

CITY-BASED PARKING AND  
ROUTING SYSTEM

by

J.L.Boehlé

A thesis submitted in partial fulfillment of  
the requirements for the degree of

Master of Informatics & Economics

Erasmus University Rotterdam

2007

Approved by \_\_\_\_\_  
Chairperson of Supervisory Committee

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Program to Offer Degree \_\_\_\_\_ Authorized \_\_\_\_\_

Date \_\_\_\_\_



ERASMUS UNIVERSITY ROTTERDAM

ABSTRACT

City-Based Parking and Routing System

by J.L. Boehlé

People driving by car from point A to point B usually try to reach their destination as soon as possible. Along the route, however they can experience all sorts of delays. Unforeseen traffic jams, road maintenance and lack of route knowledge on the side of the driver are some of the most common problems that cause delays. Modern day navigational systems like TomTom are able to help the driver reach his destination as soon as possible by calculating a route that in theory represents the shortest route in time from point A to point B. However, the service offered by these devices is limited. The shortest routes in time presented by these devices are only theoretical since most devices are unable to take traffic jams, accidents and road maintenance into account relying solely on static routing algorithms. Furthermore, once the destination is reached the driver himself is required to search for suitable parking place, which can cause considerable delays and hampers the flow of other traffic within the city.

In this thesis, we propose to solve these two hiatuses in the level of service of navigational systems by introducing a City-Based Parking and Routing System (CBPRS). The CBPRS is a distributed system that provides its participants with the comfort of a guaranteed parking place at their point of destination in combination with a time optimal route towards their destination from the outskirts of a city environment monitored by the CBPRS. The CBPRS consists out of a large set of intelligent lampposts placed near intersections and parking places that allow the CBPRS to communicate with CBPRS participants and monitor the status of parking places. Participating drivers reserve parking places monitored by the system via wireless communication with one of the intelligent lampposts placed at each intersection in the city. The system then guides the driver using a constant stream of directional information via the most time optimal path towards the reserved parking place. On the route towards the parking place, the driver provides the CBPRS with route delay information so that the system can calculate the most optimal route towards the destination.

The ant colony meta-heuristic – originally deduced from the behavior of Argentinean ants – is adapted for hierarchical distributed routing in a city environment. The algorithm implements a hierarchy that separates normal roads from main roads and high ways. The normal roads are classified into colonies between which vehicles can travel via the main road network that resides on a higher hierarchical level. The ant-based algorithm is fully distributed with routing tables residing at intelligent lampposts located at intersections. Route delay information communicated to the CBPRS is exchanged between lampposts by use of intelligent agents called ants. The ant-based algorithm is thereby able to route vehicles using current and up-to-date information concerning the actual state of the traffic network, resulting in routes that are optimal in practice.

Via a series of experiments in ever larger and complex city environments, we tested the behavior of the CBPRS. During these experiments, we found that once environments reached the size of cities the CBPRS performed significantly better than static routing solutions offered by navigational systems. Vehicular distributions created by the ant-based algorithm of the CBPRS proved to decrease overall travel times, increase traffic flows and lower the amount and duration of traffic jams within the environment.



## Acknowledgements

The author would like to thank Drs. Dr. L.J.M. Rothkrantz of the Technical University Delft in giving me the opportunity to work with him again on the interesting topic vehicular routing problems and the Ant-based algorithm. His help and ideas in the formulation of the proposal for this thesis document were indispensable. His cheerful mood, personality, commitment and knowledge made our meetings during the writing of this thesis highly interesting and helpful.

Dr. M. van Wezel – my supervisor at the Erasmus University Rotterdam – also has my gratitude for his support during the writing of this thesis. Without his support for my master thesis proposal, I certainly would have had more difficulty in getting it accepted by the University board. His insightful and relevant questions, hints and suggestions during our meetings and his general enthusiasm towards the subject have contributed greatly to this thesis.

I would also like to thank my family – especially my mother and father – for their support through all my years of study. Their encouragements, support and belief in my abilities has helped me in finishing yet another education and this one the final goal of my student career. Finally, I would like to thank Lloyd – our late dog – for the early morning and evening walks that were both relaxing and invigorating.

I hope that reading this thesis will be interesting and provides a clear insight in my activities of the last few months.

*Joost Laurens Boehlé,*

*Capelle aan den IJssel,*

*Wednesday, 25 April 2007*



# Table of Contents

<b>List of Figures .....</b>	<b><i>xi</i></b>
<b>List of Tables .....</b>	<b><i>xiii</i></b>
<b>List of Equations .....</b>	<b><i>xv</i></b>
<b>Preface .....</b>	<b><i>17</i></b>
<b>Chapter 1: Introduction .....</b>	<b><i>19</i></b>
1.1 Statement of Problem .....	<i>19</i>
1.2 City Based Parking and Routing System .....	<i>20</i>
1.2.1 User Perspective .....	<i>22</i>
1.2.2 Operator Perspective .....	<i>23</i>
1.2.3 CBPRS Behavioral Factors .....	<i>23</i>
1.2.4 CBPRS Variable Relationships .....	<i>24</i>
1.3 Purpose of Study .....	<i>25</i>
1.4 Scope of Study .....	<i>25</i>
1.5 Problem Approach .....	<i>27</i>
1.6 Environmental Assumptions .....	<i>27</i>
1.6.1 Vehicular Assumptions .....	<i>27</i>
1.6.2 Technical Assumptions .....	<i>28</i>
1.7 Thesis Project Description .....	<i>28</i>
1.8 Thesis Document Structure .....	<i>29</i>
<b>Chapter 2: Literature Survey on Routing Algorithms .....</b>	<b><i>31</i></b>
2.1 Static Routing Algorithms .....	<i>31</i>
2.1.1 Bellman-Ford Algorithm .....	<i>32</i>
2.1.2 A* Algorithm .....	<i>33</i>
2.1.3 Dijkstra’s Algorithm .....	<i>34</i>
2.2 Dynamic Routing Algorithms .....	<i>36</i>
2.2.1 Routing Information Protocol .....	<i>36</i>
2.2.2 Open Shortest Path First .....	<i>38</i>
2.2.3 Ant Colony Based Algorithm .....	<i>38</i>
2.3 Navigational Systems .....	<i>42</i>
2.3.1 Radio Data System - Traffic Message Channel .....	<i>42</i>
2.3.2 TomTom plus Traffic .....	<i>43</i>
<b>Chapter 3: Distributed Hierarchical Routing .....</b>	<b><i>45</i></b>
3.1 Network Model .....	<i>45</i>
3.1.1 Hierarchy of Sectors .....	<i>46</i>
3.1.2 Routing tables .....	<i>47</i>

3.2 Traffic Condition Information.....	48
3.3 Route Finding System.....	48
3.3.1 Local Ants .....	49
3.3.2 Global Ants.....	52
3.4 Limitations and Possibilities .....	53
<b>Chapter 4: Design of the Simulation Environment.....</b>	<b>55</b>
4.1 Global Design of the Simulation Environment .....	55
4.2 Simulation .....	56
4.2.1 Design patterns .....	57
4.2.2 Software design.....	57
4.3 Infrastructure .....	58
4.3.1 Roads and Lanes.....	59
4.3.2 Intersections.....	59
4.3.3 Vehicular Movement.....	60
4.3.4 Software Design .....	62
4.4 Routing System .....	63
4.4.1 Dijkstra Routing.....	63
4.4.2 Ant Routing .....	63
4.4.3 Software design.....	64
4.5 Parking System.....	64
4.5.1 Software Design .....	65
4.6 Limitations and Possibilities .....	65
<b>Chapter 5: Development of the Simulation Environment .....</b>	<b>67</b>
5.1 Application Environment .....	67
5.2 Technical Details .....	68
5.2.1 Intersections.....	70
5.2.2 Vehicle Types and Behaviors.....	71
5.2.3 Serialization.....	73
5.3 Possibilities and Limitations .....	74
<b>Chapter 6: Experimental Design .....</b>	<b>75</b>
6.1 Conceptualization .....	75
6.2 Operationalization .....	76
6.2.1 Vehicular Operationalization.....	76
6.2.2 Operationalization of the Infrastructure .....	77
6.2.3 Operationalization of the Routing system.....	78
6.3 Method of Experimentation.....	78
6.4 Validity .....	78



6.5 Procedure of Experiments.....	79
6.6 Environmental Inputs.....	80
6.6.1 Simulation Parameters.....	80
6.6.2 Vehicle Parameters .....	80
6.6.3 Ant-Routing Parameters.....	80
6.6.4 Generator Parameters .....	81
<b>Chapter 7: Experiments and Results.....</b>	<b>83</b>
7.1 Validation of the Simulation Environment.....	83
7.1.1 Environmental Settings .....	84
7.1.2 Expected Results .....	86
7.1.3 Results of the Experiment .....	86
7.2 Parking System Performance in a Town.....	89
7.2.1 Environmental Settings .....	89
7.2.2 Expected Results .....	90
7.2.3 Results of the Experiment .....	90
7.3 Parking System Performance in a Village.....	92
7.3.1 Environmental Settings .....	92
7.3.2 Expected Results .....	93
7.3.3 Results of the Experiment .....	93
7.4 Parking System Performance in a City .....	95
7.4.1 Environmental Settings .....	95
7.4.2 Expected Results .....	97
7.4.3 Results of the Experiment .....	97
7.5 Parking System Performance in a Metropolis.....	100
7.5.1 Environmental Settings .....	101
7.5.2 Expected Results .....	102
7.5.3 Results of the Experiment .....	103
7.6 Description of findings .....	106
<b>Chapter 8: Conclusions and Recommendations.....</b>	<b>109</b>
8.1 Conclusions .....	109
8.2 Future Research .....	112
<b>Bibliography.....</b>	<b>115</b>
<b>Appendix I: Data Figures of the Experiments .....</b>	<b>119</b>
I.1 Validation Experiments .....	119
I.1.1 Validation Experiment One .....	119
I.1.2 Validation Experiment Two .....	122
I.2 Experiment Two: a Rural Town .....	124

I.3 Experiment Three: a Village .....	127
I.4 Experiment Four: a City.....	128
I.5 Experiment Five: a Metropolis .....	131
<b>Appendix II: Technological Requirements .....</b>	<b>135</b>
II.1 City Infrastructure .....	135
II.2 Intelligent Lampposts.....	136
II.3 Power Line Communication .....	137
II.4 Parking place sensors.....	138
<b>Appendix III: User Manual .....</b>	<b>141</b>
III.1 Toolbars .....	142
III.1.1 Interaction Toolbar.....	142
III.1.2 Infrastructure Toolbar .....	142
III.1.3 Routing Toolbar.....	143
III.1.4 Parking Toolbar .....	144
III.1.5 Simulation Toolbar .....	144
III.2 Cityscape construction.....	145
III.3 Intersection Interfaces .....	146
III.4 Road Interfaces .....	147
III.5 Simulation Data Extraction.....	149
<b>Appendix IV: Infrastructure Fact sheet.....</b>	<b>151</b>

## List of Figures

Figure 1: Parking service communication .....	20
Figure 2: Intelligent Lamppost in action.....	21
Figure 3: CBPRS Variable relationships .....	24
Figure 4: Variables under study .....	26
Figure 5: A simple graph .....	31
Figure 6: Bellman-Ford algorithm pseudo- code .....	32
Figure 7: A* algorithm unidirectional variant pseudo-code.....	33
Figure 8: Dijkstra's Algorithm in pseudo code.....	35
Figure 9: Dijkstra's Algorithm pseudo code extensions.....	35
Figure 10: Pseudo code for reversing the shortest path.....	36
Figure 11: A network graph .....	37
Figure 12: Ant pheromone laying behavior.....	39
Figure 13: TomTom Navigation Device interface.....	43
Figure 14: An intelligent lamppost at an intersection.....	45
Figure 15: Simplistic representation of a city road network.....	46
Figure 16: Road network hierarchy.....	46
Figure 17: Valid neighbors for the current vehicle .....	51
Figure 18: Layers represented in the simulation environment.....	55
Figure 19: Simulation environment Package Diagram.....	56
Figure 20: Simulation package class diagram.....	57
Figure 21: Rule-30 cellular automaton.....	58
Figure 22: Intersection representation [24] .....	60
Figure 23: Vehicular movement in the cellular automaton [26] .....	62
Figure 24: Infrastructure class diagram .....	62
Figure 25: Incoming driving lanes onto an intersection .....	63
Figure 26: Routing Class Diagram.....	64
Figure 27: Parking system class diagram.....	65
Figure 28: The Cityscape graphical user interface .....	68
Figure 29: Simulation process flow chart.....	69
Figure 30: Vehicle movement procedure flow chart .....	71
Figure 31: Vehicle lane changing procedure flow chart .....	72
Figure 32: Simple Class hierarchy.....	73
Figure 33: Cityscape GUI.....	74
Figure 34: Variables under study .....	75
Figure 35: Generator interface.....	81
Figure 36: City environment for validation experiment one.....	84
Figure 37: City environment for validation experiment two .....	85
Figure 38: Environment for validation experiment three and four .....	85
Figure 39: Ant algorithm parameter experiments .....	88
Figure 40: Travel times of validation experiment three and four.....	88
Figure 41: Town environment .....	89
Figure 42: Experimentation result of the town environment .....	91
Figure 43: Parking place occupancy graph .....	91
Figure 44: Village environment .....	92
Figure 45: Average route time for parking vehicles per experiment. ....	94
Figure 46: Average route time for non-parking vehicles per experiment.....	94
Figure 47: Parking place occupancy graph .....	95

Figure 48: City Environment.....	96
Figure 49: Average travel time for parking vehicles per experiment .....	98
Figure 50: Average travel time for non-parking vehicles per experiment .....	99
Figure 51: Metropolis environment .....	101
Figure 52: Average travel time for parking vehicles during the experiments.....	103
Figure 53: Average travel times for non-parking vehicles during the experiments.....	104
Figure 54: Relationships of variables .....	110
Figure 55: Research question.....	111
Figure 56: Validation experiment one, Average number of vehicles.....	120
Figure 57: Validation experiment one, intersection four routing table.....	120
Figure 58: Validation experiment one, generator 'A' vehicle travel times .....	121
Figure 59: Validation experiment one, generator 'B' vehicle travel times .....	121
Figure 60: Validation experiment one, generator 'C' vehicle travel times .....	121
Figure 61: Validation experiment one, generator 'D' vehicle travel times .....	122
Figure 62: Vehicle count during simulation.....	122
Figure 63: Vehicle travel times for vehicles originating at generator A .....	122
Figure 64: Vehicle travel times for vehicles originating at generator B .....	123
Figure 65: Vehicle travel time for vehicles originating at generator C.....	123
Figure 66: Intersection 9 pre- and post simulation probabilities for colony 0.....	124
Figure 67: Vehicle count of experiments in the town environment.....	124
Figure 68: Experiment one generator to parking place travel time .....	125
Figure 69: Experiment one generator to parking place travel time (scatter graph).....	125
Figure 70: Experiment two generator to parking place travel time .....	125
Figure 71: Experiment two generator to parking place travel time (scatter graph) .....	125
Figure 72: Experiment three generator to parking place travel time.....	126
Figure 73: Experiment three generator to parking place travel time (scatter graph).....	126
Figure 74: Vehicles present in simulation environment .....	127
Figure 75: City experiments vehicle Counts.....	129
Figure 76: Overview of occupied places.....	130
Figure 77: Overview of parking place reservations .....	130
Figure 78: Development of ant traffic during simulations .....	131
Figure 79: Vehicle count during experiments.....	133
Figure 80: Parking place occupancy developments .....	133
Figure 81: Parking place reservation developments .....	134
Figure 82: Development of ant traffic during metropolis experiments .....	134
Figure 83: Communication layers within a city environment.....	135
Figure 84: The intelligent lamppost .....	136
Figure 85: Power line infrastructure .....	137
Figure 86: Power line broadband overview © HowStuffWorks .....	138
Figure 87: Cityscape GUI.....	141
Figure 88: Road construction .....	145
Figure 89: Parking place addition screen .....	146
Figure 90: Default intersection interface .....	147
Figure 91: Ant routing table.....	147
Figure 92: Road Interface.....	148
Figure 93: Lane Interface .....	148
Figure 94: Data extraction screen.....	149

## List of Tables

<i>Table 1: CBPRS advantages and disadvantages in comparison a non-CBPRS situation.....</i>	<i>22</i>
<i>Table 2: RIP routing table development over time.....</i>	<i>37</i>
<i>Table 3: ABC Routing Table.....</i>	<i>39</i>
<i>Table 4: Ant data packet layout.....</i>	<i>40</i>
<i>Table 5: Updated probabilities at intermediate router i.....</i>	<i>41</i>
<i>Table 6: Node routing table basic layout.....</i>	<i>47</i>
<i>Table 7: Local routing table layout example.....</i>	<i>47</i>
<i>Table 8: Global routing table layout example.....</i>	<i>48</i>
<i>Table 9: Local ant memory layout.....</i>	<i>49</i>
<i>Table 10: Global ant memory layout.....</i>	<i>52</i>
<i>Table 11: Vehicle velocities.....</i>	<i>59</i>
<i>Table 12: Validation experiments vehicular distribution.....</i>	<i>84</i>
<i>Table 13: Validation experiment one shortest paths.....</i>	<i>86</i>
<i>Table 14: Validation experiment two; route statistics.....</i>	<i>87</i>
<i>Table 15: Validation experiment two; traffic jams.....</i>	<i>87</i>
<i>Table 16: Ant algorithm pair couplings and results.....</i>	<i>88</i>
<i>Table 17: Town environment vehicular distributions.....</i>	<i>90</i>
<i>Table 18: Village environment vehicular distributions.....</i>	<i>93</i>
<i>Table 19: City environment vehicular distributions per experiment.....</i>	<i>97</i>
<i>Table 20: Traffic jams statistic overview per experiment.....</i>	<i>99</i>
<i>Table 21: Travel times of routes leading to parking garages per experiment.....</i>	<i>100</i>
<i>Table 22: Metropolis environment vehicular distributions per experiment.....</i>	<i>103</i>
<i>Table 23: Traffic jam statistics for the metropolis experiments.....</i>	<i>105</i>
<i>Table 24: Validation experiment one; intersection distances.....</i>	<i>119</i>
<i>Table 25: Dijkstra validation route statistics.....</i>	<i>120</i>
<i>Table 26: Road delays after simulation.....</i>	<i>123</i>
<i>Table 27: Statistics for the rural town experiments.....</i>	<i>124</i>
<i>Table 28: Experimental data overview for experiment three.....</i>	<i>127</i>
<i>Table 29: Traffic jam overview per experiment.....</i>	<i>127</i>
<i>Table 30: Statistics for the city experiments.....</i>	<i>128</i>
<i>Table 31: Statistics gathered during experiments one through three.....</i>	<i>131</i>
<i>Table 32: Statistics gathered during experiment four and five.....</i>	<i>132</i>



## List of Equations

<i>Equation 1: Routing System performance.....</i>	<i>26</i>
<i>Equation 2: Variable measurements.....</i>	<i>26</i>
<i>Equation 3: The probability of the route via node n.....</i>	<i>40</i>
<i>Equation 4: The delay estimated for a communication link.....</i>	<i>40</i>
<i>Equation 5: The virtual delay experienced on the route <math>i \rightarrow d</math>.....</i>	<i>41</i>
<i>Equation 6: The average delay for the route <math>i \rightarrow d</math>.....</i>	<i>41</i>
<i>Equation 7: The probability reinforcement strength.....</i>	<i>41</i>
<i>Equation 8: Probability updating.....</i>	<i>42</i>
<i>Equation 9: Delay measurement for road r.....</i>	<i>48</i>
<i>Equation 10: The virtual delay on the route between node i and d.....</i>	<i>50</i>
<i>Equation 11: The average delay toward node d.....</i>	<i>50</i>
<i>Equation 12: Probability reinforcement strength.....</i>	<i>51</i>
<i>Equation 13: Probability adjustment for the optimal neighbor.....</i>	<i>51</i>
<i>Equation 14: Probability updating.....</i>	<i>51</i>
<i>Equation 15: Cellular automaton rule-30 in binary format.....</i>	<i>58</i>
<i>Equation 16: Vehicle speed and position determination.....</i>	<i>61</i>
<i>Equation 17: Vehicle lane changing Security Constraint.....</i>	<i>61</i>
<i>Equation 18: Vehicle Right to left lane changing rule.....</i>	<i>61</i>
<i>Equation 19: Vehicle Left to right lane changing rule.....</i>	<i>61</i>
<i>Equation 20: CBPRS routing relationships.....</i>	<i>110</i>
<i>Equation 21: CBPRS routing benefits.....</i>	<i>111</i>





## Preface

Imagine yourself running out of the office, quickly stepping into your vehicle while hoping you will still be able to make it to that important meeting in time. You quickly start the vehicle's engine and setup your navigational system by entering in the destination address allowing it to calculate the route. The navigational system guides you quickly onto the highway and informs you about traffic jams. You quickly notice that none of traffic jams appear on your route and breathe a sigh of relief. After a speedy forty-five minute journey you reach the appropriate off ramp on the highway and notice that there are still fifteen minutes left in order to make to the meeting in time. "Plenty of time" you say quietly to yourself.

Turning right at traffic lights on the end of the highway off ramp – as the navigational system suggested – you notice a slight increase in traffic density but nothing to worry. Conceiving on the basis that traffic in the city is always busy at this time of day. Turning left at the next intersection you suddenly find yourself in a traffic jam, you wonder what the reason behind the traffic jam is. You suddenly notice that in the distance there is a large delivery truck maneuvering on the middle of the road in order to drive onto the loading platform of a supermarket. After five minutes, the delivery truck has cleared the road and traffic starts moving again. Ten minutes left you think. The next leg of the journey goes smoothly and the navigational system indicates that by making a left turn at next intersection your destination is only five minutes driving. At the next intersection, the turn to the left is forbidden due to road maintenance. Instead, you head straight on while hearing the navigational system say: "Recalculating route... Please wait". A glance on the display of the navigational system informs you that the new route is a circular detour that again attempts to route you over the road that was closed due to maintenance. You quickly press a few buttons on the device informing it that that specific road is not possible. Looking up from the navigational system you just manage to stop the vehicle in time before crashing into the vehicle that left its road side parking place while you where configuring your navigational system.

Feeling the sweat running along your back you notice that there are only three minutes left to reach your destination. You curse at the traffic and decide to inform the persons you are going to meet that you are slightly delayed. Eventually after driving through the city for almost half an hour the navigational system happily informs you that: "You have reached your destination" – although fifteen minutes later than you had planned. Driving onto the parking lot of the meeting location you notice that all places are taken, "great just what I needed" you breathe to yourself. Driving back into the street you fail to notice any free parking places. You head to the intersection at the end of street hoping that there will be a parking place somewhere. Suddenly out of the blue the navigational system wakes up on reaching the intersection demanding that you turn 180 degrees. Ten minutes later you manage to find a parking place, you quickly park the vehicle and notice that the meeting should have started almost twenty-five minutes ago. Running fully packed to the meeting location you curse and say to yourself that you should have left an hour earlier. Thirty minutes too late you enter the meeting room – sweat running from your face – you apologize to people in the room and immediately notice that they are not happy. While sitting down you prepare yourself for a hard and complex meeting, and hope that a deal is still possible.



## Introduction

People driving by car from point A to point B usually try to reach their destination as soon as possible. Along the route, however they can experience all sorts of delays. Unforeseen traffic jams, road maintenance and lack of route knowledge on the side of the driver are some of the most common problems that cause delays. Modern day navigational systems like TomTom are able to help the driver reach his destination as soon as possible by calculating a route that in theory represents the shortest route in time from point A to point B. More elaborate versions of navigational systems are even able to receive information about traffic jams on major highways and provide alternative routes to minimize possible delays.

In theory, a driver using a modern day navigational system should reach his destination as soon as possible. However, there are two hiatuses in the service provided by navigational systems. The most obvious is that while a navigational system is able to reroute drivers around traffic jams on major highways it has no knowledge about traffic jams in cities. This could lead to a situation where the driver is able to reach the outskirts of a city without delays only to find that on the last leg of his journey he is plagued by inner city traffic jams that his navigational system cannot avoid due to lack of information about the traffic conditions in cities. The second hiatus occurs when a driver has reached his destination and finds all parking places taken. His navigational system is unable to assist him in finding a parking place and therefore the driver is forced to drive around in the vicinity of his destination in search of a place to park his car. While this is not only frustrating to the driver, it also causes delays for other drivers. This is because a driver who is actively searching for a parking place is not fully committed to navigating his car through traffic and therefore hampers the flow of other traffic.

The ideal situation from the driver's perspective could therefore look as follows. A driver travelling by car for example from Delft to the Euromast of Rotterdam uses the navigational system in his car to determine a route to his destination in the centre of Rotterdam. The navigational system guides the driver from Delft to the ring road or outskirts of Rotterdam depending on traffic conditions on the highway to Rotterdam. Once the driver reaches the outskirts of the city, the car automatically contacts a beacon of the parking system of Rotterdam via wireless communication and asks for a reservation of a parking place near the driver's destination in Rotterdam. After the parking system has located a suitable parking place the beacon transmits the new destination to the car and gives partial routing information about the route the driver needs to follow in order to reach his parking place as soon as possible. While driving to the parking place the car automatically requests new routing information from the parking system, in order to avoid any traffic jams that might delay the journey to the parking place. Once the driver arrives at his parking place, the parking system registers this and bills the driver according to the local tariff and the amount of time the car was parked on the parking place. When the driver leaves his parking place the parking system routes the driver out of the city centre again avoiding as many traffic jams as possible.

### 1.1 Statement of Problem

The driver in the preface was confronted with two distinct problems that delayed his arrival. The first problem was the lack of information that his navigational system had about the inner city traffic conditions such as accidents, road maintenance and traffic jams. The second problem was the lack of information about the availability of parking places in the area in which the driver wanted to park. The delay experienced however could have been prevented if the missing traffic condition and parking place availability information was supplied to him.

The delay experienced by one driver is not significant in a city where thousands of vehicular movements occur every day. However, when adding up all the delays experienced by all drivers in the city the cost in terms of economical damage quickly add-up. The department of economic affairs of the Netherlands has estimated that the costs of traffic jams, in terms of economical damage sustained for the whole of the Netherlands, exceed 1.2 billion euro's for the year 2006 [2]. The low accessibility of a city also has a negative impact on the economical climate.

Businesses that are faced with low accessibility experience a host of problems that range from delivery problems – most significant for retailers – to traffic jam frustrated and exhausted employees. These businesses will then be more likely to opt for relocation to other areas where their accessibility is increased in order to minimize their share of economical damages sustained.

The current solutions proposed by governmental agencies and city councils however do not aim at providing the navigational system of the driver with sufficient information in order to optimize his route to his destination. Instead these solutions opt for the construction of large parking facilities on the periphery of cities [3] in the hope that drivers choose to park their vehicle there and use the public transport services in order to reach their destination somewhere in the city. While this solution could alleviate the inner city traffic pressures, they introduce the hassle and uncertainty of traveling via one or more forms of public transportation.

To alleviate these problems we propose another solution in the form of a City Based Parking and Routing System (CBPRS). A CBPRS that would constantly gather up-to-date information about the current traffic conditions and the status of parking places within the city could provide the driver with the information needed to reach his destination without significant delays. The effect of the routing information provided by such a system could have positive consequences not only to those who actively use the system but possibly also to those who do not participate by distributing traffic flows more evenly over the entire city. The CBPRS is described in detail in the following section.

## 1.2 City Based Parking and Routing System.

The CBPRS is an integrated system that provides its participants with the comfort of a guaranteed parking place at their point of destination and a time optimal route towards their destination from the outskirts of a city environment monitored by the CBPRS. This section describes the workings of the CBPRS in detail by first discussing the manner of user interaction with the CBPRS followed by explanation of the workings of the parking service and routing service provided by the CBPRS. After that, the parking system is dissected into two different perspectives discussed in sub-section 1.2.1 for the users and 1.2.2 for the operators. Section 1.2.3 discusses the factors that directly influence the behavior of the CBPRS as a whole. Section 1.2.4 depicts how these perspectives and behavioral factors are assumed to interact with each other and forms the theory behind the working of the parking system as developed for this thesis.

In order to use the parking and routing services offered by the CBPRS participants need to possess a navigational system in order to process and visualize routing information provided by the CBPRS and also a wireless communication device to communicate with the CBPRS while travelling through the city environment. Once the user reaches the operational area of the CBPRS, it sends a request for a parking place reservation within the operational zone of the city. Participants can influence the parking place assigned to them by specifying three factors; their destination, the maximum distance of the parking place form their destination and the maximum tariff they are willing to pay for a certain parking place. These three factors are weighted by the driver according to his preferences to influence the parking place scheduling service as incorporated into the parking service of the CBPRS. Once the parking place is determined, the vehicle is guided by the dynamic routing service of the CBPRS to the assigned parking place. While travelling through the city the ‘driver’ also assists the dynamic routing service of the CBPRS in finding optimal routes by providing location and route delay information to the CBPRS via its wireless communication device.

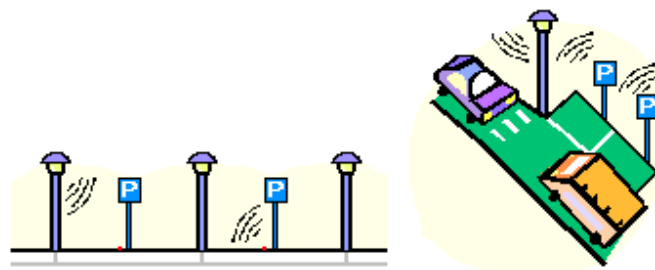


Figure 1: Parking service communication

The parking service provided by the CBPRS keeps track of a number of parking places assigned to the system. However, a system that only stores static information concerning a parking place has serious limitations in terms of billing drivers and optimal scheduling of parking places since arrival and departing information is not provided in real-time. Therefore, each parking place is equipped with a sensor that allows it to detect whether or not a vehicle is parked on it. These intelligent parking places inform the CBPRS parking service when a vehicle has arrived and when it has left so that the driver in question only pays for the period he has actually parked his vehicle on the parking place. The communication between the CBPRS and individual parking place sensors is conducted via intelligent lampposts as depicted in Figure 1. The intelligent lampposts are positioned near the intelligent parking places and monitor one more intelligent parking places within their zone of operation. The intelligent lampposts relay the information received by the sensors via the power-line infrastructure to the central parking system computer that processes and schedules parking places to individual drivers. The scheduling process of the CBPRS finds the most optimal parking place based on the preferences of the driver. The term 'optimal parking place' in this case however is an appraisal between the preferences of the individual driver and all other participants currently requesting a parking place. This should lead to an optimal distribution of all participants over the parking places maintained by the system.

After the reservation of a parking place has been confirmed by the CBPRS, the driver is notified and the destination of the driver is altered to represent the location of the reserved parking place. The driver could then use a static routing algorithm – such as the well-known Dijkstra's algorithm – present in his navigational system to find the parking place. However, this has several drawbacks. While the route provided by the navigational system in theory might represent the shortest path in time to the parking place, in practice this is seldom the case since the navigational system does not account for traffic conditions within the city. The driver's navigational system is also not aware of any accidents or road maintenance in the city. Thus while the driver is assigned a parking place the journey towards the parking place might still be plagued with delays. Therefore, the CBPRS also provides dynamic routing information to the driver.



**Figure 2: Intelligent Lamppost in action**

The routing part of the CBPRS provides and receives information through a system of interconnected intelligent lampposts that are located at every intersection throughout the city. The intelligent lampposts possess a wireless communication device to communicate with vehicles as depicted in Figure 2. While travelling through the city the participating vehicles provide delay and positional information to the intelligent lampposts indicating the delay they encountered while travelling along a certain road. The intelligent lampposts provide the vehicle with up-to-date dynamic routing information for the most time optimal route towards the destination of the driver. Each individual intelligent lamppost represents a routing beacon that is able to route a vehicle towards any destination within the CBPRS operational zone. The delay information received from each individual driver is collected by a variant of the Ant-based routing algorithm that is adapted to support hierarchical and distributed routing (see chapter three). The Ant-based routing algorithm exchanges information between the different lampposts via a series of intelligent agents named ants and is so able to communicate fluctuations in traffic conditions in rapid succession over a wide area. Each beacon of the routing system possesses its own routing tables. Therefore, the routing service is able to operate with one or more intelligent lampposts defunct. Information exchange between intelligent lampposts used for the routing service is facilitated via the power-line infrastructure in the same manner as intelligent lampposts use for the parking system. The heavy use of lampposts by the CBPRS might deter readers. However, the cost of implementing a CBPRS – especially Dutch cities – could be kept low by adapting lampposts currently placed throughout the city. Lampposts within Dutch cities are usually placed on alternating sides of the road with a spacing of 25 meters; this high density of lampposts suits the needs of CBPRS without the requirement of adapting all lampposts within the city. Table 1 summarizes the advantages and disadvantages of the CBPRS.

**Table 1: CBPRS advantages and disadvantages in comparison a non-CBPRS situation**

Advantages	Disadvantages
CBPRS participants are guaranteed to get a parking place at or near their point of destination.	CBPRS capable hardware required in vehicles in order to communicate with the system.
Reduced travel time for participants within the CBPRS zone of operation via the use of dynamic routing information.	Subscription with possible fee required in order to participate in the system.
Increased traffic flows throughout the city caused by a more optimal distribution of vehicles.	Cost associated with the conversion of lamppost into intelligent lampposts throughout the city.
Decrease in the number of traffic jams throughout the city resulting from the increase in traffic flows and more optimal distribution of vehicles throughout the city.	The design and placement of parking place sensors at parking places monitored by the CBPRS.
Improvements in air quality due to increased traffic flows and decreased number of traffic jams.	Requirement of a user base before benefits associated with the system exhibit themselves in a noticeable manner.
Parking place sensors provide a clear and concise set of statistics concerning parking places allowing for the application of optimal vehicular distribution algorithms to the scheduling of parking places.	Routing information exchanged between the vehicles and the CBPRS is 'historical' information reaction to major incidents requires time.
Improved reachability of locations within the city due to higher traffic flows.	City features such as buildings and power lines can influence wireless transmission range of data.

The CBPRS can be viewed upon from two distinct perspectives. The first perspective is that of the user while the second perspective is that of the operator. These two perspectives describe a set of generalized variables to which a weight is assigned that indicates the importance of that variable to each user or operator. The resulting set of weights therefore always differs and is likely to be determined by the specific situation of the user or operator. The set of variables defined in these perspectives is by no means exhaustive but comprises those variables that effect or are influenced by the CBPRS in the most direct manner. The variables defined for the CBPRS itself list those that have the most profound impact considering the services provided by the CBPRS and the variables in the user and operator perspective. These variables are then grouped into a dependency chart that lists the dependencies between the variables.

### 1.2.1 User Perspective

The CBPRS, once deployed in a city environment, is confronted with many different types of individuals that all possess certain characteristics and behaviors specific to that particular individual. These individual characteristics determine the manner in which a certain individual will interact with the CBPRS and why this individual chose to use the services of the CBPRS in the first place. The manner in which an individual interacts with the CBPRS depends on a number of factors of which the most important are his knowledge of the CBPRS and his familiarity with the current local environment. The reasons why an individual chose to use the services of the CBPRS can be generalized by means of the services the CBPRS provides. Based on the CBPRS user services and the user knowledge five composite variables have been defined that symbolize the different reasons why individuals would or would not use the CBPRS services. The guaranteed parking place offered to participants of the CBPRS is without doubt a reason for users to choose the CBPRS. However, since the guaranteed parking place is not a service of the CBPRS but a fact caused by the definition of the CBPRS the guaranteed parking place is not recorded as a variable but is incorporates in the composite variables U1 through U3.

- U1. Distance between original destination and the parking place assigned by the CBPRS.
- U2. Reduced travel time to and from the parking place assigned by the CBPRS.
- U3. Monetary costs associated with parking at a specific parking place.
- U4. Costs associated with a subscription fee for using the CBPRS.
- U5. The usability of the CBPRS as perceived by the user in terms of interaction between the user and the system.

The first three variables symbolize the added value of the CBPRS to the user. The weight assigned to these variables by individuals is volatile and depends on their specific individual needs. The fourth variable is a composite variable that symbolizes the cost. The cost of the CBPRS is expressed as a mixture between monetary units, physical and mental effort in order to effectively use the system. This variable encompasses CBPRS subscription fees, CBPRS equipment installment and maintenance, CBPRS reliability and trustworthiness among others. The fifth variable is a composite variable that symbolizes the ease-of-use that CBPRS services possess. The ease-of-use represents the manner in which the CBPRS provides simplified menus and destination configuration possibilities for novice users or those that are not familiar with the current city. More extensive and complex menus for expert users could be used to specify the destination in more elaborate detail and options. The weight assigned to this variable is likely to change overtime from simple to expert as users become more accustomed to the CBPRS.

### 1.2.2 Operator Perspective

The operator perspective encompasses all individuals that are assigned with the task of operating the system but focuses on the actual exploiters of the system. A private company could exploit the CBPRS, however the city council or a venture between the city council and parking garage operators seems more plausible. The focus therefore lies on the variables that are deemed important from the city council operator perspective. The city council has to consider the public benefit when deciding when to invest public funds in a certain project. The operator perspective consists out of six variables that focus on the public benefit the CBPRS could deliver.

- O1. Improvement in traffic flows throughout the city.
- O2. Decrease in the number and length of traffic jams throughout the city.
- O3. Improved air quality and other environmental benefits.
- O4. Costs associated with deploying and maintaining the CBPRS infrastructure.
- O5. Revenue derived from operating the CBPRS system.
- O6. Economical benefits

The difference between a public venture and private is that the requirement for public projects does not have to encompass profit per se, if and only if the public benefits resulting from the system outweigh or offset the monetary cost. However, the point at which such a situation might occur differs for every city council and a situation where the system generates revenue while providing public benefits is always preferred. The first two variables symbolize the reachability of the city that would result from improved traffic flows and a reduction of traffic jams. The third variable symbolizes the improvement of the city's environmental quality with an emphasis on the air quality. This variable however is depended on the first two. Variables four and five deal with the financial aspects of operating the CBPRS. The sixth variable symbolizes the economical benefits a city's economy could gain if the CBPRS is utilized to its full extend.

### 1.2.3 CBPRS Behavioral Factors

Many variables influence the efficiency and effectiveness of the CBPRS within a city environment. The measure in which these variables affect the CBPRS however is not known beforehand and is subject to experimentation and validation. The CBPRS parking and routing services as described before lead to the definition of nine variables that all affect the CBPRS.

- F1. Percentage of CBPRS participants.
- F2. Scheduling of parking places to participants.
- F3. City road traffic infrastructure.
- F4. Traffic pressures within the city.
- F5. Number of parking places and their distribution throughout the city.
- F6. The time participants park their vehicles on a certain parking place.
- F7. Transmission ranges of the intelligent lampposts within the city environment.
- F8. Vehicular behaviors exposed by drivers.
- F9. Routing algorithm effectiveness and efficiency.

The CBPRS is depended on information for its operations. Therefore, the number of informational contributors referred to commonly as participants is of major importance for the quality of solutions presented by the CBPRS in terms of routes and parking places. The second variable encompasses the scheduling process of parking places to participants. The CBPRS should be able to schedule parking places on an individual basis without losing sight of the global situation. The CBPRS should weigh individual goals such as the destination, parking place monetary cost and parking place distance in such a manner that CBPRS provides the greatest benefits to all its participants. The third variable is the road traffic infrastructure. The CBPRS depends for its routing solutions on the distribution of vehicles over the roads within the city to avoid traffic jams and other causes of delay. This however requires that the infrastructure is able to support this. The traffic pressure in the fourth variable is depended on the previous but is also influenced by the type of city in which the CBPRS is deployed. A rural town with a limited population is likely to have lower traffic pressures than a metropolis. Variables five and six influence the effectiveness of scheduling solutions provided by the CBPRS. The seventh variable, the transmission range of intelligent lampposts, influences the monetary cost of deployment. A low transmission range due to severe signal loss would require the deployment of more intelligent lampposts. Individual driver behaviors, as represented in the eighth variable, not only model the driving style of the driver but also its tendency to follow CBPRS instructions. The ninth variable represents the behavior of the routing system itself a depended of previous variables but also a of its own algorithmic behavior in terms of its effectiveness and efficiency in achieving the benefits associated with the CBPRS.

### 1.2.4 CBPRS Variable Relationships

The relationship diagram as shown in Figure 3 depicts in detail how the user and operator perspective variable interact with each other and the CBPRS variables. The user perspective variables are prefixed with the letter 'U', the operator perspective variables are prefixed with the letter 'O' and the CBPRS variables are prefixed with the letter 'F'. The relationship between these variables defines the hypothetical interactions between the CBPRS, the users and operators. These hypothetical relationships are subject to validation through means of computer-simulated experiments.

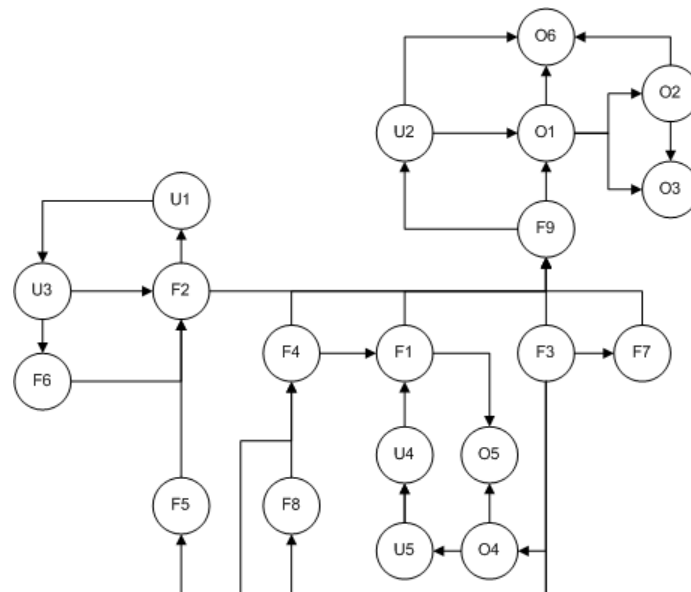


Figure 3: CBPRS Variable relationships

The diagram shows only one independent variable 'F3' that represents the city road infrastructure as described in the previous section. All other variables are assumed depended on one another to a certain degree and eventually should cause three phenomena's to become visible; an increase in economic benefits, an improvement in environmental benefits and revenue derived from the system. These three benefits do not occur within the operator perspective without reason.



The operator of a system requires one or more benefits derived from that system to justify the existence of such a system considering the resources invested. Furthermore, the goal of a system such as the CBPRS should be to bring benefits to its operator whereas the benefits to the user are only a means to a goal from the operator's perspective.

### 1.3 Purpose of Study

The CBPRS constantly gathers information about the traffic densities and delays experienced on the roads within the city. Based on this local information it is able to provide routes through a city that should have minimal delays. Providing the extra service of a guaranteed parking place once the driver arrives at his destination the CBPRS does not only minimize any possible delay for the driver in question but also for other drivers since the time to locate the parking place and execute the parking procedure is minimized. This in theory leads to a state wherein drivers, business and the city council benefit from such a system.

The purpose of this study is to determine the impact of a city-based parking and routing system, that employs distributed dynamic routing instead of static routing as present in navigational systems, within the bounds of a city environment in terms of reducing the delay in travel time experienced by drivers. Furthermore, this study also evaluates the economical feasibility of CBPRS by measuring the number of participants required for an optimal effect off the system and the benefits this has in terms of reduced economical-damages. This paragraph is formalized into a research question stated below.

*“Could a parking system that allows drivers to reserve a parking place at or near their point of destination within a city and that provides dynamic routing information – using the ant-based routing algorithm – for the most time optimal route through the city to the parking place based on current traffic conditions within that city improve the overall traffic flow in a city as compared to a situation wherein drivers are responsible for finding a parking place at their destination and use Dijkstra’s static routing algorithm to guide them to their destination?”*

The null-hypothesis in this study is based on the assumption that drivers currently use either their static routing navigational system or their own local road knowledge to lead them to their point of destination. The route preferred by these drivers is, for the purpose of this study, interpreted as the shortest path in time between the point of origin and the driver's destination. Dijkstra's well-known static routing algorithm is used to represent the routes taken by these drivers. The shortest path in time is deferred from the relation between road length and maximum vehicle speed allowed. Drivers in this situation are not aware of traffic conditions within the city and therefore assume that their chosen path is the shortest path available. Once these drivers reach their point of destination they will need to search for a parking place near their destination.

Our hypothesis, based on the variable relationships presented in the previous section, is that drivers that opt for usage of the CBPRS will achieve significant travel time benefits as opposed to drivers in the null-hypothesis situation. The dynamic routing information provided via the ant-based algorithm should enable the driver to drive via a route that provides the most optimal traffic flow towards its destination while the guaranteed parking place – chosen according to the driver's specification – alleviates the hassle of finding a suitable parking place. This in theory would lead to a decrease in the overall travel time for these drivers. The travel time benefits for participating drivers could lead to a better distribution of traffic within the city allowing for increased traffic flows throughout the city. This increase should be observed when comparing this hypothesis with the city traffic flows as resulted from the situation described for the null-hypothesis. In order to verify our hypothesis we measure the effects of the CBPRS in comparison to the null-hypothesis through a series of experiments conducted in a simulation environment specifically developed for this study.

### 1.4 Scope of Study

The CBPRS as presented in this chapter covers a wide range of scientific fields. Not only does the CBPRS encompass scheduling and routing problems but an all-encompassing study would also require a detailed investigation of social behavior to determine the user and operator perspective variables and influencing factors in detail. The research question posed in section 1.3 focuses on benefits derived from providing distributed hierarchical routing to individuals and the impact of such a routing algorithm on city wide traffic flows. These benefits are to be determined via experiments conducted within a simulation environment specifically designed and developed for this purpose.

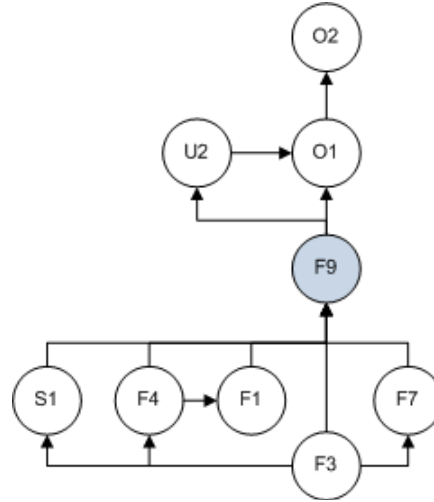


Figure 4: Variables under study

The scope of this study therefore condenses the CBPRS problem to one that focuses on the dynamic distributed hierarchal routing service 'F9' provided by the CBPRS and makes assumptions concerning problem-fields related to the CBPRS. Figure 4 depicts the variables and their causal relationships that are subject of study within this thesis. The scheduling service as provided by the CBPRS and its related variables are incorporated into a new composite variable named 'S1'. This service provides parking place scheduling services on an individual basis and does not attempt to find a global optimal solution.

While this limits the contribution of the scheduling service to the CBPRS it allows for a more detailed study of the effects caused by the routing service of the CBPRS. The performance of the routing service provided by the CBPRS can then be depicted using the equation as presented in Equation 1. The parameter  $\alpha$  represents effects of the routing algorithm itself on its performance when all other variables are zero,  $\beta$  represents the weight of each variable in the equation and  $\epsilon$  represents the standard error in measurement. For the purposes of this study, we assume a linear relationship between the variables under study.

$$F_9 = \alpha + \beta_1 \cdot F_1 + \beta_2 \cdot F_3 + \beta_3 \cdot F_4 + \beta_5 \cdot F_7 + \beta_6 \cdot S_1 + \epsilon$$

Equation 1: Routing System performance

Economic benefits expressed in monetary costs derived from the CBPRS are hard to measure from experiments conducted through simulations alone and depend not only on the performance of the CBPRS in a realistic environment but also on the valuation of benefits derived from the system by users and operators. Conclusions on the basis of this study concerning these three phenomena 'O3', 'O5' and 'O6' would be invalid.

$$U_2 = \beta_1 \cdot F_9 + \epsilon$$

$$O_1 = \beta_1 \cdot U_2 + \beta_2 \cdot F_9 + \epsilon$$

Equation 2: Variable measurements

Therefore, research focuses on measuring the effects of the routing system on the variables 'U2' and 'O2'. The effects measured concerning these variables allow for valid conclusions concerning the CBPRS in terms of routing. However, the data gathered during the experiments performed in this document is insufficient in order to determine actual individual weights assigned to these variables under study. Therefore, only indications concerning the effect of an increase or decrease will be given in order to draw conclusions. These conclusions are then used to give an indication concerning the possible economical benefits; these indications however are not suited for generalization beyond the means of this study. Equation 2 shows the equations for measuring the effects of the variables 'U2' and 'O1'. The equation for the variable 'O2' is not listed since it has direct causal relation with the variable 'O1'.

## 1.5 Problem Approach

The dynamic routing system part of the CBPRS ('F9') depicted in Figure 4 in the previous section is colored blue since it is the pivotal point of this study. The variables depicted below the routing system are those that directly influence the performance of the routing system. Combined with the behavior of the routing system, that is inherent to the ant-based algorithm, the effectiveness of the routing algorithm on individual travel times ('U2'), an increase in traffic flows ('O1') and a decrease in traffic jams ('O2') can be measured. The measurement of these three variables would then allow for validation of the hypothesis as posed in section 1.3 and provides an answer to the research question. The problem approach applied in this thesis can be subdivided into three distinct parts; the first part is a theoretical study into routing algorithms and the development of an ant-based routing algorithm that suites the needs of this study. The second part is the development and implementation of a simulation environment. The third and final part is concerned with experimenting and answering the research question.

Two different routing algorithms are used in this thesis. Dijkstra's routing algorithm is applied to provide static routing services that do not account for traffic conditions. The ant-based algorithm is applied to provide the dynamic routing services required for the CBPRS. Before experiments are conducted with these algorithms, a literature study is conducted in order to investigate the workings of both algorithms. Based on the literature study the ant-based algorithm is adapted to suit the distributed and hierarchical needs of the CBPRS.

Measuring of the variables is done via a computer-based simulation environment specifically developed for the purposes of the study. The simulation environment will allow for simulations of traffic behaviors in a pre-defined city road infrastructure. The simulations are executed multiple times on the same city road network to measure traffic conditions using the null-hypothesis method – static routing via Dijkstra's algorithm – and the CBPRS system – dynamic hierarchical distributed routing via the Ant-based algorithm.

Measuring these variables in a controlled simulation environment enables the negation of undesired influences present in a realistic environment but not desired for the purposes of this study. Details concerning the design of the simulation environment and the subsequent translation into a computer language can be found in chapters four and five of this thesis document.

Experiments are conducted to compare the performance of vehicles routed via Dijkstra's static routing algorithm and the dynamic hierarchical distributed Ant-based routing algorithm. These experiments are conducted within five different cities that mimic city environments ranging in size from a rural town to a metropolis. The measurements obtained through these simulations are then used to formulate conclusions on the performance of the CBPRS in each city environment as well as a global conclusion that gives an answer to the research question of section 1.3.

## 1.6 Environmental Assumptions

The focus of the simulation environment lies on functionality and methods that assist in the task of measuring the effects of the variables under study. To support this measuring task certain assumptions have been made about the conditions under which the CBPRS will operate and how vehicles will interact with the CBPRS. While these assumptions do not conflict with the purpose of this thesis, they are not maintainable when attempting to implement the CBPRS in a real-life environment. The assumptions have been grouped into two sections; the first discusses the assumptions made concerning vehicular interaction with CBPRS, the second discusses the assumptions made concerning the infrastructure of city and the technical details of the CBPRS.

### 1.61 Vehicular Assumptions

Different human beings exhibit different forms of behavior and are liable to draw different conclusions on the basis of identical information. This makes the interaction between the CBPRS and the drivers participating in the system highly complex and difficult to simulate. The simulation of situations in which drivers opt to ignore the routing instructions provided by the CBPRS or choose to park in a parking place that was not reserved for them does not add to the current simulations. This has lead to the definition of four assumptions, listed below, on the interactions between the CBPRS and vehicles.

- The vehicles travelling through the simulated city environment abide by the rules of the parking system. All drivers are assumed to be fair-play parking fanatics in the sense that they will never park on a parking place monitored by the system that is not reserved for them and that routing directions provided to the driver are followed without question.
- Vehicles that make use of the service provided by the CBPRS are equipped with the appropriate hardware and software needed for communication with the CBPRS and possess a device, most likely a modern day navigational system that is able to show them the routing information received from the parking system.
- Drivers that participate in the CBPRS, but who are currently not seeking a parking place support the CBPRS by providing traffic condition information.
- Drivers that do not participate in the CBPRS use routing information generated using Dijkstra's algorithm to travel between two locations. While this assumption in practice does not hold, since drivers might choose sub-optimal routes based on their own preferences, it enables simulation of non-CBPRS participants in a predictable manner.
- Public transportation service busses travelling through the city provide traffic condition information to the parking system.

### 1.6.2 Technical Assumptions

The infrastructure consisting of parking place sensors, intelligent lampposts and the electricity network is complex and highly error prone when implemented in reality. Solving these problems in the simulation environment however does not add to the purpose and goals of this thesis. Therefore, it is assumed that the CBPRS is able to function without errors. This assumption is further supported by the five sub-assumptions listed below.

- Other objects in the city that in reality could cause signal interference do not influence the transmission range of the wireless communication devices installed in vehicles and lampposts.
- The transmission range of the wireless devices is in all cases sufficient to support communication between the CBPRS and parking places or vehicles that are within reasonable distance of a wireless communication device. The reasonable distance of these wireless communication devices is set to 40 meters at maximum. This represents a fourth of the theoretical upper limit of WIFI communication via the IEEE 802.11b standard at 11 Mbps.
- The sensors installed into the parking places can be read and updated without error by their controlling lamppost.
- The sensors installed into the parking places are able to obtain enough electricity to remain operable.
- Communication via the electricity infrastructure provides data transmission speeds that satisfy the needs of the CBPRS and the current electricity infrastructure is able to handle the communication method as required by the CBPRS.

## 1.7 Thesis Project Description

This thesis project is conducted through the execution of eleven distinct steps that all contribute to answering the research question. The steps form the basic plan of this study and are represented in the thesis document structure described in the next section.

1. Literature review.
2. Design initial routing algorithm.
3. Design the simulation environment.
4. Develop the simulation environment and algorithm.
5. Conduct experiments using the simulation environment
6. Validation.
7. Design revised model of the routing algorithm and the simulation environment.
8. Program the revised routing algorithm and the simulation environment.
9. Conduct experiments.
10. Validation.
11. Analyzing of the results.

## 1.8 Thesis Document Structure

The document structure of this thesis has the following layout. Chapter two describes the static and dynamic routing algorithms evaluated for the purposes of this study. This chapter also gives a brief overview of the possibilities of current day navigational systems to allow the user to compare the difference between these modern day hardware devices and the proposed CBPRS. Chapter three describes the distributed hierarchical Ant colony based routing algorithm developed for this study and used to conducted routing system experiments. Chapter four describes the software design of the CBPRS simulation environment and motivates the reasons behind design choices made. Chapter five discusses the implementation of the CBPRS simulation environment and the technical details. Chapter six discusses the experimental design devised for conducting experiments with the CBPRS simulation environment in order to answer the research question. Chapter seven describes the experiments conducted and discusses their purpose, environmental settings, expected results and outcomes. Chapter eight combines the information gained from the experiments and gives an answer to the research question using the insights gained from the conducted experiments.



## Literature Survey on Routing Algorithms

Static routing algorithms present in most modern day navigational systems route the driver to his point of destination within a city via a set of predetermined costs assigned to the roads in a city. The costs assigned to these roads are predetermined and stored in memory by the manufacturer. These costs usually express the time it would take the driver to travel along a road or the length of the road in meters depending on the preferences of the driver. The downside of routing drivers through a city in this manner is that the routing algorithm used in the navigational system is unaware of inner city traffic conditions such as traffic jams, accidents and road maintenance. Therefore, the driver can be lead along a route that in theory was the most optimal route to the destination but in practice turns out to be a suboptimal route due to current traffic conditions.

The solution to this problem can be found in dynamic routing algorithms that route vehicles, on the basis of up-to-date information about inner city traffic conditions, to their destination. The costs assigned to roads by these types of algorithms changes constantly depending on the current traffic conditions of the roads within the city. The benefit of up-to-date information about the state of the inner city traffic network is that the dynamic routing algorithm is able to recognize traffic jams, accidents and road maintenance. Therefore, the routing algorithm can calculate routes through the city that avoid these delays leading to routes that are time optimal for the current state of road network within the city. The downside of these algorithms is that they require a constant up-to-date supply of traffic condition information that, due to the high proximity of roads within a city, makes them impractical for application in navigational systems.

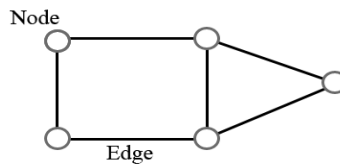


Figure 5: A simple graph

The routing algorithms described in the current chapter all operate on a graph; an example is shown in Figure 5. A graph  $G(V, E)$  consists out of a set of nodes  $V$  also commonly referred to as vertices and a set of edges  $E$ . The edges represent the connections between two nodes and can represent either a bidirectional or a unidirectional connection. In the case wherein all edges in a graph are unidirectional, the graph is said to be a directed graph. The road network is an example of a directed graph. The edges in the graphs searched by routing algorithms are weighted with a certain value that represents the cost of travelling over such an edge to another other node.

### 2.1 Static Routing Algorithms

Static routing algorithms are those types of routing algorithms that operate on graphs with fixed edge weights. This means that once a shortest path is calculated any change in the graph requires a complete recalculation of the shortest path. As previously stated static routing algorithms are commonly used in modern day vehicle navigational systems to quickly calculate routes between two points. Over the years many different types of static routing algorithms have been developed. Some of the most well known static routing algorithms are the Bellman-Ford algorithm, A\* and Dijkstra's algorithm. While these static routing algorithms find the shortest path in a graph the requirements they impose on such a graph differs. Algorithm execution time also differs between these algorithms. This section describes Bellman-Ford, A\* and Dijkstra's algorithm in detail to show their main similarities and differences. The emphasis of this section however lays with Dijkstra's algorithm since this algorithms plays a vital role in this thesis project.

## 2.1.1 Bellman-Ford Algorithm

The Bellman-Ford algorithm [4] [5] is a static routing algorithm that is able to find the shortest path in a graph between a given source node and destination node. The Bellman-Ford algorithm is a special variant of static routing algorithms since it is able to find the shortest path in a graph where some of the associated edge weights are negative. The Bellman-Ford algorithm operates in a manner similar to Dijkstra's algorithm presented in the next section but executes an extra step to ensure that no negative path cycles are formed that prevent proper path finding. The Bellman-Ford algorithm running time is  $O(V \cdot E)$  where  $V$  represents the nodes in the graph and  $E$  the edges.

```
01 function Bellman-Ford(G, s)
02   for each node v in V[G] do // Iterate through each node in the set V[G].
03     d[v]:= ∞ // Set the weight of each node to infinity.
04     previous[v]:= null // Set the previous node on the shortest path to null.
05   done
06   d[s] = 0; // Set the weight of the source node to zero.
07   for each node u in V[G] do // Iterate through all nodes in the set V[G].
08     for each edge (u,v) where v ≠ u do // Iterate through all edges of the current node v.
09       if d[u] + w(u,v) < d[v] then // Determine if the current route to v via u is shorter.
10         d[v]:= d[u] + w(u,v) // Relax the weight of node v.
11         previous[v]:= u // Set the previous node on the shortest path to v as u.
12       end if
13     done
14   done
15   for each edge (u,v) in E[G] do // Loop through all edges in the set E[G].
16     if d[u] + w(u,v) >= d[v] then // Determine if the current to d via u is shorter.
17       Error // A negative cycle is detected.
18     end if
19   done
```

Figure 6: Bellman-Ford algorithm pseudo-code

The pseudo code of the Bellman-Ford algorithm presented in Figure 6 shows the algorithmic steps required to find the shortest path between the source node  $s$  and all other nodes in the graph  $G$ . The graph  $G$  should be of the type  $G(V, E)$  in which  $V$  represents the set of nodes in the graph and  $E$  the set of edges of which the weights can be negative. The algorithm starts – lines 02 through 05 – by iterating through all nodes and setting the weight of each node to infinity, to indicate that the optimal route towards that node is unknown. The previous node on the shortest path is set to undefined as no shortest path exists at present. This step removes any entries made by possible previous searches on the graph  $G$ . Then the algorithm sets the weight of the source node  $s$  to zero to ensure that shortest paths originate from the source node.

After initialization is complete, the algorithm enters the path finding process – lines 07 through 14. The lines 07 and 08 show two iterations, line 07 iterates through each node  $u$  in the graph  $G$  and the line 08 iterates through each edge of that specific node  $u$ . The algorithm on line 09 compares if the weight associated to node  $v$  located on the opposite side of the currently selected edge is higher than the weight assigned to node  $u$  plus the weight needed to cover the edge between node  $u$  and  $v$ . If the comparison evaluates to true the weight associated with node  $v$  is set to the weight associated with node  $u$  and the weight of the edge between node  $u$  and node  $v$ , as shown on line 10. The previous node on the shortest path towards the source node  $s$  for node  $v$  is set to  $u$ . Once the iteration process completes the shortest paths between the source node and all other nodes in the graph  $G$  are known.

The final phase of the algorithm – lines 15 through 19 – verifies that no negative weight cycles are present in the graph. If a negative weight cycle exists in the graph and lies upon the shortest path leading from the source there exists no shortest path since for every path a shorter path can be found by traveling via the negative weight cycle. The verification process is started by iterating through all edges as shown on line 15. Then for each edge in the graph  $G$  the algorithm evaluates if the weight of node  $u$  plus the weight of the edge between node  $u$  and node  $v$  is greater than the weight stored in node  $v$ .



If this comparison evaluates to true the graph  $G$  contains a negative weight cycle and no shortest paths exist, the algorithm exits by giving an error as shown on line 17. If the comparison on line 16 evaluates to false no negative weight cycles exist and a shortest path exists within the graph  $G$ .

The shortest path between the source node and any destination in the graph  $G$  can then be found by iterating through the list of previous nodes starting with the destination node  $d$  and continuing until the source  $s$  has been reached. The reversal of this list is then the shortest path between node  $s$  and the destination node  $d$ .

## 2.1.2 A\* Algorithm

The A\* algorithm [6] is a well known static routing algorithm that uses a heuristic estimate function to find the shortest path between a given source and destination node in a graph with non-negative weights. The A\* algorithm uses the heuristic estimate function to focus the shortest path search in the direction of the destination node. This focusing is done by adding the value resulting from the heuristic estimate function for the current node and the destination to the final edge weight using the function  $f[v] := h[v] + d[v]$ . In this function  $d[v]$  represents the weight associated to the current node  $v$ ,  $h[v]$  represents the heuristic estimate and  $f[v]$  represents a composite value that represents the estimate of the best route through  $v$ .

An example of such a heuristic function is the astronomical distance between two nodes. For each node being evaluated by the A\* algorithm the astronomical distance is incorporated into the final node weight. The astronomical distance for a node that lays in-between the source and destination node is shorter than for a node that does not. Therefore, the node in-between is favored. The A\* algorithm is thereby able to visit less nodes. If the null heuristic estimate function is applied to A\* algorithm – this heuristic estimate function output is zero for every combination of nodes - the algorithm operates in exactly the same fashion as Dijkstra's algorithm. The default running time for the A\* algorithm depends on the used heuristic function in the worst case the running time is exponential  $O(V^V)$  while in the most optimal case it is polynomial [7]. Further reduction of the search time can be achieved by executing two concurrent A\* algorithm path searches where one starts at the source node and one at the destination node. The point where search areas of these concurrent A\* algorithms meet represents the shortest path between the source and destination node.

```

01 function A*( G, s, d )
02   for each node v in V[G] do           // Loop through all nodes in the set V[G].
03     d[v]:= ∞                          // Set the weight of each node to infinity.
04     previous[v]:= null                // Set the previous node on the shortest path to null.
05   done
06   d[s]:= 0                            // Set the weight of the source node to zero.
07   Q:= s;                              // Add the source node to the open node set.
08   C:= ∅                                // Create an empty set for closed nodes.
09   while Q ≠ ∅ do                      // Iterate through Q until the set is empty.
10     u := min(Q)                       // Extract the node from Q with the lowest composite (f) value.
11     C := C ∪ u                        // Add the extracted node to the closed list.
12     if u == d then                   // Does the node u represent the destination.
13       return                          // Destination found end algorithm.
14     end if
15     for each edge (u,v) where v ≠ u do // Iterate through each outgoing edges of u.
16       if (v ∈ Q or v ∈ C) and d[u] + w(u,v) > d[v] then // Determine if weight should be relaxed.
17         continue                      // skip current node and continue with next.
18       end if
19       d[v]:= d[u] + w(u,v)            // Relax the weight of node v.
20       h[v]:= heuristic(v)            // Execute the heuristic function on node v.
21       f[v]:= d[v] + h[v]             // Set the composite value for node v.
22       previous[v]:= u                // Set the previous node on the shortest path to v as u.
23       Q:= Q ∪ v                      // add v to the open set.
24   done
25 done

```

Figure 7: A\* algorithm unidirectional variant pseudo-code

The pseudo code of the A\* algorithm presented in Figure 7 shows the algorithmic steps required to find the shortest path between the source node  $s$  and a destination node  $d$  in the graph  $G$ . The graph  $G$  should be of the type  $G(V, E)$  in which  $V$  represents the set of nodes in the graph and  $E$  the set of edges of which the weights should all be positive. The algorithm starts – lines 02 through 05 – by iterating through all nodes in the graph  $G$ . The weight of each node is set to infinity to signal that the shortest path between the source node  $s$  and the current node  $v$  is unknown. The previous node on the shortest path leading from the source towards node  $v$  is set to undefined since no shortest path is known at present.

The weight assigned to the source node  $s$  is set to zero on line 06. The resulting composite value  $f[s]$  for node  $s$  depends on the heuristic function used. The source node  $s$  is added to the open set on line 07. The open set represents the nodes that need to be evaluated by the algorithm. Line 08 defines the closed set, this is a set of nodes that have been evaluated by the algorithm and have been found to be on the shortest path. The closed set is initially constructed empty.

The A\* algorithm then starts the actual route finding process which is shown on lines 09 through 25. On line 09, the algorithm initializes a loop that iterates until the open list is empty. The algorithm extracts the node from the open list  $Q$  that has lowest composite value  $f[u]$ . The node  $u$  that is extracted is removed from the open list upon extraction. The node  $u$  is added to the closed list on line 11. The extracted node is added to the closed list since it has the lowest  $f$  value of all nodes, this indicates that it lies on the shortest path. Lines 12 through 14 then evaluate if the extracted node  $u$  represents the destination node. If the evaluation results in ‘true’ the destination node has been found and the shortest path is found. The A\* algorithm then terminates on line 13. If the evaluation results in ‘false’ the search is continued.

The A\* algorithm then starts to iterate through each outgoing edge from node  $u$  on line 15. Line 16 then evaluates if the node  $v$  located on the opposite side of the edge from node  $u$  is present on the open or closed list and if the weight of node  $v$  is greater than the weight associated with node  $u$  plus the weight associated with edge between node  $u$  and  $v$ . If the evaluation evaluates to ‘true’ the node  $v$  is not processed in this iteration and the next edge is selected, otherwise the algorithm proceeds with line 19. Line 19 through 21 update the values  $d[v]$ ,  $h[v]$  and  $f[v]$  stored in node  $v$ . The value of  $h[v]$  is updated by executing the heuristic function currently in use. The previous node on the shortest path from the source node  $s$  to the current node  $v$  is set to  $u$  on line 22. Finally, the node  $v$  is added to the open list to ensure that outgoing edges from node  $v$  are evaluated if the A\* algorithm in a subsequent step selects the node.

The shortest path between the source node and any destination node  $d$  in the graph  $G$  can be found by iterating through the list of previous nodes starting with the destination node  $d$  and continuing until the source  $s$  has been reached. The reversal of this list then represents the shortest path between node  $s$  and the destination node  $d$ .

### 2.1.3 Dijkstra’s Algorithm

Dijkstra’s algorithm [8] is a shortest path-finding algorithm that is guaranteed to find the shortest path in a single-source shortest path search for a directed graph that has no negative weights attached to its edges. Dijkstra’s algorithm is a greedy algorithm. This stems from the fact that in each step of the path finding process it chooses the local optimum, which is the edge with the lowest weight attached. This in order to find a global optimum, which is the shortest path from the source node to the destination node. This method leads to a search pattern that can be visualized as an expanding circle around the source node that keeps expanding in all directions until the destination node is encountered. The running time of Dijkstra’s algorithm for implementation that extracts the minimum edge via a linear search through the collection of edges is  $O(V \cdot E)$  in which  $V$  is the set of nodes in the current graph and  $E$  the set of edges.

```

01 function Dijkstra (G, s)
02   for each node v in V[G] do // Iterate through each node in the set V[G].
03     d[v] := ∞ // Set the weight of the current node d to infinity.
04     previous[v] := null // Set the previous node on the shortest path to null.
05   done
06   d[s] := 0 // Set the weight of the source node d to zero.
07   S := ∅ // Create an empty set for the visited nodes.
08   Q := V[G] // Fill the set of nodes to visit with nodes from the set V[G].

```

```

09   while Q ≠ ∅ do           // Iterate through Q until the set is empty.
10       u := min(Q)         // Extract with the lowest weight from the set Q.
11       S := S ∪ {u}       // Add the extracted node to the set of visited nodes.
12       for each edge (u,v) where v ≠ u do // Iterate through each outgoing edge of u.
13           if d[u] + w(u,v) < d[v] then // Determine if current route to v via u is shorter.
14               d[v] := d[u] + w(u,v) // Relax the weight of node v.
15               previous[v] := u      // Set the previous node on the shortest path to v as u.
16           end if
17       done
18   done

```

Figure 8: Dijkstra's Algorithm in pseudo code

The pseudo code of Dijkstra's algorithm presented in Figure 8 shows the algorithmic steps required to find the shortest path between the source node  $s$  and all other nodes in the graph  $G$ . The lines 02 through 05 represent the initialization process of the Dijkstra's algorithm. Here the weight assigned to each node is set to infinity signaling that the distance between the current source node and the destination node is unknown. The previous node on the shortest path is set to null to signal that there is no shortest path from the current node to the source node. The initialization process is executed to erase any possible previous searches conducted on the same graph.

The weight of the source node is then set to zero on line 06. This is done to guarantee that the source node will be the first node evaluated by Dijkstra's algorithm since its weight is zero while the weights of all other nodes are set to infinity. The set constructed on line 07 represents the set of visited nodes. This set  $S$  is initially constructed empty but contains all nodes once the algorithm is finished. The second set  $Q$  used by Dijkstra's algorithm is constructed containing all nodes in the graph  $G$  once Dijkstra's algorithm is finished this set is empty.

The actual path finding process commences on line 09, here a loop is initialized that executes as long as there are nodes in the set  $Q$ . The node with minimum (lowest) assigned weight is extracted and removed from the set  $Q$  and stored in  $u$ . The extracted node  $u$  represents the node that is closest to the source node at current. The first node extracted by Dijkstra's algorithm is the source node  $s$  since the weight assigned to this node is less in comparison to all other nodes in the graph  $G$ . The extracted node  $u$  is added to the set of visited nodes  $S$  on line 11. Once a node is added to the set  $S$  the shortest path between the source node and that node is determined, there is no other path leading to that node which is shorter than the one currently found.

Dijkstra's algorithm then enters a phase known as edge relaxation – lines 12 through 17. Each edge outgoing from the currently selected node  $u$  is evaluated and the node  $v$  connected to the other end of the edge is selected. If the selected node  $v$  is the same as node  $u$  indicating a circular edge path the edge is ignored. The weight of the selected node  $v$  is relaxed if the weight assigned to the node  $d[u]$  and the cost of traveling via the edge between these nodes  $w(u, v)$  is less than the weight currently stored in node  $d[v]$ . If this is true, the shortest path from the source node to node  $v$  leads via node  $u$ , otherwise the path currently found is sub-optimal. If the shortest path is extended through node  $v$  the weight of node  $d[v]$  is modified as shown on line 14 and the previous node on the shortest path is set to node  $u$ . The process of edge relaxation continues until the set  $Q$  is empty.

```

01 function Dijkstra (G, s, d) // The destination is represented by d.
...
11     if u is d // If the extracted minimum node u is d.
12         return // End the algorithm.
13     end if
...

```

Figure 9: Dijkstra's Algorithm pseudo code extensions

Searching a graph for the shortest path between a specified source node  $s$  and all other nodes in the graph can be quite time consuming and unnecessary. Dijkstra's algorithm can therefore be modified to find the shortest path between a specified source node  $s$  and a destination node  $d$ , modifications required are shown in Figure 9.

Line 01 needs to be altered to allow for specification of the destination node  $d$ . Line 11 through 13 need to be inserted in-between lines 10 and 11 of the pseudo code listed in Figure 8. Once Dijkstra's algorithm extracts the minimum node  $u$  that equals node  $d$  from the set  $Q$  it is known that the shortest path from the source node  $s$  to node  $u$  is found. The line 11 in Figure 9 evaluates to true and the search is ended.

```

01   P := ∅ // Create an empty set to store the path.
02   n := d // Set n to the destination node.
03   while previous[n] ≠ null do // While there are node along the path do.
04       P := {n} ∪ P // Add the current node stored in n to the beginning of the set P.
05       n := previous[n] // Move to the previous node on the shortest path.
06   done

```

Figure 10: Pseudo code for reversing the shortest path

The chain of nodes representing the shortest path between the source node  $s$  and the destination node  $d$  runs from node  $d$  to node  $s$ . This path needs to be reversed via the pseudo code listed in Figure 10 in order to be presented to the driver as the shortest path to the destination. Line 01 constructs an empty set  $P$  that is used to store the shortest path. Line 02 creates a variable  $n$  used to reverse the path found; it is initialized by assigning the destination node  $d$  to it. The reversal process is listed in lines 03 through 06. The reversal process loops through all nodes of the shortest path until there are no more nodes on the path meaning that the current node stored in  $n$  is the source node  $s$ . The node  $n$  is added to the beginning of the set  $P$  in line 09 and the node stored as  $previous[n]$  in node  $n$  is assigned to node  $n$ .

## 2.2 Dynamic Routing Algorithms

Dynamic routing algorithms – a subset of the routing algorithms field – are those algorithms that are able to react to ever-changing conditions in the graph on which they operate. Dynamic routing algorithms operate on graph for which the edge weights are volatile. The clearest example of such a graph is the Internet. The Internet consists out of a series of interconnected routers – the nodes – and communication lines – the edges – that are used to exchange data between them. The topology of the Internet is prone to changes without notice. The number of routers on the network can change at any given point in time due to new additions, removals or failures. However, data exchange should not suffer from these alterations. Therefore, dynamic routing algorithms are used to maintain the path of communication between routers. While dynamic routing algorithms have been developed for other uses than the Internet this chapter focuses on those developed for packet switched networks; the type of network for which the Ant colony based algorithm was intentionally designed.

### 2.2.1 Routing Information Protocol

The routing information protocol (RIP) [8] is the most well known member of the set of algorithms known as distance-vector routing algorithms. The RIP allows routers – a device to guide data packets – within a packet switched communication network to react in dynamic manner to changes within the network topology. Distance-vector routing algorithms use the Bellman-Ford algorithm, described in section 2.2.1, to construct and maintain an up-to-date representation of the network within a router in order to allow the router to guide data-packets along the most optimal route to a specific destination network. The most optimal route between two networks is defined as the route that requires the lowest amount of time when travelling between two networks. In order to determine which route requires the least amount of time the RIP maintains a routing table at each router. This routing table contains among others the following information:

- The network address of each other router within the network.
- The router that lies next on the shortest path towards another router within the network.
- A metric value that indicates the cost of transmitting a data-packet to a router within the network.
- Timestamp value indicating the time at which the last route update message was received.

The RIP protocol works by storing a graph representation of the known network topology within the routers memory and providing this as input for the Bellman-Ford algorithm in order to determine the shortest paths. The routers within the network form the nodes of the graph, the communication links between the routers form the edges and the cost of transmitting a data-packet via the communication link represents the weight of the edge. The number of hops (intermediate routers) or the cumulative transmission speed of the links between two routers can be used to represent the cost of transmitting a data packet. Dynamic factors such as load, measured delay and reliability of a communication link can however not be used.

Packet switched networks, such as the Internet, are unstable. Routers and communication links can be added or removed (the removal can also be caused by failure of the device or connection) from the network without warning causing previously calculated routes to fail. Therefore, each router within an RIP network periodically shares (every 30 seconds by default) its complete routing table with its direct neighbor routers, i.e. those routers that can be reached without visiting other routers, within the network. Once an update is received, the router updates its internal graph of the network and executes the Bellman-Ford algorithm to compute the current set of shortest paths.

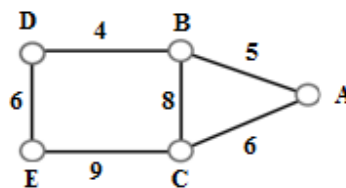


Figure 11: A network graph

To illustrate how RIP operates a description is given for the developments of shortest paths within the routing table of node A as shown in Figure 11. We assume that router A is properly configured and aware of the network topology of the network. Table 2 shows the development of the routing entries at three different time steps. The routing table of node A initially contains only entries for the destination routers {B, C, D, E} and its direct neighbors {B, C} without shortest path information. During the first update ( $t = 30$ ), the routers make their existence known to their direct neighbors by broadcasting a small data packet. Node A receives packets from its direct neighbors B and C and executes the Bellman-Ford algorithm to reveal the shortest paths. During the next update interval ( $t = 60$ ), the routing table of node B is received first and metrics for the node D, a direct neighbor of B, is added to the routing table. Then the routing table of node C is received and metrics for node E are added to the routing table. During the last update interval discussed  $t = 90$  the entire network state is revealed to node A. Updates continue to arrive at node A during each next update interval, allowing node A to keep its routing table up-to-date.

Table 2: RIP routing table development over time.

$t = 30$	Via B	Via C
To B	5	14
To C	13	6
To D	-	-
To E	-	-

$t = 60$	Via B	Via C
To B	5	14
To C	13	6
To D	9	-
To E	-	15

$t = 90$	Via B	Via C
To B	5	14
To C	13	6
To D	9	18
To E	15	15

The downside of RIP and distance-vector routing algorithms in general is that by using the Bellman-Ford algorithm changes are slowly propagated through the network and data-exchange requirements to keep the algorithm up-to-date are computationally intensive [9]. Another problem with the RIP protocol is that a router is cannot determine if it is on a certain route when receiving routing table information from a neighboring node. This problem can lead to the count to infinity problem in which loops in the network routes prevent packages from being routed towards their destination. Furthermore, the number of routers within a certain network cannot exceed 15, a design limitation imposed on RIP. These downsides of distance-vector routing algorithms have lead to the development of a new class of routing algorithms known as link-state routing algorithms of which the Open shortest path first algorithm is discussed in the following section.

### 2.2.2 Open Shortest Path First

The Open Shortest Path First (OSPF) [11] routing algorithm is a member of the Link-state routing protocols [10] class of routing algorithms. OSPF was designed to replace RIP for routing on the internet and therefore remedies problems associated with RIP and distance-vector routing algorithms. OSPF routers use Dijkstra's algorithm described in section 2.1.3 to calculate the shortest paths between two routers in the network. Each router maintains a routing table or routing database containing the following information:

- The network address of each other router within the network.
- The router that lies next on the shortest path towards another router within the network.
- The link age that represents time passed since the link to the router was inserted into the database.
- A checksum associated with each router to verify proper data transmission of information associate with that router.
- The number links available on each router.

The router also maintains a table of costs for each communication link attached to the router. The cost maintained by these routers represents the bandwidth of each communication link. The OSPF router works by broadcasting a 'hello' data-packet to all other routers within its network. The 'hello' packet informs other routers within the current network that router is alive and operating. The 'hello' packet is broadcasted every 10 seconds by all routers. Routers that are connected to each other via a direct link, i.e. a communication link with no intermediate routers in-between form an adjacency. Once an adjacency is formed the routers exchange information concerning the links stored in their database. After a router has processed the contents of the other routers database, it can request link state information from an adjacent router to update its own routing table metrics. The router then calculates the shortest path by running Dijkstra's algorithm on a graph representation of its routing database. OSPF routers their adjacent routers of any newly received information by sending out link state information. Paranoia updates are executed every thirty minutes. During these paranoia updates, adjacent OSPF routers exchange the entire contents of their routing table to ensure that all link state information is up-to-date.

OSPF is a hierarchical routing algorithm that is able to divide a single network into a hierarchy of sub-networks. Routers that reside on the periphery of such a sub-network and have active communication links to other sub-networks are named border routers. These border routers use and maintain multiple routing databases, one for each sub-network they are connected to. The exchange of data between sub-networks is done via a backbone network to which all sub-networks must be connected. This backbone network consists out of series routers within the network that are tasked with distributing routing information between sub-networks.

While the link-state routing algorithms such as OSPF do not suffer from the problems of distance-vector routing algorithms such as high data exchange requirements, loops within routes and a restricted number of routers within a network it is deemed complex. The complexity of OSPF can lead to poor implementations within networks reducing the effectiveness of OSPF. The information stored in comparison to RIP is large requiring more memory present in routers and link-state updates can be costly in terms of router CPU time when number of link state changes if large. However, OSPF is not able to react in a dynamically to real-time changes within the network such as link loads and utilization of router capacity. Another class of routing algorithms was developed that took inspiration from the behavior of ants in order to solve this problem. The next section discusses a routing algorithm of this class named the Ant colony based algorithm.

### 2.2.3 Ant Colony Based Algorithm

The Ant-Colony meta-heuristic [12] [13] was originally developed for finding shortest paths in packet-switched communication networks. The meta-heuristic was deduced from the behavior that Argentinean ants exhibit when a source of food is located. Ants travelling back from a food source to their colony deposit a natural pheromone – a chemical substance that is used to exchange messages between ants – that marks the trail they have taken. At each obstacle encountered on the route back to the colony the ants make a decision about which path to follow next. Since ants do not rely on visual information for path finding every choice the ant makes when diverting an obstacle is randomly chosen.



Figure 12: Ant pheromone laying behavior

Figure 12 displays the strengthening process of pheromone tracks in four steps. The first image in the series shows ants that encounter a small puddle of water prohibiting them from walking straight on. The ants are forced to make a decision to go either left or right round the puddle, due the random behavior in the path finding process approximately fifty percent will choose to go right, the others choose to head left. The ants that choose to go via the right (bottom) route will travel a shorter distance than the ants that go via the left (top) route. This means that the ants travelling via the bottom route will reach the other side of the puddle faster thus leaving a pheromone trail that indicates to other ants that a route to the food source goes via that path. In the remaining time, before the ant that travels via the top route reaches the other side of the puddle, ants will be enticed to travel via the bottom route due to the pheromone track there. When the ants traveling via the top route reach the other side of the puddle a transition period will take place between the two routes due to both now having a pheromone track. However, on average more ants will travel via bottom route since the concentration of pheromone there is higher than that on the top route as is shown in the last image of the series. The pheromone on the top route eventually evaporates leaving only the bottom route available for the ants. Eventually when the food source is consumed, ants stop travelling on the route altogether letting the pheromone track evaporate and freeing the ground for new routes to other food sources.

Table 3: ABC Routing Table

Destination address	Neighbor 1	Neighbor 2	Average delay
$d$	$P_{d,n1}$	$P_{d,n2}$	$\mu_d$

The ant colony based (ACB) algorithm has translated the natural behavior of ants to one applicable to packet switched networks. Each router within the network maintains a routing table as shown in Table 3. The routing table contains the addresses of all other routers  $d$  within the network, the routers that are direct neighbors of the current router  $n$ , a probability  $P_{d,n}$  that is used as a metric to determine which neighboring router of the current is most optimal when travelling towards the destination and the average delay encountered on the route towards the destination. The probability value for a given neighbor and destination is determined via evaluation of real-time network information through the means of two types of intelligent agents called ants.

The forward-ant, as described in section 2.2.3.1, attempts to locate a route towards a destination router. The forward-ant does this by evaluating probabilities and real-time link information at each intermediate router on the path towards the destination and visiting each router with highest compounded probability. For each visited router the forward ant maintains a record in its memory containing the address of the router and a compound value based on real-time information concerning the communication link between the current and next node on the route. This real-time information consists out of the bandwidth, transmission queue length and propagation delay of the communication link.

The backward-ant, as described in section 2.2.3.2, uses the information gathered by the forward-ant to retrace the route between the destination router and the source router. At each intermediate router including the source router, the backward-ant uses the compounded real-time delay information to modify the probability for the destination router. The probability of the neighboring router that lies on the path towards the destination is increased while probabilities of other neighbors are decreased. This process strengthens the pheromone scent of routers along the optimal (shortest) paths while decreasing pheromone scent of routers along sub-optimal paths. Routers then use these updated probabilities in their routing tables to guide normal data-packets between destinations.

The ACB algorithm is able to find shortest path in packet switched communication networks by incorporating a metric based upon real-time network information. The routes found by the ACB algorithm consider more information than the distance-vector routing algorithms and the link-state algorithms. However, the reliance of the ACB algorithm upon ants makes it sensitive to packet loss and inadequate for large networks due to its reliance on real-time information. Chapter three describes an proposed version of the ACB algorithm that is able to remedy these problems by introducing a hierarchy.

### 2.2.3.1 Forward-ant

The forward-ant attempts to discover the shortest route between a given source router and a destination router in a packet switched communication network. Each router within the network periodically transmits a forward-ant to every destination router in its routing table. At each intermediate router between the source router  $s$  and the destination router  $d$  the forward-ant determines the most optimal next router on the route towards the destination. Equation 3 is used to determine the most optimal next router by calculating a temporary probability  $P'_{dn}$  value for each neighbor from the set  $N_i$  of the current node that consists out of the probability  $P_{dn}$  for the neighbor  $n$  and a normalized value  $l_n$  that is proportional to number of bytes in queue for the communication link toward that neighbor. The router associated with the highest  $P'_{dn}$  value is chosen as the next router on the route towards the destination.

$$P'_{dn} = \frac{P_{dn} + \alpha \cdot l_n}{1 + \alpha \cdot (|N_i - 1|)}, \alpha = 0.4 \in [0,1]$$

Equation 3: The probability of the route via node n

It is not guaranteed that the forward-ant will reach its destination since events such as packet loss, path loops and communication link failure among others can prevent the ant from reaching its destination. Therefore, the forward-ant cannot update routing tables with new probabilities upon arrival at a router but instead needs to memorize the metric for the backward-ant. Once the forward-ant has chosen the next router on the path towards the destination, it calculates an estimated delay for the communication link towards router  $n$  by using real-time information. Equation 4 is applied to determine the communication link delay  $T_{in}$  between the current router  $i$  and the chosen neighbor  $n$ . The size of the transmission queue  $q_n$  in bits for the communication link and the size of the forward ant  $F_{sd}$  in bits are divided by the bandwidth  $B_{in}$  of the communication link. This value is increment with the propagation delay  $D_{in}$  of the communication link to form the final estimated delay for the communication link. The estimated delay and the address of the current router  $i$  are stored into memory by the ant.

$$T_{in} = \frac{q_n + \text{Size}(F_{sd})}{B_{in}} + D_{in}$$

Equation 4: The delay estimated for a communication link

Table 4 shows the contents of a forward-ant that has traveled between two routers. Apart from the routing information used to guide the ant towards its destination it also contains the route traveled between the source and destination router. The route consists out of the address of each router encountered on the route towards the destination router and the estimated delay for the communication links the ants has traveled along. The forward-ant having reached its destination is transformed into backward-ant in order to retrace the route followed.

Table 4: Ant data packet layout

<b>Destination Address:</b>	192.168.0.4	
<b>Source Address:</b>	192.168.0.7	
<b>Previous Address:</b>	192.168.0.5	
<b>Route:</b>	192.168.0.7	9
	192.168.0.6	10
	192.168.0.5	14

The RIP suffered from the count to infinity problem because routers were unable to tell whether or not they were part of certain route. To prevent this from happening ACB employs the forward-ant. The forward-ant verifies at each intermediate node that the id of the router encountered is not already stored in its memory. If the address of the router is found in its memory, a loop is detected. The forward-ant may remove all entries upon to the duplicate entry if the number of entries is smaller than half of the total amount of entries. The forward-ant destroys itself if the number of entries is greater than half of the total amount of entries.



### 2.2.3.2 Backward-Ant

Once the forward-ant reaches the destination router, an optimal route between source and destination router is known to exist. This implies not only that the route between source and destination router is optimal but also that the route from each intermediate router towards the destination router is optimal. The probabilities stored for the destination at the routers along the path can now be altered to reflect the information gathered by the forward-ant. Since the forward-ant stored its entire route in memory the backward-ant is able to retrace the path taken by the forward-ant towards the source router and update the probabilities within the routing tables.

**Table 5: Updated probabilities at intermediate router  $i$**

Updated Probability	Virtual Delay
$P_{k+1,k+1}$	$T_{i,k}$
$P_{k+2,k+1}$	$T_{i,k+1}$
$P_{k+3,k+1}$	$T_{i,k+2}$
...	...
$P_{k+n,k+1}$	$T_{i,k+(n-1)}$

Backward-ants in the process of retracing the route taken by their forward-ant are required to update not only the probability for the route from the current router  $i$  to the destination router  $d$  but also the probabilities of all sub-routes to intermediate routers. Table 5 shows the probabilities that are updated at each intermediate router  $i$  when the route of the backward-ant is  $n$  routers long. The virtual delay for each (sub-)route  $T_{id}$  is calculated using a simple summation of the virtual delays  $T_{jk}$  stored in the backward-ants memory, as shown in Equation 5.

$$T_{id} = \sum T_{jk}, \text{ where } j, k \in i \rightarrow d$$

**Equation 5: The virtual delay experienced on the route  $i \rightarrow d$**

The virtual delay  $T_{id}$  is used to alter the average delay  $\mu_d$  for the destination of the currently selected route. The average delay for destination allows the ACB algorithm to determine how optimal a given route is and adjust probabilities accordingly. The average delay is calculated via Equation 6 with  $\eta$  determining the influence of single measurement upon the average.

$$\mu_d = \mu_d + \eta(T_{id} - \mu_d), \eta = 0.1 \in [0,1]$$

**Equation 6: The average delay for the route  $i \rightarrow d$**

As shown in Equation 7, the virtual delay  $T_{id}$  for the current route is divided by average delay  $\mu_d$  multiplied with a scaling factor  $c$  to determine the strength of the reinforcement  $\lambda$  given to the probability  $P_{dn}$ . If the virtual delay for the route between router  $i$  and  $d$  is less then the average delay the probability eventually receives a positive increment, otherwise the probability is left unaltered.

$$\lambda = \begin{cases} \frac{T_{id}}{c \cdot \mu_d}, & \frac{T_{id}}{c \cdot \mu_d} < 1, c = 1.1 \in [1,2] \\ 1, & \frac{T_{id}}{c \cdot \mu_d} \geq 1 \end{cases}$$

**Equation 7: The probability reinforcement strength**

The probability updating process alters the probabilities for all neighbors of the current router  $i$  for the given destination  $d$  as is shown in Equation 8. The probability for the neighbor  $n$  of the current router that lies on the optimal path receives a positive stimulus while all other neighbors  $j$  receive a negative stimulus. All probabilities are then normalized so that the cumulative result is one.

$$P'_{dn} = P_{dn} + (1 - \lambda)(1 - P_{dn})$$

$$P'_{dj} = P_{dj} - (1 - \lambda) \cdot P_{dj}$$

**Equation 8: Probability updating**

Once the backward-ant reaches the source node  $s$  the ant is destroyed and the path between the source router and the destination router is updated to reflect the current network state. This process guarantees that all routes are evaluated at each interval and reflect the current state of the network. The downside of the ACB algorithm is scalability. As discussed extensively in chapter three the longer the path the backward-ant and forward-ant have the travel to less likely it is that the information stored in the ant reflects the current state of the network leading to the formation of sub-optimal paths. The following section discusses how current navigation systems make use of static and dynamic routing algorithms to calculate optimal routes.

## 2.3 Navigational Systems

This section presents an overview of the dynamic traffic information services offered for use with current day navigational systems. The power and capabilities of navigational software devices have increased significantly in the last few years. Major players on the car traffic navigational systems market like TomTom and Mio are constantly inventing new solutions to route vehicles to and from their destination via an optimal path, either in time or in travel distance. The focus of navigational systems is shifting from static routing algorithms to dynamic routing algorithms that are able to incorporate current traffic conditions. However, static routing for vehicles is still the most common phenomena among vehicle navigational systems. These systems rely on the roadmap information bought by the user to calculate the most optimal routes. There are however two supplement subscription services that convert these static routing algorithms to dynamic routing algorithms. The first of these services is the so called Radio Data System - Traffic Message Channel (RDS-TMC) and the second is TomTom plus traffic, a service only available to TomTom customers.

### 2.3.1 Radio Data System - Traffic Message Channel

RDS-TMC [13] is a system that adds data to existing radio frequencies in such a fashion that it does not disrupt radio transmissions on those frequencies. The most common application of RDS today is noticeable when tuning the car stereo to a certain FM frequency and the name of broadcaster is displayed in combination with song titles and even commercials. RDS-TMC allows for the transmission of traffic condition information via an FM frequency or via the language neutral DATEX protocol.

The downside of the RDS-TMC is that the amount of data that can be transmitted is limited, the maximum transmission frequency is 650 bits/s that must be shared with all services using RDS to transmit their data. The company TMC4U – a collaboration between Siemens and the Royal Dutch Touring Club (AWNB) – supplies the RDS-TMC traffic condition information in the Netherlands. The information provided by TMC4U only informs about traffic conditions on major highways and secondary roads. Local information is not yet available and will be offered as a fee-based subscription service in the future. RDS-TMC was seen as an important technology by the EU but since a directive obligating providers to transmit traffic condition information failed to get enough support the effort concerning RDS-TMC seems to have died down.

All major navigational system manufacturers provide the hardware needed in order to receive RDS-TMC. While there is at current no subscription fee for the service offered the routing is only altered for routes via highways. This in itself causes problems since in order to divert from highway traffic jams the routing algorithm can only choose between two options. The first option is to take another highway to the destination if that highway is less congested than the one the driver is currently headed for. The second option is to take a highway exit and drive via another road towards the destination. Since the system does not provide local information the route then suggested might in practice be heavily congested resulting in a route that is far worse than the traffic jam on the highway. The second option therefore remains a gamble to driver.

### 2.3.2 TomTom plus Traffic

TomTom offers a subscription service named 'TomTom plus traffic' that is able to provide up-to-date traffic condition information for roads [14]. The backbone of this system is the mobile telephone infrastructure in the country where the service is offered – for the case of the Netherlands this infrastructure is provided by Vodafone. The system gathers information by triangulating GSM telephone signals in order to determine the position of vehicles on certain highways. The accuracy of triangulation [14] ranges between 50 and 200 meters when only GSM technology is used, with the assistance of GPS the accuracy can be increased to between 5 and 30 meters. Once the position and heading of the vehicle are determined the system searches through its database of roads to find the one the vehicle is currently driving. Then based on frequent interpretation of the GSM telephone signals the system is able to determine the journey time and speed of the vehicle. Thus when monitoring a sufficiently large volume of GSM signals the system is able to determine where traffic jams occur.



Figure 13: TomTom Navigation Device interface

Subscribers to the TomTom plus service need to pay a subscription fee to TomTom that is about 30 euro's per year for the Dutch road system at present. Beside the subscription fee the driver should also possess a mobile GSM telephone capable of Bluetooth communication to interface with the TomTom navigational system and GPRS communication for downloading the traffic condition information from the TomTom servers. The subscriber can then configure the TomTom navigational system to update information only when feeding the route to the system or at certain intervals while driving to the destination.

The TomTom plus traffic service however seems to have a number of deficiencies [15] [16]. The lack of a sufficient amount of data concerning traffic jams is the most encountered complaint leading to situations where a driver is redirected to avoid one traffic jam only to be rerouted into another traffic jam that was not registered with TomTom's system. The requirement for GPRS communication between the GSM and TomTom also is problematic since not all GSM network operators have enabled GPRS on all their base stations leading to situations where the system is unable to download up-to-date information to the device or stops mid-process. The last complaint concerns the costs associated with the downloading of traffic condition information. While these costs are charged by GSM network operators the impact on the acceptability of a broad range of the consumers still remains questionable.



## Distributed Hierarchical Routing

This chapter describes the distributed hierarchical swarm-based routing algorithm used for routing participants of the CBPRS to and from their destinations. Where the previous chapter discussed a variety of dynamic and static routing algorithms this chapter fully focuses on the ant-based algorithm designed for the CBPRS. The algorithm collects traffic condition information consisting out of location and delay information from vehicles driving through the city in order to calculate dynamic shortest time routes through the city. The route calculation process is executed via a set of intelligent agents called ants that travel through a representation of the network to find shortest time paths. Many types of ant-based routing algorithms have been developed in the past, however most focus on packet switched communication networks [12] [17] [18]. The ant-based algorithms presented in [19] and [20] are designed for routing within traffic networks, these form the basis of the algorithm presented in this chapter. The ideas presented in [20] are extended in this chapter to allow for fully distributed hierarchical routing.

The network model, in section 3.1, describes the creation of a fully distributed hierarchical network model based on intelligent lampposts. Section 3.2 describes how traffic condition information is gathered from vehicles and used to determine delays on roads. Section 3.3 describes how routes are found using intelligent agents called ants. Section 3.4 discusses possibilities and limitations of the current implementation of the ant-based algorithm.

### 3.1 Network Model

The ant-based routing algorithm as described in this chapter operates on the premises that by collecting local traffic condition information it is able to determine time optimal routes on a global scale. The collection of local traffic condition information needs to occur in real-time in order for the ant-based algorithm to have an accurate image of the current state of the traffic network. Vehicles driving through the road network therefore gather the traffic condition information required by the ant-based algorithm. While many different methods could be used to facilitate the routing service and exchange data between the vehicles and the CBPRS, we have opted for intelligent lampposts.



Figure 14: An intelligent lamppost at an intersection

Intelligent lampposts are modified lampposts that are able to communicate with vehicles via short-range wireless communication and with other intelligent lamppost within the city via wired power-line communication. The intelligent lampposts are placed at each intersection that falls within the CBPRS zone of operation. Each intelligent lamppost contains a small computing device that enables the intelligent agents called ants – described in section 3.3 – of the ant-based algorithm to maintain routing tables containing information concerning the optimality of a given neighboring intelligent lamppost for a given destination. This network model allows ant-based algorithm to work in fully distributed manner without the need for central coordination of the algorithm. The distribution of information collection and processing services also makes the CBPRS routing service less vulnerable to sabotage or system failures that could occur when all services are combined into a single set of central servers. If in the current network model one intelligent lamppost fails, other intelligent lampposts can still perform their task without failure although some routes through the city might become less optimal than possible.

### 3.1.1 Hierarchy of Sectors

The classical AntNet [12] and ABC [17] algorithms exhibit scalability problems when the number of nodes in the network increases. As described in [18] the AntNet and ABC algorithms both have a direct relationship between the number of nodes and the number of ants in the network. Both algorithms require that every node in the network sends – at a certain user defined time interval – an ant to every other node in the network. The algorithms therefore require a total number of ants equal to  $N * (N - 1)$  where  $N$  represents the number of nodes in the network. The scalability is also influenced by the inverse relationship between the time taken by an ant to complete its journey and value of the data stored in the ant. An increase in the number of nodes causes ants to have longer travel times between some nodes, leading to situations where the data stored in the ant does not reflect the current state of the road network that in turn could lead to suboptimal paths. Solutions presented in [18] and [20] that solve the scalability problems of AntNet and ABC suggest that the network of nodes should be divided into a number of smaller sectors between which communication via ants is allowed although in a restricted manner. These solutions reduce the number of ants in the network and the length of the paths taken by the ant thereby eliminating any scalability problems.

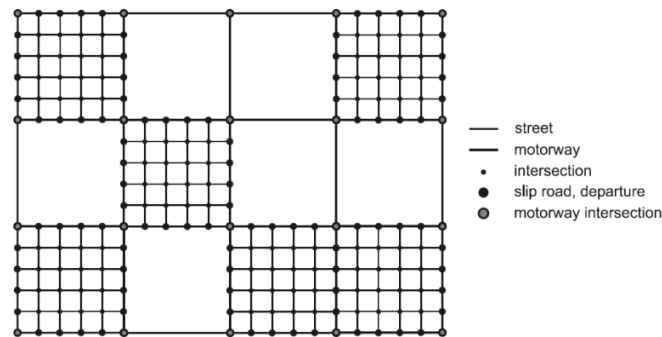


Figure 15: Simplistic representation of a city road network

The road network within a city, such as shown in Figure 15, can be described via an analogy with the human cardiovascular system in which a network of veins and arteries is responsible for distributing blood to the different organs of the human body. The arteries within the human body transport large amount blood from the heart and long to the liver, brain, etc. Once the blood arrives at the organs smaller veins transport the blood to the place where oxygen and nutrients – among others – are extracted. The road network on the other hand consists out of a set of arterial roads (arteries) that enable vehicles to move quickly from one location in a city to another. Once the vehicle approaches the destination it uses a number of streets (veins) the reach it final destination. These arterial roads effectively segment the road network into a series of sectors that contain a number of streets, as the sectors formed in Figure 15 by the motorways clearly shows. These ‘natural’ sectors can be found in every modern city when studying overview maps.

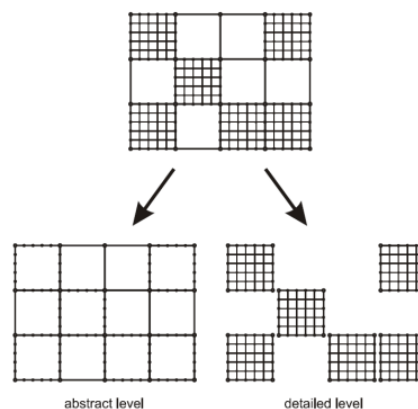


Figure 16: Road network hierarchy

The ant-based routing algorithm of the CBPRS employs these ‘natural’ sectors within a city to construct a two level hierarchy, as shown in Figure 16. The construction process used by the ant-based algorithm consists out three individual phases that determine the type and sector memberships of each individual lamppost. The first phase constructs the abstract hierarchical level which level consists out of all intelligent lampposts that are placed on an intersection to which an arterial road is connected. Roads are treated by algorithm as an arterial road if they are of the type road, main road, arterial road, highway or freeway. Each intelligent lamppost that is found to be on the abstract level of the hierarchy is marked as global routing node. Global routing nodes are responsible for routing traffic between sectors within cities. The second phase constructs the detailed level of the hierarchy by forming sectors with the road network. By, temporarily, subtracting the abstract hierarchical level roads and intersections from the road network cohesive sets of intelligent lampposts materialize that can be grouped into sectors. Sector numbers are assigned to these intelligent lampposts and they are marked as local routing nodes. The third and final step determines the sector number of the global routing nodes by reviewing their neighboring local routing nodes and adding them to their sectors. After these three steps have been completed every intelligent lamppost marked as local routing node is member of one sector, every global routing node is a member of one or more sectors within the network. The sector numbers present on the detailed level of the hierarchy are treated as virtual nodes on the abstract hierarchical level.

### 3.1.2 Routing tables

Once the network of intelligent lamppost is present within a city and the network hierarchy established according to the method described in the previous section routing tables can be determined at the individual intelligent lampposts. The intelligent lamppost within the network use routing tables to store information that lists which neighboring intelligent lamppost is optimal for a given destination. The ant-based algorithm of the CBPRS uses local and global routing tables to route vehicles on the detailed and abstract level. The local and global routing tables have the same basic layout, as is shown in Table 6. The routing table contains an entry for each destination  $i$  known to the intelligent lamppost. This entry consists out of the address of the destination  $D_i$ , a probability value for each neighbor  $P_{nn \rightarrow i}$  that represents the strength pheromone trail for the current destination via the neighbor  $N_n$  and a average delay  $\mu_i$  representing the average time required to travel from the current intelligent lamppost towards the destination.

**Table 6: Node routing table basic layout**

Destinations	Neighbor 1	...	Neighbor n	Average Delay
$D_i$	$P_{n1 \rightarrow i}$		$P_{nn \rightarrow i}$	$\mu_i$

The local routing table is used route vehicles between intelligent lampposts within sectors on the detailed level of the network hierarchy. Each intelligent lamppost within the network has at least one local routing table, an example is show in Table 7. Intelligent lampposts that are designated as global routing nodes have one local routing table for each sector they are a member of. The local routing table contains a destination entry for each intelligent lamppost within the sector. The set of neighbors of an intelligent lamppost is determined by the layout of the road network within the city and the network hierarchy. Only intelligent lampposts that are connected to the intelligent lamppost via a single road and are member of the same sector are listed as neighbors.

**Table 7: Local routing table layout example**

	N1	N2	N3	Average delay
1	0,3	0,5	0,2	1m34s
2	0,25	0,4	0,35	53s

Each intelligent lamppost within the network has one global routing table although the manner in which the global routing table is interpreted differs between types of intelligent lampposts. Each intelligent lamppost whether on the abstract, detailed or both levels has a single global routing table. The global routing table contains a destination entry for each virtual node within the network hierarchy, an example is shown in Table 8. If the global routing table resides on intelligent lamppost that functions as a local routing node the probability values kept for every virtual node destination represent routes to the most optimal global routing node for the given virtual node destination that is a member of the same sector as the local routing node. The average delay kept for these destinations is a summation of the average delay required to travel from the local routing node to the global routing node plus the average delay stored for the virtual node in the global routing node.

The set of neighbors in this case is determined in same fashion as for the local routing table. If the global routing table resides on a intelligent lamppost that functions as a global routing node the probabilities for a node virtual node destination represent the actual path towards that virtual node via the abstract hierarchical level. The set of neighbors in this case consist out the neighboring intersections that reside on the abstract hierarchical level.

**Table 8: Global routing table layout example**

	N1	N2	Average delay
V1	0,3	0,7	5m48s
V2	0,6	0,4	4m03s

The following to sections describe how the network hierarchy affects the manner in which the ant-based algorithm operates and explains how the algorithm acquires the necessary information to find and maintain optimal routes and alters the routing tables.

### 3.2 Traffic Condition Information

The network model described in the previous section describes how the intelligent lampposts cooperate to form a network of routing nodes throughout the city. This interconnected network of intelligent lampposts supplemented with information concerning the length and maximum speed of all roads within the CBPRS zone of operation would allow the CBPRS to route vehicles in static manner. To route vehicles in a dynamic manner through the city dynamic traffic condition information is required, this section discusses how this information is gathered and stored by the intelligent lampposts.

Drivers travelling through a city constantly gathered precious traffic condition information concerning the roads they travelled. Under normal circumstances, the traffic condition information gathered remains with the driver and does not benefit other drivers within the network. The CBPRS participants share the traffic condition information they gather with the CBPRS in order to enable to provide optimal routes through the cities road network. The participating vehicles are all equipped with a navigational device capable of determining the position of the vehicle and a wireless communication device that enables the vehicle to share and receive information with the intelligent lampposts at the intersections. After each communication with an intelligent lamppost, the vehicle automatically stores its location and initiates a timer before driving on to the next intelligent lamppost. Once the vehicle enters the transmission range of another intelligent lamppost, it communicates its previous and current position and the time required to traveled between these positions by means of the timer. The intelligent lamppost then knows the time it has taken a vehicle to travel between current and previous intelligent lamppost.

$$M_r = M_r + \omega \cdot (D_r - M_r), \omega = 0.3 \in [0,1]$$

**Equation 9: Delay measurement for road r**

The intelligent lamppost adds this newly received information to an average delay for the road  $r$  the vehicle has just travel on using Equation 9.  $M_r$  represents the weighted mean of the delays reported to the lamppost. The weight  $\omega$  determines the influence that a newly reported delay  $D_r$  has on the measurement and how long such a reported delay influences this measurement. The value of  $\omega = 0.3$  indicates that the latest  $\pm 17$  delays reported by vehicles influence this measurement. The number of weights that really influence the measurement can be approximated via  $\approx 5(1/\omega)$  [12]. This relatively small amount of delays that influence the average delay allows the routing algorithm to react quickly to ever-changing traffic conditions. The intelligent lamppost stores this delay information in this manner for each road with incoming lanes on intersection it is monitoring.

### 3.3 Route Finding System

The previous section described how the intelligent lamppost translates traffic condition information received from vehicles into an expected delay for a certain road. While this information is required for proper dynamic behavior of the algorithm, it does not enable the CBPRS to calculate optimal dynamic routes at present.



The dynamic delay information gathered needs to be distributed over the individual intelligent lampposts in such a manner that optimal routes can be found on the abstract and detailed level of the network hierarchy. To achieve this goal the ant-based algorithm of the CBPRS uses intelligent agents called ants. The ant-based algorithm employs local ants to find and maintain routes that are optimal on the detailed level while global ants are used to find and maintain routes on the abstract level. The following two subsection discuss the types of local and global ants used and the manner in which the ants find and maintain optimal routes throughout the network using the traffic condition information gathered by vehicles.

### 3.3.1 Local Ants

The local ants are those that are restricted in their movement to the sector in which they originated. At present, the algorithm employ two types of local ants named the Forward ant and backward ant. The forward ant is used to find routes from a given origin intelligent lamppost to a destination which can either be another intelligent lamppost within the same sector or a virtual node representing another sector within the network. At each intermediate intelligent lamppost on the route toward the destination, the forward ant gathers the traffic condition information stored. Once at its destination the forward ant transfers the information it accumulated to a backward ant that retraces the route taken by the forward ant. Along the route, the backward ant adjusts the local routing table so that the route optimal route found by the forward ant is reflected.

**Table 9: Local ant memory layout**

<b>Destination Address</b>	192.168.0.4	
<b>Source Address</b>	192.168.0.7	
<b>Previous</b>	192.168.0.5	
<b>Path</b>	<b>Address</b>	<b>Average Delay</b>
	192.168.0.6	10
	192.168.0.5	9
	192.168.0.4	5

The forward and backward ants operate using identical memory layouts, as shown in Table 9. The first two fields are used to identify the destination and source of the local ants. The previous field stores the intelligent lamppost last visited, this field helps the forward and backward ants to determine which information to retrieve or modify at the current intelligent lampposts. The forward ant when travelling to a certain destination constructs the path field. The path field contains the address of each intermediate intelligent lamppost visited on a route towards a destination and the average delay recorded for the road that lies between a previous and current intelligent lamppost. The backward ant uses the average delay information stored in the path field by the backward ant to adjust routing tables and reflect optimal routes. The following sections describe the manner in which the forward and backward ants operate in detail.

#### 3.3.1.1 Forward ant

The ant-based algorithm of the CBPRS dynamic routing service uses forward ants to discover and maintain optimal paths within sectors. Forward ants are spawned – at user-defined intervals – with a destination that is a randomly chosen intelligent lamppost that resides in the same sector or a virtual node identifier of another sector within the network. The forward ant’s freedom of movement is restricted to sector it originated in. Therefore, routes towards virtual node destinations lead to the most optimal global routing node that is a member of the current sector. Abstract level routing services within the global routing node are responsible for determining the most optimal route toward the virtual node’s sector. The restricted freedom of movement of the forward ant enables it to carry larger payloads in terms of collected data since the since the number of intermediate intelligent lampposts on a route towards a destination is limited.

Forward ants locate the optimal route towards a destination by inspecting the probability tables at intelligent lampposts. Forward ants with a destination within the current sector inspect the local routing table for the current sector while those with a virtual node as destination inspect the global routing table stored in the intelligent lamppost. The forward ant determines the next intelligent lamppost on the path towards the destination by choosing the neighbor of the current intelligent lamppost with the highest probability value associated with the destination of the forward ant.

At every intelligent lamppost the forward ant encounters upon leaving the intelligent lamppost where it originated it collects traffic condition information. The forward ant uses the address of the previous intelligent lamppost to determine the road between the previous and current intelligent lamppost. Once the road is determined, the ant retrieves the average delay stored by the current intelligent lamppost for that specific road. The forward ant adds the address of the current intelligent lamppost and the average delay to its path field. If the forward ant visits a intelligent lamppost twice – thus creating a cycle – the path entries of the cycle are removed. If the detected cycle of visited intelligent lampposts is greater than half of the ant's current age – measured in the number of lampposts visited – the forward ant is destroyed.

The journey ends when the forward ant has reached its destination. If the forward ant's destination was an intelligent lamppost within the same sector, the journey ends at the intelligent lamppost. If, however, the forward ant's destination was a virtual node identifier the journey of the forward ant ends when it encounters a global routing node that is a member of the sector in which the forward ant originated. At the destination, the forward ant stores the final piece of delay information into its memory before transferring entire path to a newly created backward ant that retraces the path taken by the forward ant and adjusts routing tables. Once the forward ant's memory is transferred successfully, it destroys itself.

### 3.3.1.2 Backward Ant

The forward ant, discussed in the previous section, is an information collection agent that stores in memory information concerning the path taken towards the destination and the average of cost of that route when travelling via roads. The forward ant, however, cannot use this information to adjust routing tables since the route taken cannot be deemed optimal until the destination is reached. Therefore, the ant-based algorithm of the CBPRS routing service uses a backward ant that retraces the route followed by the forward ant. At each intelligent lamppost encountered when retracing the route the backward ant uses the average delay information collected by the forward ant to adjust the local routing tables for the current sector. The path taken by the forward ant between the source and destination intelligent lamppost is found by choosing the highest probability values for the destination of the forward ant at each intelligent lamppost encountered. This means that once the destination of the forward ant is reached the route between source and destination is optimal. This also means that the route between the destination and every intermediate intelligent lamppost along the path taken by the forward ant is optimal. Therefore, at each intelligent lamppost along the path the backward ant updates the probability values for the destination intelligent lamppost – the origin of the backward ant – and the intelligent lampposts between the current and destination intelligent lamppost.

$$T_{id} = \sum T_{jk}, \text{ where } j, k \in i \rightarrow d$$

Equation 10: The virtual delay on the route between node i and d

Upon arrival at an intelligent lamppost, the backward ant calculates the average delay  $T_{id}$  between the current intelligent lamppost  $i$  and the destination intelligent lamppost  $d$  by summing up the individual average delays  $T_{jk}$  for the roads between the intelligent lampposts on the path between  $i$  and  $d$  – using Equation 10. The average delay  $T_{id}$  between the current lamppost and the destination lamppost as determined by the ant is used to alter the average delay  $\mu_d$  for the route from the current intelligent lamppost towards the destination as stored by the intelligent lamppost. The average delay  $\mu_d$  is determined via Equation 11. The variable  $\eta$  is used as a weight to limit the influence of each delay on the average delay.

$$\mu_d = \mu_d + \eta(T_{id} - \mu_d), \eta = 0.1 \in [0,1]$$

Equation 11: The average delay toward node d

As shown in Equation 12, the virtual delay  $T_{id}$  for the current route is divided by average delay  $\mu_d$  multiplied with a scaling factor  $c$  to determine the strength of the reinforcement  $\lambda$  given to the probability  $P_{dn}$  where  $n$  represents the next intelligent lamppost on the path towards the destination intelligent lamppost. If the virtual delay for the route between intelligent lampposts  $i$  and  $d$  is less than the average delay the probability eventually receives a positive increment, otherwise the probability is left unaltered.

$$\lambda = \begin{cases} \frac{T_{id}}{c \cdot \mu_d}, & \frac{T_{id}}{c \cdot \mu_d} < 1, c = 1.1 \in [1,2] \\ 1, & \frac{T_{id}}{c \cdot \mu_d} \geq 1 \end{cases}$$

**Equation 12: Probability reinforcement strength**

Once the probability reinforcement strength is known, the backward ant can adjust the probability values for each valid neighbor  $V_n$  from the set of neighbors  $N_n$  of the current intelligent lamppost. The set of valid neighbors is a subset of the set of neighbors for a certain intelligent lamppost, the manner in which this set is determine is discussed in the following paragraph. The probability value  $P_{dn}$  for neighbor  $n$  on the shortest path towards to destination intelligent lamppost  $d$  receives a positive stimulus using Equation 13 where  $\alpha$  is used as a scaling factor to dampen oscillation caused by the probability updating process [18].

$$P_{dn} = \alpha \cdot (1 - \lambda)(1 - P_{dn}), \alpha = 0.1 \in [0,1]$$

**Equation 13: Probability adjustment for the optimal neighbor**

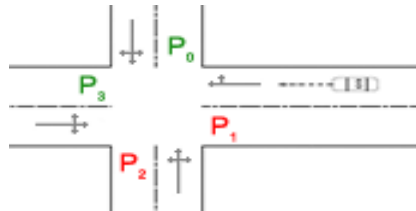
The probabilities  $P_{dr}$  of the other intelligent lamppost  $r$  out of the set of valid neighbors  $V_n$  are decreased using Equation 14. This decrease in probability values for these intelligent lamppost reflects their current status as sub-optimal routing solutions for the CBPRS routing service.

$$P_{dr} = -(1 - \lambda) \cdot P_{dr}, \quad \text{where } dr \neq dn, dr \in V_n, V_n \subset N_n$$

**Equation 14: Probability updating**

After alteration of the probabilities, the values are normalized and clipped between 0.05 and 1. The lower value of 0.05 is set to prevent ants from ignoring a possible route towards the destination via an alternative intelligent lamppost. Intelligent lampposts between which the route should be disabled can set their probability to zero for that specific neighbor; this prevents ants from inspecting that route. The backward ant repeats this process on every intermediate node between  $s$  and  $d$ . Once the backward ant arrives in  $s$  – the source of the forward-ant - the backward ant is destroyed and the path between the source and destination nodes is updated according to current information available to the algorithm.

The probability adjustment process discussed in the previous section described how probabilities were adjusted for neighboring intelligent lamppost that are member of the set of valid neighbors for the current destination. Where as in packet switched communication routers each neighbor of a router can be optimal with exception of the router the data-packet was received from road networks impose rules limiting the freedom of movement for a vehicles. Therefore, each intelligent lamppost at an intersection needs to be aware of the traffic rules that apply to all incoming lanes in order to guide vehicles through the road network with requiring drivers to break traffic rules.



**Figure 17: Valid neighbors for the current vehicle**

Figure 17 depicts an intersection that is monitored by and intelligent lamppost. The intersection has four incoming lanes that allow vehicles to head in any direction with the exception on the incoming lane of the right side of the intersection that only allows for a right turn or straight ahead. When a vehicle approaches the intersection from the right incoming lane, it requests a route toward its destination from the intelligent lamppost.

The intelligent lamppost monitors an intersection with four lanes to neighboring intelligent lampposts that together make up the set of neighbors  $N_n$ . However, not all neighbors out of this set can be considered valid for routing the vehicle towards its destination. If the probability value  $P_1$  for neighbor one represents the most optimal route the vehicle is forced to make a 180 degree turn on the intersection, such behavior –although legal at most intersections – is not common for navigational systems and should therefore not be allowed. If the probability value  $P_2$  for neighbor two represents the most optimal route the driver of the vehicle is forced to disregard traffic rules applicable to the lane he currently driving on, such behavior is undesirable. Therefore, the set of valid neighbors  $V_n$  for the current vehicle consist out of neighbor three and zero represented by the probability value  $P_3$  and  $P_0$ . During the probability updating process the set of valid neighbors is determined to prevent adjustments of probabilities for neighbors that could not be reached when vehicle would travel the same route of intelligent lampposts as the forward ant towards a destination.

### 3.3.2 Global Ants

The local ants, described in the previous section, were prohibited from leaving the sector in which they were spawned. This restriction of movement reduced the length of the route ants had to travel between a source and destination lamppost and improved the value information stored in the ants. While this restriction of movement leads to the distribution and formation of ‘optimal’ routes within sectors, routes between sectors are not updated. Therefore, in order to establish and maintain routes between sectors two additional types of ants are employed to maintain routes on the abstract level of the hierarchy. These global ants called the forward exploring ant and backward exploring ant can only travel between lampposts that are a member of the abstract level of the hierarchy.

**Table 10: Global ant memory layout**

<b>Destination virtual node</b>	11			
<b>Source virtual node</b>	9			
<b>Previous</b>	192.168.0.5			
<b>Source Address</b>	192.168.0.7			
<b>Path</b>	<b>Address</b>	<b>Average Delay</b>	<b>Address</b>	<b>Virtual node</b>
	192.168.0.6	10	192.168.0.5	10
	192.168.0.5	9	192.168.0.4	11
	192.168.0.4	5		

The layout of the global ants, as shown in Table 10, is similar to the layout used for the local ants. The first two fields are used to identify the destination and source virtual node of the ant. The previous field stores the intelligent lamppost last visited, this field helps the forward exploring and backward exploring ants to determine which information to retrieve or modify at the current intelligent lampposts. The source address field contains the address of the intelligent lamppost at which the forward exploring ant originated and to which the backward exploring ant is destined. The path field is constructed by the forward exploring ant and contains two additional fields, address and virtual node, in comparison to the local ant layout. The additional address and virtual node field are used to record when the forward exploring ant encounters an intermediate intelligent lamppost that is a member of sector not previously visited by forward exploring ant. The backward exploring ant uses the information stored in the path field to adjust the global routing tables of intelligent lampposts residing on the abstract hierarchical level. The following section discusses the manner in which the forward exploring ant and backward exploring ant operate.

#### 3.3.3.1 Forward Exploring Ant

The ant-based algorithm of the CBPRS dynamic routing service uses forward exploring ants to discover and maintain optimal routes between sectors. Forward exploring ants are spawned - at user-defined intervals – by intelligent lampposts that reside on the abstract level of the network hierarchy. The destination of the forward exploring ant is a randomly chosen virtual node identifier of another sector within the network. The forward exploring ant’s freedom of movement is restricted to intelligent lampposts of the abstract level of the network hierarchy. Therefore, routes towards virtual node destinations lead to the first encountered intelligent lamppost that represents a global routing node of the destination virtual node. Detailed level routing services are responsible for determining the most optimal route towards an intelligent lamppost within the destination sector.

The restricted freedom of movement of the forward ant enables it to carry larger payloads in terms of collected data since the number of intermediate intelligent lampposts on a route towards a destination is limited. Forward ants locate the optimal route towards a destination by inspecting the global probability tables at intelligent lampposts. The forward exploring ant determines next intelligent lamppost on the path towards the destination by choosing the neighbor of the current intelligent lamppost with the highest probability value associated with the destination of the forward ant.

At every intelligent lamppost, the forward exploring ant encounters upon leaving the intelligent lamppost where it originated it collects traffic condition and sector information. The forward exploring ant determines and stores the average delay between the previous and current intelligent lamppost and the address of the current intelligent lamppost in the same manner as the forward ant. If the current intelligent lamppost belongs to sector that has not been encountered before by the forward exploring ant before it also stores the address of the current intelligent lamppost and the sector identifier. If the forward ant visits an intelligent lamppost twice – thus creating a cycle – the path entries of the cycle are removed. If the detected cycle of visited intelligent lampposts is greater than half of the ant's current age – measured in the number of lampposts visited – the forward ant is destroyed.

The journey of the forward exploring ant ends when it reaches an intelligent lamppost that belongs to the sector of the destination virtual node. At the destination, the forward exploring ant stores the final piece of delay and sector information into its memory before the transferring entire path to a newly created backward ant that retraces the path taken by the forward exploring ant and adjusts global routing tables. Once the forward exploring ant's memory is transferred successfully, it destroys itself.

### **3.3.3.2 Backward Exploring Ant**

The forward exploring ant, discussed in the previous section, is an information collection ant that stores in memory information concerning the path taken towards the destination and the average of cost of that route when travelling via those roads. The forward exploring ant, like the forward ant, is unable to adjust routing tables since the taken cannot be deemed optimal until the destination virtual node has been reached. Therefore, the ant-based algorithm of the CBPRS routing service uses a backward exploring ant to adjust the global routing tables of intelligent lampposts along the most optimal route.

At each intelligent lamppost encountered when retracing the route the backward exploring ant uses the average delay and sector information collected by the forward ant to adjust the global routing tables. The backward exploring ant updates the probability value for the route between the current intelligent lamppost and the destination virtual node and every virtual node encountered in between using the same equations and procedure as the backward ant. However, the set of valid neighbors is further restricted on the abstract level of the hierarchy since only intelligent lampposts that reside on the abstract level are taken into account by the backward exploring ant.

## **3.4 Limitations and Possibilities**

The main limitation of the ant-routing algorithm when applied to traffic networks is its dependency on traffic condition information provided by vehicles. Without a constant supply of information concerning the delays on at least the main roads in city the algorithm is unable to guarantee a time optimal path from source towards the destination. During the period in which this problem might seem likely to occur – nighttime – the algorithm is however capable to calculate shortest time paths through the city. The reason for this is that due to the low number of vehicles influencing the average delay for a road the values are – under normal circumstances – sufficiently decreased by vehicles entering and leaving after the evening main traffic peak hours. These vehicles will set the average road delay for them main roads back to the original values for the roads or at least very near to those values.

Another limitation imposed by this implementation of the ant-based algorithm lies within the forward-exploring-ant and backward-exploring-ant. These two ants are required to travel significant distances via the main roads of a city between different colonies that could lead to alternations of probabilities with outdated information.

The impact of this limitation however depends on the speed at which the intelligent lampposts are able to communicate with each other. In light of the setup described in chapter one the bandwidth available is sufficient to communicate the ants between origin and source with a delay of a few seconds. If the roundtrip time for the exploring-ants exceeds the minimum requirements for proper functioning of the algorithm one could instead of investing in more bandwidth increase number of hierarchical levels. By clustering colonies and restricting exploring-ants further one could decrease the travel times of these exploring-ants and further lower the size of the ant in bytes.

The clustering of certain intersections into colonies does not limit itself to towns, villages or cities but could be applied nationwide. Such a configuration would allow for guidance of traffic throughout the country facilitating better distribution of traffic on the nation's available roads. The algorithm if modified and implemented correctly to deal with such a large flow of information could then reduce the problem of traffic jams and increase the overall traffic flows. Although the investment required in such a system would be huge, the reduction in economic damages done by traffic jams would more than compensate that.

Finally, the currently proposed ant-based algorithm only deals with historical data. While the age of the information gathered is not old it only allows the ant-based algorithm to react on events already motion. An ant-based algorithm able to deduct patterns from frequently occurring traffic patterns and predict future traffic development could increase the effectiveness of the algorithm even further. Allowing the input of information gathered by emergency services into the ant-based algorithm would also the algorithm to respond faster to accidents or divert vehicles from major incidents such as building fires.

## Design of the Simulation Environment

This chapter discusses the design of the CBPRS simulation environment as proposed in chapter one. Finding an answer to the research question posed in chapter one is the most important task of this thesis. The design of the CBPRS simulation environment is certainly the second most important since it determines along which path an attempt is made to answer the research question. The design of the simulation environment presented in this chapter further limits the possibilities for the conduction of experiments. This however is not necessarily a negative but should be kept in mind when developing when developing a software application.

Therefore, the focus of this chapter does not lay on the software design of the CBPRS simulation environment but on answering the question why certain design decisions have been made and how they influence the final software application. The actual software design is modeled via the unified modeling language (UML) approach. This approach focuses around a number of different diagrams detailing the inner workings of the final application. For the sake of clarity, this chapter only discusses the package diagram and class diagram. A package is set of classes and interfaces that all focus on providing one distinct type of service. The package diagram shows in a clear manner the relationship between packages and their relations. The class diagram shows the actual classes and interfaces in the packages and displays the relationships between them.

Section 4.1 of this chapter discusses the global design of the simulation environment by discussing how the description of the CBPRS in chapter two is decomposed into packages. Section 4.2 discusses the simulation package that provides common functionality. Section 4.3 discusses the infrastructure package that forms the road network infrastructure within the simulation environment. Section 4.4 discusses the routing package and details how Dijkstra's static routing algorithm and the Hierarchical distributed routing algorithm as discussed in the previous chapter are incorporated. Section 4.5 discusses how the parking system of the CBPRS is designed. Finally, section 4.6 discusses the limitations of the design as presented in this chapter and the possibilities for readers interested in extending the design. The project website [22] contains the information and diagrams presented in this section including full documentation of the entire project.

### 4.1 Global Design of the Simulation Environment

The first task undertaken when developing an application is decomposition of the application description into distinguishable software components. The description given of the CBPRS in chapter one serves as the foundation for this task. The CBPRS system is active on three layers of the city infrastructure namely the power-line infrastructure, the road network infrastructure and the wireless communication infrastructure. The lampposts that are part of the CBPRS bind these layers together. The lampposts communicate between themselves and vehicles that navigate the city road network to support the routing and parking services of the CBPRS.

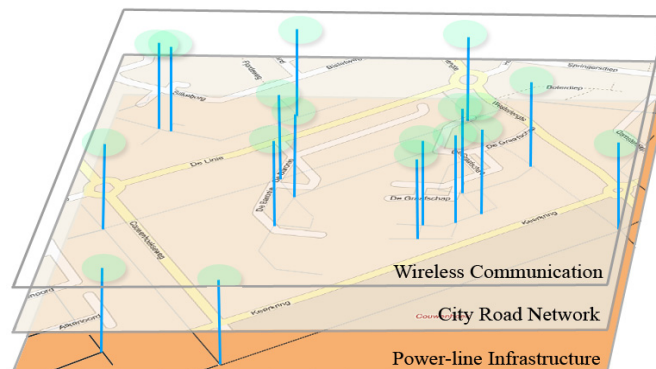
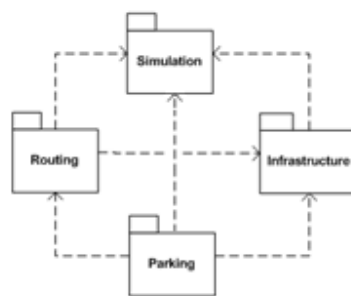


Figure 18: Layers represented in the simulation environment

This brief synopsis of the decomposition process reveals that the CBPRS consists of three main packages; the road infrastructure, the routing system and the parking system. The infrastructure package contains all functionality concerning roads, intersections and the movement of vehicles in the simulation environment. The routing system package contains the static and dynamic routing algorithms used to guide vehicles through the city. The parking system package is responsible for monitoring the parking places and all other parking related services.

The perceptive reader notices an omission in the package decomposition presented in this section. A logical decomposition would have revealed a communications package to simulate the communication via the ether (wireless) and power-line. The focus of this thesis however does not lie with realistic simulation of the behavior communication protocols and networks. Therefore, this package is omitted and the functionality required for communication is incorporated into the appropriate packages. The omission of this package however does not infer that communication is not accounted for in the design of the application.



**Figure 19: Simulation environment Package Diagram**

To minimize the overlap in functionality provided by the packages and bind them together via a common interface a fourth package is introduced. The simulation package provides common functionality and often used constructs as described in section 4.2. Figure 19 contains the package diagram, this diagram shows the relationships between the packages in terms of dependence on functionality.

Once the relationship at the top level of the design is determined via the package diagram the lower class diagram level can be considered. This level of design requires the definition of a number of focal points to guide the design of the software application. The focal points defined for CBPRS simulation environment are extensibility, decoupling of objects, modularity of packages and generic behavior. By actively using these focal points, the design will show non contra-dictionary relationships between classes on the detail level. Furthermore, the application developed on the basis of the design will be flexible and adaptable to changing demands and new avenues of research. The concepts resulting from these focal points are discussed in detail in the next sections.

## 4.2 Simulation

Every building needs a proper foundation in order to support the floors that are constructed on top of it. In software design this principle also applies and in order to build the different packages of the CBPRS simulation environment a solid foundation is needed. The simulation package described in this section is the foundation of the entire CBPRS simulation environment and contains a number of frequently used interfaces and classes.



### 4.2.1 Design patterns

The incorporation of the design focal points into the CBPRS simulation environment is achieved via the application of object-oriented design patterns. Design patterns [21] are solutions to common problems within the world of software design. Over the years a great number of these patterns have been developed to ease software design. Four of those patterns are incorporated into the CBPRS simulation environment namely; the interface pattern, the façade pattern, the factory pattern and the delegation pattern. The design patterns and their usage within the simulation environment are discussed briefly.

The Interface design pattern is used to create a basic set of methods that have to be implemented by inheriting classes. The interface that is inherited serves as a basic contract between developer and user by specifying the functionality the user can expect from a certain class. This pattern is used extensively to provide the user not only with set of guaranteed functionalities but also as a method to extend the application easily by allowing the user to implement his own classes that abides by the interface. Together with factory pattern discussed below this creates a decoupled, extendable and highly flexible environment.

The façade design pattern is used to provide a single point of access to the functionality provided by a number of other classes. Every package with the exception of the simulation package is a subsystem of the CBPRS simulation environment by providing a certain service. The infrastructure package for example provides the user with the possibility to create roads, intersections and vehicles. However, this functionality is spread over different classes. The subsystem class or façade class provides a single point of access to this functionality giving the user a better understanding of the program flow.

The implementation of the interface design pattern allows for the usage of the factory design pattern. The factory design pattern is able to create an object by choosing from a set of classes registered with it. To improve the capability of the factory the delegation pattern is also applied. This pattern allows for the delegation of certain tasks to other objects. In the case of the factory design pattern the construction of objects is delegated to another part of the simulation environment. This for example allows the user to register his own type of vehicles with the simulation environment without altering the existing classes of the application.

The singleton design pattern prevents the creation of multiple objects of the same type. By combining this pattern with the façade pattern the CBPRS simulation environment is able to guarantee that only a single instance of a certain package is loaded. The singleton pattern also creates a single point of access for each subsystem further simplifying the flow of the program and improving the accessibility of objects.

### 4.2.2 Software design

The class diagram for the simulation package as shown in Figure 20 depicts the relation between interfaces and classes. The main relations between the interfaces – the dashed lines – consist out of dependencies of functionality provided by other classes. This arises from the fact that interfaces are only allowed to define method bodies but have no actual implementations.

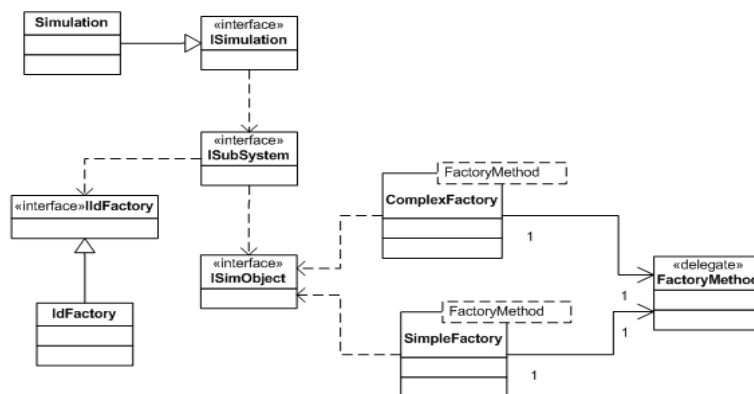


Figure 20: Simulation package class diagram

The *ISimObject* interface defines the basic contract within the simulation environment and serves as the root object for all objects that can be simulated. The *ISubSystem* defines the basic contract for singleton façade classes. The *ISimulation* interface and its inherited class *Simulation* define a basic multi-threaded environment for executing simulations. The *Simulation* class can be inherited by implementers to alter the execution path of the simulation environment. The factory classes incorporate the factory design pattern described in the previous sub-section, which are used in other packages.

### 4.3 Infrastructure

The simulation of traffic movements in a city environment is a complex and potentially computational heavy task, certainly when modeling a microscopic simulation environment with a large set of parameters that influence the behavior of vehicles and drivers. Certain decisions have to be made in order to keep the simulation environment suitable for implementation and modern day computer hardware capabilities.

The most computational effective manner in which to model a microscopic traffic simulator is via the use of a cellular automaton [22]. Cellular automata are discrete mathematical models [23] first developed by Von Neumann that operate on a regular grid of infinite size and dimensions. Each cell in regular grid can be in one of a finite number of predefined states and consulted via an index function ( $f(x, y)$  for a two dimensional regular grid). The state of specific cell depends on the rule-set used for the CA and the states of the cells immediate neighbors in time period  $t - 1$ . The direct neighbors of a cell are also known as its neighborhood. Time, the states of the CA and space in the CA are treated as being discreet. When time is allowed to progress from  $t$  to  $t + 1$  the CA updates all cells synchronously moving the entire CA into a new generation. By allowing time to progress for a progress for a number of steps complex patterns can result from the CA as described in the following paragraph.

Assume a cellular automaton that operates on a one-dimensional regular grid wherein each cell can be either black or white. Since the grid is one-dimensional, the neighbors of cell can be defined as the cells that lie directly to the left or right of the current cell forming a neighborhood of three cells. As each cell can be in one of two states and the neighborhood consists out of 3 cells the total number patterns possible is  $2^3 = 8$ . The number of rule-sets therefore applicable to the CA is  $2^8 = 256$ . Wolfram [23] noted that these a rule numbers fall within the range  $\{0,255\}$  and can therefore converted to an eight digit binary representation consistent with the number patterns possible in a cell neighborhood.

$$\frac{111\ 110\ 101\ 100\ 011\ 010\ 001\ 000}{0\ 0\ 0\ 1\ 1\ 1\ 1\ 0}$$

Equation 15: Cellular automaton rule-30 in binary format

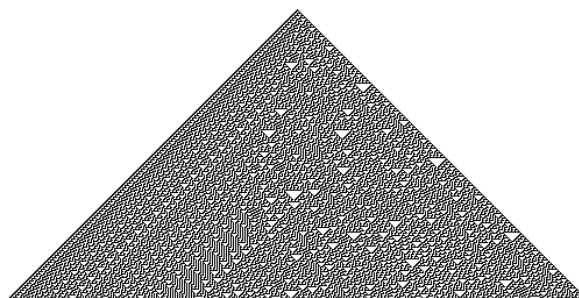


Figure 21: Rule-30 cellular automaton

To illustrate the previous a description is given of rule-30 that according to Wolfram [23] is the key to the secret of natural automata. Equation 15 shows rule-30 converted to its binary representation 00011110 and applied to the possible patterns a cell neighborhood can assume. Starting out with a single cell with state black (1) and applying the rule-set in Equation 15 five hundred times Figure 21 is formed which shows the CA at each time step  $t$  from top to bottom in descending order. The pattern that emerges from this iteration provides seemingly random output without the input of randomly chosen numbers or factors. The middle column of this CA can be used to generate pseudo random numbers.

The manner in which the concept of cellular automaton is applied to the infrastructure of the simulation environment is described in the next section. Then the concept of intersections is discussed in section 4.1.2. Section 4.1.3 discusses the updating rules vehicles apply when determining their velocity and position on the road. Section 4.1.4 then groups these together and discusses the UML software design. An individual description of each of the main infrastructural types can be found in appendix IV.

### 4.3.1 Roads and Lanes

When asked the road by a driver that has lost his bearings one normally replies in sentences such as “take the next left onto road a, then go right onto road b...”. The driver is then able – if he understood the directions correctly – to navigate to his destination by presorting into the left turn lane at the intersection onto road a and then presorting onto the right turn lane to enter road b. While the driver was informed only about the turns he should take at the intersections, he proceeded to execute a complex procedure of lane changing events in order to reach his destination.

The road functions thus merely as a parent to a collection of driving lanes over which movement is possible between two intersections. The road groups a set of lanes under a ‘unique’ name facilitating the process of giving directions. The road dictates the maximum velocity at which vehicles are allowed to travel over its driving lanes. From a simulation point of view, the function of the road is that of a container for grouping objects. The vehicles interact solely with the driving lanes of the road.

Driving lanes are a unidirectional portion of paved roadway that allows vehicles to move from one intersection to another. Each driving lane has a certain length in meters, a maximum vehicle density and – based on the maximum speed – a maximum traffic flow whereby it influences the traffic conditions in the immediate vicinity of the lane. The driving lane in the simulation also specifies a direction in which the vehicle is allowed to cross the intersection at the end of the lane. This directional indicator allows vehicles to determine if they are on the right lane considering their destination or whether they should attempt to change lanes before reaching the end of the driving lane.

**Table 11: Vehicle velocities**

Cells per step	Velocity (km/h)	Velocity (mi/h)
0	0	0
1	27	16.78
2	54	33.55
3	81	50.33
4	108	67.11
5	135	83.89
6	162	100.66

The application of the cellular automaton principle to the road infrastructure causes each lane to consist out of a number of cells. These cells are 7.5 meters (8.20 yards) long and represent the space one vehicle would consume when standing still in a traffic jam. Each cell can therefore only be occupied by one vehicle. The cell length restricts the velocities vehicles can achieve when travelling on a driving lane, as shown in Table 11.

### 4.3.2 Intersections

The intersections in the simulation environment have a great impact on the patterns of vehicular movement within a simulation. Intersections that allow only a single vehicle to pass at each time step unnecessarily delay the progress of other vehicles causing traffic jams and delays to appear on roads where they would not appear in reality. Therefore, certain precautions have to be taken in order to insure that multiple vehicles can cross the intersection at the same time without leading to collisions and disobedience of the precedence rules applicable to the intersection.

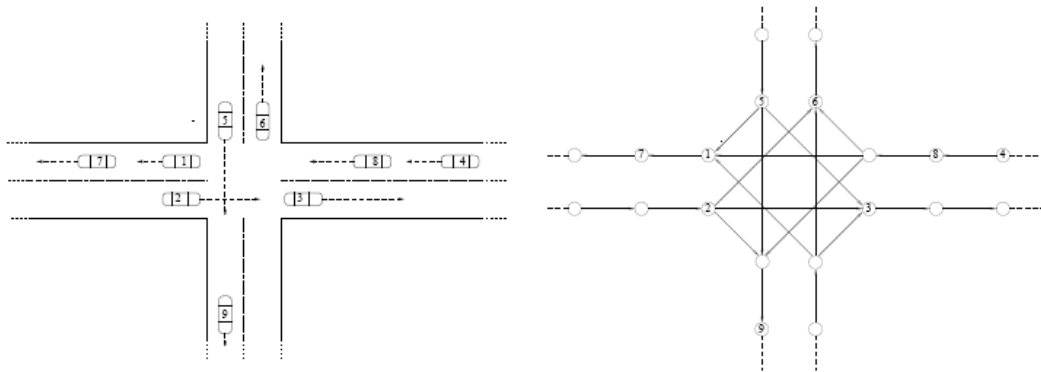


Figure 22: Intersection representation [24]

Each intersection consists of a number of inbound lanes and outbound lanes. A vehicle traveling on an inbound lane has the possibility to cross the intersection onto an outbound lane with the exception of the outbound lane that belongs to the same road as the inbound lane the vehicle is currently driving. Therefore, depending on the number of inbound and outbound lanes and the allowed turn directions a directed graph can be generated in which the edges represent a path from an incoming lane to an outgoing lane as shown in Figure 22. The set of edges in this directed graph then represents all valid manners in which vehicles can cross an intersection.

The resulting directed graph is not directly applicable to the simulation since it contains a number of conflicting edges. The conflicting edges are those edges that would cause collisions between two vehicles when being travelled on at the same time. In order to eliminate the problem of conflicting edges a conflict matrix is constructed on the basis of the directed graph of the intersection. The conflict matrix consists of a series of entries that specify if two edges can be active at the same time without leading to vehicular collisions.

Human drivers are able to deduce from the signal given by the traffic light or the precedence situation on the intersection when they can cross the intersection or when they should stop and wait for a green signal or give precedence. The vehicles in the current simulation environment could simulate this human behavior. However, this would require a large amount of computational power; for each vehicle near an intersection would have to calculate the intersection traffic situation at each simulation time step.

The solution to this problem is found through a messaging system. Each vehicle that approaches an intersection notifies the intersection of its arrival and the edge it intends to follow when crossing the intersection. The intersection then verifies if the edge that the vehicle wants to follow is available – meaning that there are no conflicting routes in use at current – and applies precedence and traffic light rules.

The intersection should then formulate a response to the vehicle stating whether it is allowed to proceed onto the intersection. In the next time step, the vehicle determines if it is allowed to continue and enter the intersection or to break to a halt before entering the intersection. This type of messaging system ensures collision free traffic movement over intersection and reduces the computational overhead.

### 4.3.3 Vehicular Movement

Vehicular behavior in the simulation environment is the most important factor of those that determine the final realism of the simulation environment. Therefore, great care should be taken when modeling the set of behaviors exposed by human drivers in a simulation environment. However, human behavior is highly complex and depends on a myriad of variables that would burden the simulation to a point where the requirements of the simulation environment would exceed modern day hardware computing capacity. To solve this dilemma a basic set of rules is implemented [25] that allows for realistic movement and lane changing behavior of vehicles in cellular automata.

$$v = \text{Min}(v_n + 1, (\Delta x)_n - 1, v_{max})$$

$$v_{n+1} = \begin{cases} \text{Max}(0, v - 1), & \text{with probability } P_{brake} \\ v, & \text{with probability } (1 - P_{brake}) \end{cases}$$

$$x_{n+1} = x_n + v_{n+1}$$

#### Equation 16: Vehicle speed and position determination

The velocity and position of a vehicle is determined via equations above. The velocity  $v$  of a vehicle is determined as the minimum of the current speed of the vehicle plus one, the distance between the front bumper of the current vehicle and the vehicle directly in front  $(\Delta x)_n$ , and the maximum speed of the vehicle  $v_{max} \in [0,6]$  in blocks per time step  $n$ . The actual speed of the vehicle in the next time step  $v_{n+1}$  is then determined using the probability  $P_{brake} = 0.05 \in [0,1]$  and models the influence of human drivers on the actual velocity of the vehicle as a result from the observations or distractions.

The position of the vehicle in the next time step  $x_{n+1}$  is a simple addition of the current position  $x_n$  and the velocity of the vehicle in the next time step  $v_{n+1}$ . These rules allow vehicles to move in a forward direction on a road while preventing collisions between vehicles. Since the new velocity of the vehicle is always less than the distance between two vehicles on the same lane. While this may be seen as shortcoming of these rules one should realize that it is not the collision that is interesting but the effect such a collision has on traffic flows in the simulation.

$$v^{o,b} \leq \Delta x^{o,b} - 1$$

#### Equation 17: Vehicle lane changing Security Constraint

The rules for changing lanes differ between changing from right to left and left to right. However, both are bound by the security constraint in Equation 17. The security constraint states that vehicles are not allowed to change to another lane if the distance to the vehicle directly behind them on the lane  $\Delta x^{o,b}$  is smaller than the current velocity of that vehicle  $v^{o,b}$ .

$$v_{max} > \Delta x - 1 \text{ And } \Delta x^o \geq \Delta x$$

#### Equation 18: Vehicle Right to left lane changing rule

The actual lane changing decision is then made on the basis of Equation 18 that states that a vehicle should only change lanes if the vehicle is obstructed from attaining its maximum speed  $v_{max}$  on the current lane and the free space on the other lane  $\Delta x^o$  is greater or equal to the free space on the current lane  $\Delta x$ .

$$\begin{cases} v_{max} < \Delta x - 1 - v_{off} \text{ AND } v_{max} < \Delta x^o - 1 - v_{off}, & \text{with probability } (1 - P_{l2r}) \\ v_{max}^{o,b} \leq \Delta x^{o,b} - 1 \text{ AND } v \leq \Delta x^o - 1, & \text{with probability } P_{l2r} \end{cases}$$

#### Equation 19: Vehicle Left to right lane changing rule

The vehicle decides to change from the left lane to the right lane when the gap between the current vehicle and the next vehicle on the same lane  $\Delta x$  is greater than maximum velocity of the vehicle  $v_{max}$  and the gap between the current vehicle and the next vehicle on the right lane  $\Delta x^o$  is also greater than the maximum velocity of the current vehicle. The parameter  $v_{off} = 4 \in [0,10]$  in this equation is used to determine the flow of vehicles influencing the tendency of drivers to remain on the left lane. Certain situations however arise that make the lane changing rule unfeasible. When traffic is moving slowly and with a sufficiently high value for  $v_{off}$  vehicles are never allowed to switch to the right lane leaving this practically unpopulated. Therefore, a second set of rules is applied to left to right lane changing decision with the probability  $P_{l2r} = 0.02 \in [0,1]$ . These rules allow the vehicle to change from left to right if such a maneuver does not hinder the vehicle directly behind on the right lane and the gap to the vehicle directly in front on the right lane  $\Delta x^o$  is greater or equal to the current velocity  $v$  of the vehicle.

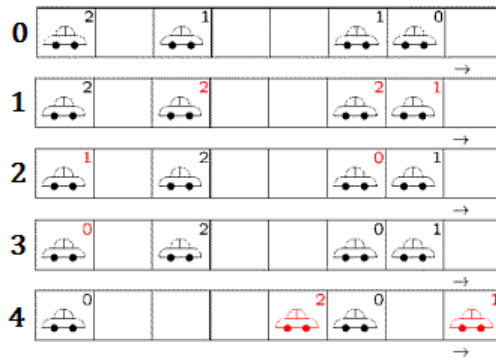


Figure 23: Vehicular movement in the cellular automaton [26]

Figure 23 depicts the progress of four vehicles on a single lane road according to the rules described above. The top image depicts the position of vehicles on the road at time step  $t$ . The model then goes through three distinct update phases. During the first phase, the speed of the vehicles is increased to the maximum allowed speed according to topmost equation in Equation 16. The second phase reduces the velocity of the vehicles if their determined speed is greater than the distance they can travel without causing incidents. The third phase then takes into account the random braking behavior exhibited by human drivers by decreases the speed of vehicles using the middle equation in Equation 16. Finally, the bottom equation in Equation 16 is applied moving the all vehicles to time step  $t + 1$  depicted as phase four in Figure 23.

#### 4.3.4 Software Design

One can model a road network infrastructure with each having certain benefits and negatives in terms of flexibility and speed on modern day computer hardware in many different ways. Therefore, it is vital that decisions made concerning the design of the infrastructure package in previous sections are not poured into concrete. By applying the focal points as described previously, the class diagram of the infrastructure package looks as follows.

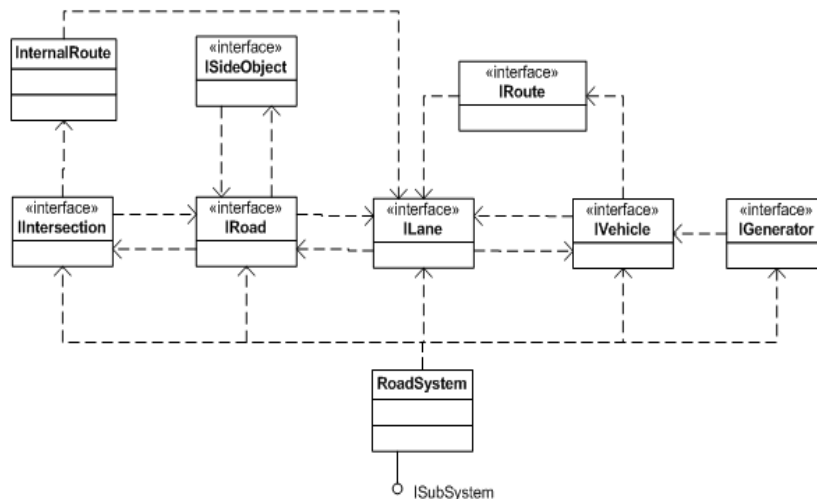


Figure 24: Infrastructure class diagram

The road network consists of intersections, roads, driving lanes and vehicles which are modeled as interfaces in Figure 24. The intersection determines via a set of *InternalRoute* classes if a vehicle can navigate from the lane it is currently driving on to the lane it desires to drive on according to its route as modeled in the *IRoute* class. The *IGenerator* class defines a basic generator class for spawning vehicles during the simulation. The interfaces in the infrastructure class diagram all have default implementations that are not shown in this chapter, interested readers are referred to source code for details.

## 4.4 Routing System

This section describes the design of the routing system part of the CBPRS simulation environment. The two routing algorithms incorporated are Dijkstra's algorithm for static vehicle routing and the ant-routing algorithm for dynamic vehicle routing. The algorithms and the manner in which they are implemented are discussed briefly in sections 4.4.1 and 4.4.2 before discussing the software design in section 4.4.3.

### 4.4.1 Dijkstra Routing

Dijkstra's algorithm is able to find the shortest paths in a graph with non-negative weights as was described in chapter two. However, implementing Dijkstra's algorithm as described in chapter two causes some unrealistic behavior of vehicles. Imagine a vehicle travelling along a shortest path  $A \rightarrow B \rightarrow C$  where the driver intends to park his vehicle on the road between  $B \rightarrow C$ . The journey of the vehicle toward the parking places is completed without problems along the shortest path. However, once the driver intends to return to  $A$  the route suggested by Dijkstra's algorithm is  $C \rightarrow B \rightarrow A$ , which is the absolute shortest path. This however means that a vehicle is forced to make a 180 degree turn at intersection  $C$  to follow the path suggested by Dijkstra's algorithm.

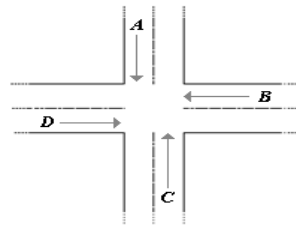


Figure 25: Incoming driving lanes onto an intersection

This would mean that if Dijkstra's algorithm is implemented in this manner all vehicles that use Dijkstra's routing algorithm to find or leave a parking place are forced to make a 180 degree turn once they return to their point of origin. While some vehicles in reality make this turn the behavior on general is not realistic and should therefore be avoided.

The solution to this problem is found by constructing a routing table for every incoming lane onto an intersection. Figure 25 shows the four incoming lanes onto the intersection for which a routing table is constructed. Instead of evaluating all paths that lead from the current intersection Dijkstra's algorithm is now instructed to only evaluate those paths that are reachable from the incoming lane for which the routing table is constructed at present. The downside of this process however is the increase in data needed to route vehicles. This problem is overcome by storing routing directions as sequence of bits containing the direction in which a vehicle should head in order to reach its destination via the shortest path for the current driving lane.

### 4.4.2 Ant Routing

The ant routing algorithm is designed into the CBPRS simulation environment according to the network model design as presented in chapter three. The lampposts required at each intersection for communication between vehicles and the routing system is however modeled as a routing network node instead of lamppost as used for the parking system. This distinction between lampposts was made due to the modular design of the CBPRS simulation environment that does not allow for direct associations between classes of different packages – this in order to avoid unnecessary dependencies. This benefit of this design is that the search time for the matching routing system lamppost at an intersection can be decreased from a linear  $O(n)$  search time to almost  $O(1)$  search time.

### 4.4.3 Software design

The *AntNode* class in Figure 26 represents the lamppost present for routing vehicles at each intersection within the city environment. This class contains a routing table and has references to one or more *AntLink* classes that contain the delays recorded for the roads. Furthermore, a collection of incoming *IAnt* types is maintained. Ants stored in this queue are received from neighboring *AntNode* classes during the current simulation time but cannot be processed until the next time step.

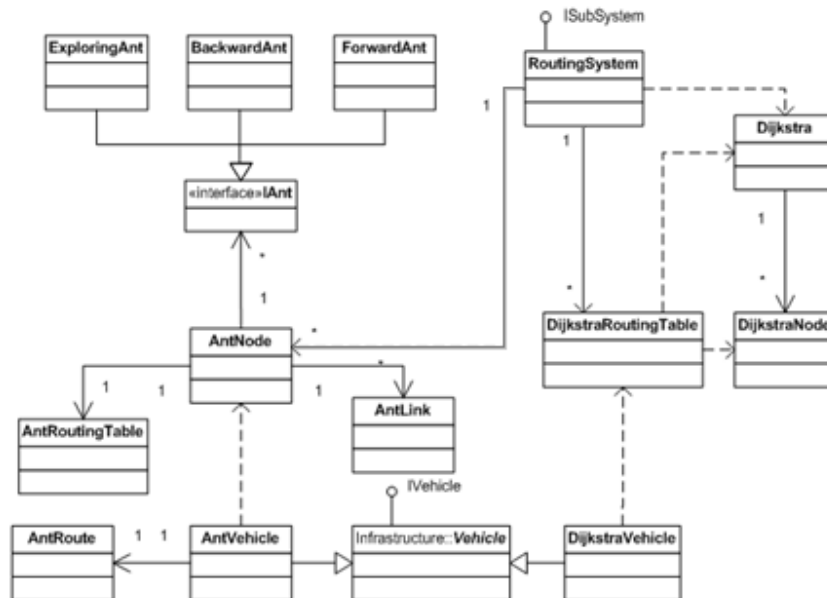


Figure 26: Routing Class Diagram

Dijkstra’s algorithm produces static information in the form of a collection of *DijkstraRoutingTables* classes. The algorithm itself needs to be executed only once so the classes *Dijkstra* and *DijkstraNode* are discarded once the routing tables have been constructed. The routing tables resulting from Dijkstra’s algorithm and the *AntNode* classes are referenced through the *RoutingSystem* façade class to provide convenient access to them.

The routing system package also defines two types of vehicles. The *AntVehicle* supplies the routing system with traffic condition information and requests dynamic routing information. The *DijkstraVehicle* only requests static information from the routing system queried from the *DijkstraRoutingTable* classes. The vehicles both extend the default vehicle as defined in the infrastructure package.

### 4.5 Parking System

The parking system is designed following the description given of the CBPRS in chapter two. The parking system therefore consists of an administrative system to store parking information concerning vehicles places and the period parked. A sector station in which the parking system sectors are modeled is constructed via the same pattern as colonies of the ant-routing algorithm in chapter four are modeled. Each sector station communicates with a set of intelligent lampposts that are located along the side of roads. These intelligent lampposts interact with the sensors located in parking places and parking garages to determine if places are empty, occupied or reserved.



### 4.5.1 Software Design

The description given above is clearly shown in the class diagram of the parking system in Figure 27. Objects that can be placed along the side of a road implement the *ISideObject* interfaces as defined in the infrastructure package. This interface allows for incorporation of objects into the road network without creating unnecessary dependencies between the parking system and the infrastructure package.

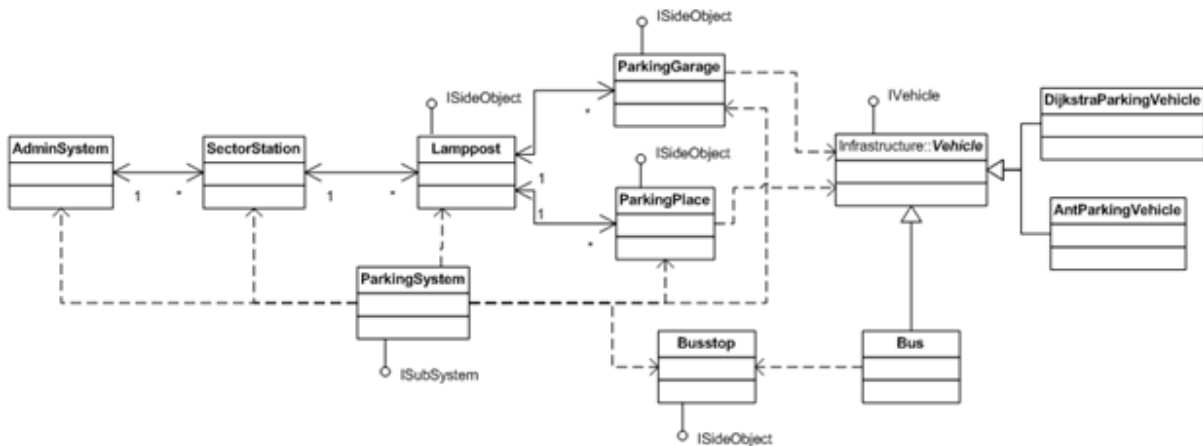


Figure 27: Parking system class diagram

The parking system package also defines three types of vehicles that inherit their basic set of functionalities from the *Vehicle* class defined in the infrastructure package. The *DijkstraParkingVehicle* and *AntParkingVehicle* classes both choose a destination from the available collection of parking places within the infrastructure. The only difference between the vehicles is the routing method they use to reach the parking place. The third type represents a *Bus* that follows and stops along a route of predefined *Busstop* classes.

### 4.6 Limitations and Possibilities

The major limitation of the CBPRS simulation environment as described in this chapter is that intersections cannot cope with more than four roads. Furthermore, the four roads can only be connected at predefined positions on the intersection. While this behavior is certainly not realistic, the design choice was made to allow for detailed drawing of the simulation environment. Drawing a city infrastructure in a detailed manner is a complex task and is further complicated by the different configuration the environment allows the user to create. Therefore, the intersections in the CBPRS simulation environment should at least have one road connected to them with a maximum of four places at the top, right, bottom or left position along the rectangular intersection.

The possibilities of the CBPRS simulation environment are virtually endless. The entire environment is designed in such a manner that the user can easily extend, modify or replace classes and even packages as long as the user abides by the 'contracts' specified in the interfaces. These interfaces are created in such a manner that they expose generic behavior adaptable to every type of infrastructure. If the user desires it is even possible to replace the cellular automaton scheme of the infrastructure package with another solution without breaking or remodeling other packages.



## Development of the Simulation Environment

This chapter describes how the software design as discussed in the previous chapter is implemented. The reader interested in the possibilities of the application concerning experimentation is referred to the following chapter. This chapter discusses how the key elements of the simulation environment were implemented. The reader interested in elements of the simulation environment not discussed in this thesis is referred to the source code that contains elaborate commentary and descriptions of purpose. Section 5.1 discusses briefly in what language the application was implemented, why this language was chosen and discusses third-party libraries used during development. Section 5.2 discusses the key technical details of the simulation environment. Finally, section 5.3 discusses the possibilities and limitations of the simulation environment due to the chosen implementation method.

### 5.1 Application Environment

The implementation of the CBPRS described in this document can be done using a number of operating environments and different programming languages. The application programmable interface (API) chosen to develop the system naturally applies a number of restrictions concerning the manner in which an application can be developed and what components are supplied in order to ease development. The time available to develop the CBPRS system is also a constraint to consider when choosing the right development environment.

The only two realistic frameworks then remaining are Sun's Java and Microsoft's Framework.NET. Java has been the default choice for many years in the scientific community and thought extensively in universities. The Framework.NET is finding its way slowly into the scientific community. Comparing these two programming languages and their supplied APIs the Microsoft Framework.NET has the advantage of extensive and well organized collection of documentation and samples, frequent extensions, full integration into Microsoft Windows leading to increased execution speed and a superior integrated development environment (IDE) that is freely available. Furthermore, cross operating system development is not a requirement of the current project while rapid application development is making the choice for Java less attractive. Therefore, the choice has fallen on the Microsoft Framework.NET and the C# programming language supplied with it, which looks similar to the C/C++ programming language on which the Java syntax is based. However, by choosing the Framework.NET the choice of operating system is restricted to Microsoft Windows XP or higher.

The development process was further assisted by the incorporation of three third-party components namely Piccolo.Net, Zed-Graph and #ZipLib. Piccolo.Net [26] is an object oriented 2D graphical framework developed for C# that eases the implementation of complex rendering hierarchies and provides functionality such as zooming, rendering from different perspectives and optimized rendering of images onto the display.

The Zed-Graph [27] library contains a rich set of different types of graphs used in the application to display simulation information to the user. #ZipLib [28] provides default implementations for a number of compression algorithms including ZIP used in the simulation environment during the saving and loading processes.

## 5.2 Technical Details

Each layer as presented in the previous chapter is implemented as a separate dynamic linking library that groups the classes and objects of the layer into a cohesive set. To distinguish between the different layers each layer implements its own logical namespace with a common namespace called CBPRS. The namespace groupings avoid cluttering of the global namespace in which all objects normally reside. The implementation via dynamic linking libraries was chosen to allow for extensibility with new layers and the possibility to remove or replace current layers if necessary in the future. The full source code and documentation accompanying the simulation environment can be found on the project website [22].

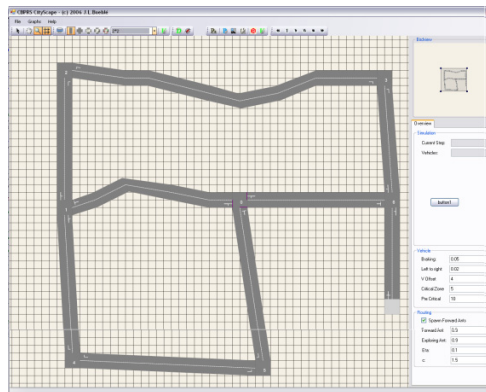
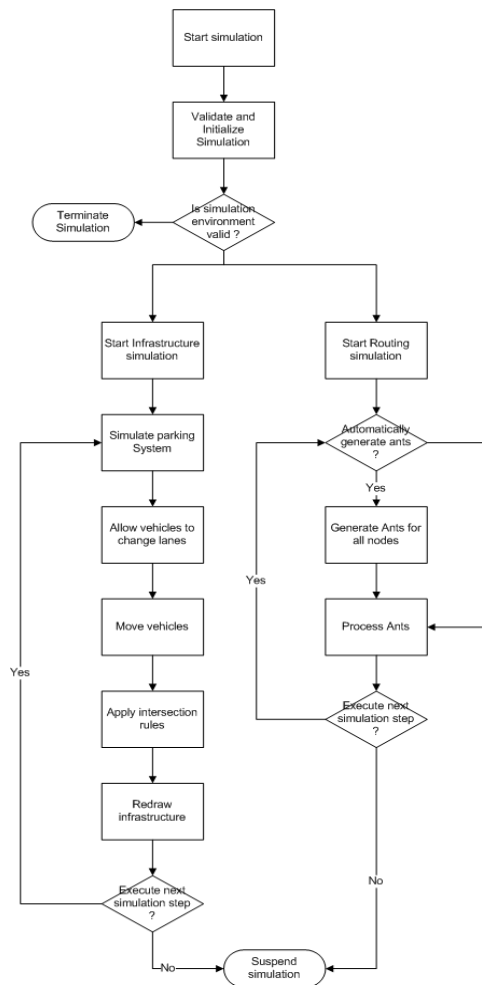


Figure 28: The Cityscape graphical user interface

The functionality provided by the layers of the CBPRS simulation environment is made accessible to the user via the Cityscape application. The Cityscape application is implemented as a tear off, meaning that it can be replaced or removed without breaking the functionality provided by the layers of the simulation environment. The Cityscape application functions as the city's infrastructure editor and simulator to produce orderly and quickly familiar interface to the user without the need to learn and use multiple separate applications. A complete description of the Cityscape GUI as depicted in Figure 28 can be found in appendix III.

The Cityscape application is implemented as a multithreaded application to allow sub-processes to execute concurrently and make optimal use of the resources available. One thread is permanently allocated to the GUI to maintain application responsiveness when executing a simulation. Two threads are allocated to the simulation environment allowing the simulation of the infrastructure and of the routing system to execute at a different pace. As the previous chapter described the simulation time steps for vehicles last one second while the simulation time steps for the routing system last 0.2 seconds, this difference was made to achieve more realistic behavior in the simulation environment [19]. These values however can be adjusted through Cityscape application although it is recommended that the routing system, which operates on a communication network, operate at a higher speed than vehicles move. The current relation in which the routing system is allowed to process ants five times per infrastructure update seems to provide the best results in terms of simulation and realism.



**Figure 29: Simulation process flow chart**

Figure 29 displays the simulation process flow chart. Once a user has configured the application in a manner that satisfies his requirements the simulation can be started. Before execution of a simulation, the entire environment is validated to ensure proper simulation results and prevent application errors while executing a simulation. If the application has found the environment to be valid, the simulation launches the infrastructure simulation and routing system simulation threads. If the application found the environment to be invalid, the user is notified and the simulation is terminated.

The infrastructure simulation thread processes all vehicle related behaviors and interactions. The process executed by this thread is static. However, the time taken to complete a simulation step depends on the size of the infrastructure and the number of vehicles present in the simulation environment. At the beginning of each step the parking system is given the opportunity to insert vehicles into the road network that have reached their parking time. If due to road traffic conditions the vehicle is unable to enter the road the parking system aborts the procedure for that vehicle and attempts to insert it on the next simulation step.

After the parking system has completed its tasks for the current simulation step the vehicles are allowed to change lanes. The lane changing process allows each vehicle in the simulation environment to test if the rules for a lane change – as defined in the previous chapter – are satisfied and act accordingly. Then the vehicles are moved one simulation step. The movement of vehicles results in a number of requests by vehicles indicating that they want to cross an intersection. These requests are processed next and the responses to those requests are determined via intersection specific rule-set. Finally, the user visible portion of the road network is drawn to show the new position of vehicles.

The routing system simulation thread is only concerned with the movement and processing of ants between the intelligent lamppost at each intersection. Ants are generated at a certain interval with randomly chosen destinations. The ant is allowed to move between two lampposts in the time span of one routing system simulation step. Under default settings an ant can be processed by four lampposts and visit five before the vehicles are moved. The process of movement and processing is repeated at each step.

## 5.2.1 Intersections

This section discusses the different types of intersections that are implemented in the infrastructure environment. These intersection types encompass the most common types of intersections found in real-life road traffic networks. The implementation of these intersections is similar with the significant difference being the rule-set governing their behavior in terms of request and responses on vehicular requests for intersection crossings. The intersection types are all based on the conflict matrix and messaging system described in the previous chapter.

### 5.2.1.1 Default Intersection

The default intersection type is a type of intersection that applies no rules to traffic crossing the intersection. While the default intersection implements the conflict matrix and messaging system for collision free crossings of the intersection it does not apply any rules to traffic wanting to cross the intersection. This type of intersection is meant to be used for constructing intersections that are collision free like freeway on-ramps and off-ramps where vehicles entering or exiting the intersections do so via presorting to a predefined traffic lane on the road they are currently driving on.

### 5.2.1.2 Precedence Intersection

The precedence intersection as its name suggests allows certain roads to have precedence over other roads. This type of intersection implements the conflict matrix and messaging system but also applies a second matrix containing predefined intersection precedence rules for incoming traffic. These precedence rules are generated by the intersection based on the priority of the incoming roads. When all incoming roads have the same priority right of way precedence is applied on the intersection. Right of way precedence is based on two rules as described below.

1. Right of way should always be given to vehicles approaching on the right hand road.
2. Right of way should always be given to vehicles approaching on the opposite road when making a left turn and the vehicle on the opposite road wants to make a right turn or intends to go straight ahead, unless the opposite road has a lower precedence.

The precedence rules are only applied to determine which vehicle has precedence when two or more vehicles want to cross the intersection and have conflicting routes. When multiple vehicles intend to cross the intersection and their routes do not conflict the precedence rules are not applied, this to mimic real-life driving behavior.

### 5.2.1.3 Traffic light Intersection

The traffic light intersection uses cycles of traffic light combinations to regulate collision free traffic flows over the intersection. The traffic light intersection implements the messaging system and collision matrix. An algorithm is implemented to automatically generate traffic light combination patterns that enable the maximum possible flow of traffic over the intersection allowing for multiple green lights.

The generation of collision free traffic patterns is achieved by applying the following sequence of steps. The first step consists of generating a traffic light for each incoming lane. In the second step, a matrix is constructed that notes all conflicts between traffic lights, this matrix is a condensed form of the route conflict matrix.

The third step then evaluates each row of the traffic light matrix and removes any rows that have conflicting routes enabled or that are a duplicate of other roads. Finally, each of the remaining rows is converted to a cycle and the red, amber and green light periods are defined.

#### 5.2.1.4 Generator / Exit

A generator intersection is an intersection that produces cars in a stochastic manner based on pre-specified rates thereby populating the road network. The exit on the other hand destroys vehicles that are no longer wanted on the road network. The generator part of the intersection allows the user to configure the probability of a certain type of vehicle to be generated at a certain time step and the amount of vehicles generated per hour. The amount of vehicles can be configured over a twenty-four hour period allowing for the creation of fluctuating traffic densities on the road network. The generator part of the intersection does not provide any default settings this in order to prevent 'un-configured' generators from influencing the simulation results.

The exit part of the intersection allows vehicles that are no longer required to leave the simulation environment. The exit does this by destroying vehicles that turn onto it from a connected road. While the exit signals the end of the modeled part of the simulation environment it restricts the amount of vehicles that can leave the simulation environment to the flow of the connected road. This results in realistic behavior by allowing traffic jams to occur near exits when the density of vehicles exceeds the maximum possible flow of the exit.

#### 5.2.2 Vehicle Types and Behaviors

Vehicles in the simulation all implement the same basic set of functionalities as described in the previous chapter. The difference however between these vehicles is their purpose and the manner in which they interact with routing system. The standard vehicle movement process that is implemented in all default vehicles in the simulation environment is depicted in Figure 30.

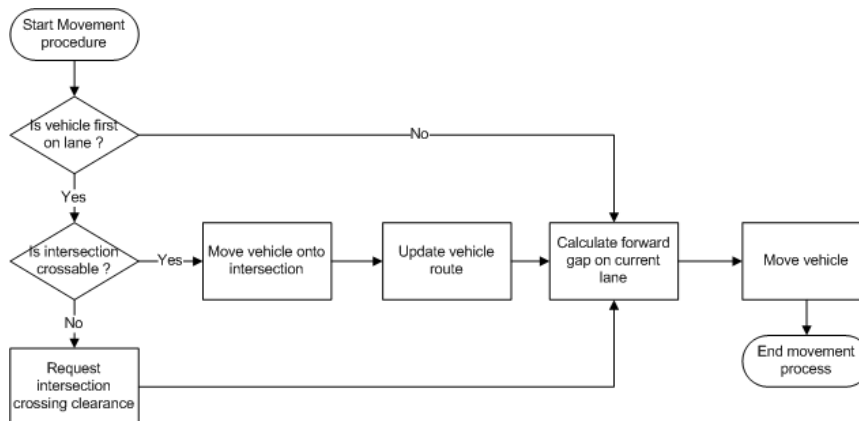


Figure 30: Vehicle movement procedure flow chart

The lane changing procedure employed by all default vehicles is depicted in Figure 31. As described in the previous chapter vehicles are free to change lanes until they encounter the lane critical zone. The critical zone limits the possibility for vehicles to change lanes by allowing only lane changes beneficiary to the route of the vehicle. This critical zone lane changing procedure is the only procedure that could lead to deadlocks in the simulation environment when two vehicles are forced to change lanes and both want to change to the lane the other vehicle is currently in. This deadlock is resolved by swapping the vehicles.

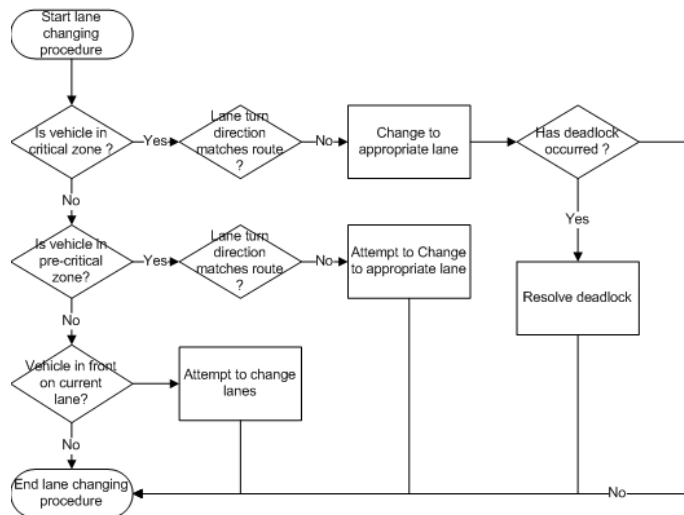


Figure 31: Vehicle lane changing procedure flow chart

CBPRS participants driving through the city need to communicate with the intelligent lampposts in order to upload their positional information and route delay information and download routing information. As described in chapter three intelligent lampposts are positioned at all intersections throughout the city, which means that vehicles can only communicate with the CBPRS near those intersections. Signal interference and other communication errors were explicitly ignored for the purposes of this thesis. However, the maximum transmission range of the intelligent lampposts was set at 40 meters according to the constraints in chapter one. To ensure that a certain amount of realism remained vehicles are only allowed to communicate with the lamppost when their position plus current speed equals the length of the lane. This in means that vehicles driving on a highway of 30 blocks with a maximum speed of five blocks per second can communicate with the intelligent lamppost at the end of the road when they reach block 25 leading to an effective communication range of 37.5 meters from the intelligent lamppost. Communication with that lamppost can continue until the vehicle reaches block five of the next road chosen after the intersection. For inner city road the communication decreases according to the maximum speed, this mimics the increased chance of signal interference within cities.

### 5.2.2.1 Dijkstra Vehicles

The simulation environment has three default implementations of Dijkstra routing orientated vehicles. The first vehicle named 'Dijkstra Vehicle' chooses a random exit in the road traffic network as its destination and uses the routing system to request static information in order to reach its destination. The second vehicle named 'Dijkstra Parking Vehicle' chooses a random parking place within the city as its destination. The vehicle drives to its parking place by consulting the static information generated by Dijkstra's algorithm in the routing system to reach the parking place. Once the parking place has been reached the vehicle determines if it is empty and if so parks the vehicle there for a random amount of time not exceeding a user defined maximum. If the parking place is taken, the vehicle enters a process of random guessing traveling through streets in the vicinity of its destination looking for an empty parking place. After the vehicles parking time has exceed it travels back to the exit of the generator where it has been spawned.

The third type of Dijkstra routing orientated vehicle is named 'Bus'. The bus uses static routing information generated by Dijkstra's algorithm to navigate along its route of bus stops. The bus follows its predetermined bus route during the entire simulation constantly navigating the same path. The traffic condition information gathered by the bus in the form of delays is relayed to routing system at each intersection. The ant-based algorithm to optimize its routes uses this information.



### 5.2.2.2 Ant-based Vehicles

The simulation environment has two default implementations of Ant-based routing orientated vehicles. The first vehicle name 'Ant Vehicle' is a semi-participant in the CBPRS system. The vehicle provides traffic condition information in the form of delays encountered and queries the system for the most time optimal route towards the destination. However, the vehicle does not use the parking facilities but instead follows a route through the city between two exits. The second vehicle 'Ant Parking Vehicle' is a full participant in the CBPRS system. Not only does it actively support the routing system but it also interacts with the parking system. The vehicle follows the route provided by the routing system until it reaches its parking place and parks there for a predetermined period before following a path back to the generator where it originated.

### 5.2.3 Serialization

The simulation environment uses a process called binary serialization to save (serialize) and load (de-serialize) the state of the environment as configured by the user. Binary serialization evaluates the entire tree of objects within an application and saves or loads objects based on their relationship with other objects within the application.

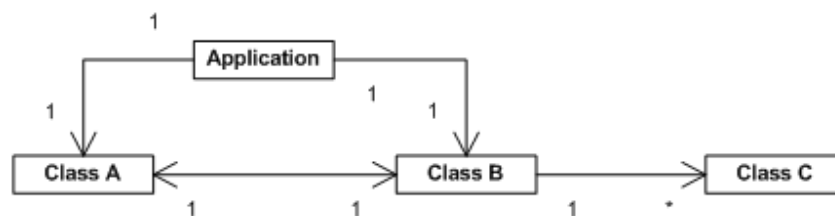


Figure 32: Simple Class hierarchy

Figure 32 displays a simple application hierarchy in which the application class holds a reference to class A and B. Class A and B hold a reference to each other and class B holds multiple references to class C. When saving or loading the information contained in the classes from disk it is vital that the references between classes remain valid in order for the program to remain in a valid state. When serializing, the binary serialization process evaluates all objects by traversing through the application tree and noting all references. The references found for each class are replaced with a unique identifier that indicates to which class the references originally pointed to. When de-serializing the object tree these references are restored so that they point to the actual classes in memory.

The major drawback to this approach is that information is stored on disk in a non-human readable format. This means that alterations to the simulation by the user must be done via the Cityscape application. However, with this drawback in mind the binary serialization process is still preferred over other serialization processes such as XML serialization. While XML serialization allows the user to manually inspect the created environment it places the burden of traversing the object tree and the elimination process of duplicate classes with the programmer making the task of saving the application state highly complex and time consuming. Furthermore, if the user is allowed to edit the simulation manually the validity of the entire environment can be compromised since there is no possibility for the simulation environment to validate the alterations made by the user.

The files resulting from the simulation have a tendency to consume a great amount of space when the application is saved during the execution of a simulation. Therefore, the application compresses the serialized files via the ZIP compression method. The compression and decompression of the files is done automatically and requires no user input whatsoever.

## 5.3 Possibilities and Limitations

The visualization is an important part of the CBPRS simulation environment and allows the user to monitor in detail the events taking places and asses the causes behind these events. From a computational viewpoint, the visualization is the most important part of the CBPRS simulation environment as it puts the greatest burden on system resources and processor time. The main focus of the CBPRS simulation environment is the simulation of vehicular movement and the interaction between vehicles and the CBPRS. Therefore, some features like those of the selection of individual cars and individual driving lanes has been sacrificed in order to optimize drawing and lower the burden on the system. The optimizations however do not mean that the information normally available to the user is lost. The information is still available although the display of data requires the navigation of a few dialogs as described in the user manual in appendix III.

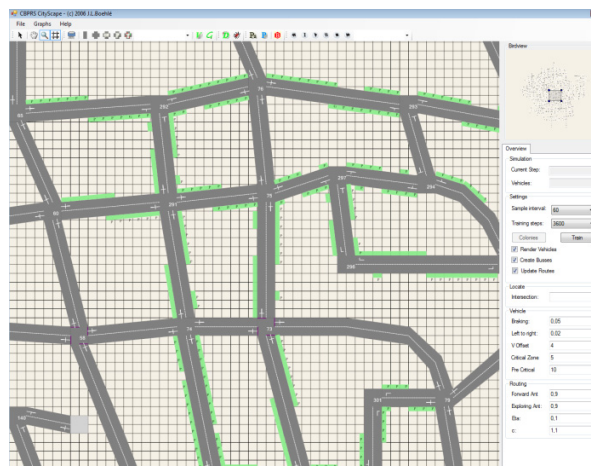


Figure 33: Cityscape GUI

Figure 33 gives an overview of the Cityscape application GUI. The GUI contains the visual representation of the road network and controls all processes related to drawing the infrastructure onto screen. As stated in previous paragraph rendering of the road network, parking places and vehicles is highly computational intensive. The Piccolo.Net library used to facilitate operations such a zooming and scrolling also contains a predefined scene graph for establishing a hierarchy for rendering objects in the correct order. While this hierarchy works, the method chosen to implement it was reasonably slow when dealing with a large number of objects such as the CBPRS. Therefore, only small portions of the infrastructure can be viewed while the simulation is running a reasonable default size has been set that allows vehicular movement to be viewed without overburdening the system beyond its capacities. Better-adapted drawing and rendering algorithms could help to increase the speed of the visualization but remain subject of further study.

The exits associated with generator intersection currently functions as the outer edge of simulation environment limiting the size of the simulation environment to the capabilities of a single computer. It is however possible to implement a fifth type of intersection namely a 'tunnel'. These tunnels could then function as gateway between CBPRS simulation environments running on two or more computers allowing vehicles to travel between the different environments. This enables the creation of even larger city environments or the coupling of one or more cities.

The maximum size of a CBPRS simulation environment depends on the capabilities of the computer on which the environment is executed. Neither the Cityscape application nor any of the layers restrict the size of the environment in any manner, allowing the user can construct maps of any size. However, it is advised not to create maps of more than a thousand intersections or more than 100.000 vehicles. These limits were found to be the upper limit of a single core three Giga-hertz processor with one gigabyte of internal memory as was used during the development of the application.

## Experimental Design

In order to find an answer to the problem stated in chapter one a simulation environment was developed to conduct experiments in order to prove or disprove the solution proposed in chapter one. The previous two chapters described the implementation and technical details of this simulation environment. In theory, it should now be possible to conduct experiments. However, before such experiments take place a proper experimental design model should be formulated so that it is clear how the experiments will be conducted and what the scientific value of those experiments will be. This chapter therefore focuses on the experimental design by addressing the topics of conceptualization, operationalization and validity.

### 6.1 Conceptualization

The most important question to ask oneself before experiments are conducted is what exactly needs to be measured. As stated in 'purpose of study' described in chapter one the goal is to determine if a causal relationship exists between a system that provides dynamic routing and parking of vehicles and optimized traffic flows within the city environment. The question that then remains however is how one specifies the meaning of traffic flows within a city environment.

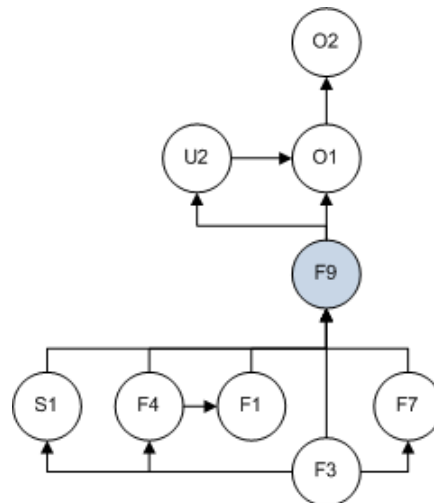


Figure 34: Variables under study

The benefits that individual users gain from the routing part of the CBPRS should express themselves in decreased travel times as opposed to a situation where they would use Dijkstra's algorithm for routing. This decreased travel time, depicted as 'U2' in Figure 34, should in turn substantiate the other benefits deducted from the CBPRS routing system. The improvement in vehicle travel times is measured on an individual basis as well as on a global basis. By executing the simulation on a map twice, once with 'DijkstraParkingVehicles' and once with 'AntParkingVehicles' under the same global vehicle load, the difference in travel times can be determined in an unbiased manner. The travel time of a vehicle is recorded as the time in simulation steps between its entry in the simulation environment and its exit. A special case however exist for vehicles that park their car upon a parking place, they record their time of parking place entry as the end of their first route and make a second entry when leaving the simulation environment. This method of measurement allows for a more detailed study of the actual travel times of the vehicles during the two different time spans of travel (once from generator to parking place in time period A and once from parking place to exit in time period B).

The traffic flow within a city, depicted as 'O1' in Figure 34, is defined as the average amount of time required by all vehicles when travelling from their point of origin to their point of destination within the simulation environment. However, the simulations executed by the simulation environment normally span a period of four hours therefore precious information is possibly lost. Two different concepts of traffic flow within the city environment are taken into account, one that represents the average amount time required by all vehicles on a four hour basis and one that represent the average amount time required by all vehicles on a hourly basis to travel between their point of origin and destination. Although people generally agree that a traffic jam occurs when the influx of traffic on a road is higher than a maximum flow for which the road section was designed, the term traffic jam still has different meanings depending on the person asked. For one person a traffic jam is a situation where vehicles are mainly standing still and moving sporadically others however might define a traffic jam as a situation in which a certain minimum speed, depending on the type of the road, cannot be achieved due to other vehicles.

A traffic jam within the simulation, depicted as 'O2' in Figure 34, is said to occur if the density of vehicles on all drive lanes of a road exceeds 80 percent. This percentage might at first glance seem too high but if one realizes that most inner city roads are single or dual lane with a relatively short road length due to a high intersection density within the city the figure is quite reasonable. It avoids classifying waiting queues for intersections as traffic jams but does not fail to recognize major congestions on the city roads.

## 6.2 Operationalization

The previous section defined the meaning of the variables under study. This section describes how these variables are measured. During each simulation, the CBPRS simulation environment collects information about individual vehicles, roads, traffic jams, the parking system and the dynamic routing system. These five sources of information not only indicate how traffic conditions within the city develop but also give an indication of usage of the CBPRS in terms of CBPRS infrastructure requirements.

Simulations within the CBPRS simulation environment minimally last for a period of four simulated hours and maximally twenty-four hours. This means that information concerning the city road infrastructure is collected in between 14.400 to 86.400 simulation steps for the infrastructure part and in between 72.000 to 432.000 simulation steps for the routing part of the CBPRS simulation environment. The information is measured by vehicles on an individual basis and represents the actual time the vehicles spend in the simulation environment. The global statistics for roads, traffic jams, the parking system and the routing systems are gathered at intervals. The default interval setting is 60 simulation steps meaning that the infrastructure part is sampled 60 times in a simulation hour while the routing system is sampled by default 360 times per hour.

### 6.2.1 Vehicular Operationalization

Vehicles travelling through the city road infrastructure can encounter many different traffic conditions. While it could be interesting to store all data concerning these situations the amount of data then generated would certainly overwhelm any person attempting to analyze the data. Furthermore, one could question if the collection of such an extensive dataset has any added value in terms of answering the research question posed in chapter two. The main interest in terms of individual vehicular data gathered from experiments lies with measurements concerning the route the vehicle has travelled.

The vehicles within the CBPRS simulation environment execute three different types of measurements while following their route through the city. The first and most obvious measurement is the measurement of time required to travel between origin of the vehicle and its destination. The route travel time is measured by accumulation of the time required to travel down roads and cross intersections. The time vehicles spend parked on parking place is not accounted for in the route travel time.

The second measurement is the number of roads travelled on between the point of origin and the destination. The number of paths that vehicles take between origin and destination varies depending on the algorithm used. While Dijkstra's algorithm will always routes vehicles via the same path the routes offered by ant-routing algorithm can differ constantly based on the current traffic conditions within the city.

This however causes discrepancies when one attempts to compare the average route travel times of different types of vehicles. Therefore, the number of roads travelled is measured by increasing a variable every time an intersection is encountered on the vehicles route.

The third type of measurement is used to measure the number of turns a vehicle has made to the right or to the left when crossing an intersection. This measurement attempts to give an indication concerning the driver's comfort of certain routes. Routes that provide a shortest time path between the origin and destination but require that the driver be constantly occupied with turning to the left over a precedence intersection might not be optimal from the driver's perspective. Turning on an intersection requires a lot more physical effort from the driver in determining if the route is clear and if all precedence situations have been checked. At every intersection the vehicle crosses, it notes if it has made a turn to the right or left while making no notes if the vehicle heads straight on over the intersection.

### 6.2.2 Operationalization of the Infrastructure

The infrastructure part of the CBPRS simulation environment collects data by sampling the current state of the CBPRS simulation environment at certain intervals. The infrastructural part only measures global variables concerning the infrastructure layer and parking system. This procedure however immediately limits to possibility for measuring variables since sampling only provides periodical snap-shots and does not collect complete datasets. The infrastructure conducts four measurements. Two main measurements during the period the simulation is active and two during and after simulation has completed.

The first measurement is concerned with measuring the number of vehicles currently present within the simulation environment. The vehicle numbers are grouped per vehicle type to give a more precise overview. This data can be used to determine a correlation between the route travelling time collected by vehicles and the density of traffic within the city environment. This measure is conducted at every sample interval by comparing the difference between the number of vehicles generated since the last sample interval and the number of vehicles that have exited the simulation environment.

The second measurement measures the status of the parking system in terms of the number of places occupied; the number of places reserved and mean parking time for all vehicles up until the current sample interval. The data gathered from this interval can help to indicate routing problems within the city if the number of reservations goes up or stays constant while the previous measurement shows a decline in vehicular density on the roads. These measurements are conducted at each sampling interval by querying the parking system for state information.

The third measurement measures the number of traffic jams currently present within the city environment. As the previous section defined, a traffic jam is said to have occurred if the number of blocks on each drive lane of the current road has an occupancy rate in excess of 80 percent. This measurement also stores the identifiers of the roads that exhibit traffic jams so that possible patterns can be determined after the simulation. The traffic jam measurements are conducted at the sample interval by querying road for the occupancy rates.

The fourth measurement calculates the mean route travel time and standard deviation of all vehicles within the simulation. This measurement is conducted on a global basis and by type. The measurement gives an indication of the flow of traffic throughout the city at the current sample interval. The representativeness of the current mean value can then easily be determined during and after the simulation. A normalized route travel time value can be constructed on the basis of the data gathered by the individual vehicles within the simulation environment. These values are calculated by collecting the route travel times of the vehicles leaving the simulation.

### 6.2.3 Operationalization of the Routing system

The routing system presents an indicator of the costs of the CBPRS in terms of communication capacity required for wireless and power-line communication. The data gathered from the routing system is acquired by sampling the state of the routing system at certain intervals. At every sample interval two measurements are conducted.

The first measurement measures the number of ants generated in period between the previous and current sample interval. This measurement is an indicator of the activity of the distributed routing system in the measured period. The determination of a correlation between the number of ant vehicles present in the simulation and routing system activity can be deduced from this measurement. The measurement is conducted by comparing the difference between the number of ants generated at the last sample interval and the number of ants generated at the present simulation interval. This measurement can be conducted without ant vehicles being generated. The measurement then measures the exploring ant generation behavior of routing nodes within the routing system.

The second measurement is concerned with measuring the average delay encountered by vehicles while travelling along the roads of the city road infrastructure. This measurement should enable the analyzer to determine a correlation between the pattern of traffic jams within the city environment and the delays reported to the nodes of the dynamic routing system. This measurement is conducted at every sample interval by querying all ant nodes in the routing system for their current delay measurements. This measurement should be ignored when no ant vehicles are generated since the average delay is then a constant representation of the function of the road length divided by the maximum speed.

## 6.3 Method of Experimentation

In order to determine if improvements in traffic flows occur when the CBPRS is used in a city environment the default situation concerning city traffic flows needs to be determined first. The default situation as opposed to a situation in which the CBPRS is active, is one in which all drivers use their own knowledge or static routing navigational systems. This group of drivers is represented in the CBPRS simulation environment by the vehicles using Dijkstra's static routing algorithm to find their way through the city or to a parking place within the city. Within the simulation environment, these vehicles always follow the theoretical shortest path in time as is determined by the function road length divided by maximum speed.

The second group of drivers is those that actively exploit the CBPRS in order to find shortest paths to their destinations or parking places. This second group of vehicles is represented in the simulation environment by the ant-based vehicles. These drivers share local traffic condition information with the CBPRS so that it can provide them with optimal routes through the city that take into account the current traffic conditions on the road.

Simulations executed with the purpose of experimenting always need to be executed twice. During the first execution, the citywide traffic flow is measured using the group of vehicles routed by Dijkstra's static routing algorithm. The second execution measures the traffic flow using a group of vehicles routed by the ant-based algorithm.

The number of vehicles generated during the entire simulation period however remains constant during all experiments. The size of the ant based vehicles group varies in order to determine the minimal group size required before any impact caused by the influence of the CBPRS can be ascertained. Via this method, it is possible to determine if the CBPRS has beneficiary effects for its participants and traffic flow throughout the city. Furthermore, the variation of the ant-based vehicle group size also gives a clear indication how the number of participant vehicles in the CBPRS relates to the improvement of overall traffic flow.

## 6.4 Validity

The experiments conducted in the following chapter generate data concerning the behavior of the vehicles within the CBPRS simulation environment. Based on the gathered information it is possible to draw certain conclusions that answer the research

question as was posed in chapter one. However, the major concern when inferring on the basis of experimental data is the validity of the data gathered.

The casual relationship that should exhibit itself during the experiments would lead to the conclusion that the incorporation of a dynamic distributed hierarchal routing system combined with a parking system, as is the CBPRS, leads to lower travel times for CBPRS participants and improves the entire traffic flow throughout the city. However, it should be ascertained that the measurements conducted during experiments reflect only those changes that are caused by the CBPRS.

As described in the procedure of experiments section, the experiments are conducted by verifying the behavior of vehicles routed via Dijkstra's algorithm and those routed via the ant-based routing algorithm. Vehicles routed by Dijkstra's algorithm to and from their parking places exhibit identical vehicular movement and lane changing behavior as the vehicles routed by the ant algorithm. The only fundamental difference between these vehicles is the type of routing system used and their parking behavior. Vehicles routed by Dijkstra's algorithm do not use the CBPRS parking system and can therefore be confronted by occupied parking places. If a parking place is found occupied, the vehicle will randomly select a next road and attempt to park there. When there are no available parking places to be found on the next street the vehicle will attempt to park on roads leading back to his original destination. If during this time the vehicle is unable to find a parking place it routes itself back to its point of origin and leaves the simulation environment. During these experiments, all further environmental parameters are kept constant to prevent unwanted influencing of the outcomes.

Once the experiments have been conducted a comparison can be made between the flows of traffic when Dijkstra's algorithm is used and the flows of traffic when the CBPRS is used. As stated above the behavior exhibited by vehicles is equal and therefore all differences between the two measured situations can directly be attributed to the influence of the CBPRS on the environment. This means that if improvements occur in the flow of traffic throughout the city, the improvement can directly be attributed to the CBPRS and thereby prove the causal relation described in the research statement.

The data gathered from the experiments and the conclusions drawn based on those experiments does however not allow for generalization of conclusions to realistic city environments. The behavior of human drivers and sophisticated communication technology within a city is unpredictable. The assumptions made in chapter two certainly would not hold within a realistic environment and would require that augmentations would be made to the CBPRS. Therefore, the conclusions drawn based on these experiments only provide an indication of possible benefits that might occur within a realistic environment if a CBPRS was ever implemented.

## 6.5 Procedure of Experiments

In order to guarantee that all experiments are conducted in the same manner a procedure of experiments is defined in this section. Each experiment is elaborated on through a series of four distinct sub-sections that all address a portion of the experiment. The four sub-sections for each experiment are purpose of the experiment, environmental settings, expected results and results of the experiment.

Via an elaboration on the purpose of the experiment the reader is informed why the experiment is conducted. In this section, the infrastructure on which the experiments are going to be executed is discussed. The environment settings elaborate on the manner in which the CBPRS simulation is configured in terms of the number and types of vehicles entering the simulation environment. This section also shows an overview image of the environment. Any adjustments that are made to the CBPRS simulation environments default settings are also mentioned here. In case of alterations to the default settings an elaboration is given on why these values were altered. The expected results section discusses the results that the author expects based on the configuration of the CBPRS simulation environment and the purpose of the experiment being conducted.

The final section of each experiment lists the results of experiment conducted and describes the outcomes. If the results from the experiment do not match the authors expectations an explanation is given that lists the (possible) causes why these were not in concordance.

## 6.6 Environmental Inputs

This section describes the individual parameters that can be altered via the simulation environment interface and how these values effect the outcomes of the simulation. The parameters are grouped into three sub-sections named simulation, vehicle and ant routing. Parameters not mentioned here are defined within the simulation environment and cannot be altered by the user.

### 6.6.1 Simulation Parameters

- **Sample rate:** The number of simulation steps between the sampling for information of the global information stored in the infrastructure and routing systems. The default value for this parameter is *60*. Please note, that decreasing this value has a negative impact on the performance of the simulation.
- **Infrastructure steps:** The delay in milliseconds between the execution of sequential simulation steps of the infrastructure part of the CBRPS simulation environment. The default value of this parameter is *100* milliseconds.
- **Routing system steps:** The delay in milliseconds between the execution of sequential simulation steps of the routing part of the CBRPS simulation environment. The default value of this parameter is *20* milliseconds.

### 6.6.2 Vehicle Parameters

- **Breaking:** The probability that a vehicle will break at random while driving through the city environment. The default value for this parameter is *0.05* percent.
- **Left to Right:** The probability that a vehicle currently driving on the left lane changes to the right lane. This probability influences the lane changing behavior of vehicles in traffic jams. The default value for this parameter is *0.02* percent.
- **V-Offset:** This parameter influences the flow of vehicles on the left lane. The default value of this parameter is *4 blocks*. Increasing this value leads to a higher flow of vehicles on the left lane while a decrease of this value leads to a higher flow of vehicles on the right lane.
- **Critical Zone:** The number of blocks before an intersection where vehicles are forced to change lanes if their current lane does not allow for continuation of their route. The default value for this parameter is *5 blocks* indicating that the critical zone starts *37.5 meters* before the intersection.
- **Pre-critical Zone:** This parameter indicates where the unrestricted lane change policy ends. This parameter also signals the beginning of a zone in which vehicles are only allowed to change lanes if the current lane does not allow the vehicle to continue its route in the required direction. The default value for this parameter is *10 blocks* and signals that the unrestricted lane changing policy ends *75 meters* before the intersection. This value should always be greater than the value specified for the critical zone.

### 6.6.3 Ant-Routing Parameters

- **Forward-Ant:** The probability that a 'normal' node generates a forward-ant with a destination within the current colony instead of a forward-ant with another colony as destination. The default value for this parameter is *0.9* percent.
- **Exploring-Ant:** The probability that a routing node generates an exploring-ant instead of a forward-ant. The default value for this parameter is *0.9* percent.
- $\omega$ : This parameter indicates the number of measurements that influences the measured delay on a certain road within the city road infrastructure. The default value for this parameter is *0.3* percent.
- **Eta ( $\eta$ ):** This parameter indicates the number of measurements that influences the average delay from the current node to the destination of a certain route. The default value for this parameter is *0.1* percent.



- $c$ : This parameter is a scaling factor used to influence the probability reinforcement strength. The default value for this parameter is *1.1*.
- $\alpha$ : This parameter is used as a scaling factor to dampen oscillations caused by the probability updating process. The default value for this parameter is *0.1* percent.

### 6.6.4 Generator Parameters

Generators placed within the city infrastructure environment generate vehicles based on a vehicle type percentage and an hourly maximum of generated vehicles. Figure 35 shows the default generator interface. This interface is dynamically constructed from the information contained within the infrastructure layer of the CBPRS simulation environment. Each generator requires manual configuration to prevent deceivable outcomes of experiments due to generators that where configured by a system default setting.

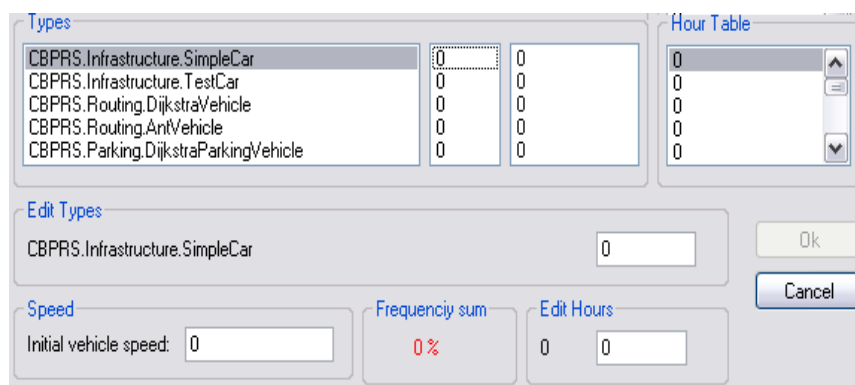


Figure 35: Generator interface

Generators operate on a twenty-four hour basis. This means that the user can specify how many vehicles a generator generates on an hourly basis. The value per hour can range from zero to 10.800. The 10.800 value can be achieved by coupling a road that has three outgoing lanes leading from the intersection and allows the generator to produce three vehicles per simulation second.

As described in the previous chapters the simulation incorporates a number of different vehicles types. The generator allows the user to specify the amount of vehicles to generate for a certain type. The user defines for each of the vehicle types a certain percentage between 0 and 100 percent. The percentages specified for the generation of the different vehicle types must sum up to 100 percent. Based on the hourly quota of vehicles to generate and the percentage specified per vehicle type the number of vehicles to generate is determined.

The generator spreads the generation of vehicles evenly over the simulated hour depending on hourly quota allocated. The type of vehicle generated at each simulation step is determined by assigning a weight to all vehicle types based on the number of vehicles of that type that need to be generated during the current hour. Via randomization, the vehicle type to be generated is chosen and the weight of the vehicle type is decreased. This procedure gives an even distribution of vehicles types generated during the hour based on the percentages set by the user.



## Experiments and Results

This chapter contains the results of five experiments conducted with the CBPRS simulation environment discussed in the previous chapters. The experiments in this chapter have been conducted based on the design described in the previous chapter and focuses on measuring the variables defined there. For each experiment conducted the purpose of the experiment, the settings used in the simulation environment, the expected results and the result gathered from the simulation environment are described. Discrepancies between the expected results and the actual results of a simulation are addressed and an attempt is made to find an explanation for these whenever they occur.

The five experiments conducted start off with relatively simple configurations and environmental settings on which subsequent experiments extend and elaborate. The first experiment conducted addresses the validity of the simulation and is conducted to determine proper functioning of the simulation environment in terms of vehicular and routing algorithm behavior. The other four experiments are then conducted on configurations of exceeding size and complexity that mimic those found in a town, village, city and metropolis.

### 7.1 Validation of the Simulation Environment

Before the experiments that evaluate the performance of the CBPRS in a number of different city environments are conducted, the validity of the outcomes of those experiments should be ascertained. The process of ascertaining the validity of the CBPRS is executed through a series of four experiments that each test a part of the functionality provided by the CBPRS through the simulation environment. The focus of these experiments lies on the proper behavior of the simulation environment; information collected from individuals is ignored when it does not serve the purpose of the validation experiment. The four experiments are listed below.

1. Validate the routes generated by Dijkstra's algorithm to verify correct implementation.
2. Validate the routes provided by the Ant algorithm and its ability to adapt to changing traffic conditions.
3. Validate the parking services offered to vehicles using Dijkstra's algorithm.
4. Validate the parking services offered to vehicles using the CBPRS.

The first experiment focuses on verifying the routes generated by the modified version of Dijkstra's algorithm used in the simulation environment. During the execution of this experiment, vehicles will be generated that use the information stored in routing tables generated by Dijkstra's algorithm to find a route through a small city environment. The resulting routes taken by these vehicles are then verified against the actual distances in the city environment to verify that the routes taken were the shortest path in time if the influences of other traffic on the roads are ignored. While this process of verification is simple, a traffic light is introduced to assess the manner in which a traffic light influences travel times.

The second experiment focuses on verifying the routes generated by the ant algorithm as described in chapter three. The verification of the routes generated by the ant algorithm is however, a two folded process. Not only should the resulting routes taken by vehicles be validated in such a manner that the traffic conditions on the points in time when the vehicle made its trip are known but also the effect the vehicle had on routes of other vehicles. This second experiment is therefore executed as a series of sub-experiments that all focus on the final behavior of the ant algorithm under different circumstances in a small city environment.

The third experiment validates the parking behavior that vehicles using Dijkstra's algorithm to locate their place exhibit. Parking places for vehicles that use Dijkstra's algorithm for their routing are not reserved via the CBPRS. These parking places are 'free-for-all' parking places that can be occupied by any vehicle. This however leads to situations wherein vehicles find parking places at the point of destination taken. If all parking places on the current lane are taken, the vehicle is required to initiate a search for another parking place in the vicinity of his destination.

This third experiment focuses on determining the success rate when locating a parking place and assesses the distance between the new parking place and the original destination of the vehicle. The outcomes of this experiment are depended on the validation of Dijkstra’s algorithm in the first experiment.

The fourth experiment focuses on the parking services provided by the CBPRS. Vehicles routed via the CBPRS always have guaranteed parking places at or near their point of destination therefore this experiment focuses only on verifying proper reservations and parking behavior exhibited by the vehicles. The routes taken by these vehicles are validated in experiment two and therefore require no further verification in this experiment.

### 7.1.1 Environmental Settings

This section contains the environmental settings for the four validation experiments. This section only contains those setting that where altered from their default simulation environment setting. For each validation experiment the simulation map or city road infrastructure is presented and details concerning its lay-out are discussed. The percentages of vehicles used per experiment are noted in Table 12, the exact number of vehicles used per experiment is described individually in this section.

**Table 12: Validation experiments vehicular distribution**

	Vehicles	Dijkstra Vehicle	Dijkstra Parking Vehicle	Ant Vehicle	Ant Parking Vehicle
<b>Experiment one</b>	1440	100 %	0 %	0 %	0 %
<b>Experiment two</b>	2160	0 %	0 %	100 %	0 %
<b>Experiment three</b>	450	0 %	100 %	0 %	0 %
<b>Experiment four</b>	450	0 %	0 %	0 %	100 %



**Figure 36: City environment for validation experiment one**

The first experiment is conducted on the map displayed in Figure 36. The city environment map contains four generators that serve both as a spawning and exit point for the vehicles. This in effect creates twelve different routes between which vehicles can travel. Dijkstra’s algorithm was initialized via the normal procedure as described in the manual (appendix III). The generators where configured to generate ‘DijkstraVehicles’ for a period of 3600 seconds with one vehicle generated every 10 seconds at each generator. Except for intersection four, which is a traffic light intersection, all other intersections are precedence intersections. All roads allowed vehicles to travel in both directions with the maximum speed fixed at one block per simulation step.

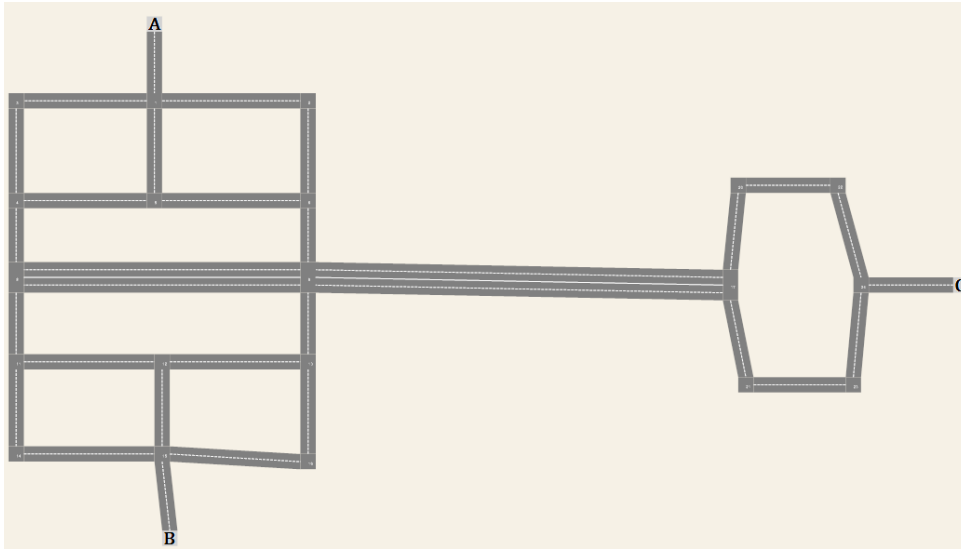


Figure 37: City environment for validation experiment two

The second experiment is conducted on the environment shown in Figure 37. This city environment contains three generators for the creation of ant-based vehicles that are labeled 'A', 'B' and 'C'. The city environment is divided into three colonies that are separated by the four-lane road in the middle of the city. The roads within the city environment all have a maximum speed of one block per simulation step with the exception of the four-lane roads that have a maximum speed of two blocks per simulation step. Before conducting the experiment, the ant algorithm was trained for 3600 simulation steps in order to allow it to settle on the most optimal routes before considering dynamic routing information. All intersections in this city environment are precedence intersections. The generators are configured to each generate 720 ant vehicles per simulated hour, which translates to one vehicle every five simulated seconds.

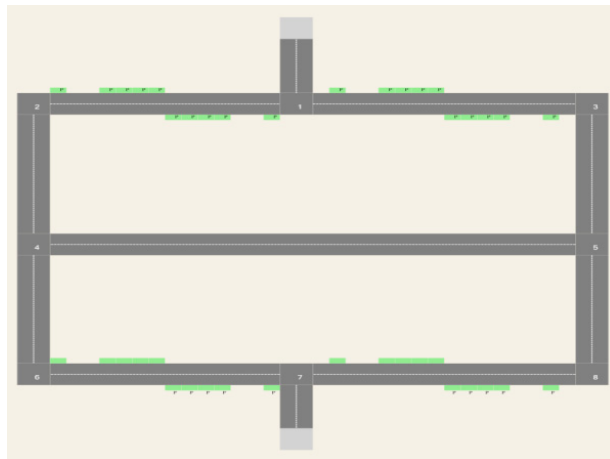


Figure 38: Environment for validation experiment three and four

Validation experiments three and four are conducted using the same environment as shown in Figure 38. The environment is designed to give optimal possibilities for visual inspection of the behavior of the parking vehicles and the reactions of other vehicles on those behaviors. In order to make the effects of parking procedures visible the validation experiments use a mixture of Dijkstra vehicles and parking vehicles during both experiments. Before conducting validation experiment four the ant-based algorithm was trained for 3600 steps the minimum available training period. The generators where both configured to generate 450 Dijkstra vehicles for a period of one simulated hour during which all measurements and observations were conducted.

### 7.1.2 Expected Results

The results from the first experiment should indicate that the vehicles follow the shortest path between their points of origin and their destinations while at the same time abiding by the traffic rules in terms of valid directions. The outcome of this experiment is expected to indicate that all vehicles travelled along the shortest paths between the generators. The high number of vehicles used on in this small environment will eventually lead to traffic jams due to the waiting time at the traffic light intersection. This in term should influence route times.

For the second experiment, the expected results are less certain at this point in time. If the ant-based algorithm operates without error, it is assumed that vehicles will follow the most optimal route through the network. Since there are no traffic light intersections present in the network, delays encountered should be minimal and should not lead to rigorous adjustments of the probabilities and routes.

It is expected that the third experiment will show that Dijkstra parking vehicles are able to find a parking place along their road of destination. Also result should indicate that the vehicle is able to safely turn direction on a two-lane road if the available parking place is located on the counter directional lane and execute the proper parking procedure. It is expected that results will indicate that a Dijkstra parking vehicle is able to locate a nearby parking place if those on the road of destination are taken or abort its search when after five attempts of visiting other roads and returning onto the original road a free parking place is not located. Vehicles are expected to find a parking place as close to road of destination as possible.

The fourth experiment deals with ant-based parking vehicles that receive their parking place from the CBPRS. This means that searching for a parking place is not required. It can naturally be expected that all vehicles will find a parking place since the number of parking vehicles does not exceed the number of parking places. The results of this experiment should also indicate that ant-based parking vehicles are able to execute the parking procedure and change lanes if the parking place lies on the counter directional side of the road.

### 7.1.3 Results of the Experiment

The results from validation experiment one show that the vehicles routed using Dijkstra’s algorithm all follow the absolute shortest paths available in the city road infrastructure. The shortest paths followed by the vehicles are shown in Table 13, manual verification shows that these routes are indeed the shortest paths available. The effects of a traffic light are clearly visible under the high amount of vehicles compared to the size of the road infrastructure in the simulation environment.

**Table 13: Validation experiment one shortest paths**

Route	Shortest Path	Distance (m)
A ↔ B	A-1-2-3-B	860
A ↔ C	A-1-4-5-C	1000
A ↔ D	A-1-4-5-8-D	1120
B ↔ C	B-3-6-4-5-C	900
B ↔ D	B-3-6-8-D	680
C ↔ D	C-5-8-D	640

In general, it can be concluded that routes that do not cross the traffic light intersection four show a steady route travel time that is close to the optimal travel time achievable considering the speed limits imposed on the roads. As time progresses the traffic light intersection becomes a bottleneck and is not able to handle the influx of vehicles. Routes that cross over the traffic light intersection ('A ↔ C', 'A ↔ D', 'B ↔ C') have a significant higher standard deviation than routes that do not. After 2000 seconds, traffic jams caused by the traffic light intersection began to affect other routes. Figure 60 – located in appendix I – depicts these events in detail. Vehicles originating at generator 'C' only cross the traffic light intersection on the routes to generators 'A' and 'B'. As the traffic jam on the road between intersections '4' and '5' lengthens it becomes impossible for other vehicles to enter the road and so the route between the generators 'C' and 'D' starts the suffer as well in terms of travel time.

The correlation shown for routes originating at generators 'C' and 'B' is caused by the effect the traffic light has on vehicle throughput. The conclusion that can be drawn from this experiment is that Dijkstra's routing algorithm as implemented in the simulation environment functions properly. Furthermore, the influence traffic lights have on vehicular throughput is significant and those effects can spread quickly over a vast area if the influx of vehicles is high enough this effect is therefore taken into account when designing city environments in the simulation environment.

**Table 14: Validation experiment two; route statistics**

Route	A → B	A → C	B → A	B → C	C → A	C → B
<b>Vehicles</b>	467	231	223	473	452	241
<b>Roads</b>	8	9	8	9	9	9
<b>Minimum</b>	85	108	89	109	119	116
<b>Maximum</b>	162	192	128	139	137	139
<b>Average</b>	96,04069	124,2987	98,99552	119,723	125,8916	124,2282
<b>Std. Deviation</b>	10,64409	15,75193	6,475524	5,88677	3,028207	4,499281
<b>Median</b>	94	120	98	119	126	124
<b>Modus</b>	94	112	94	116	125	122
<b>Correlation</b>	-0,33192	-0,46141	0,1261	-0,14428	0,165135	-0,32254
<b>R<sup>2</sup></b>	0,110169	0,212897	0,015901	0,020816	0,02727	0,10403
<b>Right turns</b>	2	3	2	4	4	3
<b>Left turns</b>	2	4	2	3	3	4

The results gathered from validation experiment two are presented in Table 14. Contrary to what was stated in the expected results the vehicles travelling through the environment of validation experiment two encountered substantial delay for the short routes on which they travelled. Intersection nine – the middle intersection in on the main road – turned out to be a traffic bottleneck. This intersection is the focal point of traffic within the environment and due to its size reached its capacity for traffic coming from generator 'A' soon after the experiment begun. Travel times for vehicles coming from generator 'A' peak quickly (see Figure 63, appendix I) after the experiment has started and slowly drop as the algorithm starts to reroute vehicles via intersection eight – the left most intersection on the main road – which leads to a normalization of travel times.

**Table 15: Validation experiment two; traffic jams**

Road	Between	Duration
9	9 ← 6	15
9	9 ← 6	14
9	9 ← 6	23
9	9 ← 6	14
9	9 ← 6	7
9	9 ← 6	7
9	9 ← 6	8
7	6 ← 2	14
7	6 ← 2	10
14	9 ← 11	6

Traffic jams detected during the experiment listed in Table 15 give clear evidence of intersection nine being the bottleneck of the environment since all traffic jams occur on roads leading to intersection nine. Overall, the differences between the opposing routes are minimal. If the additional delays encountered by vehicles in a traffic jam are omitted then the remaining discrepancies between route travel times can be explained by the application of precedence rules of intersections that in the current environment do not favor vehicles coming from generator 'A'

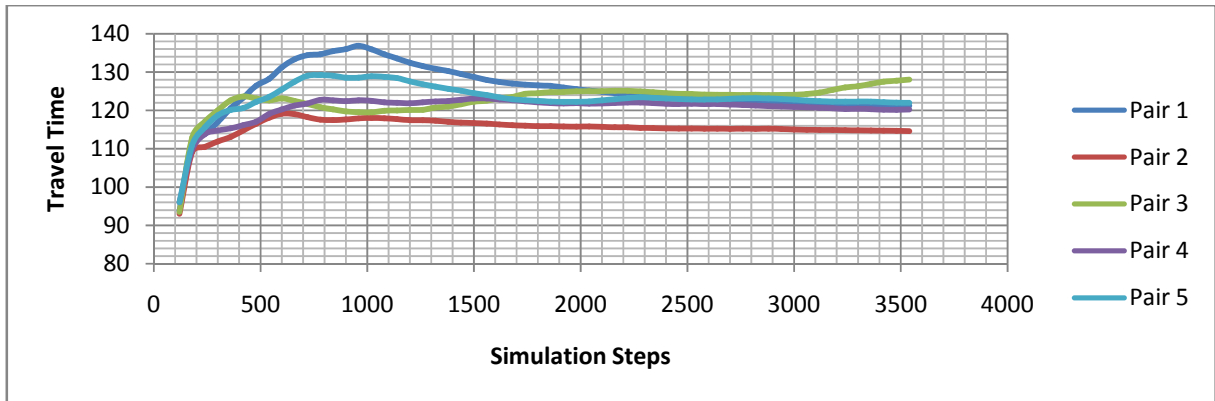


Figure 39: Ant algorithm parameter experiments

In order to justify the parameters chosen which influence the ant algorithm in terms of probability adjustments experiments have been executed that cover the complete range of possibilities allowed for the parameters  $c$  and  $\alpha$ . Figure 39 contains five of the most interesting results in light of the chosen values used by pair two. Table 16 contains an overview of the pairs including the values used and the average travel time for all vehicles during the simulation. The difference between pair two and the other pairs is significant for an environment of the size used in validation experiment two. While other pair combinations could in certain cases result in better averages in other environments pair two is chosen as the default setting for all further experiments.

Table 16: Ant algorithm pair couplings and results

	$c$	$\alpha$	Average travelling time
Pair 1	1,1	0,01	121,2045
Pair 2	1,1	0,1	114,5902
Pair 3	1,1	0,2	128,054
Pair 4	1,5	0,1	120,2199
Pair 5	2,0	0,1	121,9697

The results gathered from validation experiments three and four indicate that both the parking vehicles using Dijkstra's algorithm as those that are using the ant-based algorithm incorporated in the CBPRS function properly. Figure 40 contains an overview of the travel time developments for all vehicles during the two simulation experiments. During the initial phase of the experiments, we find that the parking vehicles in the experiments function similar. Just before simulation step 1950 almost all parking places are taken and the Dijkstra orientated parking vehicles are required to search for a parking place in the vicinity. While all Dijkstra parking vehicles remain able locate a parking the time consumed in the search increases to almost 2,5 times the original travel time. However, such behavior was expected and the vehicles performed without fault conforming to expectations.

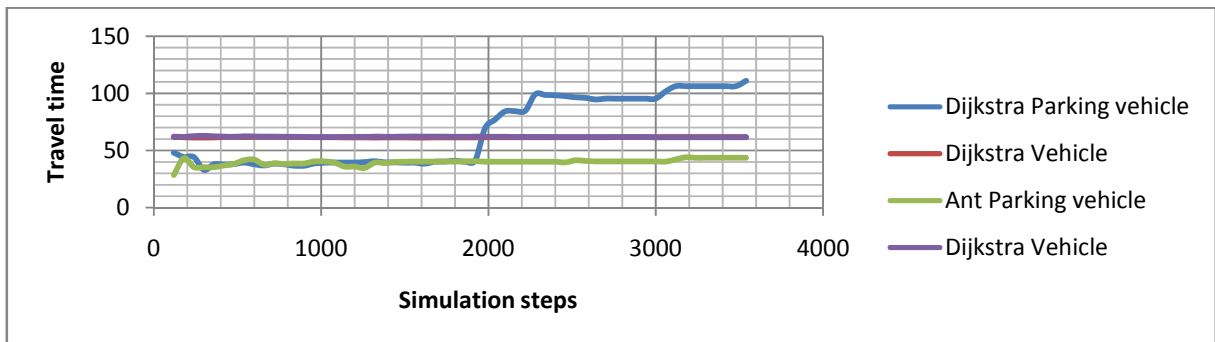


Figure 40: Travel times of validation experiment three and four



## 7.2 Parking System Performance in a Town

This experiment is the first of a series of four city environments that become progressively more complex. This first experiment tests the performance of the CBPRS in a rural town environment. The town used for the conduction of the current experiment is a fictional town inspired by small towns found in the Dutch countryside. The main characteristics of these towns are the single main road running straight through town with town streets fanning out from it. Commercial activity in the form of shops and other small businesses are usually located directly along the main road or in its direct vicinity. Another characteristic of these towns is the low amount of traffic volume passing through on an hourly basis.

The main purpose of this experiment is to test the performance and effects of the CBPRS in a town environment that has high traffic flows and lacks traffic jams due to the low traffic volume. The experiments conducted on the town environment should in theory show that when the vehicular traffic flow is high and traffic jams are non-existent both routing algorithms implemented in the simulation environment should route vehicles along similar routes towards their destinations. The differences in the average routing time for Ant-based vehicles and Dijkstra vehicles should be small. Leading to a situation wherein differences in vehicular route travel time can be explained by superior parking place scheduling process of the CBPRS used by the Ant-based vehicles.

### 7.2.1 Environmental Settings

Figure 41 displays the town environment on which this experiment was conducted. The town is divided into two colonies that are separated from each other by the main road. Two more colonies are located at the right and left edge of the town to represent destinations that lie outside of the town. Fourteen generator/exit intersections were placed at the boundaries of the town to guarantee that all roads are part of a certain route a vehicle can travel along during the simulation process. Parking places – one hundred in total – are located in the centre of the city to mimic a conglomeration of commercial activity in the centre of the town.

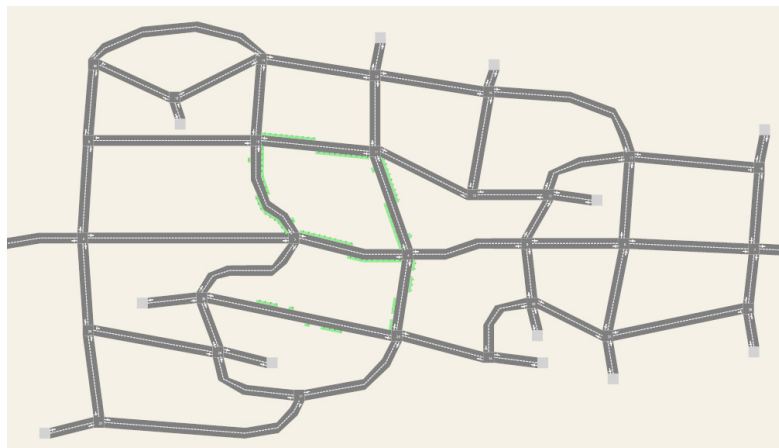


Figure 41: Town environment

The main road running in an almost straight line through the town allows vehicles to move at speed of a maximum of two blocks per simulation step while the other roads allow vehicles to move at one block per simulation step. This setup was chosen to mimic the 30 km/h zones present in most Dutch residential areas. The vehicles that were destined for a parking place were generated at the right and left edge of the city, while the other vehicles were distributed over the generators within the city to create a certain noise for the parking vehicles making their way to their places. No alterations were made to the default settings of the simulation environment.

**Table 17: Town environment vehicular distributions**

	Vehicles	Dijkstra Vehicle	Dijkstra Parking Vehicle	Ant Vehicle	Ant Parking Vehicle
<b>Experiment one</b>	100	50 %	50 %	0 %	0 %
<b>Experiment two</b>	100	50 %	0 %	0 %	50 %
<b>Experiment three</b>	100	0 %	0 %	50 %	50 %

To ensure that simulation results provide relevant results in light of the research question three experiments (see Table 17) will be carried out at first. These experiments will be conducted over a period of four simulated hours with the traffic volume in the town environment set one hundred vehicles. The first experiment will measure the optimal flow within the city by only using Dijkstra oriented vehicles. The results of this experiment then serve as an indication of how route travel times develop without the use of the CBPRS. Experiments two and three then serve to test how the results of parking vehicles using the ant-based algorithm develop. Experiment two tests the performance of the ant-based algorithm when only ant parking vehicles provide dynamic routing information. Experiment three tests the performance of the ant-based algorithm when all vehicles provide dynamic routing information. If differences in average route travel times between experiments are deemed significant further experiments will be conducted mixing the percentage of Dijkstra vehicles and ant vehicles in order to find an optimal distribution that would signify the lower boundary of participants required for the optimal performance of the CBPRS in a town environment.

## 7.2.2 Expected Results

The main point of interest during this experimentation process is the low volume of vehicles travelling along the roads of the town. This low volume leads to a situation wherein dynamic routing information supplied to the CBPRS is scarce and updates are far and few between. Since traffic jams are non-existent, ant-based vehicles are expected to report travel time delays close to the optimal delay for every road they travel along. Some variation might occur due to vehicles that are turning in the middle of the road or are busy parking. In general, these variations should be too small to alter the routes provided by the ant-based algorithm.

Therefore, it is expected that the ant-based algorithm incorporated in the CBPRS will function in an almost static manner mimicking the route travel times of Dijkstra's algorithm closely. The parking scheduling method of the CBPRS will outperform the Dijkstra's vehicles parking place procedure only when the majority of parking places are occupied. The additional delays Dijkstra parking vehicles then encounter could add significantly to the average travel time. On average however, these additional delays should not be significant enough to justify the existence of the CBPRS in a town environment with low traffic pressures. The results are therefore expected to indicate that the CBPRS does not provide additional benefits such as described in the research question.

## 7.2.3 Results of the Experiment

This section contains the most the important data gathered from the three experiments conducted with the town environment. Other data gathered from these experiments can be found in appendix I. Figure 42 contains the results of the three experiments in terms of average travel time to and from the parking place towards the generator. The graph shows the vehicles using the ant-based algorithm performing better in terms of average travel time than the vehicles using Dijkstra's algorithm. The ant-based algorithm however shows at the beginning of the simulation that the presence of ant-vehicles allows the algorithm to settle more quickly on optimal routes. In the end, these advantages disappear as the ant-based algorithm of experiment two and three converge. Further experimentations wherein the number of ant-based vehicles and Dijkstra vehicles varies are therefore not conducted.

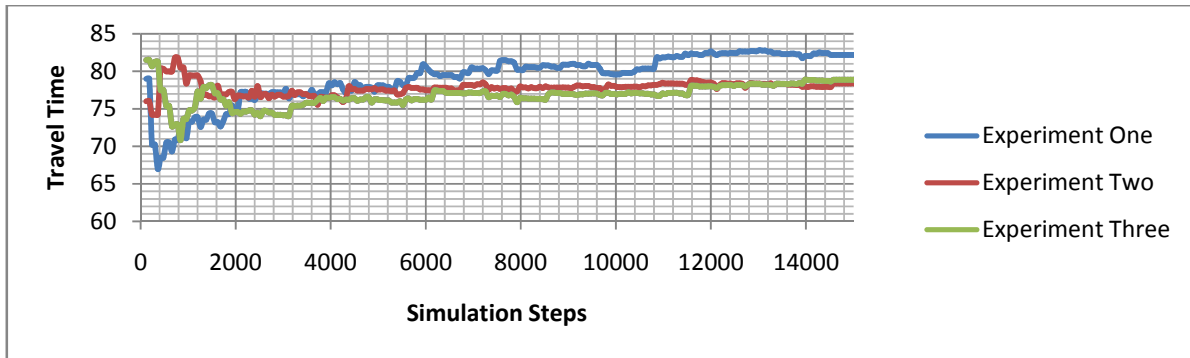


Figure 42: Experimentation result of the town environment

Dijkstra algorithm seems to be underperforming the ant-based algorithm quite significantly. The trend starts to exhibit itself around simulation step 5600 when the ant-based algorithm of experiment two and three seem to converge. Since the routes provided by Dijkstra’s algorithm are static, the average route time therefore should converge to those of the ant-based algorithm in experiments two and three as it does during simulation steps 2000 through 4800. The explanation for this behavior can be found in the parking place occupancy graph in Figure 43.

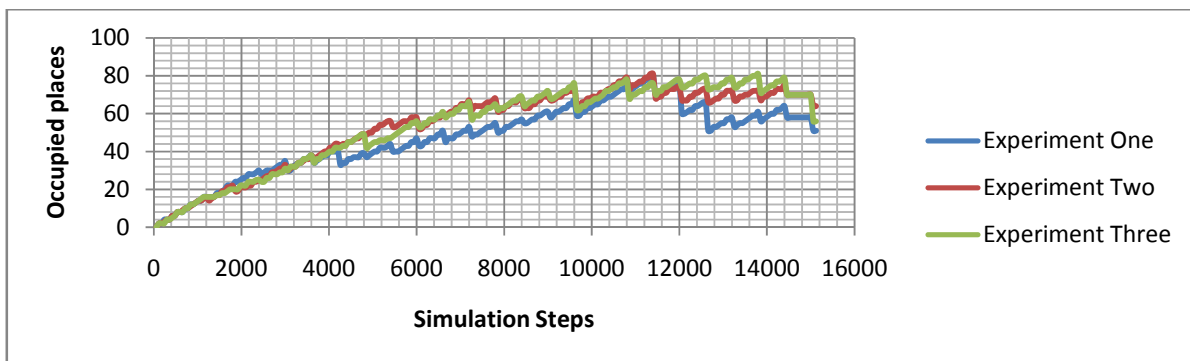


Figure 43: Parking place occupancy graph

The numbers of occupied places for the three simulations develop via nearly similar patterns due to the configuration of the generators that in each experiment output a similar amount of vehicles. While the vehicles in experiment two and three use the CBPRS that guarantees a parking place the vehicles in experiment one where forced to find one themselves. This means that as the number of occupied parking places increases during experiment one the chance that parking places on the desired road are all taken and a search has to be conducted in order to find a free parking place. As Figure 41 shows the parking places in the town environment are all clustered together in the center of town meaning that any search conducted by a vehicle would find a free parking place with minimal delay. Once the number of vehicles parked in experiment one drops – around simulation step 12800 – the average travel time starts bending slowly back towards the average route time of experiments two and three. The slightly higher average route time for Dijkstra parking vehicles can therefore almost entirely be explained by the manner in which the parking occupancy rate develops. The other variations between the three experiments can be explained by random process such as a random vehicular driving behavior, other vehicles and the time to vehicle chooses to park once it has located its parking place.

Taking all these factors into account it can be stated that the CBPRS does not provide benefits over Dijkstra’s algorithm. Differences that do occur are too small to be of any significance and do not justify the investments required to implement a system such as the CBPRS. This experiment further reveals that the ant-based algorithm such as implemented in the CBPRS is able to route vehicles correctly through an environment that is characterized by low traffic volume and high traffic flows resulting in the same performance as Dijkstra’s static routing algorithm.

## 7.3 Parking System Performance in a Village

The results of the previous experiment indicated that a town environment with a reasonably simple infrastructure and low traffic volumes did not provide an environment for the CBPRS to show added benefits. In this experiment, a slightly more complex village environment is used with higher traffic volumes. The main characteristic of the village environment used is the presence of multiple main roads leading into the city, the division of the village into multiple areas and the more even distribution of parking places over these areas. The volume of traffic encountered in this village is also larger than was present in the rural town.

The main purpose of this experiment is to determine how the CBPRS will react when confronted with an environment where multiple paths along main roads can be taken to reach a certain destination. The increased traffic volume present in the village further affects the performance of the CBPRS as a whole. The experiments conducted on this map should show how great these effects are and if an optimal distribution of vehicles can be found under which the CBPRS performs reasonably in comparison to a situation where Dijkstra's algorithm is used.

### 7.3.1 Environmental Settings

The village environment on which the experiments in this section are conducted is depicted in Figure 44 shown below. This fictional village is divided into four different residential areas by the main road laid out as a plus sign running through the village. This environment in comparison with the previous town environment is almost twice as big. The village environment is also more complex including multiple traffic light intersections, bus routes and speed variations. Parking vehicle generators are placed at the end of the main roads. These generators – colored gray – are distinguishable from the other by the fact that they are the most outlining generators. Other generators are used to generate the non-parking vehicles. The total amount of vehicles generated for this experiment is 250 per hour. This amount of vehicles will ensure that interaction between vehicles will increase leading to possible traffic jams and increasing the likelihood of other vehicle related travel time delays.

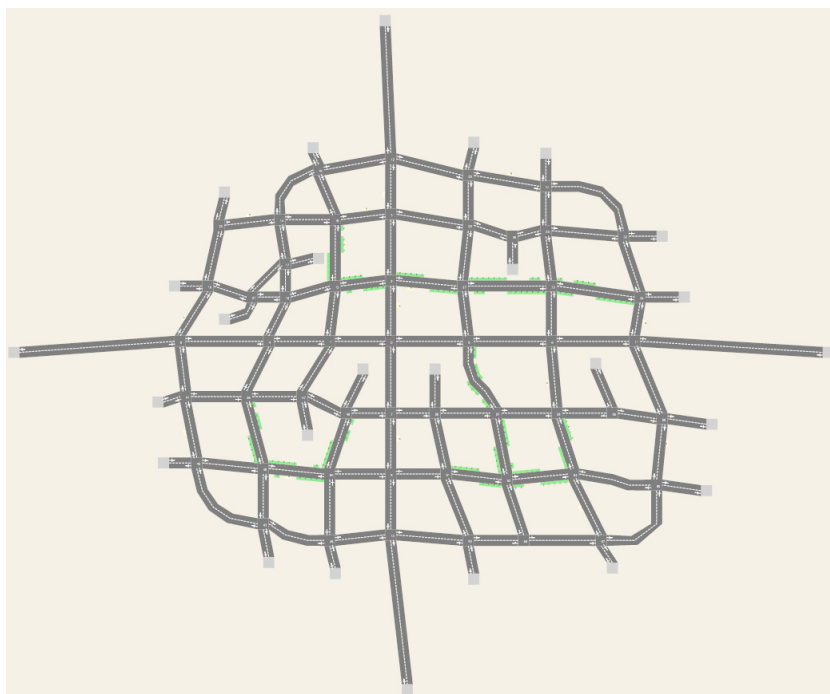


Figure 44: Village environment

The village environment contains only parking places, this stems from the fact that it is highly unlikely to find a parking garage in Dutch villages. The parking places are distributed over the four residential areas of the village and are positioned in such a manner to mimic the conglomerations of shops in a certain area. The northern two residential areas have parking places along a road running parallel along the main road while the southern areas have u-shaped areas with parking places. Parking places are not positioned along the main roads in order to investigate the effect this has on the performance of ant-based vehicles that are guided along the main road when traveling between sectors. The effect of main roads on ant-based vehicle performance will be subject to a more detailed study in the following experiments when the number of areas/colonies increases.

**Table 18: Village environment vehicular distributions**

	Vehicles	Dijkstra Vehicle	Dijkstra Parking Vehicle	Ant Vehicle	Ant Parking Vehicle
<b>Experiment one</b>	250	80 %	20 %	0 %	0 %
<b>Experiment two</b>	250	80 %	0 %	0 %	20 %
<b>Experiment three</b>	250	0 %	0 %	80 %	20 %

Table 18 depicts the vehicular distribution used in the three initial experiments. The first experiment will measure the normal flow of traffic when the influence of the CBPRS is non-existent. The two subsequent experiments will measure the performance of the ant-based parking vehicles when no additional route delay information is available and when full route delay information is available. If differences in average travel times are significant further experiments will be conducted to measure the required number of participants under which the CBPRS performs most effective in comparison to the results of experiment one. Each experiment conducted will last for four simulated hours with traffic pressures being equal during these hours. Parking vehicles will only be generated if there are free parking places available in order not to show abnormal deviations for the Dijkstra parking vehicles in the results. None of the default settings will be altered during the simulation of the experiments described above.

### 7.3.2 Expected Results

The village environment contains two interesting characteristics that influence the CBPRS. The first is the high number of roads and therefore routes available in the four colonies and secondly the increased number of vehicles within the environment. In general, ant-based vehicles should stick to the main road as long as possible since the maximum speed along this road exceeds the speed within the colonies. Once in the colonies the most optimal route towards a parking place can become subject to constant alterations. It is expected that during experiment two the ant-based algorithm of the CBPRS will underperform Dijkstra's algorithm for a period in excess of two hours (7200 simulation steps) before sufficient dynamic routing information is gathered to stabilize the ant-based algorithm and decrease travel times close or near to those experienced by vehicles of experiment three. The adjustment of the ant-based algorithm in experiment three should provide ant-based parking vehicles with optimal routes soon after the experiment has started and most roads are visited. It is expected that in this case the CBPRS will outperform Dijkstra's algorithm in terms of average travel times within two hours. The expected differences between experiment two and three will not require further experimentation where a mix of Dijkstra vehicles and ant vehicles is used to measure the minimal number of participants required for the CBPRS the function effectively.

Differences between vehicles not using the CBPRS and those that are using the CBPRS are expected to be small. The difference in route travel time is not expected to exceed more than 20 simulation steps on average. The relatively small environment, the vehicle volume and the lack of major congestive features caused by highway to normal road conversions and others mainly cause this. These expected results would indicate that an investment in a system such as the CBPRS in a village environment cannot be justified by the small decrease of travel times.

### 7.3.3 Results of the Experiment

This section contains the most the important data gathered from the experiments conducted with the village environment, other data gathered from these experiments can be found in appendix I subsection two. Figure 45 contains an overview of the average travel time during the simulation for all three experiments conducted. Experiments one and three have an almost identical development in terms average travel time. Around simulation step 7000 the average travel time for vehicles using Dijkstra's algorithm starts to increase steadily.

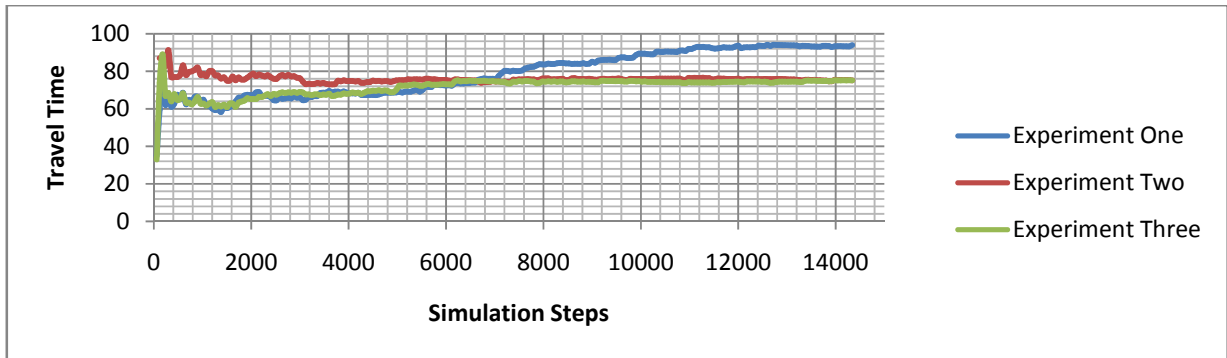


Figure 45: Average route time for parking vehicles per experiment.

Around the same simulation step the lack of data provided to the CBPRS in experiment two has been overcome by use of the information provided solely by the ant-based parking vehicles. From there on the average travel time for parking vehicles in experiments two and three remains almost equal. Thus once sufficient roads have been traveled the ant-based algorithm is able to converge on the most optimal routes with minimal delay information when the traffic volume within the environment remains constant.

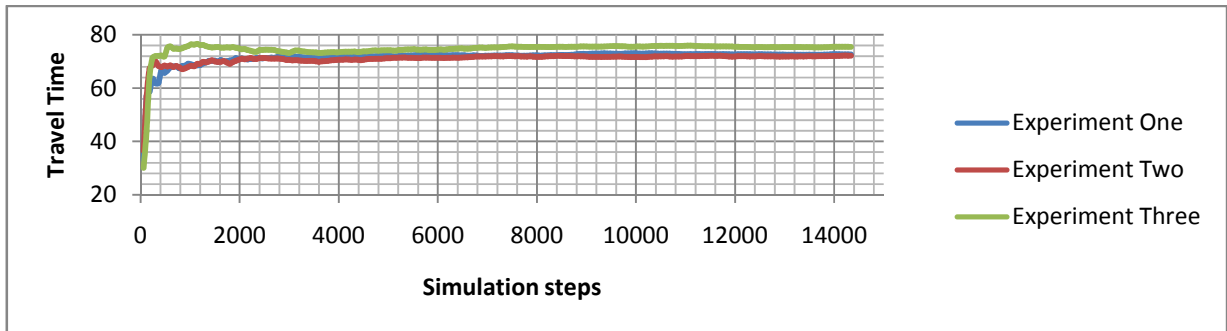


Figure 46: Average route time for non-parking vehicles per experiment

Figure 46 contains the average travel times for vehicles that are passing through the village without parking. While the average travel time for these vehicles do not vary much over the three experiments two conclusions can be drawn. Firstly, in the case of experiment three that used only ant-based vehicles the average travel time was longer than that for experiments two and three. The cause of this was traced back to the traffic light intersection placed on the intersection of the two main roads running through the city. Ant-based vehicles are guided via the main roads when travelling between colonies. This causes in the case of experiment three a large increase of vehicles travelling via that particular intersection. The traffic jams (see appendix I) resulting from this increase in traffic cause the increase in average travel time. Secondly, the average travel time for vehicles guided by Dijkstra's algorithm is less in experiment two than in experiment one although the difference is marginal. This indicates that ant-based parking vehicles have a positive influence on the flow of traffic within the village environment. If that effect is caused solely by the layout of the village infrastructure or by the ant-based algorithm of the CBPRS must become clear from the results of following experiments.

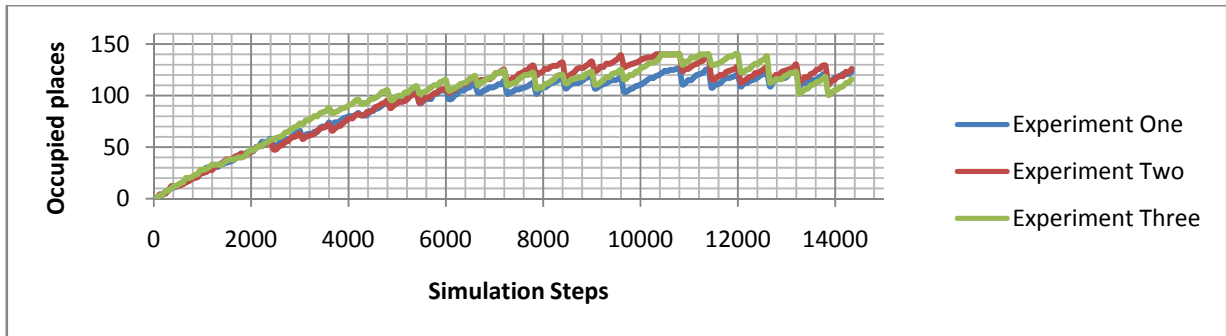


Figure 47: Parking place occupancy graph

While the difference in average travel times is certainly larger than in the previous town environment the benefits gained from the CBPRS would probably only exhibit themselves on a Saturday when most people go shopping and the demand for parking places in the commercial areas of the village raises to the levels as used in the three experiments. In terms of effectiveness and performance of the CBPRS the results of these experiments clearly show that expected benefits in terms of increased traffic flows and reduced travel times for individual participants are visible, although the former remains insignificant during. The goal of reducing traffic jams however has not been met in this environment. While in experiment one there are no reported traffic jams experiment two and three report some (see appendix I). The causes for these are explained above and these traffic jams are expected to disappear when the simulation environment grows larger with more main roads for ant-based vehicles to travel on.

## 7.4 Parking System Performance in a City

The city environment offers in comparison to the two previous experiments extended possibilities for the CBPRS to reroute vehicles over alternative routes towards their point of destination. The city environment for this experiment is modeled after the city of Apeldoorn that has a ring road along its periphery. The city environment can further be distinguished from the rural town and village by the fact that traffic volumes are reasonable high and remain high for almost the entire duration of the day sometimes stretching well into the evening. A further characteristic is the presence of multiple distinct neighborhoods within the city and the clustering of commercial activity within a certain neighborhood of the city.

The purpose of this experiment is to test the performance of the CBPRS in the city environment. In comparison with the previous environments, the city environment is the first with a scale and traffic volume large enough to allow the CBPRS to distinguish itself from Dijkstra's algorithm in a significant manner. The main goal of this experiment is to determine how significant the difference between the CBPRS and Dijkstra's algorithm will be. This experiment will also attempt to find the most optimal number of participating vehicles in relation to the total number of vehicles under which the CBPRS shows significantly improved performance, if such performance could indeed be determined in the first place.

### 7.4.1 Environmental Settings

The city environment on which the experiments discussed in this section are conducted is shown in Figure 48. The city environment although fictional was based upon the layout of main roads of Apeldoorn, a large city in the Netherlands. The red dots present on the environment map represent intersections that are part of the 80 km/h ring road running along the periphery of the inner city. The yellow dot represents main roads running through the city. Together these red and yellow marked intersections of the main road network of the city are meant to allow vehicles to travel quickly from the outskirts of the city into the city proper. The colored areas present in the map are the colonies generated by the ant-based algorithm that is incorporated in the CBPRS.



Figure 48: City Environment

The parking facilities present in the city have been distributed evenly over the inner city colonies and two parking garages have been placed in the purple and green colonies in the middle of the city. The total number of parking place in the city environment is 750, in which both garages account for one hundred parking places each. All parking places are located within the inner city. Parking vehicles are generated at generators placed at the outer edge of the environment – these generators can be recognized by the directional information displayed alongside. Other vehicle types are generated at one of the 75 other generators spread across the city environment. During the simulation period of four hours, each of these 75 generators will generate 90 vehicles per hour resulting in a flow of 6750 vehicles per hour.

Five bus routes are present within the environment. These bus routes guide busses through the interiors of inner city colonies in order to collect additional delay information for the ant-based algorithm of the CBPRS. The busses do not travel via the main roads since the traffic volume on those roads should provide sufficient delay information for the ant-based algorithm to function properly.

The experiments will be conducted following the procedure defined for the previous experiment. The experiment one will measure the average time when only Dijkstra oriented vehicles are used. The two subsequent experiments will test the performance of ant parking vehicles with non-parking Dijkstra and ant-based vehicles. If the outcomes in terms of average travel times between experiments two and three are significant, further experimentation will be conducted in order to find a distribution of vehicles under which the CBPRS performs optimally. No alterations were made to the default settings of the simulation environment.



## 7.4.2 Expected Results

The size of the city environment is considerably larger than the size of the previous environments. This increase in size has created a more complex and extensive main road network that enables the ant-based algorithm of the CBPRS to reroute ant-based vehicles via more possible alternative routes if traffic jams occur in certain parts of the environment. In theory, the ant-based algorithm should outperform Dijkstra’s algorithm in terms of average travel time. The increased distribution of vehicles, that results from an increase in main roads, should also lead to fewer traffic jams and lower travel times for non-participants.

The number of parking places available in the environment is significantly higher than the number present in the previous experiment. The main interesting feature of this environment is the two parking garages that both allow one hundred vehicles to park. This large capacity of both garages will create a situation wherein Dijkstra oriented vehicles are less likely to initiate an extensive parking place search. The average travel times for Dijkstra oriented parking vehicles that desire a parking place near the garages should therefore be less than the average time required by vehicles that want to park in other areas of the city.

The size of the environment and parking place configuration only influence the outcomes of the experiments if the number of vehicles also increases. The increase in number of vehicles has the most profound effect on the ring road (red). While the ring road allows vehicles to travel at a maximum of three blocks per simulation step, almost every intersection with another main road has traffic lights. While the ant-based vehicles should divert to other main roads (yellow) once the reported delay for the roads on the ring road increases substantially, Dijkstra’s vehicles will still be travelling along the ring road. The higher maximum speed allowed on the ring road should enable it to compete with the inner city main road that allows vehicles to travel at two blocks per simulation step. The lower speed allowed on these roads is compensated by a decrease in number of traffic light intersections present. The outcomes of the experiments however are expected to indicate that during the initial period of simulation both ring road and main roads allow vehicles to travel to their destination in similar times. As time progresses, the experiments using Dijkstra’s vehicles should show significant traffic jams on the ring roads. The ant-based algorithm should enter a transition period that initially favors the main roads before dividing the vehicular load evenly over both main road networks.

On average, it is expected that the CBPRS will outperform the Dijkstra oriented vehicles. Depending on the manner in which traffic patterns develop during experimentation the differences in average travel time between parking vehicles should amount to between 40 and 500 simulation steps for all destinations not near the parking garages. For those destinations near the parking garages, the differences in expected travel time are expected to be small. The non-parking vehicles in the experiments should show that ant-based vehicles perform better and are less prone to delays caused by traffic jams. The overall expectation is that the CBPRS will generate the benefits – described in chapter one – as expected from the system.

## 7.4.3 Results of the Experiment

This section contains the results of the experiments conducted using the city environment. Appendix I contains additional statistical information and graphs used when formulating the conclusions presented in this section. Table 19 contains an overview of the vehicular distributions used when conducting the experiments. Experiments one through three represent the default experiments conducted on every environment. Experiment four is the culmination of a series of sub-experiments which goal was to find the most optimal distribution under which the CBPRS performed most beneficiary. The results gained from the experiment with this optimal distribution are described in this section. Experiment five compares the performance of Dijkstra’s algorithm and the ant-based algorithm by allowing ‘DijkstraParkingVehicles’ to reserve parking places via the CBPRS.

**Table 19: City environment vehicular distributions per experiment**

	Vehicles	Dijkstra Vehicle	Dijkstra Parking Vehicle	Ant Vehicle	Ant Parking Vehicle
<b>Experiment one</b>	7500	90 %	10 %	0 %	0 %
<b>Experiment two</b>	7500	90 %	0 %	0 %	10 %
<b>Experiment three</b>	7500	0 %	0 %	90 %	10 %
<b>Experiment Four</b>	7500	40 %	0 %	50 %	10 %
<b>Experiment Five</b>	7500	90 %	10 %	0 %	0 %

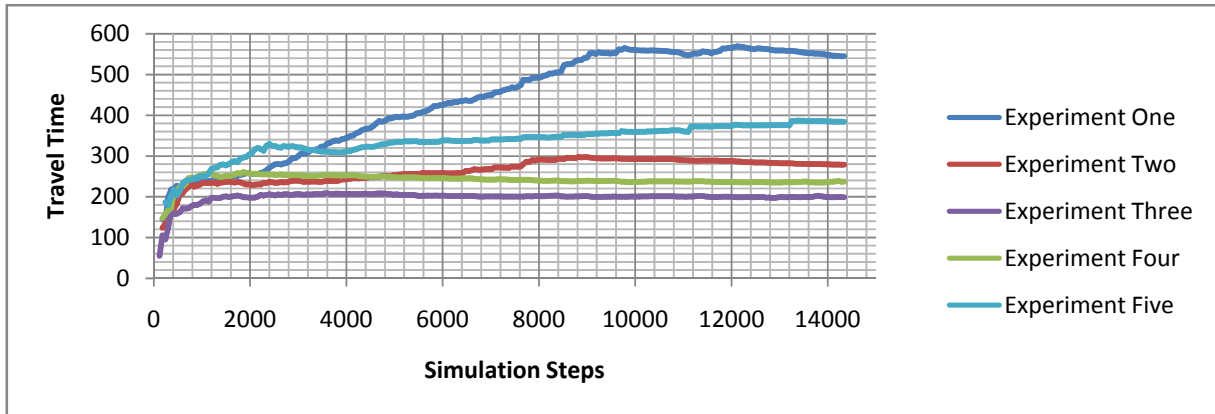


Figure 49: Average travel time for parking vehicles per experiment

The main feature of the current environment is the ring road running around the inner city. This ring road allows vehicles to move at an increased maximum speed of 80 km/h per hour giving vehicles the opportunity to travel quickly from one side of the city to another. This prevents overcrowding of the roads within the inner city that have a lower maximum speed. While such a ring road has certain benefits, it can also be the source of congestions within a city. Figure 49 depicts the development of travel times for parking vehicles in the four experiments discussed in this section. Using the same amount of vehicles for each experiment, we find that the Dijkstra parking vehicles of experiment one require a considerably larger amount of time to reach their parking places than the ant parking vehicles used in the other three experiments, as shown in Figure 49.

During experiment one all vehicles within the simulation environment used Dijkstra's algorithm in order to follow the shortest path to their destination. The shortest path towards most destinations however includes traveling along the ring road for a certain distance. Taking into account that all other vehicles also use Dijkstra's algorithm it is clear that the traffic volume on the ring road is liable to increase beyond its capacity and therefore causing traffic jams as is described in the following paragraphs. This leads to a situation where the parking vehicles of experiment one are forced to wait in the traffic jams caused mainly by the non-parking vehicles thereby increasing the travel time significantly in comparison to the other three experiments.

Experiments two and three are conducted using ant-based parking vehicles. The difference between these experiments is that experiment two made use of non-parking Dijkstra oriented vehicles while experiment three relied solely on ant-based vehicles. The CBPRS in experiment two is faced with a situation in which dynamic routing information is scarce. The ant-based algorithm is constantly struggling to find optimal routes. As the simulation progresses we see that the minimal information received is insufficient and travel times start to increase steadily. In experiment three the CBPRS is supplied with an abundance of dynamic routing information and is able to quickly find the most optimal paths through the city. The final travel time averages differ by 80 seconds indicating that a vehicular distribution exists wherein less information is provided to the CBPRS while average travel times and the associated variance outperform the results of experiment one and two.

The optimal distribution of vehicles in experiment four was found by methodically trying different combinations and comparing results. The results of this process are shown through experiment four that represents the most optimal distribution of non-parking Dijkstra vehicles and ant-based vehicles. The combination as shown in Table 19 represents the lower limit under which the benefits associated with the CBPRS show themselves to the fullest extent. The number of traffic jams within the environment remained relatively low while the average waiting time decreased significantly. Variances in travel times also decreased dramatically in comparison to experiments one and two. The mixture of non-parking vehicles however required a longer period of adjustments from the ant-based algorithm. Once the simulation progressed, the ant-based algorithm was able to find an optimal distribution that proved beneficiary to all vehicles within the city environment.

Experiment five compares the performance of Dijkstra's algorithm and the ant-based algorithm by allowing Dijkstra orientated parking vehicles to use the CBPRS parking service and so duplicating the settings of experiment two. The results of the experiments clearly show a reduction in travel times for Dijkstra orientated parking vehicles when comparing to the situation in experiment one.

When comparing the result to experiment Dijkstra’s algorithm still underperforms the ant-based routing algorithm of the CBRS. Furthermore, the travel time trend of experiment two shows small decline over the course of the experiment where experiment five shows a steadily increasing trend that has not level off at the end of the experiment indicating that travel times for experiment worsen if the experiment has progressed longer.

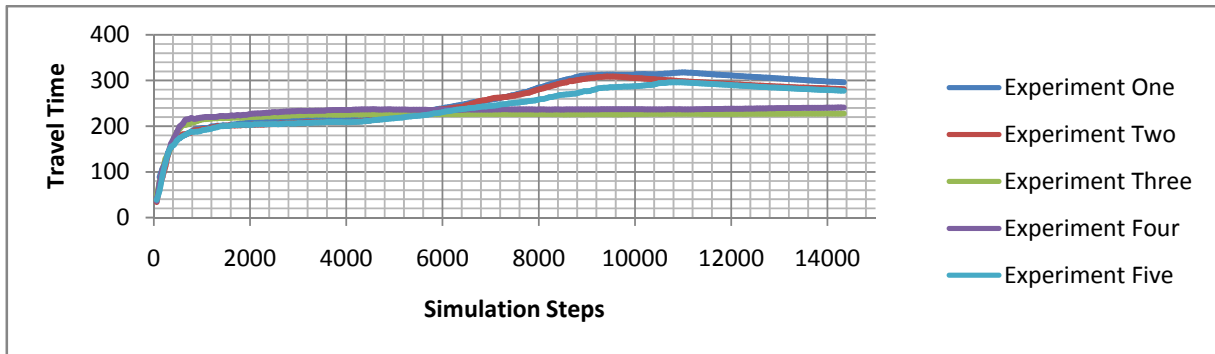


Figure 50: Average travel time for non-parking vehicles per experiment

Figure 50 shows the development of average travel times for non-parking vehicles during the five experiments. Experiments one, two and five all made use of Dijkstra oriented vehicles. These vehicles while initially outperforming those of experiment three, four show deteriorating travel times from simulation step 6000 onward. The reason for these deteriorating travel times is found in the low vehicular distribution during these experiments and the resulting traffic jams. Experiment three shows the optimal situation when all non-parking vehicles are participants in the CBPRS. After an initial period of around 3000 simulation steps the ant-based has acquired enough information to route the vehicles to their destination in the most optimal manner. Experiment four shows the same trend as experiment three however the non-parking vehicles used here are mixture of Dijkstra oriented vehicles and ant-based vehicles. While the Dijkstra vehicles have a slightly negative impact on the average travel time, the distribution achieved using this mixture outperforms the averages of experiments one and two significantly.

Table 20: Traffic jams statistic overview per experiment

	Traffic Jams	First appearance	Average Time
Experiment One	655	331	125,0168
Experiment Two	388	419	104,9739
Experiment Three	19	2648	38,33333
Experiment Four	165	251	47,86585
Experiment Five	378	628	128,9285

Table 20 contains a summarized overview of traffic jams that occurred during the five experiments. The first column contains the number of traffic jams that occurred during the simulation, column two contains the time at which the first traffic jam was reported during the experiment and column three contains the average time a traffic jam lasted. The number of traffic jams that occurred during experiment one is significant when comparing with the other three experiments. Experiments one and two demonstrate that Dijkstra’s algorithm negatively affects the distribution of vehicles and causes a great amount of traffic jams that although not very long on average do add up in total. Experiment three that relied only on ant-based vehicles shows its superiority in vehicular distribution by reducing the number of traffic jams to a mere nineteen. Experiment four that used a mixture of non-parking Dijkstra and ant-based vehicles shows that the distribution of vehicles decreases causing an increase in traffic jams. The average time a vehicle spends in a traffic jam is however less than half of that of experiments one, two and four and only nine seconds higher than experiment three. Experiment five shows that the number of traffic jams decreases when Dijkstra oriented parking vehicle are allowed to reserve a parking places. The ‘normal’ tendency for traffic jams does not disappear resulting in traffic jams with lengths comparable to experiment one. The CBPRS shows that it is able to decrease the number of traffic jams in all experiments and the average time spent waiting by improving the total vehicular distribution and traffic flows within the city.

**Table 21: Travel times of routes leading to parking garages per experiment**

	Experiment One	Experiment Two	Experiment Three	Experiment Four	Experiment Five
<b>Samples</b>	169	225	227	217	183
<b>Averages</b>	439,4083	331,9244	205,9207048	261,6497696	328,8756
<b>Std. Deviation</b>	730,3944	350,0968	84,57213	82,31718	364,65654
<b>Minimum</b>	113	110	52	116	110
<b>Maximum</b>	6797	3069	430	555	2985

Table 21 contains an overview of the statistics gathered from parking vehicles that chose to park their vehicles in one of the two parking garages within the centre of the city. Since both garages lie close to each other, the data has been combined. During all five experiments, the parking garages were never completely occupied. Therefore, none of the Dijkstra oriented vehicles was required to initiate a search that could have disrupted measurements. The fact that none of the vehicles during experiment one was required to initiate a search for a parking place did have a positive benefit in terms of average travel time and standard deviation. In general, there were no additional benefits gained from parking garages. Their effect on the large volume of traffic they attract has a negative impact for vehicles in experiment two, three and four that all have greater traffic flows than experiment one.

Taking into account the conclusions drawn in this section and in the supplementary information provided in appendix I we can state that the benefits associated with CBPRS that were expected to present themselves did materialize. The data gathered during the four experiments and town and village simulations shows that the infrastructure of the environment influences the performance of the CBPRS. The effect of traffic pressures on roads is clearly visible in the formation of traffic jams such as occurred during experiment one. Experiments two and four have shown that the number of participants in the CBPRS influences the performance of the ant-based algorithm and the measure in which benefits materialize. These factors all influenced the ability of the ant-based algorithm to function efficiently and effectively.

The benefits associated with the CBPRS did all manifest themselves to varying degrees during experiments two and four. Experiment two showed that by introducing a small amount of CBPRS participants in the form of ant-based parking vehicles average travel times for parking vehicles decreased. The average travel time for non-participants in the CBPRS decreased thereby increasing the total traffic flow throughout the city. The increased traffic flow in the environment then contributed to decreasing the number of traffic jams and the average duration of such a traffic jam. Experiment four shows the benefits of the CBPRS more strongly in comparison to experiment two. This did however require 60 percent of all vehicles to participate in the CBPRS, such a participant base can be built up slowly over time while benefits start to exhibit themselves once 10 percent of all the vehicles participate in the CBPRS. Experiment five showed that allowing Dijkstra oriented parking vehicles to use the CBPRS parking service improved travel times although results still underperformed the ant-based solution significantly. Therefore, the conclusion is drawn that a system such as the CBPRS provides significant benefits for an environment as used in these experiments and would certainly justify investments.

## 7.5 Parking System Performance in a Metropolis

The simulation of the CBPRS within a metropolis is the final and most complex experiment conducted in this thesis and should provide the information required in order to answer the research question as stated in chapter one to its fullest extent as possible. The Netherlands only has three cities to which the term metropolis applies Amsterdam, Rotterdam and The Hague. For the purpose of this experiment, we chose the city of Rotterdam as an interesting city for experimentations. The city of Rotterdam possesses a clearly defined highway network around the city supplemented by an extensive network of main roads. The natural division of the city caused by the river Meuse provides another interesting challenge to the CBPRS in terms of routing between the northern and southern parts of the city using the few available connections. While the basic layout of the city of Rotterdam is incorporated into the environment, the modeled city itself should by no means be considered a realistic representation.

The previous three experiments have shown that the benefits related to the CBPRS, as described in chapter one, are realizable. The previous experiment clearly indicated that benefits from the CBPRS would justify the investment in a city such as the one modeled. The purpose of this experiment is to validate the results and the conclusions drawn for the previous experiment by

increasing the size, complexity and traffic volume of the environment even further. The data gathered from this experiment should enable the formulation of conclusions in order to prove or disprove the possibility of assumed benefits associated with the CBPRS. The outcomes of the results of these experiments will also indicate the ability of the ant-based algorithm to cope with large environments. Thereby providing an answer to the proposed solutions for ant-based algorithm shortcomings as described in chapter three.

### 7.5.1 Environmental Settings

The metropolis environment modeled after the city of Rotterdam is depicted in Figure 51. The metropolis environment consists out of 601 intersections, more than a thousand roads and 120 vehicle generators. The metropolis environment was modeled in such manner that the environment remained within the simulation environment imposed limits.

The metropolis environment features a highway network distinguishable by the red dots that run around the entire city. This highway network forms one of the Netherlands most important stretches of highway carrying hundreds of thousands of commuters in and out of the city on a daily basis. The maximum speed for the highway was set to four blocks per simulation step resulting in maximum achievable velocity of 108 km/h. The main road network of the city depicted by the yellow dots consists mostly of four lane roads that allow traffic to move at two blocks per simulation step, allowing vehicles to achieve a maximum velocity of 54 km/h. The colored zones within the environment depict the 43 colonies formed by the ant algorithm.

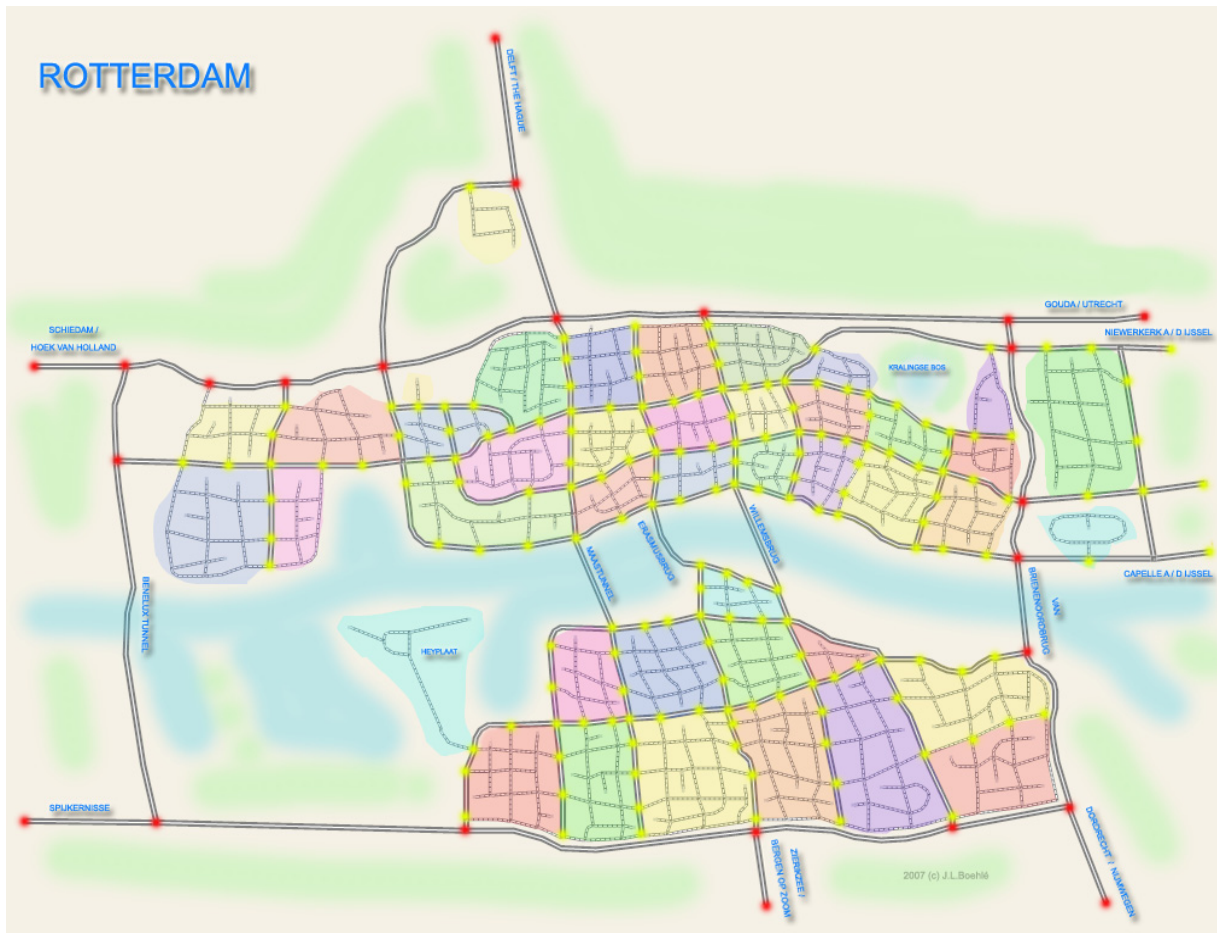


Figure 51: Metropolis environment

The city of Rotterdam is divided into more than a hundred different parking zones covering the entire municipality. In forty parking zones, the driver is required to pay a specified fee depending on the average occupancy rate of that particular zone. The forty zones together contain more than 50.000 parking places along the side of roads and in publicly owned parking garages. The center of the city – the middle section on the northern riverbank – also contains a number of privately owned parking garages that vary in size.

The environment used for the experiments, as stated before, is not a full representation of Rotterdam. Therefore, it is impossible to mimic the layout of the parking zones as present in Rotterdam. The parking places and garages in the environment are positioned near main conglomerations of commercial activity. Meaning that areas containing major shopping malls such as the city center contain a higher amount of the total parking places allocated throughout the city. This clustering of parking places also prevents Dijkstra parking vehicles from having an unfair disadvantage once a large amount of the parking places have been taken and searches are required.

The number of 50.000 parking places was not feasible in the current environment and has therefore been reduced to 1.500. This in order not to exceed the capabilities of modern hardware and simulation environment imposed limitations. The main conglomerations of parking places can be found in the center, leftmost and rightmost colonies on the northern bank of the river to represent the major shopping areas in the center, in Schiedam a neighboring municipality and the Oosterhof a large commercial center on the right most side of Rotterdam. Two other conglomerations of parking places can be found on the southern riverbank. In the northern part parking facilities represent the new developments conducted on the 'Kop van Zuid' while in the southern part parking places are positioned to represent the shopping centre 'Zuidplein' and the Ahoy events complex.

Twenty bus routes are present in the environment with each bus route covering at least five colonies as defined by the algorithm. The busses following these routes travel along the interior of the colonies supplying the ant-based algorithm with detailed information of colony interior traffic volumes and states. The bus routes are modeled to avoid the main roads as much as possible since the bulk of ant-based traffic will travel along these routes generating sufficient information for the ant-based algorithm to operate.

The experiments will be conducted following the procedure defined for the previous experiments. Experiment one measures the average travel time when only Dijkstra oriented vehicles are used while two subsequent experiments will test the performance of ant parking vehicles with non-parking Dijkstra and ant-based vehicles. During these experiments, all generators will generate 75 non-parking vehicles per hour. Ten generators positioned onto the highway entrances into the city generate a 100 parking vehicles per hour. If the outcomes in terms of average travel times between experiment two and three are significant further experiments will be conducted in order to find a distribution of vehicles under which the CBPRS performs optimally. No alterations were made to the default settings of the simulation environment.

### 7.5.2 Expected Results

The previous experiments conducted on the rural town, village and city environments have shown that benefits associated with the CBPRS materialize once the environment reaches a certain size. The metropolis environment is in terms of size paramount to the previous environments. The environment features a high-speed multiple lane highway around the city allowing vehicles to move quickly from the northern part of the city to the southern part and vice versa. The network of main roads running through the city is also quite extensive consisting of nearly 100 percent four lane roads that allow large volumes of traffic to pass through the city without delays. The layout of the environment can be considered highly suitable for the ant-based algorithm with ample opportunities to divert vehicles when routing them between city colonies. The only downside to the layout of the environment is the low number of crossings between the northern and southern part of the metropolis environment. The main question here is how this will affect both types of vehicles.

Parking place conglomerations spread over five locations in the environment should attract significant amounts of parking vehicles. These conglomerations should enable the Dijkstra orientated parking vehicles of experiment one to find a parking place without initiating extensive searches. While this configuration mimics reality it also further serves to reduce any unfair

benefits in terms of parking place locating problems for the Dijkstra vehicles. The most interesting situation that can arise from these conglomerations of parking places is the high vehicular influx into these areas that might lead to distribution problems. The high number of eight parking garages should allow for a better comparison of the effect of traffic volume and the resulting traffic jams on the average travel times. By almost guaranteeing a parking place for Dijkstra orientated vehicles, the garages further decrease any possible additional travel time by exhaustive parking place searches.

On average, it is expected that vehicles using Dijkstra’s algorithm will suffer more in terms of average travel times than ant-based vehicles. The ant-based algorithm of the CBPRS should generate a significantly better distribution of vehicles during the entire span of the simulation leading to a more optimal use of the road network and lower travel times for all participants. The benefits associated with the CBPRS of lower travel time for participants, improved traffic flows and less traffic jams should then exhibit themselves in this environment most profoundly indicating that CBPRS as a whole would certainly contribute to more optimal traffic conditions for drivers. The experiment is therefore expected to indicate that the system as proposed leads to undeniable benefits for a metropolis type of city justifying investments to be made in a system such as the CBPRS.

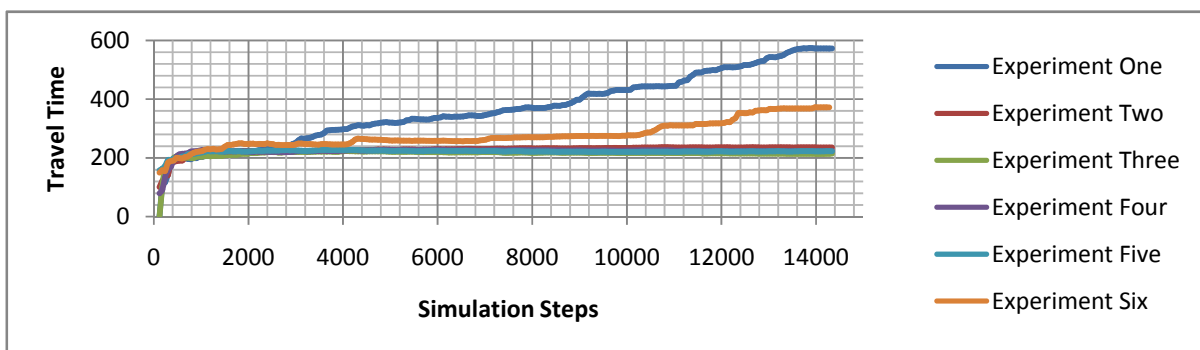
### 7.5.3 Results of the Experiment

This section contains an overview of the data gathered from the five main experiments conducted on the metropolis environment. Additional statistical information in the form tables and graphs, that are located in appendix I, support the information presented in this section and the conclusions drawn based on this information. Table 22 contains an overview of the vehicular distributions used during the five experiments. Experiment one serves as a null-measurement that gathers information in a situation where the CBPRS is non-existent. Experiments two through five have measured the performance of the CBPRS and the effects on traffic flows by varying the amount of dynamic routing information presented to the CBPRS. All five experiments were conducted using the same number of vehicles in order to compare the results of the experiments.

**Table 22: Metropolis environment vehicular distributions per experiment**

	Vehicles	Dijkstra Vehicle	Dijkstra Parking Vehicle	Ant Vehicle	Ant Parking Vehicle
<b>Experiment one</b>	10000	90 %	10 %	0 %	0 %
<b>Experiment two</b>	10000	90 %	0 %	0 %	10 %
<b>Experiment three</b>	10000	0 %	0 %	90 %	10 %
<b>Experiment Four</b>	10000	60 %	0 %	30 %	10 %
<b>Experiment Five</b>	10000	40 %	0 %	50 %	10 %

The metropolis has two major characteristics – without discussing size – that set it apart from the previous simulation environments. The first and most obvious is the extensive high-speed highway network running around the city. The capacity of the highway network enables vehicles to travel quickly around the metropolis without the chance of becoming stuck in inner city traffic. The second is the large number of multilane main roads running through the metropolis. This combination of both a highway network being able to divert vehicles from the inner city as well as an extensive inner city main road network that allows the CBPRS to reroute vehicles via a myriad of paths provides benefits to both participants and non-participants as Figure 52 and Figure 53 show.



**Figure 52: Average travel time for parking vehicles during the experiments**

The parking vehicles of experiment one relied solely on Dijkstra’s algorithm for their routing and had to search for a parking place near their point of destination. The large conglomerations of parking places distributed over the five previously mentioned zones enabled most parking vehicles to locate a parking place without much delay or extensive searches, a trend that remained stable during the entire experiment. The major problem faced by the parking vehicles of experiment one however were traffic jams. The absolute shortest path generated by Dijkstra’s algorithm in combination with the traffic flows of non-parking vehicles proved to exceed the capacity of certain portions of roads causing a large number of prolonged traffic jams as described in the following paragraphs. Experiment two used the same amount of non-parking Dijkstra vehicles but instead relied on the CBPRS for routing the ant-based parking vehicles towards their destination. Figure 52 shows effect of this in the form of significantly decreased travel times for the ant-based parking vehicles of experiment two as opposed to the Dijkstra parking vehicles of experiment one. The main conclusion that can be drawn from these results is that the manner of distribution of vehicles within the city has a significant impact on traffic flows and resulting travel times.

Average travel times for parking vehicles that used the CBPRS did not differ much in the experiments two through five. Additional experiments were however conducted to establish the effect that different mixtures of ant-based and Dijkstra vehicles have on the average travel times. The large number of main roads within the city gives the ant-based algorithm of the CBPRS a significant advantage over Dijkstra’s algorithm, a trend that was also visible during the city environment experiments. Once traffic is spread sufficiently over the entire set of main roads and highways we find that the average travel time for vehicles is diminished during each experiment conducted. Experiment three, the utopian situation, shows the most optimal results under the current traffic loads. The differences in average travel times for parking vehicles were not significant when comparing experiment four and five. During these experiments, we find that the presence of 40 percent ant-based vehicles in experiment four as opposed to 60 percent in experiment five does not provide additional travel time reductions for parking vehicles. Travel times for non-parking vehicles did decrease by thirty seconds showing that an increase in ant-based vehicles has an effect on traffic flows. The fact that parking vehicle times do not differ much is explained by the slight reduction of seven seconds that can be obtained when using 100 percent ant-based vehicles as experiment three did.

The reduction in travel times during experiment six can be attributed to the fact that Dijkstra oriented parking vehicles were allowed to use the CBPRS parking service. This delayed the creation of certain traffic jams and postponed the travel time trend visible during experiment one. When comparing the results from experiment two with those of experiment two through five we find that the ant-based algorithm performed in constant manner and significantly better than Dijkstra algorithm in experiment two.

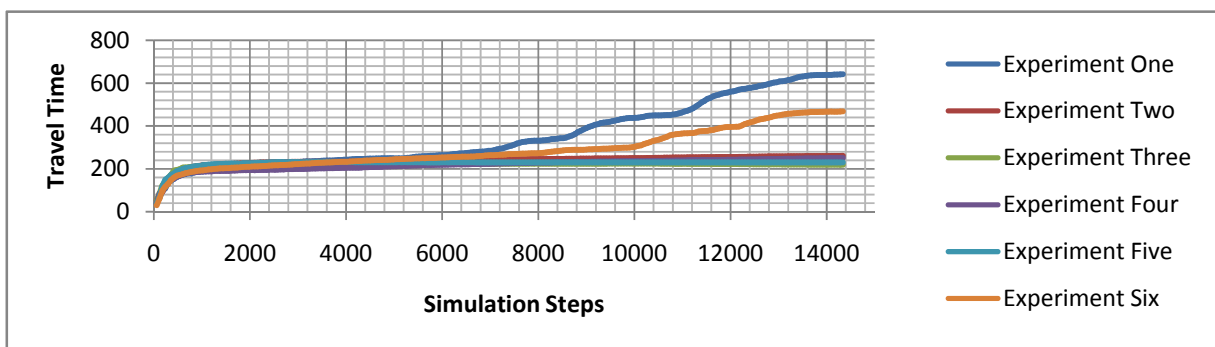


Figure 53: Average travel times for non-parking vehicles during the experiments

Figure 53 provides an overview of the average travel times for vehicles that traveled through the city without any intent of parking. The non-parking vehicles of experiment one used Dijkstra’s routing algorithm to guide them through the city environment. The routing provided by Dijkstra’s algorithm guided vehicles along absolute shortest paths that lead mostly through the inner city. This caused major traffic jams to occur around the river crossing as described in the following paragraph. The effect of this becomes visible around simulation step 5.000 where the average travel time for experiment one starts to deviate from the other experiments. Once traffic jams start to occur the constant inflow of traffic prevented the traffic jams from dissolving. Experiments two through five showed reasonably similar developments of the average travel times for non-parking vehicles. Experiment four, that used a mixture of ant-based and Dijkstra oriented vehicles, seems to outperform the



other experiments. Only when the first traffic jam appears around step 5.019 are the average travel times going up. Experiment six shows similar performance to experiment one although the increase in traffic time is progresses more steadily than during experiment one. An interesting trend visible is that when the number of participants in the CBPRS increases the average travel time for vehicles goes down underlining the effect of the CBPRS.

**Table 23: Traffic jam statistics for the metropolis experiments**

	Traffic Jams	First appearance	Average Time
Experiment One	475	907	562,76193
Experiment Two	155	1536	216,31254
Experiment Three	33	8785	147,1211234
Experiment Four	53	5019	187,6792453
Experiment Five	39	7015	164,9842412
Experiment Six	389	434	411,9674074

Table 23 contains an overview of the traffic jams reported during the five experiments. During experiment one and six, the number of traffic jams is large in comparisons to the other four experiments. The most interesting observation however is the difference in number of traffic jams and average duration between experiments one, six and two. All of these experiments used non-parking Dijkstra orientated vehicles for the normal traffic while only the parking vehicles differed.

Since traffic volumes were not altered during any of the six experiments it must be concluded that the Dijkstra parking vehicles follow routes through the city that have significant disturbing effect on the flow of normal traffic. The ant-based parking vehicles of experiment two follow different routes upon entering the inner city that avoid after a period of time the intersections most frequented by the non-parking Dijkstra vehicles. This reduces traffic jams significantly and decreases overall travel times quite significantly as shown in the previous paragraphs.

Experiments four and five show that by mixing the number of non-parking Dijkstra and ant-based vehicles traffic jams can be reduced to within 18 seconds of the most optimal situation depicted by experiment three that relied solely on ant-based vehicles. While traffic jams have decreased significantly in experiments three, four and five it can be observed that they have not disappeared completely. The cause of this was found to be the traffic light intersections on the inner city main roads that hamper the flow of traffic significantly. The effects of these was most profound around the three inner city riverbank crossings where traffic volumes were higher than elsewhere in the city. Overall, the average waiting time has decreased by almost six minutes when comparing experiment one and two. A further minute of traffic jam related delays can be deducted when comparing experiment two and three that shows that the CBPRS routing, when utilized fully, can reduce traffic jam related delays by seven minutes.

The superior ability of the ant-based algorithm incorporated in the CBPRS to distribute vehicles over a large city environment using the main road network of such an environment to its fullest extend leads to significant decreases of overall travel times for all vehicles present in the environment. We find that when using ant-based parking vehicles the average travel times decrease dramatically for both parking and non-parking vehicles. This trend of reduction continues during each experiment constantly having the most impact on the average travel times for non-parking vehicles. Reviewing the benefits associated with the CBPRS we find that during each experiment the travel time for participants has decreased and that the effects of the CBPRS on non-parking participants are even greater due to a better overall distribution of vehicles. We find that overall traffic flows improve once the number of ant-based vehicles increases and traffic jams are reduced. Therefore, it is concluded that using the CBPRS within a metropolis environment has significant benefits for all vehicles within the environment and justifies any investment made a system such as the CBPRS. Taking into account the results for the previous simulation using the city environment we find that the trend visible in that environment and the conclusions drawn are in line with results presented in this section. This indicates that results found during both these experiments were caused by the influence of the CBPRS on traffic rather than environmental circumstances that could affect the outcomes.

## 7.6 Description of findings

This section discusses the results from the experiments of the previous sections by taking the results from each individual experiment and comparing it with other experiments conducted. By discussing the data of the experiments in such a manner, trends and patterns can be found on which the conclusions of the following chapter can be established or serve as supplemental prove to the conclusions presented. References to variables made in this chapter relate to those presented in the CBPRS variable relationship diagram of chapter one.

The initial validation experiments conducted before testing the CBPRS in city environments might be seen as trivial. The results of those experiments however contributed greatly to the manner in which following experiments were conducted. The first validation experiment provided prove that Dijkstra's algorithm was implemented correctly. While this result was expected since tests had been conducted during the development phase, the significant manner in which traffic light intersections influence the traffic flows and propagated delays was not. The second experiment that validated the ant-based algorithm provided another indication of the influence of infrastructural features and the manner in which they influenced the ant-based algorithm when a precedence intersection on the main road was found to seriously delay traffic from one generator. The results of these experiments have shown that manner in which a simulation environment was developed could have serious impacts on the results of the experiments and thereby has proven the influence of variable 'F3' on the CBPRS routing service.

The simulation environment allows users to place generators within cities that spawn a certain amount of vehicles based on user-defined parameters. While this method of generating vehicles is generally preferred [29] and allows for precise specification of vehicle types and amounts to be generated, the process is highly time consuming and difficult.

Every environment developed has a certain (unknown) natural traffic flow that allows a certain amount of vehicles to move through that the environment in a realistic manner. When the numbers of vehicles configured are too low for the environment on which experiments are conducted, the performance of the routing algorithm remains equal. If on the other hand the number of configured vehicles exceeded the maximum flow of the environment, unrealistically long traffic jams and gridlocks started to develop making analysis of the performance of the routing algorithms tenuous. The effects of traffic volumes on the routing algorithms and the results of the experiments have proven the existence and influence of the variable 'F4', its effect on experimental outcomes is most profound.

The simulation environments used for the conduction of experiments were of ever-increasing size and complexity. The results of the experiments conducted on the environments show that increases in environmental size results in improving the performance of the CBPRS. During experiment one conducted in the rural town, the ant-based was unable to distinguish itself from Dijkstra's algorithm caused by the lack of alternative routes and the low volume of traffic within the environment. The village environment had a more elaborate network of main roads by which the ant-based algorithm could lead vehicles from one colony to another. While the CBPRS managed to reduce travel times for participants, the effects in comparison to Dijkstra's algorithm where insignificant. The city simulation environment provided an elaborate network of main roads together with increased traffic volumes. The size of the environment and the number of alternative routes for routing vehicles between colonies proved decisive in allowing the ant-based algorithm to outperform Dijkstra's algorithm significantly.

In the city environment, the CBPRS was able to show a decrease in travel time for participants, an increase in the overall traffic flow for all vehicles within the environment and a reduction of traffic jams and traffic jam durations. Thereby showing that with each increase in dynamic routing information presented to the CBPRS it was able to distribute vehicles in a more efficient manner over the city without causing delays. The metropolis environment underlined the conclusions drawn for the city environment. The size of the environment, extensiveness of the main road network and the high amount of vehicles travelling through show that when alternative routes are available the ant-based algorithm exploits them and thereby outperforms the solution presented by Dijkstra's algorithm significantly.

The conclusion drawn from this is that the size of the environment has a significant effect on the performance of the ant-based routing algorithm and the performance of the CBPRS as a whole. The complexity and capacity of the main road network of the environment determine if the ant-based algorithm can outperform Dijkstra's algorithm.

In small cities or those with an underdeveloped main road network, the benefits of the CBPRS will be low and insignificant. In large cities where main road networks are better developed the CBPRS delivers the benefits associated with it as was shown during the experiments on the city and metropolis environments.

The effects of the parking service provided by the CBPRS helped to reduce travel time for ant-based parking vehicles. The benefit of a guaranteed parking place at the destination point helped to reduce disturbances in traffic flows. This trend was also visible when Dijkstra oriented parking vehicles were allowed to the CBPRS parking service. While the parking service as currently implemented in the CBPRS only provides simple scheduling of parking places to participants, benefits were clearly visible once the number of occupied parking places increased. Additional benefits that could be gained from scheduling individual users and their preferences in global optimal manner could further enhance the performance of the CBPRS as a whole but have not been researched during this project. Dijkstra orientated parking vehicles found themselves significantly disadvantaged once the number of occupied parking places increased. On average, the hinder to Dijkstra parking vehicles from an inferior routing solution was more significant than the time lost when searching for a parking place. However, the influence on the outcomes of the experiments of the parking service cannot be denied and thereby proving the influence of variable 'S1'.

The number of CBPRS participants during the experiments played an important role in the manner in which the ant-based algorithm of the CBPRS was able to distribute traffic and locate optimal routes. During experiments conducted in the town and village environment, the decreases in travel time for participants were insignificant. This being caused by the lack of alternative routes via main roads and the low traffic volume. The experiments using the city and metropolis environments show in a clear manner that an increase in participants leads to a decrease in overall travel time.

While an exact relation between the increase in dynamic routing information and the reduction of overall travel time could not be found, all experiments conducted showed either improvement for parking vehicles, non-parking vehicles or both. An increase in the number participants always leads to a reduction of variance for all travel times. The results of the experiments clearly show that the number of participants influences the performance and effectiveness of the ant-based algorithm thereby proving the influence of variable 'F1' on the CBPRS.



## Conclusions and Recommendations

This chapter contains the conclusions drawn from the experiments conducted with the simulation environment in relation to the CBPRS as proposed in chapter one. This chapter also contains a section with recommendations for future research. The recommendations presented focus on improving the realism of the simulation environment, further extending the CBPRS to encompass an elaborate scheduling mechanism for parking places and recommendations for the improvement of the ant-based algorithm as presented in chapter three.

### 8.1 Conclusions

In the introduction of this thesis, we argued that static routing algorithms present in modern day navigational systems contain two hiatuses in the service they offered. The first being the inability of the static routing algorithm to react to the ever-changing traffic conditions within cities in the form of traffic jams, accidents and road maintenance. The second being the fact that once arrived at the destination the driver is obliged to find a parking place in the vicinity of his destination without further support from the navigational system resulting in an incomplete routing experience on the side of the driver. We proposed to remedy these hiatuses by introducing a City-based parking and routing system.

The CBPRS proposed consists out of two distinct services that both remedied one hiatus. The parking service schedules parking places to participants based on their preferences in terms of distance between parking place and destination, cost of the parking place per hour and the driver's destination. The centralized scheduling of parking places should enable the CBPRS to find the most optimal parking places for individuals and all participants as a whole. Parking place sensors installed in every parking place monitored by the CBPRS report on the status of the parking place via wireless communication through an intelligent lamppost in the vicinity. The routing service of the CBPRS contains a dynamic routing algorithm that is based on the food searching behavior of Argentinean ants. The CBPRS allows vehicles to reserve parking places and receive routing information via wireless communication through a network of intelligent lampposts positioned at each intersection. The intelligent lampposts exchange data via the power-line infrastructure present within the city. Participants that drive through the city provide route delay information to the CBPRS in order to allow it to respond to current traffic situations.

In order to find an answer to the research question, which questioned if a system such as the CBPRS could provide benefits in terms of reduced travel times for participants and improved traffic flows throughout the city over a situation where drivers used only static routing, we developed a simulation environment. The simulation environment allows for the modeling of city road networks consisting out of multiple and single lane roads, precedence intersections and traffic light intersections. Vehicles travelling through these environments were given the ability to change lanes depending on a set of predefined rules and follow complex routes through the environment without violating traffic rules. In this simulation environment, we modeled a complete representation of the routing service offered by the CBPRS and a reduced version of the parking service that scheduled parking places looking only at the destination of the driver. The reduction of the parking system was done to allow the focus of this thesis project to lie fully on the routing problem.

The development of the simulation environment consumed the majority of the time spent during this six month master thesis project. The development of a simulation environment from the ground up allowed for the implementation of features not possible in previously used simulation environments such as the simulation environment developed by Kroon [20]. The process of designing and implementing an entire simulation environment is a daunting task, certainly when one wants to keep the program flexible and extendable for future students that can hopefully build upon the ideas and results of this thesis document. Although the author knew this from the start, the time invested far exceeded his initial expectations. This in turn has led to sacrificing some additional functionality that would have eased the burden for future developers even more by providing a larger number of abstract components to develop new software.

However, having said this, the most important parts of the simulation environment required for the experiments conducted in the previous chapter have been realized and implemented fully. The creation of simulation environments although quite labor intensive in terms of finding the right settings for each experiment is made simple through the many different tools that assist the user in modeling. The graphical representation of the simulation environment provides clear insights into the movement of individual vehicles and the effects of rule-sets and parameters upon the simulation. The speed with which simulations can be executed in a non-visual manner has exceeded all expectations and allows users to quickly conduct a large number of experiments in order to find the optimal settings. The conclusion that can therefore be drawn is that the simulation environment has met all criteria's set at the beginning of this master thesis project and although the finish of certain parts is rough around the edges the simulation environment allowed us to execute the desired experiments under the desired conditions.

The simulation environment was used to conduct a series of experiments in settings that represented a rural town, village, city and metropolis. During these experiments, measurements where conducted that measured the effects of different environments, general vehicle counts and participant counts on the CBPRS. While the experiments provided valuable information concerning the effect of the CBPRS, the configuration of the experiments was found to be challenging and required large amounts of time before any valid results could be extracted.

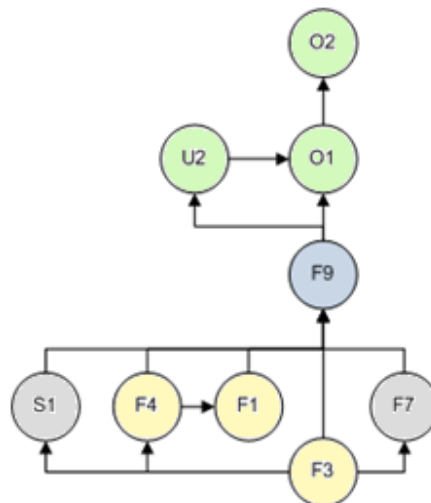


Figure 54: Relationships of variables

Figure 54 depicts the relationships of the variables under study as defined in chapter one. The variables have been color coded into groups to signify their meaning and influence per experiment. The gray variables are those that were kept constant during each experiment. The composite variable 'S1' represented the scheduling services for parking places. The manner in which parking places where scheduled was the same during all experiments conducted. The wireless transmission range of intelligent lampposts 'F7' was not altered and had the same communication range during all experiments. The yellow variables are those that were modified with each experiment. For each experiment conducted, the number of participants in the CBPRS as represented by variable F1 was altered. Variable 'F3' represents the environment that during each simulation became larger and more complex. Variable 'F4' represents the traffic pressures on the city environment that grew every time the simulation environments was expanded. Variable 'F9' represents the final performance of the CBPRS ant-based routing algorithm and is colored blue to signify its role as a pivotal point in this study. A mathematical representation of these relationships is shown in Equation 20.

$$F_9 = \alpha + \beta_1 \cdot F_1 + \beta_2 \cdot F_3 + \beta_3 \cdot F_4 + \beta_5 \cdot F_7 + \beta_6 \cdot S_1 + \epsilon$$

Equation 20: CBPRS routing relationships

We found during the experiments that once the number of roads and intersections grew the effectiveness of the CBPRS always improved. The increase in main roads was found to have a direct causal relation with the ability of the CBPRS to route vehicles over alternative routes. The effects of an increase of traffic pressures within each successive simulation environment were determined by the number of participants in the CBPRS. High traffic pressures combined with low volumes of dynamic routing information supplied to the system yielded in travel time results that while outperforming the static routing solution of Dijkstra’s algorithm were far below optimal performance. When dynamic routing information was scarce and traffic pressures high, the benefits of the CBPRS were marginal. An increase in the number of participants in the CBPRS always led to an improvement of performance and was found to be most substantial when the vehicular pressure on the environment was large. The performance of CBPRS when sporadic or no delay information was available mimicked the performance of Dijkstra’s algorithm.

$$U_2 = \beta_1 \cdot F_9 + \epsilon$$

$$O_1 = \beta_1 \cdot U_2 + \beta_2 \cdot F_9 + \epsilon$$

**Equation 21: CBPRS routing benefits**

The benefits that were expected to materialize when using the CBPRS are colored in green. Their relation to the routing service of the CBPRS is also represented in a mathematical form in Equation 21. The variable ‘U2’ represented a decrease in travel time for participants of the CBPRS. During the experiments, we found that a reduction in travel times for participants always occurs but is only significant when the simulation environment reaches the size of the city. The average variance for all participants is also reduced. This reduction of travel times and variance grows larger once the number of participants increases. The traffic flows within the city represented by variable ‘O1’ increased in the two largest city environments. The level of increase was dependent on the amount of dynamic routing information provided to the system by the participants. Increases in participants in the CBPRS always led to improvements in traffic flows. However, an exact relation between the increase in participants and subsequent increase in traffic flows could not be found and depended on the environment. Experiments showed that an increase in traffic flows within the city environment always led to a decrease in the number of traffic jams and their duration.

We can conclude that the CBPRS as tested in the simulation environment is able to decrease travel times for participants, increase the overall flow of traffic and decrease the number of traffic jams and the amount of time spent in such a traffic jam by vehicles once the simulation environment reaches the size of a city. Furthermore, we find that the CBPRS is able to match the performance in terms of travel times for statically routed vehicles in small environments while the CBPRS always significantly outperforms the static routing solution in larger environments. Based on the experiments conducted with the CBPRS we are able to conclude that the CBPRS when implemented in large city environments distributes vehicles in a more optimal manner, reducing travel times for all vehicles. We find that providing participants with a guaranteed parking place reduces travel times however the most significant reduction results from the more optimal distribution of vehicles in the environment.

*“Could a parking system that allows drivers to reserve a parking place at or near their point of destination within a city and that provides dynamic routing information – using the ant-based routing algorithm – for the most time optimal route through the city to the parking place based on current traffic conditions within that city improve the overall traffic flow in a city as compared to a situation wherein drivers are responsible for finding a parking place at their destination and use Dijkstra’s static routing algorithm to guide them to their destination?”*

**Figure 55: Research question**

To answer the research question of chapter one – listed above in blue – we can state that under the assumptions made in the chapter one and taking the limitations of the simulation environment into account that the CBPRS and the ant-based algorithm incorporated into it are able to find the most optimal routes through a city environment. Once the environment reaches the size of a city, the CBPRS also improves the overall traffic flows within that city and minimizes the number of traffic jams.

The strength of these reductions however depends on the number of participants. We found that in small environments like rural towns and villages, the CBPRS was not able to improve travel times and traffic flows beyond the performance of Dijkstra's algorithm. In larger cities and metropolises, the reduction in travel time and traffic flows was significant in comparison to the situation that resulted when using Dijkstra's algorithm.

If the benefits provided by CBPRS could be translated to real world cities, the reduction of travel times, increases in traffic flows and lower number of traffic jams would lead to certain economical benefits. The increased distribution of vehicles caused by the CBPRS would show itself even at low amounts of participants, enabling the CBPRS to convince potential participants of the benefits. The increased reachability of the city could entice business to remain in or relocate to the city. However, the simulation environment developed for this thesis project does not provide a one hundred percent realistic manner under which comparisons between the simulation and real world cities could be made. The results of the simulations however indicate that further research could indeed prove that the CBPRS is able to realize the hypothesized benefits in real world cities. The following section contains a number of possible directions on which future research could focus.

## 8.2 Future Research

This section contains two directions for further research in relation to the CBPRS and the simulation developed for the experimental phase of this research project. The directions in this section are focused on improving the quality of the experiments in order to further support the conclusions drawn based on the experiments in this thesis, since with every step towards realism the results of beneficial performance of the CBPRS in a real-life environment be proven further. A third direction for possible research focuses on the ant-based algorithm in particular without being directly related to the CBPRS however future improvements to the ant-based algorithm can only further enhance the benefits resulting from a system such as the CBPRS.

The simulation environment developed for this thesis was developed in a few months and while a solid planning and design phase has preceded the implementation of the CBPRS the time span was short and all final decisions as to how to implement the system were taken by the author after discussions with those directly involved. While in general the simulation environment performed well during the experimental phase certain shortcomings once mended could further enhance the realism factor of the environment. Small improvements that could be made are more realistic behavior of traffic lights, a greater variety in vehicle velocities without losing the realism currently incorporated into the method implemented and optimized vehicle drawing routines using hardware extensions in modern day hardware to allow for increased possibilities of visual inspection.

Driving abilities vary between humans resulting in a variety of styles from cautious to casual and from defensive to aggressive. The different driving styles influence traffic patterns [30] and can alter the flow of traffic significantly throughout cities. The simulation environment is designed to be flexible enough to allow for the incorporation of different driving styles. The question of how those driving styles should be modeled and in what manner they influence the traffic within in cities is interesting especially in relation to the somewhat static behavior of vehicles in the current simulation environment. Furthermore, the influence of different driving styles, if modeled realistically, could further support the findings of this thesis and perhaps find additional benefits currently not associated with the CBPRS.

Since the focus of this research project rested on the performance of the routing part of the CBPRS the parking place scheduling method was implemented in a rather minimal fashion considering the possibilities in this area. For the purposes of these experiments, it allowed us to focus completely on the behavior of the routing algorithms and the differences between them. However, in order to prove that the CBPRS can also provide benefits in a fully realistic environment an enhanced scheduling method is required. This method of scheduling should allow users to reserve parking places, based on their preferences, at or near their destination while also taking the collective of CBPRS participants into account. This scheduling mechanism should distribute participants in the most optimal way over the parking places available to the CBPRS while taking the preferences of all users into account. This type of scheduling could then be used to prove and strengthen the conclusions drawn in relation to benefits seen from the user perspective as well as further enhance the performance of the CBPRS as a whole. The creation of an optimal scheduling algorithm for selecting parking places on the basis of individual and group preferences would represent an entire thesis subject on its own.



The performance of the ant-based algorithm depends mainly on the formula with which probabilities are updated and the manner in which vehicles collect dynamic routing information. The algorithm presented in this thesis uses one probability update function for finding routes within colonies as well as between colonies. Colonies generally possess less nodes than there are routing nodes within a city therefore the chosen solution could lead to sub-optimal results. Future research could evaluate the probability formula chosen and test different formulas or even entirely different manners of collection and processing. Then it could also be determined if it is beneficial to employ two or more probability update formulas depending on the type of node or the position it is in.

In chapter two reference was made to the fact that the ant-based algorithm underperforms other algorithms when the number of nodes increases beyond a certain size. The solution proposed and implemented in this thesis opted for the division of the map into a number of smaller colonies and only incorporates the nodes on the periphery of the colonies into routing tables that allow travel between colonies.

The environment used in the last experiment of the previous chapter showed a great number of colonies and therefore contains a large number of routing nodes that enable travel between colonies. For cities greater than the current, the same problems of underperforming on the side of the ant-based algorithm can exhibit themselves again. Future research could therefore strive to determine the most preferable hierarchical division and even introduce new layers of hierarchy in between or replacing the current layers.



## Bibliography

- [1]. **B.Tatomir, L.J.M. Rothkrantz.** *Hierarchical routing in traffic using swarm-intelligence*. Delft : TU Delft, 2006.
- [2]. **Affairs, Department of Economic.** Welke ontwikkelingen staan ons te wachten op het gebied van Mobiliteit & ICT. *MinEZ*. [Online] Dutch Department Of economic affairs, 2006. [Cited: December 12, 2006.] <http://www.minez.nl/content.jsp?objectid=34987>.
- [3]. **Katwijk, J. van.** *Reader Kennisatelier 11 "Bereikbaarheid van steden"*. s.l. : Kennis Centrum Grote Steden, 2003.
- [4]. *On a Routing Problem.* **Bellman, Richard.** 1, 1958, Applied Mathematics, Vol. 16, pp. 87-90.
- [5]. *Flows in Networks.* **Lester R. Ford jr., D. R. Fulkerson.** s.l. : Princeton University Press, 1962.
- [6]. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* **P. E.Hart, N. J.Nilsson, B.Raphael.** s.l. : IEEE, 1968, IEEE Transactions on Systems Science and Cybernetics, pp. 100–107.
- [7]. **S.J.Russel, P.Norvig.** *Artificial Intelligence: A Modern Approach*. s.l. : Prentice Hall, 2003. pp. 97-104. 0-13-790395-2.
- [8]. *A Note on Two Problems in Connection with Graphs.* **E.W.Dijkstra.** Nuenen : Numerische Mathematik, 1959, pp. 269 - 271.
- [9]. **Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein.** *Introduction to Algorithms*. s.l. : MIT Press and McGraw-Hill, 2001. pp. 588–592. ISBN: 0262032937.
- [10]. **John M. McQuillan, Isaac Richer and Eric C. Rosen.** *ARPANet Routing Algorithm Improvements*. Cambridge : BBN Report, 1968. Report No. 3803.
- [11]. **Moy, J.** *OSPF Version 2*. s.l. : The Internet Society , 1998.
- [12]. *AntNet: Distributed Stigmergetic Control for Communication Networks.* **G.Di Cario, M.Dorigo.** 9, Brussel : IEEE, 1998, Journal of Artificial Intelligence Research, pp. 317-365.
- [13]. **EPISODE.** European Pre-Operational Implementation Survey on Further Development and Evaluation of RDS-TMC. [Online] European Union, 1994-1998. [Cited: January 23, 2007.] <http://www.rds.org.uk>.
- [14]. **Holdings, ITIS.** *Sources of Traffic Information - Estimation Technology*. [Brochure] s.l. : ITIS Holdings, 2007.
- [15]. TomTom Go Traffic. *pocketgpsworld*. [Online] [Cited: 01 23, 2007.] <http://www.pocketgpsworld.com/tomtom-go-traffic.php>.
- [16]. [lordpercy.com](http://www.lordpercy.com). *LordPercy*. [Online] [Cited: January 23, 2007.] [http://www.lordpercy.com/tomtom\\_traffic.htm](http://www.lordpercy.com/tomtom_traffic.htm).
- [17]. *Ant-Based Load Balancing in Telecommunication Networks.* **R.Schoonderwoerd, O.E.Holland, J.L.Bruten, L.J.M.Rothkrantz.** 2, Delft : s.n., 1996, Adaptive Behavior, pp. 169-207.
- [18]. *Adaptive-SDR: adaptive swarm-based distributed routing.* **I.Kassabalidis, M.A.El-Sharkawi, R.J.Marks II, P.Arabshahi, A.A.Gray.** Honolulu, HI, USA : IEEE, 2002. Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on. pp. 351-354.
- [19]. **R.Kroon.** *Dynamic Vehicle Routing Using Ant-Based Control*. Delft : TU Delft, 2002.
- [20]. **H.Dibowski.** *Hierarchical Routing System using Ant Based Control*. Delft / Dresden : TU Delft / TU Dresden , 2003.

- [21]. **E.Gamma, R.Helm, R.Johnson and J.Vlissides.** *Design Patterns - Elements of Reuseable Object-Oriented Software.* s.l. : Addison-Wesley Professional, 1995. ISBN-13: 978-0201633610.
- [22]. *A Cellular Automaton model for Freeway Traffic.* **K.Nagel, M.Schreckenberg.** Koln : s.n., 1992, Journal Physics France, pp. 2221-2229.
- [23]. **Neumann, J. von.** The General and Logical Theory of Automata. [book auth.] A. H. Taub. *John von Neumann Collected Works; Volume 5.* s.l. : Pergamonn Press.
- [24]. *Scheduling Automated Traffic on a Network of Roads.* **A.Giridhar, P.R.Kumar.** 5, Urbana, IL, U.S.A. : IEEE, September 2006, Vehicular Technology, IEEE Transactions on, Vol. 55, pp. 1467 - 1474.
- [25]. **P.Wagner, K.Nagel, D.E.Wolf.** *Realistic Multi Lane Traffic Rules for Cellular Automata.* Koln : Physica A, 1996.
- [26]. **lab, Human computer interaction.** Piccolo.NET. [Online] University of Maryland. [Cited: January 23, 2007.] <http://www.cs.umd.edu/hcil/piccolo/>.
- [27]. ZedGraph. *ZedGraph.* [Online] [http://zedgraph.org/wiki/index.php?title=Main\\_Page](http://zedgraph.org/wiki/index.php?title=Main_Page).
- [28]. **IC#Code.** SharpZipLib. *ICSharpCode.* [Online] IC#Code. [Cited: January 23, 2007.] <http://www.icsharpcode.net/OpenSource/SharpZipLib/>.
- [29]. **D.Krajzewicz, G.HertKorn, P.Wagner, C.Rossel.** *SUMO (Simulation of Urban MObility).* Berlin, Germany : s.n., 2002.
- [30]. *Microscopic traffic simulation with reactive driving agents.* **J.M.Rothkrantz, Patrick A.M.Ehlert and Leon.** Oakland : IEEE, 2001. IEEE Intelligent Transportation Systems Conference Proceedings.
- [31]. **NSS, Interview.** *Snelle opmars digitale tv en digitaal bellen.* s.l. : Synovate, 2006.
- [32]. **Lucas van Grinsven.** Google and cable firms warn of risks from Web TV. *Reuters Top News.* [Online] Reuters, February 7, 2007. [Cited: February 12, 2007.] [http://today.reuters.com/news/articlenews.aspx?type=topNews&storyID=2007-02-07T230017Z\\_01\\_L0767087\\_RTRUKOC\\_0\\_US-CABLE-WEBTV.xml&pageNumber=1&imageid=&cap=&sz=13&WTModLoc=NewsArt-C1-ArticlePage1](http://today.reuters.com/news/articlenews.aspx?type=topNews&storyID=2007-02-07T230017Z_01_L0767087_RTRUKOC_0_US-CABLE-WEBTV.xml&pageNumber=1&imageid=&cap=&sz=13&WTModLoc=NewsArt-C1-ArticlePage1).
- [33]. **Baken, prof dr ir N.H.G.** *Contra-Expertise Slagkracht door Glas.* Delft : TU Delft, 2003.
- [34]. **Committee, LAN MAN Standards.** *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* s.l. : IEEE Computer Society, 1999 / 2003.
- [35]. Streetlight. *Streetlight N.V.* [Online] 7 February, 2007. [Cited: 12 February, 2007.] [www.streetlight.nl/](http://www.streetlight.nl/).
- [36]. **group, P1901 Working.** *IEEE P1901 Draft Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications.* s.l. : IEEE-SA Standards Board, 2005.
- [37]. Growth of Broadband over Power Line to outpace Cable and DSL. *Parks Associates.* [Online] January 17, 2007. [Cited: Februari 12, 2007.] [http://www.parksassociates.com/press/press\\_releases/2007/bpl1.html](http://www.parksassociates.com/press/press_releases/2007/bpl1.html).
- [38]. **Christopher P. Garcia, Bei-jiann Chang, and Donald W. Johnson, David J. Bents and Vincent J. Scullin, Ian J. Jakupca.** *Round Trip Energy Efficiency of NASA Glenn Regenerative Fuel Cell System.* s.l. : NASA, 2006.
- [39]. *A Simulation-based Analysis of Parking System Performance.* **S.U.Randhawa, S.J.White, S.Burhanuddin.** Corvallis, Oregon, U.S.A. : IEEE, 1993. Proceedings of the 1993 Winter Simulation Conference.

- [40]. **Municipalities, Association of Netherlands.** Navigatiesystemen met aparte routes vrachtauto's wenselijk. *Association of Netherlands Municipalities*. [Online] January 12, 2007. [Cited: January 24, 2007.] <http://www.vng.nl/smartsite.dws?id=62483&ch=DEF>.
- [41]. *Statistical Physics of Traffic Flow.* **A.Schadschneider.** 101, Koln, Germany : Physica A285, 2000, Vol. 2000.
- [42]. **M. Dorigo, M. Birattari, T. Stützle.** *Ant Colony Optimization-- Artificial Ants as a Computational Intelligence Technique.* s.l. : IEEE Computational Intelligence Magazine, 2006.
- [43]. **Stützle, M. Dorigo & T.** *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Handbook of Metaheuristics.* 2002.
- [44]. **Socha, M. Dorigo & K.** *Approximation Algorithms and Metaheuristics - An Introduction to Ant Colony Optimization.* s.l. : CRC Press, 2007.
- [45]. *Dynamic routing in traffic networks and MANNETs using Ant Based Algorithms.* **B.Tatomir, J.L.Boehlé, L.J.M. Rothkrantz.** Delft : s.n., 2005. 7th International Conference on Artificial Evolution .
- [46]. **B.Tatomir, L.J.M. Rothkrantz.** *Advances in Natural Computation.* [book auth.] T.Bossomaier, J. Wiles H.A. Abbass. *Recent Advances in Artificial Life.* Sydney : World Scientific Publishing Company, 2005.
- [47]. **group, Green Light District Project.** Green Light District. *Intelligent Systems Group.* [Online] University of Utrecht, 04 15, 2001. [Cited: Januari 10, 2007.] <http://www.students.cs.uu.nl/swp/2001/isg/>.



## Data Figures of the Experiments

This appendix contains the results of the experiments as described in chapter seven. The figures in this appendix were gathered by executing simulations. Graphs presented here were created based on vehicular statistics collected during the experiments. Each experiment consists out of a table listing route statistics concerning the travel time of vehicles on the different routes present in the simulation environment. However since most experiments are executed on maps that contain a myriad of routes only the most interesting from the perspective of this thesis are listed. Graphs are included in each experiment to underline the findings of each of the experiments in a visual manner. Other data is included if such data appears to be relevant for explaining the significance of certain findings.

### I.1 Validation Experiments

This section contains the results of validation experiments conducted to validate proper functioning of the simulation environment. These experiments differ from the experiments in the following section since they do not contain any comparison between vehicular performances and only test proper functioning of the vehicular rule-set and routing algorithms.

#### I.1.1 Validation Experiment One

This section contains an overview of the data gathered from experiment one. The purpose of this experiment was to determine if vehicles using Dijkstra's algorithm did indeed follow the shortest paths in a city road infrastructure. Furthermore, this experiment also determined the influence that traffic lights have on travel times of individual vehicles. Table 24 presents the distances in meters between the different intersections in the road infrastructure. Table 25 contains the statistics gathered during the experiment for the different routes present in the city road infrastructure. Figure 57 depicts the organization of the routing table used to route vehicles incoming on intersection four. Figure 58 contains a graphical overview of the travel time development for routes originating in generator 'A'. Figure 59 contains the travel time development over time for the routes originating in generator 'B'. Figure 60 contains the travel time developments over time for routes originating in generator 'C'. Figure 61 contains the travel time developments over time for routes originating in generator 'D'.

**Table 24: Validation experiment one; intersection distances**

From	To	Distance (m)
A	1	160
B	3	120
C	5	260
D	8	200
1	2	240
1	4	420
2	3	340
2	4	200
3	6	180
4	5	160
4	6	180
6	8	180
5	8	180

Table 25: Dijkstra validation route statistics

Route	A → B	A → C	A → D	B → A	B → C	B → D
Vehicles	87	176	88	94	90	173
Roads	4	4	5	4	5	4
Minimum	44	54	60	46	48	36
Maximum	72	143	142	57	116	50
Average	47.81609	91.86932	99.67045	48.94681	80.6	38.7052
Std. Deviation	20.22163	3.986981	17.82206	2.147101	21.54942	2.447074
Median	47	90	100.5	48.5	82	38
Modus	46	80	101	48	49	38
Correlation	-0.20156	0.063571	0.101198	0.019232	-0.07662	-0.195
R <sup>2</sup>	0.040625	0.004041	0.010241	0.00037	0.005871	0.038024

Route	C → A	C → B	C → D	D → A	D → B	D → C
Vehicles	38	66	35	68	85	91
Roads	4	5	3	5	4	3
Minimum	62	56	33	68	36	34
Maximum	655	697	541	384	85	219
Average	384.7632	382.5	271.3429	211.8182	44.89412	88.62637
Std. Deviation	138.24	133.5034	136.919	62.29858	13.56556	50.93234
Median	406	391.5	244	194	39	64
Modus	454	622	244	194	37	45
Correlation	0.756574	0.659679	0.737757	0.787312	0.629631	0.761405
R <sup>2</sup>	0.572403	0.435176	0.544285	0.619861	0.396435	0.579737

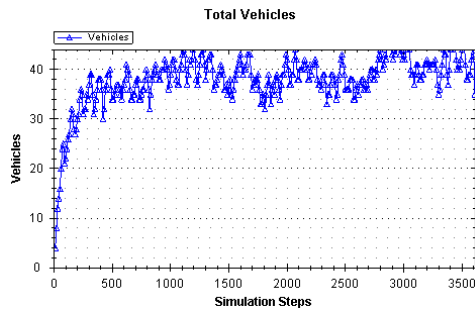


Figure 56: Validation experiment one, Average number of vehicles

	Destination	2	6	5	1
▶	0	Right	Ahead	Left	Left
	1	Right	Ahead	Left	Left
	2	Right	Right	Ahead	Left
	3	Left	Right	Right	Ahead
	4	None	None	None	None
	5	Ahead	Left	Right	Right
	6	Left	Left	Right	Ahead
	7	Ahead	Left	Right	Right
	8	Ahead	Left	Right	Right
	9	Left	Right	Right	Ahead
	10	Ahead	Left	Right	Right

Figure 57: Validation experiment one, intersection four routing table



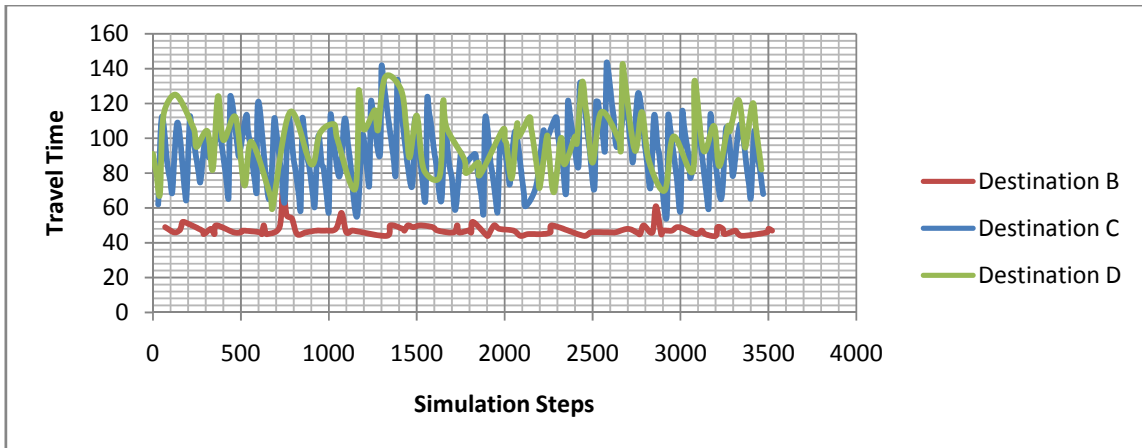


Figure 58: Validation experiment one, generator 'A' vehicle travel times

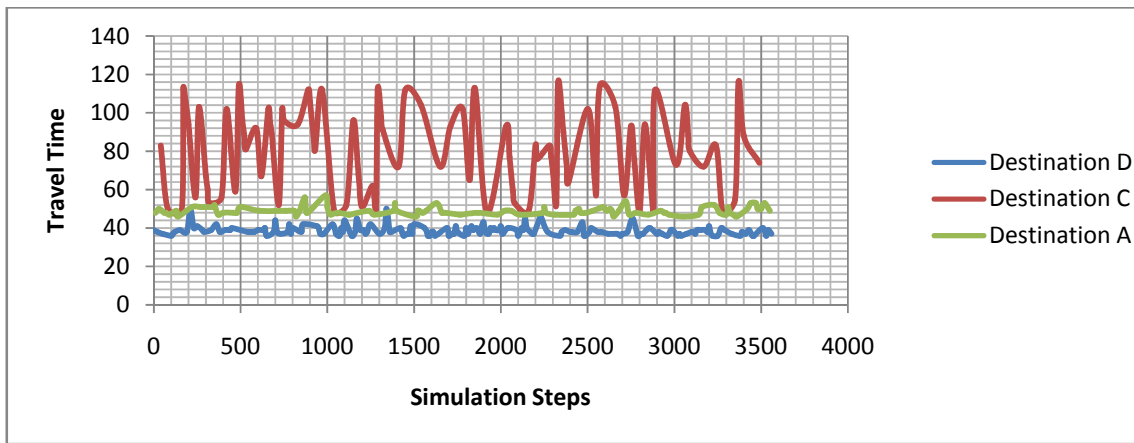


Figure 59: Validation experiment one, generator 'B' vehicle travel times

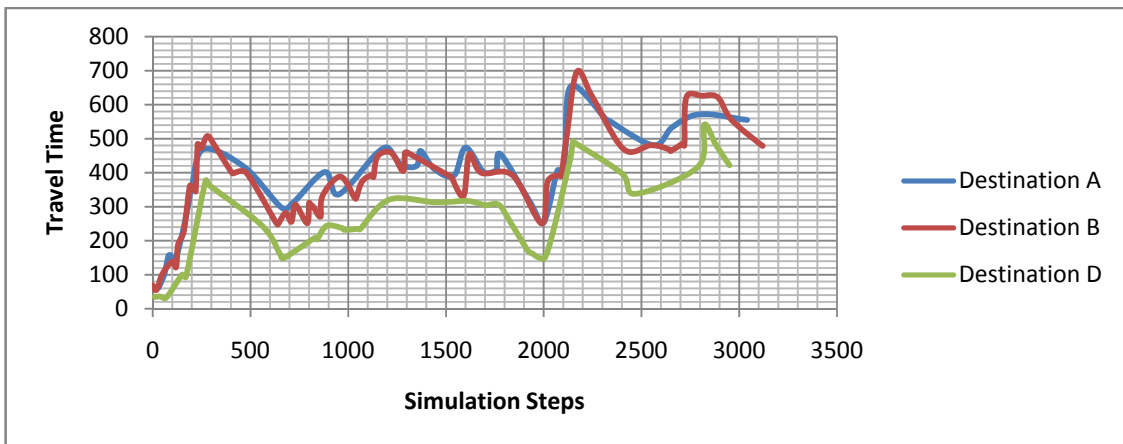


Figure 60: Validation experiment one, generator 'C' vehicle travel times

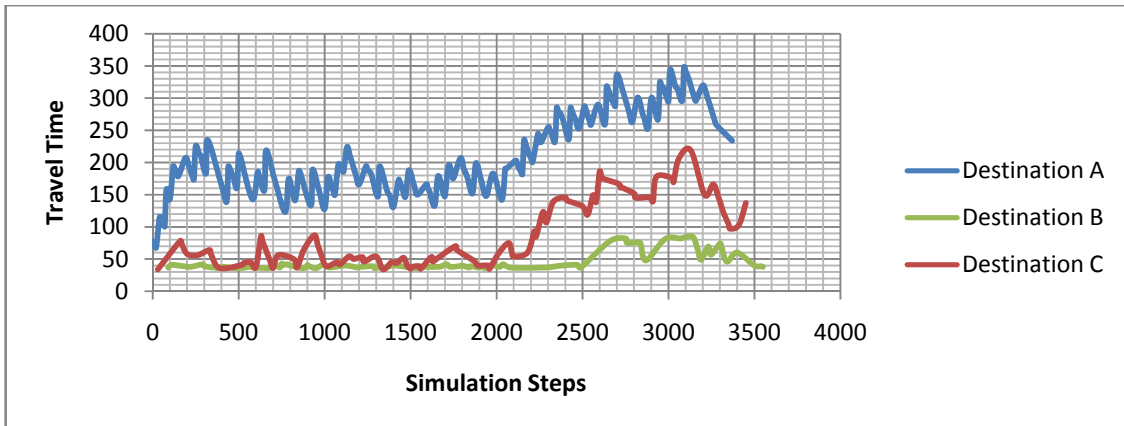


Figure 61: Validation experiment one, generator 'D' vehicle travel times

### I.1.2 Validation Experiment Two

This section contains an overview of the data gathered for validation experiment two. The purpose of this experiment was to determine the proper functioning of the Ant routing algorithm incorporated into the simulation environment. A detailed description of the purpose, environmental settings, expected results and conclusions based on the results in this section is found in chapter seven. Figure 62 shows the amount of vehicles present during each time step of the simulation. Figure 63 gives an overview of travel time developments for routes originating at generator 'A'. Figure 64 gives an overview of travel time developments for routes originating at generator 'B'. Figure 65 gives an overview of travel time developments for routes originating at generator 'C'. Table 26 gives an impression of the alterations made by the vehicles to the expected delay for the roads within the environment. Figure 66 depicts the colony routing table for colony 0 present in intersection 9 before and after the experiment.

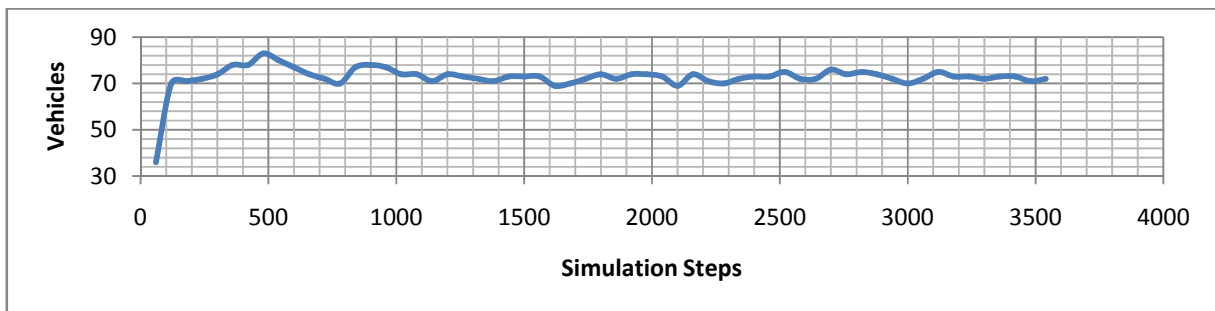


Figure 62: Vehicle count during simulation

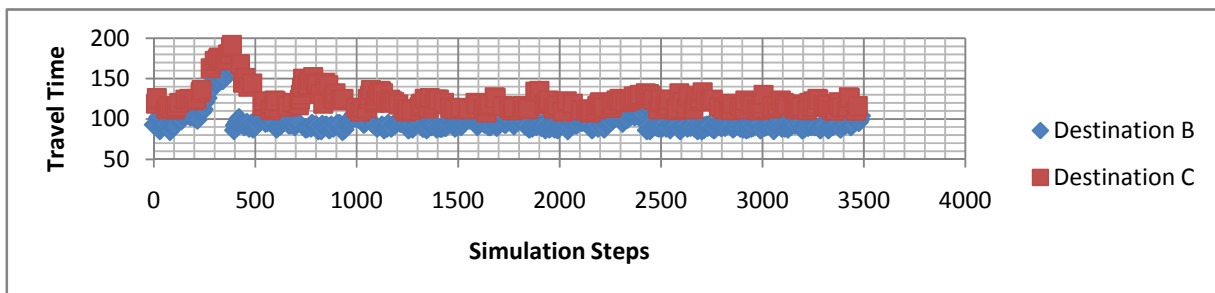


Figure 63: Vehicle travel times for vehicles originating at generator A

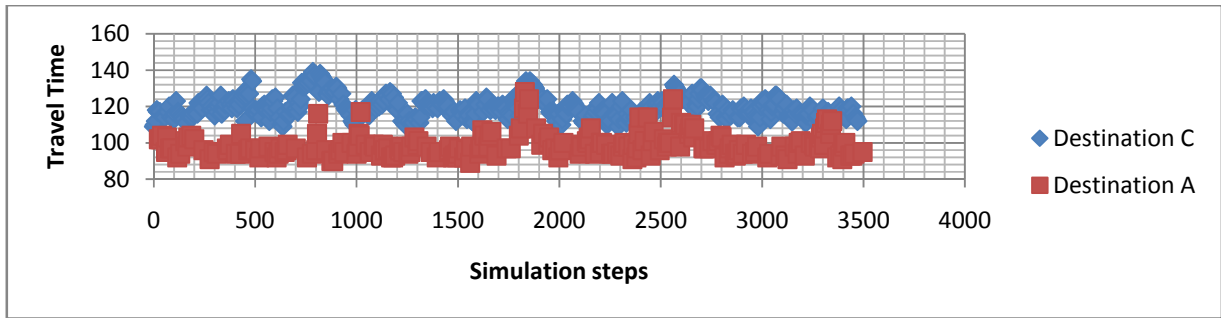


Figure 64: Vehicle travel times for vehicles originating at generator B

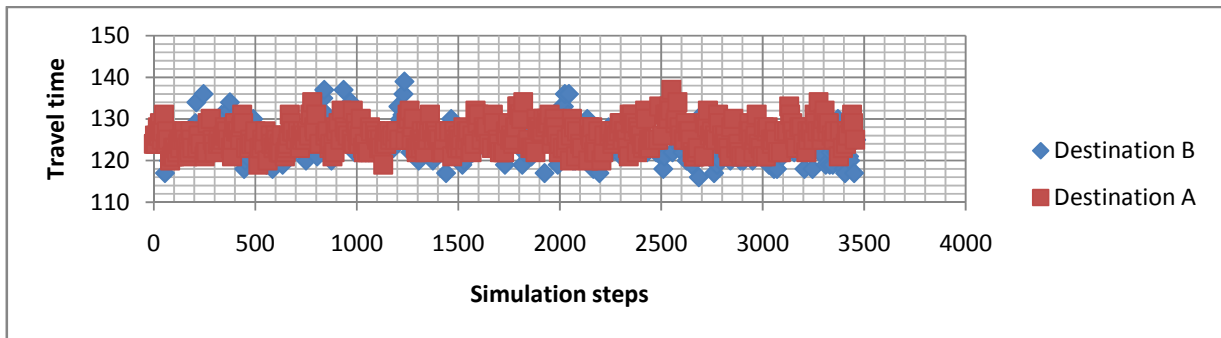


Figure 65: Vehicle travel time for vehicles originating at generator C

Link Id	Normal Delay	Alpha Id	Alpha delay	Beta Id	Beta delay
1	18	1	22,01715	2	20,24493
2	16	1	21,74638	3	17,4084
3	11	3	13,66319	4	14,14995
4	16	4	16,4875	5	16,28113
5	11	5	13,19284	1	14,01385
6	18	5	20,12132	6	19,92703
7	11	6	28,12972	2	13,69719
8	7	4	11,47681	7	14,33632
9	7	6	9,016836	8	10,01128
10	18	8	18	7	18
11	8	7	11,25519	9	12,23897
12	17	9	20,17621	10	18,43147
13	17	10	18,59408	11	19,17066
14	8	11	8,921059	8	11,33955
15	10	9	11,40694	12	12,10165
16	17	12	19,58207	13	20,16009
17	10	13	11,41325	10	12,69079
18	17	13	20,96302	14	18,17354
19	11	14	12,61914	11	13,63757
20	26	15	28,25594	8	29,42526
21	10	15	12,81219	16	10,74967
22	10	15	12,15398	17	22,83071
23	11	16	13,65167	18	12,25055
24	11	18	12,86397	20	12,89009
25	11	20	13,19159	19	13,58116
26	12	19	14,39913	17	13,76557
29	8	22	8	1	8,575409
30	9	13	8,637547	0	9
31	11	20	12,06013	21	11

Table 26: Road delays after simulation

To/Ma	11	7	Average Delay	To/Ma	11	7	Average Delay
0	0,9147367	0,4185548	49,475	0	0,9832957	0,443852	58,09203
13	0,9794931	0,2885513	41,14969	13	0,9993291	0,145648	49,09374
10	0,9633802	0,2976113	36,36525	10	0,9974453	0,1176601	41,25434
14	0,9915704	0,2922415	23,27408	14	0,9987869	0,1336388	22,78421
12	0,360392	0,9971201	40,30976	12	0,1588709	0,9994474	45,06721
11	0,985779	0,2259968	8,047918	11	0,9992939	0,07596072	8,949649
9	0,3408498	0,9818307	30,32691	9	0,08037943	0,9999751	30,38504
7	0,3337969	0,9851153	17,99999	7	0,09824353	0,9972208	18,00043

Figure 66: Intersection 9 pre- and post simulation probabilities for colony 0

## I.2 Experiment Two: a Rural Town

This section contain relevant data graphs gather during the execution of the three experiments conducted using the town environment. The graphs and comments in this section serve to assist and strengthen the conclusions drawn in chapter seven.

Table 27: Statistics for the rural town experiments

	Experiment One	Experiment Two	Experiment Three
<b>Samples</b>	344	285	307
<b>Roads traveled</b>	6,03	5,58567	5,65798
<b>Minimum</b>	32	33	33
<b>Maximum</b>	323	177	128
<b>Average</b>	82,06395	78,57009	78,73941
<b>Std. Deviation</b>	32,4846	22,17929	21,45334
<b>Median</b>	76	76	78
<b>Modus</b>	75	75	73
<b>Correlation</b>	0,046569	0,036651	0,099972
<b>R<sup>2</sup></b>	0,002169	0,001343	0,009994
<b>Right turns</b>	1,453	0,816199	0,7684
<b>Left Turns</b>	1,235	0,785047	0,768729
<b>Avg. Parking</b>	5076	4765	5202

Table 27 shows the averages gathered from the three experiments. Experiment two and three that used the 'ant parking vehicle' show that in the town environment the usage of 'ant vehicles' to provide extra dynamic routing information to the ant-based algorithm does not lead to reduced travel times. However, it is clear that when using 'ant vehicles' the maximum travel time is reduced significantly leading to a more stable performance of the ant-based algorithm. Experiment one that used 'Dijkstra parking vehicles' suffers clearly from the fact that vehicles are forced to search for a parking place. The maximum travel time compared to experiment two and three is almost doubled.

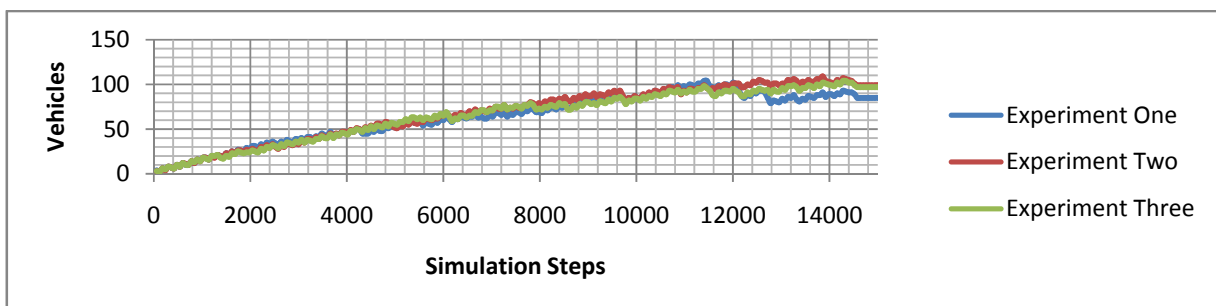


Figure 67: Vehicle count of experiments in the town environment

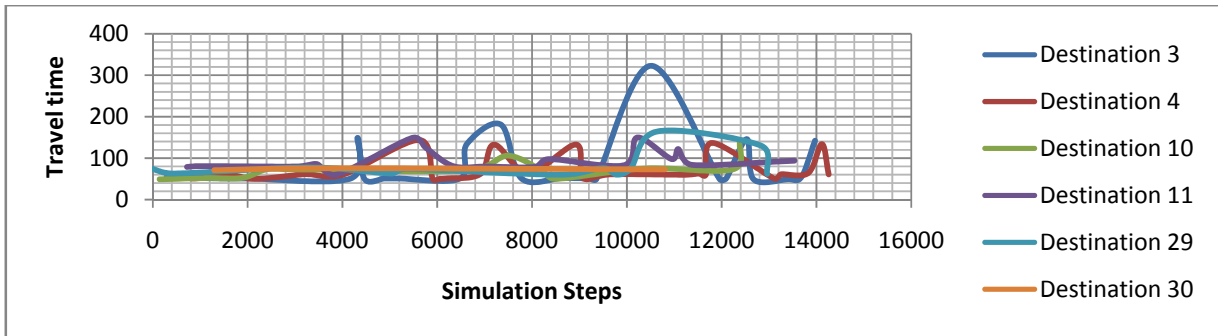


Figure 68: Experiment one generator to parking place travel time

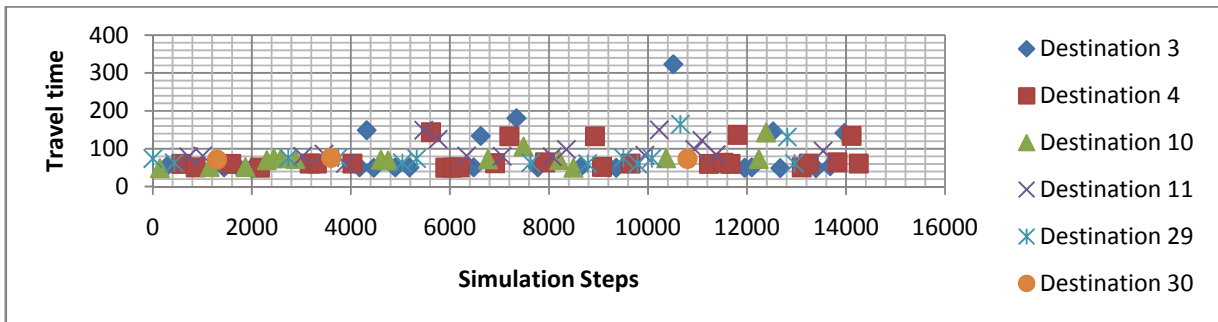


Figure 69: Experiment one generator to parking place travel time (scatter graph)

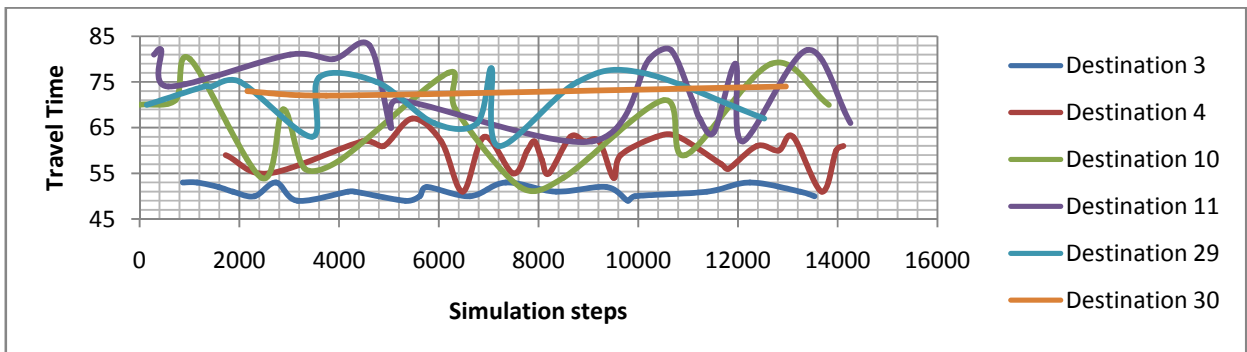


Figure 70: Experiment two generator to parking place travel time

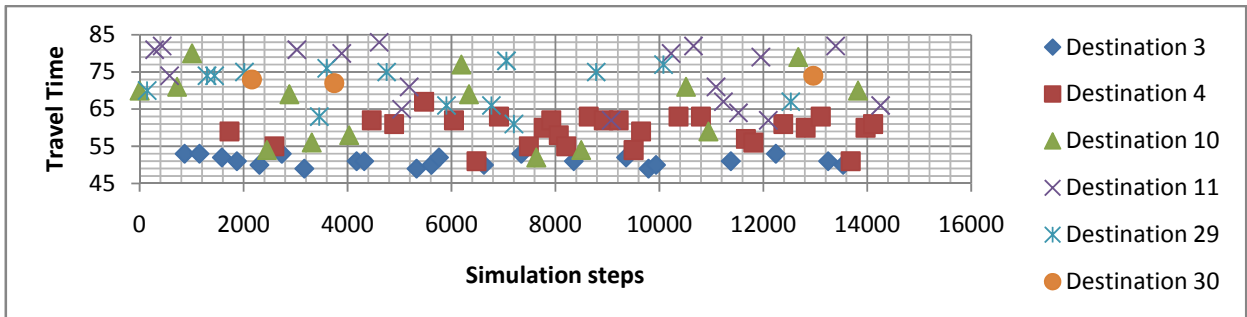


Figure 71: Experiment two generator to parking place travel time (scatter graph)

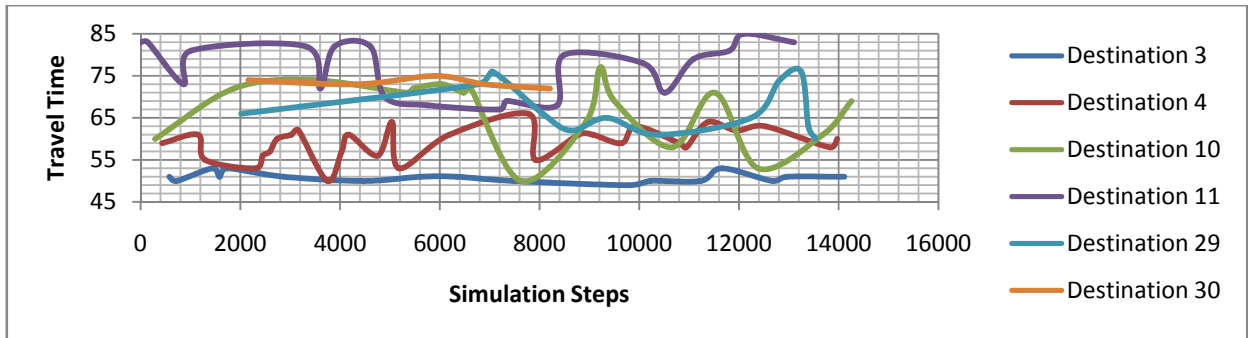


Figure 72: Experiment three generator to parking place travel time

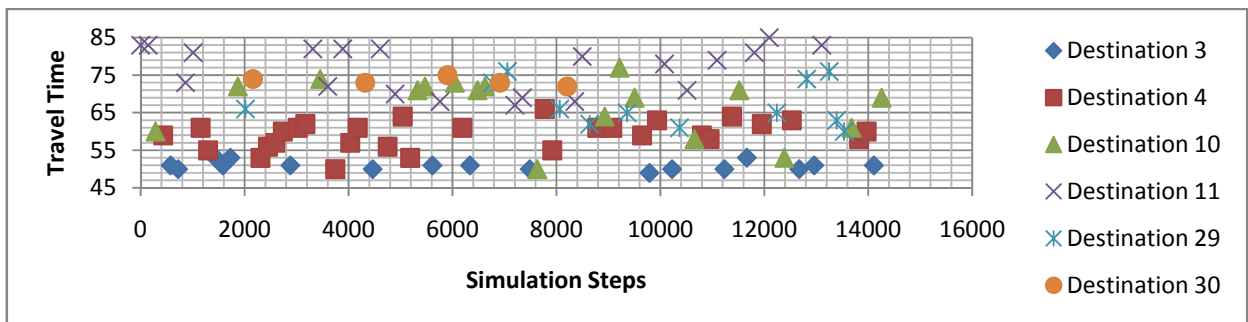


Figure 73: Experiment three generator to parking place travel time (scatter graph)

Figure 68 through Figure 73 show the travel times for vehicles from the parking vehicle generator positioned at the left side of the town environment on the main road. From these graphs it is clearly visible that the ‘Dijkstra parking vehicles’ in the early phase of the experiment perform equal to the ‘ant parking vehicles’ of experiment two and three. While travel times in experiment two and three show little variation the variation in the first experiment grows steadily larger. This increase in variation for experiment can be completely contributed to the parking place searching behavior of the ‘Dijkstra parking vehicles’. The smaller and constant variation found in all three experiments for travel times to the same destination is caused by the position of the parking place on the road or the delay suffered from waiting on another vehicle to finish its parking procedure.

### I.3 Experiment Three: a Village

This section contains graphs and overviews gathered during the experiments conducted in the village environment. The information presented in this section serves to further support the conclusions presented in chapter seven. Figure 74 shows the development of vehicular pressure on the village environment.

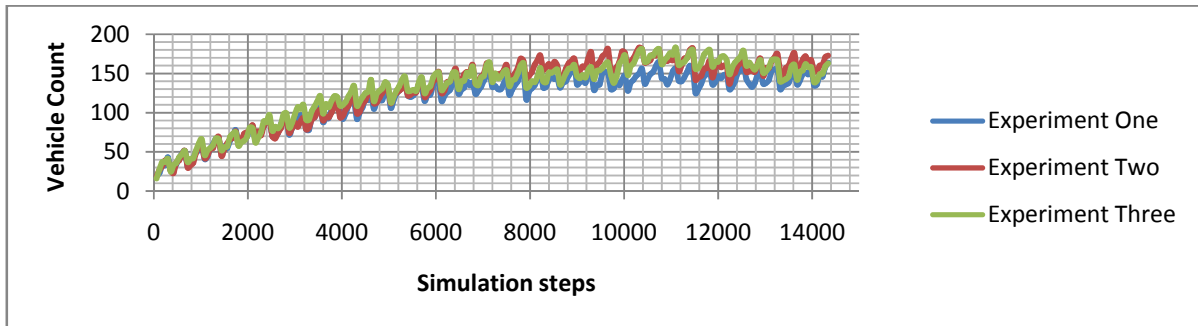


Figure 74: Vehicles present in simulation environment

Table 28 contains an overview of the findings of the three experiments. Where the results gathered during the simulations using the town environment showed that the results of the three experiments were nearly similar differences between vehicles using Dijkstra's algorithm and those using the CBPRS are more visible during these experiments. The vehicles of experiment one required more time to find a parking place and had to travel more roads on average before a parking place.

Table 28: Experimental data overview for experiment three

	Experiment One	Experiment Two	Experiment Three
<b>Samples</b>	666	621	619
<b>Roads traveled</b>	8,394619	6,628617	6,664516
<b>Minimum</b>	8	9	8
<b>Maximum</b>	515	181	200
<b>Average</b>	93,95067	75,18489	75,15968
<b>Std. Deviation</b>	84,04065	35,73912	34,9928
<b>Median</b>	77	70	71
<b>Modus</b>	35	34	34
<b>Correlation</b>	0,134808	-0,03058	0,056712
<b>R<sup>2</sup></b>	0,018173	0,000935	0,003216
<b>Right turns</b>	2,588939	1,09164	1,13871
<b>Left Turns</b>	2,880419	1,152733	1,096774
<b>Avg. Parking</b>	5247	4778	5217

The difference between experiments two and three is minimal. The average travel time during the simulations is nearly identical as is the median and modus. The standard deviation of experiment three is however smaller than that of experiment two. The significant increase of ant-based vehicles in experiment three over experiment two and the thereof resulting increasing in dynamic routing information provided to the CBPRS lead to a decrease in the variation of travel times. However considering the size of the increase in ant-based vehicles the decrease in variation is insignificant.

Table 29: Traffic jam overview per experiment

Experiment Two			Experiment Three		
Road	Between	Duration	Road	Between	Duration
4	0 → 5	14	4	0 → 5	26
4	0 → 5	6	0	0 → 1	16
			0	0 → 1	22
			6	0 → 7	32

Table 29 contains an overview of traffic jams and their durations during the experiments two and three, experiment one did not produce any traffic jams. The traffic jams were found to occur only at the traffic light intersection of the two main roads running through the city. The reason these traffic jams occur stems from the fact that the ant-based algorithm incorporated in the CBPRS guides ant-based vehicles along the main road when travelling between colonies. The traffic light intersection placed at this crossing of main roads entices the creation of traffic jams. In experiment two the number of ant-based vehicles was limited to the parking vehicles and the number of traffic jams was low and relatively short of duration. In experiment three where all vehicles were ant-based the number of traffic jams increases to almost all incoming roads on the intersection of the two main roads and the duration of traffic jams increases. These results while illustrating a main feature of the ant-based algorithm implemented in the CBPRS also illustrate a potential weakness when the number of main roads within a city is low.

## I.4 Experiment Four: a City

This section contains supplementary data gathered from the four experiments conducted using the city environment. The data presented in this section serves to strengthen the conclusions drawn in chapter seven and provides additional detail. Table 30 contains an overview of the averages gathered from all the vehicles present in the simulation environment during each experiment. The data presented in the following graphs was gathered via sampling using 60-second intervals.

**Table 30: Statistics for the city experiments**

	Experiment One		Experiment Two		Experiment Three	
	<i>Parking</i>	<i>Normal</i>	<i>Parking</i>	<i>Normal</i>	<i>Parking</i>	<i>Normal</i>
<b>Samples</b>	666	14162	692	14353	630	14261
<b>Roads traveled</b>	18,72022	10,40203	10,09971	10,38282	9,642743	11,49566
<b>Minimum</b>	52	10	32	11	11	13
<b>Maximum</b>	8511	5269	3069	3263	629	633
<b>Average</b>	544,4459	295,6903	278,698	281,0082	198,6427	227,8707476
<b>Std. Deviation</b>	680,2905	349,155	288,3248	319,4662	103,021	114,4710029
<b>Median</b>	330	228	230	221	196	222
<b>Modus</b>	262	229	61	201	53	221
<b>Correlation</b>	-0,07067	-0,04323	-0,04576	-0,05671	-0,08381	-0,00975907
<b>R2</b>	0,004995	0,001868	0,002094	0,003216	0,007024	9,52395E-05
<b>Right turns</b>	7,907147	2,282991	1,647399	2,265501	1,62201	2,726679014
<b>Left Turns</b>	7,400122	2,389819	1,904624	2,379824	1,744817	2,705039478
<b>Avg. Parking</b>	5126	-	4851	-	4973	-

	Experiment Four		Experiment Five	
	<i>Parking</i>	<i>Normal</i>	<i>Parking</i>	<i>Normal</i>
<b>Samples</b>	666	13785	585	14379
<b>Roads traveled</b>	10,24018	11,54607	16,36068	10,37339
<b>Minimum</b>	16	12	40	10
<b>Maximum</b>	637	2120	6790	4145
<b>Average</b>	234,7266	241,0761	384,1521	277,4136
<b>Std. Deviation</b>	105,1684	135,2602	490,7459	320,7281
<b>Median</b>	227	233	251	217
<b>Modus</b>	199	289	201	169
<b>Correlation</b>	-0,13729	0,014331	0,068376	-0,02652
<b>R2</b>	0,018849	0,000205	0,004675	0,000703
<b>Right turns</b>	1,647147	2,740576	6,136752	2,276375
<b>Left Turns</b>	1,926426	2,734944	5,928205	2,373948
<b>Avg. Parking</b>	5264	-	4902	-



Table 30 shows the effect of different vehicle type combinations during the experiments. During experiment one, that relied on routing using Dijkstra’s algorithm, travel times for vehicles are large with a considerable standard deviation for parking and non-parking (normal) vehicles. While traffic jams certainly influence travel times the parking vehicles of experiment one show also a greater number of lanes and turns taken in comparison to parking vehicles of other experiments.

This indicates that certain parking vehicles were forced to conduct a search for a parking place near their destination causing considerable delay. The median and modus show that most vehicles were able to locate a parking place at their point of destination.

Comparing experiment one and two, we see that it is clear that participating in the CBPRS has benefits even if only 4.6 percent of all other vehicles participate in the system and provide dynamic routing information. The differences in travel times amount to almost five minutes when looking at the averages for all vehicles. When comparing the medians or modus for both experiments the difference decreases but is still nearly one and half minute. The parking vehicles of experiment two decrease the travel time for non-parking Dijkstra orientated vehicles slightly, the difference of ten seconds is however not significant. The results of experiment two allow the conclusion to be drawn that using the city environment the CBPRS vehicles perform identical non-parking Dijkstra vehicles.

Experiment three was conducted to show the optimal achievable performance of the CBPRS when all vehicles participate in the CBPRS. The average travel times of both types of vehicles used decrease dramatically when compared to experiment one and two. The decrease in standard deviation and maximum travel time shows that the CBPRS and the ant-based algorithm incorporated into it are able to perform in a constant optimal manner during the entire experiment. The lack of traffic jams because of improved vehicular distributions, as discussed in chapter seven, shows that optimal usage of the available road infrastructure can lead to significant improvements in travel time over a prolonged period.

Experiment four is the culmination of a series of sub-experiments conducted to find a minimal required amount of CBPRS participants before the full benefits of the CBPRS can be observed on a constant basis. As was described in chapter seven a distribution of 50 percent ant-based vehicles and 40 percent Dijkstra-oriented vehicles gave the most optimal results in terms of travel times for all vehicles in the environment. The vehicular distribution that arises from this configurations decreases the average travel time for both types of vehicles with forty seconds while keeping traffic jams at a minimum. The optimal number of 50 percent is higher than expected although the benefits in terms of travel time and traffic jam decrease as opposed to the situation in experiment one would almost certainly entice individuals to purchase the required CBPRS subscription.

Experiment five allowed Dijkstra oriented parking vehicles to uses the CBPRS parking service. The setting removed the need for extensive parking place searches during the latter periods of the simulation experiment. The results from this experiment clear show a decrease in overall travel time for normal and parking vehicles compared to experiment one. However, when compared to experiment two, which used the same vehicular distribution the performance of the parking vehicles, the travel time improvement still does not achieve the reduction of the ant-based algorithm.

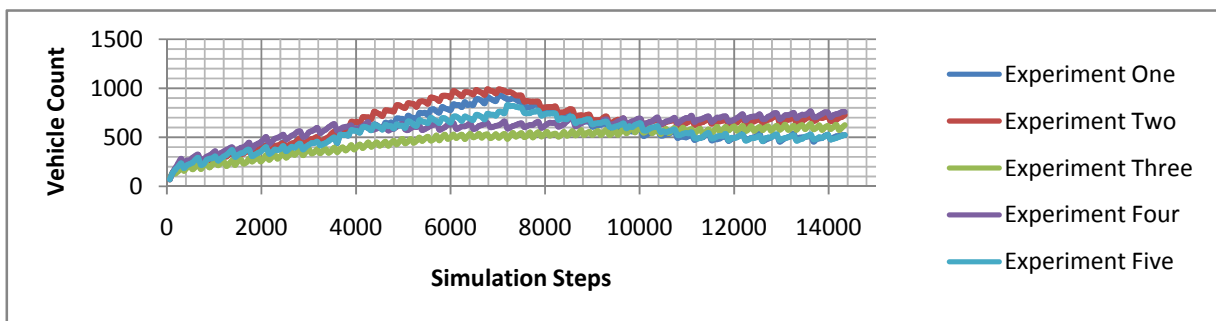


Figure 75: City experiments vehicle Counts

Figure 75 shows the development of the vehicle count during each of the four experiments. In line with averages for the vehicles described in the previous paragraph experiment one and two and experiment three and four show similar developments. While traffic pressures remain constant during all four experiments one and two show a curve line. The non-parking Dijkstra vehicles used during experiment two created a similar traffic pattern within the environment. Pressures on the environment increase until simulation step 7.200. During this period, the vehicles are slowly spreading themselves through the city and create traffic jams at traffic light intersections. After this period, the vehicles have spread sufficiently over the city to allow vehicles to move with more freedom.

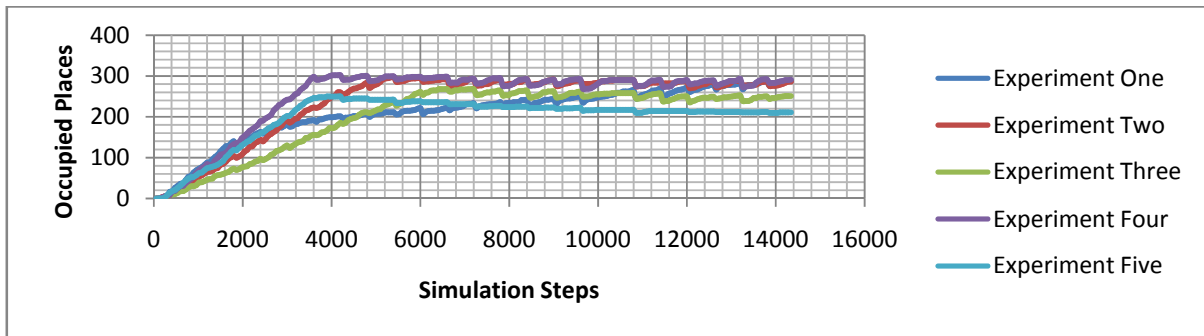


Figure 76: Overview of occupied places

Figure 76 contains the development of occupied places during each of the four experiments. The number of parking places available during each experiment amounted to 350. This figure consisted out of two parking garages of 100 parking places each in the center of the city with 150 individual parking places distributed over the surrounding colonies. The line drawn for experiment one shows that up until simulation step 11.000 the Dijkstra parking vehicles used in experiment one have more difficulty in reaching their parking place due to traffic jams (see chapter 7). Experiments two, three, four and five show similar occupancy lines although offset. These discrepancies were caused by initially randomly chosen periods for parking and normalize over time.

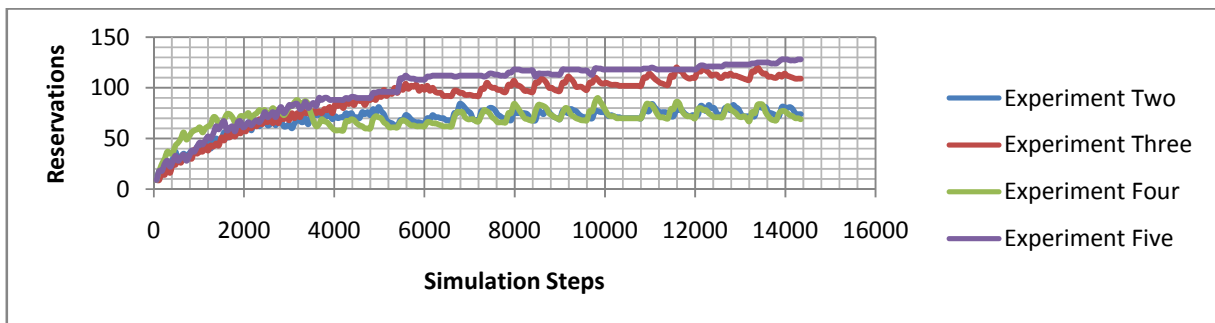


Figure 77: Overview of parking place reservations

Figure 77 depicts the development of parking place reservations with CBPRS by ant-based parking vehicles. Reservations for experiments two and three develop via similar pattern with on average 60 reservations during the entire simulation. Experiment three averages just above one hundred reservations. This higher number of reservations is caused by the occupancy development in the previous graph, which is lower than for experiments two and four.

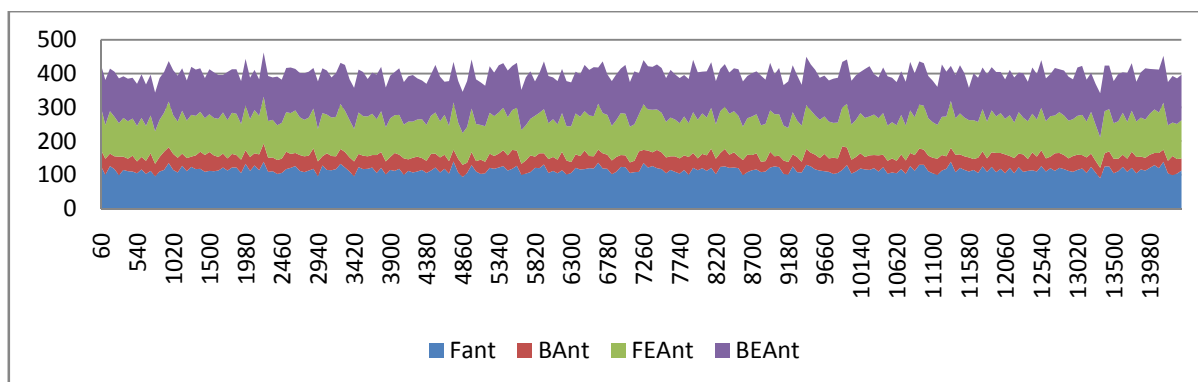


Figure 78: Development of ant traffic during simulations

Figure 78 shows the number of ants generated per simulation step. The number of forward-ants (FANT) generated per experiment is higher than the number of backward-ants (BANT) generated. The main cause behind this difference is the introduction of the traffic lane direction rules into the algorithm as described in chapter three. This causes that a certain amount of forward-ants is unable to reach its destination via the internal paths of colony. On average slightly more than 50 forward-ants and backward-ants reach their destination each simulation step. The forward exploring-ant (FEAnt) and backward exploring-ant (BEAnt) that maintain routes between colonies show an evenly offset pattern with for both on average 150 ants per simulation step.

## I.5 Experiment Five: a Metropolis

This section contains data gathered from the experiments conducted on the metropolis environment. The data presented in this section is used to support and further strengthen the conclusions drawn in chapter seven and supplements the graphical information presented there. Table 31 and Table 32 contain the averages compiled based on the route of all vehicles that traveled through the environment during each experiment. The data presented in the graphs in this section was based on statistics gathered by sampling with a 60-second interval.

Table 31: Statistics gathered during experiments one through three

	Experiment One		Experiment Two		Experiment Three	
	Parking	Normal	Parking	Normal	Parking	Normal
<b>Samples</b>	3565	16452	3264	17671	2949	16954
<b>Roads traveled</b>	21,49453	14,21402	14,55423	15,85909	13,89963	15,60431
<b>Minimum</b>	72	15	65	13	55	15
<b>Maximum</b>	7099	11005	1685	2365	2194	1573
<b>Average</b>	573,1335	647,6954	235,674	261,3705	215,0478	219,0969
<b>Std. Deviation</b>	884,5087	1088,227	120,9569	201,6569	92,41508	111,6737
<b>Median</b>	266	222	211	220	203	208
<b>Modus</b>	197	138	199	208	175	197
<b>Correlation</b>	0,117744	0,15391	0,020161	0,05349	-0,06784	-0,09679
<b>R2</b>	0,013864	0,023688	0,000406	0,002861	0,004602	0,009367
<b>Right turns</b>	8,486396	2,628799	2,153186	3,255843	2,097999	3,269859
<b>Left Turns</b>	9,486676	2,628799	2,348652	3,255843	2,264836	3,321112
<b>Avg. Parking</b>	5096	-	5097	-	5036	-

Table 31 contains the information gathered from the three default experiments conducted in every simulation environment. The vehicles used in experiment one did not use the CBPRS for routing and parking. The average travel times for both parking and non-parking (normal) vehicles in this experiment are significantly higher than during the other two experiments. The reason for this as discussed extensively in chapter seven are traffic jams caused when all vehicles opt for the same selection of main roads when driving towards their destination.

Comparing the average travel times of experiment one with experiment two we find that the parking vehicles generated per hour cause great disturbances in the flow of traffic during experiment one. In experiment two, the parking vehicles use the CBPRS and ant-based routing which results in vehicles being guided along other main roads towards their destination. This alleviates the pressures on the roads taken by the normal (Dijkstra) vehicles causing a reduction of traffic jams and a sharp decrease in average travel times and standard deviation.

Experiment three depicts the utopian situation wherein all vehicles participate in the CBPRS. While in comparison to experiment two the average travel time and standard deviation have decreased, the gain in reduced travel time is not as great as expected. For parking vehicles the difference in travel times is 30 seconds while for normal vehicles the reduction is almost one and half minute. The gain in reduced travel times between experiment one and three amounts to five minutes for parking vehicles and more than seven minutes for normal vehicles.

During experiment two 15,6 percent of vehicles used the CBPRS and followed routing directions provided by the ant-based algorithm. The figure of 15 percent was found to be the lowest limit acceptable. Ant-based vehicle counts below this start to show deteriorated routes over time since the area covered by these vehicles is too small for the algorithm to maintain the necessary overview.

The reduction in travel times for normal vehicles in experiment two and three where found to be significant enough to allow for further experimentation. A series of experiments was conducted that, while keeping the total number of vehicles in the environment constant, used different mixtures of 'normal' Dijkstra orientated vehicles and ant-based vehicles to determine the effect this might have on travel times.

**Table 32: Statistics gathered during experiment four and five**

	Experiment Four		Experiment Five		Experiment Six	
	Parking	Normal	Parking	Normal	Parking	Normal
<b>Samples</b>	3319	17758	3276	17443	3030	17098
<b>Roads traveled</b>	14,14462	14,47128	14,18315	15,94458	17,54877	14,50744
<b>Minimum</b>	65	15	64	14	72	15
<b>Maximum</b>	1889	3007	530	678	7335	8228
<b>Average</b>	223,8283	251,0298	222,9814	229,8146	371,936	469,052
<b>Std. Deviation</b>	107,3369	268,7105	87,00529	115,3335	665,6382	745,5331
<b>Median</b>	208	189	205	217	226	211
<b>Modus</b>	196	148	200	210	184	132
<b>Correlation</b>	-0,04778	0,04981	-0,01056	-0,03122	0,155917	0,121748
<b>R2</b>	0,002283	0,002481	0,000112	0,000974	0,02431	0,014823
<b>Right turns</b>	2,252486	2,624226	2,155372	3,25835	4,9133	2,630865
<b>Left Turns</b>	2,441097	2,649454	2,306471	3,385182	5,003448	2,655749
<b>Avg. Parking</b>	5132	-	5082	-	5048	-

Table 32 contains two experiments selected from the subset of experiments conducted that provide an insight into the changes of travel times when using different combinations of normal Dijkstra oriented and ant-based vehicles. The most interesting conclusions that can be drawn from the results of experiment four and five is that an increased number of normal ant vehicles that experiment five has over experiment four does not affect the average travel time of the parking vehicles. The standard deviation however decreases by twenty seconds. The travel time for the combined set of normal vehicles decreases by thirty seconds while standard deviation even decreases by almost two and half minutes. The conclusion can therefore be drawn that an increase in the number of ant-based vehicles results in lower travel time variance for all vehicles and decreased average travel times for normal vehicles. Experiment six that enabled Dijkstra orientated parking vehicles to use the CBPRS parking service showed a reduction in travel times compared to experiment one but could not match the improvements found during experiment two.

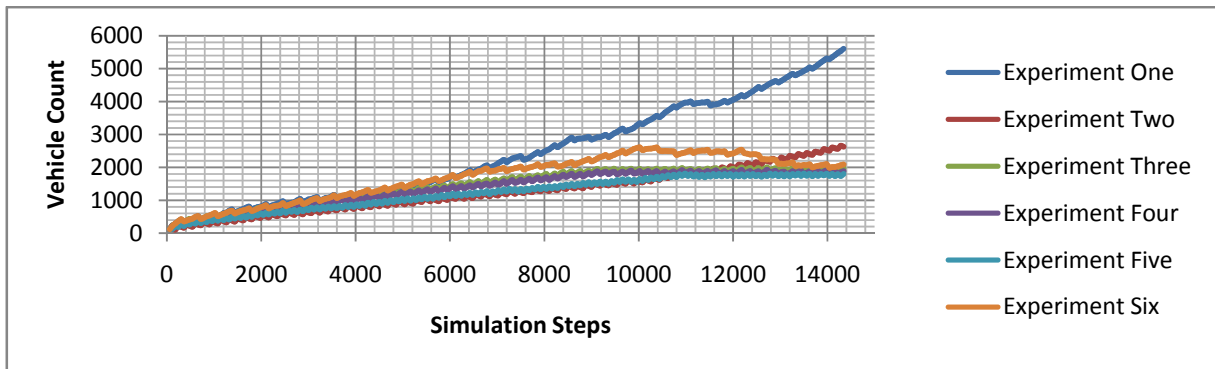


Figure 79: Vehicle count during experiments

Figure 79 shows the development of the number of vehicles present in the simulation environment per time step for each of the five experiments. The initial development for all five experiments follows the same trend. Around simulation step 5.000, the number of Dijkstra oriented vehicles in experiment one begins to increase slowly as the number of traffic jams starts to rise. Experiment two also suffers from traffic jams although here the symptoms start to exhibit themselves relatively late during the experiment. Experiments three, four and five show similar vehicle counts as was to be expected considering the results discussed in the previous paragraph.

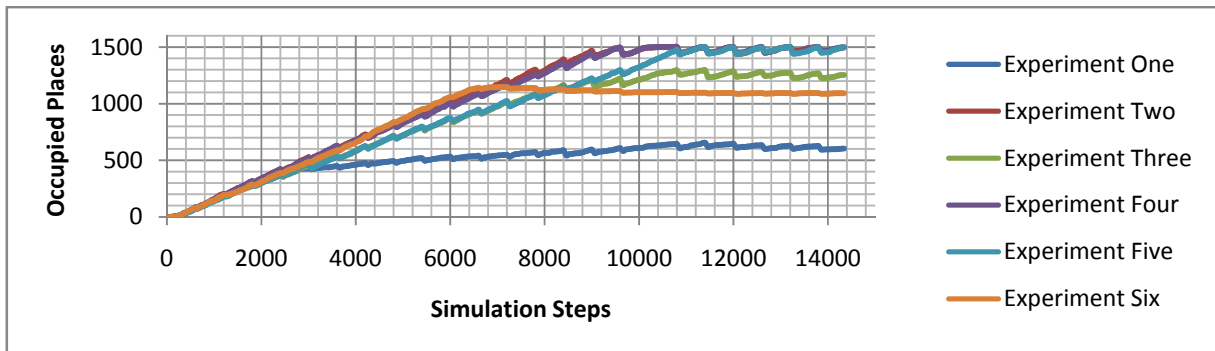


Figure 80: Parking place occupancy developments

Figure 80 shows the development of occupied places for the five experiments discussed in this section. Experiment one conducted with Dijkstra parking vehicles shows a low rate of occupancy in comparison to the other experiments. The reason for this is that most Dijkstra parking vehicles became stuck in traffic jams for prolonged periods of time thereby decreasing the influx of vehicles onto parking places. The ant-based parking vehicle experiments two, four and five suffer less from traffic jams and are able to reach their parking places in a more timely manner. This in the end results in a situation where all parking places are taken. Experiment three conducted solely with ant-based vehicles does not occupy all palaces as with the previous experiments. The reason behind this was found to be the random number generator used that produced parking times that caused a large number of vehicles to depart during simulation step 9000.

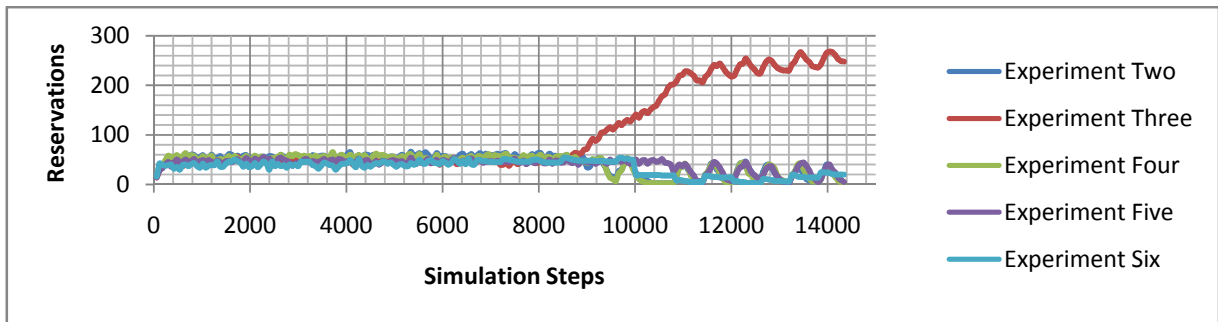


Figure 81: Parking place reservation developments

Figure 81 shows the development of parking place reservations registered with the CBPRS during the simulation. All four experiments have similar patterns for reservations with on average 65 reservations. After simulation step 9000 however the number of occupied places increases and the reservations naturally decrease. Experiment three shows a different trend after simulation step 9.000 for which the reasons where described in the previous paragraph.

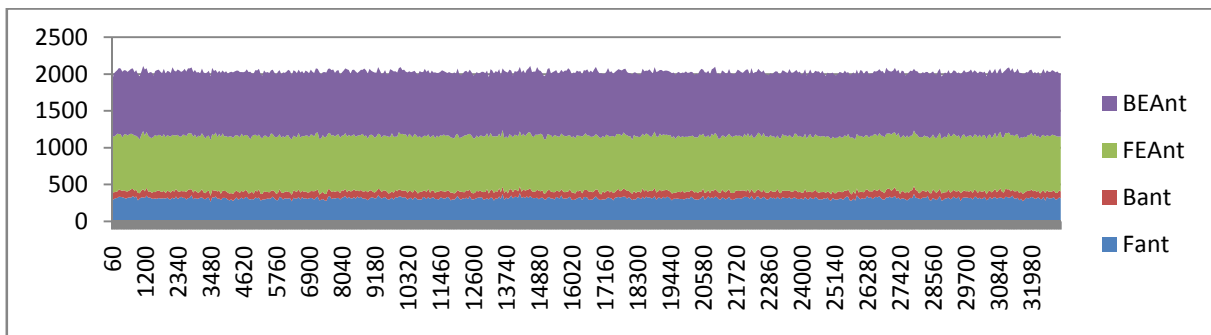


Figure 82: Development of ant traffic during metropolis experiments

Figure 82 shows the development of the number of ant's active during the simulation. The backward exploring-ant (BEAnt) and forward exploring-ant (FEAnt) are responsible for launching the greatest number of ants. This is caused by the large size of the main road network in the metropolis environment. Routes along which backward exploring-ants and forward exploring-ants are required to travel between each of the 50 colonies require a significant amount of communication to keep up-to-date. The Forward-ants (FAnt) and Backward-ants (BAnt) show normal communication patterns similar to those of the city environment.

## Technological Requirements

The proposed method of operation concerning the CBPRS was discussed in chapter one. Based on the description of the CBPRS a number of questions can arise concerning the applicability of the CBPRS in a real city environment. Therefore, this appendix gives a brief overview of the technological requirements and the state of current day technology that needs to facilitate these requirements. The following sections of this chapter will discuss the city infrastructure, intelligent lampposts, Power-line communication and parking place sensors. The solutions proposed in this chapter are possibilities, the actual applicability of these proposed solutions and ideas should be subject to further research.

### II.1 City Infrastructure

The dependence of individuals on wireless and wired technology is ever increasing since the introduction of the internet, mobile telephones and the recent rise in popularity of digital television on the Dutch market [31]. This increase in dependence and the ever higher requirements for bandwidth related to these services requires continuous innovations and extensions of the current infrastructure. The expected increase in High Definition Television (HDTV) programs and broadcasts over the internet and wireless mediums will boost the need for higher bandwidth even further [32]. In the rapport 'Contra-Expertise Slagkracht door Glas' professor dr. ir. N.H.G. Baken [33] discusses the expected rise in connectivity and the needs and uses for an increase in bandwidth in relation to the proposal made by the city council of Amsterdam to deploy a citywide fiber network.

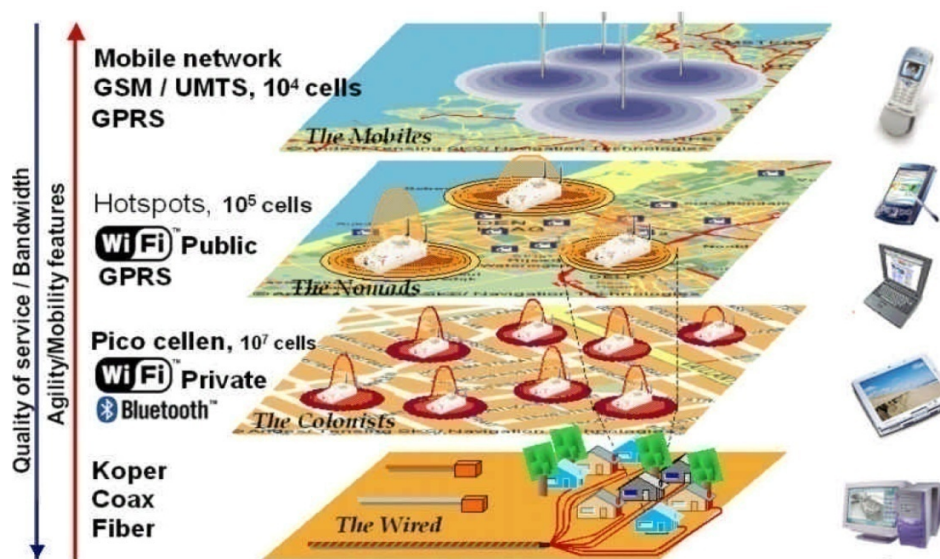


Figure 83: Communication layers within a city environment

Figure 83 gives an overview of the different communication methods available in city environments and the layers on which they operate. The CBPRS as proposed in chapter one relies heavily on wireless communication in order to gather and share information between participants and the CBPRS. This means that the CBPRS is depended on the wired layer in order to facilitate communication between individual lampposts and parking system control systems. For communication between individual participants and intelligent lampposts, the CBPRS is depended on the services offered by Wi-Fi hotspots.

The wired layer consists out of the set of communication transportation mediums that are imbedded within the soil. These mediums generally are able to provide high bandwidth and quality of service (QOS) to it users. In the current Dutch city environments copper connections run between each house and a central neighborhood connection point, from there on the network consists out of glass fiber connections. While it is only a question of time before glass fiber connections are extended toward homes, the time period required to complete these connections is uncertain. Therefore, the proposed CBPRS proposes to use the power-line infrastructure in order to facilitate data exchange between individual intelligent lampposts. The benefit of exploiting the power-line infrastructure is the fact that it is already in place and covers entire cities. This leads to lower deployment costs in comparison to extending glass fiber cables. The downside of using the power-line infrastructure is a lower bandwidth than is achievable using glass fiber. This method for transporting large amounts of data is preferred over any wireless alternatives due to fact of high signal interference within cities, cost of placing powerful wireless beacons on buildings throughout the city and public concerns over health issues relating cancer and other illnesses to wireless signals. Once the penetration of glass fiber connections reaches individual homes, the replacement of power-line communication with glass fiber communication is certainly preferable. However, since this is not the case power-line communication is the most preferable and therefore described in section four.

The wireless layer is the most important layer of the CBPRS and is depicted in Figure 83 as the hotspot layer. Wireless fidelity (Wi-Fi) is a brand name used to describe devices that operate based on the IEEE 802.11 standard [34]. The IEEE 802.11 standard contains a number of postfixes (a through y) that all detail certain standard methods of operation for transmitting data via the ether at different data transmission speeds. The current IEEE 802.11g standard allows wireless transmissions at speeds of 54 Mbit/s. According to [33] the number of public hotspots will exceed  $10^5$  in the period 2010 through 2015 in comparison to the 1542 that were present at end of 2004 in the Netherlands. The growth potential attributed to these hotspots is huge, not only due to increased internet demands of people on the move but also by the possibility of providing on-demand services such as route finding, table reservation, points-of-interest, etc. Thus, while the coverage of these wireless devices encapsulated in intelligent lampposts is not present at current the growth potential and possibilities certainly allow for realistic and sustainable deployment of those devices in the near future.

## II.2 Intelligent Lampposts

Intelligent lampposts [35] are a new phenomenon and an uncommon sight within many cities. The Dutch city of Zoetermeer is the first city in the Netherlands that has deployed these lampposts see Figure 84. The lamppost provides, via a direct connection to the city's main broadband network, free internet and local services. The lamppost allows users to access these services via a touch screen connected at the bottom of the lamppost but also provides Wi-Fi services to allow users to access the internet with their own mobile devices. The intelligent lamppost in this case consists out of a wireless communication device and a broadband device that forwards requests for information. This simple design however allows for multiple concurrent wireless connections with a maximum transmission range of one hundred meters around the device. The actual transmission range can however vary depending on the environment.



Figure 84: The intelligent lamppost



The technological requirements that the CBPRS imposes on these intelligent lampposts is already available. The only difference between the CBPRS and the intelligent lamppost as deployed in Zoetermeer is that we propose to exploit the existing power-line infrastructure over the installation of broadband network connections since these might not be available in every city. These intelligent lampposts can further be used to display commercial advertisements to fund their deployment even further; this idea is already applied in Zoetermeer and can be used to cover deployment and operational expenses.

## II.3 Power Line Communication

Broadband over Power Lines (BPL) is a relatively new form of a broadband communication network infrastructure that utilizes electricity cables to provide internet access for consumers. The BPL system allows users to utilize electricity sockets as network connection points for internet access. These socket connections can achieve communication speeds of 45 Mbit/s after passing a repeater. Newly demonstrated techniques have shown that speeds in excess of one Gbits/s are among future possibilities. Different IEEE groups have published standards or draft standards concerning power line and BPL communication. The most interesting of these is the draft standard IEEE P1901 [36] that attempts to define a medium access control and physical layer for communication via electricity cables. The IEEE P1901 draft standard should standardize the data link layer of the Open systems interface (OSI) reference model and thereby enable the design of communication protocols that resemble those used on current computer networks instead of propriety solutions commonly used today.

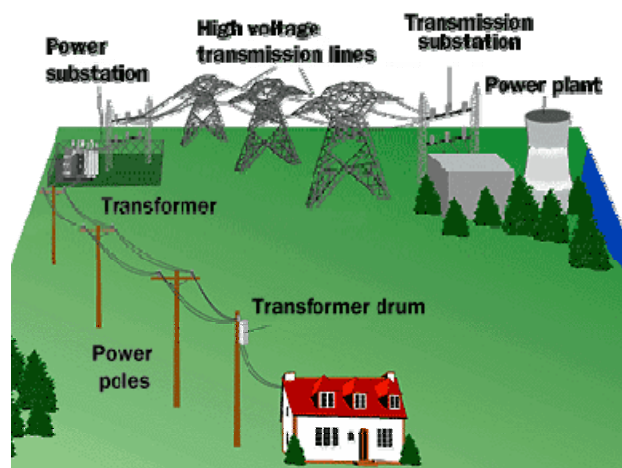


Figure 85: Power line infrastructure

Figure 85 displays how the power line infrastructure is organized. The signals transmitted and received via BPL cannot be passed through a distribution transformer that connects the local area electricity network with the high voltage electricity backbone. Therefore, extra equipment should be placed in neighborhood power distribution stations that filter the BPL signals out of the electricity lines before they enter the distribution transformer and reinsert them back in to the electricity line after the distribution transformer. The costs associated with this extra investment in the required hardware are significantly lower in Europe than in the United States due to the fact that neighborhood distribution facilities in Europe distribute electricity to hundreds of homes whereas in the United States only a few homes are connected per station. Power line broadband connectivity is however to increase faster in the United States than in Europe this to the lack of a proper infrastructure for other broadband services in the rural areas of the United States [37].

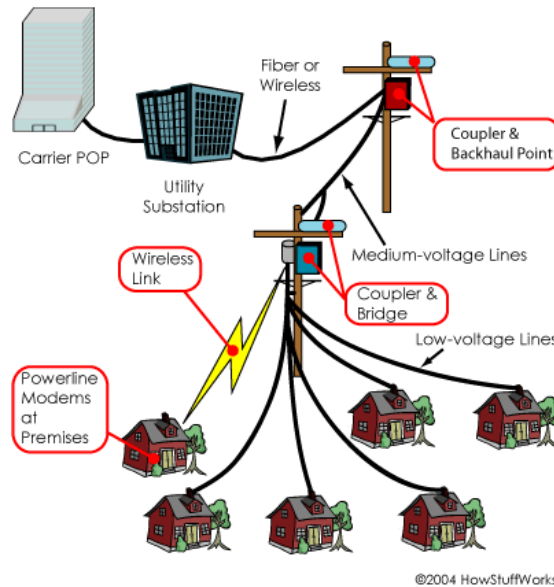


Figure 86: Power line broadband overview © HowStuffWorks

The CBPRS exploits the presence of the power line infrastructure and the manner in which the lampposts are connected to each other to form a citywide low cost high bandwidth network. This network can be organized without interfering or jeopardizing the power supply of households in any way. Investments to deploy such a network are only required in the utility substations where data should be extracted in inserted into the power line infrastructure and in deploying the intelligent lampposts. Furthermore, a proprietor of the CBPRS should also pay a fee in order to use the power line network.

## II.4 Parking place sensors

Parking place sensors used by the CBPRS provide limited functionality and could be omitted if the CBPRS controls all parking places within the city and all vehicles are subscribed to the system. However, since this is not likely the case upon deploying the CBPRS within a city the parking place sensors are required to indicate to other drivers if the parking place is available for parking or reserved for another driver. Since actual parking place sensors do not exist at present, the sensors and the technology proposed here only serves as an indication of how such a sensor could operate.

Parking place sensors are required to monitor the occupancy of the parking place, communicate change in occupancy to the intelligent lamppost that is the topological parent of the parking place sensor and inform other users of the status of the parking place. Once a vehicle parks on the parking place, the sensor should detect the presence of the vehicle and communicate this event to the intelligent lamppost that monitors a set of parking place sensors in its vicinity. The detection process of the vehicle arriving or leaving could be done by using light sensitive sensors, sensors that monitor the magnetic fields around the parking place, sensors that monitor detection, or by emitting and receiving radio waves. Light sensitive sensors and movement sensors are out of the four mentioned the least likely, this due to the fact that parking place sensors are incorporated into the road surface. This can lead to situations where the sensor is blocked by leaves or debris and could lead to false information streams into the CBPRS. Magnetic field sensors are more applicable since they can be configured to detect the anomaly in the magnetic field that occurs once a vehicle arrives. However, interference in the magnetic fields caused by the city infrastructure can also lead to false information streams. Radio waves emitted according to the radar principle are the most certain way to detect vehicles parked on the parking place. This method could however require large amounts of electrical power that cannot not be supplied by a parking sensor. The most probable solution to this problem will probably be found in a mixture of one of these technologies with a combination between the magnetic sensor and the radar principle being the most probable.

Communication between a parking place sensor and its controlling lamppost is another problem. The spacing between lampposts on Dutch inner-city roads is about 25 meters; however using all lampposts within a city would make the CBPRS infeasible due to the high costs associated with such an extensive network of intelligent lampposts. The most likely situation therefore to occur is that one intelligent lamppost will monitor a multitude of parking place sensors within a specific street. Therefore communication between the parking place sensor and the intelligent lamppost should be able to exceed distances of a few meters and should most preferably extend between fifty and one hundred meters. The only probable solution for these requirements is making use of wireless communication in the same manner as participating vehicles do.

The parking place sensor as a whole should preferably be incorporable within the road surface without any major infrastructural work such as digging new power line cables to facilitate power demands of the parking sensor. The optimal solution for the parking place sensors would incorporate self-sufficient devices. A number of solutions are available to overcome this problem. The first solution would be to incorporate a small solar panel and battery, however electricity gained from these might not be sufficient with a small surface available. The second solution would be to incorporate a small fuel cell into the parking place sensor. The electricity provided by the fuel cell [38] would certainly meet the requirement of the parking place sensors however; the fuel for the cell should be refilled every now and then. A parking place sensor could indicate when a fuel refill is required by communicating via the CBPRS. However, the actual refilling process remains manual and leads to an extra financial burden on the CBPRS proprietor. The third possibility is wireless power transfer. While this technology has not yet reached the mainstream market, the possibilities provided are applicable to parking place sensors. Vehicles that participate in the CBPRS could be equipped with a device to transfer power between the vehicles battery and the battery of the parking place sensor. This solution is only feasible if the frequency of parking place occupancy over all parking places is high enough to keep the parking place sensors operational. The key to this problem however remains the moderate use of electricity on the side of the parking place sensor.



## User Manual

This user manual describes the graphical user interface (GUI) of the simulation environment called 'Cityscape'. The Cityscape program combines the functionality provided by the different layers of the simulation environment in one single application while not sacrificing on modularity. The Cityscape application is a tear-off application build by combining the functionality of the different layers (simulation, infrastructure, communication, routing and parking) that is made available via a set of dynamic linking libraries (DLL).

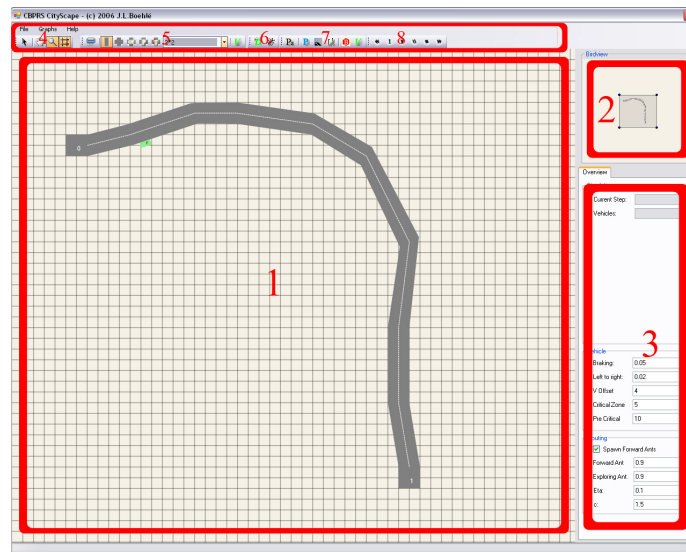


Figure 87: Cityscape GUI

Figure 87 displays the GUI of the Cityscape application subdivided into four blocks of functionality enclosed by a red rectangle. Block number one represents the visual workspace in which the user is able to construct the network and visualize simulations. Block number two contains a 'bird view' overview of the entire city and allows the user to zoom and scroll over the entire road network. Block three contains simulation information such as the current simulation step and the input parameters of the simulation environment that can be altered before or during a simulation; the meaning of these parameters is described in detail in chapter seven and will be discussed here. The final block containing the numbers four to eight is the main menu area of the simulation and is discussed in detail in section one.

This chapter further discusses the creation of a new road network and the procedures of loading and saving a simulation in section two. The interfaces used for configuring intersections and roads in respectively section three and section four. Finally section five discusses the manner in which information gathered via a simulation can be extracted from the Cityscape application.

## III.1 Toolbars

The layers in the simulation environment present their functionality to the user via a set of toolbars that allow the user to configure them or create objects within the simulation environment. This section describes the purpose of each toolbar and gives an explanation of the buttons present on these toolbars and the manner in which they influence the simulation environment.

### III.1.1 Interaction Toolbar

The interaction toolbar, shown in Figure 87 under number four, allows the user to interact with the visual representation of the simulation via three different types of mouse interaction event methods, selection, dragging and zooming. The buttons on this toolbar allow for the interaction of the user with roads and intersections, their precise function is described below.



**Selection:** This button enables the mouse selection event. When this button is active the mouse pointer can be used to select object within the simulation environment. Selection of an object within the simulation will display the configuration screen of that object.



**Dragging:** This button enables the mouse dragging event. When this button is enabled the mouse pointer can be used to drag intersections over the simulation environment map.



**Zooming:** This button allows the user to zoom in and out on the simulation environment map. The zooming function controls the detail of the simulation, when zooming in or in default zoom full detail renderings are presented to the user. When zooming out the detail of the simulation is gradually decreased until only the contours of the roads and intersections remain visible.



**Grid:** This button displays or hides the grid to which all intersections are aligned. The grid is automatically disabled when the user zooms out beyond a certain threshold. During configuration of the simulation environment it is recommended that the grid is used, during simulations the grid should be disabled to minimize unnecessary drawing operations.

### III.1.2 Infrastructure Toolbar

The infrastructure toolbar, shown in Figure 87 under number five, provides the user with means to create a road network infrastructure. The buttons in this toolbar however only provide the means to create roads and intersections, the configuration of these roads and intersection should be done by the user via the appropriate configuration screens or left to the simulation environment. The functions of the buttons on the infrastructure toolbar are described below.



**Shredder:** Pressing this button enables the shredding mode of the simulation environment. In this mode the user is able to destroy roads and intersections that are no longer required in the road network are that were placed by accident and are unwanted at present.



**Road:** This button enables the road construction process between two intersections currently present on the map. Once this button is enabled sections of road can be added by clicking the left mouse button and removed by clicking the right mouse button. The precise manner in which this tool operates is described in the construction section of this chapter.



**Rule-less Intersection:** This button enables the construction of a rule-less intersection. This basic type of intersection does not apply any rules to vehicles attempting to cross the intersection. To prevent collisions on the intersection this intersection randomly picks a vehicle from the set of conflicting vehicles and allows that one to cross the intersection. The use of this intersection in simulations is not recommended since the behavior of this intersection does not mimic realistic behavior.



**Generator Intersection:** This button enables the construction of generator intersections. Generator intersections are intersections that spawn a number of vehicles at each time step depending on the configuration settings made by the user, by default these intersections do not generate vehicles. Only one road can be connected to a generator intersection since the generator does not allow incoming vehicles to cross but instead consumes them and discards them from the simulation environment. When placing generators one should position them in such a manner that they can represent starting points for highways originating in other cities or sections of densely populated areas in cities from which vehicles spawn during certain time and end their journey at another time.



**Precedence Intersection:** This button enables the construction of precedence intersections. The precedence intersection applies precedence rules to vehicles attempting to cross the intersection. The precedence rules can be configured automatically by the simulation environment on the basis of the type of roads meeting at the intersection, or manually via the intersection configuration screen.



**Traffic Light Intersection:** This button enables the construction of traffic light intersections. The traffic light intersection allows vehicles to cross on the basis of a traffic light cycle matrix. This matrix is constructed in such a manner that lanes that allow vehicles to cross the intersection at the same time are set to green in the same cycle allowing the optimal amount of vehicles to pass. The traffic light cycle matrix is configured automatically by the simulation environment, traffic light signal times can be configured by the user via the intersection configuration screen.



**Network Validation:** This button starts the road network validation process by attempting the configuration of the roads and intersections present in the network. This process ignores intersections that have been configured by the user. Once this process is complete the user is given notice of the state of the road network and in the case of invalidity is allowed to modify the road network.

### III.1.3 Routing Toolbar

The routing toolbar, shown in Figure 87 under number six, allows the user to initiate the setup of the routing algorithms present in the simulation environment. The setup of the routing algorithms cannot be influenced via these buttons. This should be done during the configuration of the infrastructure by setting the types of roads and the maximum speed allowed. The buttons on the routing toolbar are described below.






**Configure Dijkstra:** This button starts the configuration process of Dijkstra's Algorithm. The routing tables for the algorithm are constructed on the information present in the road network and can be viewed once the process is completed in the intersection configuration screen. Before construction the environment asks if the user intended the operation, since the construction of the routing table can consume a rather large amount of time for large road networks.



**Configure Ant-Routing:** This button configures the ant-routing algorithm by constructing the hierarchy of colonies and the placement of nodes near intersection. The configuration process fills the routing tables with default values. This however means that the algorithm does not immediately yield shortest paths but should be given some time to settle.





### III.1.4 Parking Toolbar

The parking toolbar shown in Figure 87 under number seven contain three buttons that allow the user to review parking system statistics, add parking places and garages to roads and create bus routes.

-  **Parking System:** This button allows the user to review the state of the CBPRS parking system. The overview presented by pressing this button allows for modification of available parking departure slots and review which vehicles are stored in which slot. Furthermore, a brief is giving of the statistics gather by the parking system in terms of places available, occupied, reserved and the mean parking time.
-  **Add Parking places:** This button allows the user to add parking places and garages to roads. Once this button the 'add parking place' feature is enabled parking places and garages can be added by selecting a road. Once the road to which parking facilities should be added is clicked a screen is brought up that allows for easy addition of parking places and parking garages.
-  **Create Bus Stop:** This button enables the creation of bus routes and bus stops with the simulation environment. Once the button is clicked the feature is enabled. To use this feature simply click to road to which the bus stop should be added. A dialog is then presented allow the user to select the appropriate bus route, the side of the road on which to add the bus stop and the positioning of bus stop along the road. Please note that in order to have busses riding along the bus routes the 'Create Busses' checkbox, that is located under number three in Figure 87, should be checked.

### III.1.5 Simulation Toolbar

The simulation toolbar, shown in Figure 87 under number eight, allows the user to control the execution of a simulation. The speed at which a simulation step executes is depended on the computer on which the simulation is executed, however the user is able to influence the number of steps the environment attempts to execute per second. The manner in which the simulation is influenced by the user when pressing a button on the simulation toolbar is described below.

-  **Start Simulation:** Starts the simulation of vehicles and the routing system. Before a simulation can start however the user must have loaded or constructed a road infrastructure network. In the case that the user has constructed a new road network or modified a loaded network validation is required by pressing the  located on the infrastructure toolbar. If the road network is valid the simulation starts executing at a processing speed which depends on the complexity of the road network.
-  **Single Step:** On pressing this button the simulation environment executes a single simulation step. The use of this button requires that a simulation is executing or paused. Once the simulation of the step has been completed the simulation is paused giving the user the option to simulate another step, continuing the simulation or stopping it.
-  **Stop Simulation:** Pressing this button stops the execution of a running simulation by aborting simulation procedures once the step that is currently executing has finished. Thus pressing this button does not stop the simulation immediately but rather allows all process to finish their current executions so that the data collected by the simulation environment is complete. The stop button does not reset the simulation; the reset process is executed before the simulation is started to allow the user to view and store data gathered from a simulation.





**Accelerate Simulation:** Decreases the period of real time the simulation environment waits after the completion of a step. Under default settings the simulation attempts to execute ten simulation steps per second. Accelerating the environment increases this pace however after each simulation step the environment relinquishes control to the operating system in order not to burden it beyond its means. This setup allows all running operating system processes to run in harmony at the cost of a small fraction of the maximum achievable execution speed. It is however important to notice that accelerating the simulation environment beyond a certain computer dependant maximum pace graphical errors can start to occur. To prevent this from happening disable visual rendering (see option panel) before running the simulation environment at maximum speed.



**Decelerate Simulation:** Increase the period of real time the simulation environment waits after the completion of a step. Under default settings the simulation attempts to execute ten simulation steps per second. Decelerating the simulation environment pace allows for a detailed study of vehicular behavior.




## III.2 Cityscape construction

The Cityscape program once executed is started in editor mode by default but supports transparent switching to simulation mode. The creation of the road network infrastructure of a city is done by means of the infrastructure toolbar described in the previous section. The following describes the default procedure.

The creation of a city environment starts with the construction of the first intersection, this intersection can be any of three types available on the infrastructure. Before pressing the button of the desired intersection the size of the new intersection needs to be specified, this size being a minimum of 2\*2 and maximum of 6\*6. The size of the intersection indicates the maximum number of lanes a road can have when connecting to an intersection, thus 2\*2 means that a road of incoming lane and one outgoing lane can be connected to the four connection points of the intersection; top, right, bottom and left. This size specification is required due to rendering restrictions and is in no way related to the design of the underlying infrastructure layer.



Figure 88: Road construction

Once one or two intersections are positioned in the simulation environment and dragged around via the drag tool , the process of creating roads can commence. Via the activation of the road tool  the mouse is configured to create roads. To determine the path of a road click on the side of the intersection at which the road should start, then the user has two options: the first one is clicking on another intersection to create a straight road between them; the second option is to define a precise path for the road. The definition of a path between two intersections is a simple matter of clicking on an empty position on the grid. After a click the simulation environment draws a white circle and a line displaying the path the road follows, see Figure 88. Roads created via this way can have any shape imaginable, however the environment does not alter the path once constructed by the user; therefore sharp angles should be prevented in order to keep the graphical representation of the road decent. The infrastructure layer however does not care about these points and guarantees that vehicles will travel along the road in accordance to the length and maximum speed. The path points of a road can be erased during construction by pressing the right mouse button, once the road is created use the shredder  to erase the road from the infrastructure.

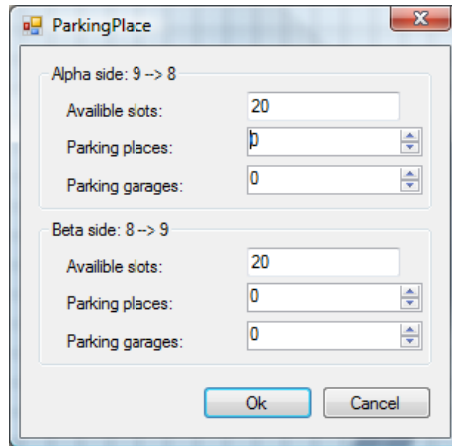








Figure 89: Parking place addition screen

Once the configuration of the city road network is complete, parking places and bus stops can be added to environment. By selecting the 'add parking place' button from the parking toolbar and clicking on a road to which parking places should be added the dialog in Figure 89 is shown. The dialog shows the available slots alongside each roadside this number depicts the maximum amount of parking places or parking garages to be placed at that side of the road. By clicking the arrows on the right side of the screen the number of parking places to be added can be influenced. Pressing the **Ok** will the parking places and garages at random positions along the selected roadside. These newly created parking places and garages will be automatically added to the CBPRS parking system and require no further user configuration before usage in a simulation.

The simulation environment can be saved at any stage during the process of construction and the user is advised to save often. Via the **File** menu the option **Save** can be located. This option presents a save as dialog, this dialog allows the user to specify a directory and a filename the simulation environment is compressed and then saved to disk. Loading a simulation environment can also be done via the **File** menu and then the option **Load** this option presents a load from disk dialog, once a valid **CBPRS** file is located the simulation loads environment from disk and restores the environment to its state of saving.

Once the road network is constructed, configured, see section three and four, and saved it needs to be validated via the  button, this process is optional and failure to do so will show when initializing other layers and then only in case of errors in the infrastructure. After the road network has been validated the routing algorithms can be initialized by pressing the  and  buttons. On successful completion of the operations the user is notified and the environment is now ready for a simulation by pressing the  or  buttons.

### III.3 Intersection Interfaces

To display the intersection interface create an intersection and activate selection , then click on the desired intersection. The intersection interface is loaded and displayed to the user. The default intersection interface displays the id of the intersection, the roads and their connected positions, the number of outbound lanes and the number of inbound lanes on the current intersection. By clicking the view button next to connected the road interface for that road will be loaded and displayed. The interface does not allow any modifications to made. Changes in the behavior of the intersection must be made via roads and lanes connected to the intersection.

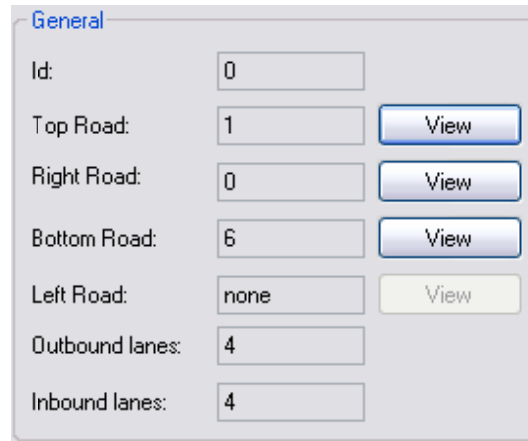


Figure 90: Default intersection interface


In order to modify behavior of specific intersection implementations like the precedence intersection or the traffic light intersection expand the nodes on upper left of the interface screen. This node-list lists depending on the type of intersection the different possibilities to alter the behavior. The process for altering is straight forward and self explanatory therefore it is not discussed here.

Virtual Nodes	Sector 0	Sector 1
	To/Via	1
▶ 0		1
1		1

Figure 91: Ant routing table

The ant routing tables are coupled to intersections and can be viewed by selecting the **AntNode** option in the node list on the left of the intersection interface. The **virtual nodes** tab displays the sector numbers known to the node and the next possible nodes on the route to the sectors. Then for each sector the intersection is a member of their exists a local routing table containing all sector nodes the possible next nodes and their probabilities. Pressing the **Refresh** button on the lower right will update the screen with current values.

### III.4 Road Interfaces

To display the road interface construct a road between two intersections and activate selection , then click on the road. This will display the road interface screen listing the different types of interfaces available for the current road. During the configuration of a simulation a typical user will interact heavily with road interface. The road interface allows the user to define the type of the road, the maximum speed and the driving lanes.

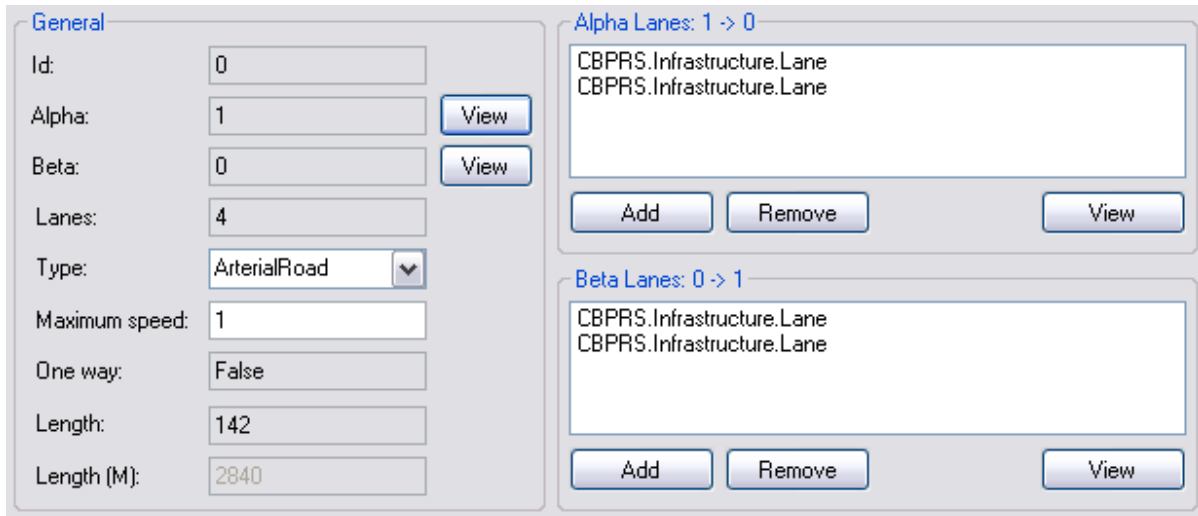


Figure 92: Road Interface

The General group list the information of the current road by stating its id, the ids of the intersections to which the road is connected, the number of lanes, the type of road, the maximum speed in blocks per second, whether or not the road is a one way road, the length of the road in block of 7.5 meters long and the length of the road in meters.

The only two configurable parameters in this group are the road type and the maximum speed. The road type eventually determines how the sector dividing process of the process of the algorithm described in chapter four executes. All road types of arterial and above are seen as sector boundary roads while all road types below are placed within a sector. The configuration of the road type is therefore highly important in relation to the effectiveness of the routing algorithm.

The Alpha lanes and Beta lanes groups list the driving lanes leading from the alpha to the beta intersection and vice versa. The alpha intersection is the intersection the user selected first when constructing the road. The terms alpha and beta are used internally to extinguish the start and end of the road and have no impact on the behavior of traffic. Clicking the **add** button allows for the addition of an extra to the road, the maximum number of lanes at present is three. The **remove** button removes the currently selected lane. Clicking the **view** button brings up a new interface showing lane information.

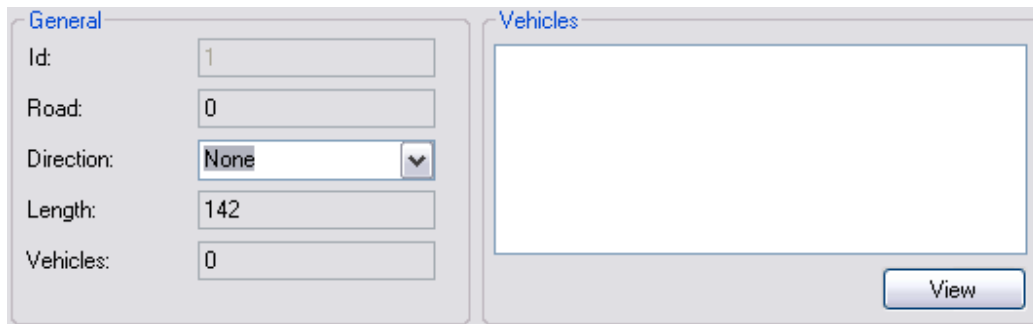


Figure 93: Lane Interface

The lane interface is one of the least complicated interfaces in the environment. However the modification of the *Direction* parameter has serious consequences for the behavior of vehicles near intersection. When constructing a lane the direction parameter is set to *none* this indicates that simulation environment is responsible for determining the direction in which vehicles are allowed to turn at the intersection.

The direction algorithm in the simulation environment configures the lanes in such a manner that the set of driving lanes on the current road leading to the intersection allows vehicles to enter all other roads connected to the intersection unless they do not have outgoing driving lanes leading from the current intersection. There are however situations imaginable where this behavior is unwanted, therefore the direction algorithm does not alter directions set by a user. To setup alternate directions select the direction in which vehicles are allowed to turn from the **direction** drop-down list and select the desired turn. Note that setting invalid directions will lead to errors when simulating as the algorithm currently does not verify directions set by users.

The vehicles group allows the user to view information about a vehicle currently travelling on the driving lane. The vehicle interface does not provide any user configurable options and is therefore not discussed in the manual. Note however that this is the only way to consult vehicle related information on a per vehicle basis it is not possible to select vehicle directly by clicking on them.

### III.5 Simulation Data Extraction

The simulation constantly gathers data concerning events taking place within the simulation environment. However, the manner in which this data can be extracted from the simulation environment differs depending on the information gathered and the frequency of information gathering. While most data is kept within the simulation environment during the entire simulation, the information gathered by vehicles is written to disk during the simulation process. Once the simulation is completed, the data kept within the simulation can also be written to disk by pressing the corresponding data save button in the simulation data extraction screen found under the **file** menu. It is however important to notice that data gathered, with the exception of the vehicular data written to disk during the simulation, is lost when the application is closed before the data is properly saved to disk.

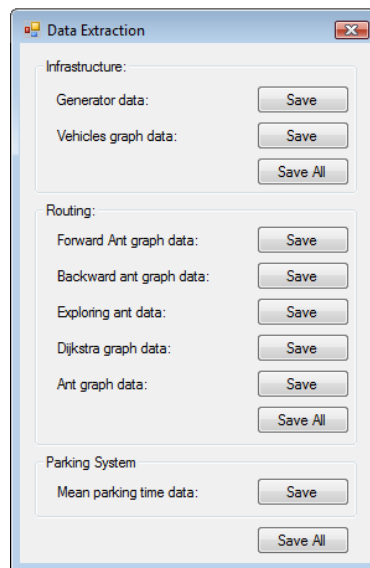


Figure 94: Data extraction screen

The data extraction screen as is shown in Figure 94 allows the user to save data gather by the simulation environment by clicking the button next to the data descriptions. The user is presented with a save dialog to allow him to specify a directory for saving. All data saved via this screen is stored with the '.csv' extension to allow for one click importing in Microsoft Office Excel versions 97 through 2007.



## Infrastructure Fact sheet

Infrastructural types	Descriptions
<i>Intersections</i>	<i>Description</i>
<ul style="list-style-type: none"> <li>- <b>Default intersection</b></li> <li>+ Connects to max 4 roads</li> <li>+ Conflict matrix</li> <li>+ Messaging system</li> <li>+ Random precedence rule</li> </ul>	<p>Default intersection implementation, provides a basic intersection implementation for all other intersection to extend. This basic implementation allows a maximum of roads to be connected. Furthermore, this implementation provides functionality to determine the vehicle crossing conflict matrix and contains a full implementation the intersection messaging system. The intersection rule-set implemented in this intersection allows multiple vehicles the cross when no conflict between crossings is determined. If a conflict is determined between two or more vehicles, the rule-set specifies that a randomly chosen vehicle should be given precedence. This intersection is not used during experiments.</p>
<ul style="list-style-type: none"> <li>- <b>Precedence intersection</b></li> <li>+ Connects to max 4 roads</li> <li>+ Conflict matrix</li> <li>+ Messaging system</li> <li>+ 'normal' precedence rules</li> </ul>	<p>The precedence intersection is based upon the default intersection and extends its behavior by implementing a 'normal' precedence rule-set. The rule set implemented consists out two rules: 1) Right of way should always be given to vehicles approaching on the right hand road. 2) Right of way should always be given to vehicles approaching on the opposite road when making a left turn and the vehicle on the opposite road wants to make a right turn or intends to go straight ahead, unless the opposite road has a lower precedence.</p>
<ul style="list-style-type: none"> <li>- <b>Traffic Light intersection</b></li> <li>+ Connects to max 4 roads</li> <li>+ Conflict matrix</li> <li>+ Messaging system</li> <li>+ Traffic light cycles rules</li> </ul>	<p>The traffic light intersection is based upon the default intersection and extends its behavior by implementing a rule-set based on traffic lights. The traffic light intersection places a traffic light at each incoming lane of a road that leads onto the intersection. Then by determining which combinations of traffic lights would allow conflict free travels across the intersection it determines a traffic light schedule. Traffic lights are switched depending on the cycles they are a member – one traffic light can be a member of any number of cycles if the route it monitors does not conflict with other traffic light routes in that cycle – off in the following order green fifteen seconds, yellow five seconds and red twenty seconds.</p>
<ul style="list-style-type: none"> <li>- <b>Generator / Exit intersection</b></li> <li>+ Connects to max 1 road</li> <li>+ No conflict matrix</li> <li>+ messaging system</li> <li>+ Generator</li> </ul>	<p>The generator exit is a special type of intersection that allows only one road to be connected. The intersection is equipped with a generator that enables the creation of different types of vehicles during the simulation period. These generated vehicles are placed upon the road that is connected to the intersection. This intersection also signals the end of the modeled part of the simulation environment by implementing an exit. This exit destroys vehicles that enter the intersection from the connected road. Before destroying the vehicles the exit retrieves statistical information concerning the route of the vehicle to be destroyed.</p>
<i>Roads</i>	<i>Description</i>
<ul style="list-style-type: none"> <li>- <b>Road</b></li> <li>+ Bi/uni-directional lanes</li> <li>+ Maximum vehicle speed</li> <li>+ Lane changing support</li> <li>+ Connect max 2 intersections</li> <li>+ Sideobject support</li> </ul>	<p>The road connects a maximum of intersections to each other between which traffic can move via traffic lanes. The road maintains two collections of lanes to support bidirectional travel. The road defines a maximum speed that is applicable to all lanes contained by the road. The changing of lanes is assisted by the road by providing simple lookup support to find the correct for a certain vehicle although the vehicles do the actual lane changing. The road allows side objects such a lamppost and parking place to be located along either side of the road.</p>
<ul style="list-style-type: none"> <li>- <b>Lane</b></li> <li>+ Blockwise movement</li> <li>+ Turning directions</li> <li>+ Vehicle container</li> </ul>	<p>The lane enables to move between two intersections. The lane itself is contained by a road that defines the place of the lane in the environment and the maximum speed allowed. The lane itself defines a certain length and maintains a linked-list of vehicles that are currently driving along the traffic lane. To assist the vehicles during routing each lane also defines the turns allowed from onto other lanes at the next intersection.</p>

<b>Infrastructural types</b>	<b>Descriptions</b>
<i>Vehicles</i>	<i>Description</i>
<ul style="list-style-type: none"> <li>- <b>Vehicle</b></li> <li>+ <i>Adjustable max. speed</i></li> <li>+ <i>Adjustable vehicle length</i></li> <li>+ <i>Accident free movement</i></li> </ul>	Default vehicle implementation, provides a basic implementation for all other vehicles within the simulation environment. This basic implementation allows the vehicle to obey speed regulations, change lanes and enables the use of signals for breaking and turning. Furthermore, this implementation guarantees accident free movement. This vehicle chooses a random path when driving through the simulation environment and is therefore not used during the experiments.
<ul style="list-style-type: none"> <li>- <b>DijkstraVehicle</b></li> <li>+ <i>Based on Vehicle</i></li> <li>+ <i>Routing via Dijkstra</i></li> </ul>	Vehicle that inherits its behavior from the default vehicle and uses the information stored in routing tables generated by Dijkstra's algorithm to route itself between two points within the environment.
<ul style="list-style-type: none"> <li>- <b>DijkstraParkingVehicle</b></li> <li>+ <i>Based on DijkstraVehicle</i></li> <li>+ <i>Routing via Dijkstra</i></li> <li>+ <i>Parking behavior rules.</i></li> </ul>	Vehicles similar to the previous but extended routing and driving behavior enables the vehicle to park in a parking place. This extended behavior allows the vehicle to execute a parking maneuver and turn 180 degrees if the desired parking place resides on the other side of a two lane bi-directional road.
<ul style="list-style-type: none"> <li>- <b>AntBasedVehicle</b></li> <li>+ <i>Based on Vehicle</i></li> <li>+ <i>Routing via Ant algorithm</i></li> </ul>	Vehicle that inherits its behavior from the default vehicle. This vehicle collects traffic condition information for the CBPRS and request routing information from the CBPRS in order to find its route through a city environment.
<ul style="list-style-type: none"> <li>- <b>AntBasedParkingVehicle</b></li> <li>+ <i>Based on AntVehicle</i></li> <li>+ <i>Routing via Ant algorithm</i></li> <li>+ <i>Parking behavior rules.</i></li> </ul>	Vehicles similar to the previous but extended routing and driving behavior enables the vehicle to park in a parking place. This extended behavior allows the vehicle to execute a parking maneuver and turn 180 degrees if the desired parking place resides on the other side of a two lane bi-directional road.
<ul style="list-style-type: none"> <li>- <b>Bus</b></li> <li>+ <i>Based on DijkstraVehicle</i></li> <li>+ <i>Routed via Dijkstra</i></li> <li>+ <i>Provides CBPRS traffic info.</i></li> </ul>	Vehicle based on the default implementation that travels along a route of bus stops. The bus uses routing tables generated by Dijkstra's algorithm to find the most optimal route between two bus stops. During its travels, the bus collects traffic condition information concerning the roads it travels and shares this with the CBPRS.