NTNU – Trondheim
Norwegian University of
Science and Technology

How Case-based Reasoning can be used to predict and improve Traffic Flow in Urban Intersections.

Ole Johan Andersen

Master of Science in Informatics
Submission date: December 2012
Supervisor: Agnar Aamodt, IDI
Co-supervisor: Anders Kofod-Petersen, IDI
Jo Skjermo, Statens Vegvesen

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

The traffic situation in urban areas has become a large problem over the recent years. Especially congestion during rush hours is becoming quite visible. Often, this can not be solved by simply building more roads, as most urban areas do not have the available physical space required. One way of solving these problems can be to focus on decreasing the number of vehicles. This can be done by increasing the amount of people traveling in each vehicle, or focusing on alternative transportation such as bicycle or public transport. Another way of utilising the existing infrastructure, is to improve the control of the traffic flow.

In this thesis we present a prototype case-based reasoning (CBR) system for the purpose of controlling the traffic lights in an urban intersection. The system uses historical vehicle counts, obtained from an intersection in the city of Trondheim, before using this knowledge in order to make new signal plans for the intersection. jCOLIBRI is used as framework for the development of our CBR-system and evolutionary algorithms are used for weighting the case base. The traffic simulator Aimsun is used for the evaluation of our solution.

Simulation results indicate that the CBR-system is able to satisfactory calculate signal plans based on the predicted traffic, in a variety of different scenarios.

Sammendrag

Trafikksituasjonen i urbane områder har blitt et økende problem de siste årene. Spesielt køer knyttet til rush-trafikken viser at dagens infrastruktur ofte ikke strekker til. Ofte kan dette løses ved å bygge og utvide veier, men de aller fleste urbane strøk har ikke denne plassen tilgjengelig. En måte man kan løse dette på, er å prøve å minke antallet kjøretøy som benytter seg av veinettet ved å fokusere på økt bruk av f.eks sykkel og kollektivtrafikk. En annen måte å løse dette problemet på, er å forbedre hvordan trafikken blir kontrollert.

I denne oppgaven presenterer vi et prototype case-based reasoning (CBR) system som kontrollerer trafikken i urbane strøk ved å kalkulere signalplaner. Systemet bruker historiske trafikktellinger hentet fra et lyskryss i Trondheim, og bruker denne kunnskapen til å lage signalplaner for dette krysset. jCOLIBRI har blitt brukt som vårt rammeverk for å lage CBR-systemet, og vekting av case basen har blitt gjort ved hjelp av evolusjonære algoritmer. Trafikksimulatoren Aimsun har blitt brukt for å evaluere løsningen.

Simuleringsresultatene indikerer at CBR-systemet er i stand til å kalkulere tilfredsstillende signalplaner basert på prediksjoner av trafikken, i et variert utvalg av forskjellige scenarioer.

Preface

This Master thesis constitutes the final work of my two year Master of science studies in Informatics. The work has been done at the Department of Computer and Information Science (IDI), at the Norwegian University of Science and Technology (NTNU), in cooperation with the Norwegian Public Road Administration (NPRA).

Acknowledgements

This work would not be possible if it were not for the help from my supervisors: Anders Kofod Petersen (IDI), Agnar Aamodt (IDI), and Jo Skjermo (NPRA).

During the research we have been working closely with the NPRA where a lot of employees have been very helpful in providing expert knowledge to our research. Especially Ørjan Tveit require a special thanks. He have provided help on crucial matters, by allocating a lot of his time in order to help us.

Tor Wiig from Swarco was able to meet with us on multiple occasions, in order to teach us more about SPOT/UTOPIA, and providing us with an issue of this system.

Trond Foss from SINTEF was able to help us get started on this project by meeting with us early after the project was assigned.

My good friends Simen Echholt, Nicolai Meltveit, and Christian Jonassen have been assisting in proof reading of this report, which I am very grateful for.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this research case-based reasoning (CBR) has been studied in the domain of traffic control, focusing on the control of traffic lights in urban intersections. The domain is quite complex, and offers a vast amount of data that is continually registered using detectors in the road, radars, and other technology used for counting and surveillance.

A literature review has been performed in order to uncover the current state of the art research done in the field of AI-methods and traffic control. This shows how the different methods have been applied in the domain, as well as their evaluated performance, and proposed future work.

Throughout the research, multiple approaches have been tried out in order to solve the problem, before choosing one to further pursue. This decision was made based on technical limitations that emerged.

A prototype system has been developed in order to utilise the potential of CBR in this domain. Weighting of the case base has been done using evolutionary algorithms.

The system uses historical data in order to be able to predict traffic demand at a given time, this is then used in combination with domain knowledge that the system uses in order to calculate signal plans. The resulting signal plans are then evaluated against static signal plans, which are commonly used in intersection control. Evaluation is done by running simulations on a proprietary traffic simulation software.

1.1 Goals and research questions

- **Goal 1**: Implement a CBR system that is able to control traffic intersections and perform better than one or more selected methods in terms of:
  - Traffic flow
  - Total travel time
  - Total waiting time
  - Total delay
- **Research question 1**: What is the current state of the art regarding:

  - Methods used for traffic control?
  - Research of AI-methods within this domain?

- **Research question 2**: Is CBR suitable for the purpose of urban intersection control?
  - How will the implemented system perform in different scenarios compared to one or more other methods?
  - Which method performs overall better?

1.2 Background and motivation

The traffic situation in urban areas has become a large problem over the recent years. Especially congestion during rush hours is becoming quite visible. Often, this can not be solved by simply building more roads, as most urban areas do not have the available physical space required. One way of solving these problems can be to focus on decreasing the number of vehicles. This can be done by increasing the amount of people traveling in each vehicle, or focusing on alternative transportation such as bicycle or public transport. Another way of utilising the existing infrastructure, is to improve the control of the traffic flow.

The domain of traffic control is highly complex and multiple methods exist for the purpose of controlling traffic. Because of this complexity, successful methods in one traffic environment may not give satisfying results in another environment. Road networks varies immensely in their design, which greatly affects the solutions that should be used. Other variables includes traffic culture, weather, day, time of year, season, and so on. Although one solution may be good at a given time, the traffic demand often changes as time passes by, and maintenance is often required to satisfy new demands. A highly dynamic method, which is able to comprehend complex domains, is desired in order to sufficiently handle this problem.

In this research we want to show that CBR provides the necessary functionality in order to offer a solution to this problem. A CBR-system can relatively easily be programmed to dynamically solve problems, as well as be able to grasp intricate concepts. CBR-systems can easily be modified by altering the cases in the system. These cases are independent of each other, which makes the system highly adaptable as the same system can be used in multiple locations using cases that cover the specific need. If the CBR-system is configured to be able to retain and revise new cases, it may by itself change and improve by learning. The system can therefore perform better over time, even as traffic demand changes throughout the years.

Because of the complexity of the domain, it is difficult to make a good model of the domain. It is also hard for experts to define rules which cover it properly. Similar methods that rely primarily on domain experts, such as rule-based reasoning (RBR), may not therefore be as applicable as CBR. Since the traffic domain have a lot of data available, the system is able to get a sufficient amount of specific traffic experiences, which will cover the vast majority of traffic situations. Using this, along with general domain knowledge, a CBR system is likely to perform well in this domain.

Simulations can be done by using proprietary simulation software in combination with a CBR system, which after simulations yield enough data to properly evaluate the system.

This thesis is written in cooperation with the Norwegian Public Road Administration (NPRA)[1]. They are responsible for planning, construction and operation of the national and county road networks, as well as vehicle inspection, driver training, and licensing in Norway. NPRA has a technology group which focuses on using technological solutions for the purpose of contributing to a safer, better and more environmentally friendly road traffic, known as "intelligent transportation systems" (ITS).

The Norwegian Ministry of Transportation and Communication presents a plan of measures in order "To provide an effective, accessible, safe and environmentally friendly transport system that covers the society's transport requirements and encourages regional development." [22] By increasing the efficiency of the traffic, it will contribute to several of the objectives presented in this plan. Two of four main objectives concerns "increase of traffic flow" and "decreasing emissions and environmental impact". These are both factors that can be positively affected by better urban intersection control.

A great motivation is therefore the possibility to help the NPRA with their work in the ITS field in order to reach the goals presented by the Norwegian Ministry of Transportation and Communication.

1.3 Report overview

In the next chapter the methodological approach is presented in order to show how this research has been conducted, describing how the literature study has been done, the iterative process of specifying the thesis, method of implementation, evaluation methods used, and more.

The following chapter presents some background information related to this research in order to give the reader a greater understanding on how traffic signal control is done today, as well as an overview of relevant AI-methods.

Chapter 4 presents the related research results. This state of the art research is presented and discussed in order to get a good foundation for the implementation of a solution.

Chapter 5 shows the iterative process of formulating the goal and research questions for this research.

Chapter 6 presents the proposed solution and describes the implementation by giving a system overview, and a more detailed description of the different modules used.

Evaluation of the system is presented and discussed in Chapter 7.

Lastly the conclusion and future work is located in Chapter 8.

[1]Norwegian Public Road Administration, http://www.vegvesen.no

Chapter 2

Methodological approach

This chapter presents the different methods used in our research for solving the tasks at hand.

First, we present our methods for the iterative decision process we had in order to decide our goals. How we performed our literature study is presented in Section 2.2. Data acquisition has been done in order to collect both traffic and weather data, as described in Section 2.3. This is followed by some methods used regarding the implementation, also introducing the tools we have used. This is followed by the methods used for evaluation in Section 2.5. Lastly, the research plan is presented.

2.1 Iterative decision process

Initially, the goals of this thesis were not defined, and thus required some effort in doing so. Increasing the flow within the traffic domain was the only thing which was set, as this research was to be done in cooperation with the "Intelligent Traffic Systems Group" at the NPRA.

We decided early that an iterative process might be suitable in order to quickly change course if our proposed approach did not seem viable after some research. Doing this we would hopefully avoid focusing for too long on the wrong tasks, and at the same time incrementally get closer to a problem definition which we felt comfortable with. This incremental approach was inspired by well-known agile methods used in project management and software development[30].

The following cycle would then be followed incrementally.

1. **Research step**: Perform literature research and discuss with domain experts and supervisors in order to get an increased knowledge of the field and inspiration for the problem statement. This step was meant to be superficial in order to as quickly as possible be able to draft a problem description. Once this decision process was finished, the real literature review is done more thoroughly as described in Section 2.2

2. **Draft step**: Draft problem statement

3. **Approval step**: Meet with supervisors to further formulate the problem, and if the supervisors agree with the choice of problem, go to the next step, or start over.
4. **Decision step**: Decide whether the problem was feasible by uncovering the technical, methodological, and temporal limitations that might occur. If it turns out to be infeasible, go to step 1 in the cycle.

During this cycle it was likely that there would be some downtime when waiting for mail correspondence or meetings, so this time has been used working with the documentation of the thesis work.

Chapter 5 presents this decision process.

2.2 Literature study

In order to get an up to date overview of the relevant research that has been done on the field, a literature study should be conducted. We have mainly used Google Scholar[1] to find material for our literature study. Since a lot of the relevant online libraries are indexed there, it was not necessary to search these libraries individually.

Relevant keywords used are located in the Table 2.1, where the different keywords in each column have been combined. Some searches have also involved the name of the authors of relevant articles, which often resulted in other relevant research that have been conducted.

AI-Method	Traffic keyword
CBR	traffic
case-based	traffic control
case-based reasoning	traffic routing
RBR	traffic management
fuzzy logic	signal control
expert system	intersection control
evolutionary algorithms	urban control
genetic algorithms	transportation
EA	transportation system
GA	

Table 2.1: Table showing the combination of keywords used

To consider an article relevant, it needs to fulfill different criteria. First and foremost, the research should have been conducted after 1990. The title was then considered, along with the keywords used for indexing, as it gives a good notion of the relevance. Finally the abstract and conclusion was read. If the study still seems relevant, most, or all of the following criteria, should be satisfied in order to include the research in our literature study:

- The study should augment our understanding of the method and/or domain.

[1]Google Scholar, www.scholar.google.com

- The study should mainly concern traffic management or traffic impact and safety.
- The study should apply AI-methods such as case-based reasoning, rule-base reasoning, fuzzy logic, or evolutionary algorithms.
- The study should include an implemented prototype system.
- The study should present results of the prototype system.
- The study should include information which are helpful to our research regarding design questions and other aspects considering implementation.

2.3 Data acquisition

In order to run simulations with realistic data, we decided to get real traffic data from measurements done in Trondheim. We therefore needed to both choose an intersection for our research, acquire traffic data, and then model it in the traffic simulator.

2.3.1 Intersection chosen for acquiring data

The intersection we used for our data extraction is depicted in Figure 2.1. One of the main reasons for choosing this intersection was because this was the only one we were able to acquire all necessary data from by ourselves. In spite of this, there are multiple aspects of this intersection that make it an interesting choice for our research.

Lanes going north (towards the right in the picture) are headed towards downtown, while the lanes going south (towards the left in the picture) are headed out of town. Both these lanes are heavily trafficked. However, the two lanes coming from east and west (up and down in the figure) are less trafficked, as there are mainly residential buildings associated with these lanes.

The lanes going towards, and out of the city both have three lanes towards the intersection, one for each possible turning. By following the lanes continuing on a straight line from the intersection, there are two lanes in both directions. The leftmost lane is used for normal traffic, while the rightmost lanes are used for public transport, taxis and priority vehicles. Use of the rightmost lane is allowed for normal vehicles towards the intersection as these are used for turning to the right. Cars that wants to turn left need to wait for their own green light, which adds an extra phase to the cycle.

Perpendicular to these lanes are two less trafficked directions. Both of these roads have two lanes going towards the intersection. The leftmost one is for going straight forward and turning left, while the rightmost one is for turning right. One lane in both directions are going from the intersection. There are no lanes for prioritised vehicles as the traffic in these lanes are comparably small when considering the lanes going north and south.

This intersection is also situated close to the home stadium of the city's football team. Therefore, it is often quite noticeable changes in the traffic flow before and after football matches.

Another interesting thing to note about this intersection is that there used to be a grocery store, down to the right in the picure, at the spot where the red little box is located. As this have been temporarily closed due to construction work, it is reasonable to assume that the traffic flows may have changed compared to right before it closed.

The overall speed limit is 50 km/h.

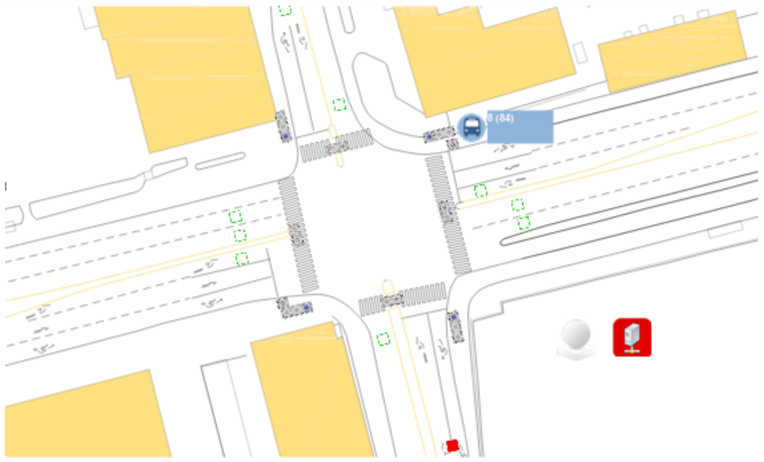


Figure 2.1: The intersection where the data was acquired from

2.3.2 Method for traffic data acquisition

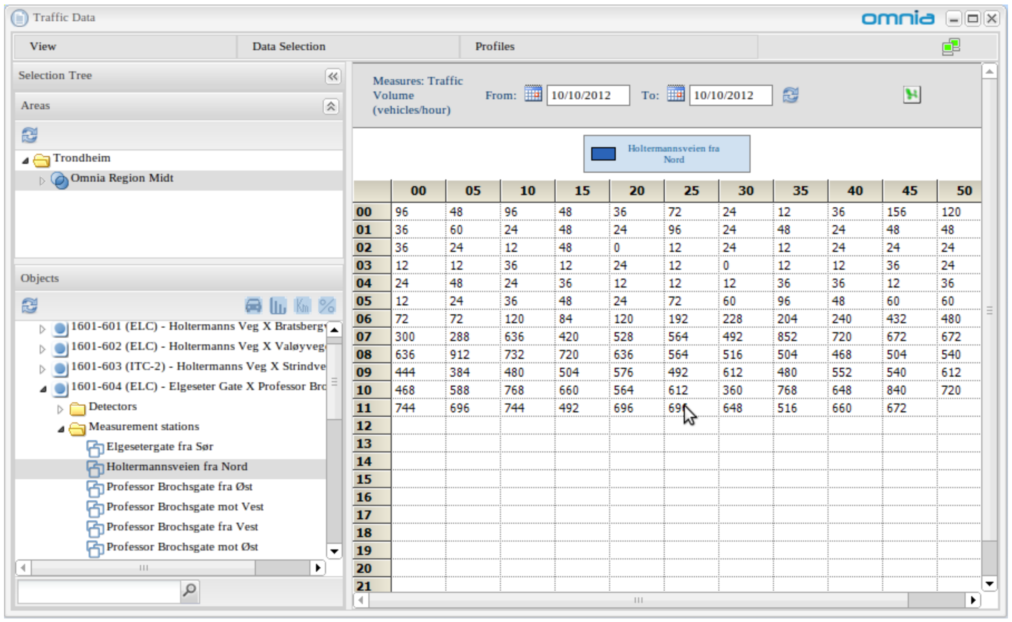
Data acquisition was done using Omnia, described in Section 2.4.2, which provide sufficient functionality in order to select the desired data. Different time intervals for extraction are chosen in order to reflect most of the scenarios that occur during a longer period. Omnia, and some of the traffic data is depicted in Figure 2.2.

In order to cover most of the regular day-to-day traffic situations, we decided to try to obtain the following data:

- Night traffic in the week and weekend.
- Morning traffic in the week (rush hour) and weekend.
- Noon traffic in the week and weekend.
- Afternoon traffic in the week (rush hour) and weekend.
- Evening traffic in the week and weekend.

Special events are also interesting to incorporate into the knowledge base, as these occasionally are responsible for causing traffic jams, or at least significant changes in the traffic picture. By having access to all traffic data from the last couple of years, it was therefore possible to extract data for abnormal traffic events.

The following special events have been added to our knowledge base:


Traffic Data
omnia
View
Data Selection
Profiles
Selection Tree
Areas
Trondheim
Omnia Region Midt
Objects
1601-601 (ELC) - Holtermanns Veg X Bratsberg
1601-602 (ELC) - Holtermanns Veg X Valøyveg
1601-603 (ITC-2) - Holtermanns Veg X Strindve
1601-604 (ELC) - Elgeseter Gate X Professor Brc
Detectors
Measurement stations
Elgesetergate fra Sør
Holtermannsveien fra Nord
Professor Brochsgate fra Øst
Professor Brochsgate mot Vest
Professor Brochsgate fra Vest
Professor Brochsgate mot Øst

Measures: Traffic Volume (vehicles/hour) From: 10/10/2012 To: 10/10/2012

🟦 Holtermannsveien fra Nord

	00	05	10	15	20	25	30	35	40	45	50
00	96	48	96	48	36	72	24	12	36	156	120
01	36	60	24	48	24	96	24	48	24	48	48
02	36	24	12	48	0	12	24	12	24	24	24
03	12	12	36	12	24	12	0	12	12	36	24
04	24	48	24	36	12	12	12	36	36	12	36
05	12	24	36	48	24	72	60	96	48	60	60
06	72	72	120	84	120	192	228	204	240	432	480
07	300	288	636	420	528	564	492	852	720	672	672
08	636	912	732	720	636	564	516	504	468	504	540
09	444	384	480	504	576	492	612	480	552	540	612
10	468	588	768	660	564	612	360	768	648	840	720
11	744	696	744	492	696	69	648	516	660	672	
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											

Figure 2.2: Omnia user interface

- Holiday.
- Football matches.
- Slippery roads.
- Days when a considerable amount of people are leaving for holiday.

Obtaining these events can be done by scanning the web-pages of the local football club and see when they have played their matches, and pick the date and time for the ones most popular in order to see the changes in the traffic. Also, by searching the local newspaper, we were able to find time and date for events such as "Slippery roads cause chaos".

2.3.3 Method for weather data acquisition

Weather data was needed as features for our cases. This data has been gathered using eKlima, described in Section 2.4.2. Reports that cover the given time and date which are used for the traffic data was acquired. The relevant weather data needed was decided at the time of implementation.

2.3.4 Method for signal plan acquisition

We were able to obtain signal plans, timetables for the plans, and a map for the phases. These were provided by the NPRA, and are being used in the intersection depicted in Figure 2.1. This information is available in Section 6.7.

2.4 Implementation

The implementation consists of a CBR-system written in Java, and a system written in Python, that organises simulations and communicate with both the CBR-system and Aimsun.

As Aimsun requires the use of a license, it has been necessary to borrow this from the NPRA. It was therefore desired from the NPRA that the implementation work was to be done at their locations, upon which we agreed.

We focused on developing using an iterative approach[30], in order to get a basic functional architecture as soon as possible. This gave us time to incrementally improve our system until the desired functionality was reached. Following this approach did also to some degree ensure that we had a working solution by our deadline. We feel that focusing on a more sequential design process could have caused more insecurity regarding the implemented system.

2.4.1 Choosing a framework for our system

A CBR-system can be implemented from scratch, or by using an already existing implementation. In order to save time, we quickly decided to use a framework. The choice was between myCBR and jCOLIBRI, which both were good candidates for our system.

jCOLIBRI

jCOLIBRI[2] is an open source framework made for developing CBR applications in Java. jCOLIBRI is made by the GAIA-group, and is released under the terms of LGPL. The framework provides a well defined architecture which simplifies the process of making an orderly CBR-system[14]. Documentation and a sufficient amount of reference implementations are provided. Features such as retrieval methods, reuse and revision methods, maintenance components, and evaluation methods, are all included into the framework. The possibility of making own methods for retrieval, reuse etc are simply done by implementing an interface. Evaluation methods, such as leave-one-out cross-validation and N-fold cross validation, are implemented, and can be effectively used by doing some configuration.

myCBR

myCBR[3] is an open source framework developed by German Research Center for Artificial Intelligence. The most popular version of myCBR is a plug-in used in an open source ontology editor known as Protégé. The seamless combination of Protégé and myCBR is one of the strengths of myCBR. In order to be able to implement additional functionality using Java, myCBR 3.0 BETA can be used. Documentation is done in Javadoc, and some examples are also provided. This myCBR version has beta status, and some functionality has yet to be implemented.

[2]jCOLIBRI, http://gaia.fdi.ucm.es/research/colibri/jcolibri

[3]http://mycbr-project.net/

Selecting a framework

Some comparisons of these two frameworks have been documented, such as one presented by Atanassov and Antonov[5]. Their conclusion states that jCOLIBRI is a good basis for complex applications, while the myCBR platform can be used for non-complex CBR applications.

The documentation of jCOLIBRI is much more extensive than the one of myCBR. Tutorials describing a lot of the relevant functionality are provided along with an implementation manual. By using this documentation, the framework is quite straight forward to comprehend and use. This ensures a good basis for future development.

jCOLIBRI contains functionality that supports the whole CBR-cycle[1] consisting of support for retrieval, reuse, revise and retain. This functionality is easy to extend by implementing interfaces which describes the desired functionality, such as ones for similarity measurement. Because of the well written architecture of jCOLIBRI, the extendability is good.

myCBR on the other hand presents a good and detailed user interface for building a case base without the need for programming. There are more options for the use of weights and modification of the similarity functions, than available through the one of jCOLIBRI. This is quite an advantage in order to fast prototype a CBR-system without the need to develop anything extra, as you might wish to do when using jCOLIBRI.

Although both frameworks are good choices we decided to go with jCOLIBRI. This is because we wanted to develop our own CBR-system, ideally by building upon a flexible platform that is easy to understand, as well as an architecture which is expandable without compromising functionality. We also think that the comprehensive documentation of jCOLIBRI gave us a head start, compared to using myCBR.

2.4.2 Tools

In addition to jCOLIBRI, which is described in the previous section, we decided to use the tools presented in the following subsections.

Aimsun

Aimsun[4] is a traffic simulation software that support models in all scales. The software is known for its ability to perform demanding simulations in high speed. Simulations can be done on a microscopic, mesoscopic and macroscopic scale, or by combining these to perform hybrid simulations.

There are also additional modules that can be applied to enhance the functionality such as adaptive control interfaces, for interfacing with traffic control systems. A programming interface is also available as an additional functionality, which enables other applications to communicate with Aimsun.

Aimsun was used in our research because of the programming interface and because we had access to a license.

[4]TSS-Transport Simulation Systems, www.aimsun.com

SPOT/UTOPIA

UTOPIA[5] is a traffic control system which offers multiple ways of controlling traffic to fulfill the different needs of road networks. The adaptive mode focuses on the current and future intersection states that is used to make close to real-time decisions, which are believed to be best suited for these situations. By assigning weights, the system can prioritise specific vehicles such as emergency vehicles and public transport, without penalising the rest of the traffic. Remarkable performance is especially shown in areas with much congestion and unpredictable traffic conditions[33].

The system is used in multiple large cities such as Trondheim, Oslo and Copenhagen.

Omnia

Is a solution for the integrated road transport environment. The Omnia[6] platform can easily connect to a number of modules which increase the functionality. This tool has been used for extracting real traffic data, which are available in real-time, and also for an extended period, which makes it possible to recover historical data. The user-interface is shown in Figure 2.2, and shows the available traffic intersections, as well as the traffic data of the one selected.

As Trondheim uses Swarco systems for traffic management, it is natural to use this tool for data extraction.

eKlima

eKlima[7] is a portal which gives access to the weather data registered by the Norwegian Meteorological Institute. Reports present measured data from selected time periods, done by a given observation station. Such data includes, temperatures, precipitation and other weather information.

2.5 System evaluation

In order to evaluate our system, we have chosen to use cross validation for evaluating our case base, and simulations in Aimsun for evaluating our CBR-system. Both methods are described in the following sections.

2.5.1 Cross validation

One common way of evaluating a CBR-systems case base, is by first training the CBR-system using training data, and then evaluate the trained system by applying a test set. If the system performs badly on the test set, it is likely to assume that the training cases does not cover

[5]Swarco, http://www.swarco.com/en/Products-Services/Traffic-Management/Public-Transport/Public-Transport-Priority/UTOPIA

[6]Swarco, http://www.swarco.com/en/Products-Services/Traffic-Management/Public-Transport/Fleet-Management/OMNIA

[7]eklima.met.no

the desired solution space, and are therefore not sufficient. If the case base is not evaluated, it is therefore difficult to say whether it will perform satisfactory on the actual problems.

In order to evaluate the case-base we decided to use cross validation. This method is often used when the number of cases available is limited, and it will cause significant performance loss if one part of the case base would be used for testing instead of training.

K-fold cross validation divides the case base into k equally sized "folds", which in turn are used as a test set. The rest of the case base is then used as training data. This has to be done k times in order to train and test the system in all different ways possible. When the case base is small, it is common to use leave-one-out cross validation which uses one case for testing, and the rest for training. Using this method saved us some time as we did not have to make a separate test-set.

jCOLIBRI has support for doing both leave-one-out and k-fold cross validation, which only need some modifications in order to work with our system, and in turn give the desired evaluation information.

Deciding whether the solution given by the system during this evaluation process is good enough, and the amount of solutions which has to be approved in order to reach this conclusion, has been decided during the implementation and evaluation process.

If it turns out that the system performs poorly on tests, the following actions can be taken:

1. Add more cases to the case base.

2. Improve feature weights.

3. Use other methods for similarity measurement.

2.5.2 Simulations in Aimsun

For the purpose of evaluating the CBR-system against other traffic control systems, it was necessary to perform simulations. Therefore, a model of an intersection was necessary before the simulations could be performed. In order to achieve this, we constructed an intersection which is similar to the one we obtained our data and signal plans for, as described in Section 6.6.

Performing the simulations

Scenarios has been designed in cooperation with the NPRA, and the CBR-system has been evaluated by running simulations. The traffic flow from the scenarios has been obtained by acquiring real traffic data from an intersection, as described in Section 2.3.2. This traffic data has then been applied to the simulation scenario by setting the traffic flow in a matrix that represent the origin and destination for the given cars, more elaborated in Section 6.6.2.

Multiple different scenarios has been run together and then evaluated against each other. One uses a static signal plan given the time and day of the scenario, and the others use the signal plan which is proposed by the CBR-system, using different configurations. The simulated time intervall was one hour. As the "random" appearances of the cars during

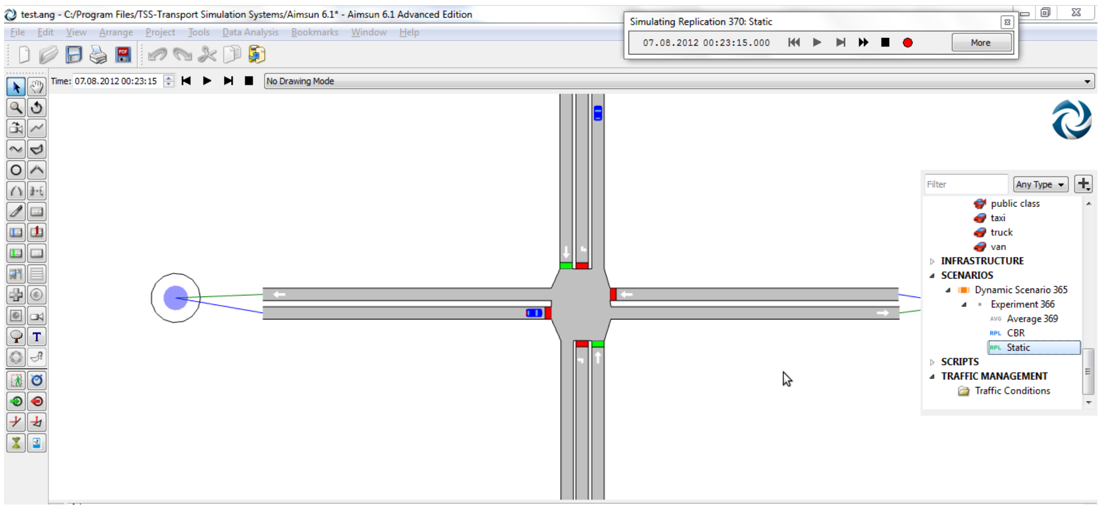

test.ang - C:/Program Files/TSS-Transport Simulation Systems/Aimsun 6.1* - Aimsun 6.1 Advanced Edition
File Edit View Arrange Project Tools Data Analysis Bookmarks Window Help
Simulating Replication 370: Static
07.08.2012 00:23:15.000
More
Time: 07.08.2012 00:23:15
No Drawing Mode
Filter
Any Type
public class
taxi
truck
van
INFRASTRUCTURE
SCENARIOS
Dynamic Scenario 365
Experiment 366
Average 369
CBR
Static
SCRIPTS
TRAFFIC MANAGEMENT
Traffic Conditions

Figure 2.3: Aimsun running a simulation

the simulation is determined by a seed, we have ensured that the two different scenarios get exactly the same traffic pattern. It is also possible to change this seed in order to perform multiple simulations, where the car appearance is based on a new random model. The amount of cars during the simulation does not change, just their time of appearance.

Evaluating the simulation results

After each simulation Aimsun gives a result set containing a large amount of statistical data. We have chosen to focus on the following, as described in the Aimsun manual[35], for our evaluation:

- Stop time - average time at standstill per vehicle per kilometer.
- Travel time - average time a vehicle needs to travel one kilometer inside the network. This is the mean of all the single travel times (exit time - entrance time) for every vehicle that has crossed the network, converted into time per kilometer.
- Speed - average speed for all vehicles that have left the system. This is calculated using the mean journey speed for each vehicle.
- Delay - average delay time per vehicle per kilometer. This is the difference between the expected travel time (the time it would take to traverse the system under ideal conditions) and the actual travel time. It is calculated as the average of all vehicles, and then converted into time per kilometer.

The results from the simulation using static plans, and the results from the corresponding simulations using plans given by the CBR-system, have been evaluated against each other. For each scenario, 5 simulations using different seeds have been run, and the results have been entered into a simulation results table, as shown in Table 2.2. After running the 5 simulations using all control types, the results were averaged and then compared against each other as shown in the evaluation table, this can be seen in Table 2.3.

#	**Control type**	**Stop time**	**Travel time**	**Speed**	**Delay**
1					
2					
...					

Table 2.2: Simulation table

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
CBR average				
Static average				

Table 2.3: Evaluation table

2.6 Research plan

This section presents the research plan, shown in Table 2.4, that shows when a given activity is desired to take place. Each of these activities has been more thoroughly described previously throughout this chapter.

Time estimate	**Activity**
8 weeks	Decision process
6 weeks	Literature research
3 weeks	Data acquisition
10 weeks	Implementation and evaluation
9 weeks	Report writing

Table 2.4: Resarch plan

The decision process was expected to be time consuming. A fundamental understanding of the domain, as well as discussions with supervisors and experts, are required in order to be able to decide on a good problem description. It was also expected that there would be multiple problem drafts before we was completely satisfied, and ready to initiate the rest of the research process.

Because of the decision process, the litterature research is a little shorter than it would normally be. During the decision process it is necessary to conduct a superficial literature research in order to decide on the feasibility of an approach. It is therefore assumed that the fundamental part of this research have already been done, and that we need to more thoroughly read and assess the available literature that we have found.

Data acquisition has gotten the smallest share of the scheduled time. It was expected that we would use some time to find good periods for obtaining both ordinary traffic data, and the ones describing more abnormal traffic situations. Acquiring weather data and signal plans, are included into this time period.

Implementation and evaluation of the solution has been done in two stages, before and after the summer. By developing iteratively we did also evaluate the system continually. However, the process of finally deciding upon the proper scenarios to use, and running the final simulations has been done at the end of the last implementation iteration.

Documenting the work, done by writing a report, has primarily been done at the end of this research. Some of the documentation has been worked on when doing the other activities, but this did primarily consist of note taking and drafts.

Chapter 3

Background

This chapter presents relevant background information for our research, giving an introduction on traffic signal control, as well as a selection of artificial intelligence methods used within this domain. For our research, we decided to use case-based reasoning for our main system, described in Section 3.2.1. For the purpose of weighting our case base, we applied evolutionary algorithms, as described in Section 3.4.

3.1 Traffic signal control

This section gives a background on methods and systems used for traffic control, presenting some of the ones that are most commonly applied. If more traffic related theory is desired, the monograph "Traffic Flow Theory"[12] explains traffic stream models, measurement of traffic, queuing models, etc.

Often used terms within the domain of signal control are:

- A **group** can be one or more traffic lanes or pedestrian groups that need green time. In Figure 3.1 there are a total of 4 different groups, where group 7 and 8 show a pedestrian group, and group 1 and 2 are both traffic lanes.

- **Phases** consist of signal groups that advantageously can be green at the same time. Figure 3.1 shows such a phase where the different traffic lanes are not conflicting with each other. It is common to have green time for a pedestrian group which is heading in the same direction as the traffic. It is generally desired to have as many groups in a phase as possible, as long as they are not conflicting.

- A **signal plan** consists of phases which are ordered in a given way, giving a cycle. When this order of phases have been executed in turn, the cycle will typically start over again. One method of calculating such cycles are shown in Section 3.1.3.

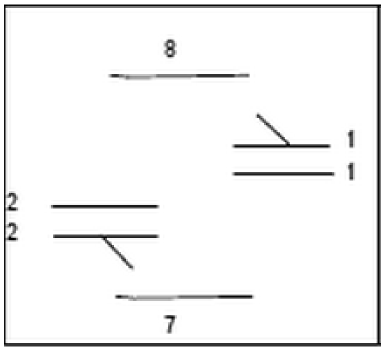


Figure 3.1: Phase 1

3.1.1 Timed traffic control

A common way of controlling traffic signals, is by using predetermined signal plans, which uses the same plan and cycle-length independent of the current traffic pattern. Timed traffic systems often coordinate neighboring intersections with each other in order to obtain a better and more continuous flow through multiple intersections, without halting for every traffic light. Methods for coordinating includes:

Simultaneous systems

When using *simultaneous systems*, all of the signaling systems change at the same time. This can be used when there are short distances between the intersections. As a result of using simultaneous systems, a trafficker may speed in order to reach more intersections in the same period. Usage of this method should therefore not surpass two intersections.

Alternating systems

Alternating systems are often used to achieve a “green wave”. A “green wave” is such a coordination between multiple intersections that causes waves of cars to pass through large distances without getting red lights. This is achieved by having every other signaling system to show a green light, while the systems in between show a red light. For this to function optimally, the distance between the intersections should be similar. An illustration can be seen in Figure 3.2.

Progressive systems

Progressive systems may have multiple plans that are changed throughout the day in order to adapt to major changes in the traffic. For example one plan can focus on the morning rush, when the pressure is focused towards downtown, and another plan in the afternoon rush, when the traffic pressure is focused in the opposite direction. A combination of the two previous methods may be applied into these plans.

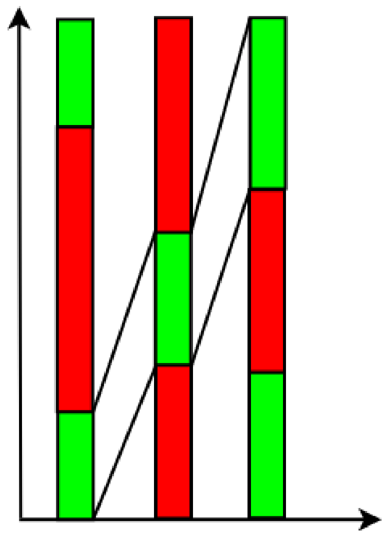


Figure 3.2: Alternating systems. The figure shows 3 adjacent intersections (x-axis) and the colours of their light in on the connecting lanes over a time period (y-axis). The cars traveling (lines between intersections) from the first intersection at the start of the green time, will reach the next one as it switch to green. Similarly, the cars traveling at the end of the green time, will reach the next intersection right before it turns red.

Disadvantages of timed systems

Timed systems can often give good results but some of the disadvantages include:

- Unexpected situations like traffic accidents and other big changes in the traffic pattern do not affect the behavior of the signal system, and an operator may have to manually change the system. If there is a known upcoming event, such as a football match, a plan can be applied to that timed period.
- Priorities to public transport like buses, are difficult to give without affecting the rest of the traffic in a relative disproportionately way.
- When the signal plans are generated, they might be optimal concerning the current traffic patterns when they are initially applied. However, the general traffic demand will typically increase in line with population growth and other factors. Often the necessary maintenance is not done to reflect these minor changes in the traffic, which can result in suboptimal traffic control.

3.1.2 Adaptive control

There are multiple systems engineered for improving the traffic flow by controlling the signaling systems. Typically, these systems focus on adaptive control, which in general means that the system changes the signal plan in order to adapt to the continuously changing traffic pattern.

Adaptive centralised systems

All known traffic data is collected and evaluated in a centralised location. Global optimisation is done, and the plans are sent to each signal system. The problem with this architecture is the vast amount of data which is needed to be processed, causing limitations in important aspects, such as time.

Adaptive distributed systems

With adaptive distributed systems, each intersection calculates a local optimisation, which is then applied. Each system communicates with its neighboring intersection which ables it to get forecasts for the expected traffic pattern in its own intersection. This can give a close to optimal traffic control from an overall perspective. As a result of distributing the optimisation, it is possible to prioritise selected groups, such as public transportation.

3.1.3 Webster's formula

Webster's formula[41] is a simplified method for calculating cycle time and the green time of each phase in an intersection. It is often used for getting a good initial estimation on how long the traffic light cycle time, and the green time of phases, should be.

Every lane has its load factor computed by the following equation:

$$y_i = \frac{q_i}{s_i} = \frac{\text{traffic volume}}{\text{capacity}}$$

The capacity for an intersection without turnings is assumed to be 1800-(total phases * 100)[37]

The traffic volume is the largest amount of traffic in the given lane.

All the load factors are then summed by the max value of y, given each phase.

$$Y = y_{\max(\text{phase 1})} + y_{\max(\text{phase 2})} + ...$$

The sum of all red-time in the cycle, L, is summed by adding the length of the red-time after each phase, where the simplification says that yellow-time is counted as green time. The handbook used for planning, operation and maintenance on traffic signal systems from the NPRA[37], assumes a base red time of 3 seconds.

The formula for optimal cycle time C$_o$ is by:

$$C_o = \frac{1.5L + 5}{1 - Y}$$

For calculating the green time of each phase based on the optimal cycle time, this formula is used:

$$g_{phase(N)} = \frac{y_{max(phase(N))}}{Y}(C - L)$$

3.2 Machine learning

Machine learning is a branch in Artificial Intelligence that focus on algorithms that improve the performance of a system based on empirical data. This is achieved by recognising patterns in the input data, which are used to make decisions. Because of the large amount of possible input data, an example solution will not exist for each of them, and the learner need to generalise in order to provide answers to input cases which has not been seen before.

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." - Tom Mitchell [20]

Many machine algorithms can be considered as "eager learners" because they will try to construct a general target function based on its example data. The purpose of this function is to give the correct output based on the input. After the target function has been made, input to eager systems will not be used in order to improve or modify this function. Lazy learning, on the other hand, will rather wait to generalise the known data until it gets a query. This is done for each query, and results in the system using information acquired during earlier queries, to be included in the new function. Case-based reasoning is an example of a lazy learner.

3.2.1 Case-based reasoning

Case-based reasoning (CBR) is a problem solving paradigm that uses earlier experiences to solve new problems, similar to how humans solve problems. A key assumption is that similar problems also have similar solutions. When a doctor diagnoses a patient, he will typically remember other specific patients with similar symptoms he has earlier diagnosed. If the doctor do not remember earlier patients with these exact symptoms, he will reason using general knowledge to find a diagnosis, or maybe partially use earlier solutions, and revise them to fit the new patient. Next time a patient has the same symptoms, the doctor should now recognise them, and apply this proper diagnosis based on that concrete experience. Schank[28] explains the dynamic memory model, which was the basis for the earliest CBR-systems.

A CBR-system will look at earlier known situations, which contains a problem description and a solution that is similar to a current problem description. If the system is able to retrieve similar situations, the solution of these situations can be reused if it turned out to give an acceptable outcome. A case is typically the combination of a problem, a solution and an outcome assessment that is stored in a case-base with other known cases. This can be considered as specific knowledge. When applying this to the patient diagnosis example, the symptoms can be seen as features that make up a problem description, and the solution is the diagnosis. The outcome of applying a solution is recorded in order to signify whether the applied diagnosis turned out to be a good one. Even though a problem has a known solution does not necessary mean it is the best, or even a good solution. Thus the outcome is necessary to assess, and store in the case.

Whether a known problem is similar to a new one is based upon the general knowledge of the system and the similarity measured between cases. If the system has no knowledge of the given domain, like a doctor without a medical education, but only specific cases stored in its base, it will only retrieve cases based on similarity measures. In contrast, a CBR-system may have additional knowledge, like a doctor with a medical education, of the domain and not only specific cases. Using the general knowledge, the system can reason to find a similar case, and also even modify solutions in order to solve the problem[1]. The modified solution can then be retained in the case-base in order to support future problem solving. It is therefore necessary to have additional knowledge if the system shall be able to reason and produce new solutions to expand its memory, which can be considered as learning.

The CBR-cycle

The CBR-cycle has been formalised into a process of four steps[2] and an illustration can be seen in Figure 3.3. A CBR-system may or may not incorporate all of these steps in its cycle.

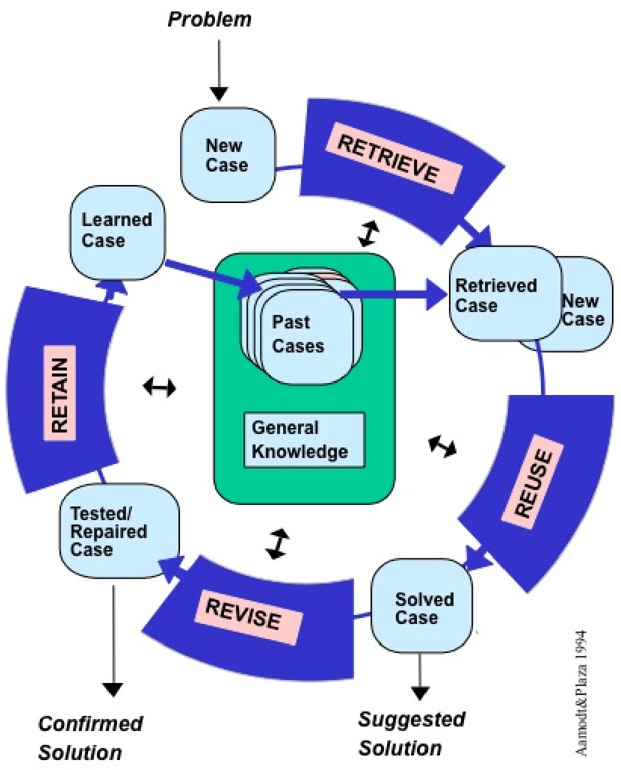


Figure 3.3: The CBR cycle adopted from [2]

Retrieval is the first step in the cycle. Which cases that will be retrieved is decided based on a given similarity threshold. The case that will be selected among the retrieved cases, and taken further in the cycle, depends on the selection strategy which is used.

Reuse is done by copying and/or adapting. Copying is the most trivial and is often done by focusing on the similarities and neglecting dissimilarities. Adapting is a more demanding process and involves finding what part of a case to reuse, and adapt either the solution or the method that gave the solution, before applying it to the new problem.

Revising is necessary to identify whether the solution from the reuse step actually turned out to give satisfying results. This is typically done by either applying the task in the given domain, or by inquiring an expert. If unsuccessful, the solution need to be repaired by modifying it so that failures does not occur. Some results may not appear immediately, like whether or not a diagnosis turned out to be right based on the treatment. In these cases the solution may be used and retained but marked as unevaluated until an evaluation can be done.

The last step involves *retaining* the knowledge gained from solving the new problem, which is considered as learning. Both successful and unsuccessful cases may be stored as long as it may improve the systems decision making. This process involves choosing what information from a case to retain, how to index it, and how to integrate this knowledge into the general domain knowledge of the system.

3.3 Fuzzy logic

Fuzzy logic is able to simulate the decision making process of human by handling the concept of partial truth. In traditional logic, variables typically have binary values, true and false. Variables in fuzzy logic have truth values quantifying the degree of truth, ranging between 0 and 1.

Linguistic variables are often used in fuzzy logic instead of numerical values. The value of the linguistic variables are decided by a membership function that maps a numerical value to a linguistic value. An example is a variable, *temperature*, which can take the values *cold*, *warm* and *hot*. The values of *temperature* are then decided by a membership function, which is shown in Figure 3.4.

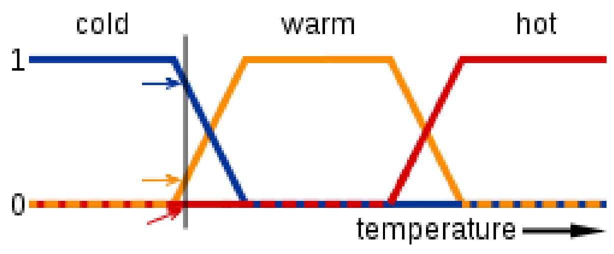


Figure 3.4: Membership function

Rules are often IF-THEN rules, or equivalently fuzzy associative matrices, and expressed as *IF variable IS property THEN action*. For example the rules for regulating temperature by controlling a fan is shown bellow:

```
IF temperature IS very cold THEN stop fan
IF temperature IS cold THEN turn down fan
IF temperature IS normal THEN maintain level
IF temperature IS hot THEN speed up fan
```

Often multiple rules can be fired, since e.g the temperature can be both cold and normal to a certain extent at the same time. The program will then evaluate all the rules that fired, and by using a defuzzification method, such as “center of gravity”, decide an appropriate response.

Defuzzification is done in order to quantify the result given in a fuzzy result set.

3.4 Evolutionary algorithms

Evolutionary algorithms (EA) is a subset of evolutionary computation, which uses mechanisms inspired by biological evolution in order to solve typically hard computational problems. These mechanisms include reproduction, mutation and natural selection. We have chosen to use genetic algorithms (GA) for our research, which is a type of EA.

The idea is to have a population of individuals consisting of a genotype and a phenotype. Initially, the population of individuals are typically randomly generated. The genotype of an individual represents a solution for a given problem, while the phenotype, or fitness, is calculated by evaluating the genotype by applying it as a solution to the problem. In order to be able to improve the solution given by the individuals, evolutionary mechanisms are applied by using an evolution engine.

An evolution engine evolves the population by applying chosen genetical operators to selected individuals, which then give a new generation. Such a cycle is depicted in Figure 3.5. Throughout the stepwise evolution of the population, it is likely that some of the individuals are approximating a good solution to a given problem. Some of the mechanisms often used by evolution engines are listed below:

- Crossover - This is used to simulate reproduction. Two individuals are combined in order to make a new one. Typically, this is done by dividing both their genotypes at a given location in their gene string, and then combine these genotypes to make a new one.
- Mutation - Is used in order to avoid the evolution to stagnate. Using just crossover may cause the population to quickly reach a local maximum as the lack of genetical diversity cause inbreeding. Mutation is often applied by randomly changing a part of the genotype.
- Selection - In order to keep the same size of the population throughout generations, a method for selecting the individuals which will advance to the next generation needs to be applied. A typical method for this is “roulette wheel selection”, this is done by giving each individual a slot in a “roulette wheel”. The size of this slot is based on each individuals fitness score. For each spin on the roulette, the individuals with the greatest fitness will have a better chance of being picked. This method will therefore prioritise the individuals which are considered the strongest, while the weaker individuals also may advance in order to preserve a diverse population.

Evolutionary algorithms have shown to often give good approximations to a large variety of problems, which is why it has been used in many different fields of research. When

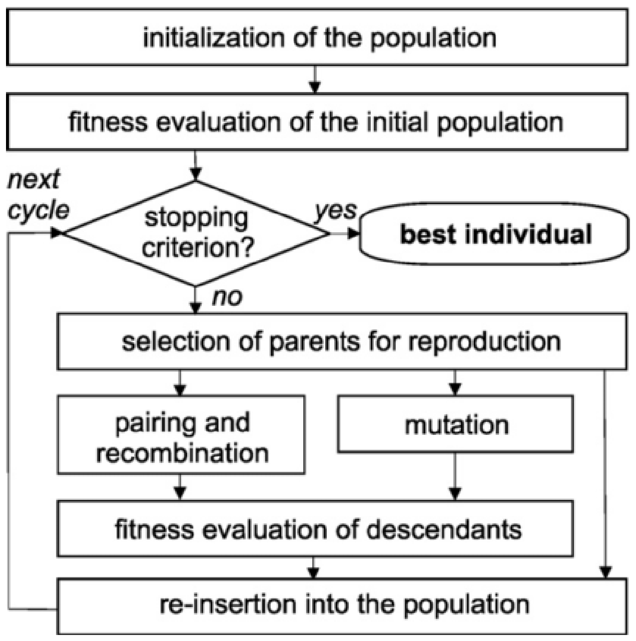

initialization of the population
fitness evaluation of the initial population
next cycle
stopping criterion?
yes
best individual
no
selection of parents for reproduction
pairing and recombination
mutation
fitness evaluation of descendants
re-insertion into the population

Figure 3.5: An example of a genetic algorithm cycle[7]

considering the use of EA in combination with CBR, it is often used in order to find a good approximation for feature weights[8][3]. The process of selecting good feature weights is quite difficult due to the vast amount of possible combinations of non-linear dependent variables that are available. Even though having a good intuition and domain knowledge may give an indication on how the features should be weighted, it does not necessarily result in satisfying feature weights. We did therefore decide to use evolutionary algorithms in order to weight our features.

3.5 Rule-based reasoning

In rule-based reasoning (RBR), domain specific systems use rules to make deductions or choices. The knowledge is represented as rules that makes up the whole knowledge base. Constructing a knowledge base is usually done with the help of experts on the given domain in order to make a good model. The inference engine, has an interpreter that cycles through all the rules in the knowledge base, matching the left side of each rule with the contents of the working memory. At the end of the cycle, if no rule matches, the interpreter halts. If one or more rules did match, the rules are executed, which typically alters the working memory as new information is gained. Then the interpreter starts over.

This process of incrementally changing the working memory by repeatedly executing the cycle, and continually deducing new information, is known as forward chaining.

An example is shown in the following enumeration:

Toby is an animal that looks like a duck, swims like a duck and quacks like a duck. We want to determine if he is able to fly. First we check the first rule, which in this case is true.

1. **IF** X looks like a duck, swims like a duck and quacks like a duck - **Then** X is a duck
2. **IF** X clucks like a hen, and lays eggs - **Then** X is a hen.
3. **IF** X is a duck - **Then** X is able to fly.
4. **IF** X is a hen - **Then** X is not able to fly.

None of the other rules will at this time match our current knowledge. At the end of the cycle, we execute the one matching line **IF** Toby looks like a duck, swims like a duck and quacks like a duck - **Then** Toby is a duck. We now add to our working memory the fact that Toby is a duck. During the next iteration, we will find that our new knowledge makes Toby match the 3. rule, and when executed, gives us the information that Toby is able to fly.

Compared to CBR, building and altering a RBR knowledge base is considered a lot more demanding, as the rules by themselves needs to cover the whole domain. CBR uses both general domain knowledge, and specific knowledge gained from cases. If a RBR system needs to be altered, this can be a tedious process, as the whole rule base needs to be checked and fixed for incompatibilities by adding or removing rules, which again can cause new problems. This can much more easily be done in CBR by adding and removing cases. Adding cases to a case base is also possible to do by non-computer experts, which means that a system does not need to be all done before deploying.

Chapter 4

Related research

A fair amount of relevant research has been done in the field of ITS and traffic. Expert systems in form of case-based or rule-based systems are represented, as well as systems which are based on probabilistic logic, such as fuzzy logic. Evolutionary algorithms are also represented.

Most of the work can be categorised into broad groups. Relevant to our project is "traffic management and control" and "traffic impact and safety"[43].

- "Traffic management and control" covers systems developed to help in traffic control and management operations by advising or assisting. This includes analysis and diagnostics of traffic, as well as incident detection and signalisation.

- "Traffic impact and safety" are concerned with systems that reduce the impact of traffic, such as accidents, noise control and investigation.

The number of studies using CBR for "traffic management and control" was limited, and we have included all the related research we could find that met our criteria on this topic. Similarly, we were not able to find many studies concerning "traffic impact and safety", but we were able to find some that focused on some of our desired criteria mentioned in Section 2.2, including CBR.

We have also chosen to present a research which applied fuzzy logic, despite the fact that it turned out to give a high amount of relevant results when using our keywords concerning fuzzy logic, as defined in Table 2.1. The reason for including only the one, is that we want to show how fuzzy logic has been applied in this domain, but we decided that the research using CBR was more relevant to our study.

Both RBR and evolutionary algorithms have also been applied within this domain. There was a limited amount of studies which we considered relevant. We have included the ones which we felt fulfilled our criteria the best.

Also, it turned out that neural networks have been frequently used within this domain, but as described in Section 2.2, this method was not one of our criteria, and have therefore not been considered.

4.1 Traffic management and control

Most of the CBR and RBR expert-systems are used for planning in an offline manner to help experts and analysts solve traffic related problems. One example is PLANiTS[15].

PLANiTS is a transportation planning support tool which determines similarity of old cases with new cases, and give information of whether it was a successful scenario or not, and in turn warns against potential failures and dangers. The system formalises cases in a structured and flexible manner, which helps the transportation planners to analyse and systematically determine similarity. Cases consist of an action which is implemented in an environment, and also the resulting impacts. In order to adjust the amount of cases that are returned, the strictness of the matcher can be controlled by the user.

Similarly, B. Schutter et al.[29] presents a system for assisting traffic operators in their decision process. Traffic operators typically have the possibility to control traffic flow on highways and urban roads using variable speed limits, dynamic route guidance, opening of shoulder lanes, and so on. The case-based approach helps the operators to more easily evaluate the traffic consequences of their actions. Results are in the form of a list ranked by outcome, where the best cases can be assessed more thoroughly. In order to make the system scalable, large networks are divided into smaller sub-networks, where each has its own case-base.

There has also been conducted research that focus more on systems that act in a more on-line manner, without having an operator interfering. These systems are often used for controlling traffic lights in urban intersections. Fuzzy logic based systems are responsible for the major part, but there also exist some based on CBR and RBR.

One of the multiple research involving fuzzy logic and traffic control are done by K. Tan, M. Khalid and R. Yusof. They describe an implementation of an intelligent traffic lights controller based on fuzzy logic technology[32]. The motivation of this research is the resulting traffic situation after Kuala Lumpur decided to use automatic traffic controllers instead of traffic policemen to control critical intersections in the city.

The system is capable of mimicking human intelligence for controlling the traffic lights by implementing rules similar to how humans may think. A human traffic policeman may use rules such as: "If the traffic on the north and south lanes are heavier than the traffic on the west and east lanes, then maybe the light should stay green longer on the north and south lane". Fuzzy logic, as described in Section 3.3, enables using similar rules with conditions such as "heavy" and "less", which are quantised and understood by the computer.

In contrast to more conventional traffic control systems, this system relies on counting cars in an intersection, and not just detecting the proximity of cars. By counting cars, the fuzzy controller can decide whether to extend green light, and for how long. The state machine controls the sequence of traffic light states, which the controller should cycle through. If there are no incoming cars detected, the controller may skip a phase.

Fuzzy variables are used to represent the number of cars arrived, number of cars in the queue, and the output that indicates the extension of green time. The membership function for the queue variable is shown in Figure 4.1. If there are zero cars in the queue, the value of the variable will be set to very small (VS). Similarly, if there is one car in the queue, the variable will to some degree be very small, and at the same time be small. Rules will therefore fire for both of them when the *queue* variable is checked.

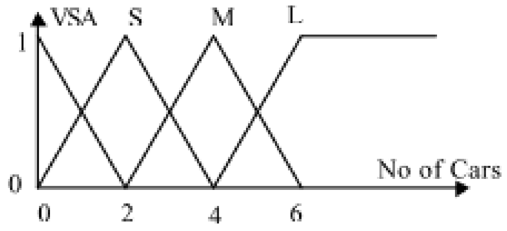


Figure 4.1: Membership function for the amount of cars in a queue

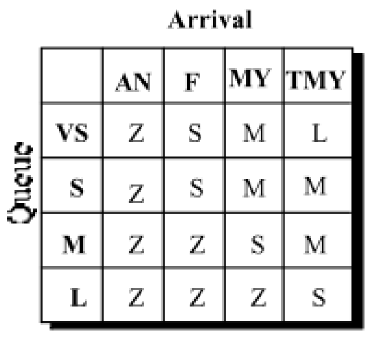
The fuzzy rule base is a collection of rules that are based on expert opinions, and thus reflects how a traffic policeman may make decisions. Following is one rule that an expert may construct:

```
If there are to many cars (TMY) at the arrival side
and very small number of cars (VS) queuing
then extend the green light longer (L).
```

These rules can be shortened as follows:

```
IF Arrival is TMY AND QUEUE is VS THEN Extension is L
IF Arrival is TMY AND QUEUU Is S Then Extension is S
```

Decision rules based on the input, can be also be presented in a matrix, like shown in Figure 4.3.



Arrival

Queue	AN	F	MY	TMY
VS	Z	S	M	L
S	Z	S	M	M
M	Z	Z	S	M
L	Z	Z	Z	S

Figure 4.2: Decision matrix with giving the decision for each input of the queue and arrival variables

When the proper rules are fired, the degree of membership of each outputted fuzzy variable is determined. All the fired rules and their outputs are combined, and the actual output is obtained through defuzzification.

Evaluation is done by comparing this system to other systems using a fixed-time or vehicle activated controllers. The results showed that the system using the fuzzy controller did perform better due to its flexibility. This flexibility is due to the extension of the green time for the lanes with high traffic density, using different lengths on the extension. Compared to using vehicle actuated control, where the green time is extended by a fixed value if a vehicle is detected, the fuzzy controller was able to reduce the total waiting time.

As mentioned, there are a lot more research focusing on fuzzy logic and traffic control, such as "Signal control using fuzzy logic"[21], however we decided to focus mostly on the CBR and RBR related research, as this was most relevant to our research.

RBR systems used for urban traffic control do exist, as W. Wen[42] shows. W. Wen suggests a solution to traffic congestion by presenting an expert system that dynamically and automatically controls traffic lights. The system focus on using RFID for the purpose of always knowing where every vehicle is within the network. Knowledge is represented in a rule base, and reasoning is done using a forward chaining procedure, as described in Section 3.5. However, by using RBR it can be difficult to formulate routing experience into simple rules, and rules may not cover every incident. An example of the rules in this system are:

```
if 3.0 < Interarrival_time then Signal_Type = "1"
if 1.7 < Interarrical_time <= 3.0 then Signal_Type = "2"
```

Part of the algorithm used for controlling all the traffic lights:

```
if(3.0 < Interarrival_time) {
then Red_light_duration = 65 and Green_light_
duration = 95;
```

Different values for the duration of lights are applied, and the results are recorded. This is done using a simulation model.

Sadek[26][27] presents the potential for using CBR to overcome the limitations of existing traffic management decision support systems. However, this is a complex task as the routing problem is computationally intensive. Strategies needs to be proposed in real time, and as soon as traffic conditions change, the new situation needs to be evaluated immediately. Route recommendations should also be a result of future estimation of traffic demand, not just the current traffic conditions.

A prototype CBR-system is developed and the initial case base is seeded by using a system based on a mathematical model. This system has been developed earlier to calculate routing plans. As it had turned out during earlier research, using a mathematical model to perform heavy calculations in order to solve this complex problem did not always give good results, as a result of the time needed for calculations, the solutions was not given in real-time.

The CBR-system is quite feasible when it comes to the problem of managing the traffic in real time by using a simple match/retrieve prototype system. Successful solutions are likely to be reused, and the traffic patterns are of a recurrent nature. Also by acquiring new cases, the system learns, and the case base is continually increasing and revised to improve the performance.

This research have also shown that it is possible to give good solutions even without complete information of the traffic situation. Such incomplete information may be caused by e.g sensor failures, and is an ever occurring problem in real life.

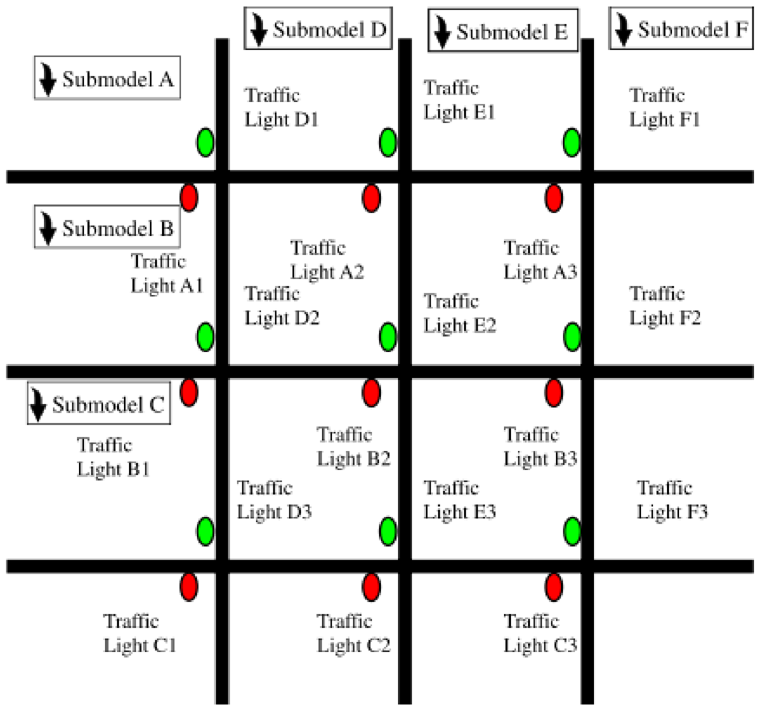


Figure 4.3: Grid, consisting of sub models, used for simulation by W. Wen[42]

In contrast to focusing on a large network of roads, Zhenlong et al.[44] proposes a CBR-framework to solve the problem of urban intersection control. The case structure, as seen in Figure 4.5, has been applied for the purpose of urban intersection control. A number of advantages of the proposed framework are presented.

Through simulations, this system was evaluated against a fixed time control cycle, and the intersection used can be seen in Figure 4.4. The fixed control cycle was calculated based on Webster's formula[41], which minimises the average delay. Results of the simulations show that the CBR-system gradually improved as it learned more cases. Also, by continually evaluating and discarding solutions with bad outcomes, it ended up outperforming the fixed time control system.

Other methods that have been applied, and have shown good results in this domain, are evolutionary algorithms. R. Hoar et al.[24], presents a swarm based traffic simulator that transmits the swarm behavior of ants into driver behavior. Optimisation of traffic lights are done by using an evolutionary algorithm and swarm voting. The cars are given a starting point and a destination in which they have to travel. Each car attempts to travel at its desired speed, but have to adjust to the nearby cars. Similar to ants, the cars communicate using pheromones, which contain information of speed and changes in its behavior. These pheromone traces are detectable by other cars, and their intensity will gradually get weaker, indicating how long since the previous car was at a given location, before disappearing.

The evolutionary design includes a fitness function, mutation operations, crossover, and swarm voting. Each car keeps track of its total driving and waiting time throughout its travel, which is then used to calculate fitness. The evolutionary operators are applied to the

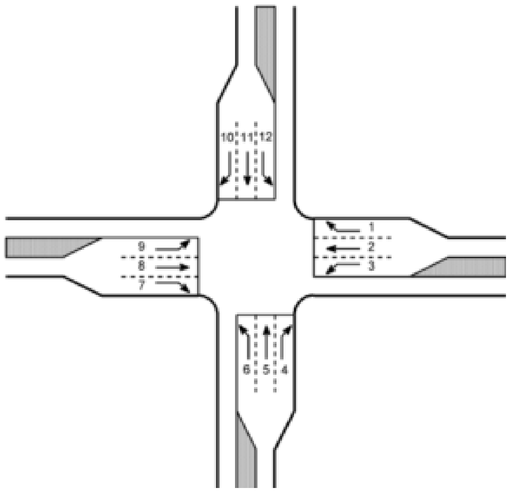


Figure 4.4: Intersection used for simulation by Zhenlong et al.[44]

signal plans for the purpose of producing phase sequences that maximise the flow. In order to initiate specific adaptions more quickly, the cars cast votes for the intersections in which they are about to encounter. When many cars have voted for a given light, the mutation probability of it to prolong its green time, or decrease the ones of the other lights, are higher than normal.

Simulations using different maps have been done, giving good results as the overall waiting time is dramatically decreased throughout the evolutions.

4.2 Traffic impact and safety

Traffic impact and safety is a very important aspect when concerning the current traffic domain. This has to some degree been focused on by the research done on expert systems. Most of these are focused on helping users by planning and analysing earlier events, and then by offering solutions to similar problems.

Boury-Brisset et al.[9] describe the architecture of an organisational memory, which is shown in Figure 4.8, that can be used for road safety analysis. It is mentioned that the current completely automated expert systems have not proved to give satisfactory results[11], and in order to overcome these problems, systems that aid the decision process can give preferable results.

The knowledge of SICAS (System with Intelligent and Cooperative functions to help in the Analysis of Sites) is acquired by studying existing domain-related documents, as well as interviewing experts. Since these documents already has a case-like structure, which makes the process of acquiring cases less complicated, it is one of the main reasons why case-based reasoning seemed like a promising approach. This approach also makes it very natural to combine the experience from multiple traffic engineers. The provided knowledge editor makes it possible to gradually build and refine the expertise model by submitting and editing cases.

Kaidong Li and Nigel M. Waters[16] presents the need for intelligent systems to facilitate

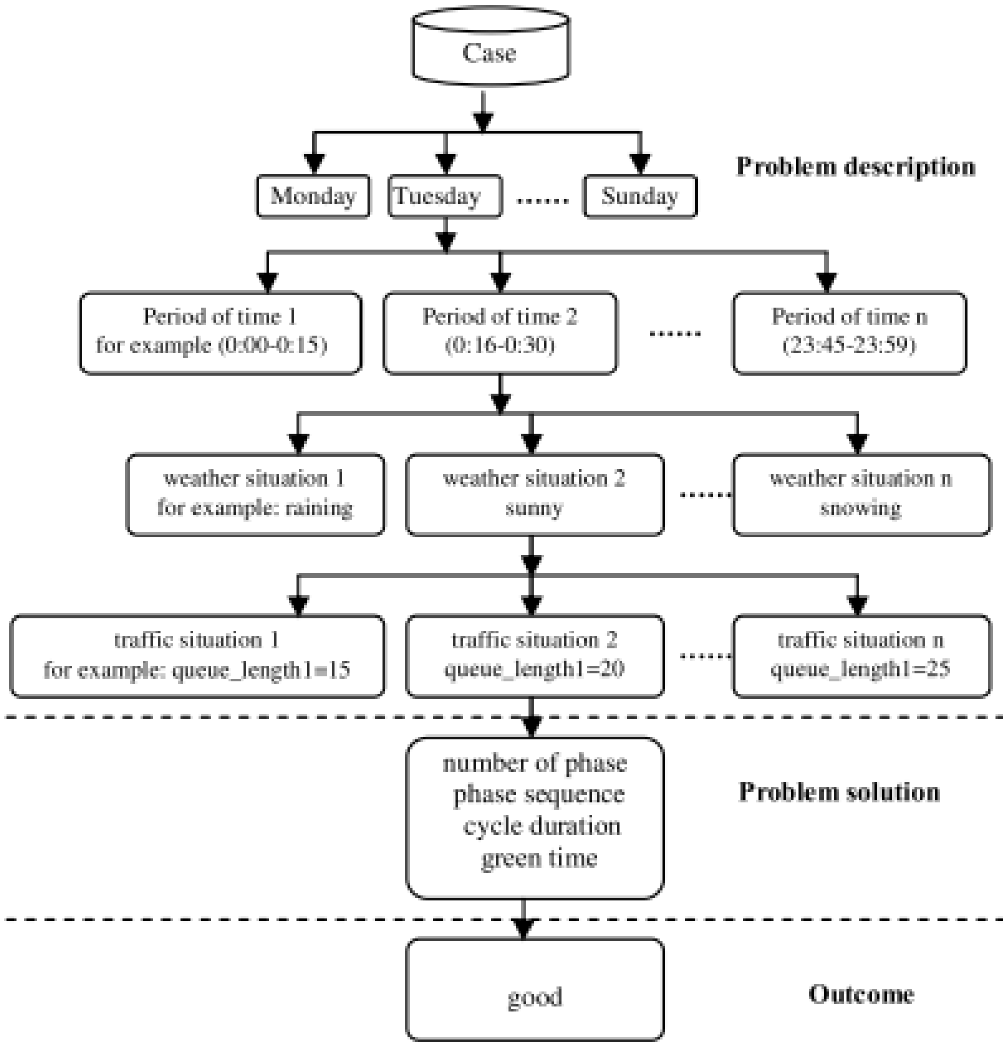

Case
Problem description
Monday
Tuesday
Sunday
Period of time 1 for example (0:00-0:15)
Period of time 2 (0:16-0:30)
Period of time n (23:45-23:59)
weather situation 1 for example: raining
weather situation 2 sunny
weather situation n snowing
traffic situation 1 for example: queue_length1=15
traffic situation 2 queue_length1=20
traffic situation n queue_length1=25
number of phase
phase sequence
cycle duration
green time
Problem solution
good
Outcome

Figure 4.5: Case organisation proposed by Zhenlong et al.[44]

and eventually enhance the safety in public roads and in other infrastructures, referred to as a geographic information system (GIS). Rule-based systems have been quite common within this domain, but case-based systems have gained increasing interest, especially in areas where situations are complex and rules are hard to properly generalise. In many cases, CBR compared to RBR, does not need an explicit and complete model of the domain to give good results, which makes the construction of such a system more affordable. Also, prior experiences may provide more specific information given different sites, compared to the rules located in the knowledge base of an RBR-system, which often are more general.

The CBR tool used in this study is delivered by eGain, and the structure of the cases can be seen in Figure 4.7. Knowledge is represented by questions and then grouped by clusters, such as “collision history” and “traffic and physical characteristics”.

We have mainly presented CBR systems used for the purpose of traffic impact and safety. The following research has been done considering the pollution caused by traffic, and whether

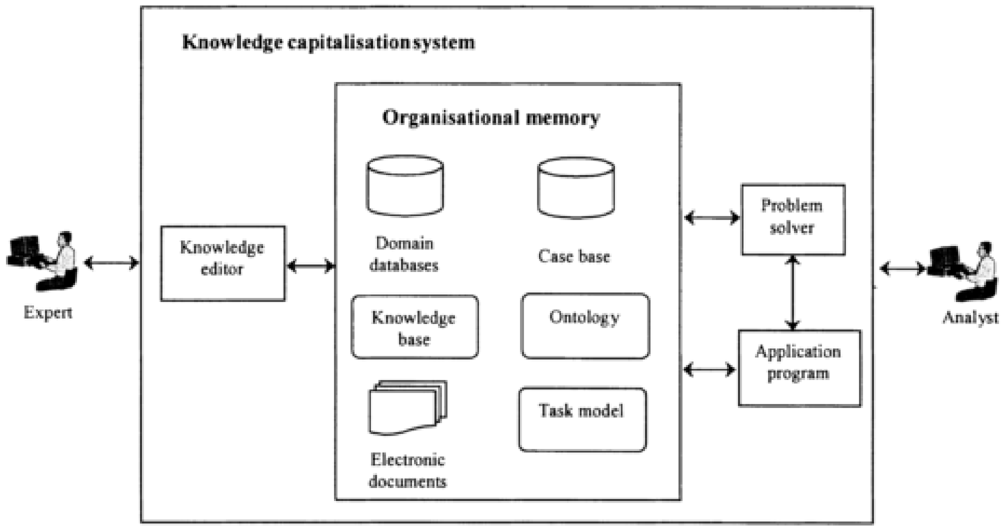


Figure 4.6: Components of SICAS

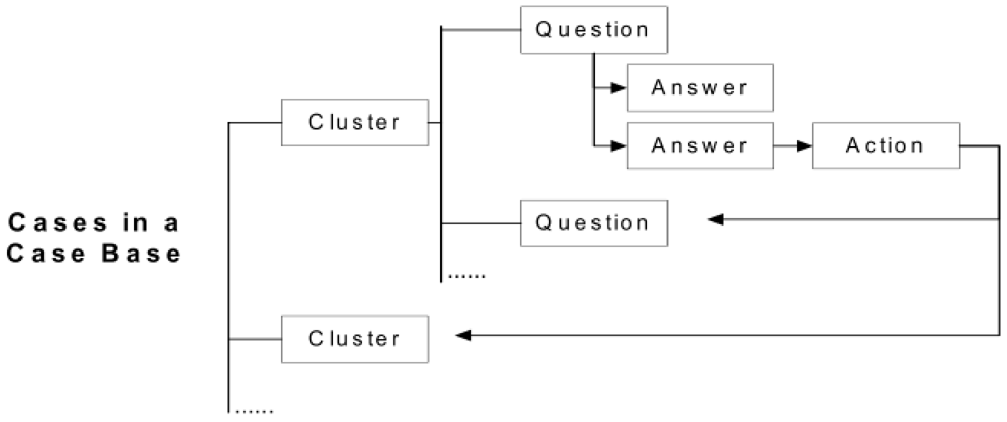


Figure 4.7: Case base structure in eGain

this can be feasible to incorporate in a fitness function of a GA, when calculating signal plans.

J. Medina et al. [19] have studied the correlation among traffic flow, greenhouse emissions, and network occupancy using evolutionary algorithms. In earlier research, they used a genetic algorithm for generating the signal plans, and a traffic microscopic simulator to help determine the fitness of the solution. This is based on the amount of vehicles that has left the network at the end of the simulation. In this research they have aimed to also include the occupancy of the network along with the total greenhouse emissions as new criteria.

In order to calculate the emissions, they have chosen to use the scalar value of the speed of every vehicle, as CO and NO_x are mostly linearly correlated with the speed of vehicles. The global sum of emissions are calculated for each simulation. Two different real life networks are simulated in order to determine if the emission values are suitable to be part of the fitness function. Results show that there are a strong linear and positive correlation between the occupancy of the network and the emission criteria. Because of this, a hypothetical function incorporating multiple criteria fitness function, they will have to choose between maximising the occupancy of the network, or minimising the total emissions.

In Singapore, Y. Wang et al.[40], have researched a methodology for scheduling of pavement maintenance activities that involves lane closures, with the purpose of minimising the total traffic delay in the network. A genetic algorithm is used for generating maintenance schedules, which is a complex constraint optimisation problem.

A population consists of different schedules, deciding which location is to be maintained at the given time. The schedules spans a 24 hour period, in which maintenance are needed on random locations. Fitness of an individual is calculated by calculating the average delay of the traffic going through the network, given by simulations.

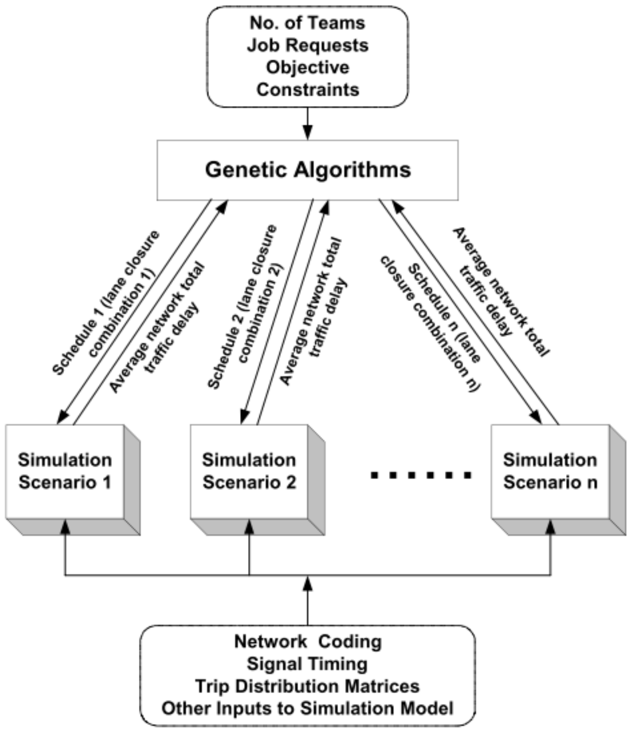

No. of Teams
Job Requests
Objective
Constraints
Genetic Algorithms
Schedule 1 (lane closure combination 1)
Average network total traffic delay
Schedule 2 (lane closure combination 2)
Average network total traffic delay
Schedule n (lane closure combination n)
Average network total traffic delay
Simulation Scenario 1
Simulation Scenario 2
Simulation Scenario n
Network Coding
Signal Timing
Trip Distribution Matrices
Other Inputs to Simulation Model

Figure 4.8: Framework of the Hybrid GA-Simulation Scheduling presented by Y.Wang et al.[40]

Results show that the evolution of the maintenance schedule has been directed towards closing lanes at non-peak hours. This study has shown that the proposed methodology is suited for the problem presented.

Chapter 5

Process of deciding the research goals

At the start of our thesis work, there were a lot of approaches that seemed viable to focus on for the purpose of answering the research questions. An effort was done deciding whether or not the given approaches were feasible, giving indications on whether or not we should continue working in that direction.

In order to manage this process we followed the method described in section 2.1. This method iterates through a cycle consisting of four main steps in order to incrementally get closer to a feasible problem definition.

The first section presents the initial steps we did in order to get a basic foundation we could use to start the decision cycle. Each of the following sections present such cycles, giving first a short overview and then discussing the details.

5.1 Initial steps

Initially, it was desired that CVIS (Cooperative Vehicle-Infrastructure Systems)[1] in combination with an existing driving simulator, should be part of the research question. Example scenarios included car-to-car communication, in order to improve local traffic flow.

We found the idea of working on improving traffic flow intriguing, but wanted to focus on a more holistic picture, such as urban intersection control, and decided to do some research on this topic. After meeting a traffic engineer at SINTEF, we got some information on relevant documents[37][38] to read, and whom to contact in order to get to know how traffic is controlled in Trondheim. While reading these documents, we scheduled a meeting with one of the people that have been working on the system in Trondheim, Ørjan Tveit.

During the meeting with Ørjan Tveit, we were told about the traffic control system that is used in Trondheim. The system SPOT/UTOPIA, presented in Section 2.4.2, is an adaptive control system which offers good support for public transport priority. Earlier researching on this system for use in Trondheim has involved using a traffic simulator [36][4].

It turned out that it would be possible to acquire an issue of SPOT/UTOPIA if our problem would involve usage it. In order to gain more information regarding the system, we were able to obtain the quick reference documentation[6]. The document presents the

[1]CVIS, http://www.cvisproject.org

architecture of the system, its functionalities and an overview of the mathematical functions that are used in order to construct the signal plans.

5.2 The first approach

Initially, we had the impression that SPOT/UTOPIA had a rule-base which was used in order to make decisions. This was based upon a discussion we had with the NPRA about how SPOT/UTOPIA makes its decisions. However, this was not correct, as will be elaborated soon. This assumption encouraged us to look at the possibilities of combining the potential rule-based system with a self made CBR-system. The idea of improving the system was something that caught our interest as we would have the possibility to contribute to an infrastructure component used in multiple cities and metropolises.

1. **Research step**: Doing research on the domain of combining CBR- and RBR-systems uncovered that this has been done successfully as described in [13]. After having a closer look on the documentation of SPOT/UTOPIA, it was uncovered that this was not the case, as its decision process is based on mathematical functions.

5.2.1 How SPOT/UTOPIA makes decisions

The following equations can be seen in the appendix of the SPOT/UTOPIA documentation along with a thorough explanation, giving an idea of how the intersection control is done.

Intersection state observer functions

In order to be able to calculate optimal signal plans, it is necessary to know the state of an intersection. The state x is a composition of smaller state vectors that each represent an incoming road link. These state vectors contain cars already in that link and also incoming cars ordered by predicted arrival time at the stop line. When calculating an optimal signal plan, it is almost as important to know future states as knowing the current state. This is presented in the state propagation equation.

$$x_{k+1} = f(u_k, x_k, c_k)$$

Figure 5.1: State propagation equation

Where
k is the current step.
u are the new arrivals on the incoming links.
c is the current signal plan.

Approximations using assumptions are done by the function in order to propagate through states with realistic results.

Intersection control functions

Signal plans are calculated every 3 seconds, and span a time horizon of 120 seconds.

The function sums the different cost elements calculated on the optimisation horizon, and selects the signal plan with the overall lowest cost. Each of the cost elements have a defined weight based on priority of the cost element.

$$min_c \sum_j w_j a_j(x)$$

Figure 5.2: Optimisation function

Where
w_i is the weight of the cost element 'j'.
a_i is the cost element 'j'.
c is the signal setting.

Examples of cost elements are the amount of cars waiting at a stop line, time spent by public transport vehicle assigned priority, and time spent by vehicles on incoming links. For the purpose of maintaining intersection control coordination with multiple intersections, there is a cost element for the time spent on outgoing links by vehicles leaving the intersection. Since the signal plan is calculated each third second, the signal plan can potentially change each time. In order to avoid this, and to maintain smoothness, there is a cost element for the deviation from the signal setting decided at the previous iteration. This is more elaborated in the quick reference guide[6].

5.3 The second approach

The first iteration of the decision cycle gave some more insight in how the system makes decisions, which gave basis for our next approach. The cost elements that the system uses in the mathematical calculations which was presented in the previous section, is of great significance regarding the resulting signal plans. It would therefore be interesting to investigate how these weights were assigned. This led us to the following approach:

1. **Research step**: By contacting experts with knowledge of the SPOT/UTOPIA system, we were able to uncover that these weight are in fact manually adjusted when configured. Because of the amount and diversity of the cost elements, we had reason to believe that the process of setting the weights could be improved.

2. **Draft step**: Before doing more research on this field, we decided to make a basic draft of the problem and a possible approach in order to get feedback from the supervisors as soon as possible. Using AI-methods in order to adjust these weights and then doing simulations for evaluating the results, was the essence in this draft.

3. **Approval step**: When discussing the problem with the supervisors, we agreed upon that this problem would be a to narrow problem definition and that we should try to focus on a problem were we could implement a prototype system.

5.4 The third approach

In earlier research done at the Norwegian University of Science and Technology (NTNU) [4][36], SPOT/UTOPIA, described in Section 2.4.2, has been used in combination with Aimsun, as described in Section 2.4.2. This combination, illustrated in Figure 5.3, has made it possible to simulate traffic flow by using SPOT/UTOPIA. There also exists a programming interface for Aimsun which makes it possible to transfer information with Aimsun and other applications. By combining SPOT/UTOPIA, Aimsun and a CBR-system, it is possible to train a CBR system by using SPOT/UTOPIA, and then evaluate both systems by simulating in Aimsun.

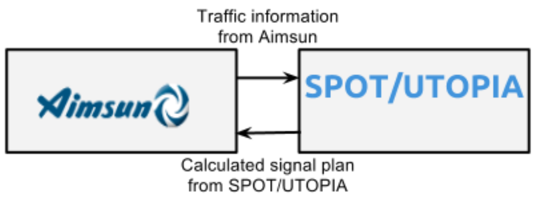

Traffic information from Aimsun
Aimsun
SPOT/UTOPIA
Calculated signal plan from SPOT/UTOPIA

Figure 5.3: Aimsun and SPOT/UTOPIA

1. **Research step**: We had to uncover whether it was possible to obtain the necessary software in order to be able to continue. After doing some inquiring, it turned out that an Aimsun license could be acquired, and also a working copy of SPOT/UTOPIA.

2. **Draft step**: Using SPOT/UTOPIA and Aimsun in order to train a CBR-system, which then uses this knowledge, and knowledge obtained through machine learning when running, was expected to be a fun and challenging problem to try solving.

3. **Approval step**: Meeting with the supervisors was quite successful and we agreed to continue working on this new problem.

4. **Decision step**: Doing a closer look on the Aimsun scripting API, it seemed to support the functionality needed in order to implement the prototype system. Obtaining a copy of SPOT/UTOPIA was to be done when meeting a traffic engineer from Swarco, so we decided to initiate the next steps of the research. However, as it would turn out, combining these systems was not a trivial task, and due to the circumstances we would have to change the problem description as described in Section 5.5.

5.4.1 Drafting: Training a CBR-system on SPOT/UTOPIA

In order to train a CBR-system on SPOT/UTOPIA, it is necessary to use proper traffic simulation software to get close to real-life traffic data, which in turn can be applied to SPOT/UTOPIA. Aimsun provides a module for connecting to different adaptive control systems, including SPOT/UTOPIA, as well as a programming interface for self-made systems.

One possible way of training the CBR-system is illustrated in Figure 5.4. Aimsun will continuously provide traffic data to SPOT/UTOPIA that calculates a signal plan each third second, and sends it back to Aimsun, which then applies it. At the same time, both the traffic data, and the signal plan are available through the programming interface in Aimsun. The CBR-system can then get these data and each third second acquire new cases. The traffic data will typically be extracted as case features, while the signal plan will be a solution.

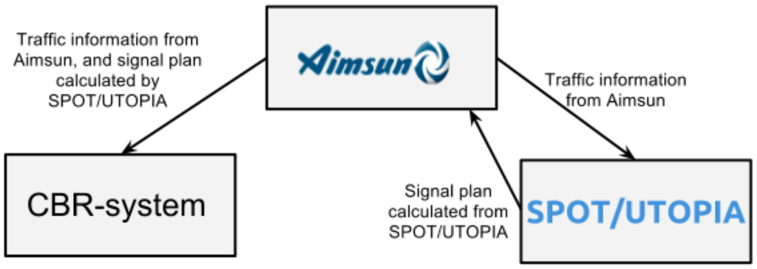

Traffic information from Aimsun, and signal plan calculated by SPOT/UTOPIA
Aimsun
Traffic information from Aimsun
CBR-system
Signal plan calculated from SPOT/UTOPIA
SPOT/UTOPIA

Figure 5.4: Training of the CBR-system

5.4.2 Obtaining the SPOT/UTOPIA software

By the help of Ørjan Tveit we were able to meet a traffic engineer from Swarco who is configuring the SPOT/UTOPIA software for use in Trondheim. We met with him twice in order to learn more about the system, and in the end, get our own issue of SPOT/UTOPIA.

5.4.3 Decision: Technical problems

After spending some time trying to combine the SPOT/UTOPIA and Aimsun software, we soon realised that we were not competent enough to by ourselves make the proper connection between the systems. We would also need to have a corresponding model of the traffic network in both these systems, which we were not sure how to accomplish.

As we already had used longer time in our decision process than originally planned, and by now realising that we needed to change our problem description, we needed help in order to decide our next step. This is described in the next section.

5.5 The final approach

This section describes the final iteration of the decision cycle that we did in order to make the problem description. After doing this, we could focus on starting our research.

As we already had used longer time in our decision process than originally planned, and by now realizing that we needed to change our problem description, we needed to urgently take action.

1. **Research step**: By doing the previous cycles we felt that we had gained knowledge and experience to quickly start drafting an alternative goal. We set up a meeting with people from the NPRA and the supervisors in order to discuss and hopefully draft a new problem description.
2. **Draft step**: By dismissing SPOT/UTOPIA and doing manual training of the system, and then evaluate against static signal plans, we could still be able to make a system that could be used as a "proof of concept".
3. **Approval step**: By drafting and assessing the new goals at the meeting, we all took part in the process and finally agreed upon the problem description.
4. **Decision step**: We did not see any big technical problems that would inhibit our research. So we decided to continue with this approach.

5.5.1 Drafting: Aimsun combined with CBR

During the meeting where the drafting took place, multiple options were discussed. A model of an area in Trondheim using the combination of Aimsun and SPOT/UTOPIA was to be presented on a conference in October by the NPRA, which meant that they also needed to obtain a working combination of the two systems. This set-up could also be available for us to use. However, this could not be guaranteed to be working before the time of the conference. In that case, it would therefore give us too little time concerning the implementation, which we evaluated to be a too high risk for us to take.

We therefore discussed the next best alternative, which was to exclude SPOT/UTOPIA, and just use Aimsun for simulation using the CBR system. The CBR-system would be trained using cases added manually, and evaluated against static signal plans, not SPOT/UTOPIA as originally planned. An illustration is shown in Figure 5.5.

The resulting research goals of this draft are described in Section 1.1. The rest of this report shows the research results reached by focusing on solving these goals.

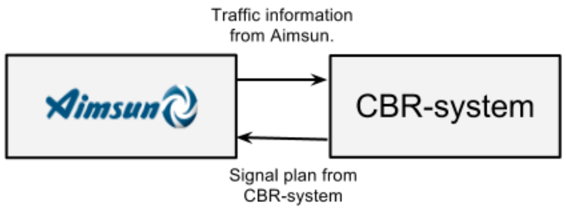


Figure 5.5: Using CBR together with Aimsun

Chapter 6

Implementation

This chapter presents our implementation as a result of our findings described in the earlier chapters. The first two section presents an overview of our system, first a functional overview, and then a more detailed one.

Our case base, along with its features, is presented and described in Section 6.3. This includes how the design changed during the research followed by the final result. How the case-base was weighted by an evolutionary algorithm, and some weaknesses, are presented at the end of this section.

The retrieval process is described in Section 6.4. This shows how the CBR-system decides which case to retrieve, after the similarities have been measured.

How the CBR-system reasons towards a signal plan, after a case solution is retrieved, is presented in Section 6.5

A model of the intersection, based on the one presented in Section 2.3, is described in Section 6.6.

The static signal plans used in the real life intersection, and how they are modified for our model, is presented in 6.7.

Finally, some implementation details are presented in the last section, Section 6.8.

6.1 Functional overview

Our system consists of three main modules: the CBR-system, the simulator manager (SM), and Aimsun. The modules, along with their input/output, are displayed in Figure 6.1.

When running the system, the SM will, based on user input, select a scenario to run. The chosen scenario is sent to the CBR-system which retrieves a predicted traffic flow based on the features of the scenario. A signal plan is then calculated, and returned to the SM. The received plan, together with scenario specific information is configured and applied to Aimsun, before running a simulation. After simulations have been run, the results are displayed in Aimsun.

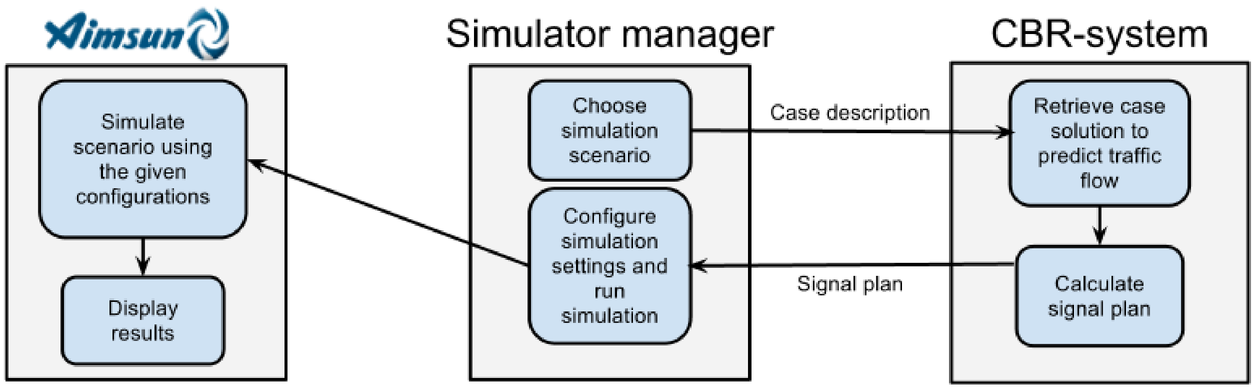

Aimsun
Simulator manager
CBR-system
Simulate scenario using the given configurations
Display results
Choose simulation scenario
Configure simulation settings and run simulation
Case description
Signal plan
Retrieve case solution to predict traffic flow
Calculate signal plan

Figure 6.1: Functional overview of the system

6.2 System overview

A more detailed overview of the system can be seen in Figure 6.2. As mentioned, there are three main components, the CBR-system, the SM, and Aimsun. The SM has access to Aimsun through its Python scripting interface, and is therefore also written in Python. The CBR-system uses the jCOLIBRI-framework, which is implemented in Java. Inter-process communication between the SM and the CBR-system is performed through sockets.

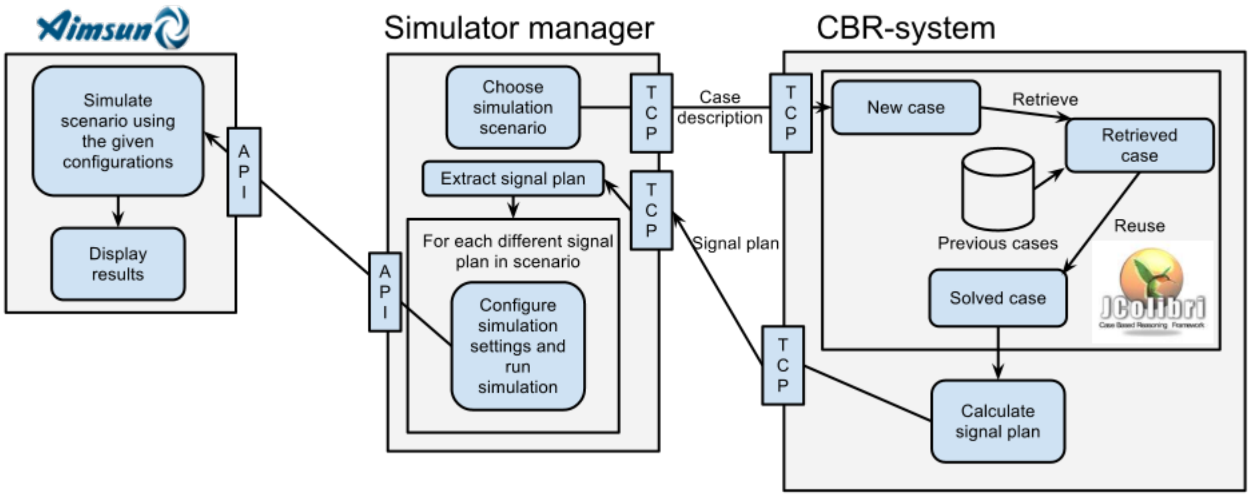


Figure 6.2: System overview

Traffic scenarios, which are presented in Section 7.2, are entered by the user into the SM. The scenarios are structured as problem descriptions, as can be seen in Section 6.3.3.

When the SM has retrieved the traffic scenarios from the user, the necessary features are extracted and forwarded to the CBR-system. A new case is constructed based on the features, and the system will then retrieve similar cases. When a good solution candidate is selected, the CBR-system uses the solution, consisting of a predicted traffic flow, and calculates a signal plan. This plan is then sent back to the SM.

The SM applies the scenario specific information, such as traffic demand, to Aimsun. This configuration will be reused for each different signal plan which is to be applied. For each new signal plan, a simulation is run. After Aimsun has finished a simulation, the results are displayed and stored in its local SQL-database.

More implementation details can be seen at the end of this chapter, in Section 6.8.

6.3 Case base

This section describes how the structure of the case base has evolved throughout the research, and how the final result emerged.

6.3.1 Drafting the case structure

At the start of the design process the case structure had a quite similar structure to the one presented by Zhenlong et al[44], and depicted in Figure 4.5. The initial draft for the problem description and solution is presented in Table 6.1 and 6.2

Feature	**Values**
Date	Day/Month/Year
Weekday	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
Time of day	A time interval (00.00 - 00.30)
Weather	Sunny, Cloudy, Precipitation, Heavy precipitation
Temperature	Hot, Medium, Freezing
Queue length for lane L1, L2, ...	Integer values separated by commas (4,6,2)
Traffic flow for lane L1, L2, ...	Integer values separated by commas (200, 60, 300)
Special event	None, Football match, Rock concert, Road accident

Table 6.1: Draft of the initial problem description

Feature	**Values**
Phase sequence	Integer values separated by commas (10,2,15,2)

Table 6.2: Draft of the initial problem solution

Time based features

The date may not always be an important feature to consider, in contrast to the weekday which always are important. However, dates do hold information that can affect the traffic a great deal, often unaffected by the day of week. This includes Christmas, holiday, national holidays and other days that on a yearly basis causes irregular traffic flows.

Some assumptions regarding the time of the year can be also done. One may expect that snow and freezing temperature in September in Trondheim, which is rather uncommon, may cause a higher impact to the traffic than snow and freezing temperature in the late winter. This can be important because people may not have prepared for this kind of driving conditions by changing tires etc, and also because people are not yet used to driving in such conditions. In February most people have gotten much practice driving in these conditions, combined with proper tires, so the impact of snow and freezing temperature may be less significant.

As mentioned, which day of the week it is will under most circumstances have a great impact on how the traffic will be. Typically, Monday to Friday are the most trafficked days, but there are also some differences between those days, although not as significant as the weekend compared to the rest of the week.

Figure 6.3 show the traffic volume, in cars per hour, during a week in September, Monday to Sunday.

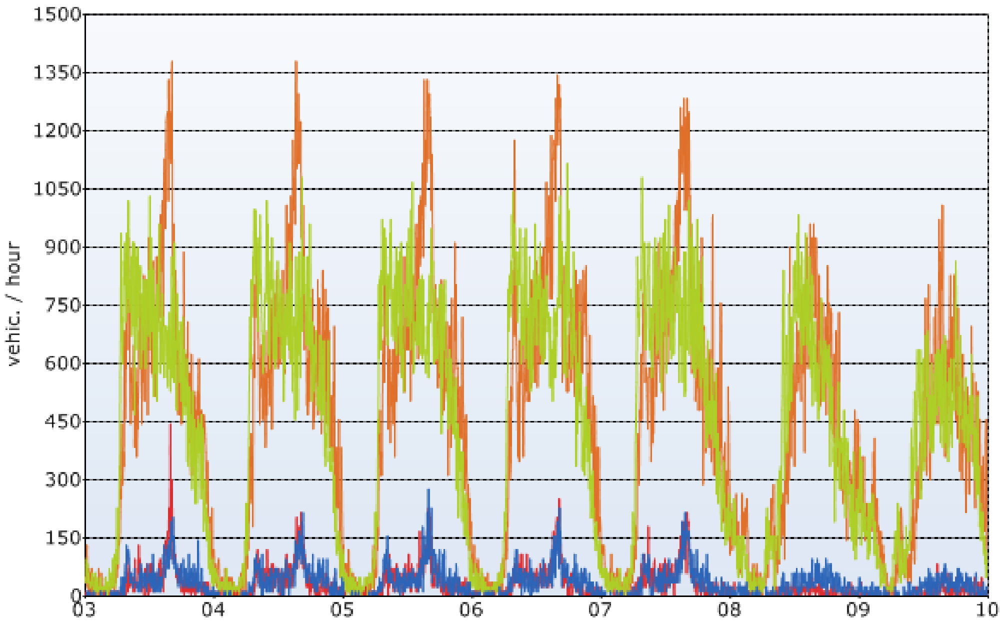


Figure 6.3: Traffic volume Monday to Sunday. The orange graph shows traffic going south, out of the city; while the green shows traffic going north, in towards the city. Red and blue shows traffic originating from east and west of the intersection.

Time of day are of great importance as many people commutes by car back and forth to their work. In Trondheim, the traffic shows clear spikes in both the morning and the afternoon rush. Other countries and cities may have other traffic patterns throughout the day. For example, it may always be a high traffic demand from 6AM to 6PM.

By using these features that considers date, weekday and time of day, this should cover most relevant information regarding time. By themselves they are important, and by combining them, they are likely to find patterns that are influential to the traffic.

Non-traffic environmental features

Weather and temperature affect the traffic in ways that are visible. Especially whether or not it precipitates combined with whether or not the temperature is freezing, may affect the behavior of drivers. Both the traffic flow and the amount of cars are affected by the weather and temperature, as shown in multiple research papers. [18][10].

Special events may bring a drastic change to the usual traffic pattern. People attending football games often causes unusually big pressure on the traffic network since everyone leaves at the same time. If the traffic lights are configured to prioritise the traffic going in to the city by default in the nighttime. Then, when a special event occurs, and when everyone leave the city, the traffic will move much slower than it could have. When there are known special events, this feature should clearly affect the decision of the system.

Traffic environmental features

If possible, it is desired to use real time traffic data for our system. By having real time data it is possible to estimate the current traffic flow, queue lengths, and also uncover unexpected patterns in the traffic. This will also give the possibility of estimating how many cars are in the proximity. In order to make a true adaptive system, this will be the most important features to incorporate in the case-base. This information is also used by SPOT/UTOPIA, described in Section 2.4.2.

6.3.2 Technical difficulties

The problem

During the implementation of the system, it turned out that even though the API provides a possibility to change signal plan, even during a simulation, the new plan will not be used in the ongoing simulation. Only when a new simulation is started, the plan that was changed during the previous simulation, is used.

This restricts the ability to be adaptive, as the signal plan cannot be changed and immediately applied during simulation, and therefore greatly limits the gain from knowing the current traffic state. Even though the system detects a large queue in one lane, it will not be able to change the signal plan and facilitate the results.

Proposed solution

In order to circumvent this problem we propose an alternative way of utilising CBR in order to control traffic.

We need to find the signal plan that will give best results within the time scope of the simulation. For this purpose, the features that represent traffic state in the initial problem description will not be useful, as we cannot change the signal plan anyway during the simulation.

However, features that describe historical traffic flow may be of use. Traffic counts are often conducted using induction loops and other counting tools, such as radars. Also, counting of traffic is quite common as this data is used for many purposes. It is therefore assumed that the traffic counts needed in order to gain accurate historical data are highly available on most roads close to urban intersections using traffic lights.

In order to have a system that responds in real-time, it is necessary to have accurate information on queue length and traffic flow in each lane. This demands more equipment, and is not as common as equipment used just for counting. While this may give a more efficient and adaptive system, it is also more vulnerable to hardware failure. If detectors stop working, the system does not know which lanes that should be prioritised, as shown in the research done by Sadek[26][27].

6.3.3 Final case structure

Feature	**Values**
Season	Winter, Spring, Summer, Fall
Weekday	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
Time of day	Night, Morning, Noon, Afternoon, Evening
Weather	Sunny, Cloudy, Precipitation, Heavy precipitation
Temperature	Hot, Medium, Freezing
Friction	Dry, Wet, Icy
Special event	None, Football match, Holiday

Table 6.3: The final problem description

Total traffic flow for the different lanes	Integer values separated by commas (330, 484, 28, 23, 27, 28)

Table 6.4: Structure of the problem solution.

The most important change in comparison to the first draft of the case structure is that the solution now contains a prediction on traffic flow. This was originally a feature that was a part of the drafted case structure, but since technical limitations prohibited us from using this information, as has been described, we opted for this approach.

As the solution does not contain a signal plan, this has to be calculated. The signal plan is reasoned towards using the case problem description and solution, combined with additional domain knowledge, which is more elaborated in Section 6.5.

Road condition	**Friction coefficient**
Wet ice	0,05-0,15
Dry ice	0,15-30
Dry sand on icy/snowy roads	0,25-0,35
Warm wetted sand	0,30-0,50
Wet road	0,40-0,90
Dry road	0,80-1,00

Table 6.5: Road conditions and corresponding friction coefficients

Feature	**Discrete value**
Season = Date	Dates are discretised into the 4 seasons
Day of week	Already discrete
Time of day	Night = 00.00-01.00, Morning = 07.00-08.00, Noon = 12.00-13.00, Afternoon = 15.00-16.00, Evening = 21.00-22.00
Weather	Already discrete
Temperature	Freezing = bellow 0°, Medium = 0-20°, Hot = above 20 °
Friction	Icy = 0.30, Wet = 0.90, Dry = 1.00
Event	Already discrete

Table 6.6: Feature value range

Non-traffic environmental features

Another change is that we have chosen to consider the friction coefficient of the road. There exists multiple different tools for determining this, such as a device pulled by a car in order to measure deceleration. More interesting for our research is the Viasala measuring device[39], which performs laser spectroscopy in order to determine the substances on the pavement, hence identifying the friction.

We therefore assume that such a measuring device is placed at the intersection in order for us to obtain this friction coefficient. Examples of different road conditions and their correlating friction coefficient are presented in a textbook for operation and maintenance of roads[31], and is illustrated in Table 6.5.

We have chosen to use "Dry road" = 1,00, "Wet road" = 0,90 and "Wet ice" = 0,15 as possible values for the friction coefficient.

Discrete values for features

In order to limit the amount of possible values our problem descriptions can have, we have decided to discretise some of the values. We have therefore defined which range a feature value needs to have in order to get a corresponding discrete value. This is shown in Table 6.6.

6.3.4 Similarity functions

As all features have multiple possible values, it is likely that some of these values are more similar to each other than others. This knowledge is important to incorporate into the system in order to conserve the relationship between concepts from the real world into our model. The similarity functions represent the similarity between the attribute values of the classes.

jCOLIBRI supports the use of different similarity functions. These include equal, interval, and threshold. The one we have chosen to use is the table similarity function that gives the possibility to assign a similarity value between 0 and 1, where 1 is exactly the same, for each of the attribute values of the different classes. We decided to use this as it is more customisable than the other available functions, and since we feel that we have enough domain knowledge to utilise this functionality.

The similarity values are manually set based on our acquired information of our research on the domain, and with some help from experts. It is therefore important to notice that these values might be improved by using methods for setting these values, or by having domain experts to set these.

Season similarity

The seasons in Trondheim can be quite varied. Winter and summer are often different when it comes to the traffic conditions. Icy and slippery roads often occur in the winter time caused by snow and ice, and sometimes even in the early spring and late fall aswell. Compared to the summer, wet roads are typically the worst road conditions related to that season. Both spring and fall are somewhat similar. In the late fall, snow is a common sight, which is also the case for early spring. The closer one gets to the summer, the road conditions are mostly wet or dry.

Whether or not people choose to drive or use the bicycle also depends to somewhat degree on the season and the weather, and therefore also affect the traffic.

Feature	**Winter**	**Spring**	**Summer**	**Fall**
Winter	1	0.8	0.4	0.6
Spring	0.8	1	0.65	0.65
Summer	0.4	0.65	1	0.7
Fall	0.6	0.65	0.7	1

Table 6.7: Season similarity

Day of week similarity

When determining the similarity of the days in a week, it is quite clear that the biggest difference is between the weekend and the rest of the weekdays. We have therefore decided to put a similarity of 0.9 between the weekdays, except for Friday which have a similarity of 0.8. The reason is that we assume that this day may differ a little from the other days concerning the working hours. People are more prone to leave at different hours than they normally would, and this day may also be the one day most often skipped when people opt for a “long weekend”.

Saturday and Sunday stands out, obviously because most people do not work on these days. They also differ because of more traffic in the night time, which both days typically have. Since Sundays does not have most stores open, there is generally less traffic than on a Saturday, except when there are football matches.

Feature	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
Mon	1	0.9	0.9	0.9	0.8	0.5	0.4
Tue	0.9	1	0.9	0.9	0.8	0.5	0.4
Wed	0.9	0.9	1	0.9	0.8	0.5	0.4
Thu	0.9	0.9	0.9	1	0.8	0.5	0.4
Fri	0.8	0.8	0.8	0.8	1	0.5	0.4
Sat	0.5	0.5	0.5	0.5	0.5	1	0.8
Sun	0.4	0.4	0.4	0.4	0.4	0.8	1

Table 6.8: Day of week similarity

Time of day similarity

In weekdays the difference between night and morning can be considered quite vast. The traffic in the morning rush can be as much as 10 times the number of counts in the night on a regular day, maybe even more. On Saturday the traffic counts for morning and night are very similar, but on Sundays the night counts are typically the double of what it is in the morning. This is one example concerning the night time, but the general difference between the rest of the time periods, can in most cases be concerned high.

The similarity between features have been calculated manually by first looking at traffic data, finding an average value of the traffic at a given day and time, and then calculating using the following formula:

$$sim = a_1 * w_1 + a_2 * w_2 + a_3 * w_3$$

Where
a_1 is the average traffic at given time in the weekdays.
a_2 is the average traffic at given time on Saturdays.
a_3 is the average traffic at given time on Sundays.

Weights are based on the number of days in a week the given number represents.
w_1 Monday to Friday, which give a weight of 5/7.
w_2 Saturday, which give a weight of 1/7.
w_3 Sunday, which give a weight of 1/7.

Feature	Night	Morning	Noon	Afternoon	Evening
Night	1	0.28	0.18	0.2	0.27
Morning	0.28	1	0.67	0.42	0.74
Noon	0.18	0.67	1	0.62	0.62
Afternoon	0.2	0.42	0.52	1	0.42
Evening	0.27	0.74	0.62	0.42	1

Table 6.9: Time of day similarity

Weather situation similarity

We assume that the different weather conditions are quite similar to each other. Some aspects to consider are that the weather may affect the amount of people driving to work, and also the traffic conditions.

Feature	Sunny	Cloudy	Precipitation	Pouring
Sunny	1	0.95	0.9	0.8
Cloudy	0.95	1	0.95	0.9
Precipitation	0.9	0.95	1	0.9
Pouring	0.8	0.9	0.9	1

Table 6.10: Weather situation similarity

Temperature similarity

When considering temperatures, freezing will be the most different from the others. This is when a typically wet road, can turn into a wet and icy road, which may greatly affect the driving conditions. Combined with the weather condition, the temperature will also affect how many people that decide to bicycle to work.

Feature	Freezing	Medium	Hot
Freezing	1	0.75	0.7
Medium	0.75	1	0.95
Hot	0.7	0.95	1

Table 6.11: Temperature similarity

Friction similarity

The similarity between the frictions can be seen in the Table 6.5, where the friction coefficient determines the similarity.

Feature	Dry	Wet	Icy
Dry	1	0.9	0.3
Wet	0.9	1	0.4
Icy	0.3	0.4	1

Table 6.12: Friction similarity

Event

We consider the events to be so different that we decided to use a similarity measurement that give equal events a similarity of 1, and non equal events a similarity of 0.

6.3.5 Evolutionary algorithm for weight setting

In order to assign proper weights to the different features used, we decided to use a genetic algorithm as described in Section 3.4.

We decided to use a binary string of the size 49 bits for our genome. For each of our 7 features we therefore have 7 bits, which can give 128 different values. Since our weights need a value between 0 and 1, we calculated the value of each of the bit-strings and divided by 127. The evolutionary operators we have decided to use are crossover and mutation.

In order to calculate the fitness of each genome, we evaluated the case base using cross-validation. Each genome is applied as weights to the case base, and then leave-one-out cross validation is executed. The average performance using the weights is calculated by summing the performance of the case base for each removed case, and then dividing by the total amount of cases. This will then return a fitness score between 0 and 1, where 1 is a 100 percent similarity.

The selection of individuals are done by using roulette wheel selection, as described in Section 3.4. We have also decided to use elitism, which ensures that the x most fittest individuals are taken to the next generation. This ensures that the population does not lose its fittest individuals, which can save us some time as these do not need to be rediscovered if they would not get selected for the next generation.

Feature	Weights
Season	0.008
Day of week	0.213
Time of day	0.984
Weather	0.039
Temperature	0.732
Road friction	0.008
Event	0.291

Table 6.13: Feature weights given by the GA

It is important to note that even though the evolutionary algorithm have proposed weights that give good results when evaluating the case-base, these weights may not give as good results when applied to the test set. This is because the weights are calculated to give optimal results given the known cases, and if new unknown cases appear they might not correspond to the estimated weights. Albeit, we assume that these weights are close to optimal considering the known cases, which we have used for training. In any case, using these weights will give a much better performance than if they were to be set manually, even by an expert.

6.3.6 Weakness concerning similarity measuring

The methods that we have used in order to evaluate similarity of the features have known weaknesses. This is because the similarity values between features are in reality relative to what values each of the features and their corresponding weights have; they are contextually dependent. An example can be when the feature “Day of week” is a Monday, compared to a Sunday.

When it is a Monday, the feature “Time of day” will have little significance when considering the traffic flow in the morning and at noon. On a Sunday, the same feature will a great significance on the traffic flow, when considering morning and noon. The weight of the “Time of day” feature, should therefore be higher when the “Day of week” feature is a Sunday, as the traffic changes more greatly than it does during a Monday.

For example, a Monday morning may have a traffic flow of 800 cars in one direction, and the same flow of 800 may be the case at noon. On a Sunday morning, the traffic flow may be 150 cars, while it can be 500 at noon. This can be seen in Table A.1. As discussed, this is not properly reflected during the similarity measurements, as the feature weights are always the same, and not relative to the feature values.

Some research has been conducted in order to improve similarity measurement. PATDEX[25] targets the problem of having contextual dependent weights. The system has local weights on its feature values in addition to weights on feature names/classes by using similarity measurements on pairs of values. Park et al.[23] propose MBNR (Memory-Based Neural Reasoning), case-based reasoning with local feature weighting by a neural network, which provide case-specific weights to the learning process.

Optimally, every combination of features should have its own weights in order to get the most accurate similarity measurement. However, we decided to go with the methods as described earlier in this chapter. The reason is because of the good results gained when evaluating the case base using these simple measurements. Also, the gain versus the time that would be spent on improving the similarity measurement, was not something that we considered to be worth it.

6.4 Retrieval

When a new problem description is given to the system, the system needs to retrieve the most similar case in order to find the best solution. This process consists of first calculating the similarity of the new case with the other cases in the case base. Weights and similarity functions, as described in section 6.3, are used in order to determine the similarity between the features, and then the total similarity of the cases. The total similarity is calculated by finding the total average similarity by summing all the n similarities and dividing by n.

The case which has the highest similarity is then chosen as the solution.

6.5 Reasoning towards a signal plan

In the first draft of the case base, we originally planned on having a calculated signal plan as the solution for each case. After a while we realised that it would be more beneficial if we did not associate a case description with a signal plan, but instead with the traffic flow. This would give us the opportunity to incorporate more domain knowledge into the process since we can alter how the calculations of the signal plans are done. The traffic flow is always a concrete number, representing the number of cars passing by. However, a signal plan can have many different values, which are a result of how our calculations are done. Therefore, we have chosen to separate the reasoning process from the rest of the system, so that this process can be changed and possibly improved without affecting the rest of the system.

The following subsections describe how we have decided to use the available information in order to reason towards new signal plans.

6.5.1 Calculating a base plan using Webster's formula

In order to estimate a basic signal plan, Webster's formula is used to calculate the length of the cycle, and phase times. Afterwards, the value of the other features, such as weather, weekday, and also general domain knowledge, will be taken into account when deciding how much the signal plan cycle should differ from the result given by Webster's formula.

Since our cycle consists of three phases the red-time, L, would normally be estimated to 9 seconds for each cycle, as described in Section 3.1.3. However, this is in our situation not the case. The total red time in a cycle varies based on which static signal plan is used, and therefore our system is forced to have the same red time dependent on the given plan in the given time period. For demonstration purposes, we say that L = 24, as is the case in Timetable 1, shown in Figure 6.8

Since there are no turnings in the intersection, the capacity, s_i, of the intersection, is assumed to be 1800-(3*100)=1500, as shown in Section 3.1.3.

Example

Following is an example where the traffic volume for one hour is:

- Traffic from east to west is 400 (phase 1).
- Traffic from west to east is 300 (phase 1).
- Traffic from north to south is 1000 (phase 2).
- Traffic from south to north is 800 (phase 2).

- Traffic from south to east is 100 (phase 3).
- Traffic from north to west is 50 (phase 3).

$$Y = y_{\text{max(phase 1)}} + y_{\text{max(phase 2)}} + y_{\text{max(phase 3)}} = 400/1800 + 1000/1800 + 100/1800 \approx 0.83$$

This will give the following calculation for an optimal cycle time:

$$C_o = \frac{1.5 * 24 + 5}{1 - 0.83} \approx 241.17$$

Rounding 241.17 to the nearest 5 gives 240. Now the length of each phase can be calculated:

$$g_{phase(1)} = \frac{0.22}{0.83}(240 - 24) \approx 58 \text{ seconds}$$
$$g_{phase(2)} = \frac{0.56}{0.83}(240 - 24) \approx 144 \text{ seconds}$$
$$g_{phase(3)} = \frac{0.056}{0.83}(240 - 24) \approx 14 \text{ seconds}$$

The total cycle will then contain a total green time of 58+144+14 = 216, which then gives room for 24 seconds of red time in a cycle of 240 seconds.

6.5.2 Applying knowledge to improve the base-plan

As described in the previous section, we have presented our approach in order to calculate a basic signal plan. This basic plan gives a good guideline on how the final plan, but does not incorporate domain specific knowledge.

One thing we have chosen to apply to the base-plan is to do some adjustments when the calculated plan ends up with a too low priority on phase 2. When the traffic flow is very low, the formula can end up giving a less than a 2 second priority for this signal group. In some cases during our simulation, and probably in real life, this is actually too little time to get cars across the stop-line, resulting in a long queue forming. The manual for planning and maintenance of traffic light intersections[37] discourage any green time of less than 3 seconds, which we have incorporated into our calculations. This has been done by ensuring that when the traffic flow is less than 50 cars an hour, we give a minimum phase time of 3 seconds. When there are more than 50 cars, we give a minimum of 4 seconds, which in most cases lets at least two cars pass before the lights change.

When there is a football match, we have also decided to do some adjustments. If the expected traffic is more than 100 cars, and if the arrival model is "as soon as possible" (ASAP), described in Section 7.2.2, the delay will be very high if the basic plan is to be applied. Therefore, phase 2 will get some extra time, based on the amount of predicted traffic, and similarly decrease the length of phase 1 with the same amount, so that the total cycle time does not change.

We did not find the need to alter the plans more than has already been mentioned, even for the slippery roads. The performance using the calculated signal plans can be seen in the simulation results, described in the Evaluation chapter.

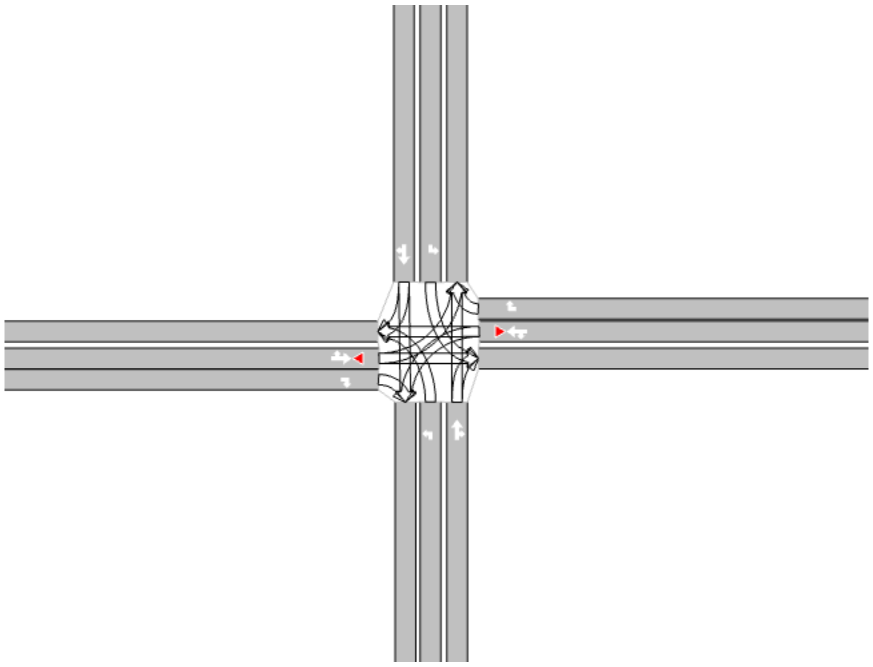


Figure 6.4: Intersection model

6.6 Modeling the intersection and traffic

The intersection, traffic, and also driver behavior, need to be modeled in a way that reflects the real world to a highest degree possible, as this greatly affects the correctness of the simulation results. Our approach is shown below.

6.6.1 Modeling the intersection

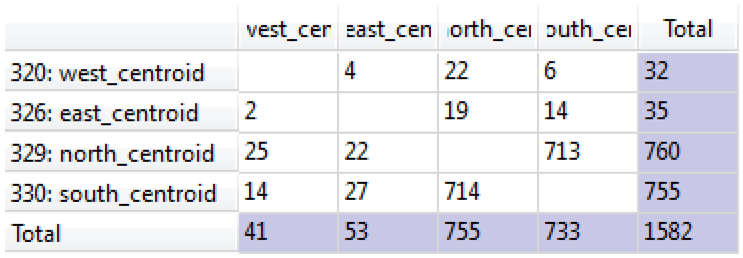
The model of the intersection can be seen in Figure 6.4. The lane going from top towards the bottom, corresponds to the lane going from the city center (north), heading out from the city (south). When comparing this to the illustration in Figure 2.1, it can be noticed that there is one lane missing in both directions, in the north and south going lanes. This is because these lanes are only for public transport and other priority vehicles. Since this group do not get any priority in the static plans against which we are evaluating our system, we have decided to not use these lanes in our implementation, as these would not affect the signal plans anyway.

Every possible turning in the real intersection is implemented. The two small red triangles mean that cars coming from these lanes have to give way if a conflict should occur.

Pedestrians are not explicitly considered in this model. In the signal plans they have green light at the same time as the cars in phase 1, but in phase 3 they typically gets a green light before the cars, see phase plan in Figure 6.6, and signal plans in Figure 6.7. In order to implicitly consider the pedestrians, we have chosen to give red lights for all cars when the pedestrians are supposed to have green by themselves. This is implemented in all signal plans made by the system in order to not give advantages to the CBR-system.

6.6.2 O/D matrix

In order to determine the volume of traffic going in a given direction, matrices that represent origin and destination (O/D matrices) are used. Figure 6.5 shows an example of how such a matrix can look.



	vest_cer	east_cen	orth_cer	outh_cer	Total
320: west_centroid		4	22	6	32
326: east_centroid	2		19	14	35
329: north_centroid	25	22		713	760
330: south_centroid	14	27	714		755
Total	41	53	755	733	1582

Figure 6.5: O/D matrix

When changing the traffic during different simulations, the O/D matrix is where these demands are adjusted.

6.6.3 How to simulate icy roads in the model

As there are no parameters for adjusting the pavement friction in order to simulate icy roads, we had to consult with the Aimsun support. They suggested that we could rather decrease the acceleration and decelarion of the vehicles, as this would to somewhat degree simulate slippery roads. By adjusting the turning speed, this will also simulate low friction, as people typically drive slower in the turnings when the roads are slippery.

6.7 Static signal plans

This section presents the static plans which have been used in the intersection, and illustrates how they are applied to our system.

6.7.1 Phases

A total of three different phases are used in this intersection. These are shown in Figure 6.6. The numbers represent a signal group, and all the signal groups in each phase can get a green light without seriously conflicting with each other. Signal group 1 (SG1) and 2 are the lanes going towards and out of the city. SG5 and SG6 needs to cross a heavy trafficked ongoing lane in order to reach their destination, which is why they have gotten their own phase. SG7,8,9 and 10 are signal groups for pedestrians. The pedestrians have priority when they have green light at the same time as other groups, and cars will therefore give way if a conflict occurs.

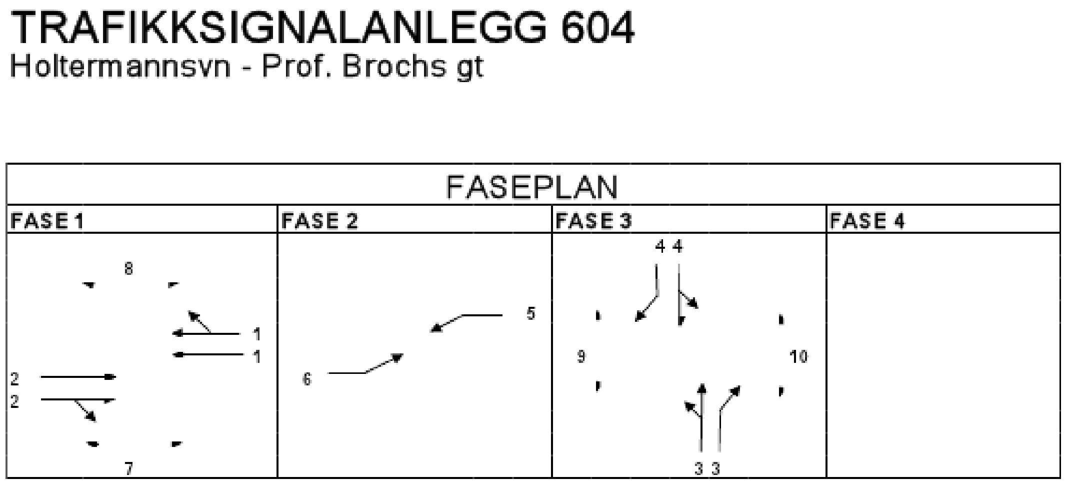
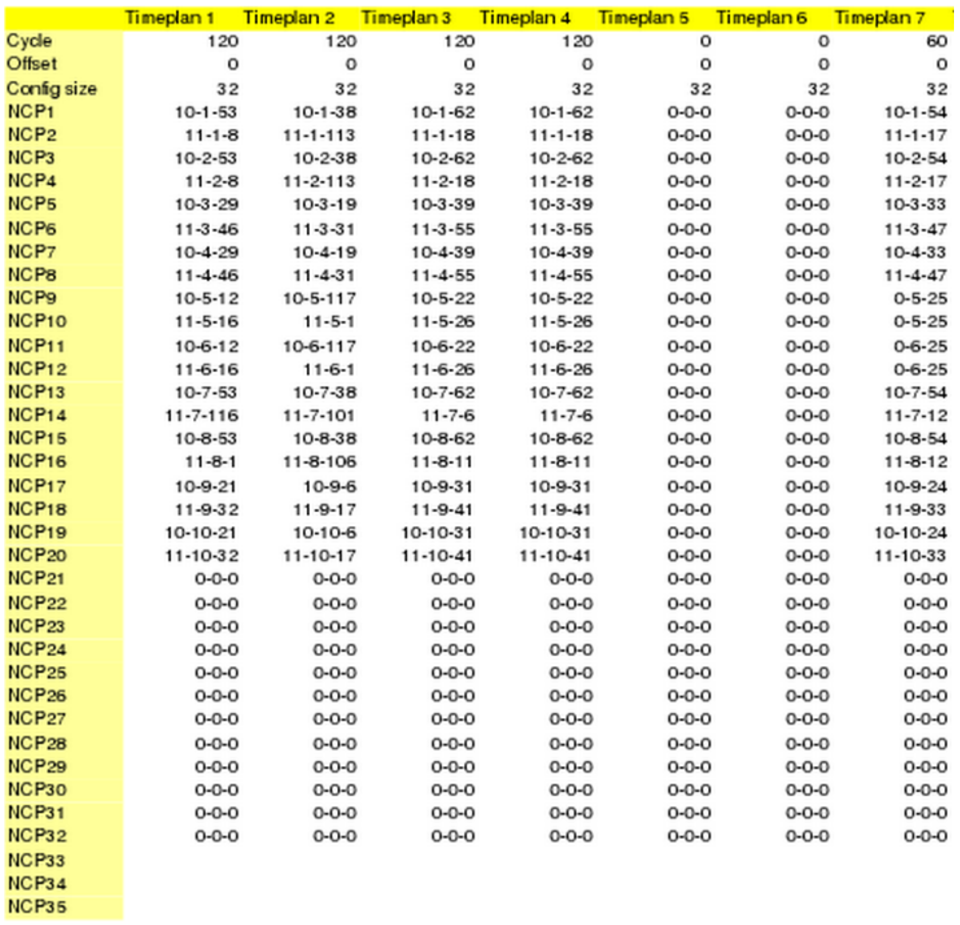


Figure 6.6: Phase plan

6.7.2 Signal plans

The static signal plans used for the intersection are given in Figure 6.7. A total of four different plans are in use, Timeplan 1, Timeplan 2, Timeplan 3 and Timeplan 7. The three numbers given in each NCP row indicates whether the signal groups turns red or green, which signal group this applies to, and the start/stop time of the action. 10-1-53 indicates that signal group 1 is to give a green light at 53 seconds within the cycle, while 11-1-8 indicates that the group is to give a red light at 8 seconds within the cycle. As the cycle is 120 seconds this means that the signal group have a total green time of 75 seconds. It is important to note that the use of static signal plans such as these has not been done for the last couple of years, according to sources at the NPRA. It should also be noted that each of the signal plans have a cycle of either 60 or 120 seconds. This is a result of the need to be able to correspond with other nearby intersections, which are also running cycles of 60 or 120 seconds. This limitation will also be applied to the CBR-system during evaluation.

Note that Timeplan 7 is different from the others. When looking at the time given for signal group 5 and 6, it says 0 seconds in the signal plan. This is because this phase does not actually exist in this plan. The assumption is that when this plan is applied, the traffic is not that high, so the cars that normally would need their own phase in order to cross the road, instead are able to do it in phase 1, when the cars are normally going just straight north or south. If the north going or south going traffic turns out to be dense, some of the turning cars will be able to wait within the intersection until this traffic gets a red light, which give it enough time to finish the turning. This can of course in worst case limit the turning traffic to just a couple of cars being able to turn for each green light, but if this is a problem, it is likely a good idea to change to a signal plan using 3 phases.



	Timeplan 1	Timeplan 2	Timeplan 3	Timeplan 4	Timeplan 5	Timeplan 6	Timeplan 7
Cycle	120	120	120	120	0	0	60
Offset	0	0	0	0	0	0	0
Config size	32	32	32	32	32	32	32
NCP1	10-1-53	10-1-38	10-1-62	10-1-62	0-0-0	0-0-0	10-1-54
NCP2	11-1-8	11-1-113	11-1-18	11-1-18	0-0-0	0-0-0	11-1-17
NCP3	10-2-53	10-2-38	10-2-62	10-2-62	0-0-0	0-0-0	10-2-54
NCP4	11-2-8	11-2-113	11-2-18	11-2-18	0-0-0	0-0-0	11-2-17
NCP5	10-3-29	10-3-19	10-3-39	10-3-39	0-0-0	0-0-0	10-3-33
NCP6	11-3-46	11-3-31	11-3-55	11-3-55	0-0-0	0-0-0	11-3-47
NCP7	10-4-29	10-4-19	10-4-39	10-4-39	0-0-0	0-0-0	10-4-33
NCP8	11-4-46	11-4-31	11-4-55	11-4-55	0-0-0	0-0-0	11-4-47
NCP9	10-5-12	10-5-117	10-5-22	10-5-22	0-0-0	0-0-0	0-5-25
NCP10	11-5-16	11-5-1	11-5-26	11-5-26	0-0-0	0-0-0	0-5-25
NCP11	10-6-12	10-6-117	10-6-22	10-6-22	0-0-0	0-0-0	0-6-25
NCP12	11-6-16	11-6-1	11-6-26	11-6-26	0-0-0	0-0-0	0-6-25
NCP13	10-7-53	10-7-38	10-7-62	10-7-62	0-0-0	0-0-0	10-7-54
NCP14	11-7-116	11-7-101	11-7-6	11-7-6	0-0-0	0-0-0	11-7-12
NCP15	10-8-53	10-8-38	10-8-62	10-8-62	0-0-0	0-0-0	10-8-54
NCP16	11-8-1	11-8-106	11-8-11	11-8-11	0-0-0	0-0-0	11-8-12
NCP17	10-9-21	10-9-6	10-9-31	10-9-31	0-0-0	0-0-0	10-9-24
NCP18	11-9-32	11-9-17	11-9-41	11-9-41	0-0-0	0-0-0	11-9-33
NCP19	10-10-21	10-10-6	10-10-31	10-10-31	0-0-0	0-0-0	10-10-24
NCP20	11-10-32	11-10-17	11-10-41	11-10-41	0-0-0	0-0-0	11-10-33
NCP21	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP22	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP23	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP24	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP25	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP26	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP27	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP28	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP29	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP30	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP31	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP32	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0
NCP33							
NCP34							
NCP35							

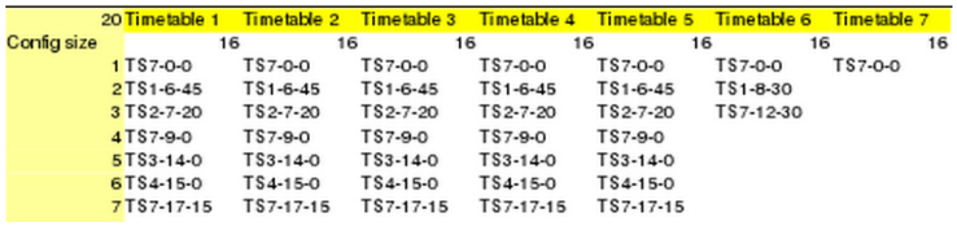
Figure 6.7: Signal Plans.

6.7.3 Schedule

A schedule for when the different signal plans should be applied is given in Figure 6.8. Each column represent one of the seven days in the week, starting on a Monday. Each of the rows contain 3 numbers. The first one indicates the signal plan plan to be utilised, and the second and third number indicates the time in which the plan is to be applied. TS7-0-0 says that the signal plan 7 should be in use from 00.00, while TS1-6-45 indicates that signal plan 1 should be applied from 06.45.

6.7.4 Modifying the signal plans for the system

As pedestrians are not explicitly simulated in the traffic simulator, we chose to take them into account by applying time with red lights for all the car lanes, in which the pedestrians would have time to cross the roads. In the static signal plans the pedestrians often have green time at the same time as the cars, which therefore does not affect the model. However, the signal groups 9 and 10, which is only pedestrian groups, do have some green time in which the car traffic does not. We have therefore chosen to reflect this by giving red light to all



	20	Timetable 1	Timetable 2	Timetable 3	Timetable 4	Timetable 5	Timetable 6	Timetable 7
Config size		16	16	16	16	16	16	16
	1	TS7-0-0	TS7-0-0	TS7-0-0	TS7-0-0	TS7-0-0	TS7-0-0	TS7-0-0
	2	TS1-6-45	TS1-6-45	TS1-6-45	TS1-6-45	TS1-6-45	TS1-8-30	
	3	TS2-7-20	TS2-7-20	TS2-7-20	TS2-7-20	TS2-7-20	TS7-12-30	
	4	TS7-9-0	TS7-9-0	TS7-9-0	TS7-9-0	TS7-9-0		
	5	TS3-14-0	TS3-14-0	TS3-14-0	TS3-14-0	TS3-14-0		
	6	TS4-15-0	TS4-15-0	TS4-15-0	TS4-15-0	TS4-15-0		
	7	TS7-17-15	TS7-17-15	TS7-17-15	TS7-17-15	TS7-17-15		

Figure 6.8: Time table.

Timetable	SG 1&2,I,SG 3&4,I(SG 9&10),SG 5&6,I
1	75,4,4,13,17,7
2	75,4,4,18,12,7
3	76,4,4,13,16,7
4	76,4,4,13,16,7
7	23,16,14,7

Table 6.14: Static signal plans for the system.

cars in this time period.

This red time will also need to be forced into the signal plans created by the system. If this is not done, it would give a huge advantage to the system as it would normally give green light to cars in this time period.

The static signal plans are edited in order to give a huge intermission when only the pedestrians have a green light (SG9 & SG10). These plans are given in Table 6.14, where I(SG9&10) shows the intermission time when pedestrians are allowed to cross.

6.8 Implementation details

This section describes how the different systems work in more detail, in order to get a better understanding of how the different part works.

6.8.1 CBR-system

The following sections present the methods that needs to be overridden in jCOLIBRI in order to create a CBR-system. All these methods have been altered, which is described in the following sections.

Configure

The configure method sets up and configures the system in order to make it ready for performing the CBR-cycle. A database connection is made, and then a SQL-script is executed. This script contains the database tables, and all the known cases of the system. All the cases consist of a problem description and a problem solution. A TCP-server is also created in order to be able to receive problems, and send signal plans to the simulator manager.

Pre-cycle

During this cycle, all the cases stored in the database are loaded into the memory, and ready for use. This is also where the TCP-server is ordered to listen for incoming messages from the simulator manager.

Cycle

The first message is always a problem, which the CBR-system is to solve. So, when the system receive the first message, it adapts it into a problem description that is used in order to retrieve similar cases. The retrieved cases have solutions that are an estimate for the traffic flow.

By using the most similar case, as elaborated in Section 6.4, the estimated traffic flow, along with the other features will be used for reasoning towards a signal plan that best suits the given scenario. The reasoning process is described in Section 6.5.

After a signal plan has been calculated, this is sent back to the simulator manager.

Post-cycle

Shuts down the system.

6.8.2 Simulator manager

The simulator manager (SM) is initiated when a script is run in order to both start the Aimsun simulator, and load a given model. The simulator manager then gains access to the current used model, which can be manipulated using the Aimsun scripting API. When all is set, simulations will be run, and the results will be stored in the database.

Simulation cycle

The SM runs x number of cycles which contains multiple simulation cycle objects. In our case, this object will either be a CBR cycle or a static signal plan cycle. A given case will be used for the simulation scenario. The static plan cycle will identify the day and time of the scenario, and then apply the signal plan given in its schedule. For the CBR cycle, the object will obtain a signal plan by sending a message containing the case information using its TCP client to the CBR system. It will then apply the received plan, and run a simulation.

For each scenario, the SM will generate a random seed which is used in Aimsun in order to get random car arrivals. This seed is the same for all different cycle objects so that the traffic will be exactly the same during the simulations.

Manipulating the model

This section describes what values we have pre-set and which values are changed while running the simulation cycles.

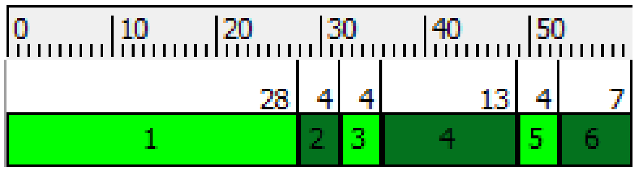


Figure 6.9: Aimsun signal plan

We have decided to set some of the values manually. This is because the Aimsun user interface sometimes is more efficient to use, as some of them only need to be set once. These values are more elaborated in the Evaluation chapter in Section 7.2.2.

Each of the cycle objects will for every simulation manipulate the model. The following values are set before a simulation is run:

- The O/D matrix, described in Section 6.6.2, is set using the traffic flow data from the current case, which the CBR-system tries to predict based on the case description.
- The signal plans in the model need to be manipulated in order to simulate the given traffic demand with the proposed signal plan. Figure 6.9 depicts an Aimsun signal plan.

Chapter 7

Evaluation

The results gained during the research are presented in this chapter. First, the evaluation of the case-base is described in the following section. Simulations are done as described in Section 2.5.2, while the simulation results and other relevant information is presented in Section 7.2. Finally, these results are summarised in Table 7.46, and discussed in Section 7.3.

7.1 Case base evaluation

Cross validation of the case base has been done, as described in Section 2.5.1, in order to get an estimate of how the case base may perform. Throughout the development, the case base size have changed, and Table 7.1 shows the results of size vs accuracy when doing cross evaluation. The case base ended up consisting of 55 cases, and the overall performance of this case base can be seen in Figure 7.1.

Number of cases	**Accuracy**
55	88%
50	87%
45	85%
40	84%
35	84%
30	84%
25	82%
20	78%
15	76%

Table 7.1: Cross evaluation using different amount of cases

Initially, we added cases describing the normal traffic patterns on weekdays and weekends. We also added some that included football matches and other events that may cause irregular patterns in the traffic. When we had about 30 cases, we added two more days describing a weekday and a weekend. This is probably why the accuracy did not increase, as the cases added was quite similar to other cases in the case base. The next cases we added, was 3 days describing the more special events in order to have more than one case describing football matches etc. If there is only one case that covers a given problem, and it is used as test case, there will not be any other case which solves the given problem in a satisfying manner. At 55 cases, we felt that most of the scenarios had multiple cases covering them, and by adding more similar cases, it would not increase the accuracy significantly. This is probably what happened when the number of cases was between 30 and 40.

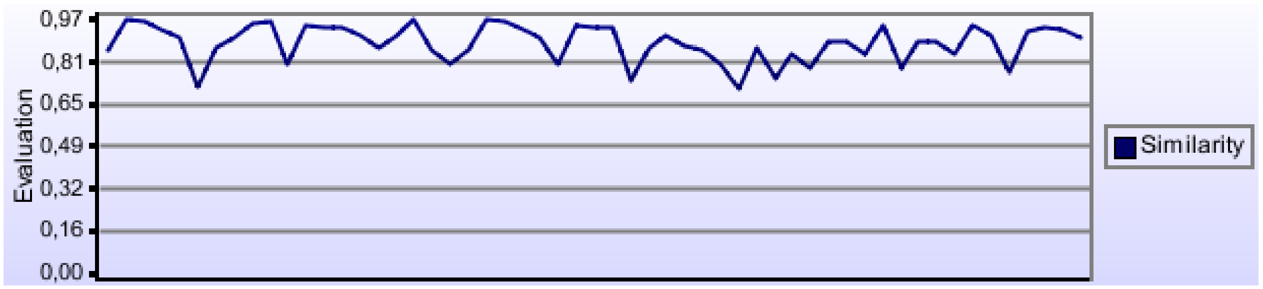


Figure 7.1: Cross evaluation using 55 cases

7.2 Simulation results

Subsequent sections present some simulation details, while the rest present the different simulation scenarios with results.

7.2.1 Simulation assumptions

Every simulation was done using a static plan, a restricted CBR plan, and a CBR plan. A restricted CBR plan means that the CBR system can only make plans which have total cycle time of 60 or 120 seconds, as that is a limitation in the cycle time of the static plans, described in Section 6.7.2. We also decided to do simulations where the CBR system is not limited to a given cycle time, to see if that gave any remarkable differences.

Each scenario was run three times using different seeds for when the vehicles arrived. This gave us different traffic patterns, and therefore ensured that the performance of the different signal plans gave similar results despite different patterns.

7.2.2 Simulation parameters

This section describes the parameters that have been altered before running simulations in Aimsun.

Simulation step

This value needs only to be set once. It is used to determine the time interval in which the next state is calculated. We adjusted this from 1 second to the lowest possible value, 0.1, as this would not greatly affect the simulation time. The problem of using high values for the steps during the simulations, is that if a car has a reaction time of .5 seconds, the model will make this 1 second as that is when the new state is calculated.

Arrival time

Traffic can be generated by multiple samples of different distributions. We have in all scenarios, except the ones stating otherwise, used samples gained from a truncated normal distribution.

The "as soon as possible" (ASAP) generation model have been used in some scenarios, in which the vehicles enters the network as soon as possible.

All these generation models are more described in the Aimsun MicroMeso user manual[34].

Vehicle parameters

There are a lot of parameters that can be adjusted for each vehicle type. We have chosen to use the ones that are default in Aimsun for normal driving conditions. Table 7.2 lists some of these parameters, which in some cases are adjusted in order to simulate slippery roads.

Name	**Mean**	**Min**	**Max**
Max acceleration	3	3	3
Normal deceleration	4	3.5	4.5
Max deceleration	6	5	7

Table 7.2: Acceleration and deceleration parameters. The units are m/s^2.

Slippery roads

As mentioned in Section 6.6.3, the Aimsun support suggested that we could adjust the acceleration and deceleration in order to simulate driving conditions on a low friction surface. Therefore, we decided to multiply the parameters given in Table 7.2 with the friction coefficient of the pavement, given in Table 6.3.3.

Turning percentage

The turning percentage of the traffic is important to incorporate into the model. We were able to obtain this with some help of the NPRA, and these numbers can be seen in the Tables 7.3, 7.4 and 7.5. The turning percentage shown in Table 7.6 will be used for all the simulations, since we were not able to obtain more of this data. It is assumed that the turning percentage does not change dramatically in the weekends, and that this will give a good approximation.

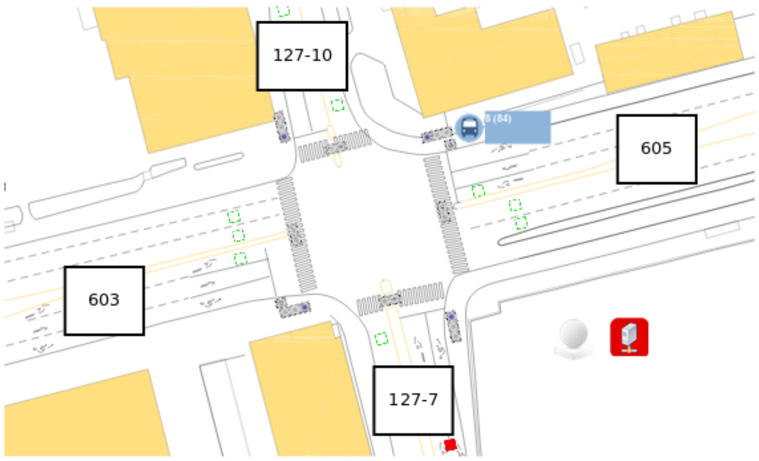


Figure 7.2: The turning tables use the lane identification shown in this figure.

From → to	**Turn percentage**
127-7 → 603	70%
127-7 → 605	20%
127-7 → 127-10	10%
127-10 → 603	55%
127-10 → 605	40%
127-10 → 127-7	5%

Table 7.3: Turn percentage used for a weekday.

7.2.3 Simulation descriptions

The first sections, 7.2.4 and 7.2.5, represents ordinary days where the traffic numbers does not deviate from most other days in a significant manner. As the weekend and week are typically quite different concerning the traffic demand, we have chosen to simulate a Monday and a Saturday focusing on five different parts of the day. These parts are:

- Night at 0.00-1.00, when the traffic typically is quite low in weekdays, but may be relative high in the weekends.
- Morning at 7.00-8.00, the time of the day when most people go to work towards downtown in weekdays. This is typically a time period with low traffic in weekends.
- Noon at 12.00-13.00, when the traffic flow is relative low while waiting for the afternoon rush.
- Afternoon at 15.00-16.00, the time of the day when most people are leaving downtown in order to go home after work hours.
- Evening at 21.00-22.00, which is typically a time with relative low traffic.

From → to	**Turn percentage**
127-7 → 603	60%
127-7 → 605	38%
127-7 → 127-10	2%
127-10 → 603	48%
127-10 → 605	49%
127-10 → 127-7	3%

Table 7.4: Turn percentage used for a Saturday.

From → to	**Turn percentage**
127-7 → 603	60%
127-7 → 605	38%
127-7 → 127-10	2%
127-10 → 603	45%
127-10 → 605	53%
127-10 → 127-7	3%

Table 7.5: Turn percentage used for a Sunday.

Some of the following scenarios will differ from the regular traffic patterns. This includes events such as football matches, slippery roads, and holiday. The different columns in the simulation and evaluation tables are described in Section 2.5. All the simulation runs from which we have calculated the average values, can be seen in Section A.2.

The simulation tables, containing detailed description for all the runs, are located in the Appendix A.2, while the evaluation tables are presented in each of the following scenarios.

7.2.4 Scenario 1 - A regular weekday

This first scenario presents a regular weekday in Trondheim, at five different times of the day.

From → to	**Turn percentage**
603 → 605	96%
603 → 127-10	3%
603 → 127-7	1%
605 → 603	96%
605 → 127-10	2%
605 → 127-7	2%

Table 7.6: Turn percentage for cars turning from 603 and 605, used for all days.

Night

Attribute	**Value**
Date	17.09.12
Weekday	Monday
Time of day	00.00-01.00
Weather	Cloudy
Temperature	9,8
Friction	Dry
Traffic flow	59,57,2,2,4,3

Table 7.7: Scenario 1 - Night description

As can be seen in Table 7.7, the traffic flow is very low, especially from the east and west lanes. This means that as little time as possible should be given to these lanes, probably so low that just 1-2 cars are able to slip through when given a green light.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	65.6	140.7	35.0	74.3
Restricted CBR average	55.0	129.3	37.2	63.0
CBR average	56.1	132.6	34.1	66.2

Table 7.8: Scenario 1 - Night evaluation table

Table 7.8 shows the average of the three different simulations. It should be noted that the restricted CBR have the best results, even though it needs to scale its signal plan to last for exactly 60 seconds, just as the static plan. The restricted CBR has in this simulation a stopping time that is 16% shorter, a travel time that is 8% shorter, 6% higher speed and 11% less delay than the results gained using the static plan.

Morning at 07.00-08.00

Mornings are a lot more demanding than the night. This is also one of the two time periods we have simulated, in which the static plan uses three phases, and have a 120 seconds cycle time. In contrast to the night, these plans should prioritise the lanes from east and west, as their flow is relatively high. But still, the traffic going in the north and south directions will get the highest priorities. The traffic at this time of the day is primarily headed towards downtown as people are going to work.

Attribute	**Value**
Date	17.09.12
Weekday	Monday
Time of day	07.00-08.00
Weather	Precipitation
Temperature	7,1
Friction	Dry
Traffic flow	612,854,53,34,190,140

Table 7.9: Scenario 1 - Morning description

In contrast to the night simulations, the difference between the static and the CBR plans are higher. However, now the unrestricted CBR plans have the overall best results. Stopping time is 25% less for the CBR plan, travel time is 12% less, the speed is 4% slower, while the delay is 21% less.

The most notable about these results is that, while the unrestricted CBR plans have better results than the static plans, the average speed is actually lower. The reason for this is because of the length of the unrestricted cycles are typically much shorter, in this case 65 seconds, which lowers the average speed as cars often need to stop. Still, they give generally better results, since the potential waiting time is relative short. When comparing to the restricted CBR, the average speed on this is higher than both the static and unrestricted one, as the restricted cycles last for 120 seconds, with longer phases.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	73.2	151.1	37.1	84.2
Restricted CBR average	58.0	135.4	38.2	68.5
CBR average	54.5	133.6	35.8	66.7

Table 7.10: Scenario 1 - Morning evaluation table

Noon at 12.00-13.00

Noon is the time of the day when the traffic typically have a similar, but relative high demand, in both south going an north going directions. This time of the day consists of two phases in the static plans.

Attribute	Value
Date	17.09.12
Weekday	Monday
Time of day	12.00-13.00
Weather	Cloudy
Temperature	9,0
Friction	Wet
Traffic flow	733,755,35,32,53,41

Table 7.11: Scenario 1 - Noon description

Here, the unrestricted CBR has the best plan, marginally better than the restricted one. Compared to the static plan, the stop time is 44% shorter, the travel time is 25% less, 22% higher speed and 44% less delay. These results are the best gained through the simulation process yet.

The main reasons for the high improvement, is mainly because the static plans have 14 seconds of green time for the west and east going lanes in a total of 60 seconds per cycle. The dynamic plan have a total cycle of 71 seconds, with a three second time for the west and east going lanes. Considering the relative low demands, it is quite advantageous to minimise the priority of these lanes.

Control type	Stop time	Travel time	Speed	Delay
Static average	75.5	154.4	31.6	88.2
Restricted CBR average	43.0	118.3	39.0	52.1
CBR average	41.2	116.0	40.4	49.8

Table 7.12: Scenario 1 - Noon evaluation table

Afternoon at 15.00-16.00

Afternoon is the time of the day when most people are leaving the downtown area in order to get home from work. This gives a higher demand to the south going lane. A signal plan consisting of three phases is used at this interval.

Attribute	Value
Date	17.09.12
Weekday	Monday
Time of day	15.00-16.00
Weather	Cloudy
Temperature	10,4
Friction	Dry
Traffic flow	1153,690,115,30,130,84,39

Table 7.13: Scenario 1 - Afternoon description

As in the previous simulations, the unrestricted CBR plan has a marginally better performance than the restricted one. Compared to the static plans, the stop time is 14% less, the travel time is 7% shorter, the speed is 4% higher, and the delay is 13% less. Here, the variety in the performance was not as great as in the previous simulation.

In this simulation the unrestricted signal plans last for 117 seconds, which is quite close to the static one. Also, the difference in the lengths of the phases is not so great. But, as seen in earlier simulations, the static plans still give too high priority for the east and west going lanes, which results in a loss of performance .

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	76.5	151.8	37.9	85.4
Restricted CBR average	67.0	141.3	39.5	75.0
CBR average	66.1	140.8	39.3	74.3

Table 7.14: Scenario 1 - Afternoon evaluation table

Evening at 21.00-22.00

The evening is the last simulated time interval of the first two scenarios. A plan consisting of two phases are used, and compared to the other periods simulated, except for the night time, the traffic is relatively low.

Attribute	**Value**
Date	17.09.12
Weekday	Monday
Time of day	15.00-16.00
Weather	Cloudy
Temperature	7,3
Friction	Dry
Traffic flow	408,366,30,22,37,12

Table 7.15: Scenario 1 - Evening description

The restricted CBR plans, gave the best results during these simulation runs. Compared to the static plans, the stopping time is 33% less, travel time is reduced by 17%, average speed is 16% higher, and lastly, the delay is 27% less.

The fact that the restricted CBR gave better results than the unrestricted one, shows that improvements can be made when calculating the unrestricted plan, as it theoretically never should be outperformed by the restricted one.

Control type	Stop time	Travel time	Speed	Delay
Static average	71.4	148.4	33.2	82.2
Restricted CBR average	47.9	122.6	38.4	56.4
CBR average	50.0	126.0	36.6	59.8

Table 7.16: Scenario 1 - Evening evaluation table

7.2.5 Scenario 2 - A regular Saturday

This scenario is similar to the previous one, in that this is just a regular day concerning the traffic. The difference is that this is a Saturday, which, along with Sunday, is quite different compared to the rest of the week.

Night

The nighttime in the weekends differ from the other days as there are a lot more traffic, typically at least doubling the amount. However, the demands are not high enough to actually affect the flow in a considerable manner.

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	00.00-01.00
Weather	Cloudy
Temperature	6,3
Friction	Dry
Traffic flow	189,156,7,8,9,6

Table 7.17: Scenario 2 - Night description

Here, both the CBR plans are quite close to each other, but the restricted one gives a marginally better result. Compared to the static plans the average stop time is 27% shorter, the travel time is decreased by 14%, the speed is 12% higher, and the delay is 25% less.

Still, the length of phase two limits the performance of the static plan, giving such high differences in performance.

Control type	Stop time	Travel time	Speed	Delay
Static average	70.4	146.5	33.8	80.3
Restricted CBR average	51.7	126.2	37.8	60.0
CBR average	51.6	133.5	34.9	61.3

Table 7.18: Scenario 2 - Night evaluation table

Morning at 07.00-08.00

The morning demands in the weekend are completely different than on the ordinary week days, even less than the demands at night time on Saturday. Here, a plan consisting of two phases are used.

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	07.00-08.00
Weather	Cloudy
Temperature	3,9
Friction	Dry
Traffic flow	150,147,53,1,2,12

Table 7.19: Scenario 2 - Morning description

Similar to the previous simulation period, the restricted CBR performs slightly better than the unrestricted one. Compared to the static plans, the stop time is reduced by 11%, the travel time is 4% shorter, the speed is 6% higher, and the delay is reduced by 11%.

The low traffic is the main cause of the small differences in the results. The restricted CBR has the same cycle length as the static one, but as earlier mentioned, the second phase gets too high priority in the static plans.

Control type	Stop time	Travel time	Speed	Delay
Static average	71.8	144.8	33.4	81.8
Restricted CBR average	63.9	139.7	35.4	73.1
CBR average	62.9	140.3	32.6	73.8

Table 7.20: Scenario 2 - Morning evaluation table

Noon at 12.00-13.00

Noon on a Saturday has a similar amount of traffic as noon on a regular week day. Also, the static 2 phase plan is used at this time.

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	12.00-13.00
Weather	Sunny
Temperature	7,5
Friction	Dry
Traffic flow	522,727,35,10,36,12

Table 7.21: Scenario 2 - Noon description

The restricted CBR plan did outperform both the unrestricted CBR and the static plans. Compared to the static plans, the stop time is 62% shorter, the average travel time is reduced by 34%, the speed is 35% higher, and the delay is reduced by 59%.

Again, the high priority of phase 2 in the static plan ruins its performance. The reason why the unrestricted CBR does not perform better than the restricted one, is likely because of the short cycle time it uses.

Control type	Stop time	Travel time	Speed	Delay
Static average	74.7	153.0	32.0	86.6
Restricted CBR average	28.7	101.7	43.1	35.6
CBR average	30.5	104.2	42.0	38.1

Table 7.22: Scenario 2 - Noon evaluation table, 2 phases

As the two phase static plan did not perform as well, we decided to see if one of the static three phase plans could achieve a better result, which is then compared to the three phase CBR-plans.

The three phase static plan did actually perform better than the two phase static plan. Both the three phase plan of the restricted and the unrestricted one, performed worse than their two phase counterparts, but still better than the static plan. In this case, the unrestricted plan did also outperform the restricted plan. Compared to the static plan, the unrestricted CBR have reduced the stop time by 37%, lessened the travel time by 17%, the speed is reduced by 1%, and the delay is 32% less.

As seen in one earlier scenario, the static plan have higher average speed than the unrestricted CBR. This is as mentioned before, caused by the shorter cycle time in the plan produced by the unrestricted CBR.

Control type	Stop time	Travel time	Speed	Delay
Static average	69.7	143.4	39.2	77.3
Restricted CBR average	49.6	123.7	39.7	57.7
CBR average	43.8	118.8	38.9	52.7

Table 7.23: Scenario 2 - Noon evaluation table

Afternoon at 15.00-16.00

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	15.00-16.00
Weather	Sunny
Temperature	9,0
Friction	Dry
Traffic flow	782,509,37,18,50,11

Table 7.24: Scenario 2 - Afternoon description

As in the week, the afternoon has a traffic flow going mainly from the city center. Because of the lower traffic number compared to the week days, the two phase plan is used at this time in the weekends. By comparing the static plan to the best performing plan, in this case the restricted CBR, we get similar results as we got when comparing the noon runs, albeit not as high difference. The stop time is 49% lower, the travel time is shortened by 28%, the speed is 39% higher, and the delay is reduced by 50%.

Control type	Stop time	Travel time	Speed	Delay
Static average	74.1	151.8	32.1	85.9
Restricted CBR average	37.8	109.2	44.6	43.3
CBR average	39.2	112.3	41.8	46.4

Table 7.25: Scenario 2 - Afternoon evaluation table

Evening at 21.00-22.00

The evening is quite similar to the night time considering the traffic demand. There are more traffic, and the east and west going roads do not require a high priority. The two phase plan is used.

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	21.00-22.00
Weather	Cloudy
Temperature	6,8
Friction	Dry
Traffic flow	267,282,14,6,22,4

Table 7.26: Scenario 2 - Evening description

The restricted CBR plan has the best simulation results. Compared to the static plan, the stop time is reduced by 32%, we get a 16% decrease in travel time, a 13% higher speed, and a reduction of the delay by 30%. These results are very similar to the ones gained from the simulations of the evening on a week day, even though the traffic is about 30% lower in the weekend.

Control type	Stop time	Travel time	Speed	Delay
Static average	67.3	143.3	34.2	77.2
Restricted CBR average	45.8	120.1	38.8	54.0
CBR average	50.2	126.1	36.2	60.1

Table 7.27: Scenario 2 - Evening evaluation table

7.2.6 Scenario 3 - Game day

This and the following scenarios present unusual traffic patterns, compared to a regular day. This scenario is on a Thursday and a Sunday evening, when a football match has just finished playing, and all the spectators are going home. As the traffic patterns in this scenario are not regular, we did try two different ways of distributing the traffic. Typically the traffic is high in the lane from east for about 15-20 minutes after the game has finished, while the rest of the hour, the traffic is back to normal.

Therefore the normal distribution may not give a realistic simulation, as all cars will be distributed along the simulated hour. We decided to try to use the ASAP (as soon as possible) arrival model on the east most lane. This is on the other hand is maybe too extreme, with an instant high demand, but it may give a more realistic picture of the situation. Simulations using both the normal and the ASAP model are run in this scenario.

Football match on a weekday

Attribute	Value
Date	04.10.12
Weekday	Thursday
Time of day	21.00-22.00
Weather	Sunny
Temperature	6,6
Friction	Dry
Traffic flow	662,589,138,65,58,16

Table 7.28: Scenario 3 - Football match on a weekday

The results using the normal distribution model are similar to what we have seen in the other scenarios. By comparing the static results to the restricted CBR, we see that the stop time is 24% shorter, the travel time is reduced by 13%, average speed is increased by 13% and the delay is 23% less. But as mentioned earlier, the high pressure caused by the football match is not properly reflected as the traffic is distributed over the entire hour of simulation.

Control type	Stop time	Travel time	Speed	Delay
Static average	76.7	154.9	31.9	88.3
Restricted CBR average	58.5	134.8	36.2	68.2
CBR average	59.5	136.3	35.2	69.7

Table 7.29: Football match on a weekday - Evaluation table

By using the ASAP model, as depicted in Figure 7.3, we got some unexpected results. The static plan is now the best, tied with the restricted CBR. The signal plans need to both be able to get the traffic from the east lane away, while still give priority to the south and north going traffic when the pressure from east is gone. As the static plan notoriously has given a too high priority to the second phase, this is actually giving good results in this case.

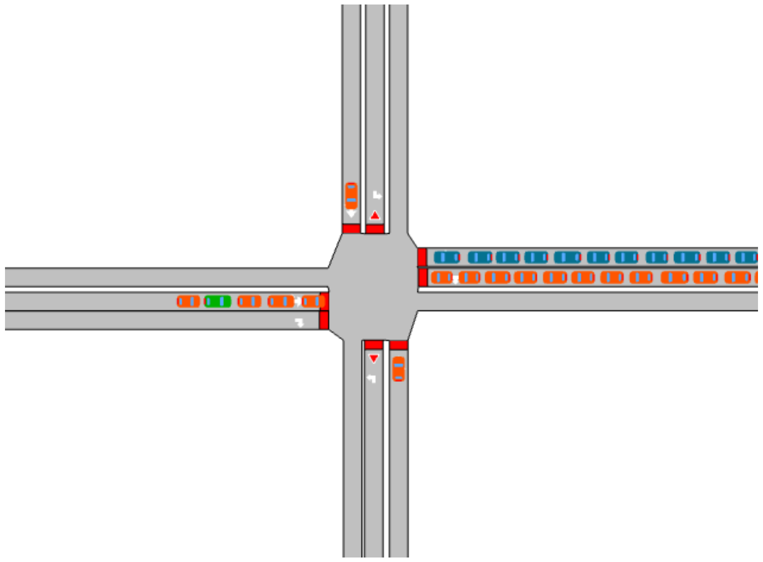


Figure 7.3: Simulating the traffic after a football match using ASAP arrival on the east lane.

Control type	Stop time	Travel time	Speed	Delay
Static average	101.6	182.9	30.0	116.3
Restricted CBR average	101.6	182.9	30.0	116.3
CBR average	103.9	185.8	28.9	119.2

Table 7.30: Football match on a weekday, ASAP - Evaluation table

Football match on a Sunday

This simulation targets the aftermath of when a football match has been played on a Sunday. The traffic count is typically lower than on a weekday, so we wanted to see if the results would differ significantly.

Attribute	**Value**
Date	16.09.12
Weekday	Sunday
Time of day	21.00-22.00
Weather	Cloudy
Temperature	10,0
Friction	Dry
Traffic flow	514,562,137,59,48,10

Table 7.31: Scenario 3 - Football match on a Sunday

First, the simulation using the normal distributed traffic have been run. The results are as expected, the restricted CBR gives the best results. Compared to the static plan, the stop time is 23% shorter, the travel time is reduced by 13%, speed is increased by 14%, and the delay is shortened by 22%.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	76.1	154.2	32.0	87.5
Restricted CBR average	58.7	134.7	36.5	68.1
CBR average	61.3	137.9	35.6	71.2

Table 7.32: Football match on a Sunday - Evaluation table

By running simulations using the ASAP arrival model, the static plan outperforms both CBR plans. Compared to the restricted CBR, the static plan has a 15% less stop time, a 8% shorter travel time, a 7% slower speed, and a 12% shorter delay. These results will be discussed more thoroughly in Section 7.3.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	100.9	182.1	30.0	115.5
Restricted CBR average	117.9	198.1	32.3	131.4
CBR average	120.5	201.0	31.7	134.3

Table 7.33: Football match on a Sunday, ASAP - Evaluation table

7.2.7 Scenario 4 - Holiday time

When there is a holiday, the traffic is naturally affected. We have decided to simulate traffic on the 25. of December, as this is a day where the traffic is assumed to be different from a regular day.

Morning

Attribute	Value
Date	25.12.11
Weekday	Sunday
Time of day	07.00-08.00
Weather	Precipitation
Temperature	3,0
Friction	Wet
Traffic flow	142,131,2,0,7,7

Table 7.34: Scenario 4 - Holiday on Sunday morning

Unfortunately, the only traffic data we could acquire for this date, was from 2011. This day is a Sunday, and the traffic may not vary as much as it could have, compared to an ordinary Sunday. The restricted CBR has the best result. Compared to the static plan, the stop time is reduced by 49%, travel time is lowered by 26%, speed is increased by 22%, and the delay is reduced by 48%. These results are quite similar to the ones gained by the simulations done on a Saturday at the same time.

Control type	Stop time	Travel time	Speed	Delay
Static average	66.1	141.6	34.6	75.7
Restricted CBR average	33.6	105.2	42.2	39.1
CBR average	34.9	107.9	41.3	41.7

Table 7.35: Holiday on Sunday morning - Evaluation table

Holiday on Sunday afternoon

The afternoon has actually a higher traffic than a typical Sunday afternoon, more similar to a Saturday.

Attribute	Value
Date	25.12.11
Weekday	Sunday
Time of day	15.00-16.00
Weather	Precipitation
Temperature	3
Friction	Wet
Traffic flow	838,500,43,45,84,25

Table 7.36: Scenario 4 - Holiday on Sunday afternoon

The unrestricted CBR outperforms both the other plans. Compared to the static plan, the stop time is 53% shorter, a 28% reduction on the travel time, a 23% higher average speed, and a 50% shorter delay time.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	80.8	155.4	31.4	89.0
Restricted CBR average	45.6	121.2	38.5	54.7
CBR average	38.2	111.3	42.7	44.9

Table 7.37: Holiday on Sunday afternoon - Evaluation table

7.2.8 Scenario 5 - Slippery roads

In order to simulate slippery roads, we took scenarios which we already had simulated, and changed the friction and temperature to “Icy” and “Freezing”. This way we can also compare our results to the same scenarios, to see the possible impact of slippery roads.

Noon

Noon has shown to be an interesting time to simulate due to the relatively high traffic counts combined with the usage of the two phase plans. We simulate both scenario 1 and scenario 2 at this hour, with slippery roads.

Attribute	**Value**
Date	17.09.12
Weekday	Monday
Time of day	12.00-13.00
Weather	Cloudy
Temperature	-1,4
Friction	Icy
Traffic flow	733,755,34,32,53,41

Table 7.38: Scenario 1, noon with slippery roads - Description table

Unlike the same scenario with dry roads, the restricted CBR beats the unrestricted on the average stop time, but not the others. Therefore we have decided to compare the static plans with the unrestricted CBR. Stop time is reduced by 39%, travel time is shortened by 26%, speed is increased by 46%, and the delay is 39% less. These result are somewhat similar to the ones gained without slippery roads, except for the average speed, which was doubled compared to the increase gained in scenario 1.

Control type	Stop time	Travel time	Speed	Delay
Static average	58.4	186.2	26.2	116.0
Restricted CBR average	30.5	140.6	33.9	74.4
CBR average	35.9	137.3	38.3	71.1

Table 7.39: Scenario 1, noon with slippery roads - Evaluation table

We also wanted to see the results gained by simulating slippery roads at the same time of the day, only on a Saturday.

Attribute	Value
Date	22.09.12
Weekday	Saturday
Time of day	12.00-13.00
Weather	Sunny
Temperature	-1,4
Friction	Icy
Traffic flow	522,727,35,10,36,12

Table 7.40: Scenario 2, noon with slippery roads - Description table

On this run, the results are actually opposite from the previous one. The restricted CBR performs better, except for in the stop time, where the unrestricted has the best results. Compared to the static plans, the stop time is reduced by 19%, travel time by 20%, speed is increased by 44%, and the delay is lessened by 32%.

These results are not as diverse as the ones gained both in the previous noon run, and in the corresponding scenario 2 run, without the slippery roads.

Control type	Stop time	Travel time	Speed	Delay
Static average	55.0	177.1	27.1	111.1
Restricted CBR average	44.3	141.6	38.9	75.6
CBR average	33.8	151.1	31.2	85.0

Table 7.41: Scenario 2, noon with slippery roads - Evaluation table

Afternoon

The plan applied for the afternoon run, uses three phases for scenario 1, and only 2 phases on a Saturday. We decided to simulate both scenarios modified with slippery roads, expecting interesting results.

Scenario 1 has a high traffic demand at this time of the day, while scenario 2 has a comparably lower traffic count. When looking at the results of these runs without the slippery roads, scenario 1 does not have as diverse results as we have seen in many of the simulations. Barely 14% decrease in stop time, just 13% less delay, and so on. In scenario 2, these numbers are much higher, up to 50%. It would therefore be interesting to see these scenarios modified with slippery roads.

Attribute	**Value**
Date	17.09.12
Weekday	Monday
Time of day	15.00-16.00
Weather	Cloudy
Temperature	-1,4
Friction	Icy
Traffic flow	1153,690,115,30,130,84,39

Table 7.42: Scenario 1, afternoon with slippery roads - Description table

The restricted CBR has the best results, except when considering the average speed. Compared to the static plan, the stop time is 19% less, the travel time is decreased by 8%, speed is increased by 0%, and the delay is shortened by 13%. These result are marginally better than those gained from the same simulation using dry roads, except for the speed, which is about the same in this scenario.

Control type	**Stop time**	**Travel time**	**Speed**	**Delay**
Static average	83.6	186.1	34.6	119.5
Restricted CBR average	68.0	170.4	34.7	103.9
CBR average	73.0	173.8	35.3	107.3

Table 7.43: Scenario 1, afternoon with slippery roads - Evaluation table

Attribute	**Value**
Date	22.09.12
Weekday	Saturday
Time of day	15.00-16.00
Weather	Sunny
Temperature	-1,4
Friction	Icy
Traffic flow	782,509,37,18,50,11

Table 7.44: Scenario 2, afternoon with slippery roads - Description table

The results show that the restricted CBR performs marginally better than the unrestricted one. Compared to the static plan, the stop time is 38% shorter, there is a 29% decrease in travel time, a 50% increase in speed, and 46% less delay. By looking at the same scenario without slippery roads, the results are somewhat similar, at least when considering the discord between the static plan and the best CBR plan.

Control type	Stop time	Travel time	Speed	Delay
Static average	54.9	177.3	27.0	111.1
Restricted CBR average	34.3	126.1	40.6	60.0
CBR average	33.9	128.3	40.6	62.2

Table 7.45: Scenario 2, afternoon with slippery roads - Evaluation table

7.2.9 Summarising the results

Table 7.46 presents a summary of the simulation results. Listed are the performance of the best CBR-plan compared to the static plan. Negative numbers indicate that the given CBR-plan performed worse than the static plan on the given criterion.

Scenario	Best plan	Stop time	Travel time	Speed	Delay
Sc1 (Weekday) - Night	Restricted	16%	8%	6%	11%
Sc1 (Weekday) - Morning	Unrestricted	25%	12%	-4%	21%
Sc1 (Weekday) - Noon	Unrestricted	44%	25%	22%	44%
Sc1 (Weekday) - Afternoon	Unrestricted	14%	7%	4%	13%
Sc1 (Weekday) - Evening	Restricted	33%	17%	16%	27%
Sc2 (Weekend) - Night	Restricted	27%	14%	12%	25%
Sc2 (Weekend) - Morning	Restricted	11%	4%	6%	11%
Sc2 (Weekend) - Noon(2p)	Restricted	62%	34%	35%	59%
Sc2 (Weekend) - Noon(3p)	Restricted	37%	17%	-1%	32%
Sc2 (Weekend) - Afternoon	Restricted	49%	28%	39%	50%
Sc2 (Weekend) - Evening	Restricted	32%	16%	13%	30%
Sc3 (Football) - Evening	Restricted	24%	13%	13%	23%
Sc3 (Football ASAP) - Evening	Static/Restricted	0%	0%	0%	0%
Sc3 (Football) - Evening	Restricted	23%	13%	14%	22%
Sc3 (Football ASAP) - Evening	Static	-15%	-8%	-7%	-12%
Sc4 (Holiday) - Morning	Restricted	49%	26%	22%	48%
Sc4 (Holiday) - Afternoon	Unrestricted	53%	28%	23%	50%
Sc1 (Slippery roads) - Noon	Restricted	39%	26%	46%	39%
Sc2 (Slippery roads) - Noon	Restricted	19%	20%	44%	32%
Sc1 (Slippery roads) - Afternoon	Restricted	19%	8%	0%	13%
Sc2 (Slippery roads) - Afternoon	Restricted	38%	29%	50%	46%

Table 7.46: Summary of the results

7.3 Discussion

In this section we will discuss the results gained from the simulations, presented in the previous sections. Different scenarios were simulated in order to evaluate our system. Both strengths and weaknesses have appeared, and many of the results were not the ones we initially expected. Table 7.46 gives a quick reference by summarising the evaluation results.

Early in the simulation process, a clear weakness appeared for the static plans. In both the three phase plans, and the two phase one, the priority for the lanes going from east and west is almost always too high. This often results in green time being wasted on an amount of cars, which only needs a fraction of the time given. As a result of this, the static plans had a bad starting point, and the CBR-plans did in turn give unexpectedly good results. Also, the static plans may have been designed in order to correspond to the traffic from adjacent intersections, which also may give some limitations.

Especially the noon scenarios would get a big performance boost when applying the CBR-plans. The reason for this seems to be that because of the high traffic flow, the unrestricted CBR was able to make cycles that both lasted longer than 60 seconds, and had a maximum priority to the north and south going lanes. This turned out to be very beneficial under these conditions.

The fact that the CBR plans, which were restricted to a 60 or a 120 second cycle, did actually perform better than the unrestricted one in 14 of the total 19 runs, was a surprise to us. A reason for this would have to be that the restricted plan often has a longer cycle time than the one calculated for the unrestricted one, which turns out to give an advantage, as each cycle iteration gives more red time. The exception is the scenarios where the three phase plan is used. Here, the unrestricted plan have the better results in three of the four simulations. These plans have a cycle time which typically is between 60 and 120 seconds, in most cases less than the restricted CBR. As a result of these observations, the algorithm reasoning towards the signal plan, should be improved, especially for the two phase plan.

Even though the general results implies that the plans with the longer cycle times often perform better, there are some aspects of having shorter cycle time that is beneficial. When the waiting time is long, both motorists and pedestrians can be impatient. Especially pedestrians can start to jaywalk when they have waited for a long time, which in turn lowers the safety of the intersection. Therefore, the signal plan having the shorter cycle time should get some kind of benefit during the evaluation.

Another surprise that emerged during the simulations was that the static plans performed similarly, or even better, than the CBR-plans during the simulations of the traffic after a football match. This was the scenario when we deciede to use the ASAP arrival model. The high priority which was given to the east and west lanes would in this case turn out to be beneficial. However, the fact that the traffic is high demanding in the east lane for the first 15-20 minutes after the match, is not properly reflected. When using the ASAP-model, the traffic demand will initially be very high, and then when all the cars have left, the traffic going north and south should get maximum priority. As the whole simulation lasts for 1 hour, the plans will need to compromise heavily, resulting in a plan that only gives a decent performance. One thing that could have been done in this situation, is to apply one plan to the first 15 minutes, and then another one to the remaining 45 minutes.

Overall the simulations gave promising results. The system seems to be able to predict the traffic well by retrieving older traffic counts, that does not vary greatly from the current ones. The cross evaluation of the case base, which can be seen in Figure 7.1, shows that most of the cases do get a satisfying solution. Although the results were overall very good, it would have been interesting to evaluate against the plans which is calculated by the currently used system in Trondheim.

Since the restricted plan did perform very well on the two phase plans, compared to the unrestricted one, shows that a lot of improvement can be done to the signal plan calculator. This, and other ideas for future work, is presented in the next chapter.

Chapter 8

Conclusion and future work

This chapter presents some thoughts on possible future work, before concluding the research.

8.1 Conclusion

In this thesis, we have presented a prototype system that uses case-based reasoning to predict the traffic flow, which is then used to calculate signal plans for use in an urban intersection. Earlier situations containing an associated traffic flow as the case solution, is stored in the case base for future use. Situations are described by different features such as time, weather, and road surface friction. When a case is retrieved, a signal plan is calculated based on a combination of the predicted traffic flow, the case features, and general domain knowledge which is incorporated into the algorithm. The traffic count is obtained from a real world intersection, that in turn has been modeled in a traffic simulator, which is then used for simulations. Evaluation is done by comparing the simulation results of the CBR-system to static signal plans, which have been used in said intersection on earlier occasions.

The system has been implemented using Java and the jCOLIBRI framework, which gives a good foundation for a CBR system. Simulations have been run using Aimsun, which offers a Python scripting interface. A Python program was made that communicates with both the CBR-system and Aimsun, for the purpose of running simulations for evaluation.

This project has been done in cooperation with the Norwegian Public Road Administration. Increase of traffic efficiency is something that contributes towards many of their future goals, which has been the main motivation for our work.

Goals and research questions for this study, as presented in Section 1.1, have been to research whether CBR is suitable for the purpose of controlling traffic in an urban intersection, and if this system can perform better than existing methods in terms of stop time, travel time, average speed, and delay. We have also presented AI-methods that have been used in this domain, along with state of the art research.

The prototype system that has been implemented, has based on evaluation against static signal plans, shown promising results throughout a wide range of scenarios. The recurrent patterns that appear in traffic is therefore suitable for the methods that have been applied in the implementation. This, in combination with a general domain knowledge that is relevant for the current traffic state, and highly obtainable, gives a good foundation for calculating signal plans. By tailoring signal plans to upcoming situations, the urban intersection was much more adaptable, and performed overall better than by the use of static plans.

The prototype system that has been made during this research have demonstrated the possibility for usage of CBR in this domain. However, the system is far from finished, and there are a lot of possible improvements that should be made during future work.

8.2 Future Work

The prototype system made during this research can be improved and enhanced in multiple ways.

First and foremost, we want to point out the possible benefits of using a system which continually analyses the traffic and makes on-line decisions. Initially we planned on making such a system, but as some technical difficulties emerged, which is explained in Section 6.3.2, we decided to do a more off-line approach. Advantages of having such an on-line system compared to an off-line one, is obviously that such a system can adapt to the current traffic by changing the lengths of the next phases within a very short time. It may even give the possibility to skip phases if there are no cars waiting, saving a lot of red time. This would in some cases increase the traffic flow significantly. By doing this, the whole CBR-system should probably be altered, as it would probably not be necessary to predict the traffic. Because, in an on-line solution, the traffic will instead be fed to the system in real time. The traffic flow would then act as a feature instead, as we presented in our initial draft of the case-base, shown in Table 6.1.

As the traffic flow in an urban intersection is often dependent upon the neighboring intersections, it would be natural to enhance the system to control multiple adjacent intersections. We think that this should be done by having individual systems controlling each intersection. In an on-line solution they would communicate directly with each other in order to predict traffic flow, which would then be considered when modifying the phases. In an off-line solution this need to be cared for when calculating the signal plans. Therefore it would probably be beneficial with a global system that gets the predicted traffic flow from each of the intersections, and then acquires signal plans that together give the best solution. In any case, this is a very important enhancement, as adjacent intersections will have to correspond to each other in order to gain a good overall traffic flow.

Machine learning can be incorporated into the system by implementing the remaining steps in the CBR-cycle. This can contribute to finding better solutions to given situations, first by evaluating the outcome, and then retaining the ones that are good enough. If a solution already did exist, old case solutions can be exchanged with new and better ones. This will ensure that the system does not lose its performance in the future, which is often the case as the traffic patterns change over time. A lot of time and money is used for maintaining such traffic control systems. Machine learning may reduce both the costs of optimising and maintaining, by applying this to such a system.

For increasing the performance of the current system the process of retrieving cases may be improved. As mentioned in Section 6.3.6, the way we have chosen to retrieve our cases may be improved. However, the evaluation of our case base have given good results, as can be seen in Figure 7.1. Also, during the simulations, the system did retrieve good predictions for the traffic flow. The bottle neck is more likely to be how the signal plans are calculated, as described in Section 6.5, and discussed in the Evaluation chapter. As mentioned, the plans which are restricted to a given cycle time, often perform better than the ones that are not. This implies that actions should be done in order to improve these plans. Indications gotten from the evaluation, shows that the cycle time for the two phase plans should in many cases be longer, at least when considering our evaluation criterion, which brings us to the next point of improvement.

In urban infrastructure there are more actors than motorists, which requires different adaptions. Pedestrians need to be able to cross the roads in a safe way, usually at the same intersection as motorists. This has not been part of our research, although it has been implicitly incorporated into our signal plans, as described in Section 6.6. For a finished system, pedestrians should be considered, and also prioritised, when making the evaluation criterion, similar to the ones made for vehicles; measuring stop time, waiting time, and delay. This will probably favor shorter cycle times. There has been done research on the combination of pedestrians and urban intersections. Strom and Kheradmandi[17] show that green time given for the purpose of pedestrian crossing, may be postponed or shortened based on the type and amount of pedestrians, also considering the traffic flow. This could be combined with such a system as is presented in this research, in order to make good and adaptive calculations concerning the pedestrians.

Public transport is also something that should be incorporated into the system. In Trondheim, many of the bus routes are given a high priority, which ensures that the people using public transport get an efficient alternative to taking the car. As the buses often have their own lanes, they can move fast even though the rest of the traffic are stuck in queues. The buses are also considered when deciding the green time of lanes, focusing on letting the buses through. As buses may contain a lot of people, it is natural to give the bus a high priority. Therefore, buses will in the real world be of great importance when calculating phase and cycle lengths, and should be incorporated into the model of the system used for traffic control.

Measuring of pollution is also something that can be incorporated. This should probably be implemented after the public transport, pedestrians, and also vehicles used for transportation of goods, have been incorporated into the model. Goals of reducing emission from motorised vehicles can be reached by ensuring that larger vehicles does not accelerate and decelerate more than necessary.

Events that we have not considered in our evaluation might also be relevant to consider. Days when large groups of people are going to, or returning from holiday, may often cause heightened demand in the traffic. Days for public celebration, like 17. of May, does also break with the ordinary traffic patterns. Other events that might occur at unforeseen moments can be prepared for. Handling of accidents is something that might be relevant for a network of multiple intersections. This could include routing of the traffic in order to avoid the site of the accident, reducing chaos in the traffic.

Lastly, the system should be evaluated by comparing simulation results to more efficient methods and/or systems for controlling traffic. Initially we were planning to evaluate against the system which is now used for traffic control in Trondheim, but as described in Section 5.4.3, this was not applicable as we would not to be able to combine SPOT/UTOPIA with Aimsun by ourselves within a reasonable time frame. However if this connection was to be made, the evaluation results would have been very interesting as this system is currently being used.

Appendix A

Appendix

In this appendix, the traffic data used in this research is first presented, along with some graphs of traffic during different days. Section A.2 presents the data gained from the simulations, in which we gained the numbers used in our evaluation tables, shown in Section 7.2

A.1 Traffic data

We were able to obtain traffic data reflecting more special events. One of these events are taken from a Sunday when the local football team are playing one of their opponents, Viking, in the evening at 26 of August 2012. This causes quite different traffic demands compared to a regular Sunday. Another event, shown in Figure A.6, depicts when snowfall and slippery roads challenged the drivers on 19 of March 2012.

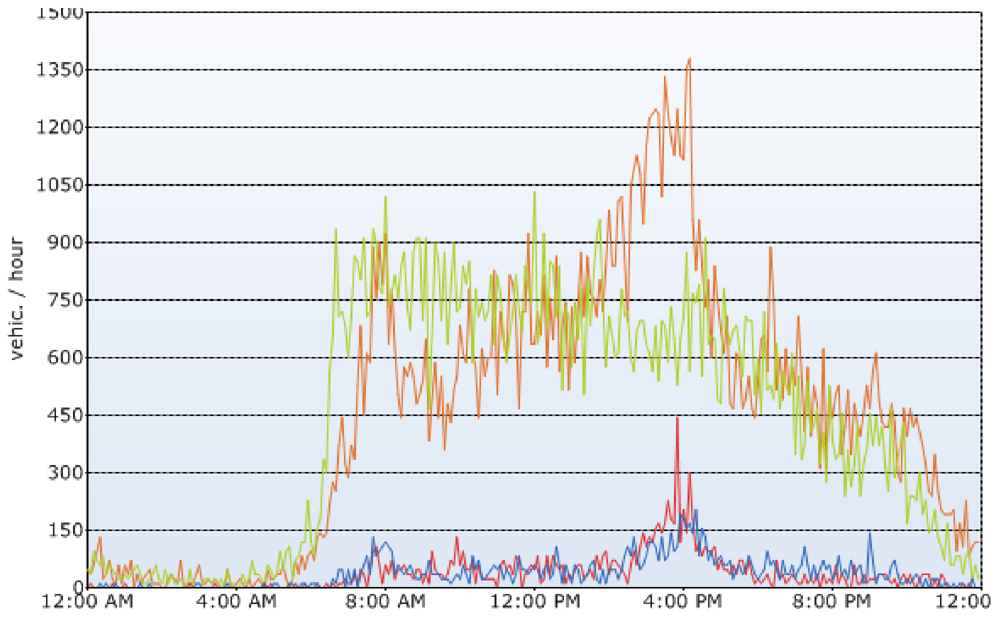


Figure A.1: Traffic volume on a regular Monday

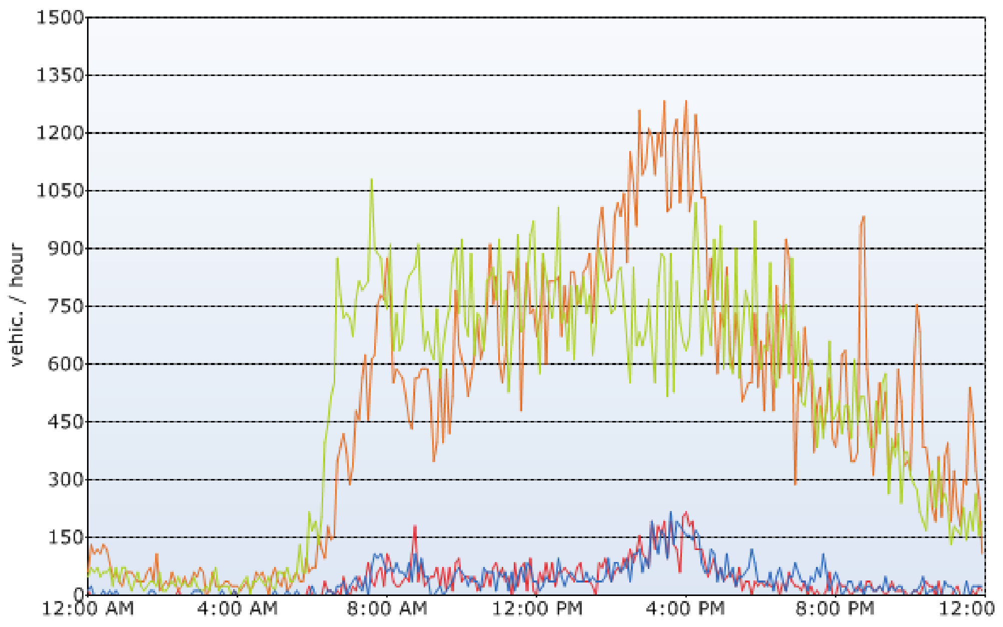


Figure A.2: Traffic volume on a regular Friday

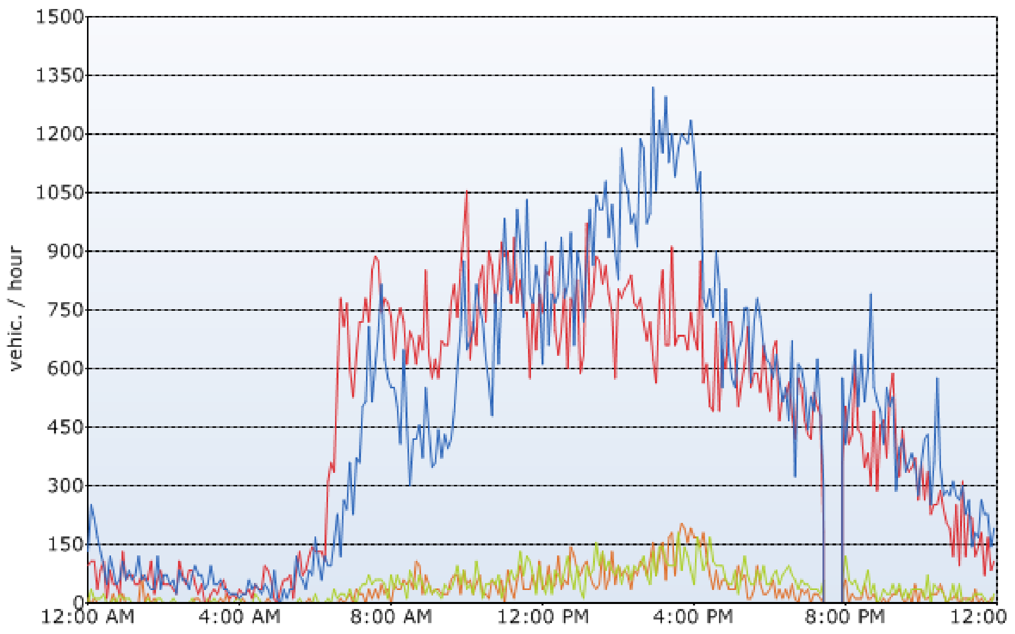


Figure A.3: Traffic volume on Thursday 22 of December 2011, when people are going for holiday

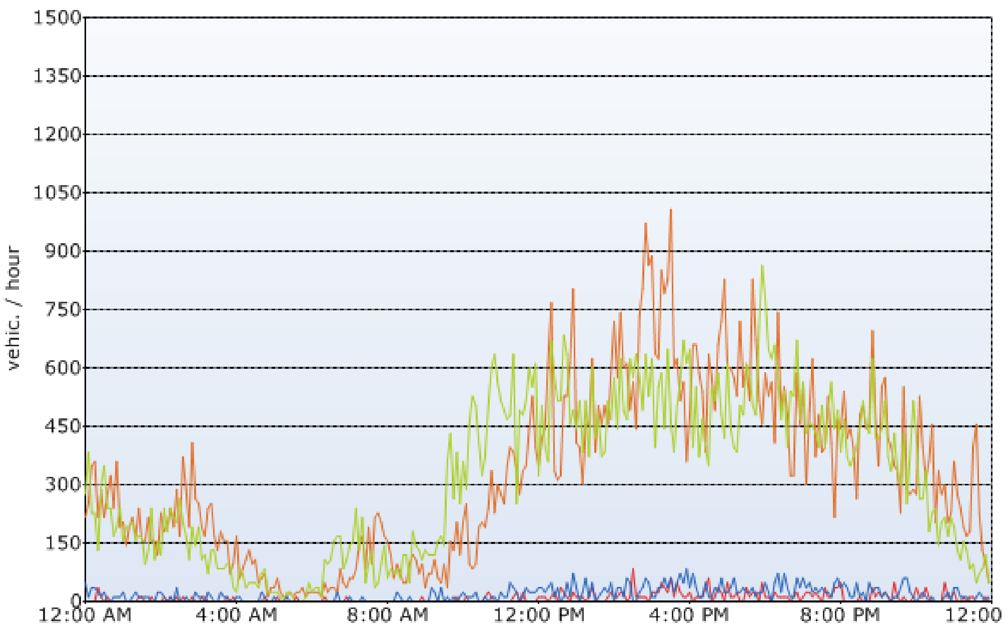


Figure A.4: Traffic volume on a regular Sunday

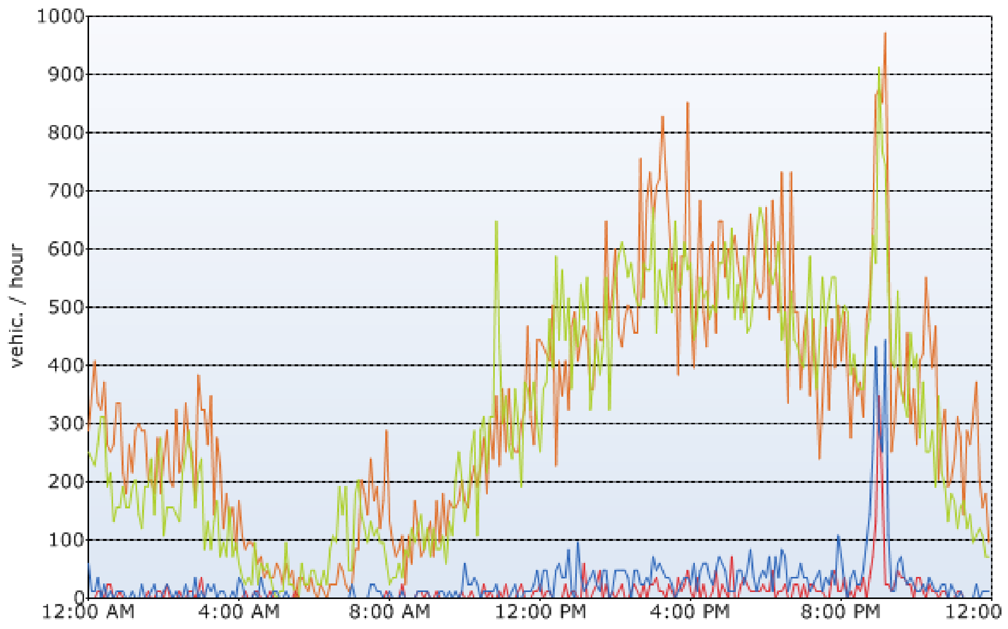


Figure A.5: Traffic volume on a Sunday when Rosenborg, the local football team, plays one of their opponents

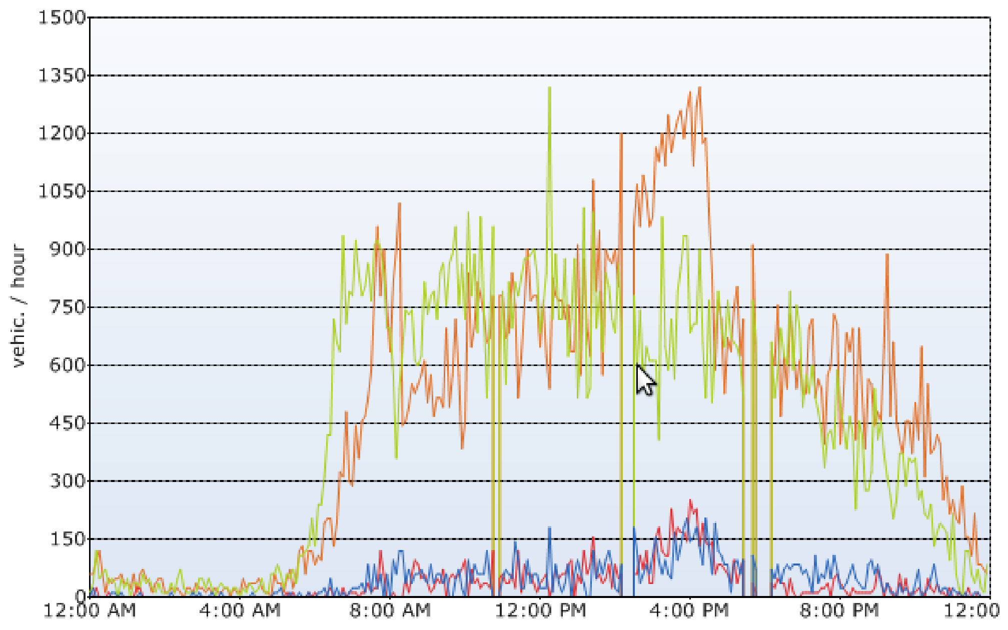


Figure A.6: Traffic volume on Monday 19 of March 2012, when the roads were quite slippery

Date	**Day**	**Time of day**	**Weather**	**°C**	**Friction**	**Special event**	**Traffic flow (NS, NS, E, W, NE, SW)**
10.09.12	Mon	00.00-01.00	Precipitation	2,9	Dry	None	(70,57,2,7,3,2)
10.09.12	Mon	07.00-08.00	Precipitation	3,8	Wet	None	(581,851,60,29,88,78)
10.09.12	Mon	12.00-01.00	Pouring	7,6	Wet	None	(706,773,43,50,39,24)
10.09.12	Mon	15.00-16.00	Precipitation	7,6	Wet	None	(1132,664,112,152,52,26,)
10.09.12	Mon	21.00-22.00	Precipitation	8,3	Wet	None	(394,373,31,11,24,4)
19.03.12	Mon	00.00-01.00	Cloudy	-2,5	Icy	None	(57,45,5,3,5,1)
19.03.12	Mon	07.00-08.00	Precipitation	-3,9	Icy	None	(608,836,37,27,79,85)
19.03.12	Mon	12.00-01.00	Precipitation	1,2	Icy	None	(706,812,48,50,36,27)
19.03.12	Mon	15.00-16.00	Sunny	1,2	Icy	None	(1178,721,117,145,31,23)
19.03.12	Mon	21.00-22.00	Sunny	1,6	Icy	None	(510,334,23,26,30,9)
07.09.12	Fri	00.00-01.00	Pouring	6,7	Wet	None	(86,58,6,1,4,1)
07.09.12	Fri	07.00-08.00	Cloudy	7,1	Dry	None	(562,826,55,37,87,78)
07.09.12	Fri	12.00-13.00	Cloudy	7,5	Dry	None	(763,768,42,52,32,29)
07.09.12	Fri	15.00-16.00	Precipitation	7,5	Wet	None	(1146,721,152,143,37,26)
07.09.12	Fri	21.00-22.00	Precipitation	7,6	Wet	None	(426,406,17,13,18,5)
16.07.12	Mon	00.00-01.00	Cloudy	8,9	Dry	Holiday	(76,78,5,1,5,1)
16.07.12	Mon	07.00-08.00	Cloudy	9,6	Dry	Holiday	(347,506,21,21,27,27)
16.07.12	Mon	12.00-01.00	Cloudy	12,3	Dry	Holiday	(630,774,42,29,32,17)
16.07.12	Mon	15.00-16.00	Cloudy	12,3	Dry	Holiday	(914,651,59,67,20,6)
16.07.12	Mon	21.00-22.00	Sunny	13,1	Dry	Holiday	(290,304,38,9,27,3)
23.07.12	Mon	00.00-01.00	Cloudy	11,7	Dry	Holiday	(74,48,3,1,3,1)
23.07.12	Mon	07.00-08.00	Cloudy	13,0	Dry	Holiday	(330,484,28,23,27,28)
23.07.12	Mon	12.00-01.00	Sunny	15.9	Dry	Holiday	(619,719,31,26,22,9)
23.07.12	Mon	15.00-16.00	Sunny	15,3	Dry	Holiday	(929,689,78,56,60,12)
23.07.12	Mon	21.00-22.00	Cloudy	13,6	Dry	Football	(492,520,177,58,20,4)
22.12.11	Mon	00.00-01.00	Cloudy	-3,7	Wet	Leaving	(122,74,12,6,6,1)
22.12.11	Mon	07.00-08.00	Sunny	-0,7	Wet	Leaving	(544,751,42,24,78,66)
22.12.11	Mon	12.00-01.00	Precipitation	-0,2	Wet	Leaving	(799,759,65,73,58,34)
22.12.11	Mon	15.00-16.00	Cloudy	1.6	Wet	Leaving	(1177,710,114,145,62,18)
22.12.11	Mon	21.00-22.00	Cloudy	0,5	Wet	Leaving	(400,400,31,19,28,9)
24.10.12	Thu	00.00-01.00	Cloudy	-4,7	Icy	None	(60,54,3,2,7,2)
24.10.12	Thu	07.00-08.00	Sunny	-0,7	Icy	None	(598,801,52,21,155,121)
24.10.12	Thu	12.00-13.00	Precipitation	-0,2	Icy	None	(645,679,43,44,63,39)
24.10.12	Thu	15.00-16.00	Cloudy	1,0	Icy	None	(990,665,92,134,68,39)
24.10.12	Thu	21.00-22.00	Cloudy	0,5	Icy	None	(564,473,0,33,38,15)
29.10.12	Mon	00.00-01.00	Cloudy	-7,5	Wet	None	(84,51,12,8,8,8)
29.10.12	Mon	07.00-08.00	Sunny	-9,9	Wet	None	(576,853,42,24,169,24)
29.10.12	Mon	12.00-01.00	Sunny	-4,7	Wet	None	(682,802,65,49,45,49)
29.10.12	Mon	15.00-16.00	Sunny	-3.0	Wet	None	(1129,720,114,144,86,144)
29.10.12	Mon	21.00-22.00	Cloudy	-4,3	Wet	None	(423,328,31,9,42,9)

Table A.1: Data acquired using Omnia

Date	**Day**	**Time of day**	**Weather**	**°C**	**Friction**	**Special event**	**Traffic flow (NS, NS, E, W, NE, SW)**
08.09.12	Sat	00.00-01.00	Precipitation	5,7	Wet	None	(184,189,9,4,9,2)
08.09.12	Sat	07.00-08.00	Precipitation	6,1	Wet	None	(177,137,6,11,3,5)
08.09.12	Sat	12.00-13.00	Sunny	7,8	Wet	None	(593,801,34,10,15,5)
08.09.12	Sat	15.00-16.00	Sunny	7,8	Dry	None	(844,630,42,17,35,7)
08.09.12	Sat	21.00-22.00	Sunny	7,4	Dry	None	(275,341,15,5,18,2)
09.09.12	Sun	00.00-01.00	Cloudy	2,9	Dry	None	(274,248,19,5,6,0)
09.09.12	Sun	07.00-08.00	Cloudy	3,8	Dry	None	(142,124,2,0,1,2)
09.09.12	Sun	12.00-13.00	Cloudy	7,6	Dry	None	(503,511,31,14,17,4)
09.09.12	Sun	15.00-16.00	Cloudy	7,6	Dry	None	(691,537,44,29,25,9)
09.09.12	Sun	21.00-22.00	Cloudy	8,3	Dry	None	(384,395,25,13,18,5)
26.08.12	Sun	00.00-01.00	Precipitation	10,8	Wet	None	(312,221,16,7,8,1)
26.08.12	Sun	07.00-08.00	Precipitation	8,8	Wet	None	(158,117,8,1,5,1)
26.08.12	Sun	12.00-13.00	Cloudy	11,2	Wet	None	(407,438,33,8,24,4)
26.08.12	Sun	15.00-16.00	Sunny	11,2	Dry	None	(653,559,36,20,31,7)
26.08.12	Sun	21.00-22.00	Sunny	10,1	Dry	Football	(509,512,114,66,20,7)

Table A.2: Data acquired using Omnia

Date	**Day**	**Time of**	**Weather**	**°C**	**Friction**	**Special event**	**Traffic flow (NS, NS, E, W, NE, SW)**
04.10.12	Thu	21.00-22.00	Sunny	6,6	Dry	Football	(662,589,138,65,58,16)
16.09.12	Sun	21.00-22.00	Cloudy	10,0	Dry	Football	(514,562,137,59,48,10)
25.12.11	Sun	07.00-08.00	Precipitation	3,0	Wet	Holi	(142,131,2,0,7,7)
25.12.11	Sun	15.00-16.00	Precipitation	3,0	Wet	Holi	(838,500,43,45,84,25)

Table A.3: Test data - miscellaneous

Date	**Day**	**Time of**	**Weather**	**°C**	**Friction**	**Special event**	**Traffic flow (NS, NS, E, W, NE, SW)**
17.09.12	Mon	00.00-01.00	Cloudy	9,8	Dry	None	(59,57,2,2,4,3)
17.09.12	Mon	07.00-08.00	Precipitation	7,1	Dry	None	(612,854,53,34,190,140)
17.09.12	Mon	12.00-13.00	Cloudy	9,0	Wet	None	(733,755,35,32,53,41)
17.09.12	Mon	15.00-16.00	Cloudy	10,4	Dry	None	(1153,690,115,130,84,39)
17.09.12	Mon	21.00-22.00	Cloudy	7,3	Dry	None	(408,366,30,22,37,12)

Table A.4: Test data - A regular weekday

Date	Day	Time of	Weather	°C	Fric-tion	Special event	Traffic flow (NS, NS, E, W, NE, SW)
22.09.12	Sat	00.00-01.00	Cloudy	6,3	Dry	None	(189,156,7,8,9,6)
22.09.12	Sat	07.00-08.00	Cloudy	3,9	Dry	None	(150,147,53,1,2,12)
22.09.12	Sat	12.00-13.00	Sunny	7,5	Dry	None	(522,727,35,10,36,12)
22.09.12	Sat	15.00-16.00	Sunny	9,0	Dry	None	(782,509,37,18,50,11)
22.09.12	Sat	21.00-22.00	Cloudy	6,8	Dry	None	(257,282,14,6,22,4)

Table A.5: Test data - A regular weekend

A.2 Simulation tables

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	71.8	147.2	33.5	81.2
1	Restricted CBR	60.6	135.1	35.9	69.1
1	CBR	53.4	129.3	34.7	63.2
2	Static	62.3	137.6	35.5	70.9
2	Restricted CBR	51.8	126.3	37.7	59.6
2	CBR	56.9	133.7	34.0	67.1
3	Static	62.6	137.2	35.9	70.9
3	Restricted CBR	52.7	126.6	38.1	60.2
3	CBR	58.1	134.7	33.6	68.4

Table A.6: Scenario 1 - Night simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	73.0	150.9	37.1	84.0
1	Restricted CBR	61.4	137.7	39.4	70.9
1	CBR	55.5	134.9	35.5	67.8
2	Static	74.4	152.4	36.8	85.4
2	Restricted CBR	53.7	133.2	35.6	66.2
2	CBR	52.5	131.3	36.3	64.6
3	Static	72.2	150.0	37.3	83.3
3	Restricted CBR	58.9	135.4	39.5	68.5
3	CBR	55.4	134.5	35.7	67.6

Table A.7: Scenario 1 - Morning simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	73.9	152.6	31.9	86.5
1	Restricted CBR	42.0	117.3	39.1	51.2
1	CBR	41.9	116.5	40.2	50.4
2	Static	76.0	154.8	31.6	88.6
2	Restricted CBR	42.6	117.7	39.2	51.5
2	CBR	40.4	114.6	40.7	48.4
3	Static	76.7	155.7	31.2	89.4
3	Restricted CBR	44.4	119.9	38.6	53.6
3	CBR	41.4	117.0	40.2	50.7

Table A.8: Scenario 1 - Noon simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	77.0	152.3	37.9	86.0
1	Restricted CBR	67.0	141.8	39.4	75.2
1	CBR	65.4	140.2	39.4	73.8
2	Static	76.7	152.1	37.8	85.6
2	Restricted CBR	67.9	141.6	39.4	75.9
2	CBR	65.6	140.3	39.3	73.8
3	Static	75.7	151.1	37.9	84.6
3	Restricted CBR	66.0	140.5	39.7	73.9
3	CBR	67.2	142.0	39.3	75.3

Table A.9: Scenario 1 - Afternoon simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	72.9	149.9	33.0	83.6
1	Restricted CBR	48.5	123.3	38.4	57.0
1	CBR	49.8	125.8	36.5	59.5
2	Static	70.0	146.8	33.3	80.9
2	Restricted CBR	47.2	121.9	38.5	55.9
2	CBR	50.2	126.1	36.7	60.1
3	Static	71.4	148.5	33.2	82.1
3	Restricted CBR	47.9	122.7	38.3	56.4
3	CBR	50.0	126.0	36.6	59.7

Table A.10: Scenario 1 - Evening simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	72.7	149.0	33.4	82.5
1	Restricted CBR	49.6	123.9	38.2	57.4
1	CBR	53.6	130.1	35.4	63.6
2	Static	70.2	146.6	33.5	80.3
2	Restricted CBR	50.8	125.5	37.7	59.3
2	CBR	52.5	128.5	35.5	62.3
3	Static	68.4	144.0	34.4	78.0
3	Restricted CBR	54.6	129.1	37.4	63.2
3	CBR	48.8	123.9	36.9	58.0

Table A.11: Scenario 2 - Night simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	72.5	149.4	33.2	82.7
1	Restricted CBR	62.5	138.3	35.6	71.6
1	CBR	60.7	138.0	33.1	71.3
2	Static	73.7	139.4	33.1	83.9
2	Restricted CBR	63.9	139.4	35.6	73.2
2	CBR	64.1	141.2	32.4	75.0
3	Static	69.3	145.6	33.9	78.9
3	Restricted CBR	65.4	141.2	34.9	74.5
3	CBR	64.0	141.7	32.2	75.0

Table A.12: Scenario 2 - Morning simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static (2 phases)	74.4	152.2	32.2	86.1
1	Restricted CBR (2 phases)	28.4	101.4	43.2	35.4
1	CBR (2 phases)	30.5	104.1	42.2	38.0
2	Static (2 phases)	73.9	152.0	32.1	85.8
2	Restricted CBR (2 phases)	28.6	101.7	43.1	35.5
2	CBR (2 phases)	31.2	105.1	41.8	38.9
3	Static (2 phases)	75.7	153.8	31.8	87.8
3	Restricted CBR (2 phases)	29.0	101.9	43.0	35.8
3	CBR (2 phases)	29.7	103.4	42.1	37.4

Table A.13: Scenario 2 using 2 phass - Noon simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	71.8	145.6	39.1	79.4
1	Restricted CBR	50.4	125.7	37.7	60.0
1	CBR	44.4	119.4	38.7	53.3
2	Static	68.1	141.8	39.3	75.6
2	Restricted CBR	46.6	118.4	43.7	52.2
2	CBR	43.6	118.7	39.0	52.5
3	Static	69.2	142.9	39.1	76.9
3	Restricted CBR	51.7	127.0	37.6	61.0
3	CBR	43.3	118.3	38.9	52.2

Table A.14: Scenario 2 - Noon simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	73.1	151.1	32.2	84.9
1	Restricted CBR	37.4	109.1	44.3	43.0
1	CBR	38.3	111.7	41.6	45.6
2	Static	75.2	152.9	32.0	87.0
2	Restricted CBR	39.8	111.3	44.5	45.4
2	CBR	39.5	112.7	41.7	46.8
3	Static	74.0	151.5	32.2	85.7
3	Restricted CBR	36.1	107.3	44.9	41.5
3	CBR	39.7	112.6	42.0	46.8

Table A.15: Scenario 2 - Afternoon simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	67.7	143.8	34.2	77.8
1	Restricted CBR	45.1	119.4	38.9	53.5
1	CBR	51.6	127.5	36.0	61.5
2	Static	68.8	145.0	33.9	78.9
2	Restricted CBR	45.3	119.6	39.0	53.5
2	CBR	50.0	125.8	36.3	59.7
3	Static	65.4	141.2	34.6	75.0
3	Restricted CBR	47.0	121.3	38.6	55.1
3	CBR	49.1	125.0	36.4	59.0

Table A.16: Scenario 2 - Evening simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	78.1	156.3	31.7	89.8
1	Restricted CBR	59.2	135.5	36.2	69.0
1	CBR	58.6	135.3	35.4	68.8
2	Static	77.5	156.1	31.5	89.3
2	Restricted CBR	59.6	136.3	35.8	69.5
2	CBR	59.7	136.8	34.8	70.0
3	Static	74.4	152.2	32.4	85.9
3	Restricted CBR	56.7	132.5	36.7	66.2
3	CBR	60.1	136.7	35.3	70.4

Table A.17: Football match on a weekday - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	100.0	181.4	30.1	114.5
1	Restricted CBR	100.0	181.4	30.1	114.5
1	CBR	102.3	184.4	28.5	117.9
2	Static	103.5	184.7	29.9	118.2
2	Restricted CBR	103.5	184.7	29.9	118.2
2	CBR	103.6	185.3	29.2	118.7
3	Static	101.4	182.6	30.0	116.1
3	Restricted CBR	101.4	182.6	30.0	116.1
3	CBR	105.8	187.7	28.9	120.9

Table A.18: Football match on a weekday, ASAP - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	77.6	155.8	31.8	89.2
1	Restricted CBR	60.3	136.3	36.3	69.7
1	CBR	63.0	139.5	35.3	72.9
2	Static	79.7	158.3	31.2	91.4
2	Restricted CBR	61.8	138.0	36.1	71.2
2	CBR	59.1	135.8	35.8	68.9
3	Static	71.0	148.5	33.0	82.0
3	Restricted CBR	54.0	129.8	37.0	63.3
3	CBR	61.9	138.3	35.6	71.7

Table A.19: Football match on a Sunday - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	99.9	181.0	30.2	114.5
1	Restricted CBR	117.3	197.5	32.5	130.7
1	CBR	118.9	199.2	32.1	132.7
2	Static	101.2	182.4	30.0	115.7
2	Restricted CBR	116.8	196.9	32.3	130.3
2	CBR	122.0	202.7	31.3	136.1
3	Static	101.6	182.8	29.8	116.2
3	Restricted CBR	119.7	200.0	32.0	133.2
3	CBR	120.6	201.0	31.8	134.2

Table A.20: Football match on a Sunday, ASAP - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	70.0	146.0	33.8	80.0
1	Restricted CBR	32.2	104.8	42.2	38.5
1	CBR	36.2	109.7	40.7	43.3
2	Static	61.7	136.5	35.5	71.0
2	Restricted CBR	31.7	103.9	42.3	38.3
2	CBR	34.3	106.6	41.8	41.0
3	Static	66.5	142.3	34.5	76.0
3	Restricted CBR	34.0	106.8	42.0	40.5
3	CBR	34.3	107.2	41.5	40.9

Table A.21: Holiday on Sunday morning - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	76.1	155.0	31.5	88.4
1	Restricted CBR	44.0	119.4	38.8	52.8
1	CBR	38.1	111.3	42.7	44.7
2	Static	91.0	157.3	30.8	91.0
2	Restricted CBR	47.4	123.4	38.0	57.1
2	CBR	38.9	112.1	42.6	45.8
3	Static	75.4	154.0	31.8	87.5
3	Restricted CBR	45.3	120.8	38.6	54.3
3	CBR	37.5	110.6	42.9	44.1

Table A.22: Holiday on Sunday afternoon - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	58.5	182.5	26.1	116.4
1	Restricted CBR	31.8	142.1	33.7	75.9
1	CBR	31.4	132.1	39.3	65.9
2	Static	57.2	180.5	26.4	114.5
2	Restricted CBR	28.3	138.2	34.0	72.2
2	CBR	41.0	142.7	37.8	76.6
3	Static	59.6	183.5	26.1	117.1
3	Restricted CBR	31.3	141.4	33.9	75.0
3	CBR	35.3	137.1	37.7	70.7

Table A.23: Scenario 1, noon with slippery roads - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	54.0	176.0	27.2	110.0
1	Restricted CBR	45.9	143.4	38.8	77.3
1	CBR	33.5	150.0	31.2	83.8
2	Static	55.4	177.9	27.0	111.7
2	Restricted CBR	42.7	140.0	39.0	73.9
2	CBR	36.4	153.1	31.0	87.1
3	Static	55.5	177.5	27.1	111.5
3	Restricted CBR	44.2	141.5	39.0	75.5
3	CBR	31.4	150.2	31.4	84.2

Table A.24: Scenario 2, noon with slippery roads - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	84.2	187.0	34.3	120.4
1	Restricted CBR	67.7	170.4	34.6	103.9
1	CBR	74.2	175.0	35.2	108.6
2	Static	83.0	185.4	34.7	118.7
2	Restricted CBR	67.5	169.5	35.0	103.0
2	CBR	70.7	171.1	35.5	104.5
3	Static	83.6	185.9	34.7	119.4
3	Restricted CBR	68.8	171.4	34.6	104.9
3	CBR	74.1	175.3	35.1	108.7

Table A.25: Scenario 1, afternoon with slippery roads - Simulation table

#	Control type	Stop time	Travel time	Speed	Delay
1	Static	55.8	178.3	26.9	112.2
1	Restricted CBR	40.5	126.6	40.5	60.4
1	CBR	33.7	127.7	40.8	61.5
2	Static	53.3	175.6	27.2	109.3
2	Restricted CBR	31.7	126.3	40.6	60.3
2	CBR	34.6	129.1	40.6	63.0
3	Static	55.6	178.1	27.0	111.8
3	Restricted CBR	30.8	125.5	40.6	59.4
3	CBR	33.4	128.2	40.4	62.0

Table A.26: Scenario 2, afternoon with slippery roads - Simulation table

Bibliography

[1] Agnar Aamodt. Knowledge-Intensive Case-Based Reasoning and Sustained Learning. pages 1–6, 1990.

[2] Agnar Aamodt. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7:39–59, 1994.

[3] Hyunchul Ahn, Kyoung-jae Kim, and Ingoo Han. Hybrid genetic algorithms and case-based reasoning systems for customer classification. *Expert Systems*, 23(3):127–144, 2006.

[4] Gunnar Arveland. *Adaptiv kjøretøystyring med spot i gatenett med skjev trafikkfordeling.* PhD thesis, Trondheim, 2005.

[5] A Atanassov and L Antonov. Comparative analysis of case-based reasoning software frameworks jColibri and myCBR. *Journal of the University of Chemical ...*, pages 83–90, 2012.

[6] Mizar Automazione and Quick Reference. Quick Reference ©. *Reproduction*, 2008.

[7] Markus Bauer, Oliver Buchtala, Timo Horeis, Ralf Kern, Bernhard Sick, and Robert Wagner. Technical data mining with evolutionary radial basis function classifiers. *Applied Soft Computing*, 9(2):765–774, March 2009.

[8] Gareth R. Beddoe and Sanja Petrovic. Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering. *European Journal of Operational Research*, 175(2):649–671, December 2006.

[9] A Boury-Brisset and N Tourigny. Knowledge capitalisation through case bases and knowledge engineering for road safety analysis. *Knowledge-Based Systems*, 13(5):297–305, October 2000.

[10] Mario Cools, Elke Moons, and Geert Wets. Assessing the impact of weather on traffic intensity. 2008.

[11] Jean-Marc David, Jean-Paul Krivine, and Reid Simmons, editors. *Second generation expert systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1993.

[12] DL Gerlough and MJ Huber. Traffic flow theory. 1976.

[13] Andrew R Golding, Marina Rey, and Paul S Rosenbloom. Improving Rule-Based Systems 1 through Case-Based Reasoning. 1991.

[14] Juan Jos. JColibri : An Object-Oriented Framework for Building CBR Systems. pages 32–46, 2004.

[15] A Khattak and A Kanafani. Case-based reasoning: a planning tool for intelligent transportation systems. *Transportation Research Part C: Emerging ...*, 4(5):267–288, 1996.

[16] Kaidong Li and N Waters. Transportation Networks, Case-Based Reasoning and Traffic Collision Analysis: A Methodology for the 21 st Century. *Methods and Models in Transport and ...*, 2005.

[17] F.Strøm and Ø.Kheradmandi. Controlling a Signal-regulated Pedestrian Crossing using Case-based Reasoning. (January), 2012.

[18] Thomas H. MAZE, Manish AGARWAL, and Garrett BURCHETT. Whether weather matters to traffic demand, traffic safety, and traffic operations and flow. *Transportation research record*, (1948):170–176.

[19] J Sánchez Medina. Study of correlation among several traffic parameters using evolutionary algorithms: traffic flow, greenhouse emissions and network occupancy. *Computer Aided Systems ...*, pages 1134–1141, 2007.

[20] Tom M Mitchell. *Machine learning.* McGraw-Hill, New York.

[21] Jarkko Niittymäki and Matti Pursula. Signal control using fuzzy logic. *Fuzzy Sets Syst.*, 116(1):11–22, November 2000.

[22] Norwegian Ministry of Transportation and Communications. National Transport Plan 2010-2019. 2010.

[23] JH Park, KH Im, CK Shin, and SC Park. MBNR: case-based reasoning with local feature weighting by neural network. *Applied Intelligence*, pages 265–276, 2004.

[24] C. Jacob R. Hoar, J. Penner. Evolutionary Swarm Traffic. *doi.ieeecomputersociety.org*, pages 1910–1915, 2002.

[25] MM Richter and Stefan Wess. Similarity, uncertainty and case-based reasoning in PATDEX. *Automated Reasoning*, pages 1–14, 1991.

[26] Adel W Sadek, Virginia Transportation, Michael J Demetsky, and Brian L Smith. Case-Based Reasoning for Real-Time Traffic Flow Management. *Computer-Aided Civil and Infrastructure Engineering*, 14:347–356, 1999.

[27] A.W. Sadek, B.L. Smith, and M.J. Demetsky. A prototype case-based reasoning system for real-time freeway traffic routing. *Transportation Research Part C: Emerging Technologies*, 9(5):353–380, 2001.

[28] Roger C Schank. *Dynamic memory.* Cambridge, 1982.

[29] B De Schutter. A multi-agent case-based traffic control scenario evaluation system. *...Systems, 2003. ...*, 2003.

[30] Ken Schwaber. *Agile Project Management With Scrum.* Microsoft Press, Redmond, WA, USA, 2004.

[31] Statens Vegvesen. Drift og vedlikehold av veger.

[32] Kok Khiang Tan, Marzuki Khalid, and Rubiyah Yusof. Intelligent traffic lights control by fuzzy logic. 9(2):29–35, 1996.

[33] Optimizing Traffic, Flows Across, and T H E Road. MIZAR AUTOMAZIONE S.p.A. pages 3–5.

[34] TSS-Transport Simulation Systems. Microsimulator and Mesosimulator Aimsun 6 . 1 User ' s Manual December 2010 © 1997-2010 TSS-Transport Simulation Systems. (December), 2010.

[35] TSS-Transport Simulation Systems. Aimsun 6 . 1 Users Manual March 2011 © 2005-2011 TSS-Transport Simulation Systems. (March):1–319, 2011.

[36] Ø rjan Tveit. *Bruk av adaptiv trafikksignalregulering i byområder.* PhD thesis, Trondheim.

[37] Vegdirektoratet Veg. *Trafikksignalanlegg Planlegging, drift og vedlikehold.* 2007.

[38] Vegdirektoratet Veg. *Trafikksignalanlegg.* 2012.

[39] Viasala Oyj. USER ' S GUIDE Vaisala Remote Road Surface State Sensor DSC111 for Mobile Applications. 2008.

[40] Ying Wang, RL Cheu, and TF Fwa. Highway maintenance scheduling using genetic algorithm with microscopic traffic simulation. *...Meeting of the Transportation ...*, (January):1–20, 2002.

[41] F V Webster. *Traffic signal settings*, volume 39 of *Road Research technical paper.* H.M.S.O., London, 1958.

[42] W Wen. A dynamic and automatic traffic light control expert system for solving the road congestion problem. *Expert Systems with Applications*, 34(4):2370–2381, May 2008.

[43] J A Wentworth. Expert Systems in Transportation. pages 24–29, 1993.

[44] Li Zhenlong. A case-based reasoning approach to urban intersection. *Science And Technology*, (60604008):7113–7118, 2008.