. The speed limits I used for this traffic network are 50 km/h and 100 km/h randomly distributed over the roads comprising the traffic network. I implemented a primitive traffic light system because this was not the focus of my research at the moment. The traffic lights I used while doing my case studies were set on 15 s intervals and were admitting vehicles from only one of the roads coming to an intersection.

4.2 The Agent (Vehicle)

In order to statistically measure the performance of each pathfinding algorithm and cost evaluation function combination under various traffic conditions I have created two types of intelligent agents:

- Random Driver Agent
- "Smart" Driver Agent

I implemented the "Smart" Driver Agents to use the algorithm and cost function I was investigating over the course of the case study. However, the Random Driver Agent was implemented to create the necessary traffic density within the traffic network. I will discuss the different types of driver agents in more detail in the next two subsections.

Each agent in my traffic simulation software is presented as a vehicle with the following set of characteristics:

- Vehicle Length - Currently set to 4 m
- Vehicle Colour - Currently used to identify random and "smart" driver agents as shown in Fig. 4.1, 4.8, and 4.15. (black colour is used for random agents, while white is used for smart agents)
- Vehicle Maximum Speed - Currently set to 215 km/h
- Vehicle Acceleration - Currently set to 3 m/s^2
- Vehicle Breaking - Currently set to -10 m/s^2
- Driver Temper - Currently used to determine how the driver agent adjusts to the speed limit of the road it is travelling on.

These characteristics were shared among the two types of agents because they describe common attributes of the vehicle and the driver, however each distinct type has its own specific set as we will see further. I implemented the characteristics about vehicle length, maximum speed, acceleration and breaking so that the driver agent model is easily extendable to different vehicle types. By just modifying these values we can differentiate

the behaviour of the agent and we can create a more complex environment by studying a vehicular traffic model that includes different vehicle types.

Furthermore, I created the driver temper attribute to add realism to the traffic simulation. In the real world we have drivers that follow the speed limit of a road, while others drive above it. I wanted to portray this into my traffic simulation and introducing a random variable that determines the behaviour of the driver towards the speed limit was the most logical way to do it. Lastly, while I used the vehicle colour property to identify random from "smart" driver agents using black and white colour respectively, I also used it to show the driver temper by multiplying the red colour component by the driver temper attribute. Using this way I could see the most reckless drivers with a bright red colour, the ones in the middle with an orange colour, and the least reckless with a yellow colour. I only used this technique to visually examine the distribution of different driver temper values among the simulated agents, however this technique can be used to control specific settings and traffic scenarios in my future research.

In addition to these shared attributes among the two agent types, I have also implemented two common agent methods. One method is used for lane changing of the driver agents, while the other one is used for regulating the speed of the driver agents.

The lane changing method takes into consideration the current vehicle speed, the current speed of the vehicle in front as well as the distance between the two vehicles. If the speed of the vehicle behind is faster than the speed of the vehicle in front, and the distance between the vehicles is close to the length of the vehicle then the faster vehicle moves into the upper lane so it can outrun the slower vehicle. In the case that both vehicles are in the fastest lane the faster vehicle slows down to the speed of the slower vehicle in front so that no traffic accident occurs.

The speed regulating method accelerates and decelerates the speed of the vehicles according to a set of rules defined using the driver temper attribute. The way this attribute adjusts the speed of the vehicle is by setting a certain speed I call the target speed. This target speed is calculated using the function presented in Alg. 4. I believe that using this distribution was a fair way of allocating target speeds to portray a somewhat realistic driver behaviour of the real world traffic. Once more in-depth research is done on this matter the coefficients and percentages can be modified to represent the real traffic drivers' behaviour. However, the difference between the random and "smart" driver agents' usage of this method is in their starting speed. While the random agent is injected into the traffic

network immediatelly travelling at its target speed, the "smart" driver agent originates at its starting point with a speed equal to zero and then has to accelerate to its target speed. This phenomena also happens at intersections for both driver agents. Once a traffic light turns red, both driver agent types have to decelerate to a complete stop, as well as accelerate to their target speeds once the traffic light switches back to green.

4.2.1 Random Agent

As already mentioned in the previous section, the purpose of the Random Driver Agent is to provide the necessary traffic density within the simulated traffic network. For this reason, I needed this type of agent to stay within the network during the whole course of one traffic scenario simulation. Therefore, I designed the Random Driver Agents to not have a destination point, but whenever one reaches an intersection it chooses the next road randomly from the ones connected to that intersection.

Algorithm 4 Target Speed Calculation

function TargetSpeed(*SpeedLimit*, *DriverTemper*)
  if (*DriverTemper* > 0.00 AND *DriverTemper* <= 0.10) **then**
    TargetSpeed = random speed between 82% - 97% of *SpeedLimit*
  else if (*DriverTemper* > 0.10 AND *DriverTemper* <= 0.25) **then**
    TargetSpeed = random speed between 85% - 94% of *SpeedLimit*
  else if (*DriverTemper* > 0.25 AND *DriverTemper* <= 0.75) **then**
    TargetSpeed = random speed between 90% - 100% of *SpeedLimit*
  else if (*DriverTemper* > 0.75 AND *DriverTemper* <= 0.90) **then**
    TargetSpeed = random speed between 94% - 103% of *SpeedLimit*
  else if (*DriverTemper* > 0.90 AND *DriverTemper* <= 1.00) **then**
    TargetSpeed = random speed between 97% - 112% of *SpeedLimit*
  end if
  return *TargetSpeed*

4.2.2 "Smart" Agent

The "Smart" Driver Agents in my traffic simulation software are different from the Random Driver Agents by the fact that they have a certain starting and destination point. These agents start their trip from a certain node in the traffic network and follow a specific route to their destination node. The route they follow is determined by the pathfinding algorithm and cost evaluation function chosen by the user during runtime of the simulation. Once a "Smart" Driver Agent reaches its destination it is removed from the traffic network.

Another specific difference these agents have to the random driver agents is that over the course of the simulation the travel times of the "smart" agents were being recorded. I then used these results to evaluate the performance of the different algorithm and cost function combinations in the various traffic scenarios.

One other very important characteristic of the "smart" driver agents is that their pathfinding is implemented to be dynamic. What this means is that each agent recalculates the best path from its current node to the destination node every time it arrives at a new node in the network. This dynamic property is particularly important in the case when I studied the congestion cost evaluation function because the congestion of the roadways in the traffic network was constantly changing, therefore every time one of my "smart" agents came across an intersection it could re-evaluate the route it has chosen and could follow a different one if it was more favourable.

4.3 Case Study

In the previous section I presented a formal definition of the intelligent driver agent and the two different types I designed. I have employed these agents in my traffic simulation software to study the impact of different pathfinding algorithms and cost evaluation functions on the travel time of the vehicles from one point to another in various traffic conditions.

In this section I will present a case study conducted using the three different pathfinding algorithms and the basic cost evaluation functions, both of which I discussed in the previous chapter. The case study includes simulation under the three distinct traffic conditions (traffic scenarios) discussed in Chapt. 2:

- Low Density Traffic
- Medium Density Traffic
- High Density Traffic (Rush Hour)

In order to accurately measure the performance of each algorithm and cost function combination, I have conducted 1,000 "smart" vehicle simulations through each traffic scenario to the total of 27,000 "smart" vehicle simulations and recorded the time needed for each of them to reach their destination. My traffic simulation software records the travel times in milliseconds, however for simplicity I have rounded these times to seconds in this and

the next chapter. The more accurate data tables are included in Appx. A. In the following subsections I will present the gathered data and discuss the results in more detail.

Additionally, the presented analysis follow a specific paragraph pattern which is there for consistency and comparison purposes between the different traffic scenarios. Initially, the scatterplots containing the travel time data for each algorithm and cost evaluation function combination are analyzed. Second, the statistical results shown in the tables are analyzed both by algorithm and by cost evaluation function performance. And lastly, the distribution histograms of all cases are discussed.

All the graphs use the following legend:

Algorithm	Symbol	Color
A*	+	red
D*	+	green
Dijkstra	+	blue

Table 4.1: Legend used in our graphs

4.3.1 Low Density Traffic

Low Density Traffic is defined to simulate number of vehicles equal to 10% of the total capacity of the traffic network. We can see the Low Density Traffic scenario being simulated in my traffic simulation software in Fig. 4.1.

My findings in this scenario indicate that my implementation of the D* algorithm copes better in these conditions, especially using the time cost evaluation function where I had the best average travel time with a value of 167 seconds needed to reach the destination. We can detect this result by referring to Fig. 4.2, where we can clearly see that D* outperforms A* and especially Dijkstra when using the time cost function. Furthermore, in Fig. 4.3, we cannot say for certain which algorithm outperforms the others using the length cost function. We can however determine that there is a traffic congestion caused in the case of A* algorithm because there is a slight increase of the travel time of each vehicle after the 450th simulated vehicle approximately. Referring to Fig. 4.4 we can see that the performance of all algorithms is very close together and we cannot determine and be sure that one should be favoured over the other in this case.

Before I induced the studies on the results I expected to see the greatest variation in the results obtained using the congestion cost function because my “smart” vehicles follow

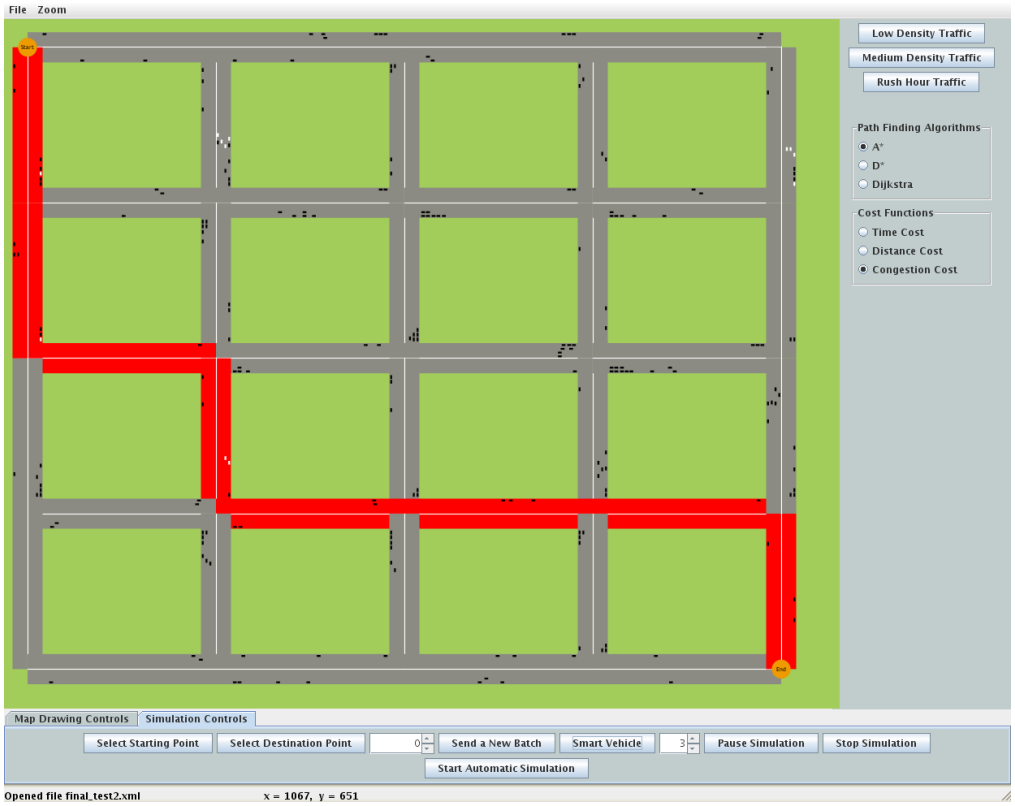

Figure 4.1: Low Density Traffic Scenario being simulated in my traffic simulation software - We can see the starting and destination points for the "smart" driver agents at each corner of the traffic network marked with an orange circle, the best path found at the moment marked with a red trail, the traffic scenario control panel together with the path finding algorithm and cost function option panel on the right side, as well as the simulation control panel at the bottom of the application window. The random driver agents are marked with black rectangles, while the "smart" driver agents with a white rectangle.

	A*	D*	Dijkstra
Mean	194 s	167 s	215 s
Standard Deviation	24 s	23 s	27 s
Coefficient of Variation	12.19 %	13.83 %	12.48 %

Table 4.2: Low Density Traffic using Time Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	203 s	189 s	186 s
Standard Deviation	35 s	28 s	26 s
Coefficient of Variation	17.48 %	14.61 %	13.70 %

Table 4.3: Low Density Traffic using Length Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	192 s	201 s	211 s
Standard Deviation	40 s	42 s	49 s
Coefficient of Variation	20.73 %	20.78 %	23.46 %

Table 4.4: Low Density Traffic using Congestion Cost Function Statistical Results

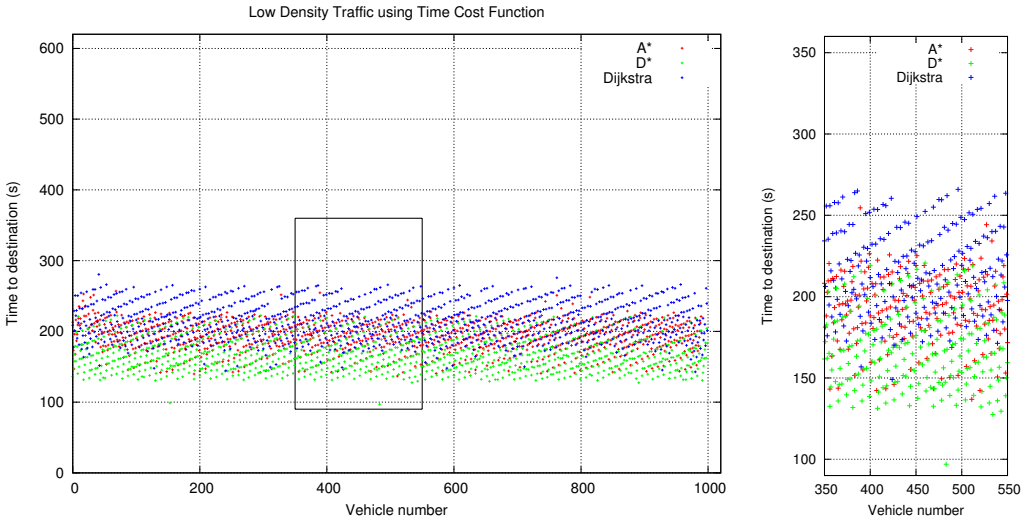

Figure 4.2: Low Density Traffic using Time Cost Evaluation Function Scatterplot - here we can see the travel times of all 3,000 "smart" vehicles simulated in the low density traffic scenario using the time cost evaluation function. In the image on the right side we see in more detail the boxed part from the graph on the left. We can notice a pattern in the gathered travel times. This pattern is due to the traffic light timings and the interval at which the "smart" vehicles enter the traffic network. It is also visible in the graph on the left that D* algorithm performs much better than the other two algorithms using the time cost evaluation function under these traffic conditions. For the graph key refer to Tab. 4.1

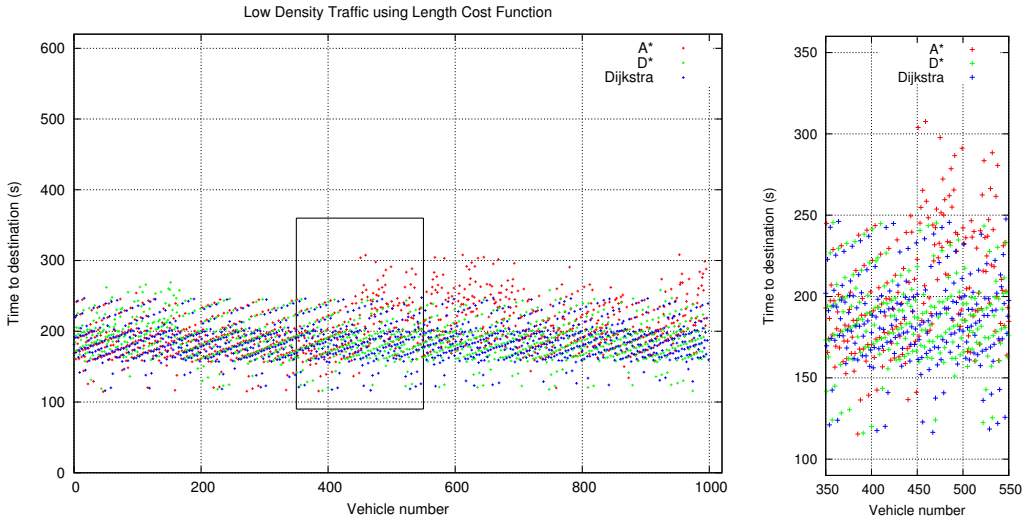

Figure 4.3: Low Density Traffic using Length Cost Evaluation Function Scatterplot - in this scatterplot we can see the travel times of 3,000 simulated vehicles in low density traffic scenarios using the length cost evaluation function. A pattern of the gathered results is again visible in this case, but we can also notice a disturbance in the results when using the A* algorithm (especially in greater detail in the image on the right), this is due to the fact that A* causes a minor traffic congestion at around the 450th simulated vehicle and therefore the travel times increase at this point. For the graph key refer to Tab. 4.1

different path to their destination every time, while using time and length cost evaluation functions the path is always the same and the variation in the results in these cases comes from traffic light timing and waiting times of the agents. Referring to Tab. 4.2, 4.3, and 4.4 where we find the mean, standard deviation and coefficient of variation results for each scenario we can see that I have in fact obtained the expected results.

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	192 s	193 s	201 s

Table 4.5: Low Density Traffic Overall Statistical Results by Cost Function

Comparing the overall mean values of the different cost evaluation functions in Tab. 4.5, we see a very close performance of the time and length cost evaluation functions with mean values of 192 s and 193 s needed to reach the destination respectively as opposed to the value of 201 s we find using the congestion cost evaluation function. I predicted I will find my time and length cost evaluation functions performing better than my congestion cost function under Low Density Traffic conditions because there are not many random agents in the traffic network to congest the roads for my "smart" agents. Inspecting the results from my simulation under these conditions we can clearly see that I have obtained the expected

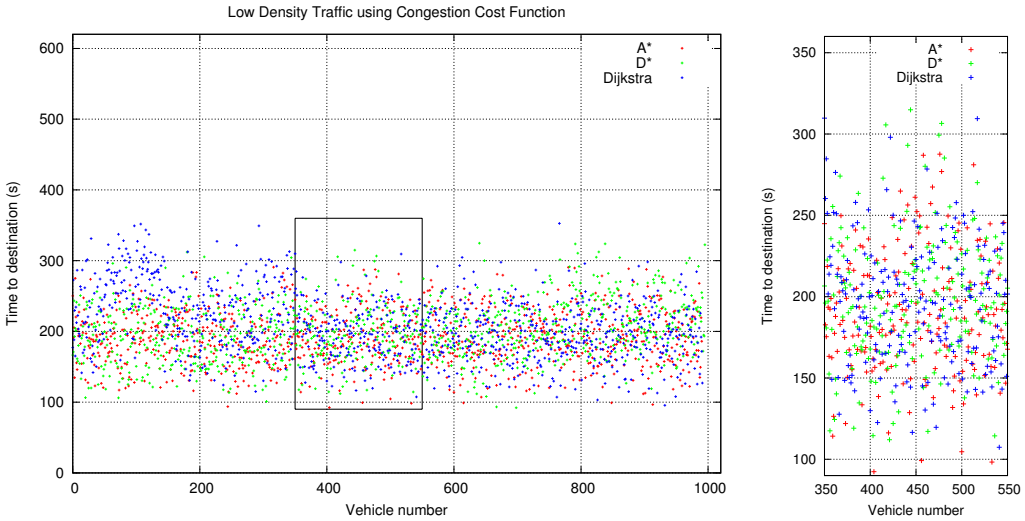

Figure 4.4: Low Density Traffic using Congestion Cost Evaluation Function Scatterplot - we see a scatterplot of the travel times of 3,000 simulated "smart" vehicles through a low density traffic scenario using the congestion cost evaluation function. It is interesting in this case that the travel times of all the vehicles vary a lot because each one of them takes on a different route to the destination due to the fact that in low density traffic scenario the congestion of the roads does not differ much among the different roads. Therefore this is the case with the greatest coefficient of variation under the low density traffic scenario. We can notice in this graph that there is no visible pattern in the results as we had in the previous scatterplots. For the graph key refer to Tab. 4.1

outcome.

Furthermore in Fig. 4.5, 4.6 and 4.7, we can see the distribution of the results gathered while simulating the Low Density Traffic scenario. We can generally notice a normal distribution pattern in most cases, but with slight disturbances depending on the coefficient of variation of the results. As seen also in Tab. 4.2, we have the narrowest distribution in our results when the time cost evaluation function is used, where we also find the lowest mean travel time when using the D* pathfinding algorithm. It is very interesting to see the distribution results using the congestion cost evaluation function under this scenario in Fig. 4.7, where we can see that all the algorithms behave very similarly proven by their distribution curves almost overlapping each other.

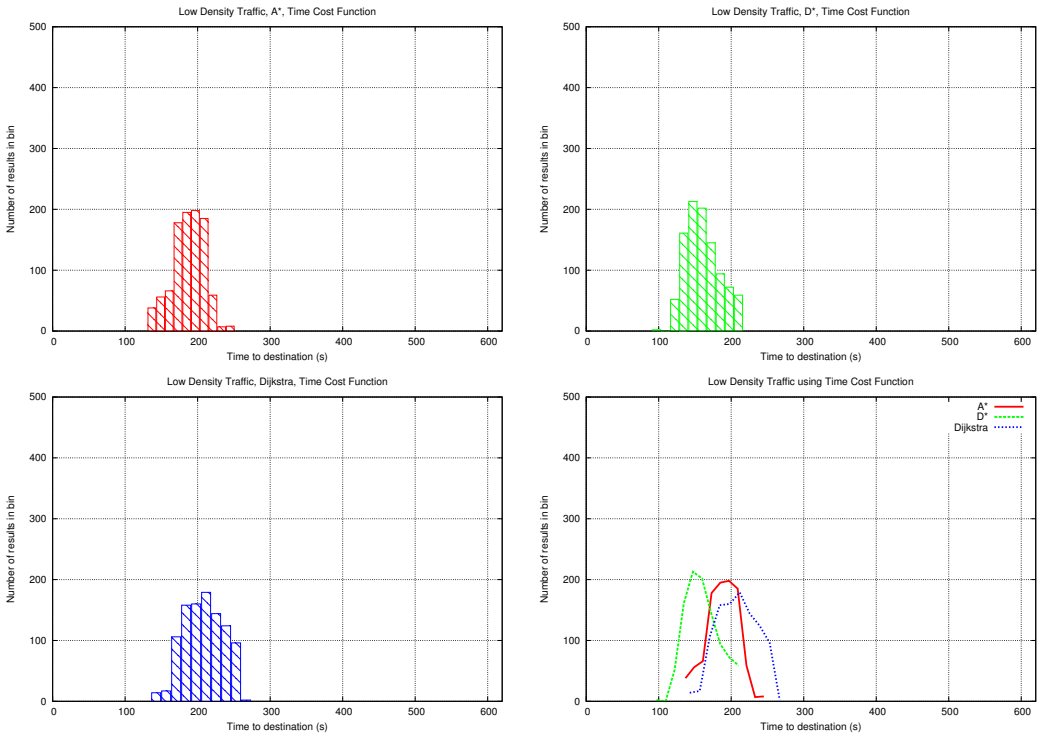

Figure 4.5: Low Density Traffic using Time Cost Evaluation Function Distribution Histograms - in this figure we can see the distribution histograms of the different algorithms using the time cost evaluation function in the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this particular case we can see that there is a clear difference between the performance of the different algorithms and we can easily say that D* outperforms A* and Dijkstra algorithms in this traffic scenario. For the graph key refer to Tab. 4.1

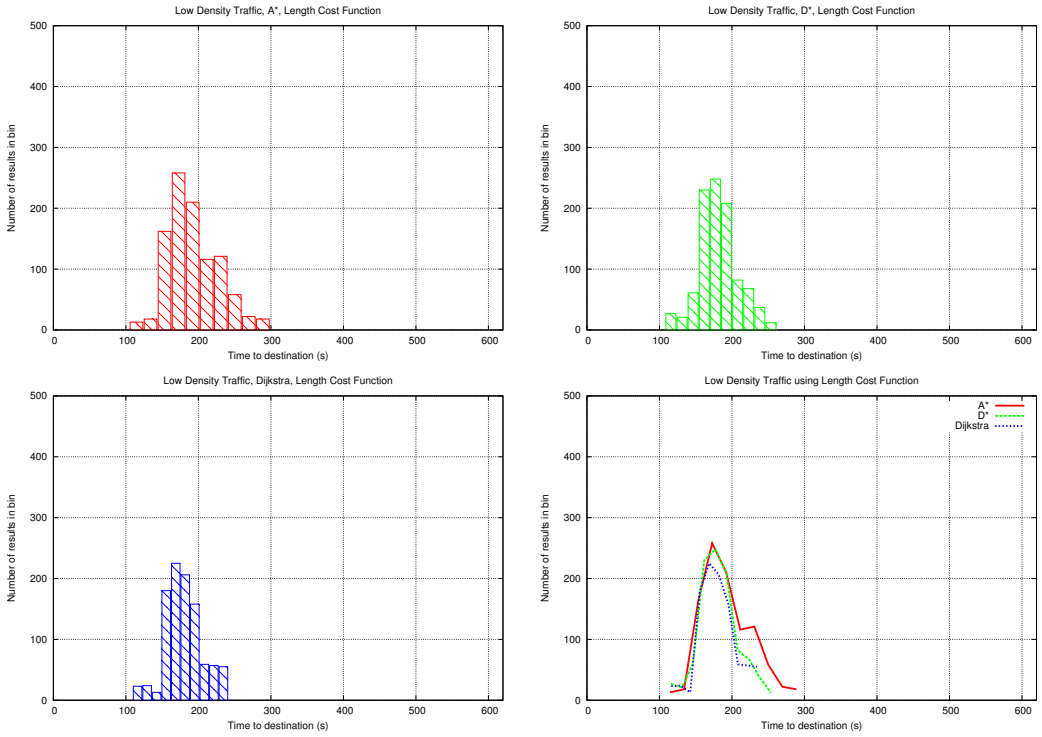

Figure 4.6: Low Density Traffic using Length Cost Evaluation Function Distribution Histograms - here we can see the distribution histograms of the different algorithms using the length cost evaluation function under low density traffic conditions. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this traffic scenario we see a fairly similar distribution using all of the algorithms except in the case of A* where we previously saw a minor traffic congestion occurring which leads to the slightly wider distribution than the other two algorithms. For the graph key refer to Tab. 4.1

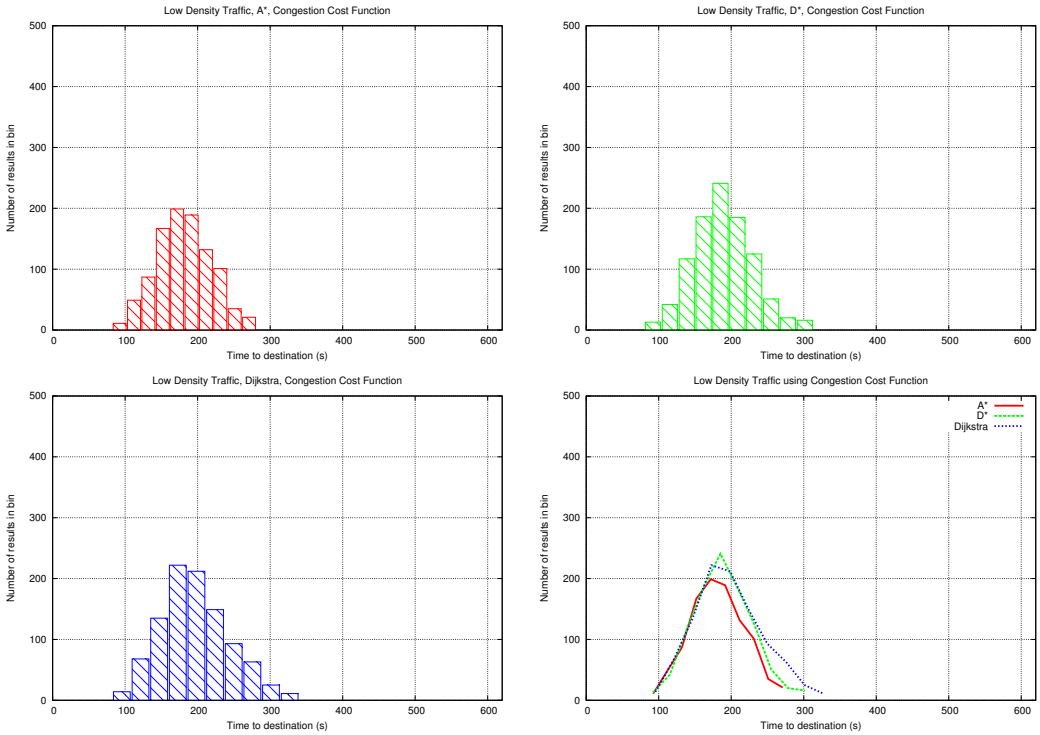

Figure 4.7: Low Density Traffic using Congestion Cost Evaluation Function Distribution Histograms - in these graphs we can see the distribution histograms of the path finding algorithms using the congestion cost evaluation function in the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. It is very interesting to see that even though the distribution in this case is much wider than the distribution when using the other cost evaluation functions all of our algorithms perform fairly similarly which is visible in their distribution histograms. For the graph key refer to Tab. 4.1

4.3.2 Medium Density Traffic

Medium Density Traffic in my traffic simulation application is defined to simulate number of vehicles equal to 29% of the total capacity of the traffic network. Fig. 4.8 shows this traffic scenario being simulated in my software.

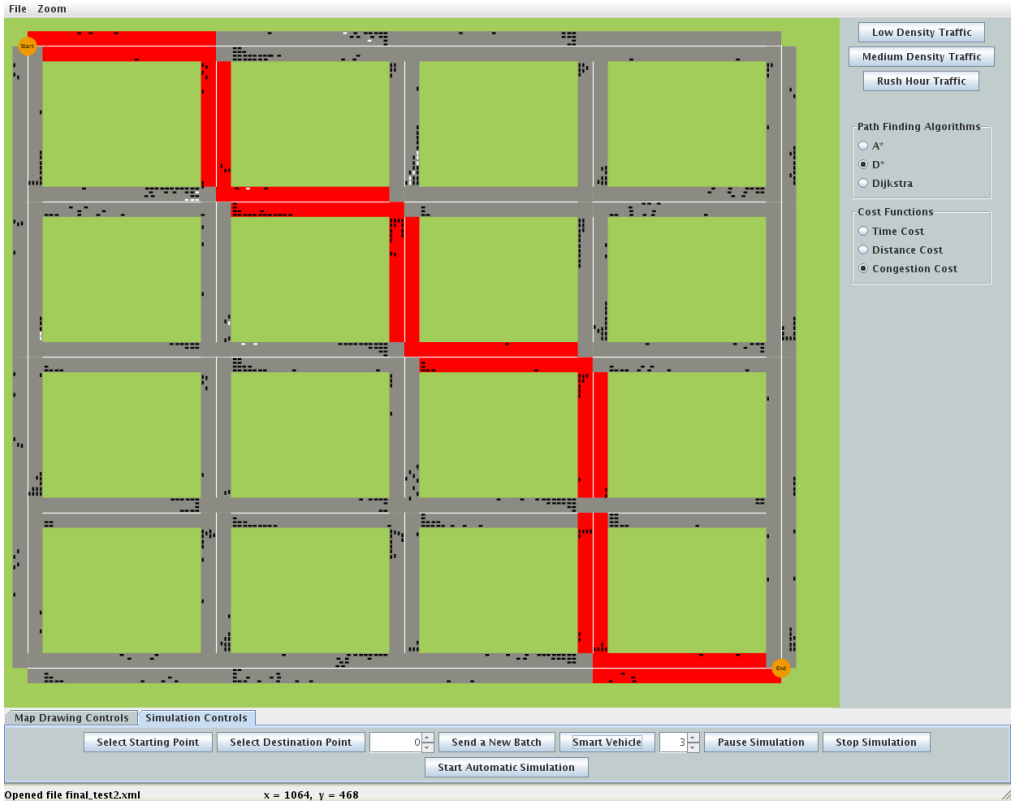

Figure 4.8: Medium Density Traffic Scenario being simulated in my traffic simulation software - In this figure we see the medium density traffic scenario (number of random vehicles simulated is equal to 29% of the total capacity of the traffic network.) We see the starting and destination nodes of the "smart" vehicles at the left top and right bottom corners of the traffic network respectively, as well as the best route between the two nodes at the moment marked with a red color. The black rectangles driving on the roads represent the random driver agents, while the white rectangles are the "smart" driver agents. Additionally there are different control panels located on the right and the bottom of the application window each one controlling different settings within the simulation application.

The results obtained in this scenario, show that my implementation of the D* algorithm performs better under these conditions. D* using the time cost evaluation function once again proved to be the best combination to use with the average travel time of 196 seconds needed to reach the destination. Just as in the case of Low Density Traffic, we can clearly see that D* outperforms the other two algorithms when looking at the scatterplot of the

algorithms by a specific cost evaluation function. This is shown in Fig. 4.9. While from these results we can conclude that D* using time cost evaluation function is the best combination in Medium Density Traffic conditions, we can observe in this scenario that the mean values obtained by the different algorithm and cost function combinations are closer to each other which is a very interesting phenomenon. This is especially true in the case when the algorithms are using my implementation of the congestion cost evaluation function, where we see approximately the same mean values with the greatest difference between the means is only 1 s, while in the other cost function cases we have 35 s and 6 s being the greatest differences when using time and length cost evaluation functions respectively. If we refer to the next two scatterplots in Fig. 4.10 and 4.11, we can see how closely the three algorithms perform in these conditions using the length and congestion cost evaluation functions, which again supports the fact that under this traffic scenario these two cost functions perform very similarly.

	A*	D*	Dijkstra
Mean	227 s	196 s	231 s
Standard Deviation	29 s	33 s	29 s
Coefficient of Variation	12.65 %	16.60 %	12.59 %

Table 4.6: Medium Density Traffic using Time Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	210 s	206 s	204 s
Standard Deviation	29 s	26 s	26 s
Coefficient of Variation	13.57 %	12.63 %	12.84 %

Table 4.7: Medium Density Traffic using Length Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	210 s	210 s	209 s
Standard Deviation	40 s	42 s	41 s
Coefficient of Variation	18.99 %	19.94 %	19.41 %

Table 4.8: Medium Density Traffic using Congestion Cost Function Statistical Results

Similarly to the Low Density Traffic scenario, I again expected to see the greatest variation in the results obtained using the congestion cost evaluation function. However in this case the variation is not as ample as in the previous scenario. I predict this is due to the fact that in the Low Density Traffic scenario the congestion percentage of the roads is so small and their difference is very little that each intelligent agent takes a totally random route to the destination, while in the case of Medium Density Traffic, the roads are more congested and each agent routes specifically through a less crowded area in order to get to the destination faster by lowering waiting times at intersections. This way we have a denser distribution of results and therefore the coefficients of variation are in the 19th percentile,

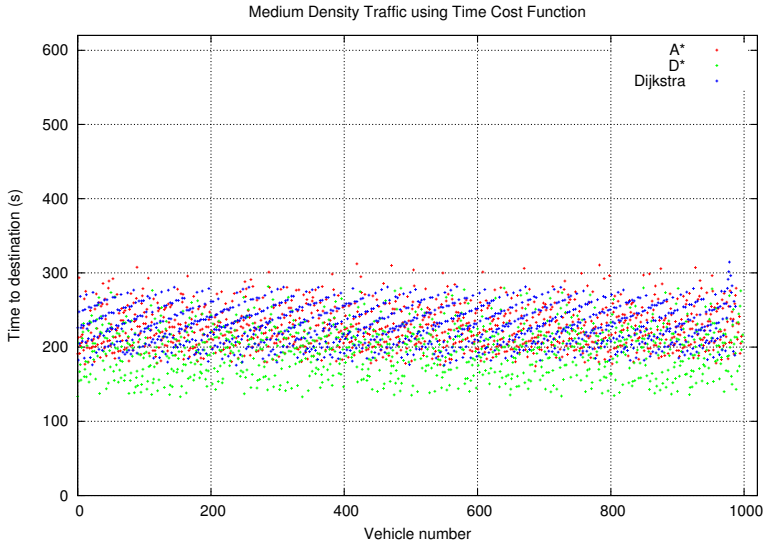

Figure 4.9: Medium Density Traffic using Time Cost Evaluation Function Scatterplot - in this scatterplot we can see a record of the travel times of the "smart" vehicles in the medium density traffic scenario using the time cost evaluation function. We can detect a similar pattern in the results that was also visible in the low density traffic scenario in Fig. 4.2, however in this case we have a much greater variation in the travel times, this is due to the fact that there is more random traffic within the traffic network and therefore our agents need more time to reach their destination. From this scatterplot we can also conclude that D* outperforms the other two algorithms in this particular traffic scenario. For the graph key refer to Tab. 4.1

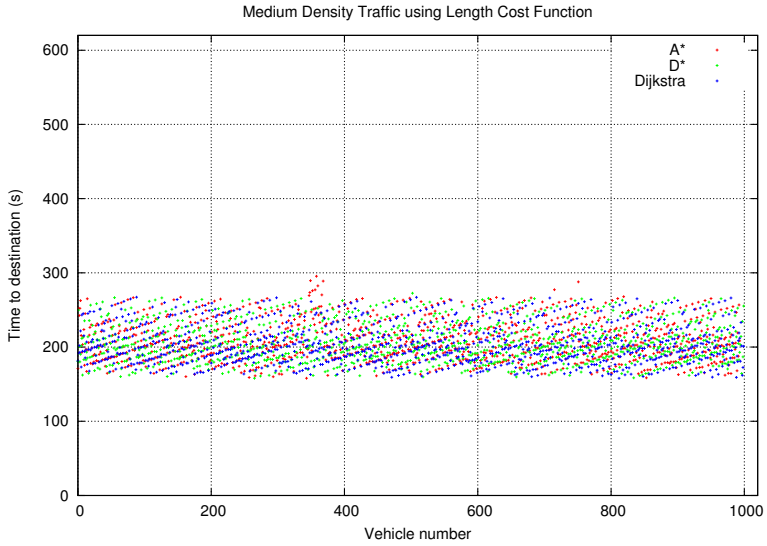

Figure 4.10: Medium Density Traffic using Length Cost Evaluation Function Scatterplot - in this figure we can see the travel times of the "smart" vehicle simulations under the medium density traffic scenario using the length cost evaluation function. We can detect a narrower distribution of the results in this scatterplot if we compare it to the scatterplots using the other cost evaluation functions under this traffic scenario. It is again interesting to see the usual pattern appearing within the gathered results but with also a few results just slightly above the upper bound. For the graph key refer to Tab. 4.1

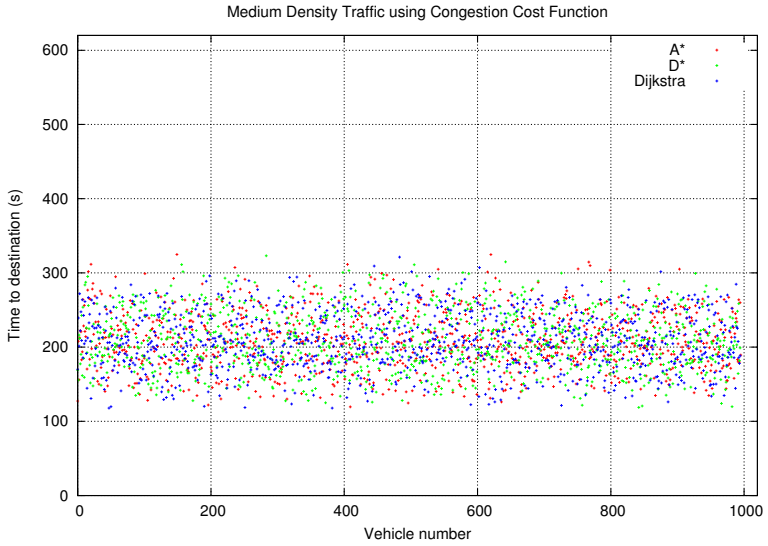

Figure 4.11: Medium Density Traffic using Congestion Cost Evaluation Function Scatterplot - here we can see the results gathered for the travel times of the 3,000 "smart" vehicles simulated in medium density traffic using the congestion cost evaluation function. Similarly to the low density traffic scenario in Fig. 4.4 we see a great variation between the travel times of our agents and there is no pattern like in the cases when using time and length cost functions. This is due to the fact that when using the congestion cost function the "smart" agents travel through different routes to the destination and this plays a major role in the travel time needed to reach this destination. For the graph key refer to Tab. 4.1

whereas in the case of Low Density Traffic they were in the 23rd percentile. Referring to Tab. 4.6, 4.7, and 4.8 where we find the mean, standard deviation and coefficient of variation results for each algorithm and cost evaluation function combination, we can again confirm what we concluded by looking at the scatterplots of each case. As already mentioned we see a substantial difference in the mean values when using the time cost evaluation function, but a very similar performance in the other cases.

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	218 s	207 s	209 s

Table 4.9: Medium Density Traffic Overall Statistical Results by Cost Function

In Tab. 4.9, we see the overall mean values of the different cost evaluation functions. While we can see that the performance of the time cost function is really poor, the performance of the other two is much better and very close. Comparing these results to the results in Tab. 4.5 for the Low Density Traffic scenario, we can see that my implementation of the congestion cost evaluation function is catching up to the other and even performed better than the time cost evaluation function in this case. From this we can conclude that the higher the density of the traffic, the better the congestion cost evaluation function will perform. According to that conclusion I was expecting to see the best performance of the congestion cost function in the High Density Traffic scenario which I will cover in the next subsection.

As previously mentioned the distributions of the length and congestion cost functions in this scenario are very similar between the algorithms. This is visible in Fig. 4.13 and 4.14. In this particular traffic scenario however, when looking at Fig. 4.12 we can detect the differences between the distributions depending on the algorithm used. In this figure we can also detect that the D* algorithm has many more results below 200 s which is the main reason why this algorithm and cost function was found to be the best combination to use under Medium Density Traffic conditions.

4.3.3 High Density Traffic

My High Density Traffic scenario, or in other words Rush Hour scenario, is defined to simulate number of vehicles equal to 68% of the total capacity of the traffic network. A preview of the Rush Hour scenario being simulated in my traffic simulation software is found in Fig. 4.15.

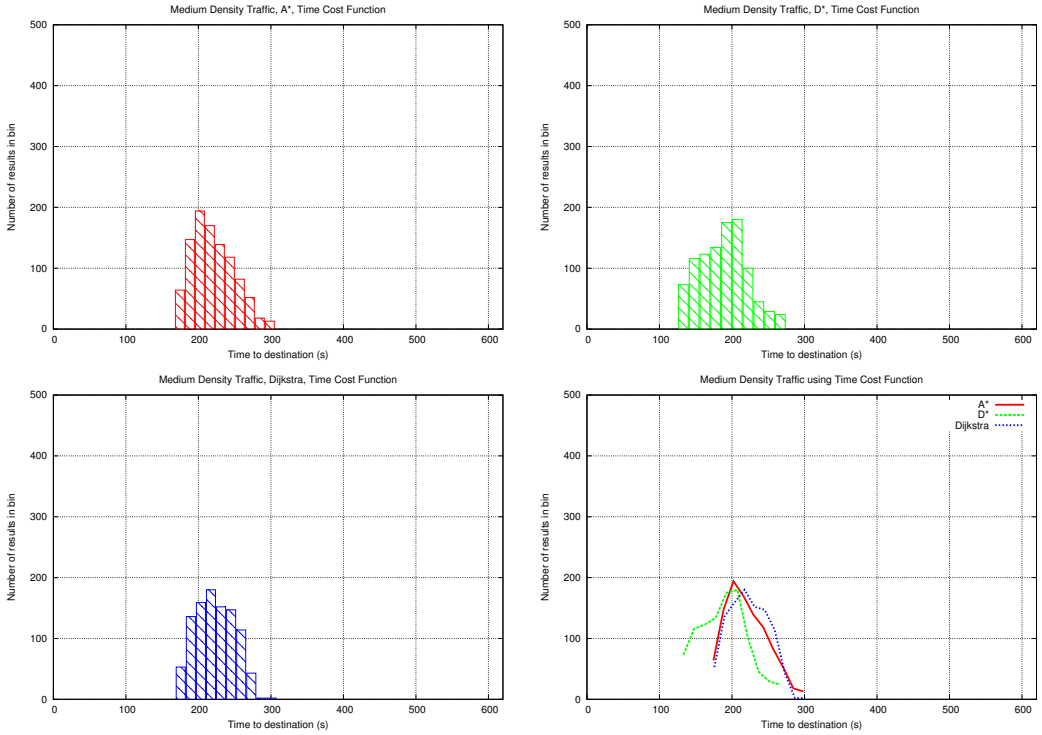

Figure 4.12: Medium Density Traffic using Time Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution histograms of the path finding algorithms using the time cost evaluation function under the medium density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this particular case we can detect a different distribution curve in the different algorithm cases, however it is clear that the results obtained using the D* algorithm are much lower than the other two algorithms which leads to the conclusion that the D* algorithm is the best option to use in this traffic scenario using the time cost evaluation function. For the graph key refer to Tab. 4.1

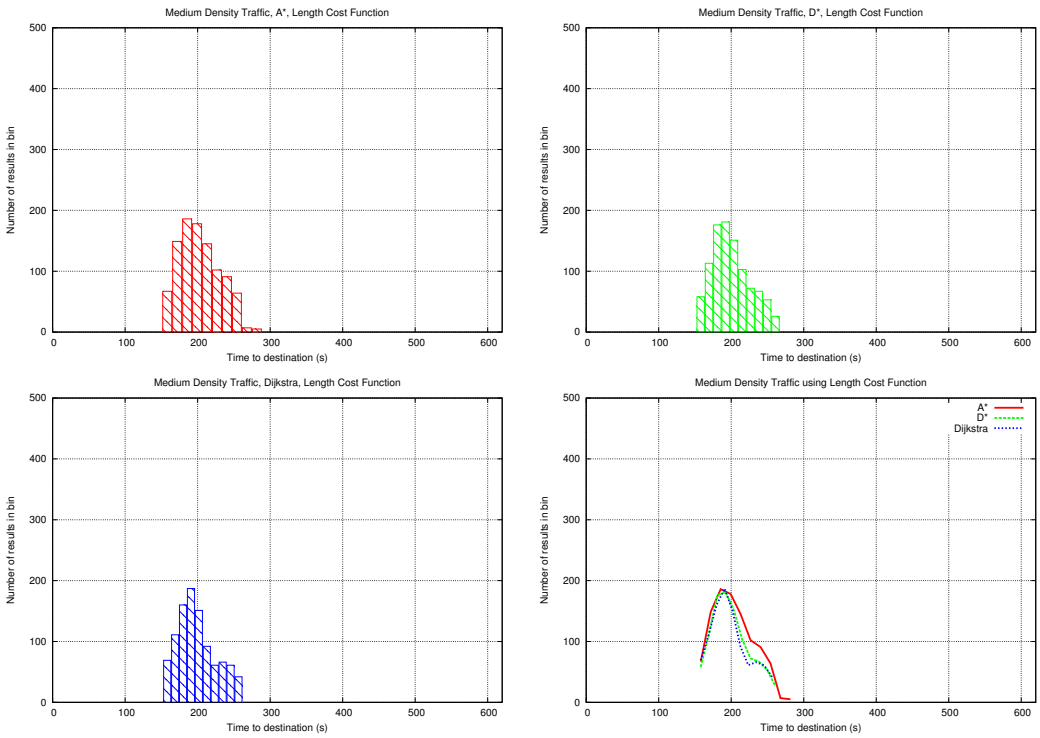

Figure 4.13: Medium Density Traffic using Length Cost Evaluation Function Distribution Histograms - in this figure we can see the distribution histograms of the different algorithms under the medium density traffic scenario using the length cost evaluation function. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this case the distribution curves of the different algorithms are very similar with the exception of the A* algorithm that has a slightly different downward slope towards the end when compared to the other two. This leads us to the conclusion that the path finding algorithms perform very similarly under these traffic conditions using the length cost function. For the graph key refer to Tab. 4.1

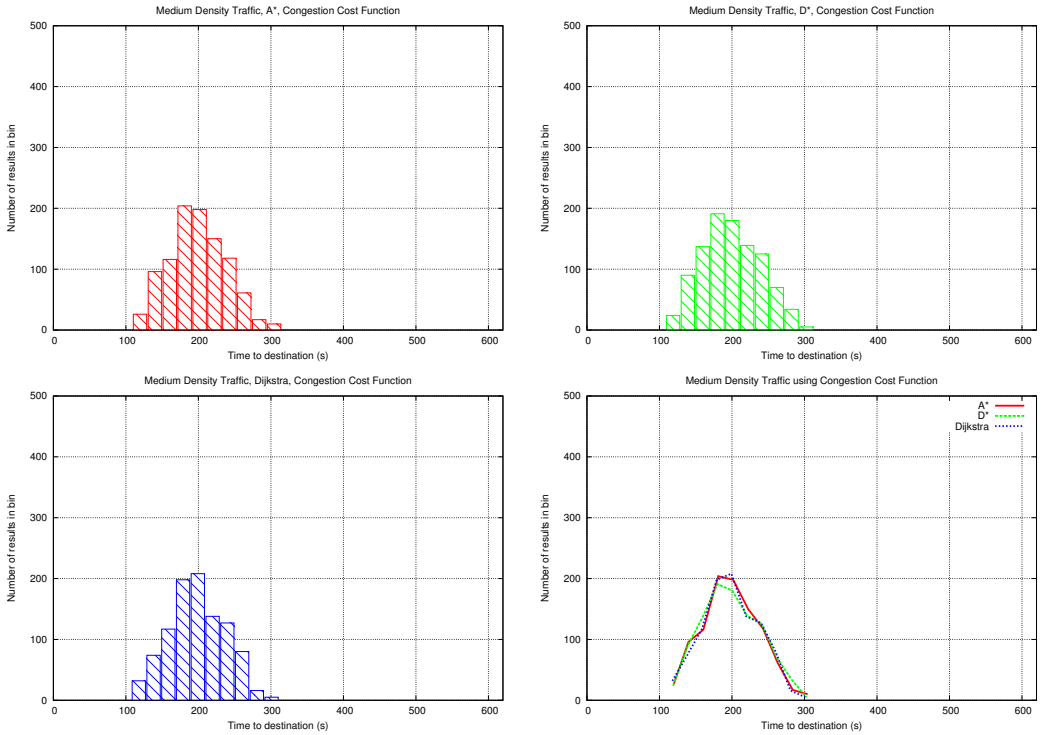

Figure 4.14: Medium Density Traffic using Congestion Cost Evaluation Function Distribution Histograms - here we can see the distribution histograms of the different algorithms using the congestion cost evaluation function under medium density traffic conditions. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In these graphs we can detect that the distribution curves of the different algorithms are almost identical to each other which leads to the conclusion that the performance of all our algorithms is very similar and we can expect approximately the same mean travel time using each of them. For the graph key refer to Tab. 4.1

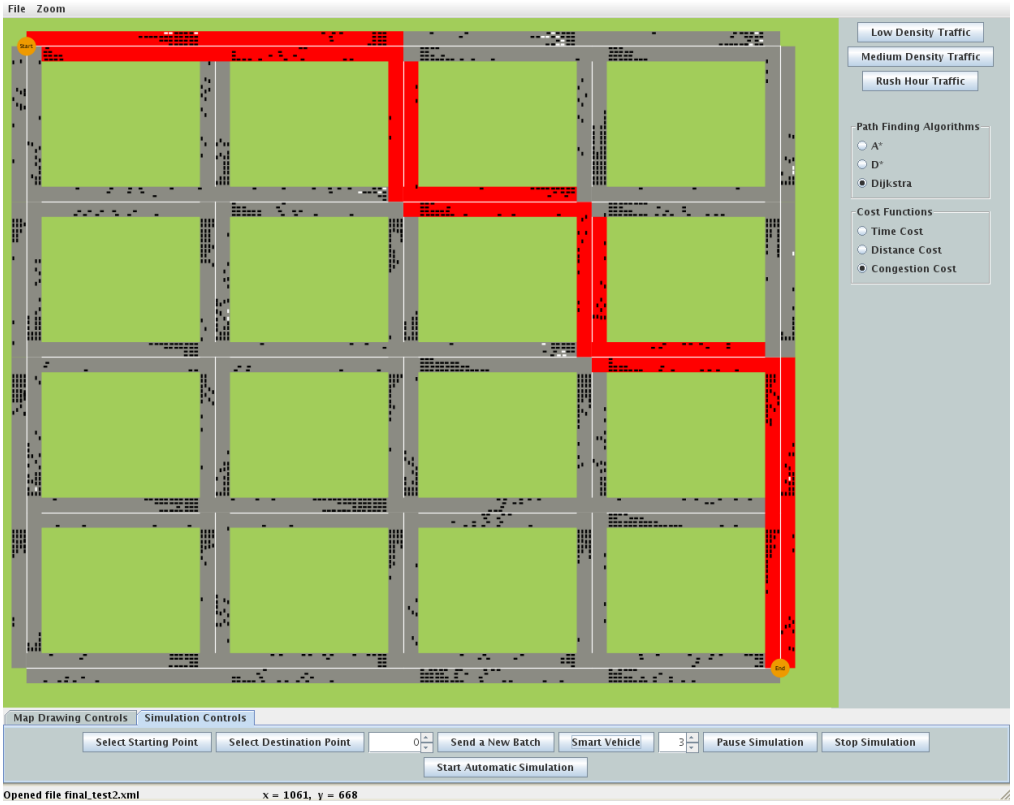

Figure 4.15: High Density Traffic Scenario being simulated in my traffic simulation software - Here we can see the rush hour traffic scenario being simulated in my traffic simulation application. The number of random simulated vehicles is equal to 68% of the total capacity of the traffic network. At the corner nodes of the traffic network we can see the starting and destination nodes for our "smart" driver agents marked with an orange circle, as well as the best path between them at the moment marked with a red route. The vehicles simulated in the traffic network are marked with black rectangles for the random driver agents and white rectangles for the "smart" driver agents. We can also see the various control panels used to control the different simulation settings on the right and bottom of the application window.

The results obtained in this scenario indicate that once again the D* algorithm performs better in the given conditions, however this time combined with the congestion cost evaluation function. This is where we find the best average travel time with a value of 247 seconds. Looking at the scatterplots of this scenario we can detect some interesting occurrences, especially in Fig. 4.16 where we see a very weak performance by the Dijkstra algorithm. In this specific case, using Dijkstra algorithm and time cost function we can see a traffic congestion occurring almost immediately at the beginning of the simulation which lasts approximately until the 550th vehicle simulated, at which point we see the travel times of the vehicles dropping; however, this is enough to bring the average travel time to the value of 312 s which is the highest average travel time I have recorded in this case study. In Fig. 4.17 we can see the performance of all the algorithms is very similar and we cannot clearly detect the best case algorithm. Furthermore in Fig. 4.18, we can see the great variation in travel times the congestion cost evaluation function brings, but in this case it records many more results below the 200 s mark unlike in the case of the time and length cost functions, which in this scenario makes the congestion cost function the best option to use.

	A*	D*	Dijkstra
Mean	287 s	274 s	312 s
Standard Deviation	33 s	29 s	43 s
Coefficient of Variation	11.44 %	10.77 %	13.88 %

Table 4.10: High Density Traffic using Time Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	256 s	255 s	260 s
Standard Deviation	33 s	39 s	33 s
Coefficient of Variation	12.87 %	15.28 %	12.79 %

Table 4.11: High Density Traffic using Length Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	265 s	247 s	252 s
Standard Deviation	55 s	43 s	42 s
Coefficient of Variation	20.65 %	17.51 %	16.58 %

Table 4.12: High Density Traffic using Congestion Cost Function Statistical Results

Coming back to the assumption I did in the previous subsection, and referring to Tab.4.10, 4.11, and 4.12, we can see that I have obtained the results I expected. It was obvious in the Medium Density traffic scenario that the performance of the congestion cost evaluation function was better as the density of the traffic was raising, and having analyzed the results in the Rush Hour traffic scenario we can clearly see that this is in fact the case, as in this scenario the congestion cost function records the minimum average travel time. Furthermore, once again we have the greatest coefficient of variation in the case when using

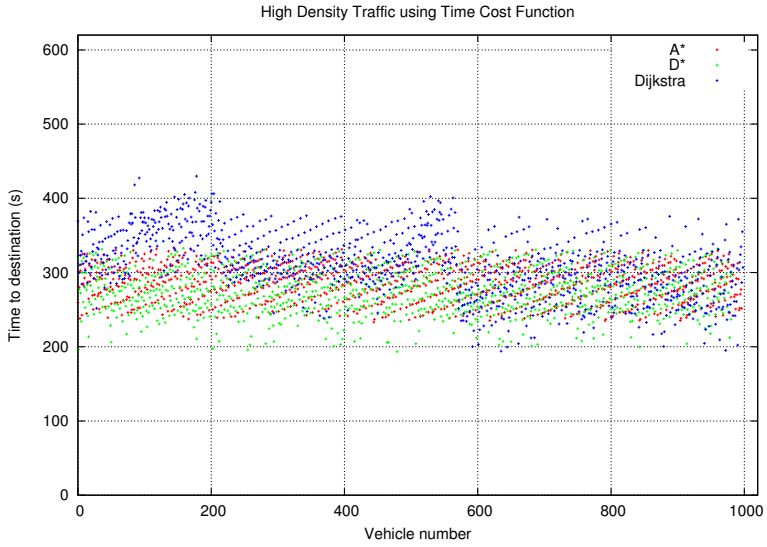

Figure 4.16: High Density Traffic using Time Cost Evaluation Function Scatterplot - in this diagram we can see the travel times of our "smart" driver agents in the rush hour traffic scenario using the time cost evaluation function. As already seen when using this cost function in Fig. 4.2 and 4.9 in the different traffic scenarios there is a pattern in the results, but this time we can also detect more disturbances mainly because the whole traffic network is overwhelmed with vehicles and there is a lot of traffic congestion. This is particularly visible when using the Dijkstra algorithm where we see higher travel times in the beginning up until the 550th simulated "smart" vehicle. For the graph key refer to Tab. 4.1

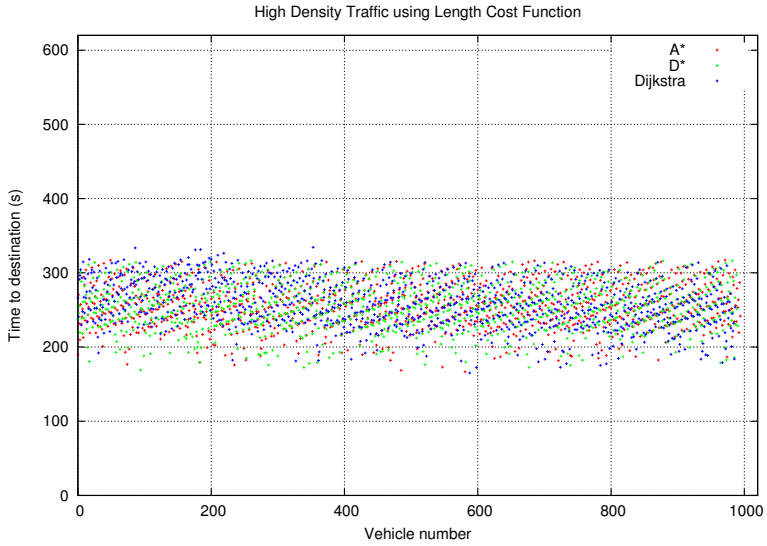

Figure 4.17: High Density Traffic using Length Cost Evaluation Function Scatterplot - in this scatterplot we see the results gathered in the rush hour traffic scenario using the length cost evaluation function. Here it is really interesting to see that all of the algorithms perform very similarly, thus we cannot tell the best algorithm for this particular case from the graph. For the graph key refer to Tab. 4.1

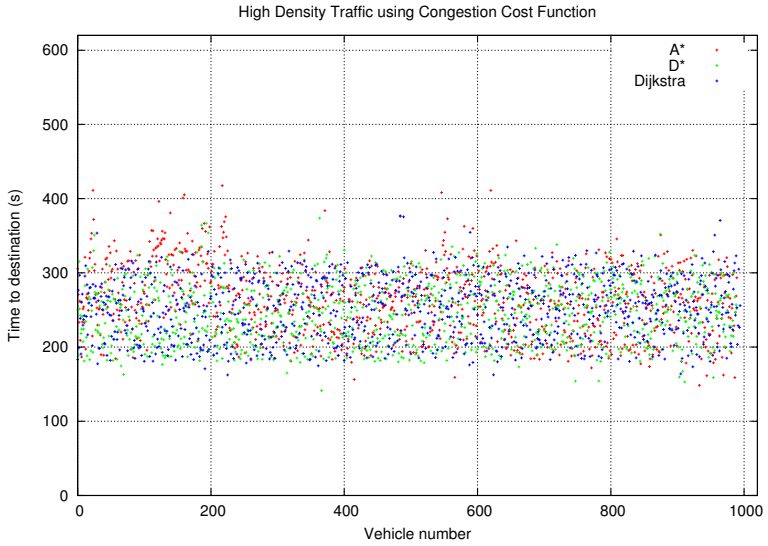

Figure 4.18: High Density Traffic using Congestion Cost Evaluation Function Scatterplot - here we see the travel time results of our simulated vehicles using the congestion cost evaluation function under rush hour traffic conditions. As usual in the case when using the congestion cost function there is a lot of variation in the travel times, however it is very interesting to see that there are a lot more travel times under the 200 s mark when compared to Fig. 4.17 and especially Fig. 4.16. Therefore we can conclude that under rush hour traffic conditions the congestion cost evaluation function performs better than the other two. For the graph key refer to Tab. 4.1

the congestion cost function and a more dense distribution in the other cases.

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	291 s	257 s	255 s

Table 4.13: High Density Traffic Overall Statistical Results by Cost Function

Referring to Tab. 4.13, where we see the performance of the different cost evaluation functions, we can detect that the congestion cost function does outperform the other two in this traffic scenario, as previously mentioned. Additionally we see the effect of the traffic congestion which occurred under Dijkstra algorithm using time cost evaluation function and brought the mean travel time when using this cost function to 291 s.

Looking at the distribution histograms in this scenario in Fig. 4.19, 4.20 and 4.21, we find some oddly shaped, but still normal distributions. The case of the length cost evaluation function is particularly interesting in this scenario as it provides almost the same distribution curve using all of the algorithms and therefore we see the lowest difference between the mean travel times in that particular cost function case. Furthermore in Fig. 4.21, we detect a wider distribution of our results, but a much higher portion of results below the 200 s range compared to the other cost evaluation functions.

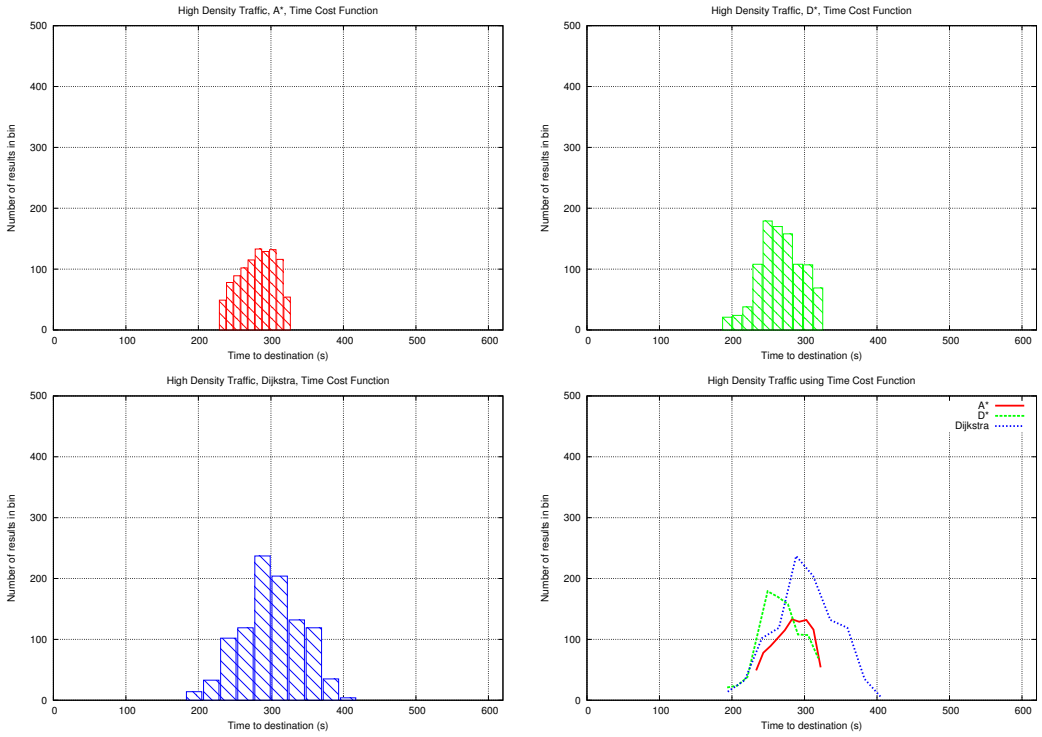

Figure 4.19: High Density Traffic using Time Cost Evaluation Function Distribution Histograms - here we can see the distribution histograms of the results gathered using the different algorithms combined with the time cost evaluation function under rush hour traffic conditions. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this case we see different distribution curves in all of the algorithms, but it is clear that using D* provides much lower travel times than the other two algorithms, thus making it the best algorithm to use in this particular case. For the graph key refer to Tab. 4.1

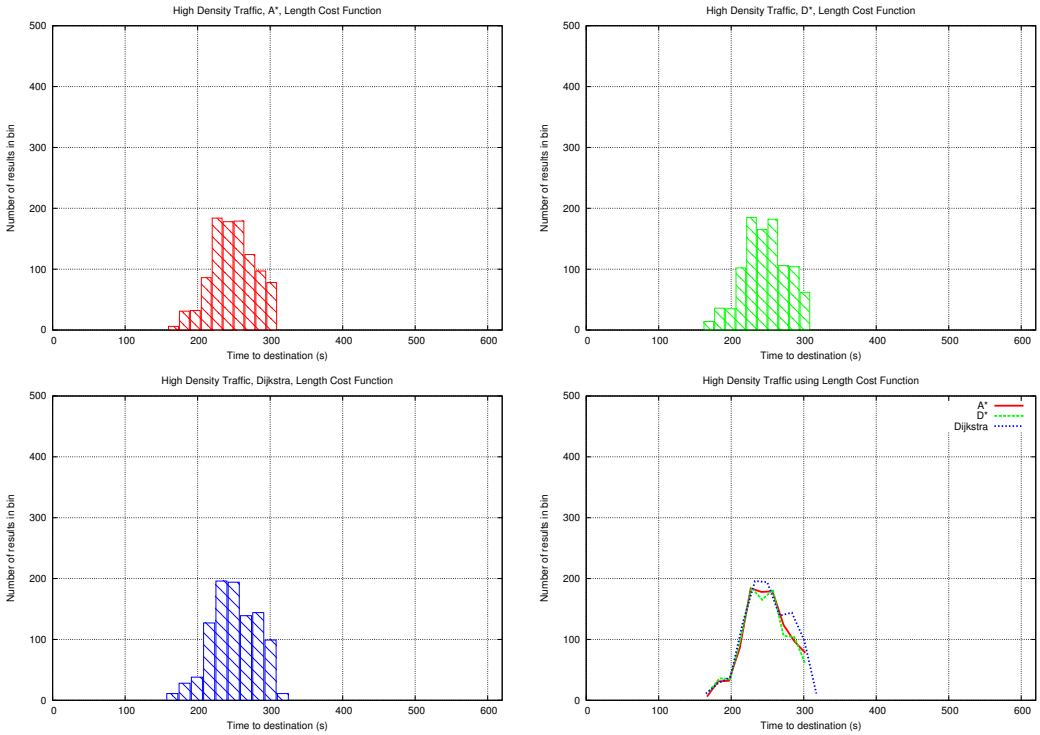

Figure 4.20: High Density Traffic using Length Cost Evaluation Function Distribution Histograms - in this figure we see the distribution histograms of the travel times gathered under rush hour traffic conditions using the length cost evaluation function. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. It is very interesting to see that even in a rush hour traffic scenario, where we have a lot of traffic congestion and many random vehicles to account for, we have almost exactly the same distribution curves for all the algorithms leading to very similar mean travel times among them. For the graph key refer to Tab. 4.1

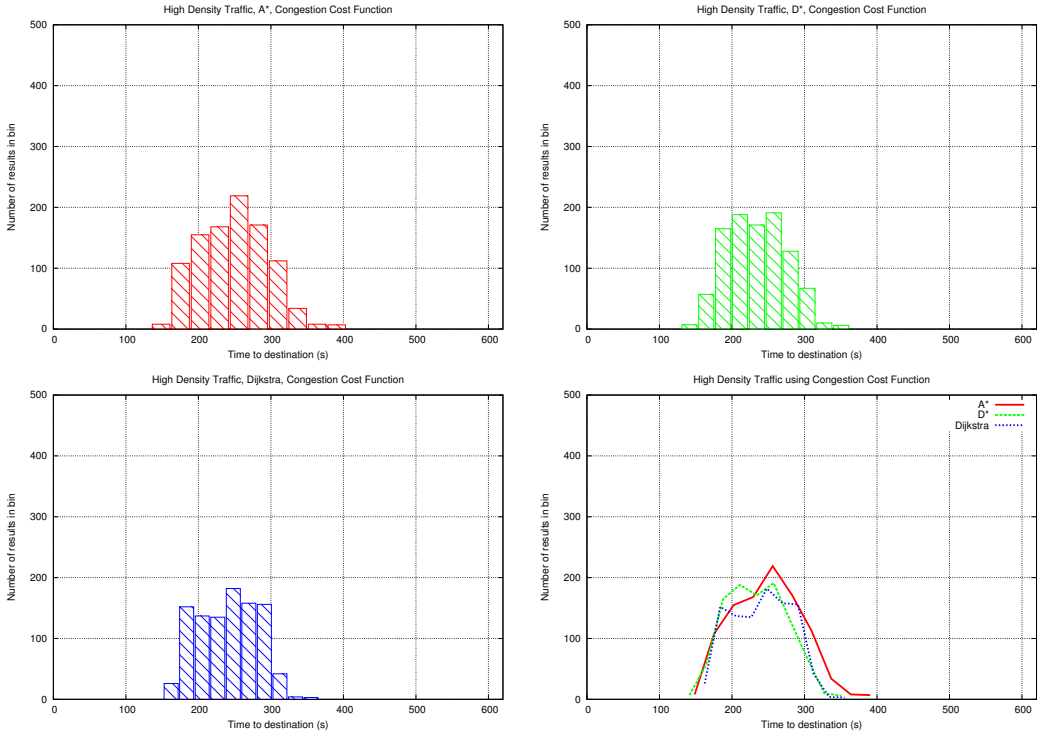

Figure 4.21: High Density Traffic using Congestion Cost Evaluation Function Distribution Histograms - in these graphs we can see the distribution curves for the different path finding algorithms using the congestion cost evaluation function under the rush hour traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. In this case we can detect a much wider distribution when we compare it with the distributions in Fig. 4.19 and 4.20, however we see many of the results are in the lower ranges especially using D* and Dijkstra algorithms, therefore we can say that the congestion cost function provides lower travel times than the other two cost functions under this traffic scenario. For the graph key refer to Tab. 4.1

Knowing a great deal is not the same as being smart; intelligence is not information alone but also judgment, the manner in which information is collected and used.

Carl Sagan

5

The "Smart" Agent

5.1 Hybrid Cost Evaluation Functions

Back in Chapt. 3, I introduced the hybrid cost evaluation functions. These are essentially combinations of the three basic cost functions I studied in the previous chapter. In this section I present the method I used to combine the basic functions into these hybrid functions, as well as I discuss their objective.

It is interesting to note that while in the case when I was using the basic functions I could use a different metric as a cost for a particular road, in the case when using the hybrid cost functions I had to scale their costs such that each basic cost function provided an equal amount of impact in the hybrid cost combination. Intuitively I chose the scale to be between 0 and 1. This way, I needed to scale only two of the basic cost functions (time and length) because the congestion cost function was already calculating a value between 0 and 1. I have used Eq. 5.1 and Eq. 5.2 to scale the actual time and length costs respectively, and Eq. 5.3 and Eq. 5.4 to scale the heuristic time and length costs. I will use these later to define the set of rules for the hybrid cost evaluation functions.

$$\text{time}_{actual} = \frac{\text{time cost}}{\text{time cost of highest time cost road in the network}} \tag{5.1}$$

$$\text{length}_{actual} = \frac{\text{length cost}}{\text{length of the longest road in the network}} \tag{5.2}$$

$$\text{time}_{heuristic} = \frac{\text{time heuristic cost}}{\text{highest time cost possible in the network}} \tag{5.3}$$

$$\text{length}_{heuristic} = \frac{\text{length heuristic cost}}{\text{highest length cost possible in the network}} \tag{5.4}$$

Using the aforementioned actual and heuristic scales for the time and length basic cost functions, and adding the congestion basic cost function I defined the following set of rules that define the hybrid cost functions:

- Actual Cost: combination of the basic actual cost functions described in Eq. 5.5;
- Heuristic Cost: combination of the basic heuristic cost functions described in Eq. 5.6;

$$\text{actual hybrid cost} = \frac{\sum \text{actual costs}}{\text{number of cost functions combined}} \tag{5.5}$$

$$\text{heuristic hybrid cost} = \frac{\sum \text{heuristic costs}}{\text{number of cost functions combined}} \tag{5.6}$$

These hybrid cost functions provide a pathfinding algorithm with more information about the traffic network when choosing a path. This is the case because they include more factors in their value when compared to the basic cost evaluation functions. Before conducting the case study using the hybrid cost functions, I believed that having more factors affecting the path my intelligent driver agents take to their destination will lower their travel times. However, while this was true for some traffic scenarios, it proved false for others. In the next section I will present my findings and discuss the best combination of algorithm and cost function to use under the different traffic conditions.

5.2 Case Study

Similarly to the previous chapter, where I presented a case study of my intelligent agents and their travel times using the basic cost evaluation functions, in this section I present the case study conducted using the hybrid cost functions. Additionally, I analyze and compare the results of the two case studies and I discuss their differences.

Once again the case study includes simulations of vehicular traffic under the three different traffic scenarios discussed in Chapt. 2:

- Low Density Traffic
- Medium Density Traffic
- High Density Traffic (Rush Hour)

However, while I had only 3 basic cost evaluation functions, as previously mentioned, I have 4 different hybrid cost evaluation functions. Following the simulation pattern from my last case study and conducting 1,000 "smart" vehicle simulations through each traffic scenario using each separate combination of algorithm and cost evaluation function brought me to the total of 36,000 "smart" vehicle simulations in this case study.

The presentation and analysis of the results follow the same pattern mentioned in Sect. 4.3 and use the legend in Tab. 4.1. In the following subsections I study the gathered results of the hybrid cost functions case study in more details.

5.2.1 Low Density Traffic

From the gathered results in this traffic scenario we can observe that my implementation of the Dijkstra algorithm performed the best especially using the time and length hybrid cost evaluation function. The average travel time found using this combination of algorithm and cost function was 163 seconds. Additionally, both A* and D* using the same hybrid cost function have very similar average results to Dijkstra, being only 414.81 ms and 88.26 ms higher respectively (see Appx. A). This was somewhat expected as I knew from the previous case study that time and length cost evaluation functions provided better results in the low density traffic scenario. I was also happy to see that combining time and length cost functions in one hybrid cost function provided even better performance than each of them individually. This will be further discussed in Subsect. 5.2.4. We can see scatterplots of this traffic scenario in Fig. 5.1, 5.2, 5.3 and 5.4. In Fig. 5.1 we can detect a very similar performance of all the algorithms, as well as the usual pattern in the results caused by the traffic light intervals, which was expected knowing that the path never changes during the simulation. In the cases where the congestion cost evaluation function was combined together with another function however, we can see scattered results which is due to the changing path during execution of the simulation. While in Fig. 5.2, and 5.3 we cannot detect a "winning" algorithm, we

can eliminate the A* algorithm in the case presented in Fig. 5.4 because it causes a traffic congestion at two occasions. This is visible in the scatterplot in the beginning and after the 280th simulated vehicle which clears up somewhere around the 450th simulated vehicle.

	A*	D*	Dijkstra
Mean	164 s	163 s	163 s
Standard Deviation	23 s	23 s	23 s
Coefficient of Variation	14.32 %	14.12 %	14.07 %

Table 5.1: Low Density Traffic using Time and Length Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	201 s	200 s	202 s
Standard Deviation	35 s	37 s	37 s
Coefficient of Variation	17.57 %	18.40 %	18.59 %

Table 5.2: Low Density Traffic using Time and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	197 s	188 s	189 s
Standard Deviation	36 s	32 s	31 s
Coefficient of Variation	18.16 %	16.80 %	16.47 %

Table 5.3: Low Density Traffic using Length and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	198 s	177 s	177 s
Standard Deviation	49 s	34 s	33 s
Coefficient of Variation	24.81 %	19.05 %	18.37 %

Table 5.4: Low Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Results

Studying the statistical results of this scenario we can observe a generally greater coefficient of variation compared to the simulations using the basic cost evaluation functions. I predict this is due to the fact that my congestion cost function is combined in 3 out of the 4 hybrid cost functions used and since the "smart" drivers change the path they follow to the destination dynamically while travelling we can see variations in the final travel times of our agents. Referring to Tab. 5.1, we can see what we confirmed from the scatterplots that the performance of all the algorithms is very similar to each other. The same goes for the next case where I use time and congestion hybrid cost evaluation function in Tab. 5.2. In Tab. 5.3 and 5.4 however, we can see differences in the results and we see both D* and Dijkstra outperforming the A* algorithm which as we saw in the scatterplots was causing traffic congestion at few occasions.

Looking at Tab. 5.5, where we see the overall mean values of the different hybrid cost

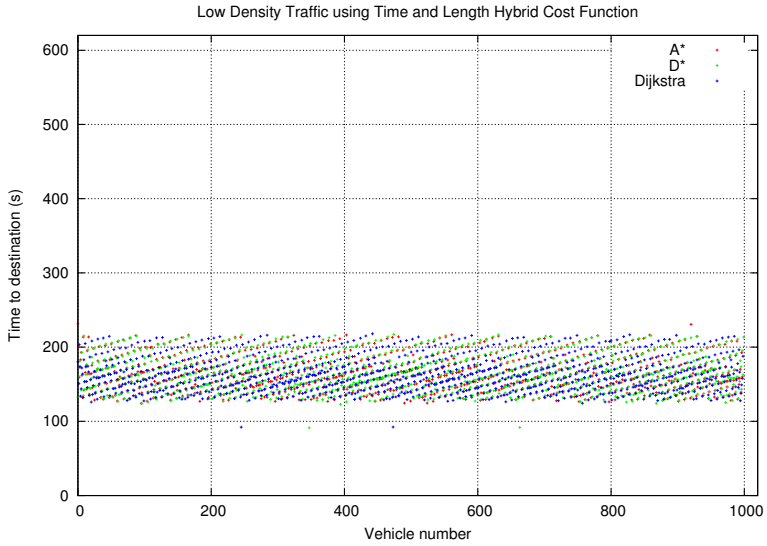

Figure 5.1: Low Density Traffic using Time and Length Hybrid Cost Evaluation Function Scatterplot - here we can see the travel times of all 3,000 "smart" vehicles simulated in the low density traffic scenario using the time and length hybrid cost evaluation function. Just like in the basic cost function scatterplots, we see a pattern in the recorded travel times which is there as a result of the traffic light intervals and the rate at which the "smart" vehicles enter the traffic network. It is not easily identifiable in this graph what algorithm performs better in this traffic scenario using this hybrid cost function because they all provide very similar results. For the graph key refer to Tab. 4.1

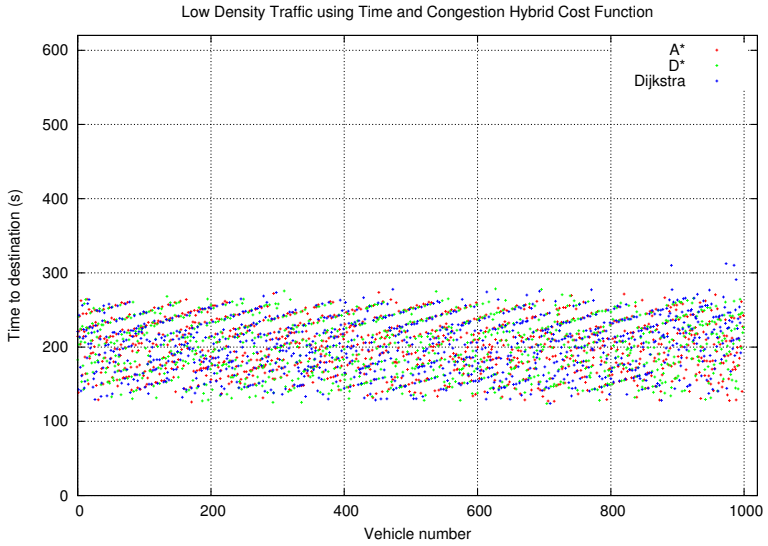

Figure 5.2: Low Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Scatterplot - here we see the travel times of the "smart" vehicles when using the time and congestion hybrid cost function in the low density traffic scenario. Unlike in the basic congestion cost function case where we could not see a pattern in the results, here we can notice some kind of pattern even though the travel times do vary a lot especially towards the end. This pattern comes from the influence the basic time cost function has on this particular hybrid function. Once again all the algorithms provide similar results, but we can notice they are in a higher range when compared to Fig. 5.1. For the graph key refer to Tab. 4.1

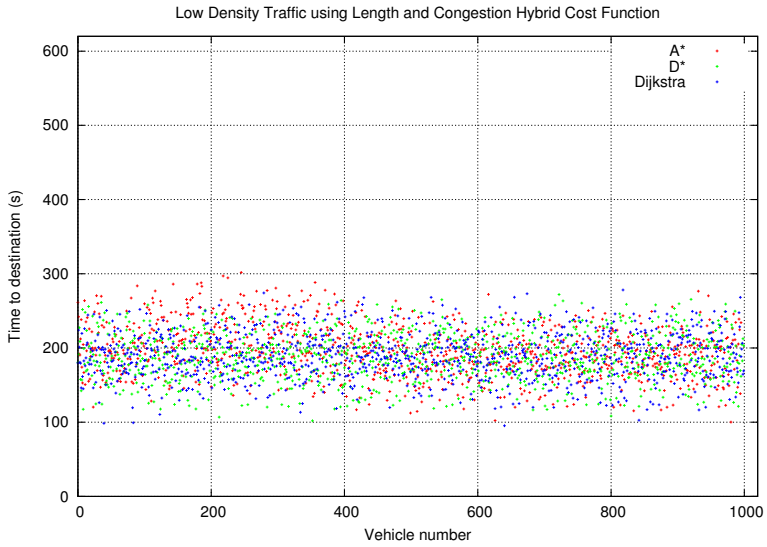

Figure 5.3: Low Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Scatterplot - here we see the data gathered when the "smart" vehicles were using the length and congestion hybrid cost function in the low density traffic scenario. It is interesting that in this scatterplot we can not notice a pattern like in Fig. 5.2 but the results are more scattered. We can also notice that in the first 400 simulated vehicles A* provides generally higher travel times compared to the other two algorithms. For the graph key refer to Tab. 4.1

	TL	TC	LC	All
Mean	163 s	201 s	192 s	184 s

Table 5.5: Low Density Traffic Overall Statistical Results by Hybrid Cost Function

functions, we can detect once again that the time and length cost functions in combination work well under low density traffic conditions. This is clear when we see the lowest mean value of 163 s is obtained using the time and length hybrid cost function, while the second best mean value comes from the hybrid combination of all basic cost functions with a value of 184 s. I was also pleased to find out that using the hybrid cost functions provided generally lower mean values than the basic cost functions in this traffic scenario.

Furthermore we have the distribution histograms of the results in this traffic scenario in Fig. 5.5, 5.6, 5.7 and 5.8. We can again detect the normal-like distribution curves with an exception of the time and congestion cost function combination where we see a somewhat different pattern with no clear peak of the results. Once again we have the narrowest distribution when we do not use a combination of congestion cost evaluation function as seen in Fig. 5.5. Additionally, it is interesting to see the distribution of the results using the length and congestion hybrid cost function in Fig. 5.7 where we have almost overlapping

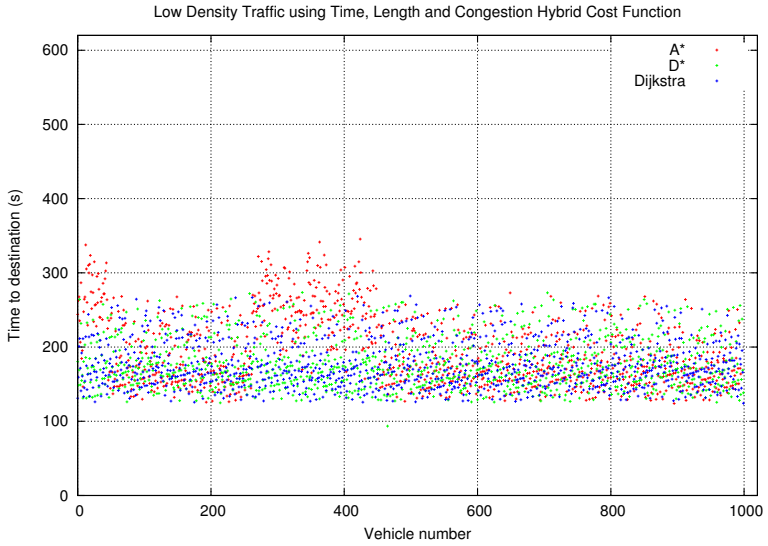

Figure 5.4: Low Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Scatterplot - in this scatterplot we see the travel times gathered in the low density traffic scenario using the hybrid combination of all the basic cost evaluation functions. There is again a certain pattern noticeable in the results, however the more important concern here is that A* is causing two big traffic congestions. One right at the beginning, the second one from around the 280th to the 450th simulated vehicle. This occurrence is what raises the mean travel times of the agents when using A* under this traffic scenario. For the graph key refer to Tab. 4.1

distribution curves with a slightly skewed A* distribution curve.

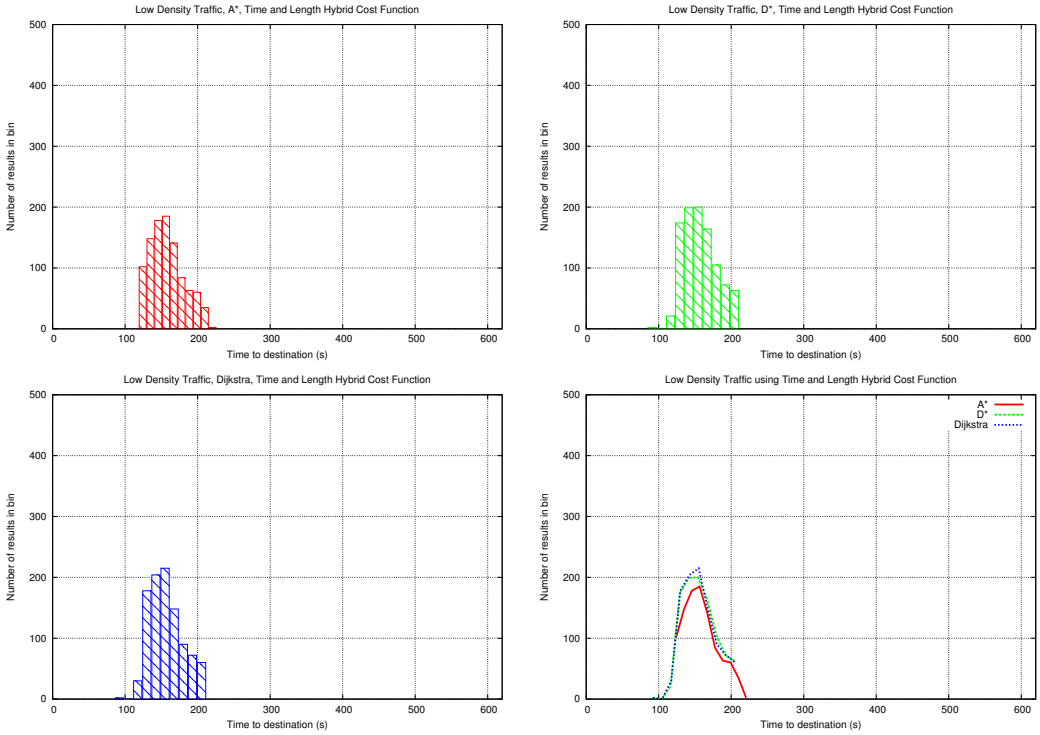

Figure 5.5: Low Density Traffic using Time and Length Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution curves for the various pathfinding algorithms using the time and length hybrid cost function under the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. We can notice a narrow distribution in all the cases and generally similar distribution curve for the different algorithms in these histograms. For the graph key refer to Tab. 4.1

5.2.2 Medium Density Traffic

Similarly to the basic cost evaluation function case study, I expected to see results showing a combination of the time cost function with another function to perform better in this traffic scenario. This was indeed the case proved by the best mean travel time with a value of 195 seconds obtained by the Dijkstra algorithm using the time and length hybrid cost evaluation function. The second best mean travel time however is very close to the first one with a value of 196 s and is obtained by the D* algorithm using the same (time and length) hybrid cost function. This is the exact phenomenon we saw in the low density traffic scenario with the exception of the A* algorithm which in this traffic scenario case study caused traffic congestion and performed rather poorly. This is also visible in the scatterplot in Fig. 5.9

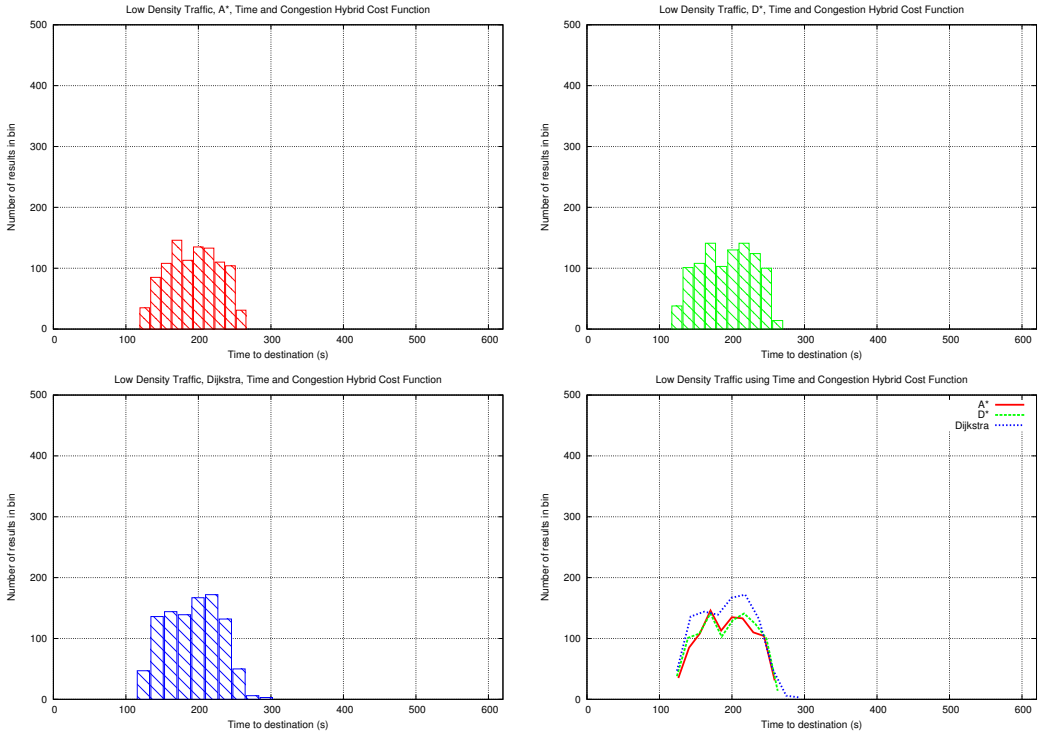

Figure 5.6: Low Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we can see the distribution histograms for the different pathfinding algorithms using the time and congestion hybrid cost function under the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Here we can see a somewhat different distribution curve with no clear peak of results. While A* and D* have almost exactly the same distribution curves, Dijkstra has a similar shape but also a few results near the 300 s range whose influence we can notice in the mean travel times for this case in Tab. 5.2. For the graph key refer to Tab. 4.1

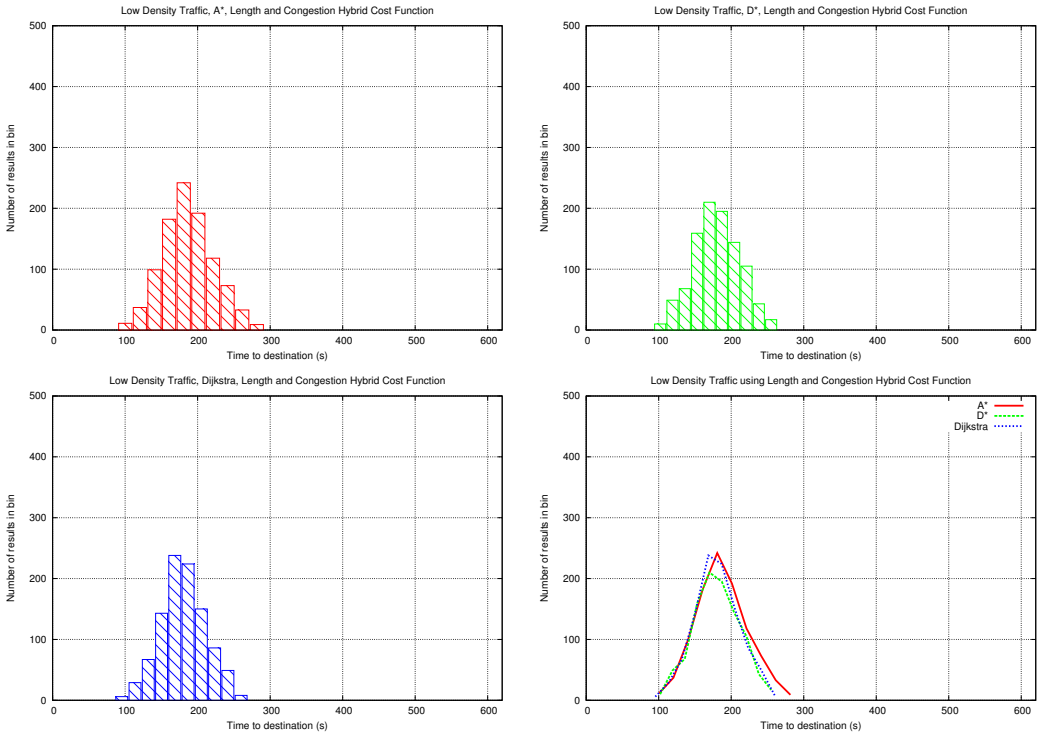

Figure 5.7: Low Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - here we can see the distribution of the travel times gathered for the different pathfinding algorithms using the length and congestion cost evaluation function under the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Looking at the distribution curves in this case we can notice a normal distribution in all the algorithms, however somewhat wider in the case of A* algorithm, which we also notice by the variation coefficient in Tab. 5.3. For the graph key refer to Tab. 4.1

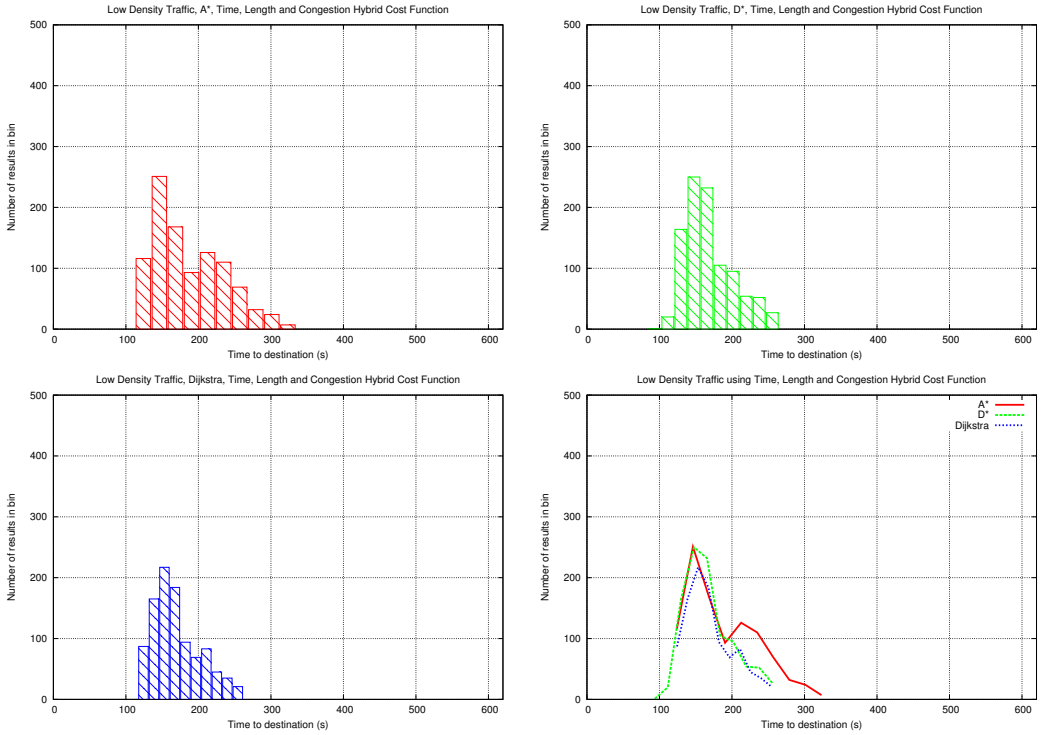

Figure 5.8: Low Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - here we see the distribution histograms for the results gathered using the hybrid combination of all the basic cost evaluation functions under the low density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. The distribution curves of the algorithms are somewhat different. While the difference is not large when we compare D* and Dijkstra algorithms, where D* has a wider distribution curve than Dijkstra, the difference when we include the distribution curve of A* in the comparison is extensive. This is all due to the two traffic congestions caused by the A* algorithm as we saw previously in the scatterplot in Fig. 5.4. For the graph key refer to Tab. 4.1

where we can observe 4 minor and 1 major traffic congestion caused by the A* algorithm. In the next two scatterplots in Fig. 5.10 and 5.11, we can observe a similar performance of all the algorithms with results distributed within a certain range. While in Fig. 5.12 we can again see a major traffic congestion caused using the A* algorithm and therefore resulting in the worse mean travel time in this traffic scenario with the value of 244 s.

	A*	D*	Dijkstra
Mean	222 s	196 s	195 s
Standard Deviation	50 s	32 s	33 s
Coefficient of Variation	22.43 %	16.32 %	16.85 %

Table 5.6: Medium Density Traffic using Time and Length Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	220 s	220 s	216 s
Standard Deviation	34 s	34 s	36 s
Coefficient of Variation	15.57 %	15.42 %	16.47 %

Table 5.7: Medium Density Traffic using Time and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	209 s	208 s	208 s
Standard Deviation	36 s	36 s	36 s
Coefficient of Variation	17.42 %	17.44 %	17.37 %

Table 5.8: Medium Density Traffic using Length and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	244 s	215 s	213 s
Standard Deviation	50 s	35 s	35 s
Coefficient of Variation	20.46 %	16.36 %	16.46 %

Table 5.9: Medium Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Results

Once again we can observe great variations in the results, especially in the case of the A* algorithm where I experienced multiple traffic congestions when using the time and length hybrid cost function and the time, length and congestion hybrid cost function. Moreover, if we refer to Tab. 5.6 and 5.9 we can see the difference these traffic congestions make. While we can see D* and Dijkstra algorithms performing very similarly using these hybrid cost evaluation functions, A* has nearly 30 s slower mean time than both of them in both cases, as well as a coefficient of variation in the 20th percentile while D* and Dijkstra are in the 16th. When using the other two hybrid cost functions however, we can observe a somewhat similar performance by all algorithms with mean travel times within 4 s and 1 s range of each other.

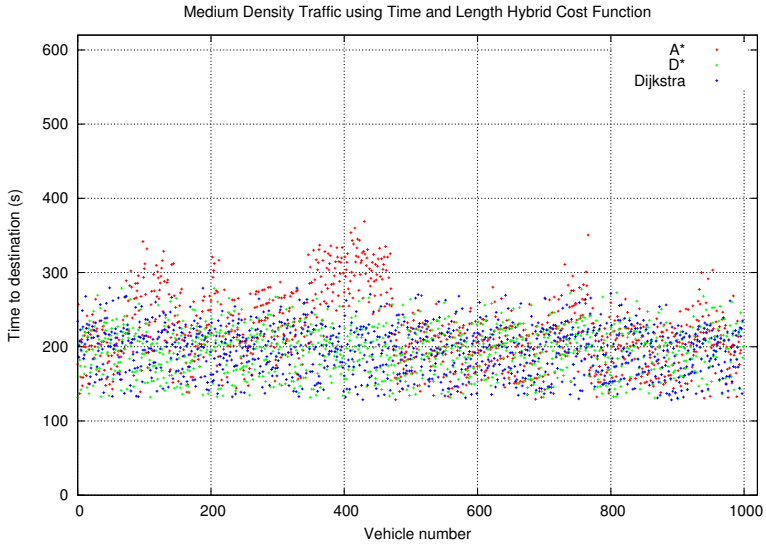

Figure 5.9: Medium Density Traffic using Time and Length Hybrid Cost Evaluation Function Scatterplot - here we can see the travel times of all 3,000 "smart" vehicles simulated in the medium density traffic scenario using the time and length hybrid cost function. In this graph we can notice the A* algorithm performing rather poorly under medium density traffic conditions. There are 4 minor and 1 major traffic congestion caused by A* in these results from where we can predict that A* will have a very high mean travel time compared to the other two algorithms in this particular case. For the graph key refer to Tab. 4.1

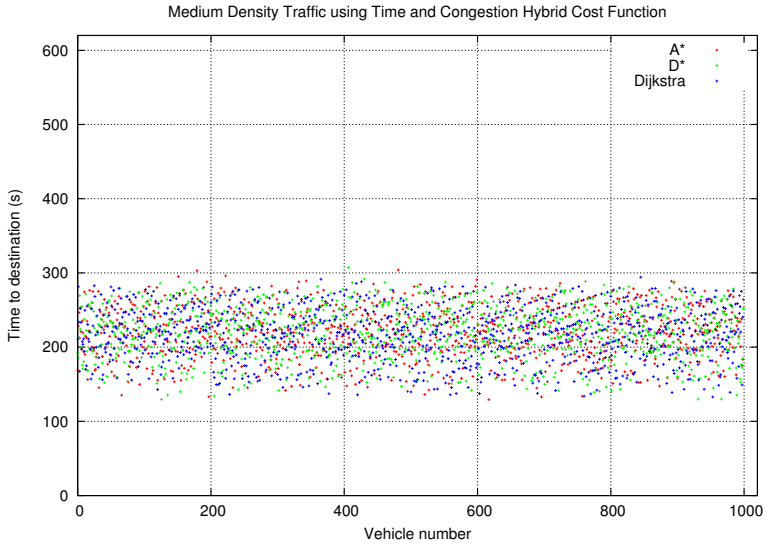

Figure 5.10: Medium Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Scatterplot - here we see the travel times gathered for the "smart" agents using the time and congestion hybrid cost function under medium density traffic conditions. In this scatterplot we can notice a very similar performance by all the algorithms with no traffic congestions caused. Additionally, we no longer notice the pattern in the results as we had in the low density case. For the graph key refer to Tab. 4.1

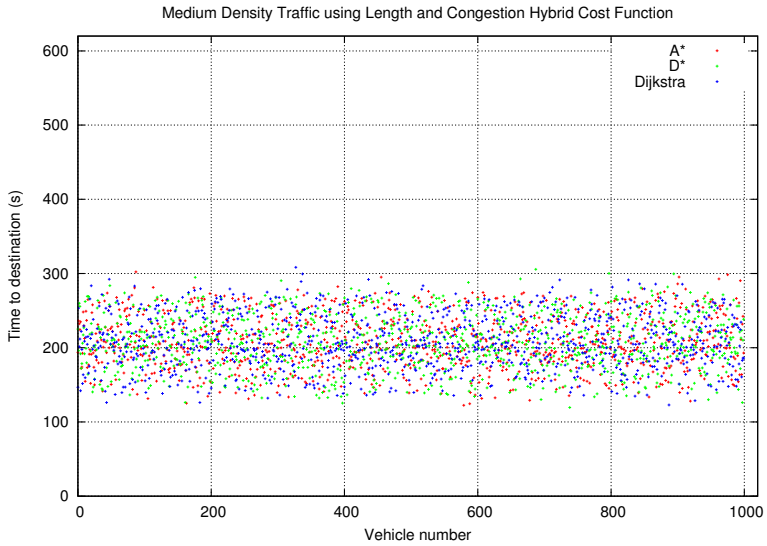

Figure 5.11: Medium Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Scatterplot - here we see the results gathered using the hybrid combination of the basic length and congestion cost functions under the medium density traffic scenario. Similarly to the previous scatterplot, here we also see a similar performance from all the pathfinding algorithms and therefore we cannot determine which one performs the best in this case. For the graph key refer to Tab. 4.1

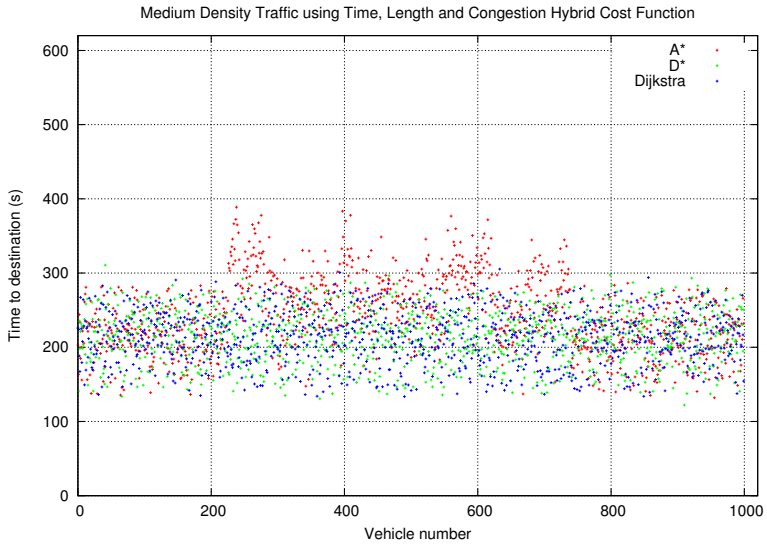

Figure 5.12: Medium Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Scatterplot - in this scatterplot we see the travel times of the 3,000 "smart" vehicles simulated in medium density traffic using the hybrid combination of all the basic cost functions. Once again we see a poor performance by the A* algorithm and a massive traffic congestion caused between the 250th and the 700th simulated vehicle. We can predict by looking at this graph that A* will have the worst mean travel time in this case. For the graph key refer to Tab. 4.1

	TL	TC	LC	All
Mean	204 s	219 s	208 s	224 s

Table 5.10: Medium Density Traffic Overall Statistical Results by Hybrid Cost Function

If we refer to Tab.5.10 we can see that the time and length hybrid cost evaluation function outperforms the other three, just as in the low density traffic case. However, in this traffic scenario the length and congestion hybrid cost function performs very closely too with a mean travel time of only 4 s higher value. This is partially due to the fact that the A* algorithm created a traffic congestion in one case thus increasing the mean travel time of my "smart" driver agents, while in the other case I did not observe any traffic congestion and the mean travel time of the agents is fair. Furthermore, I found that the hybrid cost evaluation functions perform somewhat better than the basic cost evaluation functions in this traffic scenario similarly as they did in the low density traffic case.

Looking at the distribution histograms of the gathered results in Fig. 5.13, 5.14, 5.15 and 5.16, we can observe that D* and Dijkstra algorithms perform very similarly using all the hybrid cost functions. However if we study the distribution of the A* algorithm we can see how much wider distribution of results it has produced in Fig. 5.13 and 5.16, while still performing similarly in the other two cases.

5.2.3 High Density Traffic

In the same manner as in the basic cost evaluation function case study where we saw the D* algorithm performing the best of all the algorithms, in the case when using the hybrid cost functions we see the Dijkstra algorithm outperforming both A* and D*. In this Rush Hour traffic scenario the lowest mean travel time was found by the Dijkstra algorithm using the length and congestion hybrid cost function. It was expected that in this scenario the hybrid functions that included the congestion cost function will perform better and that was in fact the case as we will see further in this subsection. Referring to the scatterplots in Fig. 5.17, 5.18, 5.19 and 5.20, we can see evenly distributed results, with a few exceptions. In Fig. 5.18 we can observe a few occurrences of traffic congestions caused when using the D* algorithm which clear out after the 600th simulated "smart" vehicle; as well as in Fig. 5.20 where we can see a steady increase in travel times after the 500th simulated "smart" vehicle using the Dijkstra algorithm.

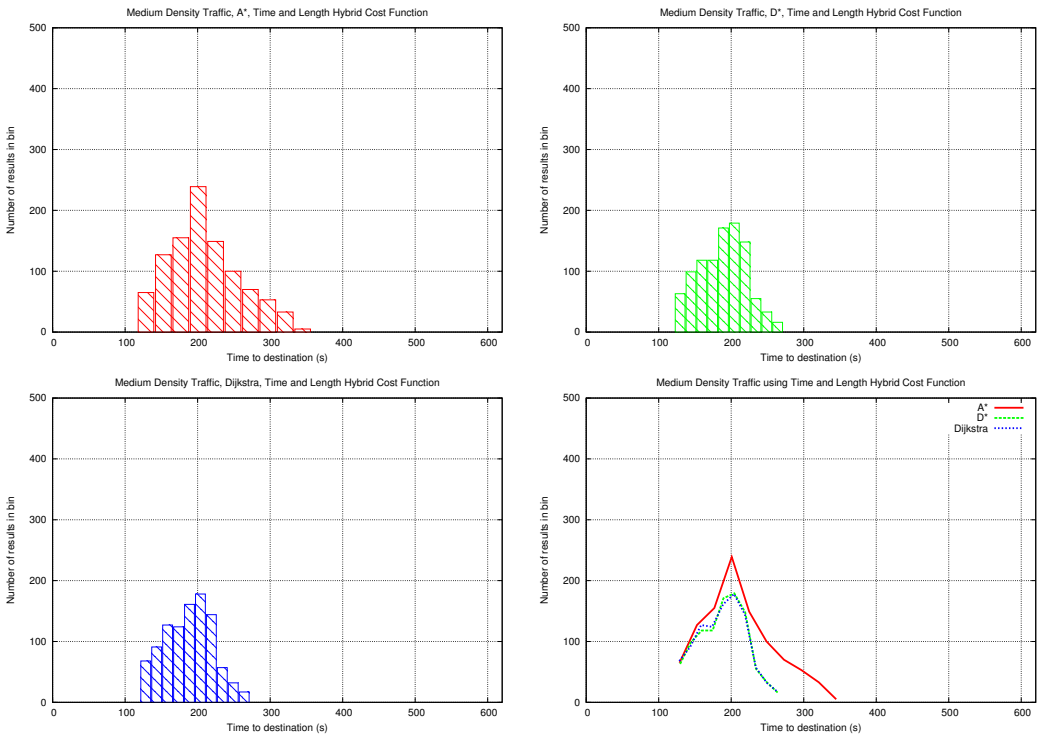

Figure 5.13: Medium Density Traffic using Time and Length Hybrid Cost Evaluation Function Distribution Histograms - in these histograms we see the distribution of the results for the different pathfinding algorithms using the time and length hybrid cost function in the medium density traffic conditions. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Just as we could expect by looking at the scatterplot of this case in Fig. 5.9, A* has a totally different distribution shape when compared to the other two algorithms which have almost overlapping distribution curves. We can also notice this occurrence by looking at the results in Tab. 5.1. For the graph key refer to Tab. 4.1

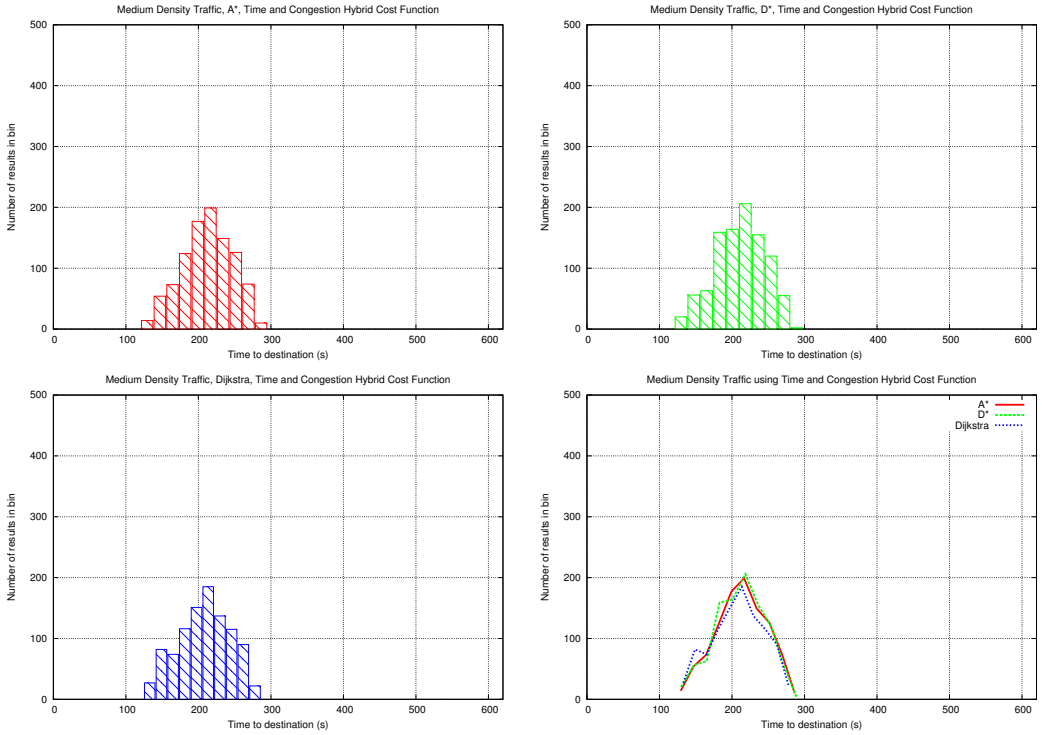

Figure 5.14: Medium Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Distribution Histograms - here we can see the results distribution curves of the various algorithms using the time and congestion hybrid cost function under medium density traffic conditions. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. While in the previous figure A* had a different distribution curve than D* and Dijkstra, in this case all of the algorithms perform very similarly and have the same distribution shape. We can confirm this by also looking at the results in Tab. 5.2 where we see that all the algorithms provide mean travel times very close to each other. For the graph key refer to Tab. 4.1

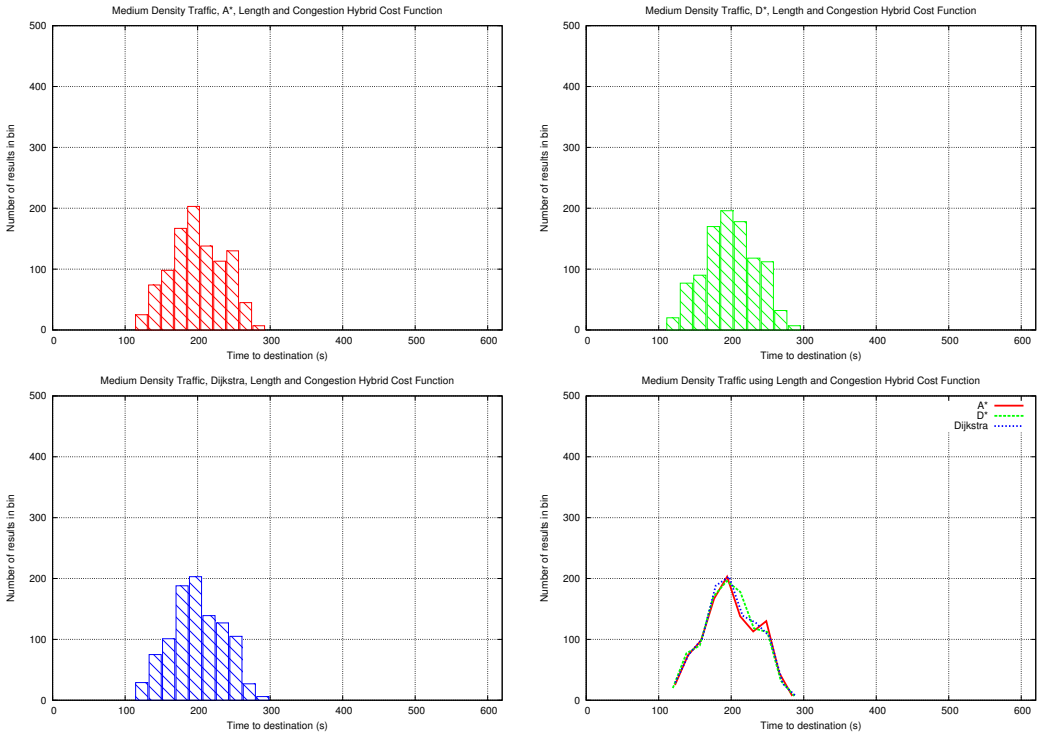

Figure 5.15: Medium Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution histograms for the different pathfinding algorithms using the length and congestion hybrid cost evaluation function in the medium density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Similarly to Fig. 5.14, we notice overlapping distribution curves by all the algorithms in this case, with a little difference in the distribution of A* where we see a small second peak of results. For the graph key refer to Tab. 4.1

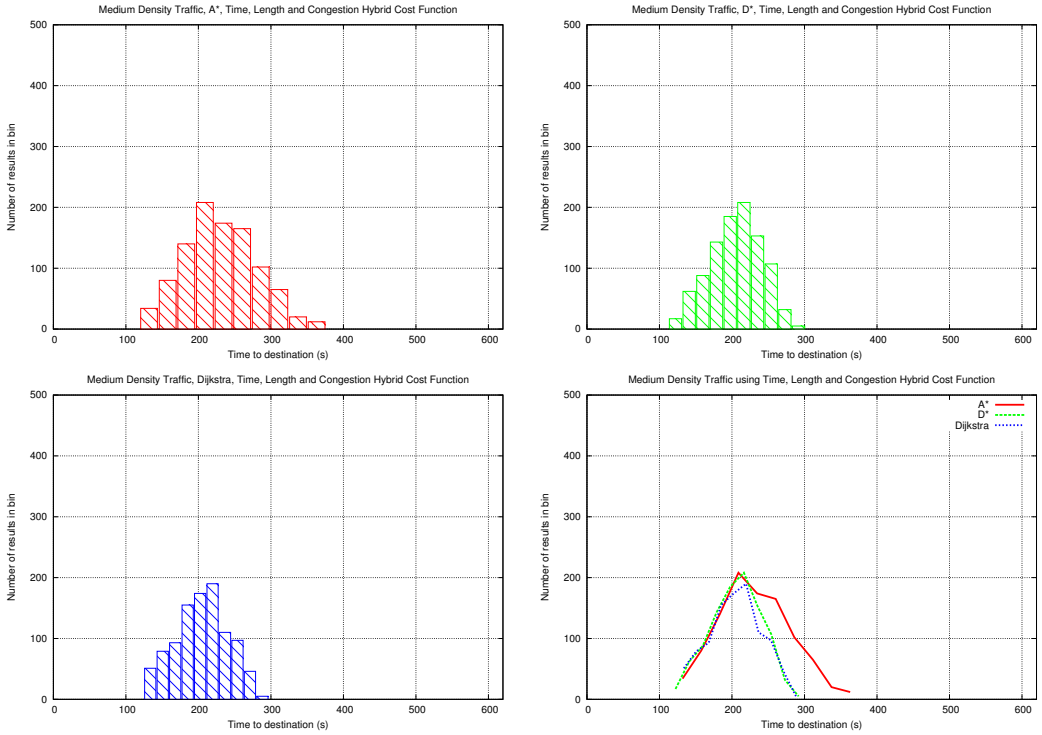

Figure 5.16: Medium Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution curves for the various pathfinding algorithms using the hybrid combination of all the basic cost evaluation functions under the medium density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Once again, as expected by looking at Fig. 5.12 where we see a major traffic congestion caused by A*, we can see the distribution of the results gathered by the A* algorithm to be much wider compared to the distributions of the other two algorithms. Therefore in this case we find the worst mean travel time under this traffic scenario. For the graph key refer to Tab. 4.1

	A*	D*	Dijkstra
Mean	283 s	285 s	284 s
Standard Deviation	31 s	51 s	41 s
Coefficient of Variation	10.80 %	17.88 %	14.49 %

Table 5.11: High Density Traffic using Time and Length Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	305 s	317 s	293 s
Standard Deviation	37 s	40 s	37 s
Coefficient of Variation	12.05 %	12.71 %	12.62 %

Table 5.12: High Density Traffic using Time and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	264 s	267 s	253 s
Standard Deviation	42 s	39 s	39 s
Coefficient of Variation	15.79 %	14.65 %	15.37 %

Table 5.13: High Density Traffic using Length and Congestion Hybrid Cost Function Statistical Results

	A*	D*	Dijkstra
Mean	281 s	281 s	295 s
Standard Deviation	33 s	33 s	32 s
Coefficient of Variation	11.89 %	11.91 %	10.93 %

Table 5.14: High Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Results

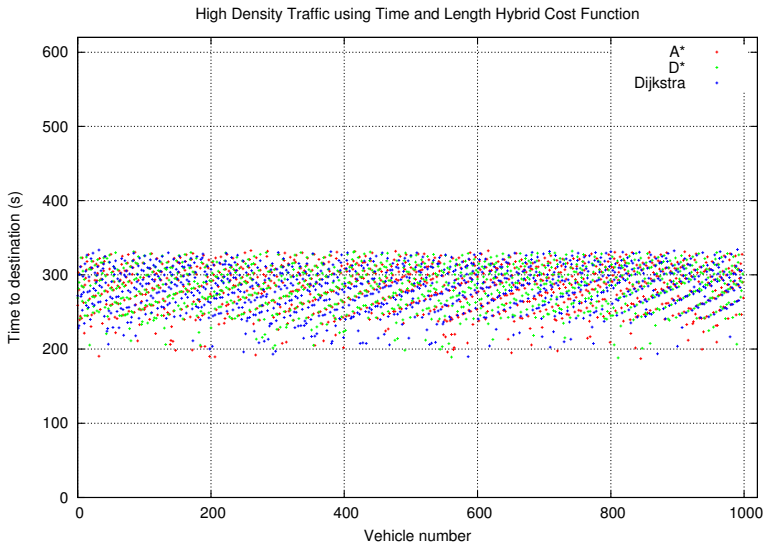

Figure 5.17: High Density Traffic using Time and Length Hybrid Cost Evaluation Function Scatterplot - here we can see the travel times of all 3,000 "smart" vehicles simulated in the high density traffic scenario using the time and length hybrid cost evaluation function. We can notice the usual pattern in the results caused by the traffic light intervals when the vehicles are not changing their route during the simulation. We can also notice here that there are no traffic congestions caused in the network in this case and all the algorithms are performing very similarly. For the graph key refer to Tab. 4.1

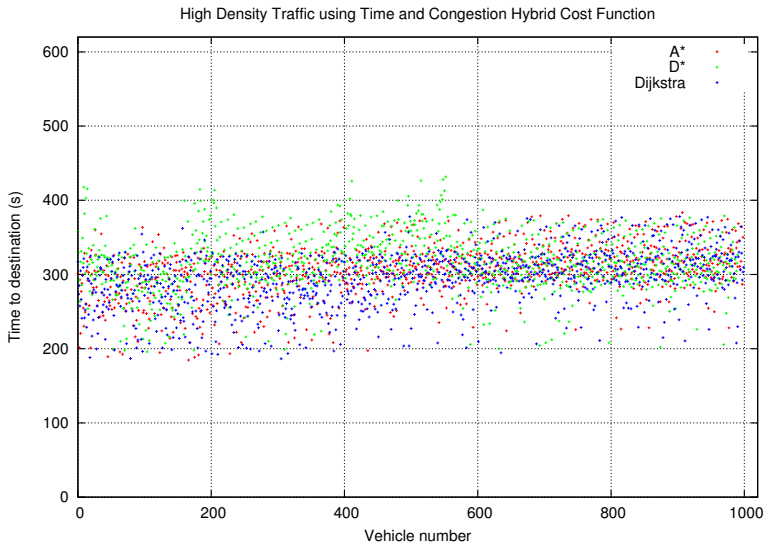

Figure 5.18: High Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Scatterplot - in this scatterplot we can see the travel times of the "smart" vehicles simulated in the rush hour scenario using the time and congestion hybrid cost evaluation function. It is interesting to see in this case that D* causes a few traffic congestions from the beginning to approximately the 600th simulated vehicle. We can predict from here that D* will have the highest mean travel time in this case. For the graph key refer to Tab. 4.1

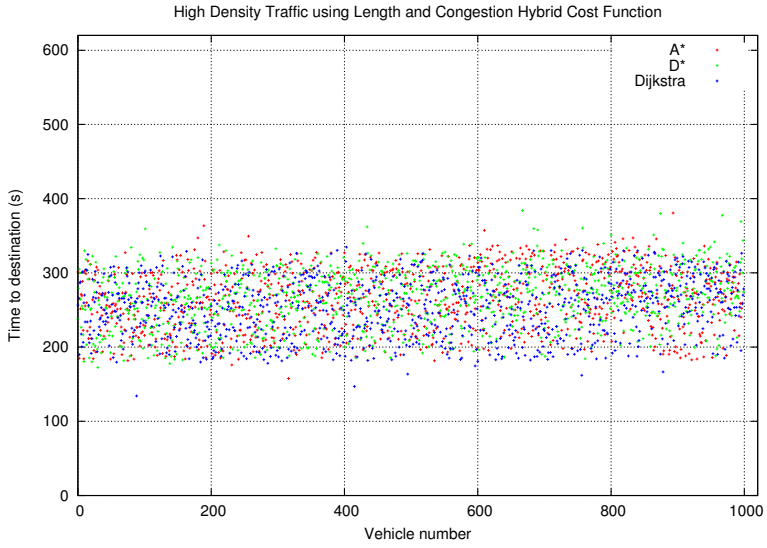

Figure 5.19: High Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Scatterplot - here we can see the results gathered under the rush hour traffic scenario using the length and congestion hybrid cost function combination. From this plot we cannot determine which one of the algorithms outperforms the others because all of them provide very similar results. For the graph key refer to Tab. 4.1

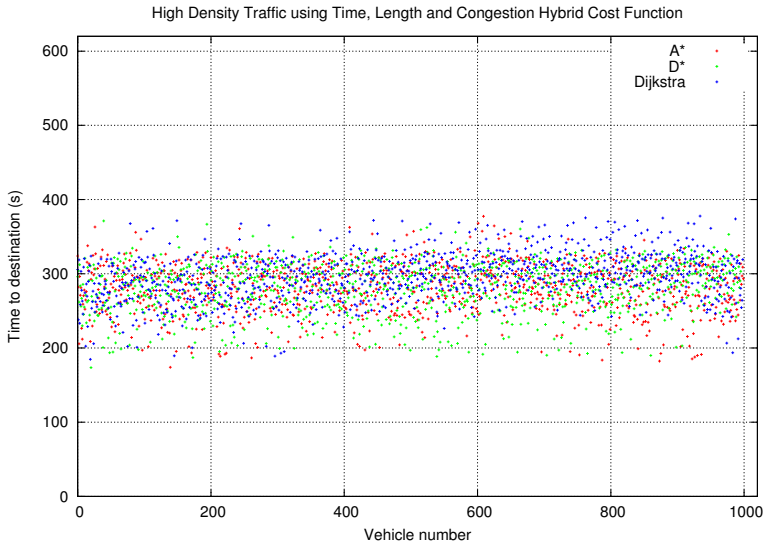

Figure 5.20: High Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Scatterplot - in this graph we can see the travel times of the vehicles simulated in the rush hour traffic scenario using the hybrid combination of all basic cost evaluation functions. We can notice in this graph that the algorithms start off performing similarly, however, somewhere around the 500th simulated vehicle the travel times for the agents using Dijkstra algorithm start increasing which is also confirmed by looking at the results in Tab. 5.14. For the graph key refer to Tab. 4.1

In this traffic scenario I observed very interesting results. If we refer to Tab. 5.11 we can see that all the algorithms performed very close to each other and have mean travel times within 3 s range of each other. This is somewhat expected knowing that the path of the "smart" vehicles in each case is the same and there is a steady traffic density with no congestion occurring during the simulations. Inspecting the results in Tab. 5.12 however, we can see the effect the traffic congestion caused by the D* algorithm had. Here we find the highest mean travel time with a value of 317 s. Tab. 5.13 is where we find the lowest mean travel times under this traffic scenario especially when using the Dijkstra algorithm where the mean is 253 s. In the last case using the time, length and congestion hybrid cost function in Tab. 5.14, we can detect a similar mean value between A* and D* algorithms but a much higher mean value using the Dijkstra algorithm. This is, as previously mentioned, due to the fact that there is a steady increase of the travel times of our agents throughout the simulation in the Dijkstra case.

	TL	TC	LC	All
Mean	284 s	305 s	261 s	286 s

Table 5.15: High Density Traffic Overall Statistical Results by Hybrid Cost Function

Looking at the overall mean travel times between the different hybrid cost evaluation functions in Tab. 5.15, we can see that the length and congestion hybrid function greatly outperforms the other ones in this traffic scenario with a mean travel time of 261 s while the second best is around 23 s higher which is a significant difference. What was more interesting in this case however, is that unlike in the previous two scenarios where the hybrid cost evaluation functions provided lower mean travel times for the driver agents, in this traffic scenario the hybrid functions perform generally worse than the basic cost functions. This will be discussed in more detail in the next subsection.

If we refer to the distribution histograms of this traffic scenario we can confirm all the points we previously discussed in this subsection. Looking at Fig. 5.21 we can detect the exact same distribution curve of all the algorithms which we already saw provided almost the same mean travel time. Furthermore in Fig. 5.22 we can observe a narrower distribution of the results, but with the peak being very close to the 300 s mark thus providing very high mean travel time of the "smart" vehicles in this case. In Fig. 5.23 on the other hand, we have a wider distribution with no particularly clear peak of results, but they vary in a lower range and therefore provide the lowest mean travel time for this traffic scenario. In Fig. 5.24, we have a similar distribution shape of all the algorithms again, however the curve of the results gathered using the Dijkstra algorithm is slightly to the right of the other two curves which is in fact there because of that increase of travel times of the agents using this algorithm in the Rush Hour traffic scenario.

5.2.4 Discussion

As already mentioned, before I conducted this case study I believed that the hybrid cost functions would provide better (lower) travel times for my intelligent driver agents because they considered more factors when evaluating the costs of taking a particular road. However, as we saw in the previous subsections this is not always the case.

Under low density traffic conditions the hybrid cost evaluation functions were performing better than the basic functions. Under medium density traffic conditions, again the hybrid cost functions have better performance, however in this case not by a large margin as in the low density traffic scenario. In the rush hour scenario on the other hand, the basic cost functions give lower travel times by a large margin. It was somewhat surprising, but we can predict this is the case because in the rush hour scenario the major factor is the congestion of the road, the less congested the road the faster the vehicle will get to its destination. That is

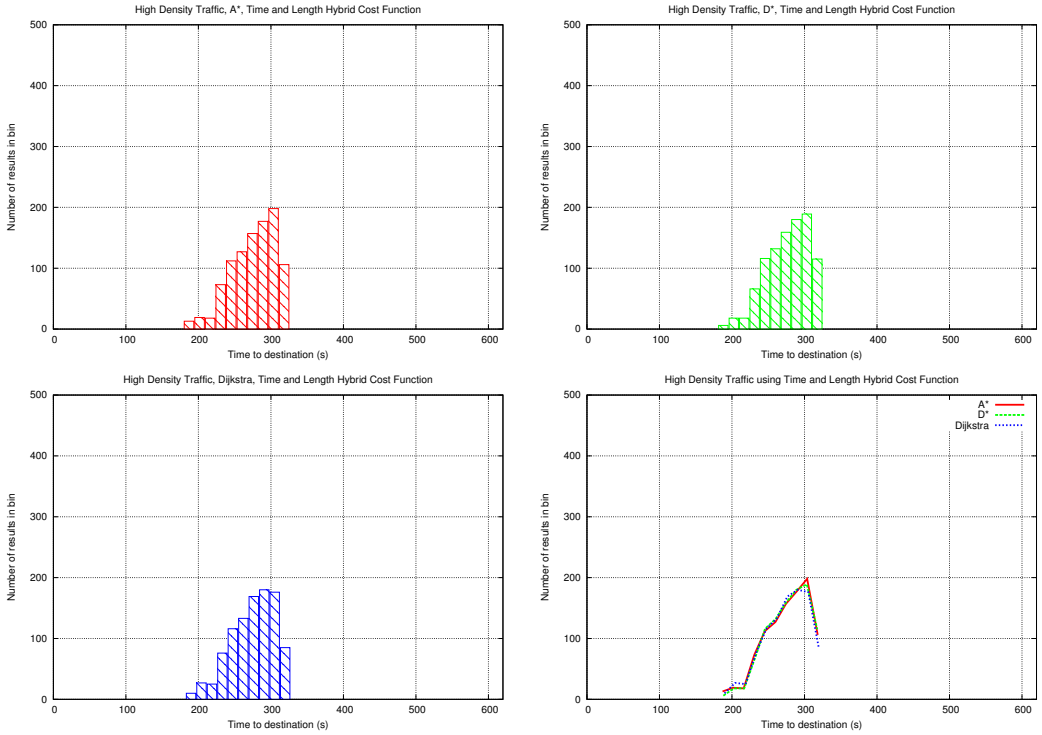

Figure 5.21: High Density Traffic using Time and Length Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution curves for the various pathfinding algorithms using the hybrid combination of the basic time and length cost evaluation functions under the high density traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. We can see that the distribution curves of all the algorithms are almost exactly the same. This is due to the fact that there is no traffic congestion caused by any of the algorithms in this case, as well as the fact that every “smart” vehicle take exactly the same path , which never changes when using this particular cost evaluation function. For the graph key refer to Tab. 4.1

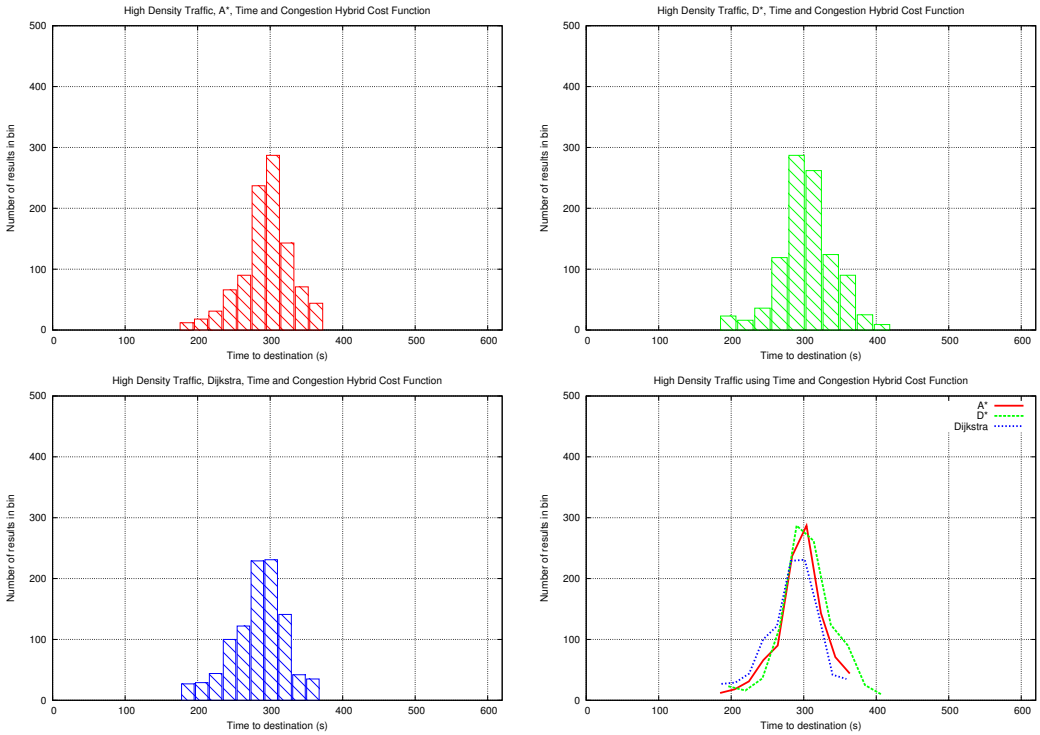

Figure 5.22: High Density Traffic using Time and Congestion Hybrid Cost Evaluation Function Distribution Histograms - in these histograms we can see the distribution curves of the different algorithms using the time and congestion hybrid cost evaluation function in the rush hour traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Here we can notice the normal-like distribution curve for all the algorithms, however generally wider in the case of the D* pathfinding algorithm with some results even in the 400 s range. This is also confirmed by looking at the mean travel time of D* in Tab. 5.12. For the graph key refer to Tab. 4.1

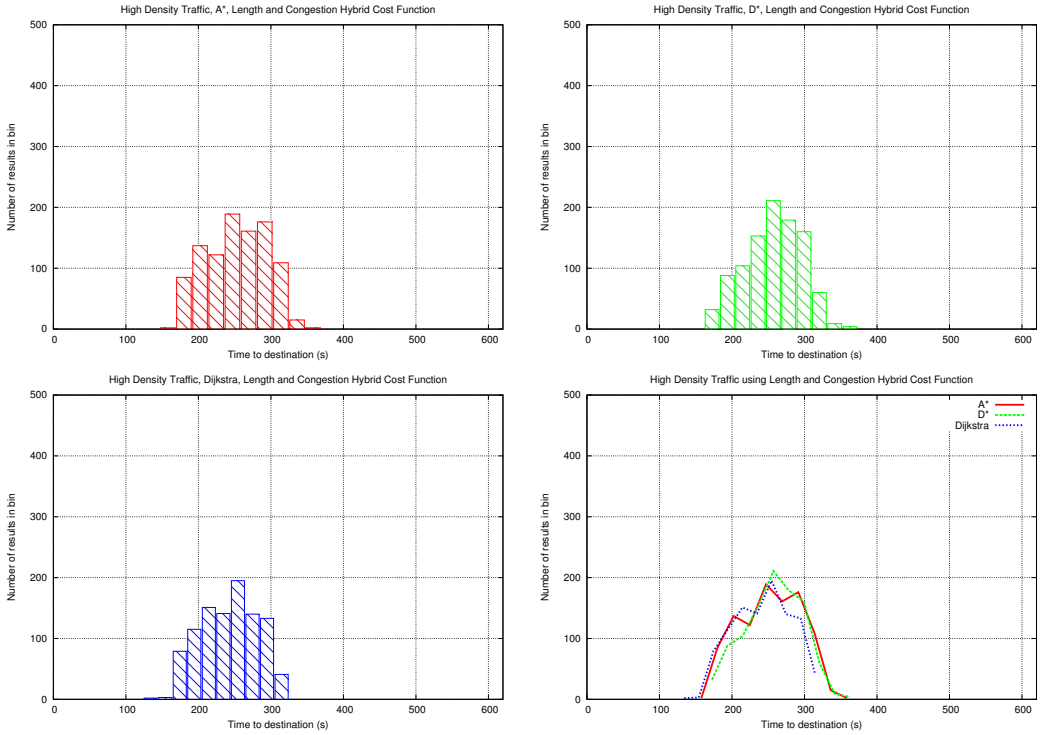

Figure 5.23: High Density Traffic using Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - in these graphs we see the distribution curves for the different algorithms using the hybrid length and congestion cost evaluation function under the rush hour traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. We notice in these histograms very interesting distribution curves for all the algorithms, and particularly important the results are distributed in a lower range when compared to the other histogram figures under this scenario. This notion is confirmed by looking at the results in Tab. 5.13 where we find the best mean travel times under this traffic scenario. For the graph key refer to Tab. 4.1

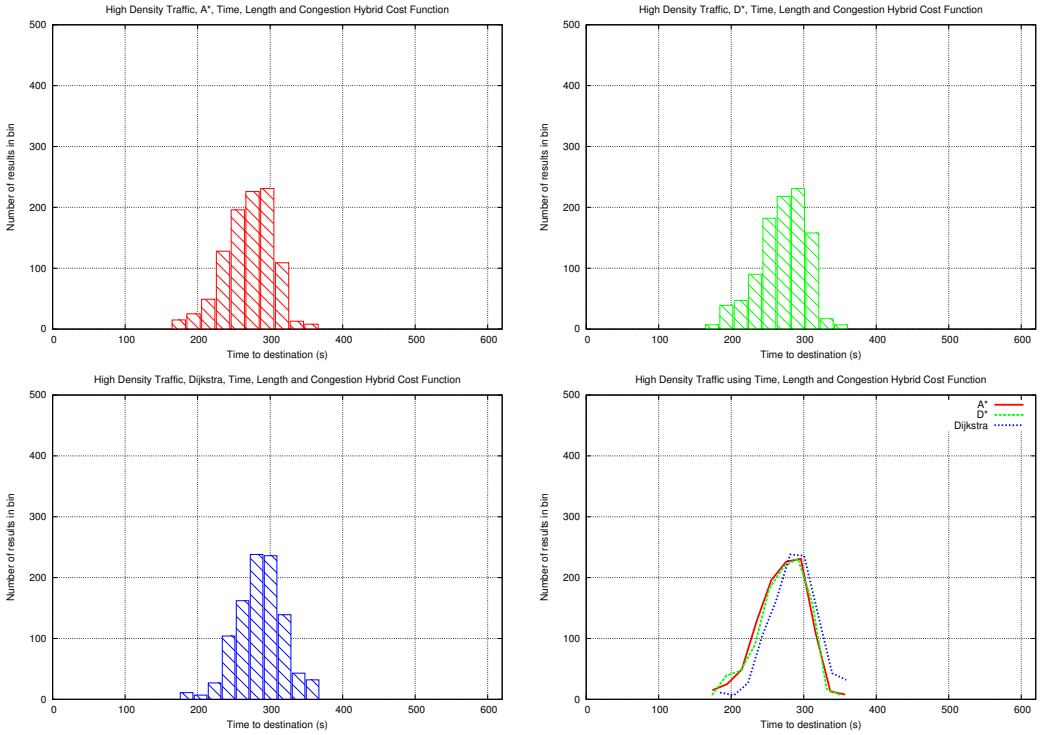

Figure 5.24: High Density Traffic using Time, Length and Congestion Hybrid Cost Evaluation Function Distribution Histograms - here we see the distribution histograms of the various pathfinding algorithms using the hybrid combination of all basic cost functions simulated in the rush hour traffic scenario. The first three graphs show the distributions separately using bins, while the graph in the bottom right corner shows them combined using lines. Here we can notice the distributions of A* and D* almost overlapping each other, however the Dijkstra pathfinding algorithm has the same distribution curve but slightly skewed to the right which brings it in the higher range of travel times. This is also confirmed by looking at the results in Tab. 5.14. For the graph key refer to Tab. 4.1

why the congestion basic cost evaluation function provides exceptionally lower travel times than any other cost function or combination of cost functions.

Having said that, in my "smart" agent proposal I decided to use the hybrid cost functions in low and medium density traffic, but the basic congestion cost function in high density traffic.

5.3 "Smart" Agent Model Proposal

The "smart" driver agent model I propose in this section takes the best combination of algorithm and cost evaluation function I have found for each traffic scenario during my case study. Currently the agent acts in accordance to the following set of rules:

- Under low density traffic scenario uses Dijkstra pathfinding combined with time and length hybrid cost function
- Under medium density traffic scenario uses Dijkstra pathfinding combined with time and length hybrid cost function
- Under rush hour traffic scenario uses D* pathfinding combined with congestion basic cost function

The way the agent adapts to the traffic conditions is by the information it gets from the traffic network. Each agent is aware of the number of vehicles currently using the traffic network and the capacity of the traffic network (see Sect. 6.2), thus each can calculate the density of traffic at any moment of time. The agent then uses this parameter to decide which algorithm and cost function combination it will apply.

Studying my findings however, we can see that there are cases where two or more of the algorithm and cost function combinations provide similar results for the travel times. This is one of the notions I will take in consideration in my future work when I plan to extend the "smart" driver agent model to use a wider range of options than the ones it currently uses. One of the popular ways to process uncertain data is using fuzzy logic, thus providing the agent with means to consider more options when deciding on what algorithm and cost function to use. Gerdelan [20, 21] has done extensive research on agent behaviour using A* and D* pathfinding and fuzzy logic hence combining my "smart" driver agent model with his fuzzy agent model would give an interesting combination for future investigation.

Perfect behavior is born of
complete indifference.

Cesare Pavese

6

Behaviour Analysis

6.1 Metrics

As already mentioned in Chapt. 2, I studied various literature on traffic systems and traffic engineering where I learned the different traffic measurements, their definitions, equations and relationships. In this chapter I discuss in more detail these traffic parameters, as well as present an analysis of the results obtained during the two case studies I presented in Chapt. 4 and Chapt. 5.

The following list is introduced by Taylor et al. in their Understanding Traffic Systems [46]; I have outlined the measurements, which form the basics for traffic analysis, and at the same time are of interest to us in determining and evaluating the different pathfinding algorithms in order to achieve the desired traffic flow within the network.

- Traffic Flow or Volume - Number of vehicles per unit time
- Speed - Time per unit length
- Concentration or Density - Vehicles per unit length

Additionally, Mannering et al. in their Principles of Highway Engineering and Traffic Analysis [31], provide mathematical equations and explanations of traffic parameters such as:

- Time Headway for a particular vehicle i - Time elapsed between the arrivals of vehicles i and i - 1
- Average Time Headway - the reciprocal of traffic flow or volume
- Spacing - Roadway length between successive vehicles
- Average Spacing - the reciprocal of traffic density
- Average Traffic Speed - Defined in two different ways:
  - Time-mean speed - the arithmetic mean of the vehicle speeds observed at some designated point along the roadway
  - Space-mean speed - determined on the basis of the time necessary for a vehicle to travel some known length of roadway

all of which are used to explain the relationship between the three basic measurements in more detail.

Taylor et al. also mention the relationship between the three basic parameters that are used to describe a traffic stream. They present this relationship in the following equation, which is also derived from the equations given in the work of Mannering et al. and Haberman in his Mathematical Models [22].

$$q = k\bar{v}_s \tag{6.1}$$

Eq. 6.1 gives us the traffic flow q as a product of the density k and the speed $\bar{v}_s$ (space-mean speed). Additionally, Mannering et al. give a categorization of the metrics in order to associate them with the appropriate level of investigation. They refer to average time headway and average spacing as microscopic measurements, while metrics such as traffic flow, average speed and traffic density are said to be macroscopic measurements. [31] They also indicate that the microscopic measures can be related to the macroscopic measures as given in their study on traffic measurements, and also supported by the work of Taylor et al. and Haberman.

However, Eq. 6.1 generally applies only to uninterrupted traffic, such as traffic flows on main highways and freeways, nonetheless it is noted by Mannering et al. that it is also applicable to interrupted traffic flow, but that there are additional complexities involved.

They go further to explain the key concepts and measures used in the analysis of interrupted traffic flow such as at a signalized intersection. Following is a list of measures, as given by Mannering et al. [31].

- Saturation Flow Rate - the maximum hourly volume that can pass through an intersection, from a given lane or group of lanes, if that lane (or lanes) were allocated constant green light over the course of an hour
- Lost Time - the portion of the cycle length that is not being completely utilized. In other terms, because the traffic light alternates the right-of-way between conflicting movements, traffic flows are continuously started and stopped. Every time this happens there is a lag due to drivers reacting to the change of traffic light signal. This lag at the beginning and the end of green and yellow signal interval results in a portion of that interval to not be completely utilized. This lag is known as lost time
- The Effective Green Time - the time during which a traffic movement is effectively utilizing the intersection
- The Effective Red Time - the time during which a traffic movement is not effectively utilizing the intersection
- Capacity - the maximum hourly volume that can pass through an intersection from a lane or group of lanes under prevailing roadway, traffic, and control conditions

$$c = s\frac{g}{C} \tag{6.2}$$

Eq. 6.2 gives capacity c (usually in vehicles per hour) as the product of saturation flow rate s (vehicles per hour) and the ratio between effective green time g and the cycle time C (both of which usually given in seconds). Therefore, adjusting the traffic light interval can influence the capacity of a certain intersection and bring it to the desired level so it can keep up with the traffic inflow passing through the intersection.

6.2 Discussion

After a study of the general measurements and parameters associated with traffic, I decided that in my case studies I will examine only the basic measurements and inspect what happens

to each of them under different traffic conditions. By neglecting the measurements associated with signalized intersections my results for the basic traffic parameters are not accurate, but as noted before they can still be used to give a general idea of the overall behaviour of the traffic within the network. I also plan to investigate the measurements related to signalized intersections in my future research.

As already mentioned in Chapt. 2, I used the basic traffic parameters to control the traffic conditions in the simulated network when I performed the case studies on my intelligent driver agents. In order to simulate varied traffic conditions I needed to control one of these basic parameters. The measurement for traffic density made most sense in this case because controlling the density I could make sure I have control over the number of vehicles that are using the traffic network. This way, I could come up with various traffic scenarios and I could conduct my case studies of the different algorithm and cost evaluation functions combinations under these scenarios.

Controlling the traffic density I came up with the three different traffic conditions in which I conducted my case studies:

1. Low Density Traffic - number of simulated vehicles equal to 10% of the total capacity of the traffic network
2. Medium Density Traffic - number of simulated vehicles equal to 29% of the total capacity of the traffic network
3. High Density Traffic (Rush Hour) - number of vehicles equal to 68% of the total capacity of the traffic network

where the capacity of the traffic network was defined using Eq. 6.3.

$$\text{capacity} = \frac{\text{cumulative length of all roads}}{\text{length of a vehicle}} \tag{6.3}$$

The density percentages were chosen arbitrarily at first, and then they were tweaked to get the ones used in the case studies. Low density traffic is very straight forward. I needed a scenario with some traffic to examine the influence it has on the intelligent driver agents. Medium density conditions were chosen at 29% so that I can investigate the affect nearly tripling the number of vehicles would have. Rush Hour traffic on the other hand was a little bit harder to come up with. I started with 80% of the total capacity of the traffic network,

but this proved to be too much and intersections were getting clogged up with vehicles all around the network, so I slowly lowered the density until I came to 68% which I used in my case study.

Before doing the case studies, I expected to receive different average travel times in the different scenarios. I expected the lowest travel time to be recorded under low density traffic conditions and the highest to be recorded under the rush hour scenario. After conducting the case studies, we can see that my results confirm my expectations. The driver agents behaviour in this case is logical knowing that in the high density traffic scenario there is more congestion and therefore we experience higher travel times and lower average speed.

$$\text{average speed} = \frac{\text{average distance}}{\text{average time}} \tag{6.4}$$

Furthermore, Eq. 6.4 is the equation I used to determine the average speed of my driver agents. In accordance with the expectation I had for the traffic density effect on the travel times of my agents, in the case of the average speed measurement I expected the higher the traffic density the lower the average speed of my agents to be. We can see this occurring in the results shown in the tables included in this section, where we see the highest average speed recorded of 47 km/h in the low density traffic scenario in Tab. 6.4 and the lowest average speed of 26 km/h in the rush hour scenario in Tab. 6.6.

	Time	Length	Congestion
Average Distance	2,179 m	2,126 m	2,192 m
Average Time	192 s	193 s	201 s
Average Speed	41 km/h	40 km/h	39 km/h
Density	10%	10%	10%
Traffic Flow	4.08 veh/h	3.97 veh/h	3.92 veh/h

Table 6.1: Low Density Traffic Metrics Analysis by Basic Cost Function

	Time	Length	Congestion
Average Distance	2,179 m	2,126 m	2,192 m
Average Time	218 s	207 s	209 s
Average Speed	36 km/h	37 km/h	38 km/h
Density	29%	29%	29%
Traffic Flow	10.44 veh/h	10.73 veh/h	10.93 veh/h

Table 6.2: Medium Density Traffic Metrics Analysis by Basic Cost Function

The last measurement I investigated was traffic flow. According to Eq. 6.1, I expected to get the highest traffic flow in the scenario where I have the highest average speed. Therefore, I thought that in the low density scenario I will have the highest traffic flow. However this

	Time	Length	Congestion
Average Distance	2,179 m	2,126 m	2,192 m
Average Time	291 s	257 s	255 s
Average Speed	27 km/h	30 km/h	31 km/h
Density	68%	68%	68%
Traffic Flow	18.33 veh/h	20.26 veh/h	21.05 veh/h

Table 6.3: High Density Traffic Metrics Analysis by Basic Cost Function

was not the case because traffic flow is also proportional to the traffic density, so the lower the density the lower the traffic flow. We can see this occurring in the results gathered in my case studies. I recorded the highest traffic flow of 21.05 vehicles/h under the high density traffic conditions in Tab. 6.3 and the lowest traffic flow of 3.91 vehicles/h under low density traffic conditions in Tab. 6.4.

Additionally, it is important to note that similarly to Chapt. 4 and Chapt. 5 the results in the tables in this section are rounded, however the more accurate data tables are also included in Appx. A.

	TL	TC	LC	All
Average Distance	2,129 m	2,179 m	2,126 m	2,129 m
Average Time	163 s	201 s	192 s	184 s
Average Speed	47 km/h	39 km/h	40 km/h	42 km/h
Density	10%	10%	10%	10%
Traffic Flow	4.70 veh/h	3.91 veh/h	3.99 veh/h	4.16 veh/h

Table 6.4: Low Density Traffic Metrics Analysis by Hybrid Cost Function

	TL	TC	LC	All
Average Distance	2,129 m	2,179 m	2,126 m	2,129 m
Average Time	204 s	219 s	208 s	224 s
Average Speed	37 km/h	36 km/h	37 km/h	34 km/h
Density	29%	29%	29%	29%
Traffic Flow	10.87 veh/h	10.40 veh/h	10.66 veh/h	9.93 veh/h

Table 6.5: Medium Density Traffic Metrics Analysis by Hybrid Cost Function

	TL	TC	LC	All
Average Distance	2,129 m	2,179 m	2,126 m	2,129 m
Average Time	284 s	305 s	261 s	286 s
Average Speed	27 km/h	26 km/h	29 km/h	27 km/h
Density	68%	68%	68%	68%
Traffic Flow	18.36 veh/h	17.49 veh/h	19.92 veh/h	18.25 veh/h

Table 6.6: High Density Traffic Metrics Analysis by Hybrid Cost Function

Now I can see a time when I can dial a taxi and it will show up and take me where I want to go, and I won't have to tip it.

Vaughan Pratt

7 Conclusion

7.1 Summary

I started this research on a quest to find an answer to the question "Is there a viable way to automate vehicular traffic, and by doing so, decrease traffic congestion?" After an extensive literature and software review, and implementation of a microscopic traffic simulation with intelligent driver agents, I was able to find a few answers to my questions on the topic.

First and foremost I asked myself: is there a viable way to automate vehicular traffic? My findings indicate that this is definitely possible. People have already created self-driving vehicles and had them autonomously drive through traffic in challenges such as the DARPA Challenge [10, 11, 12] and VIAC [52]. Although the so called robot vehicles used in these challenges are still not commercially available, most of their technology is. Therefore, we can say that driverless vehicles are no more science fiction, they do exist, and we might be able to see them on our streets in the future.

Furthermore, the traffic radar used in the smart vehicles in [24] is also available. Being able to share information through a wireless network among all the vehicles currently driving is very powerful. Our driverless cars of the future can use this information to intelligently navigate through the traffic network to our destination.

Lastly, a central navigation system used by FROG AVG Systems [18] could be employed to gather the knowledge of all the vehicles, quickly combine the relevant information in small packages and relay that information back to the vehicles that demand it.

Combining these three concepts and implementing my intelligent driver agent into the robot vehicles could essentially create a self sustainable automatic vehicular traffic network.

The second part of my question remains. Will automating vehicular traffic decrease traffic congestion? My findings indicate that traffic congestion can be decreased with a well designed automatic vehicular traffic system, but cannot be totally eliminated. We have seen that each traffic network has its own capacity. If we overwhelm the network with vehicles, traffic congestion is unavoidable. This is essentially what happens in the big cities today. We have a traffic network designed to support a particular number of vehicles, but during certain periods of the day the number of vehicles that demand to use this traffic network is a multiple of the capacity of the network. As we saw in the Rush Hour traffic scenario in my study case, when we increase the traffic density (i. e. the number of vehicles using the traffic network) the average speed of the vehicles decreases. This leads to traffic congestion occurring at different instances within the network. While my intelligent driver agents find various ways to navigate through a traffic network in a timely manner, I still experienced some traffic congestion in my simulations.

The only way to successfully eliminate traffic congestions from occurring is to decrease the number of vehicles using the network. While my work shows this issue, it does not solve it because I have used constant traffic density in my case studies. In order to decrease the number of vehicles using a certain traffic network we would have to start designing our future traffic systems in the manner Michael Arth proposes [2]. If we changed our privately owned vehicles with self-driving public taxis in a perfectly automated traffic network, we would decrease the number of vehicles demanding to use the traffic network and therefore eliminate traffic congestion.

7.2 Future Research Directions

My foremost goal in the future work in this area will be to implement more cost evaluation functions and more pathfinding algorithms that my intelligent driver agents can use. Investigating a variety of options through simulations and case studies is key to determine the best options and combinations to use under different traffic conditions. This will require implementing a non-graphical simulation of traffic in my microsimulation software so that it can gather a larger amount of results in a shorter amount of time.

Another important step is to extend my intelligent driver agents to use fuzzy logic so

that they have means to process uncertain data and consider more than a few options of algorithm and cost evaluation function combinations to use under each traffic scenario.

Furthermore, I plan to implement ways to indicate areas in the simulation system that could attract more vehicular activities such as shopping centres, schools etc. as well as ways to add vehicles travelling at fixed schedules to simulate buses and other public transports. These would help improve the realism of the traffic simulation and therefore provide a more realistic set of results that could be further studied.

A more distant step is to work on an intelligent traffic lights system that can be combined with the driver agents in order to create harmonious automatic traffic. Intelligent traffic lights and adaptable traffic light intervals will definitely have great impact on the overall behaviour of the my driver agents, and I believe will improve their travel times even further. However, I still doubt there will be even need for traffic lights if all the vehicles in a traffic network are controlled by intelligent driver agents. More suitably, a "smart" give-way system can be designed for each agent that will coordinate its behaviour at every intersection.

To conclude, I believe that the intelligent driver agent model proposed in this work is suitable to be used as a navigation model for self-driving vehicles in traffic simulation software. I also feel that it can be implemented as a navigation model for robot vehicles and with the right technology can be studied in the outside world. The descriptive design of the proposed intelligent driver agent model allows it to be implemented in virtually any programming language, thus giving us freedom to further experiment with it in different traffic simulation software as well as real world traffic environments.

A

Accurate Data Tables

As previously mentioned in Chapt. 4, Chapt. 5 and Chapt. 6 my traffic simulation software records travel times in milliseconds, however for simplicity purposes the results were rounded and given in seconds. In the following sections we can see the accurate travel time data gathered in the case studies, as well as the traffic parameters analysis in both the basic and hybrid cost function cases.

A.1 Low Density Traffic Basic Cost Functions

	A*	D*	Dijkstra
Mean	194,025.51 ms	167,082.64 ms	215,314.14 ms
Standard Deviation	23,649.41 ms	23,108.82 ms	26,870.42 ms
Coefficient of Variation	12.19 %	13.83 %	12.48 %

Table A.1: Low Density Traffic using Time Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	202,607.53 ms	189,158.83 ms	186,250.39 ms
Standard Deviation	35,412.97 ms	27,637.62 ms	25,514.09 ms
Coefficient of Variation	17.48 %	14.61 %	13.70 %

Table A.2: Low Density Traffic using Length Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	192,380.46 ms	200,799.82 ms	210,520.60 ms
Standard Deviation	39,878.47 ms	41,732.61 ms	49,384.77 ms
Coefficient of Variation	20.73 %	20.78 %	23.46 %

Table A.3: Low Density Traffic using Congestion Cost Function Statistical Restuls

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	192,140.76 ms	192,672.25 ms	201,233.63 ms

Table A.4: Low Density Traffic Overall Statistical Results by Cost Function

A.2 Medium Density Traffic Basic Cost Functions

	A*	D*	Dijkstra
Mean	227,165.40 ms	195,997.27 ms	230,856.13 ms
Standard Deviation	28,740.89 ms	32,544.37 ms	29,067.95 ms
Coefficient of Variation	12.65 %	16.60 %	12.59 %

Table A.5: Medium Density Traffic using Time Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	210,168.92 ms	206,010.91 ms	204,483.57 ms
Standard Deviation	28,530.37 ms	26,014.3 ms	26,258.08 ms
Coefficient of Variation	13.57 %	12.63 %	12.84 %

Table A.6: Medium Density Traffic using Length Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	209,517.56 ms	209,808.76 ms	208,818.14 ms
Standard Deviation	39,784.27 ms	41,832.56 ms	40,540.73 ms
Coefficient of Variation	18.99 %	19.94 %	19.41 %

Table A.7: Medium Density Traffic using Congestion Cost Function Statistical Restuls

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	218,006.27 ms	206,887.8 ms	209,381.49 ms

Table A.8: Medium Density Traffic Overall Statistical Results by Cost Function

A.3 High Density Traffic Basic Cost Functions

	A*	D*	Dijkstra
Mean	287,133.04 ms	273,806.23 ms	312,345.79 ms
Standard Deviation	32,858.49 ms	29,476.4 ms	43,342.17 ms
Coefficient of Variation	11.44 %	10.77 %	13.88 %

Table A.9: High Density Traffic using Time Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	256,475.21 ms	254,587.99 ms	259,593.53 ms
Standard Deviation	33,002.24 ms	38,896.07 ms	33,207.92 ms
Coefficient of Variation	12.87 %	15.28 %	12.79 %

Table A.10: High Density Traffic using Length Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	265,106.39 ms	247,434.53 ms	252,205.99 ms
Standard Deviation	54,731.57 ms	43,331.22 ms	41,810.09 ms
Coefficient of Variation	20.65 %	17.51 %	16.58 %

Table A.11: High Density Traffic using Congestion Cost Function Statistical Restuls

	Time Cost Function	Length Cost Function	Congestion Cost Function
Mean	291,095.02 ms	256,885.58 ms	254,915.64 ms

Table A.12: High Density Traffic Overall Statistical Results by Cost Function

A.4 Low Density Traffic Hybrid Cost Functions

	A*	D*	Dijkstra
Mean	163,508.11 ms	163,181.56 ms	163,093.30 ms
Standard Deviation	23,422.37 ms	23,042.76 ms	22,946.89 ms
Coefficient of Variation	14.32 %	14.12 %	14.07 %

Table A.13: Low Density Traffic using Time and Length Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	200,619.65 ms	200,363.56 ms	201,655.20 ms
Standard Deviation	35,240.26 ms	36,867.85 ms	37,481.26 ms
Coefficient of Variation	17.57 %	18.40 %	18.59 %

Table A.14: Low Density Traffic using Time and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	197,381.72 ms	188,371.28 ms	189,484.47 ms
Standard Deviation	35,836.63 ms	31,654.21 ms	31,211.39 ms
Coefficient of Variation	18.16 %	16.80 %	16.47 %

Table A.15: Low Density Traffic using Length and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	198,184.14 ms	177,379.55 ms	176,952.10 ms
Standard Deviation	49,176.21 ms	33,784.05 ms	32,510.13 ms
Coefficient of Variation	24.81 %	19.05 %	18.37 %

Table A.16: Low Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Restuls

	TL	TC	LC	All
Mean	163,260.99 ms	200,879.47 ms	191,745.83 ms	184,171.93 ms

Table A.17: Low Density Traffic Overall Statistical Results by Hybrid Cost Function

A.5 Medium Density Traffic Hybrid Cost Functions

	A*	D*	Dijkstra
Mean	222,029.85 ms	196,014.95 ms	195,366.92 ms
Standard Deviation	49,807.15 ms	31,985.49 ms	32,911.66 ms
Coefficient of Variation	22.43 %	16.32 %	16.85 %

Table A.18: Medium Density Traffic using Time and Length Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	220,497.04 ms	219,714.25 ms	216,094.33 ms
Standard Deviation	34,336.41 ms	33,873.84 ms	35,582.43 ms
Coefficient of Variation	15.57 %	15.42 %	16.47 %

Table A.19: Medium Density Traffic using Time and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	208,584.85 ms	207,773.99 ms	208,376.34 ms
Standard Deviation	36,338.17 ms	36,228.24 ms	36,188.46 ms
Coefficient of Variation	17.42 %	17.44 %	17.37 %

Table A.20: Medium Density Traffic using Length and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	243,585.20 ms	214,685.06 ms	213,182.11 ms
Standard Deviation	49,826.69 ms	35,121.41 ms	35,472.45 ms
Coefficient of Variation	20.46 %	16.36 %	16.46 %

Table A.21: Medium Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Restuls

	TL	TC	LC	All
Mean	204,470.57 ms	218,768.54 ms	208,245.06 ms	223,817.46 ms

Table A.22: Medium Density Traffic Overall Statistical Results by Hybrid Cost Function

A.6 High Density Traffic Hybrid Cost Functions

	A*	D*	Dijkstra
Mean	282,886.63 ms	285,234.10 ms	283,824.63 ms
Standard Deviation	30,556.52 ms	51,006.91 ms	41,139.10 ms
Coefficient of Variation	10.80 %	17.88 %	14.49 %

Table A.23: High Density Traffic using Time and Length Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	304,720.10 ms	317,175.74 ms	293,214.86 ms
Standard Deviation	36,710.03 ms	40,308.97 ms	37,014.92 ms
Coefficient of Variation	12.05 %	12.71 %	12.62 %

Table A.24: High Density Traffic using Time and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	263,920.71 ms	267,289.25 ms	252,626.69 ms
Standard Deviation	41,666.41 ms	39,148.24 ms	38,821.33 ms
Coefficient of Variation	15.79 %	14.65 %	15.37 %

Table A.25: High Density Traffic using Length and Congestion Hybrid Cost Function Statistical Restuls

	A*	D*	Dijkstra
Mean	280,561.78 ms	280,795.88 ms	295,329.99 ms
Standard Deviation	33,367.00 ms	33,455.20 ms	32,2656.66 ms
Coefficient of Variation	11.89 %	11.91 %	10.93 %

Table A.26: High Density Traffic using Time, Length and Congestion Hybrid Cost Function Statistical Restuls

	TL	TC	LC	All
Mean	283,981.79 ms	305,036.90 ms	261,278.88 ms	285,562.55 ms

Table A.27: High Density Traffic Overall Statistical Results by Hybrid Cost Function

A.7 Metrics Analysis Basic Cost Functions

	Time	Length	Congestion
Average Distance	2,179.43 m	2,126.41 m	2,192.07 m
Average Time	192,140.76 ms	192,672.25 ms	201,233.63 ms
Average Speed	40.83 km/h	39.73 km/h	39.22 km/h
Density	10%	10%	10%
Traffic Flow	4.08 veh/h	3.97 veh/h	3.92 veh/h

Table A.28: Low Density Traffic Metrics Analysis by Basic Cost Function

	Time	Length	Congestion
Average Distance	2,179.43 m	2,126.41 m	2,192.07 m
Average Time	218,006.27 ms	206,887.80 ms	209,381.49 ms
Average Speed	35.99 km/h	37.00 km/h	37.69 km/h
Density	29%	29%	29%
Traffic Flow	10.44 veh/h	10.73 veh/h	10.93 veh/h

Table A.29: Medium Density Traffic Metrics Analysis by Basic Cost Function

	Time	Length	Congestion
Average Distance	2,179.43 m	2,126.41 m	2,192.07 m
Average Time	291,095.02 ms	256,885.58 ms	254,915.64 ms
Average Speed	26.95 km/h	29.80 km/h	30.96 km/h
Density	68%	68%	68%
Traffic Flow	18.33 veh/h	20.26 veh/h	21.05 veh/h

Table A.30: High Density Traffic Metrics Analysis by Basic Cost Function

A.8 Metrics Analysis Hybrid Cost Functions

	TL	TC	LC	All
Average Distance	2,129.35 m	2,179.43 m	2,126.41 m	2,129.35 m
Average Time	163,260.99 ms	200,879.47 ms	191,745.83 ms	184,171.93 ms
Average Speed	46.95 km/h	39.06 km/h	39.92 km/h	41.62 km/h
Density	10%	10%	10%	10%
Traffic Flow	4.70 veh/h	3.91 veh/h	3.99 veh/h	4.16 veh/h

Table A.31: Low Density Traffic Metrics Analysis by Hybrid Cost Function

	TL	TC	LC	All
Average Distance	2,129.35 m	2,179.43 m	2,126.41 m	2,129.35 m
Average Time	204,470.57 ms	218,768.54 ms	208,245.06 ms	223,817.46 ms
Average Speed	37.49 km/h	35.86 km/h	36.76 km/h	34.25 km/h
Density	29%	29%	29%	29%
Traffic Flow	10.87 veh/h	10.40 veh/h	10.66 veh/h	9.93 veh/h

Table A.32: Medium Density Traffic Metrics Analysis by Hybrid Cost Function

	TL	TC	LC	All
Average Distance	2,129.35 m	2,179.43 m	2,126.41 m	2,129.35 m
Average Time	283,981.79 ms	305,036.9 ms	261,278.88 ms	285,562.55 ms
Average Speed	26.99 km/h	25.72 km/h	29.30 km/h	26.84 km/h
Density	68%	68%	68%	68%
Traffic Flow	18.36 veh/h	17.49 veh/h	19.92 veh/h	18.25 veh/h

Table A.33: High Density Traffic Metrics Analysis by Hybrid Cost Function

Bibliography

[1] 2getthere, "Automated People Moving Systems Website," 2010. [Online]. Available: http://www.2getthere.eu/

[2] M. E. Arth, "New urbanism and new pedestrianism in the 21st century," in *Congress for the New Urbanism XVI*, vol. CNU XVI, 2008.

[3] S. Ashley. (1998, May) Smart cars and automated highways. Mechanical Engineering Magazine. [Online]. Available: http://www.memagazine.org/backissues/membersonly/may98/features/smarter/smarter.html

[4] M. Balmer, N. Cetin, K. Nagel, and B. Raney, "Towards truly agent-based traffic and mobility simulations," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 60–67.

[5] A. Birch. (2008, May) Most Cars Can Be Eliminated In 20 years, Says Urban Designer Michael E. Arth. Corrupt. [Online]. Available: http://www.corrupt.org/news/most_cars_can_be_eliminated_in_20_years_says_urban_designer_michael_e_arth

[6] R. Bishop. (2001, August) Whatever Happened to Automated Highway Systems (AHS)? Traffic Technology International. [Online]. Available: http://faculty.washington.edu/jbs/itrans/bishopahs.htm

[7] Centre for Applied Informatics (ZAIK) and the Institute of Transport Research at the German Aerospace Centre, "Simulation of Urban Mobility - SUMO Website," 2009. [Online]. Available: http://sumo.sourceforge.net/

[8] S. Cheon. (2010) An Overview of Automated Highway Systems (AHS) and the Social and Institutional Challenges they Face. The University of California Transportation Center (UCTC). [Online]. Available: http://www.uctc.net/papers/624.pdf

[9] B. C. da Silva, R. Junges, D. de Oliveira, and A. L. C. Bazzan, "Itsumo: an intelligent transportation system for urban mobility," in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2006, pp. 1471–1472.

[10] DARPA, "DARPA Grand Challenge 2004 Website," 2004. [Online]. Available: http://www.darpa.mil/grandchallenge04/

[11] ——, "DARPA Grand Challenge 2005 Website," 2005. [Online]. Available: http://www.darpa.mil/grandchallenge05/

[12] ——, "DARPA Urban Challenge 2007 Website," 2007. [Online]. Available: http://www.darpa.mil/grandchallenge/index.asp

[13] ——, "DARPA Website," 2010. [Online]. Available: http://www.darpa.mil

[14] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[15] European Commission Information Society, "i2010: Intelligent Car," 2010. [Online]. Available: http://ec.europa.eu/information_society/activities/intelligentcar/docs/icar/intelligent_car.pdf

[16] ——, "Intelligent Car Initiative Website," 2010. [Online]. Available: http://ec.europa.eu/information_society/activities/intelligentcar/index_en.htm

[17] J. Ferber, *Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999.

[18] Frog AGV Systems, "Frog AGV Systems Website," 2010. [Online]. Available: http://www.frog.nl/

[19] D. T. Gantz and J. R. Mekemson, "Flow profile comparison of a microscopic car-following model and a macroscopic platoon dispersion model for traffic simulation," in *WSC' 90: Proceedings of the 22nd conference on Winter simulation*. Piscataway, NJ, USA: IEEE Press, 1990, pp. 770–774.

[20] A. P. Gerdelan, "Driving intelligence: A new architecture and novel hybrid algorithm for next-generation urban traffic simulation," Massey University, Tech. Rep. CSTN-079, February 2009.

[21] A. Gerdelan, "A solution for streamlining intelligent agent-based traffic into 3d simulations and games," Massey University, Tech. Rep. CSTN-072, January 2009.

[22] R. Haberman, *Mathematical Models: Mechanical Vibrations, Population Dynamics and Traffic Flow*. Prentice-Hall, Inc., 1977.

[23] P. Henderson. (2007, November) SUV with mind of its own wins robot car race. Reuters. [Online]. Available: http://www.reuters.com/article/idUSN0419493620071104

[24] P. Horrell. (2004, September) Intelligence: Behold the All-Seeing, Self-Parking, Safety-Enforcing, Networked Automobile. Popular Science. [Online]. Available: http://www.popsci.com/cars/article/2004-09/intelligence-behold-all-seeing-self-parking-safety-enforcing-networked-automobi

[25] S. Jain and C. R. McLean, "A concept prototype for integrated gaming and simulation for incident management," in *WSC '06: Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 493–500.

[26] T. Johnson. (2005, November) 'Driverless' Vectra is go. Auto Express. [Online]. Available: http://www.autoexpress.co.uk/news/autoexpressnews/62349/vauxhall_vectra.html

[27] S. Koenig, "Sven Koenig Official Website," 2010. [Online]. Available: http://idm-lab.org/

[28] S. Koenig and M. Likhachev, "D* Lite," in *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 476–483.

[29] R. D. Kühne and M. B. Rödiger, "Macroscopic simulation model for freeway traffic with jams and stop-start waves," in *WSC '91: Proceedings of the 23rd conference on Winter simulation*. Washington, DC, USA: IEEE Computer Society, 1991, pp. 762–770.

[30] M. Likhachev, "Maxim Likhachev Official Website," 2010. [Online]. Available: http://www.seas.upenn.edu/~maximl/

[31] F. L. Mannering, S. S. Washburn, and W. P. Kilareski, *Principles of Highway Engineering and Traffic Analysis*, 4th ed. John Wiley & Sons, Inc., 2009.

[32] J. Markoff. (2007, November) No Drivers, but a Lot of Drive. The New York Times. [Online]. Available: http://www.nytimes.com/2007/11/11/technology/11stream.html?_r=1\&ref=business

[33] McTrans Moving Technology, "TSIS: Traffic Software Integrated System Website," 2009. [Online]. Available: http://mctrans.ce.ufl.edu/featured/tsis/

[34] J. J. Olstam, J. Lundgren, M. Adlers, and P. Matstoms, "A framework for simulation of surrounding vehicles in driving simulators," *ACM Trans. Model. Comput. Simul.*, vol. 18, no. 3, pp. 1–24, 2008.

[35] P. Paruchuri, A. R. Pullalarevu, and K. Karlapalem, "Multi agent simulation of unorganized traffic," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems.* New York, NY, USA: ACM, 2002, pp. 176–183.

[36] Quadstone Paramics, "Quadstone Paramics Website," 2009. [Online]. Available: http://www.paramics-online.com/

[37] M. Radecký and P. Gajdoš, "Intelligent agents for traffic simulation," in *SpringSim '08: Proceedings of the 2008 Spring simulation multiconference.* San Diego, CA, USA: The Society for Computer Simulation, International, 2008, pp. 109–115.

[38] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cli s, NJ., 1995.

[39] L. A. Schaefer, G. T. Mackulak, J. Cochran, and J. L. Cherilla, "Application of a general particle system model to movement of pedestrians and vehicles," in *WSC '98: Proceedings of the 30th conference on Winter simulation.* Los Alamitos, CA, USA: IEEE Computer Society Press, 1998, pp. 1155–1160.

[40] C. Squatriglia. (2008, January) GM Says Driverless Cars Could Be on the Road by 2018. WIRED. [Online]. Available: http://www.wired.com/autopia/2008/01/gm-says-driverl/

[41] Stanford Racing, "Stanford Racing Stanley Website," 2005. [Online]. Available: http://cs.stanford.edu/group/roadrunner//old/index.html

[42] ——, "Stanford Racing Junior Website," 2006. [Online]. Available: http://cs.stanford.edu/group/roadrunner/

[43] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, pp. 89–100, 1993.

[44] ——, "Anthony Stentz Official Website," 2010. [Online]. Available: http://www.frc.ri.cmu.edu/~axs/

[45] Tartan Racing, "Carnegie Mellon Tartan Racing Website," 2007. [Online]. Available: http://www.tartanracing.org/

[46] M. A. P. Taylor, W. Young, and P. W. Bonsall, *Understanding Traffic Systems.* Avebury Technical, 1996.

[47] B. Templeton, "Where Robot Cars (Robocars) Will Really Take Us," 2010. [Online]. Available: http://www.templetons.com/brad/robocars/

[48] Trafficware, "Trafficware Website," 2009. [Online]. Available: http://www.trafficware.com/

[49] M. Treiber, "Microsimulation of Road Traffic Applet," 2009. [Online]. Available: http://www.traffic-simulation.de/

[50] TSS - Traffic Simulation Systems, "Aimsun Website," 2009. [Online]. Available: http://www.aimsun.com/site/

[51] VisLab, "BRAiVE Website," 2010. [Online]. Available: http://www.braive.vislab.it/

[52] ——, "The VisLab Intercontinental Autonomous Challenge," 2010. [Online]. Available: http://viac.vislab.it/

[53] ——, "VisLab Website," 2010. [Online]. Available: http://vislab.it/

[54] J. Wittmann, J. Göbel, D. Möller, and B. Schroer, "Refinement of the virtual intermodal transportation system (vits) and adoption for metropolitan area traffic simulation," in *SCSC: Proceedings of the 2007 summer computer simulation conference*. San Diego, CA, USA: Society for Computer Simulation International, 2007, pp. 1–5.