UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

ANDRÉ LUIS OGANDO PARAENSE

A MACHINE CONSCIOUSNESS APPROACH TO URBAN TRAFFIC SIGNAL CONTROL

UMA ABORDAGEM DE CONSCIÊNCIA DE MÁQUINA AO CONTROLE DE SEMÁFOROS DE TRÁFEGO URBANO

Campinas

2016

ANDRÉ LUIS OGANDO PARAENSE

A MACHINE CONSCIOUSNESS APPROACH TO URBAN TRAFFIC SIGNAL CONTROL

UMA ABORDAGEM DE CONSCIÊNCIA DE MÁQUINA AO CONTROLE DE SEMÁFOROS DE TRÁFEGO URBANO

Thesis presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering, in the area of Computer Engineering.

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica na Área de Engenharia de Computação.

Supervisor: Ricardo Ribeiro Gudwin

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA PELO ALUNO ANDRÉ LUIS OGANDO PARAENSE E ORIENTADA PELO PROF. DR. RICARDO RIBEIRO GUDWIN.

Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): CNPq, 153206/2010-1

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

P212m

Paraense, André Luis Ogando, 1983-
A machine consciousness approach to urban traffic signal control / André Luis Ogando Paraense. – Campinas, SP : [s.n.], 2016.

Orientador: Ricardo Ribeiro Gudwin.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Tráfego urbano. 2. Semáfóro. I. Gudwin, Ricardo Ribeiro,1967-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Uma abordagem de consciência de máquina ao controle de semáforos de tráfego urbano
Palavras-chave em inglês:
Urban traffic
Semaphore
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora:
Ricardo Ribeiro Gudwin [Orientador]
Marcio Lobo Netto
Carlos Henrique Costa Ribeiro
Fernando Antônio Campos Gomide
Eric Rohmer
Data de defesa: 29-03-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: André Luis Ogando Paraense RA:008076

Data da Defesa: 29 de março de 2016

Título da Tese: "Uma Abordagem de Consciência de Máquina ao Controle de Semáforos de Tráfego Urbano"

Prof. Dr. Ricardo Ribeiro Gudwin (Presidente, FEEC/UNICAMP)

Prof. Dr. Marcio Lobo Netto (EPUSP/USP)

Prof. Dr. Carlos Henrique Costa Ribeiro (ITA)

Prof. Dr. Fernando Antônio Campos Gomide (FEEC/UNICAMP)

Prof. Dr. Eric Rohmer (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

To my family.
For their support and
understanding.

À minha família.
Pelo apoio e compreensão.

Acknowledgement

I would like to thank:

My family for all the support and understanding. Anything good that I have become is because of my mother Delfina, my father Odilon and my sister Mariana. To me, they have always been the synonym of love. This thesis is dedicated to them.

Prof. Dr. Ricardo Ribeiro Gudwin for his guidance. Artificial Intelligence is a vast field, and without the guidance provided by him, I would have easily got lost. His science knowledge and engineering skills have always been a constant source of motivation.

The philosopher Bernard J. Baars for providing the intellectual framework behind this thesis and the many other scientists on whose shoulders I somehow stand on in this work.

My colleagues from the CogSys group at FEEC. Specially to Klaus Raizer, for our combined efforts at developing the foundation of the cognitive architecture. I would also like to thank Suelen Mapa for helping me in running the experiments at our lab.

All the professors at FEEC (Faculdade de Engenharia Elétrica e de Computação) for all they have taught me on Computer Engineering, specially to Fernando Von Zuben, Romis Attux and Rafael Mendes, for the insights and guidance in this work.

CAPES, CNPQ and FAPESP, for funding support.

ABSTRACT

In this work, we present a distributed cognitive architecture used to control the traffic in an urban network. This architecture relies on a machine consciousness approach - Global Workspace Theory - in order to use competition and broadcast, allowing a group of local traffic controllers to interact, resulting in a better group performance. The main idea is that the local controllers usually perform a purely reactive behavior, defining the times of red and green lights, according just to local information. These local controllers compete in order to define which of them is experiencing the most critical traffic situation. The controller in the worst condition gains access to the global workspace, further broadcasting its condition (and its location) to all other controllers, asking for their help in dealing with its situation. This call from the controller accessing the global workspace will cause an interference in the reactive local behavior, for those local controllers with some chance in helping the controller in a critical condition, by containing traffic in its direction. This group behavior, coordinated by the global workspace strategy, turns the once reactive behavior into a kind of deliberative one. We show that this strategy is capable of improving the overall mean travel time of vehicles flowing through the urban network. A consistent gain in performance with the "Machine Consciousness" traffic signal controller during all simulation time, throughout different simulated scenarios, could be observed, ranging from around 10% to more than 20%, when compared to the "Parallel Reactive" controller without the artificial consciousness mechanism, producing evidence to support the hypothesis that an artificial consciousness mechanism, which serially broadcasts content to automatic processes, can bring advantages to the global task performed by a society of parallel agents working together for a common goal.

Keywords: Global Workspace Theory, Traffic Lights Control, Machine Consciousness, Codelets.

RESUMO

Neste trabalho, apresentamos uma arquitetura cognitiva distribuída usada para o controle de tráfego em uma rede urbana. Essa arquitetura se baseia em uma abordagem de consciência de máquina - Teoria do Workspace Global - de forma a usar competição e difusão em *broadcast*, permitindo que um grupo de controladores de tráfego locais interajam, resultando em melhor desempenho do grupo. A ideia principal é que controladores locais geralmente realizam um comportamento reativo, definindo os tempos de verde e vermelho do semáforo, de acordo com informações locais. Esses controladores locais competem de forma a definir qual deles está experienciando a situação mais crítica. O controlador nas piores condições ganha acesso ao workspace global, e depois realiza uma difusão em *broadcast* de sua condição (e sua localização) para todos os outros controladores, pedindo sua ajuda para lidar com sua situação. Essa chamada do controlador que acessa o workspace global causará uma interferência no comportamento local reativo, para aqueles controladores locais com alguma chance de ajudar o controlador na situação crítica, contendo o tráfego na sua direção. Esse comportamento do grupo, coordenado pela estratégia do workspace global, transforma o comportamento reativo anterior em uma forma de comportamento deliberativo. Nós mostramos que essa estratégia é capaz de melhorar a média do tempo de viagem de todos os veículos que fluem na rede urbana. Um ganho consistente no desempenho foi conseguido com o controlador "Consciência de Máquina" durante todo o tempo da simulação, em diferentes cenários, indo de 10% até mais de 20%, quando comparado ao controlador "Reativo Paralelo" sem o mecanismo de consciência artificial, produzindo evidência para suportar a hipótese de que um mecanismo de consciência artificial, que difunde serialmente em *broadcast* conteúdo para processos automáticos, pode trazer vantagens para uma tarefa global realizada por uma sociedade de agentes paralelos que operam juntos por uma meta comum.

Palavras-chave: Teoria do Workspace Global, Controle de Tráfego, Consciência de Máquina, Codelets.

List of Figures

1.1 Two lists: on one side the 20 biggest engineering achievements of the 20th century and on the other side the 14 biggest engineering challenges to be achieved in the 21st century. Source: National Academy of Engineering; Credits: Steve Wisbauer/Getty Images 22

1.2 Traffic jam in São Paulo, the biggest Brazilian city. 24

1.3 The scientific hypothesis is that consciousness brought competitive advantages to animals which permitted them to deal with unexpected situations and somehow focus attention in solving the most critical problem. 25

2.1 Junction illustrating vehicle flow relations (Guberinic et al., 2008) 30

2.2 Control variables (Guberinic et al., 2008) 32

2.3 Signal control phases (Guberinic et al., 2008) 33

2.4 Urban roads representation. Panel 2.4a shows the map and Panel 2.4b shows the corresponding graph. 37

2.5 Detailed junction. Panel 2.5a, adapted from Maroto et al. (2006), shows possible conversions, while Panel 2.5b shows the corresponding graph. . . . 38

2.6 Microscopic model structures 40

2.7 Road segment in the microscopic model. Arcs are divided in 7.5 m wide cells. Each car has a discrete velocity between 0, ..., $Vmax$. Vehicles represent ideal drivers, who follow security rules, never accelerating more than the gap to the next vehicle. 41

2.8 Arc scheme and quantities related to lane changing rules. Colored cells are occupied by vehicles. 43

3.1 Figure from Descartes' *De homine (1662)*, showing the location of the pineal gland, identified by the letter H. 50

3.2 Illustration of the thought experiment proposed by Searle called Chinese Room Experiment. 51

3.3 Art representing the Homunculus hypothesis. 53

3.4 In A, computing ϕ in simple models of neuroanatomy suggests that a functionally integrated and functionally specialized network—like the corticothalamic system—is well suited to generating high values of ϕ. In B, C and D, architectures modeled on the cerebellum, afferent pathways, and cortical-subcortical loops give rise to complexes containing more elements, but with reduced ϕ compared to the main corticothalamic complex. In E, ϕ peaks in balanced states; if too many or too few elements are active, ϕ collapses. In F, a bistable ("sleeping") system (same as in E), ϕ collapses when the number of firing elements (dotted line) is too high (high % activity), remains low during the "DOWN" state (zero % activity), and only recovers at the onset of the next "UP" state. Figure adapted from Tononi (2008). 56

3.5 In A, three simple prototypes of binary images of a dog, a helicopter and a table lamp are shown in the left, and nine variations of these in the right. In B, Euclidean distances between the prototypes and the variations are shown. The closest prototype is the one that is at minimum distance from the image. If we use Euclidean distance as a measure of similarity, most of the images are misclassified. Image 4, which is actually a helicopter, will be classified as a table lamp. It can be concluded from this plot that a Euclidean distance measure does not inform us about perceptual similarity. Figure adapted from George (2008). 58

3.6 Baars' interactive theater. The audience is shown as light yellow circles in the purple area, and the stage in gray with red and orange actors. The bigger the circles, the more activated they are. The dark yellow area is the area under the spotlight, representing the conscious processes, which broadcast their information. 62

3.7 The LIDA model 64

3.8 Machine consciousness is the emergence of a serial integrated information flow on top of a group of parallel interactive devices. 67

4.1 A Typical Cognitive Architecture, reproduced from Franklin & Graesser (1997). 69

4.2 The CST Core subsystem 72

4.3 Our Java implementation of the CST Core subsystem 74
4.4 The CST Reference Architecture 75

5.1 Network model “Simple T”, where the experiments took place. 93
5.2 Network model “Twin T”, where the experiments took place. 94
5.3 Network model “Corridor”, where the experiments took place. 94
5.4 Network model “Manhattan”, where the experiments took place. 95
5.5 Network model “Downtown Campinas”, where the experiments took place. 96
5.6 Network model of the sensed regions in front of the traffic lights, represented by letters (a) to (k), and the different possibles phases which can be chosen by the controller, drawn on top right, considering junction West and junction East. 97
5.7 Model of the controllers built on top of CST for the simplified Figure 5.6 network model. Four types of codelets were used. Three types were implemented for this specific application - a sensory codelet, a behavioral codelet and a motor codelet. Also, the consciousness codelet provided by CST (also a contribution of this work) was used in the model. In the example shown above, junction East is chosen by the consciousness codelet to gain the spotlight. The consciousness codelet does not change the information of the memory object of the junction East in the motor memory, but just highlights it as the conscious content. The output of the behavioral codelet junction East is then broadcasted to the global input B of the behavioral codelet junction West. 98
5.8 Conscious broadcast and interference mechanism. Panel 5.8a shows the parallel controllers acting. In panel 5.8b, the consciousness codelet finds the most critical junction and broadcast the information of its output memory objects (that is, the chosen phase) to all other junctions. Panel 5.8c illustrates an example of which junctions might decide to cooperate with the most critical junction in order to solve the problem it is facing. 102
5.9 Our Java implementation of the codelets implemented in the Machine Consciousness CST Architecture built to control the traffic signals. 105

6.1 Simple T Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.1a, 6.1b, 6.1c and 6.1d represent four distinct simulation experiments. 110
6.2 Simple T Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.2a, 6.2b, 6.2c and 6.2d represent four distinct simulation experiments. 111
6.3 Simple T Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.3a, 6.3b, 6.3c and 6.3d represent four distinct simulation experiments. 112
6.4 Simple T Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.4a, 6.4b, 6.4c and 6.4d represent four distinct simulation experiments. 113
6.5 Simple T Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.5a, 6.5b, 6.5c and 6.5d represent four distinct simulation experiments. 114
6.6 Simple T Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.6a, 6.6b, 6.6c and 6.6d represent four distinct simulation experiments. 115
6.7 Simple T Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.7a, 6.7b, 6.7c and 6.7d represent four distinct simulation experiments. 116
6.8 Simple T Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.8a, 6.8b, 6.8c and 6.8d represent four distinct simulation experiments. 117

6.9 Twin T Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.9a, 6.9b, 6.9c and 6.9d represent four distinct simulation experiments. 119
6.10 Twin T Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.10a, 6.10b, 6.10c and 6.10d represent four distinct simulation experiments. 120
6.11 Twin T Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.11a, 6.11b, 6.11c and 6.11d represent four distinct simulation experiments. 121
6.12 Twin T Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.12a, 6.12b, 6.12c and 6.12d represent four distinct simulation experiments. 122
6.13 Twin T Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.13a, 6.13b, 6.13c and 6.13d represent four distinct simulation experiments. 123
6.14 Twin T Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.14a, 6.14b, 6.14c and 6.14d represent four distinct simulation experiments. 124
6.15 Twin T Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.15a, 6.15b, 6.15c and 6.15d represent four distinct simulation experiments. 125
6.16 Twin T Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.16a, 6.16b, 6.16c and 6.16d represent four distinct simulation experiments. 126

6.17 Corridor Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.17a, 6.17b, 6.17c and 6.17d represent four distinct simulation experiments. 128
6.18 Corridor Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.18a, 6.18b, 6.18c and 6.18d represent four distinct simulation experiments. 129
6.19 Corridor Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.19a, 6.19b, 6.19c and 6.19d represent four distinct simulation experiments. 130
6.20 Corridor Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.20a, 6.20b, 6.20c and 6.20d represent four distinct simulation experiments. 131
6.21 Corridor Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.21a, 6.21b, 6.21c and 6.21d represent four distinct simulation experiments. 132
6.22 Corridor Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.22a, 6.22b, 6.22c and 6.22d represent four distinct simulation experiments. 133
6.23 Corridor Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.23a, 6.23b, 6.23c and 6.23d represent four distinct simulation experiments. 134
6.24 Corridor Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.24a, 6.24b, 6.24c and 6.24d represent four distinct simulation experiments. 135

6.25 Manhattan Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.25a, 6.25b, 6.25c and 6.25d represent four distinct simulation experiments. 136
6.26 Manhattan Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.26a, 6.26b, 6.26c and 6.26d represent four distinct simulation experiments. 137
6.27 Manhattan Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.27a, 6.27b, 6.27c and 6.27d represent four distinct simulation experiments. 138
6.28 Manhattan Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.28a, 6.28b, 6.28c and 6.28d represent four distinct simulation experiments. 139
6.29 Manhattan Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.29a, 6.29b, 6.29c and 6.29d represent four distinct simulation experiments. 140
6.30 Manhattan Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.30a, 6.30b, 6.30c and 6.30d represent four distinct simulation experiments. 141
6.31 Manhattan Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.31a, 6.31b, 6.31c and 6.31d represent four distinct simulation experiments. 142
6.32 Manhattan Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.32a, 6.32b, 6.32c and 6.32d represent four distinct simulation experiments. 143

6.33 Downtown Campinas Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. 144
6.34 Downtown Campinas Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. 144
6.35 Downtown Campinas Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.35a, 6.35b, 6.35c and 6.35d represent four distinct simulation experiments. 145
6.36 Downtown Campinas Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.36a, 6.36b, 6.36c and 6.36d represent four distinct simulation experiments. 146
6.37 Downtown Campinas Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.37a, 6.37b, 6.37c and 6.37d represent four distinct simulation experiments. 147
6.38 Downtown Campinas Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.38a, 6.38b, 6.38c and 6.38d represent four distinct simulation experiments. 148
6.39 Downtown Campinas Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.39a, 6.39b, 6.39c and 6.39d represent four distinct simulation experiments. 149
6.40 Downtown Campinas Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.40a, 6.40b, 6.40c and 6.40d represent four distinct simulation experiments. 150

List of Tables

5.1 Action selection in the *Junction East* codelet. In this example, phase number 3 was selected because it gives the best sum of green lanes activations. 100

Contents

1 Introduction **21**

1.1 What is the Motivation Behind Engineering Applications in Urban Traffic Signal Control? . . . 21

1.2 Why Studying Consciousness in Animals? . . . 24

1.3 Why Applying Machine Consciousness to Urban Traffic Control? . . . 26

1.4 Main Contributions . . . 26

1.5 How this Thesis is Organized . . . 27

2 Understanding Urban Traffic Signal Control **28**

2.1 Theory of Urban Traffic Signal Control . . . 28

2.1.1 Mathematical Model of Control in one Isolated Signalized Junction 29

2.1.2 Mathematical Model of an Urban Network of Roads with Signalized Junctions . . . 35

2.1.3 Traffic Simulation Techniques . . . 37

2.2 State of the Art on Traffic Signal Control . . . 43

3 An Emerging Science of Consciousness **47**

3.1 Philosophical Theories of Consciousness . . . 49

3.1.1 René Descartes and the Mind-Body Dualism . . . 49

3.1.2 John Searle and the Biological Argument . . . 50

3.1.3 Daniel Dennet - Denying Consciousness . . . 52

3.1.4 David Chalmers and the Naturalistic Dualism . . . 52

3.2 Scientific Theories of Consciousness . . . 54

3.2.1 Giulio Tononi - Consciousness as Integrated Information . . . 54

3.2.2 Christof Koch, the Romantic Reductionist . . . 55

3.2.3 Jeff Hawkins and the Memory Prediction Framework . . . 55

3.2.4 Bernard J. Baars and The Dynamic Global Workspace Theory . . . 59

3.3 The Global Workspace Algorithm . . . 61

3.4 State of the Art in Applications of the GWT . . . 63

3.4.1 Stan Franklin and the Learning Intelligent Distribution Agent (LIDA) 63

3.4.2 Dehaenne and the Excitatory Neurons Model . . . 66
3.4.3 Shanahan and the Imagination Architecture . . . 66
3.5 Our Definition of Machine Consciousness . . . 66

4 The Cognitive Systems Toolkit 68
4.1 Cognitive Architectures . . . 68
4.2 Cognitive Systems Toolkit . . . 71

5 Materials and Methods 89
5.1 Materials . . . 90
5.1.1 The SUMO Traffic Simulator . . . 90
5.1.2 Network Models - Test Bed . . . 93
5.1.3 Our CST Architecture - The Traffic Controllers . . . 94
5.2 Methods . . . 107
5.2.1 Simulation Scenarios . . . 107
5.2.2 Controlled Experiments . . . 108

6 Results 109
6.1 Simple T Model . . . 109
6.2 Twin T Model . . . 118
6.3 Corridor Model . . . 127
6.4 Manhattan Model . . . 136
6.5 Downtown Campinas Model . . . 140
6.6 Experiments Raw Data . . . 151

7 Conclusion and Future Work 153
7.1 Main Findings . . . 153
7.2 Publications . . . 154
7.3 Limitations and Future work . . . 155
7.4 Conclusion . . . 156

Bibliography 157

"... some anatomists have estimated that the typical human neocortex contains around thirty billion neurons (30,000,000,000) ... Those thirty billions cells are you. They contain almost all your memories, knowledge, skills, and accumulated life experience... I still find this fact astounding. That a thin sheet of cells sees, feels, and creates our worldview is just short of incredible... the dreams we have for a better world are somehow the creation of these cells... the mind is the creation of the cells in the brain. There is nothing else, no magic, no special sauce, only neurons and a dance of information... We need to understand what these thirty billion cells do and how they do it... how it gives rise to the human mind."

Jeff Hawkins, On Intelligence.

Chapter 1

Introduction

1.1 What is the Motivation Behind Engineering Applications in Urban Traffic Signal Control?

On November 2009, IEEE Spectrum Magazine published an article called *Engineering Achievements: The Two Lists* (Lucky, 2009), showing on one side the 20 biggest engineering achievements of the 20th century and on the other side the 14 biggest engineering challenges to be achieved in the 21st century. Both lists were elaborated by the American National Academy of Engineering (NAE)[1]. Figure 1.1 presents both lists.

The choice of the biggest achievements was based mainly on the benefits posed to society. According to the author, Robert Lucky, even though the "old list" has profoundly changed society, it began with simple inventions, in the classic style, made by a small group of scientists, focused in solving small problems, that later represented big benefits in bigger systems, as in a *bottom-up* approach. For instance, the transistor was invented to improve telephone circuits and the Internet was invented to transfer files between Mainframes, in a time when it was believed that the market for computers was composed solely by huge corporations with specific demands.

Lucky says that the "new list" brings changes in this paradigm. The author observes a *top-down* approach, with benefits to society pointed a priori and then being used to motivate engineers to invent systems that solve the demands posed. This approach requires multidisciplinary teams oriented towards innovation.

However, what is the most significant in both lists, more than the paradigm change in the approach, is the cause and effect relation between the two lists. The automobile was not projected to populate huge cities neighborhoods, nor the air conditioning was invented

[1] http://www.engineeringchallenges.org/challenges.aspx

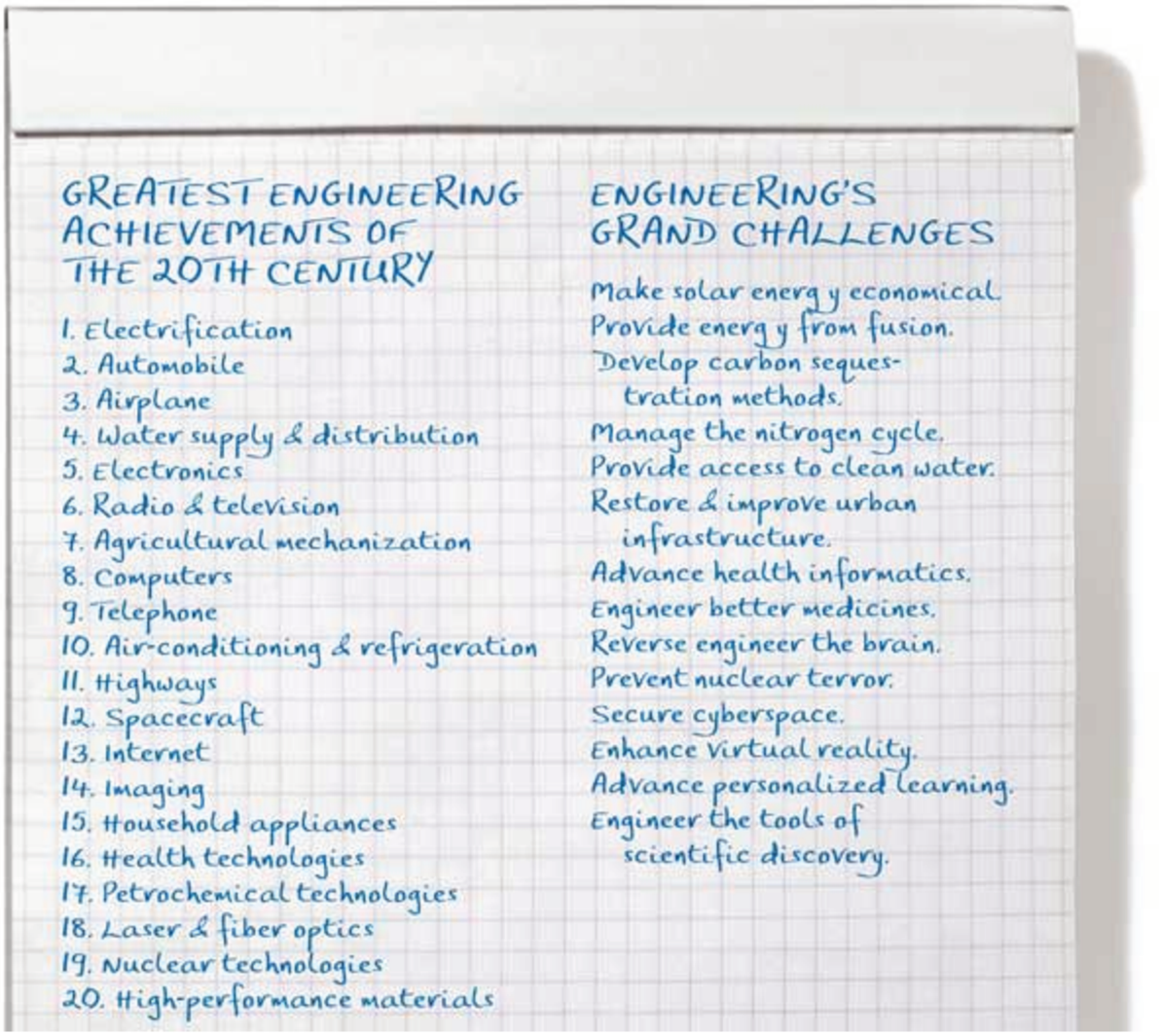


Figure 1.1: Two lists: on one side the 20 biggest engineering achievements of the 20th century and on the other side the 14 biggest engineering challenges to be achieved in the 21st century. Source: National Academy of Engineering; Credits: Steve Wisbauer/Getty Images

to make it possible to have huge cities in hot climate zones. Nevertheless, these inventions collaborated to the rural exodus and the uncontrolled growth in cities populations. The inventions of the 20th century changed society in a more profound way than their inventors could have planned or expected. In the present century, as a matter of fact, we have a challenge list dealing mostly with the consequences of the uncontrolled population growth, mainly in big cities.

From the "new list", the challenge mostly related to this work is "restore and improve urban infrastructure", although we can say that "reverse engineer the brain" also plays an important role. In Brazil, a large rural exodus happened during the second half of the 20th century, giving birth to huge cities like São Paulo, with more than 20 million people in its metropolitan area. According to the architect de Medeiros (2006), Brazilian cities are, from a transportation point of view, the least accessible ones in the world, with urban mobility indexes lower than cities in the Middle East, which in most cases resemble a maze. The current state of our infrastructure is inadequate to deal with the current urban mobility demand: there is a gap in investments since the 1980's, in opposition to the already mentioned population growth and the vehicle fleet growth. Only in the state of São Paulo, the number of vehicles grew from 9.85 millions in 1999 to 19.139 millions in 2009.

Population in Brazilian cities grows faster than the transportation network capacity, in such a way that the public administration can not keep up with transportation requirements in urban areas. As a consequence, inhabitants spend a considerable amount of their daily work time commuting, locked in traffic jams, with unpredictable travel times, leading to personal implications, such as physical and mental health deterioration, social problems, caused by the reduction of public areas inside the city, economic problems, caused by waste of working hours, fuel and accidents, and environmental problems, such as atmospheric pollution and greenhouse effect.

This problem is not a Brazilian privilege. In Asia, for instance, where there is the biggest concentration of cities with more than 10 million people in the world, rural exodus is still in course and the same kind of issues arise, only in this case in much bigger versions.

Figure 1.2: Traffic jam in São Paulo, the biggest Brazilian city.

One approach to reduce this problem is the use of adaptive traffic light controllers, able to change its control policy based on local information. Even though they are not able to completely solve traffic problems, they produce significant improvements without the need to change current infrastructure or transportation models (Sánchez-Medina et al., 2010).

1.2 Why Studying Consciousness in Animals?

In vertebrate mammals, consciousness is a dynamic, integrated, multimodal mental process (Fabbro et al., 2015). The scientific hypothesis for neural correlates sufficient for this process is that they were naturally selected during animal evolution because they permitted animals to plan for future events and deal with unexpected situations they had never experienced before, in a complex and ever changing environment (Crick & Koch, 2003; Baars & Franklin, 2009; Edelman et al., 2011; Baars et al., 2013). The main advantages brought by such a mechanism are twofold. First, the serial stream of consciousness provides an executive summary of perceptions. From all perceptual information at a given moment being processed unconsciously (and in parallel), the most relevant information

Figure 1.3: The scientific hypothesis is that consciousness brought competitive advantages to animals which permitted them to deal with unexpected situations and somehow focus attention in solving the most critical problem.

for the animal survival becomes conscious, generating a unique and integrated content, conveying the necessary information for the animal to better deal with unexpected and novel situations that differ from its common habits. Second, the attentional characteristic of consciousness provides an adequate framework for behavioral learning, with the processes of automation and deautomation of behaviors. During the automation process, as novel situations become more and more frequent, conscious content is stored in long term memory, becoming accessible for planning and making predictions. Once the process of behavioral automation is concluded, action selection can happen without conscious interference. However, if an automated behaviour produces unexpected results, consciousness regains control of action selection and information processing, which might result in the deautomation of this previous behavior, and its consequent suppression. This mechanism allows the creation and suppression of habits in animals, making their behavior extremely adaptive to changing environments, what constitutes a great competitive advantage for survival.

1.3 Why Applying Machine Consciousness to Urban Traffic Control?

Control and optimization of traffic lights phases is a key topic in improving cities traffic conditions (Brockfeld et al., 2001; Srinivasan et al., 2006; Sánchez-Medina et al., 2010). For large urban networks, though, there is a prohibitive number of variables, states, stochastic aspects, uncertainty, interactions between subsystems, mutually exclusive goals, and other issues (Guberinic et al., 2008), which make scenarios like these nearly impossible to be solved with conventional strategies. In other words, traffic signal control systems are large complex nonlinear stochastic systems. Therefore, it is hard to find optimal traffic signal settings (Zhao et al., 2012).

However, traffic lights control in an urban network can be seen as a set of subsystems operating in parallel, where each subsystem is a single junction composed of n traffic lights influencing and being influenced by its neighbor subsystems. In most cases, each subsystem is operated in isolation. However, for the network to function properly, it would be interesting to have these subsystems interacting in a way that critical situations might be avoided, such as in gridlocks and big traffic jams. This scenario is similar to what is found in the animal body, where different isolated subsystems are coordinated by an executive control nervous system, relying in both unconscious and conscious processes in order to generate its overall behavior. In this central executive mechanism, consciousness can be viewed as a supervisor process that takes care of many semi-autonomous subsystems.

The scientific hypothesis of this work is that an artificial mechanism, inspired on some properties and models of consciousness, can bring advantages to automatic processes, such as urban traffic lights control.

1.4 Main Contributions

The main original contribution of this work is the application of a machine consciousness approach to urban traffic control, with the design and implementation of a solution

proposal to the problem.

The application of a cognitive architecture with machine consciousness capabilities to the control of a urban traffic network at the same time brings an enhancement to traffic control technologies and advances to the understanding of the consciousness phenomenon and to the possibility of its simulation (or emulation) in artificial creatures.

Moreover, during the time of this work, we were part of the team which designed and implemented the foundations of the Cognitive Systems Toolkit, a toolkit to build cognitive architectures, presented in further details in this publication, which will be used in this work to build the cognitive architecture that will be controlling the traffic lights with the machine consciousness capability.

1.5 How this Thesis is Organized

The remainder of this thesis is organized as follows. In Chapter 2, we explain the basics of the theory on urban traffic signal control and present its current state of the art. In Chapter 3, we talk about the main theories of biological and machine consciousness, including some background about Global Workspace Theory, the consciousness model we will be relying on, in order to develop our machine consciousness approach. In Chapter 4, we show the CST - the Cognitive System Toolkit - which is being developed by our research group, and is used as the main basis for the construction of the cognitive architecture which will be controlling our traffic lights. In Chapter 5, we present the materials and methods for our experiments, describing the traffic simulation tool we used and some details about how it models urban traffic networks. In this Chapter, we also explain the cognitive architecture controlling the traffic lights by using Global Workspace Theory. In Chapter 6, we present the main results we obtained with our simulations, and in Chapter 7 we provide a discussion for these results and the main conclusions.

Chapter 2

Understanding Urban Traffic Signal Control

In order to be able to test our hypothesis, which means designing and implementing a machine consciousness mechanism applied to the control of urban traffic lights, the first step is capturing the basics of the theory on urban traffic signal control and its current state of the art. We do so in the following way: first, in section 2.1, we introduce the theory of urban traffic signal control. In subsection 2.1.1, we present the mathematical model of controlling traffic lights in an isolated junction. In subsection 2.1.2, we present a model of how isolated junctions can be connected by a network of roads in order to constitute a more complex scenario. In subsection 2.1.3, we talk about the main traffic simulation techniques, which can be used in our experiments to transform the output vehicles flow from a junction in input flows in the next junctions, over time, considering the mathematical models presented before. Finally, in section 2.2, we present the state of the art on traffic signal control, so we can do our best to stand on the successful previous works on the matter. Indeed, one of these works will be chosen to be used as the main heuristic of our controller.

2.1 Theory of Urban Traffic Signal Control

Traffic in an urban network can be modeled as a network of roads crossing in particular regions which are called "junctions", together with a number of vehicles (and possibly pedestrians) flowing into it. Junctions are mutually exclusive regions, because they support a limited number of vehicles in a given time, what requires a discipline to accommodate the many possible flows competing to use the junction region. Usually, competing flows must be interrupted for some time, allowing other flows to have access to

the junction, being this interruption resumed later such that all flows are able to share a common resource. Each junction can be signalized (that is, controlled by traffic lights) or not. Non-signalized junctions usually follow some prescribed set of rules, which discipline which vehicle will get preference at a given time, in order to use the junction space. In a particular case, a whole network might operate only with non-signalized junctions. This might be an efficient way of controlling the traffic flows when the traffic is not intense. Nevertheless, as soon as traffic increases in the network, some sections of a road might have their traffic time reduced, in such a way that it becomes very inefficient, and the probability of accidents in competing flows builds up. In these cases, usually the solution is to start installing traffic lights in some specific junctions, such that a better discipline might be in operation. In a typical scenario in large cities, there is a significant number of junctions which are signalized, but there are also many junctions which are not. Those junctions requiring the installation of lights are dependent on the traffic patterns typically present on the urban network and evolve together with the network as traffic patterns evolve in time.

A junction can be either isolated, with negligible influence on others, or can be sufficiently close to others in order to influence their performance. Hence, junction control must be formulated separately for different cases, such as an isolated junction, a sequence of signalized junctions on a road or a network of roads with a significant number of signalized junctions influencing one another.

In this work, the latter case is the main objective of treatment. However, in each one of these cases, it is important to maintain the feasibility of the chosen phases to control each junction. Consequently, we will start by formulating the control of an isolated junction.

2.1.1 Mathematical Model of Control in one Isolated Signalized Junction

For each junction, we define a tuple (U,Σ,W,Y,O_1,O_2), where:

- U is the set of light control functions $u(t) = \{u_1(t), u_2(t), ..., u_p(t), ..., u_P(t)\}$, where P is the number of independent lights in the junction and $u_p(t)$ is a function which

can assume values 0 or 1, being 0 considered as effective red and 1 as effective green[1] controlling a specific light p. Usually, $u_p(t) = u_p(t + kc)$, where $k = 0, \pm1, \pm2, ...$ and c is a constant called *cycle time*, which means $u_p(t)$ is a periodic function. This condition might be violated if an adaptive control law is applied.

- Σ is the set of input arrival flows of vehicles (or pedestrians) described by functions $\sigma(t) = \{\sigma_1(t), \sigma_2(t), ..., \sigma_i(t), ..., \sigma_I(t)\}$, where I is the number of input flows. These arrival flows are usually modeled using Poisson distributions.

- W is the set of queue states $w(t) = \{w_1(t), w_2(t), ..., w_i(t), ..., w_I(t)\}$, whose components represent the size of the vehicles queues formed as a consequence of the arrival flows.

- Y is the set of output flows in the junction $y(t) = \{y_1(t), y_2(t), ..., y_h(t), ..., y_H(t)\}$, where H is the number of output flows in the junction.

- O_1 - state transformation function - $W_t \rightarrow W_{t+1}$

- O_2 - output function - $W_t \rightarrow Y_t$

Basic Relations in Vehicle Flow Sets

Consider Figure 2.1 as an example of the many flows possibly crossing a junction.

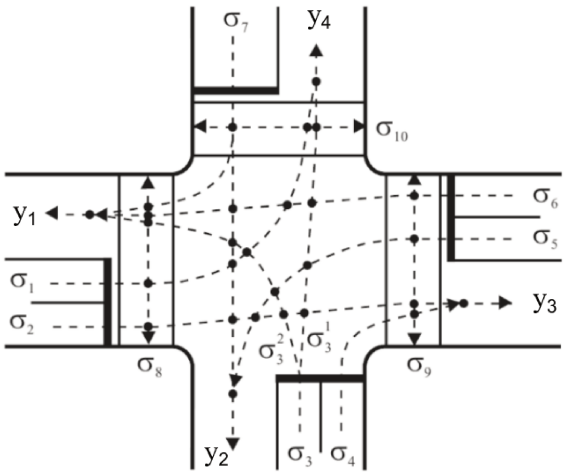


Figure 2.1: Junction illustrating vehicle flow relations (Guberinic et al., 2008)

[1]Yellow sign is incorporated in the effective red and green

The flows must be compared to each other in order to obtain the constraints on conflicting flows. The main relations, depicted in Figure 2.1 are:

- Conflict - trajectories cross or merge (σ_1 and σ_3).
- Non-conflict - trajectories do not cross or merge (σ_1 and σ_2).
- Compatibility - conflicting flows can go simultaneously, being resolved by the drivers themselves. This solution can be used in a very intense traffic situation, despite the risk of incidents, which is a disadvantage (σ_3 and σ_7).

Junctions or Signal Groups

If two flows n and m are in conflict, necessarily this means that while one of them has $u_n(t) = 1$ (green light), the other one must have $u_m(t) = 0$ (red light). This means that the number of necessary independent lights P in a junction is calculated based on how the input flows i are connected to the output flows h. For each possible conflict, there must be a pair of lights with opposite controls. Depending on the situation, flows in a relation of compatibility might also require a pair of lights to discipline them.

When one or more input-output flows are controlled by the same traffic light $u_p(t)$, they form a junction (or signal) group.

Control Variables

A typical example for a particular junction group p is shown in Figure 2.2. In this example, the following terms apply:

- $u_p(t)$ - 0 for effective red and 1 for effective green;
- c - cycle time;
- t'_p - time in which an effective green light starts for an specific junction group p, in relation to the beginning of the cycle;
- t''_p - time in which the effective green light ends for an specific junction group p, in relation to the beginning of the cycle;

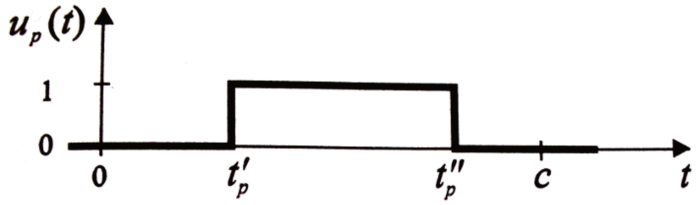


Figure 2.2: Control variables (Guberinic et al., 2008)

Signal Plan

Vector $u(t)$ is called a Signal Plan, representing the traffic control in a signalized junction in a cycle with duration c.

Control Vectors - Phases

An alternative representation for $u(t)$ considers that, in any signal plan, there are intervals in which all $u_p(t)$ remain the same, assuming either a value of 0 or 1 during all the interval. So, we can name such interval a *phase* and use the duration τ of each phase in order to represent the full signal plan $u(t)$. Hence, the plan can be represented as a control vector containing each phase f and its duration τ:

$$u(t) = [(f_1, \tau_1), (f_2, \tau_2), ..., (f_k, \tau_k), ..., (f_K, \tau_K)], \tag{2.1}$$

in which $f_k = (s_1, s_2, ..., s_p, ..., s_P)$, $s_p \in \{0, 1\}$ and K is the number of phases in a cycle, as shown in Figure 2.3, where e.g. $f_1 = (0, 1, 0, 1)$, $P = 4$ and $K = 9$.

Even though a completely adaptive set of phases shall be envisioned, usually a signal plan maintains a fixed set of phases. One of the reasons for that is to avoid drivers in presuming that there is a malfunctioning in the controller. In a completely fixed controller, the set of phase durations $\{\tau_k\}$ is also fixed. An adaptive controller might change the durations τ_k of each phase. This might happen maintaining a fixed overall cycle period time

$$c = \sum_{k=1}^{K} \tau_k$$

or even changing the duration of a cycle. The set of phases is usually called a "signal plan structure". A signal plan structure must be *feasible* as will be defined later.

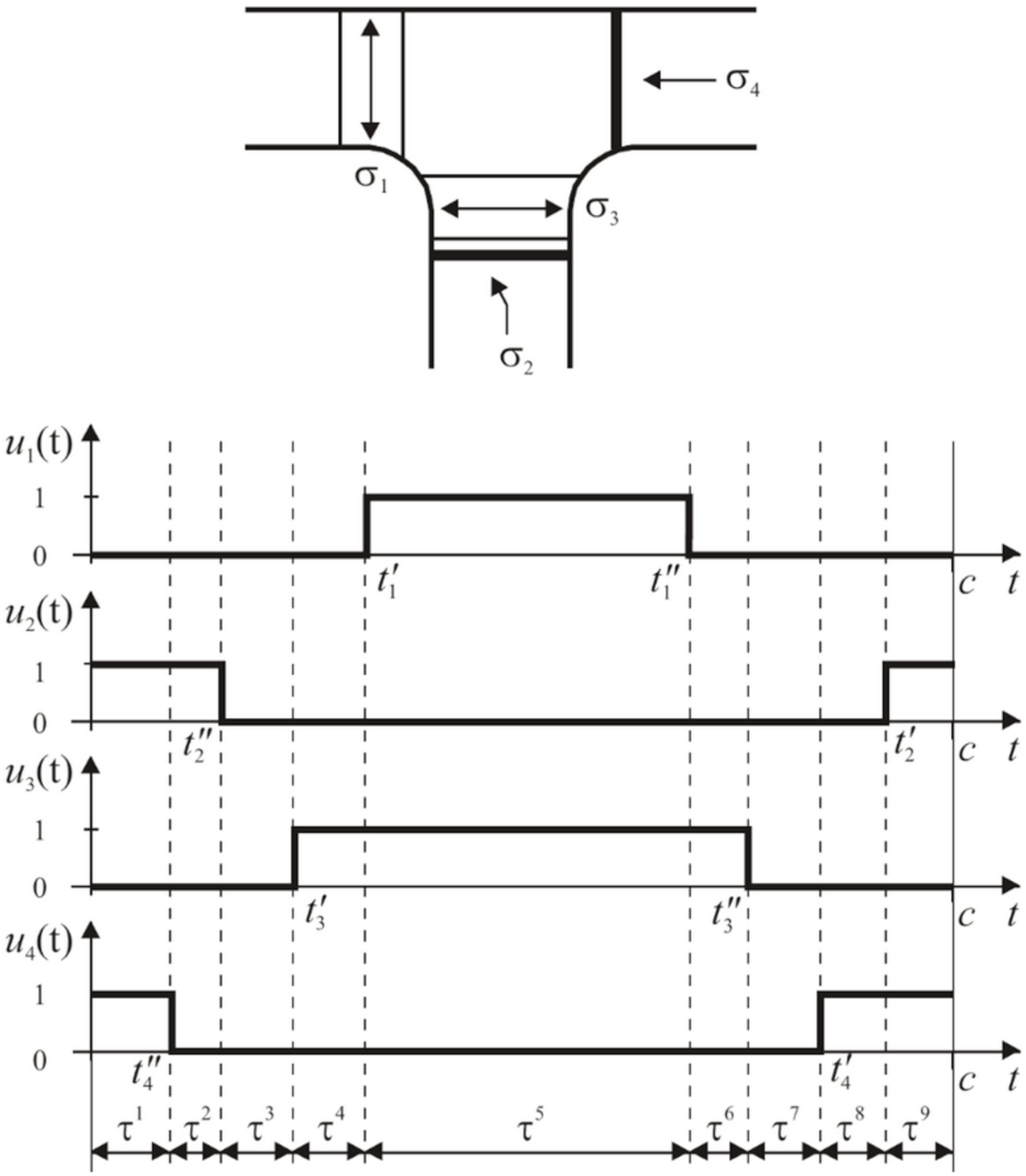


Figure 2.3: Signal control phases (Guberinic et al., 2008)

Output Function - Vehicle Output Flow

Arrival flows σ_i are transformed by the signals, and the output flows y_h are formed by linear combinations of one or more arrival flows transformed. Output flows are of no importance to control isolated junctions, but in the case of groups of junctions in a network, its transformation when traveling between junctions is the most relevant function in the solution model.

The Feasible Control Set

The set of feasible signal control plans is defined by the following constraints (Guberinic et al., 2008):

1. Minimum of one green interval in a cycle for each junction;
2. Minimum green time;
3. Maximum red time;
4. Capacity restriction;
5. Simultaneous green lights for compatible groups only;
6. Minimum time between green lights;
7. Phases sequence restriction;
8. Sum of phases duration should be equal to the cycle total time;
9. Maximum cycle time.

Constraints 1, 3 and 9 can be relaxed in extreme situations, but not the others, because of safety reasons.

General Formulation of the Isolated Junction Control Problem

A general formulation for the isolated junction control problem can be stated like that:

"Determine the phases duration τ_k in the control $u(t)$ (signal plan), according to a feasible set of phases f_k, such that some sort of optimization criteria is employed."

2.1.2 Mathematical Model of an Urban Network of Roads with Signalized Junctions

The control of signalized junctions in an urban network can be considered as a generalization of the isolated junction control. Instead of using an optimization criteria for just a single junction, some sort of optimization criteria for the whole network shall be employed. In order to do so, it is necessary to consider the network topology, and the interactions between the output vehicles flow from a junction and the input flows of its neighbor junctions, what implies in modeling the network of roads as part of the system.

An urban network of roads located in a given geographic region R can be modeled as a graph $G_R(t) = (O, E, W_e, M)$, where O is a set of vertices representing the junctions, E is a set of arcs related to the roads, W_e is a tuple of weight sets related to the arcs E and M is a set of graphs corresponding to the detailed information of each vertex, elements of the set O (see Figure 2.5b for an example of such graphs).

We define that:

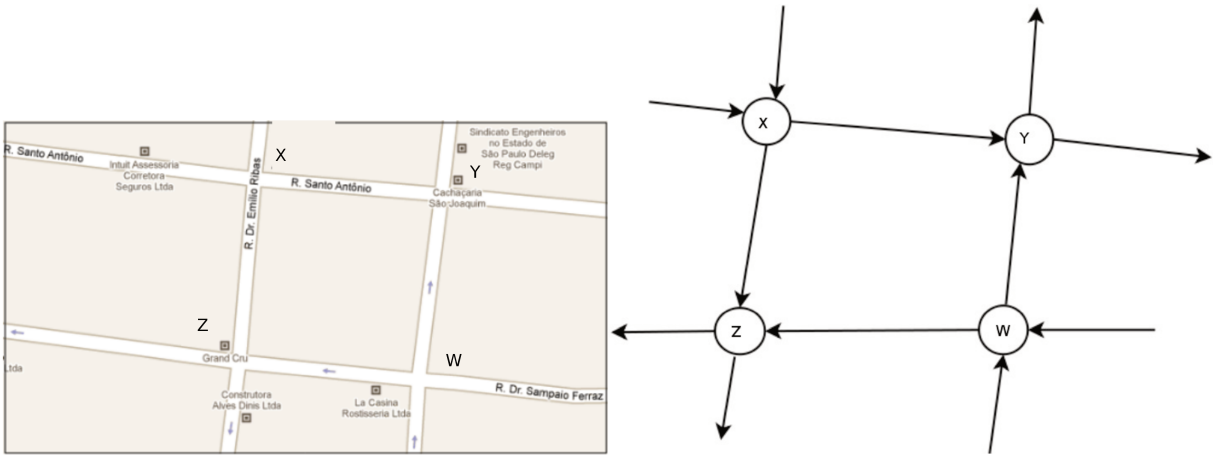
- For each junction, there will be a corresponding vertex $o \in O$ that represents a junction, which can be one among three types:
  - X, Y, L and T[2] (Zanin, 2004) junction types;
  - traffic signals;
  - roads narrowing or enlarging.
- Each arc $e \in E$ is so that $e = (r, z)$, where $r, z \in O$;
- For each pair of junctions $r, z \in O$, there will be a road segment $e = (r, z) \in E$, if and only if there is a direct path between junctions r and z, where r is the origin and z is the destination, without intermediary junctions.
- W_e will be a tuple of weight sets $(D, V, \Re, \Phi, L, S, J, F)$, all referring to the set E, with cardinality $|E|$. Each element of each set of the tuple W_e is related to an

[2] X, Y, L and T here do not refer to the variables of the mathematical model, but instead to actual junction types, as these letters are used to represent junction types iconically similar to the letters' formats, as in Zanin (2004).

element of the set E. Each set is defined as follows:

- D is the set of distances d, in which d refers to the euclidean distance between intersections r and z related to a road segment $e \in E$.
- V is the set of average velocities v of the vehicles traveling in a road segment $e \in E$ in a given time t.
- $\Re$ is the set of vehicle densities ρ related to the road segments $e \in E$ in a given time t.
- Φ is the set of vehicle flows ϕ related to the arcs $e \in E$ in a given time t.
- L is the set of number of lanes l existing in the road segments $e \in E$ in a given time t.
- S is the set of maximum velocities s defined for the road segments $e \in E$ in a given time t.
- J is the set of maximum densities w defined for the segments $e \in E$ in a given time t.
- F is the set of maximum flows f defined for the segments $e \in E$ in a given time t.

- For each vertex $o \in O$, there is a related graph $N_z \in M$, being $N_z = (A, B, C)$, that defines the junction characteristics, with A being the set of vertices that represent the end of each lane, B being the set of arcs which represent all possible transitions between each lane and C the tuple $(D', V', \Re', \Phi', S', J', F')$ of weight sets of the arcs B. Hence, we have:
  - For each lane, the point immediately before the junction is represented by the vertex $a \in A$.
  - Whenever it is possible to transition from lane a_i to lane a_j, considering $i \neq j$, there will be an arc $b = (a_i, a_j)$ whose direction is from a_i to a_j, considering $b \in B$.



(a) (b)

Figure 2.4: Urban roads representation. Panel 2.4a shows the map and Panel 2.4b shows the corresponding graph.

– The sets $D', V', \Re', \Phi', S', J'$ and F' of the tuple C with cardinality $|B|$, each one of their elements corresponds to an element of B and their semantics is equivalent to the one of the sets $D, V, \Re, \Phi, S, J$ and F of the tuple W_e, respectively.

For instance, take the graph G_R, with the set of vertices O and the set of arcs E, shown in Figures 2.4a and 2.4b. The case of the graph N_Z, whose vertices set is A and the arcs set is B, is shown in Figures 2.5a and 2.5b.

2.1.3 Traffic Simulation Techniques

In order to consider the interactions between different junctions in the network topology, we need to be able to transform the output vehicles flow from a junction into an input flow in the next junction, over time. This can be achieved using traffic simulation techniques.

From a traffic flow perspective, it is possible to classify traffic simulation techniques in three different distinct types: microscopic, mesoscopic and macroscopic. The differences between these models are related to the abstraction levels in which vehicle movement is modeled and if the set of vehicles is treated as an aggregated mass or not.

- microscopic approach: vehicle movement is traced explicitly and they are repre-

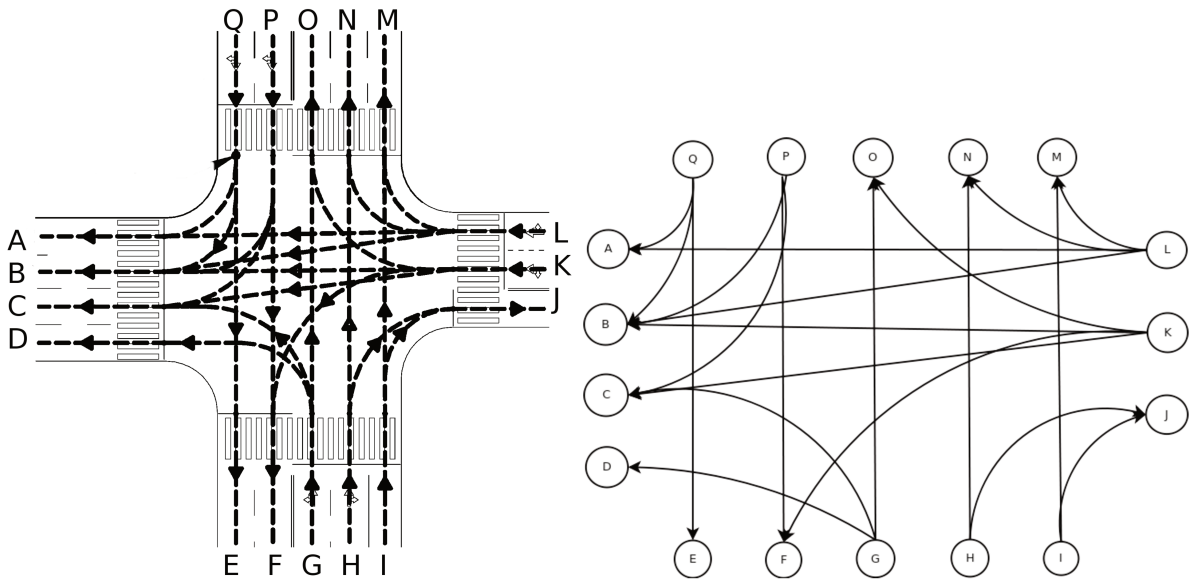


Figure 2.5: Detailed junction. Panel 2.5a, adapted from Maroto et al. (2006), shows possible conversions, while Panel 2.5b shows the corresponding graph.

sented individually. Each vehicle is considered as an agent and there is an equation describing each one of them.

- macroscopic approach: vehicle movement is traced implicitly and they are treated as an aggregated mass. Similar to fluid dynamics (Richards, 1959), it is necessary to determine the following variables: *flux* $q(x,t)$, the number of vehicles flowing in distance x during time t; *space mean speed* $v(x,t)$, corresponding to the instant average velocity in a distance x and time t and *density* $k(x,t)$, corresponding to the number of vehicles is a space unit.

- mesoscopic approach: vehicle movement is traced explicitly and they are represented as an aggregated mass. A probability function *f(t,x,v)* is defined, which expresses the probability of one vehicle existing in time t, in position x and velocity v. One way of obtaining this function is resolving *Boltzmann* equations (Kutz, 2004).

Macroscopic and microscopic approaches are the most common ones used, with clear advantages and disadvantages, depending on the application.

The macroscopic approach has the following advantages:

- Less inputs necessary to model a large scale networks, such as a whole city;

- Less requirements related to map details;
- Less computational time to calculate outputs;
- Model calibration is easier.

The *LWR* macroscopic model described by Lighthill & Whitham (1955); Richards (1959) was one of the first models to make it possible to simulate traffic in environments with few map details (Kutz, 2004).

However, macroscopic approaches are incapable of modeling events such as conflicts between cars, traffic light effects, among others, which are sometimes necessary. Microscopic approaches are capable of modeling these microscopic events, requiring on the other hand more computational power, that might become prohibitive, depending on the size of the network, when all inputs are given.

In the domain of microscopic approaches, the use of cellular automata (Nagel & Schreckenberg, 1992) makes it possible to reconstruct traffic situations by modeling the interaction between running vehicles. Volume data and route distributions are sufficient to guarantee satisfactory results, even when there is just small real time traffic data (Wahle et al., 2002).

For instance, consider the following cellular automata, adapted from Knospe et al. (2002), which represents a microscopic description of the vehicles movement using a set of rules for each step.

The microscopic model structures, represented in Figure 2.6, are the following:

- Node: represents an intersection between roads;
- Arc: a road connecting to nodes.
- Lane: road segment where vehicles flow.
- Cell: discrete parts of one lane that are occupied by vehicles.

The urban network graph described in section 2.1.2 is transformed into an automata structure mapping vertices into nodes, edges into arcs and spliting the arcs into cells.

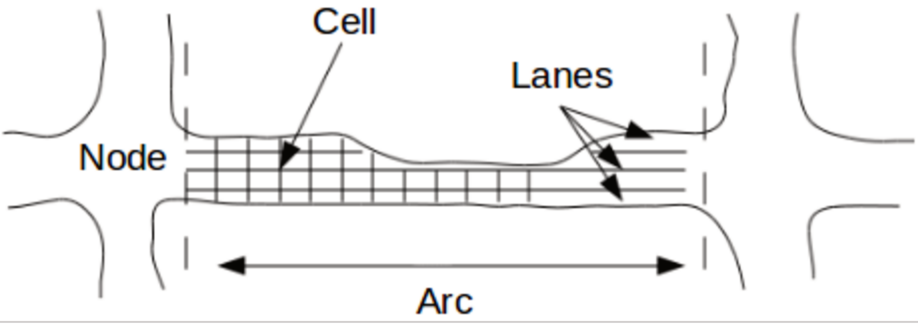


Figure 2.6: Microscopic model structures

Vehicles move advancing through the cells, changing lanes and arcs. In this model, cells have an extension of 7.5 m, which is the space occupied by an average commercial vehicle. Therefore, each cell can be occupied or free at any time. One cell may contain obstacles, such as traffic lights, which might block vehicle flows. A bigger vehicle, like a truck, might occupy more than one cell. Road maintenance can be represented by special vehicles occupying many cells with velocity equals to zero.

A sophisticated behavior can be achieved with one simple strategy combining three simple rules, as represented in Figure 2.7 (Knospe et al., 2002):

1. Vehicles accelerate until maximum velocity, in the long run. Stopped vehicles have a smaller acceleration. The velocity of the first car in the arc is anticipated, permitting smaller gaps and greater velocities.

2. Vehicles anticipate the need to break by watching next vehicle's brake lights;

3. Vehicles always adjust velocity in order to maintain a security distance.

Vehicle movement is realized through the following variables, parameters and rules, which are applied in parallel for each vehicle:

- Variables:
  - x - position
  - v - velocity
  - $vanti$ - anticipated velocity

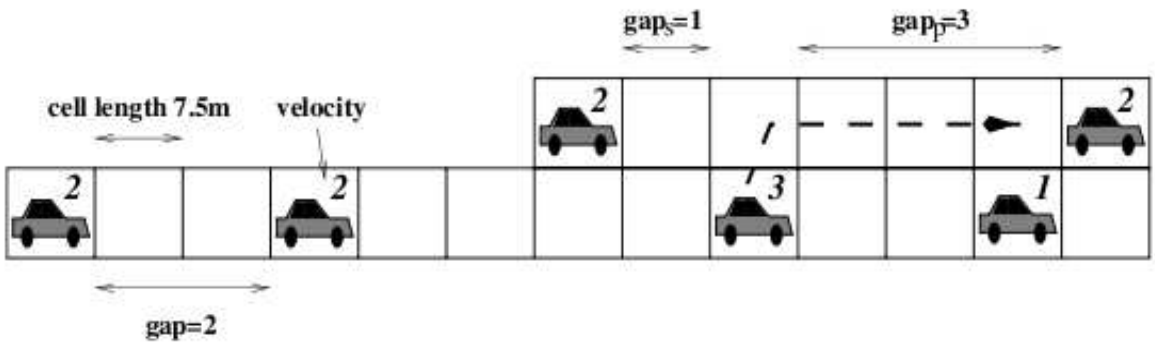


Figure 2.7: Road segment in the microscopic model. Arcs are divided in 7.5 m wide cells. Each car has a discrete velocity between 0, ..., $Vmax$. Vehicles represent ideal drivers, who follow security rules, never accelerating more than the gap to the next vehicle.

  - d - gap distance
  - $deff$ - effective gap distance
  - b - brake light status
  - p - deceleration probability

- Parameters:
  - $gapsafety$ - effectiveness of anticipation control
  - $vmax$ - maximum velocity
  - pb - deceleration probability
  - $p0$ - deceleration probability
  - pd - deceleration probability
  - th - gap time
  - ts - security gap time

1. Acceleration rule
   - $b_n\ (t+1) = 0$
   - if $((b_n + 1\ (t) = 0)$ and $(b_n\ (t) = 0))$ or $(t_h >= t_s\ )$ then: $v_n\ (t+1) = \min(v_n\ (t) + 1, vmax\ )$.
2. Deceleration rule (to avoid accidents)

- $v_n\ (t+1) = \min(deff_n\ ,\ v_n\ (\text{t+1}))$
- if $(v_n\ (t+1) < v_n\ (t))$ then: $b_n\ (t+1) = 1$.

3. Stochastic deceleration rule

- if $(\text{rand}() < p)$ then:
- $v_n\ (t+1) = \max(v_n\ (t+1)$ - 1, 0)
- if $(p = pb\ )$ then: $b_n\ (t+1) = 1$.

4. Movement rule

- $x_n\ (t+1) = x_n\ (t) + v_n\ (t+1)$.

Where:

- $deff_{pred} = d_{pred} + \max(vanti$ - $gapsafety\ ,\ 0)$
- $t_h = d_n\ / v_n\ (t)$
- $t_s = \min(v_n\ (t),\ h)$
- $p = p(v_n\ (t),\ b_n{+}1\ (t),\ t_h\ ,\ t_s\ ) =$
  - pb, if $b_n + 1 = 1$ and $t_h < t_s$
  - po, if $v_n = 0$ and not $(b_n + 1 = 1$ and $t_h < t_s\ )$
  - pd, elsewhere

According to Knospe et al. (2002), these parameters give a realistic result:

- $vmax = 2$
- $pd = 0.1$
- $pb = 0.94$
- $po = 0.5$
- $h = 6$

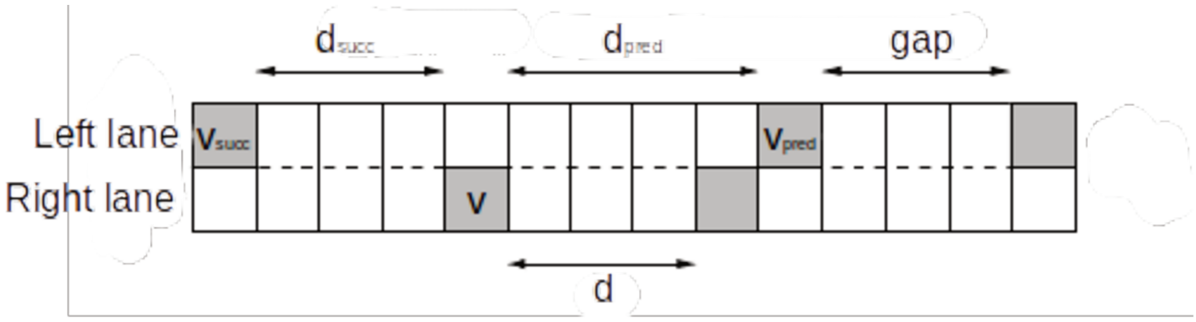


Figure 2.8: Arc scheme and quantities related to lane changing rules. Colored cells are occupied by vehicles.

- $gapsafety = 1$

Besides the movement rules presented above, the cellular automata also considers lane changing rules. Actually, the system update is done in two steps, being the first step the application of lane changing rules and the second the movement rules, applied in parallel for each vehicle.

Figure 2.8 depicts quantities involved in lane changing rules. In order to change lanes, the following rules are considered :

- Initiative rule
  - $(b_n = 0)$ and $(v(t) > d)$
- Security rule
  - $(deff_{pred} >= v(t))$ and $(d_{succ} >= v_{succ})$

In this cellular automata, one step may correspond to 1 second, 0.1 second, and so on. Using a maximum velocity of two cells in each step, we reach 54 km/h, which is close to the limit of 60 km/h that is found in many urban roads in Brazil.

2.2 State of the Art on Traffic Signal Control

The first applications of traffic signal control began using fixed-time control methods, with a predetermined cycle and split time plan (green duration as a portion of the cycle time), which were convenient for relatively stable and regular traffic flows. These signal

control systems assume a periodic operation, in which each light goes through its sequence of phases with calculated split parameters in a cycle that is offset from its neighbors. A fixed-time plan can be built using offline optimization tools, such as TRANSYT (Robertson, 1969), SYNCHRO (Husch et al., 2003) and VISGAOST (Stevanovic, 2007), based on historical traffic flow data, for specific time periods.

Later on, real-time traffic-responsive control came into practice with the help of sensing technologies. The idea is that any traffic control action is made under a certain control strategy according to real-time traffic data. The cycle length might be dynamically adjusted to meet the traffic demand. The SCAT system (Sims & Dobinson, 1980) dynamically adjusts common cycle length of a given traffic network (or sub-network) to meet the traffic demand. The SCOOT method (Robertson & Bretherton, 1991) and the ACS-Lite (Luyanda et al., 2003) are examples which rely on the fixed-time offline calculation as a baseline or contingency plan. In some traffic control systems, each intersection decides which phase to apply in order to enforce safety and other constraints, rather than being oriented towards a parametric timing plans (Mirchandani & Head, 2001; Papageorgiou et al., 2003).

Finally, one major latter attempt was the introduction of computational intelligence, trying to simulate the intelligence of nature to some extent by the usage of certain computational methods, which include artificial neural networks, fuzzy systems, and evolutionary computation algorithms (Zhao et al., 2012).

Some approaches consider the problem only locally, restricted only to a small number of traffic lights (Sik et al., 1999). Some reactive methods can achieve good performance for isolated intersections, making decisions quickly based on traffic flow, like interval between vehicles or anticipated queues of vehicles, but these methods are susceptible to suboptimal decisions (Viti & van Zuylen, 2010; Lämmer & Helbing, 2008). However, when dealing with the whole urban network, because of the non linear and stochastic events which happen in the network and their inter-dependencies, the actual state of traffic becomes hard to assess and the effects of changes in traffic control becomes almost impossible to forecast (Srinivasan et al., 2006). Hence, how to achieve scalable network-

wide optimization remains a challenging problem. Some methods try to produce more optimal solutions considering a time horizon, which is divided into discrete intervals based on a fixed time resolution, forming a state space which is searched via an optimization process. Some examples include PRODYN (Henry et al., 1983), COP (Sen & Head, 1997), ALLONS-D (Porche & Lafortune, 1999), OPAC (Gartner et al., 2002), ADPAS (Kim et al., 2005) and CRONOS (Boillot et al., 2006). Network-wide control systems, such as RHODES (Mirchandani & Head, 2001) and RT-TRACS (Gartner et al., 2002), either apply additional signal control guidance from neighboring intersections that incorporates non-local impact or extend the prediction horizon with flow information from neighboring intersections. The SchIC method (Xie et al., 2012) can achieve search space reduction and state elimination by exploiting structural flow information in the prediction horizon and the intersection control problem is formulated as a scheduling problem, based on an aggregate representation on flow data. Recent works investigated different approaches to this problem, such as dynamic programming (Heung et al., 2005; Heydecker et al., 2007), neuro-fuzzy networks (Choy et al., 2003) and reinforcement learning (Cai et al., 2009). Nakamiti (1996) developed a distributed control traffic system using a distributed computational intelligence approach. In this approach, agents have to interact with one another seeking for cooperation, despite their incomplete, uncertain or even inconsistent knowledge, using a symbiosis among distributed artificial intelligence, fuzzy sets theory, case-based systems and genetic algorithms. Nakamiti's distributed traffic control system was applied to a central region of the city of Campinas, Brazil, concerning six junctions with eighteen traffic lights. In simulations using real data, the distributed traffic controller performed better than traditional fixed-time techniques, showing delay times 43% lower and cars queues 24% smaller than the traditional fixed-time techniques optimized for critical traffic scenarios.

Zhao et al. (2012) surveyed some commonly used computational intelligence (CI) paradigms, analyzed their applications in traffic signal control (TSC) systems for urban surface-way and freeway networks, and introduced current and potential issues of control and management of recurrent and non recurrent congestion in traffic networks.

They showed fuzzy controllers applied to single junctions that can bring up to 13% of improvement, and also fuzzy controllers for multiple junctions and lanes that brought 25% performance improvement. Applications of artificial neural networks were shown to produce optimal instantaneous signal timing, while automatically adapting to long term changes. Even in rush-hour conditions, improvements were still from 14.79% to 18.11% in average delay time and from 11.79% to 14.21% in average travel time, compared with the isolated fixed-time control method. Genetic algorithms and Swarm Intelligence approaches showed better real-time overall performances, effectively alleviating urban traffic pressures and reducing waiting time of vehicles. The authors conclude that CI methodologies and technologies are effective solutions for TSC problems. According to them, there is no criterion to determine which technology is more suitable or how to apply these methodologies in the field of TSC. They also believe that some specific problems, such as "blue corridors" for emergency vehicles, will find new ways to be solved. The researchers state that more research work is needed to build the basis for the area and finally poses that CI technology will play an active role in future intelligent traffic systems development.

Box & Waterson (2013) developed one traffic light controller which learns strategies based on previous experience. They used human experts to control a single microscopic traffic simulation of an area in Southampton's urban road network. The researchers used the human experts' decisions to train a neural network, which was later used to control the simulation and achieved better results than earlier applied algorithms and benchmarks. In Chapter 5, you will notice that we stood on their work to choose the main heuristic used in the parallel reactive part of our controller.

Chapter 3

An Emerging Science of Consciousness

After having captured the basics of the theory on urban traffic signal control and its current state of the art in Chapter 2, it is time to apply the same method to the theories of consciousness, as we intend to take advantage of the growing scientific knowledge about consciousness processes in human mind, in order to be able to apply some of these concepts in the design of our machine consciousness controller.

Consciousness is the fundamental problem of the human mind. Even though some of the ideas of the philosopher René Descartes about the mind-body duality are rejected by the mainstream community nowadays, one of his statements seems to be unanimous among cognitive scientists, as it has captured so well and briefly the matter: "Cogito, ergo sum"[1]. As all the experience of life in the physical world is accomplished through a serial mainstream of thoughts known as "Consciousness", the only way of being sure of one's existence is realizing one is able to think, that is, to somehow produce this train of thoughts. The most difficult aspect of consciousness is the hard problem called qualia (Chalmers, 1996). Qualia refers to the redness of red, the painfulness of pain, and so on. How is it possible that physical processes going on inside the brain could give rise to the sujective richness of consciousness? No one has produced any plausible explanation as to how the subjective experience of the redness of red could arise from the objective actions of the brain, not yet. However, we know that consciousness is part of the natural world, and therefore must depend only on the imperfectly known laws of physics, chemistry, and biology, which means it does not arise from some magical or otherworldly quality.

Despite the importance of this subject, the study of consciousness remained censored in the scientific community during a long time. The main reason for this was a debate on how to approach a subjective phenomenon in an objective way, that is, considering the

[1] "I think, therefore I am".

scientific method. It was not until the second half of the 20th century that more works on the subject started to flourish, and more recently with the participation of famous scientists, such as Crick & Koch (2003), the field started to take off and became more respected to the point that it is today recognized that, in order to build a general-purpose, real-life computational equivalent of a human mind (Samsonovich, 2012c), the problem of consciousness still have to be solved.

As engineers, we do not want to focus in the philosophical issues of consciousness, but in the practical ones. As with most of the biological phenomena, Engineering has a lot to benefit from the growing knowledge about consciousness processes in human mind. Engineers have perceived since long ago that they can take advantage of the solutions nature has developed for many hard problems, through the iterations of natural selection along millions of years of life evolution on planet Earth, and use the power of analogy to build new infrastructure and frameworks that are powerful and can tackle real world problems (Castro, 2006). In the case of consciousness, as argued in Chapter 1, it brings great competitive advantage for animal survival by giving them power to deal with novel unexpected situations, arming them with skills such as integrating information from all senses in a single scene or perception, and also the power to automatize learned behaviours. These skills can be used to solve many real world problems in engineering, and would be of great value in an artificial agent.

The rest of this Chapter is organized as follows: in sections 3.1 and 3.2, we present some of the theories of consciousness developed recently. Our objective in presenting these theories is not to cover all approaches that can be found in literature, neither produce a tutorial comparing these different approaches. Some authors, such as Cavanna & Nani (2014), have dedicated a whole book for this matter. Our intention is to illustrate some of the different theories, so the reader can take a short trip, before getting to the concepts of the main theory we are going to stand on, and have a glimpse of what human mind has recently produced trying to explain the consciousness phenomenon. We will be using the same structure as Cavanna & Nani (2014) in order to present the many approaches: split into philosophical theories (section 3.1), whose purpose is to examine and sift out

different conceptual possibilities, in order to provide a theoretical framework for a valid account of mental phenomena, and scientific theories (section 3.2), whose purpose is to use experimental applications of scientific techniques (e.g., PET and fMRI scans of living brains) to provide useful insights into the brain mechanisms, which bring about our mental functions, the so-called neural correlates of consciousness. As this work's focus is the Global Workspace Theory, we will present in section 3.3 a detailed explanation of the Global Workspace Algorithm. After that, in section 3.4 we present some of the state of the art of applications of the Global Workspace Theory. We close this Chapter by stating our definition of machine consciousness in section 3.5, to be used hereafter.

3.1 Philosophical Theories of Consciousness

3.1.1 René Descartes and the Mind-Body Dualism

René Descartes (1596-1650), the only author cited in this Chapter who was not alive by the time this work was produced, is commonly considered the father of modern philosophy. Descartes' view of the natural world in pure mechanical terms heavily influenced the course of the Western thought (Cavanna & Nani, 2014).

Descartes proposed the mind-body problem as we know it. His position, known as Cartesian or classical dualism, holds that the human beings are composed of two substances, one which is purely material or corporeal, the res extensa, and another which is purely immaterial or intellectual, the res cogitans. According to Descartes, human beings are the only creatures in the world to be born with a soul. Animals, on the contrary, are made by one ingredient, that is, matter, and thereby can be exclusively described in mechanical terms.

Descartes was never able to fully explain how the material and the intellectual substances were supposed to interact. His hypothesis was that these two substances would encounter in one brain organ called pineal gland, shown in Figure 3.1, which Descartes believed to be the only brain organ that was not paired. Descartes' hypothesis of the pineal gland as the point of privileged contact between mind and body failed to give a

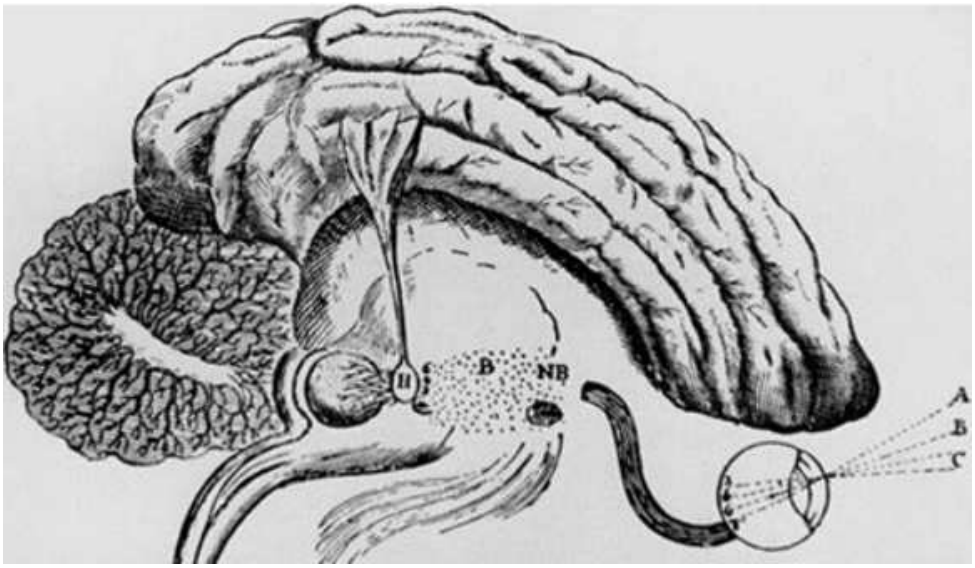


Figure 3.1: Figure from Descartes' *De homine (1662)*, showing the location of the pineal gland, identified by the letter H.

satisfactory solution of the mind-body problem. We now know, in fact, that the pineal gland is involved in the regulation of the circadian rhythm by secreting melatonin and has, like the other structures of the brain, a left and a right side that are mirror images of each other (Cavanna & Nani, 2014).

3.1.2 John Searle and the Biological Argument

Searle (1980) introduced a famous thought experiment known as the "Chinese Room", to challenge the claim that it is possible for a computer running a program to have a "mind" and "consciousness" in the same sense that people do.

Searle's thought experiment begins with this hypothetical premise: suppose that artificial intelligence research has succeeded in constructing a computer that behaves as if it understands Chinese. It takes Chinese characters as input and, by following the instructions of a computer program, produces other Chinese characters, which it presents as output. Suppose that this computer performs its task so convincingly that it comfortably passes the Turing test: it convinces a human Chinese speaker that the program is itself a live Chinese speaker. To all of the questions that the person asks, it makes appropriate responses, such that any Chinese speaker would be convinced that he is talking to another Chinese-speaking human being.

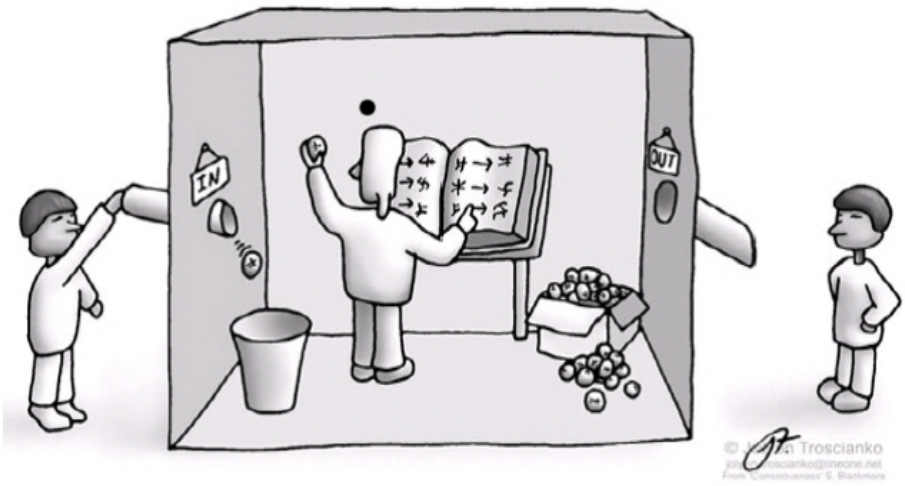


Figure 3.2: Illustration of the thought experiment proposed by Searle called Chinese Room Experiment.

The question Searle wants to answer is this: does the machine literally "understand" Chinese? Or is it merely simulating the ability to understand Chinese? Searle calls the former position "Strong AI" and the latter "Weak AI".

Searle then supposes that he is in a closed room and has a book with an English version of the computer program, along with sufficient paper, pencils, erasers, and filing cabinets. Searle could receive Chinese characters through a slot in the door, process them according to the program's instructions, and produce Chinese characters as output. If the computer had passed the Turing test this way, it follows, says Searle, that he would do so as well, simply by running the program manually, as shown in Figure 3.2.

Searle asserts that there is no essential difference between the roles of the computer and himself in the experiment. Each simply follows a program, step-by-step, producing a behavior which is then interpreted as demonstrating intelligent conversation. However, Searle would not be able to understand the conversation. Therefore it follows that the computer would not be able to understand the conversation either.

Searle argues that without "understanding" (or "intentionality"), we can not describe what the machine is doing as "thinking" and since it does not think, it does not have a "mind" in anything like the normal sense of the word. Therefore, he concludes that "Strong AI" is false.

According to Searle's argument, a program cannot give a computer a "mind", "under-

standing" or "consciousness", regardless of how intelligently it may make it behave. The argument is directed against the philosophical positions of functionalism and computationalism, which hold that the mind may be viewed as an information processing system operating on formal symbols.

Searle (1997) describes consciousness as a biological problem. He refutes what Strong AI states, that implementing the right software in any hardware might produce mental states as emergent properties. For Searle, consciousness arises not from software, but from the "hardware", from organic biological brains, an organic machine. Consciousness, according to the philosopher, "is caused by lower-level neuronal processes in the brain and is itself a feature of the brain".

3.1.3 Daniel Dennet - Denying Consciousness

Dennett (1991) states that there is no hard problem, the whole idea of the inner subjective movie is nothing but an illusion or confusion. In Dennet's view, if we are able to explain the dynamics, functions and behaviors of the brain, then we have explained everything that there is to be explained. To think otherwise would be as believing that, inside the brain, in this privileged place, an "I" or a sort of homunculus would watch all the mental representations like a spectator in a theater.

3.1.4 David Chalmers and the Naturalistic Dualism

Chalmers (1995, 1996) is a scientific materialist, but he argues that all forms of physicalism, whether reductive or non-reductive, that have dominated modern philosophy and science, fail to account for the existence of consciousness itself. He views consciousness as an anomaly that we can not integrate in our current view of the world. He proposes an alternative dualistic view he calls naturalistic dualism, in which consciousness or "experience" is a fundamental building block, just as some aspects of the universe, like space, time or mass in physics. These fundamental building blocks have properties and laws that do not have an explanation as anything more basic. They are taken as primitives and the world is built upon them. Chalmers argues that, just as Maxwell could not ex-

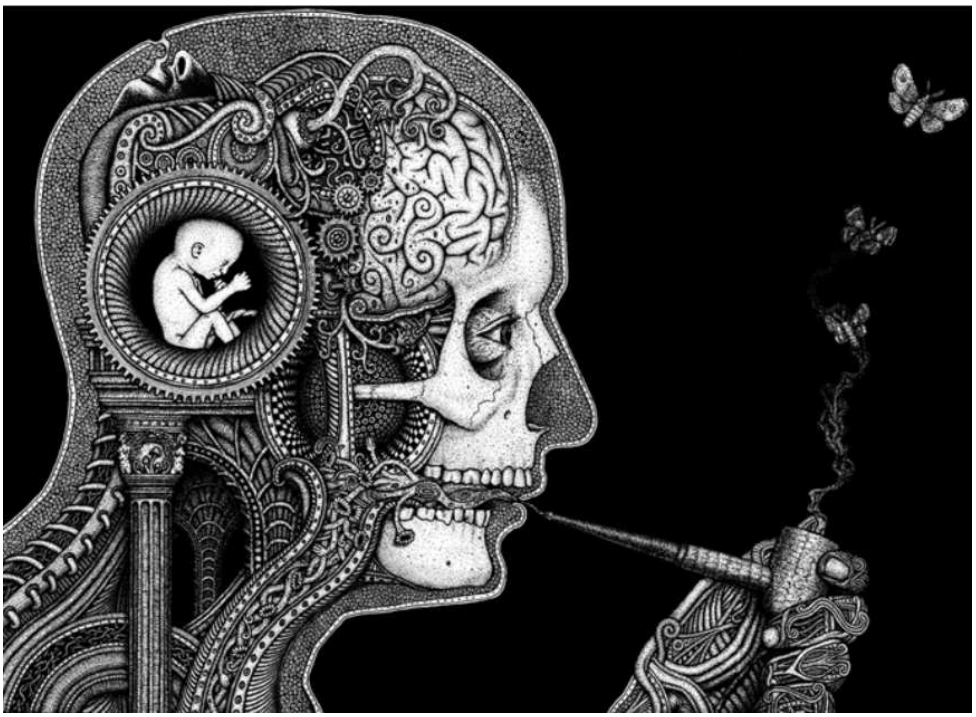


Figure 3.3: Art representing the Homunculus hypothesis.

plain electromagnetism in the 19th century with the fundamentals space, time and mass from Newtonian laws, and had to write new fundamentals, like charge, and develop the laws that connected them to space, time and mass, so we must do to be able to explain consciousness - postulate "experience" as a fundamental building block of nature. In this view, just like space and time are universal, so would be "experience", a property of any system in nature, from photons to human brains, in different degrees, a panpsychic view.

One important concept that Chalmers put forward is the organizational invariance, which states that if two systems have the same fine-grained functional organization, they will also have qualitatively identical experiences. This principle predicts that computers will be conscious, when they are able to replicate the functional organization of the human brain. Many philosophers maintain that consciousness merely depends on a specific functional organization, which in turn does not rely on the characteristic structure of the system or the "hardware".

Chalmer's theory confronts us with a new metaphysical viewpoint on nature. According to his theory, all information might have an intrinsic phenomenal aspect, so that where there is simple information processing, there is also simple conscious experience,

and where there is complex information processing, there is also complex conscious experience. This is the dualism in his theory, different from Descartes' Cartesian dualism. Since information is everywhere, we should conclude that consciousness too is, in different degrees, everywhere. This position is called panpsychism by philosophers and considered by many as a very counterintuitive conception of reality, as it is difficult to believe that a thermostat may have a phenomenal experience whatsoever (Cavanna & Nani, 2014).

One important argument Chalmers developed to confront the idea that mind and brain are one thing is the idea of what is called philosophical zombies. A philosophical zombie is a perfect physical copy of a person, but with no inner conscious experience. Chalmers argues that, since it is possible to conceive the idea of such a zombie, the relationship between the conscious mind and brain is to be contingent rather than necessary. Some philosophers accept this argument, while others take it as fallacious and misleading. Their main point is that the argument is logically incoherent, because if mind and brain are the same entity, then it is not coherently possible to imagine one without the other. In other words, following the philosophical zombie argument would be similar to conceiving a pencil that does not write. What kind of pencil would it be? And would it still be a pencil?

3.2 Scientific Theories of Consciousness

3.2.1 Giulio Tononi - Consciousness as Integrated Information

Tononi (2008) developed the integrated information theory (IIT) in order to measure consciousness as an integrated information property of matter. The author states that:

- consciousness is a specific process by which information is integrated;
- the quantity of consciousness corresponds to the amount of integrated information generated by a complex of elements;
- the quality of experience is specified by the set of informational relationships generated within that complex;

- Integrated information (ϕ) is defined as the amount of information generated by a complex of elements, above and beyond the information generated by its parts.

The author describes a different and new view of nature based on these assumptions. Tononi's view that every system has a measure ϕ of consciousness is a panpsychic view, aligned with the idea of consciousness as a fundamental primitive of the universe, like Chalmers states.

Figure 3.4 illustrates some results of the calculation of ϕ according to Tononi's integrated information theory relating to neuroanatomy, in A, B, C and D, and neurophysiology, in E and F. The more integrated information and states a system is able to generate, the greater is the measure ϕ of the system.

Tononi's theory is currently reputed as one of the most promising theories in the scientific community (Cavanna & Nani, 2014). It is believed that, if we can explain how these little unities of consciousness add up from a microbe to a human mind, we are on our way to fully explaining consciousness and integrating it in our scientific view of the world.

3.2.2 Christof Koch, the Romantic Reductionist

Koch (2012) believes that consciousness is a fundamental, an elementary, property of living matter that can not be derived from anything else - a simple substance. The author believes that the property of consciousness arise not from the mechanism of how the neuron works, but from how the network of neurons is organized, from its complexity. The author refers to the measure of complexity ϕ from Tononi's theory to state that brains with a higher ϕ are somehow superior to brains that are less integrated.

3.2.3 Jeff Hawkins and the Memory Prediction Framework

According to Hawkins (2004), the brain is not a computer, but a memory system that stores experiences in a way that reflects the true structure of the world, remembering sequences of events and their nested relationships and making predictions based on those memories. It is this memory-prediction system that forms the basis of intelligence,

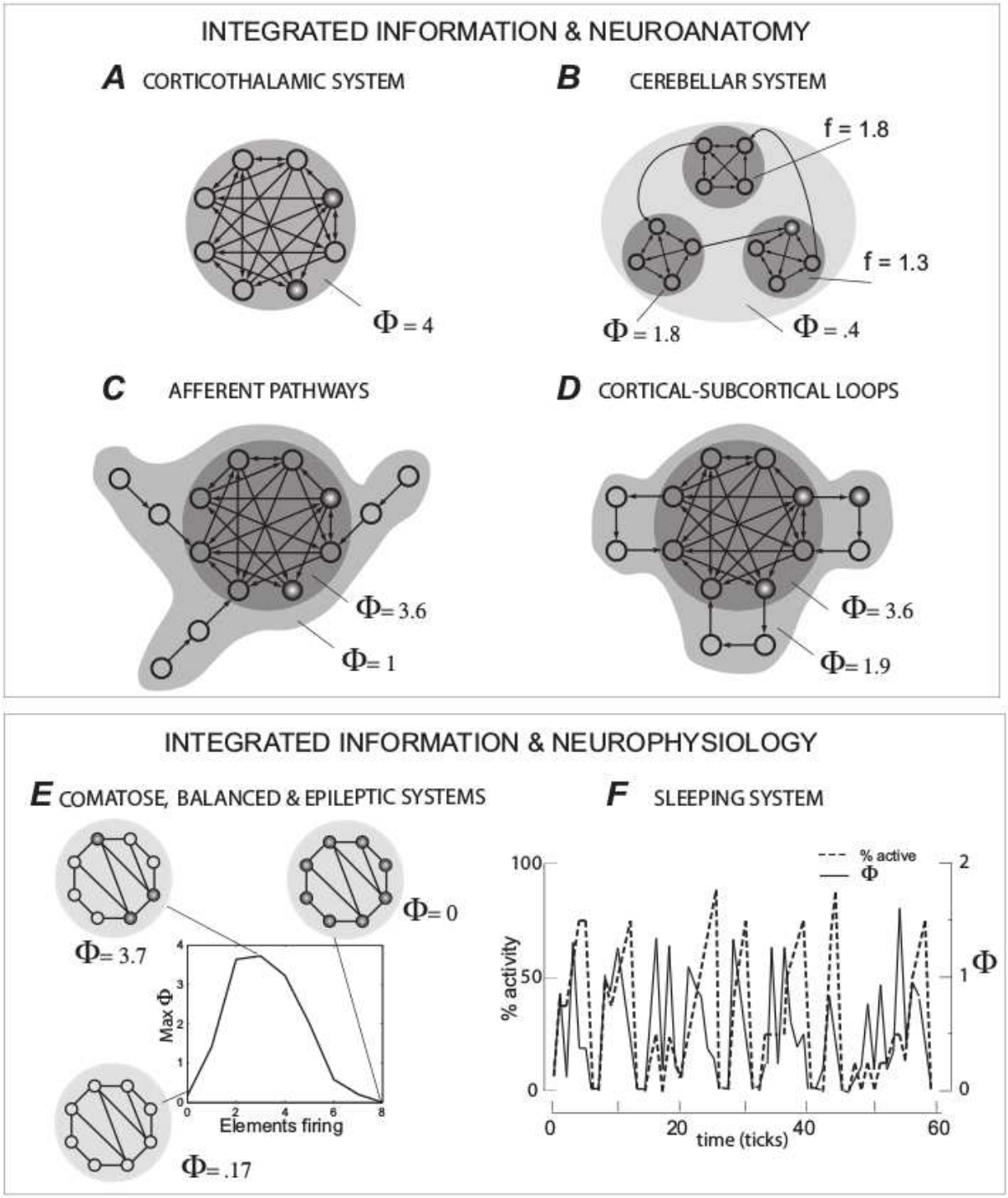


Figure 3.4: In A, computing ϕ in simple models of neuroanatomy suggests that a functionally integrated and functionally specialized network—like the corticothalamic system—is well suited to generating high values of ϕ. In B, C and D, architectures modeled on the cerebellum, afferent pathways, and cortical-subcortical loops give rise to complexes containing more elements, but with reduced ϕ compared to the main corticothalamic complex. In E, ϕ peaks in balanced states; if too many or too few elements are active, ϕ collapses. In F, a bistable ("sleeping") system (same as in E), ϕ collapses when the number of firing elements (dotted line) is too high (high % activity), remains low during the "DOWN" state (zero % activity), and only recovers at the onset of the next "UP" state. Figure adapted from Tononi (2008).

perception, creativity and consciousness.

The main points of the Memory-Prediction framework can be summarized as follows:

1. The neocortex is constructing a model for the spatial and temporal patterns that it is exposed to. The goal of this model construction is the prediction of the next pattern on the input.

2. The cortex is constructed by replicating a basic computational unit known as the canonical cortical circuit. From a computational point of view, this canonical circuit can be treated as a node that is replicated several times.

3. The cortex is organized as a hierarchy. This means that the nodes – the basic computational units – are connected in a tree shaped hierarchy.

4. The cortex function is to model the world that it is exposed to. This model is built using a spatial and temporal hierarchy by memorizing patterns and sequences at every node of the hierarchy. This model is then used to make predictions about the input.

5. The neocortex builds its model of the world in an unsupervised manner.

6. Each node in the hierarchy stores a large number of patterns and sequences. The pattern recognition method employed by the cortex is largely based on storing lots of patterns.

7. The output of a node is in terms of the sequences of patterns it has learned.

8. Information is passed up and down in the hierarchy to recognize and disambiguate information and propagated forward in time to predict the next input pattern.

To illustrate the differences between a computer and this memory-prediction system, Hawkins uses the example of the task of catching a ball. He argues that this seems to be a simple task for humans, until they try to program a robot arm to do the same. It is actually nearly impossible. When engineers or computer scientists approach the problem, they try to calculate the flight of the ball to determine where it will be when it reaches

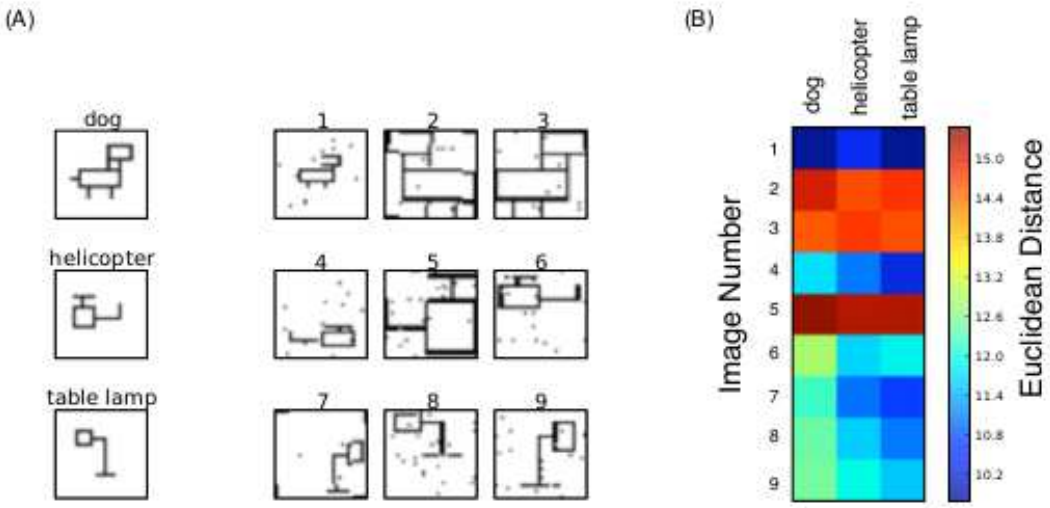


Figure 3.5: In A, three simple prototypes of binary images of a dog, a helicopter and a table lamp are shown in the left, and nine variations of these in the right. In B, Euclidean distances between the prototypes and the variations are shown. The closest prototype is the one that is at minimum distance from the image. If we use Euclidean distance as a measure of similarity, most of the images are misclassified. Image 4, which is actually a helicopter, will be classified as a table lamp. It can be concluded from this plot that a Euclidean distance measure does not inform us about perceptual similarity. Figure adapted from George (2008).

the arm. This requires solving a set of simple equations. Second, all the joints of a robotic arm have to be adjusted to move the hand in the proper position. A set of mathematical equations more difficult than the first ones have to be solved. Finally, this algorithm has to be repeated multiple times, as the ball approaches. It will require millions of steps in order to catch the ball, which would be impossible for a biological brain to perform in seconds. Hawkins believes the brain does it using memory and invariant representations, which handle variations in the world automatically. Retrieving a memory is much faster than computing millions of steps. However,the problem of understanding how the cortex form invariant representations is yet to be solved.

Figure 3.5 illustrates how the problem of invariant representations, so well resolved by a 4 years old human brain, is far from trivial computationally. In A, three simple prototypes of binary images of a dog, a helicopter and a table lamp are shown in the left, and nine variations of these in the right. In B, Euclidean distances between the prototypes and

the variations are shown. The closest prototype is the one that is at minimum distance from the image. If we use Euclidean distance as a measure of similarity, most of the images are misclassified. Image 4, which is actually a helicopter, will be classified as a table lamp. Human brains somehow store an invariant representation of objects like dogs, helicopters and table lamps, and are very good at classifying them in real time, no matter how distorted, cut or broken they are. According to Hawkins' theory, the brain does so by retrieving invariant representations and predicting patterns all the time.

For Hawkins, prediction is the most important feature of the brain, the essence of intelligence. He states that prediction, not behavior generation, is the real advantage that the neocortex brought to humans: seeing some steps into the future gives great advantage to our species. The neocortex use basically memories and invariant representations to predict.

Consciousness, in this memory-prediction framework, is seen by Hawkins as "what it feels like to have a cortex", that is, the subjective experience of being able to recall memories and make predictions, that can also be used as inputs to other predictions, and so on.

The Memory-Prediction framework, as expressed in Hawkins (2004), is a biological theory. George (2008) worked on the foundation established by Hawkins and developed the algorithmic and mathematical counterparts of the Memory-Prediction framework, called Hierarchical Temporal Memory (HTM). The author also developed a set of algorithms to deal with learning and invariant recognition for hierarchical-temporal data, a theory for analysis of generalization in hierarchical-temporal models and a mathematical model for cortical microcircuits.

3.2.4 Bernard J. Baars and The Dynamic Global Workspace Theory

Bernard Baars has developed one of the most "implementable" theories in consciousness studies. The Global Workspace Theory (GWT) (Baars, 1988; Edelman et al., 2011; Baars et al., 2013) is largely inspired by the "blackboard model" from the beginning of

artificial intelligence (Nii, 1986). Due to its computational origins, it is, among other consciousness theories, one which is particularly interesting for deriving computational models, being very popular in the field of "machine consciousness". GWT suggests that only one integrated sensory content can be dominant in the brain in a given moment. Unconscious content compete for access to the limited capacity of this workspace. This dominant content is then broadcasted to other regions of the brain, in a nervous system seen as a set of massive distributed small networks with specialized purpose. In such a system, coordination, control and resolution of new problems take place with the exchange of centralized information. This theory tries to conciliate the limited capacity of conscious content with the vast repertoire of long term memories. It states that the limited capacity of conscious content brings advantages to animal survival, because it helps the animal to focus in what is most important in a given critical situation. The most recent version of GWT is called Dynamic GWT, since it fits the evidence for conscious cognition in the cortex. The concept of a dynamic core provides a mechanism for events in the Global Workspace, as it projects brain signals in the cortex in a reentrant manner (Edelman et al., 2011; Baars et al., 2013).

According to this model, consciousness is like an information gateway to the brain, because it allows a widespread structure of neuronal networks to operate in order to integrate, provide access, and coordinate the processing of many specialized brain sites, which would otherwise operate autonomously. This widespread architecture of neuronal networks has been described by Baars using the metaphor of the global workspace theater, a sort of cognitive stage in which mental functioning occurs at both conscious and unconscious level. Within this picture, consciousness would be like a spotlight on the stage of working memory, guided by selective attention.

In the GWT, the unconscious processes have a very important role. These processes remain in the dark or "behind the scenes" and their activity is, accordingly, invisible. They are however able to influence and orient consciousness. Baars calls these invisible brain processes contexts, which are coalitions of unconscious processes relatively stable in time. Some examples are the executive functions, which operate like the theater director; the

linguistic modules, which can be compared to scriptwriters. The dorsal cortical stream of the visual system, which shapes vision like stagehands shape the theatrical scenery. Despite this hidden activity, a thought or a content of experience must always enter the global workspace or, metaphorically speaking, be on the theater stage in order to be conscious.

Since neuroimaging studies have shown how the conscious mind involves the activation of widespread neuronal networks, Baars' global workspace model has gained increasing attention within the neuroscientific community. The model has therefore been developed not only in its theoretical aspects but also and especially within a neuroanatomical context (Cavanna & Nani, 2014).

3.3 The Global Workspace Algorithm

In order to understand the GWT algorithm, that is, how consciousness works in the brain, according to Baars' theory, the metaphor of the interactive theater is of special interest (see Figure 3.6). In this metaphor, consciousness works like an interactive theater play. In this theater, we find a big silent audience surrounding a center stage, where all activities take place. There are many things going on at the stage, but just a selected part of the stage is illuminated by a spotlight. The workers under the spotlight can be viewed by all those at the audience and at the stage. At any point, a member of the audience, looking at what is going on under the spotlight, might be compelled to do some work. It then moves to the stage and starts its performance. By finishing its performance, it comes back to its place at the audience. During the workers performance, there is an intrinsic competition for the spotlight at the stage. Those workers at the stage who cry louder might attract attention from the spotlight manager, and gain access to the light. Once they gain the light, they can be seen by all other agents both at the audience and at the stage. The play is a never ending play. Agents can go down to stage, develop their performance, and return to the audience many times. Once they are at the stage, they are intrinsically competing for the spotlight. Once a performer gains the spotlight it can be seen by all the others, which might be compelled by its activity, to go to stage and start performing on

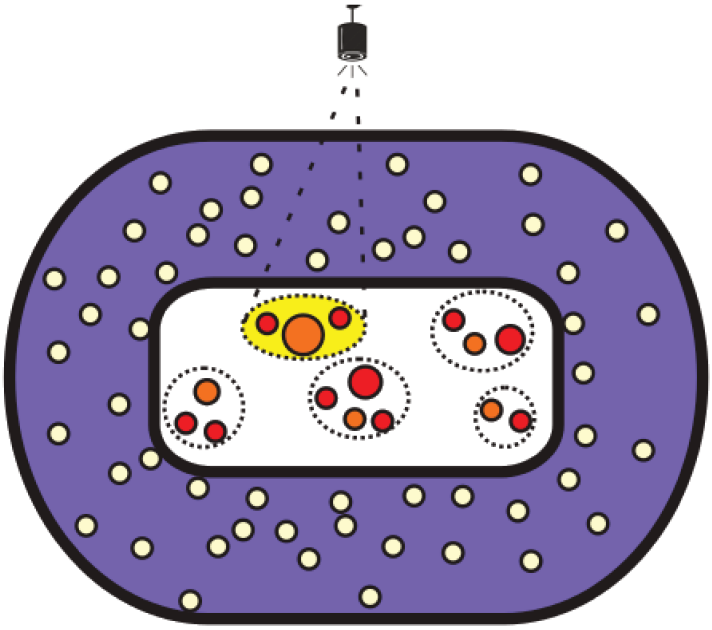


Figure 3.6: Baars' interactive theater. The audience is shown as light yellow circles in the purple area, and the stage in gray with red and orange actors. The bigger the circles, the more activated they are. The dark yellow area is the area under the spotlight, representing the conscious processes, which broadcast their information.

their own. This process goes on indefinitely. In this metaphor, the agents are independent brain processes. Those processes at the audience, together with those processes at the stage who are not at the spotlight (the "context") are unconscious processes. While they are silent at the audience or doing something at the stage, they are all unconscious. Only those at the spotlight are said to be conscious. Conscious processes have the unique characteristic of being able to influence silent unconscious processes to perform some activity. Any process might become conscious for some time, while performing at the stage and attracting the attention of the spotlight manager. Processes can become unconscious or conscious, at any time. The stage represents the working memory - brain processes and information ready and close to becoming conscious.

Brain processes are able to collaborate to each other. While doing a collaborative work they are said to integrate a coalition. Coalitions can be formed and dissolved while agents are compelled to work due to what is going on at the spotlight. When many agents are collaborating together, their voice becomes louder, as a group, and there is a greater chance of being chosen by the spotlight manager. In GWT, conscious content is the result of coalitions being identified as important or particularly relevant in a given instant of

time. This illustrates the limited and serial capacity of consciousness.

In conclusion, GWT states that there is a global workspace formed by a dominant coalition of contexts. This core is dynamic, in a sense that it is not localized in a specific region of the brain, but it moves around according to the activation of the small networks in the brain, as if it was a serialization of the parallel activity in the brain. This idea leads us to our definition of Machine Consciousness, which will be presented in section 3.5.

3.4 State of the Art in Applications of the GWT

3.4.1 Stan Franklin and the Learning Intelligent Distribution Agent (LIDA)

Stan Franklin's research group, from the University of Memphis, was the first to develop a machine consciousness algorithm inspired in the Global Workspace Theory (Baars, 1988). This algorithm was applied with success to solve different problems such as the creation of a virtual personal secretary (C-Mattie), an American Navy tasks planner (IDA - Intelligent Distribution Agent) and an intelligent tutor system to train Canadian astronauts on how to operate a mechanical arm for the International Space Station (Bogner, 1999; Negatu, 2006; Dubois, 2007).

Based in his previous experience in developing systems with consciousness, Franklin evolved the LIDA architecture (Learning Intelligent Distribution Agent) (Baars & Franklin, 2009; Franklin et al., 2014a), as both a conceptual and computational model grounded mainly in the GWT (Baars, 1988).

The LIDA model and its architecture are based in a cognitive cycle that can be divided in three phases: perception, interpretation and action. An agent's life can thus be seen as a continuous sequence of such cognitive cycles. The architecture uses many computational mechanisms known in the literature, such as the Copycat Architecture (Hofstadter & Mitchell, 1994), Sparse Distributed Memory (Kanerva, 1988), Schema Mechanism (Drescher, 1991) and Behavior Net (Maes, 1989). The LIDA model is presented in Figure 3.7.

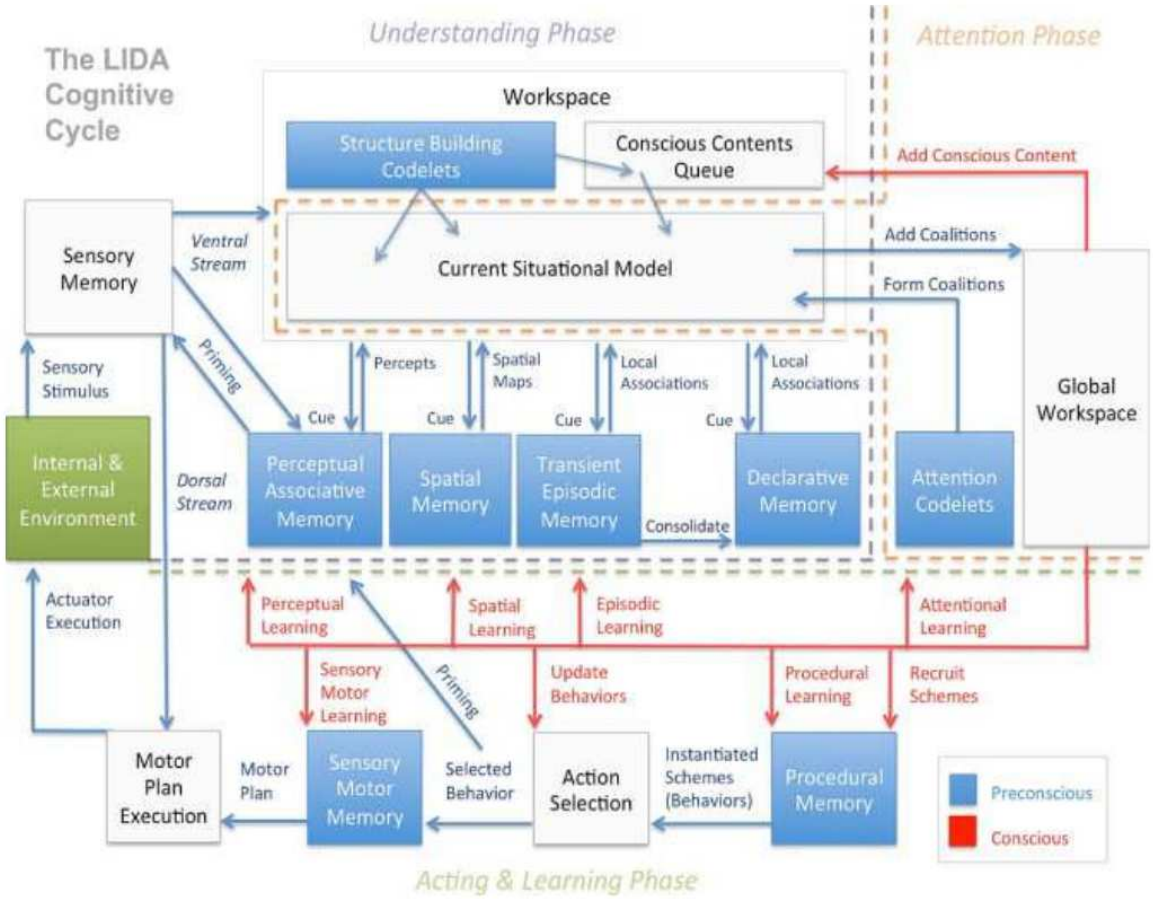


Figure 3.7: The LIDA model

Franklin's LIDA makes heavy use of the concept of Codelets. Codelets, following Hofstadter & Mitchell (1994), are small pieces of non-blocking code, each of them executing a well defined and simple task. The idea of a codelet is of a piece of code which ideally shall be executed continuously and cyclically, time after time, being responsible for the behavior of a system's independent component running in parallel. A codelet can be seen as a nervous system basic functional unit, like the cortical column Mountcastle (1978) first described. In computational systems, codelets are scalable nodes in a network.

Coalitions are groups of one or more codelets. Coalitions are formed by codelets that sum their skills in order to perform more complex tasks, which they would not be able to perform by themselves in an isolated manner.

In the perception phase, sensory stimuli are grabbed from the internal and external environment and stored in the *Sensory Memory*. These stimuli work as cues for the *Perceptual Associative Memory*, which will give rise to Percepts in the Current Situational

Model. This mechanism works as follows. When one stimulus is identified as relevant, one node in the Perceptual Associative Memory receives a greater activation. This node passes on this activation to other related nodes, called links. When this process stabilizes, the node group that received sufficient activation is referenced as a *Percept* and moved to the Current Situational Model. During this stabilization phase, the *Percept* is seen as a group of ontology elements relevant to the stimulus. It is written as a binary vector called a *cue*, that is later used to search for an autobiographical memory, a declarative memory, and a transient episodic memory. The transient episodic memory has information about events that the agent remembers. The declarative memory has information learned during lifetime, including some pieces of information which were once in the transient memory. The search return new *cues*, that are in turn used for a new search. This process repeats itself until a new information is returned.

During the interpretation phase, the *percept* created and stabilized in the perception phase is copied to the long term working memory, in a workspace secondary partition, where it will join older *percepts*. Attention Codelets analyze long term working memory content, searching for interesting elements. Coalitions of attention codelets and *percepts* are formed whenever it is possible. The coalition with greatest activation wins and its content is moved to the Global Workspace to be broadcasted.

In the action phase, the conscious broadcast goes to each agent's subsystem, including the Procedural Memory. The Procedural Memory is a collection of organized schemes in the form of a scheme net. When the conscious broadcast provides information that combines the context of one or more schemes, the procedural memory will suggest schemes that should be copied to the action selection model. The action selection model, implemented as a Behaviour Net, will chose which action is most appropriated for the context in place. The action is sent to the Sensory-Motor Memory, that will execute the action in the environment.

In da Silva (2009), our research group investigated the LIDA model. This was our first contact with the GWT. The work suggested that this mechanism is in fact capable of promoting an executive summary of the perception, and the automatization of new

behaviours.

3.4.2 Dehaenne and the Excitatory Neurons Model

Dehaene & Naccache (2001) also proposed a cognitive architecture with a Global Workspace. In their model, sensory stimuli mobilize excitatory neurons by means of cortico-cortical axons, giving rise to activity patterns in the workspace neurons.

3.4.3 Shanahan and the Imagination Architecture

Shanahan (2006) proposed a cognitive architecture with concepts of consciousness, imagination and emotion. Shanahan adopts an information flow model based on the GWT. The planning in his model is realized by means of internal simulations, resembling the imagination process in the brain.

3.5 Our Definition of Machine Consciousness

The concept of machine consciousness we adopted in this work, to be used hereafter, is the following, depicted in Figure 3.8: machine consciousness is the emergence of a serial integrated information flow on top of a group of parallel interactive devices (Dennett, 1991; da Silva, 2009; Baars & Franklin, 2009).

The reader might be conjecturing by this time, what does the interactive theater metaphor and the GWT algorithm have to do with the notion of consciousness. Let's make a brief analysis of the GWT algorithm and its consequences. Assuming the brain processes as a large set of parallel devices which remain silent for some time, but perform some activity from time to time, the spotlight manager works like a serialization process on top of this network of parallel devices. By focusing attention step by step in different devices, the spotlight manager makes it emerge something which might be compared to what William James refers as the "stream of consciousness". For most of the time, what is appearing to consciousness (what is at the spotlight) are perceptual processes, abstracting some particular pattern identified at the sensory space. But from time to time, some

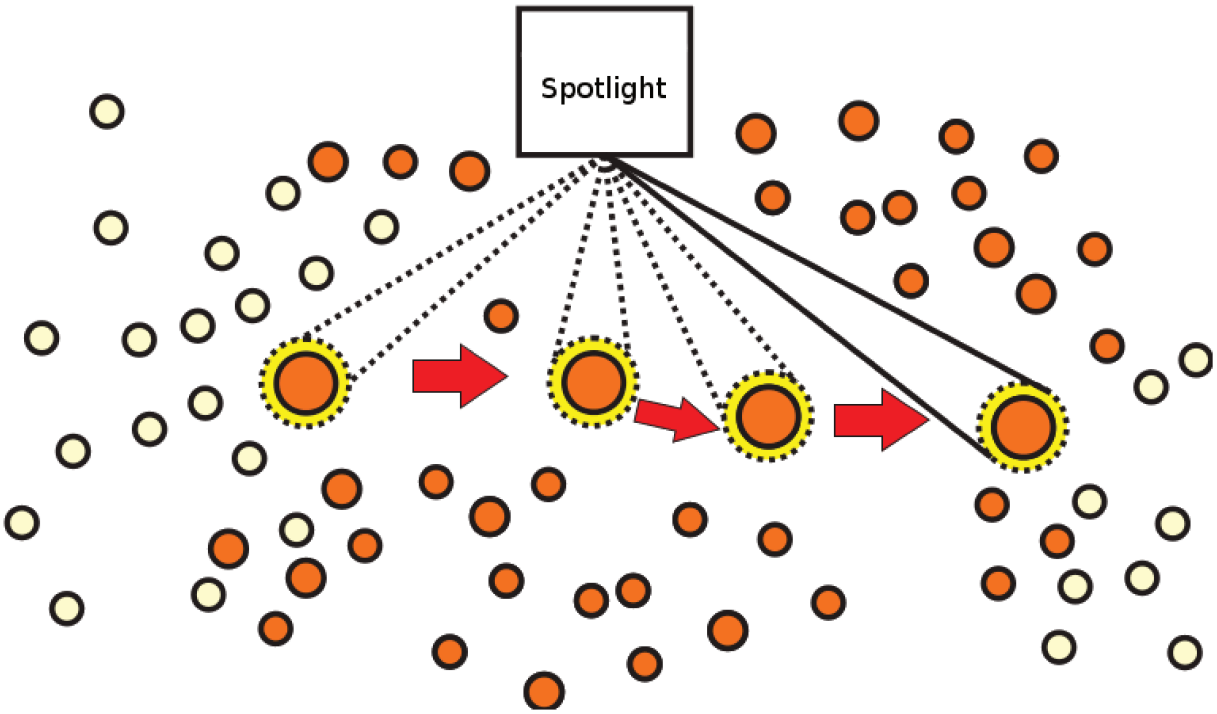


Figure 3.8: Machine consciousness is the emergence of a serial integrated information flow on top of a group of parallel interactive devices.

memory processes gain access to consciousness, and we remember something which is important. After that, some planning activity might gain the consciousness, and a sequence of possible future actions are being broadcasted through consciousness. Depending on what is going on, perceptual processes start to gain again access to consciousness, and we see that a serial flow of brain processes start to emerge as the effect of a coordination process.

Machine consciousness can be seen then as a special algorithm in a parallel network of processes, which is capable to coordinate such processes in a way leading to the emergence of a serial flow, where the most important or relevant piece of information is available, and made broadcast to all other processes. This creates order in a potentially chaotic manifold of information, emphasizing the information which is, at each time instant, the most relevant in driving future system behavior.

This coordinative characteristic of machine consciousness, emphasizing relevance at each timestep, is essential for us to understand its utility in computational systems.

Chapter 4

The Cognitive Systems Toolkit

The last theoretical issue we will cover as a background to our thesis is the domain of cognitive architectures. This issue started to arouse as an offspring on the study of intelligent agents. We can understand a Cognitive Architecture as the reuse of specific cognitively inspired strategies and designs in the pursue of a control architecture for an intelligent agent (Langley et al., 2009; Samsonovich, 2012a). These architectures are a special kind of control system that use inspiration on how cognitive functions from the human (or animal) mind might be computationally implemented and used to provide intelligence to an artificial agent.

In section 4.1, we provide an overview on the concept of cognitive architectures and cite the most classical approaches. In section 4.2, we present the CST - Cognitive Systems Toolkit, a toolkit being developed by our research group, which we used as the main basis for the construction of the cognitive architecture controlling our traffic lights.

4.1 Cognitive Architectures

Cognitive architectures are control systems inspired by scientific theories developed to explain cognition in humans and animals. Cognitive Architectures have been employed in many different kinds of applications, since the control of robots to decision-making processes in intelligent agents. Usually, a cognitive architecture is composed by modules implementing cognitive capabilities, like perception, attention, memory, reasoning, learning, behavior generation, etc. Cognitive Architectures are, at the same time, theoretical models for how different cognitive processes interact to each other in order to sense, reason and act, and also a software framework which can be reused throughout different applications. An example of a typical cognitive architecture is given in Figure 4.1, taken from Franklin & Graesser (1997).

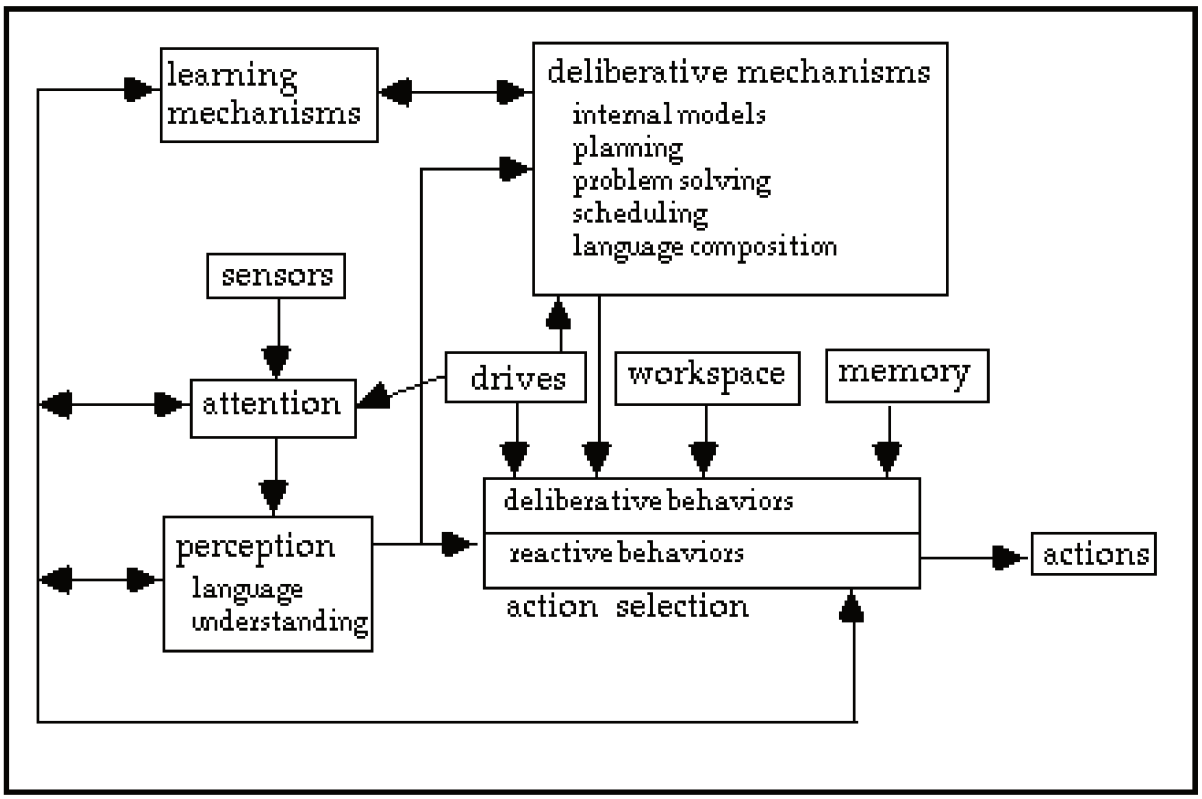


Figure 4.1: A Typical Cognitive Architecture, reproduced from Franklin & Graesser (1997).

There are some classical cognitive architectures described in the literature, such as SOAR (State, Operator And Result) (Laird, 2008) and ACT-R (Atomic Components of Thought) (Anderson et al., 2004). Even though these architectures have their roots on rule based systems, they recently evolved to become full cognitive architectures. SOAR was originally proposed by John Laird, Allen Newell and Paul Rosembloom, starting around 1983 and is in constant development, evolving through many different versions. More recently, many specialized architectures were proposed (see Samsonovich (2012b) for a comprehensive table), such as CLARION (Connectionist Learning with Adaptive Rule Induction ON-line) (Sun, 2006) and LIDA (Learning Intelligent Distribution Agent) (Baars & Franklin, 2009), emphasizing different aspects of human and animal cognition, like emotion, attention, memory, consciousness and language. LIDA is an evolution of a previous cognitive architecture called IDA, which was proposed by Stan Franklin around the 90's based on Bernard Baars' Global Workspace Theory. CLARION was proposed by Ron Sun also in the 90's and was recently restructured.

Each one of these architectures has advantages and disadvantages, when compared to each other. Lucentini & Gudwin (2015) presented a theoretical analysis comparing SOAR, LIDA and CLARION, three of the most popular cognitive architectures, which have a long tradition of development in their research groups. The methodology used by the authors considered input data/perception (how is sensor data processed and understood?), goals (how do goals and motivations influence on action selection?), action selection (how to choose the best action in the short and long term?) and learning (which are the learning mechanisms and how do they help the agent in order to take an action?). The authors' comparison shows similarities and differences among the architectures, identifying advantages and disadvantages to advise a potential user on how to choose the best architecture to employ, depending on the situation.

The most popular cognitive architectures usually have their code available at the Internet (with different kinds of licenses), such that different researchers are able to download this code and make experiments with these architectures.

4.2 Cognitive Systems Toolkit

There are mainly two kinds of cognitive architectures: frameworks and toolkits. Frameworks provide reuse of previous designs by reapplying the same code in new applications. Toolkits provide a more flexible kind of reuse, when particular cognitive functions and strategies can be chosen in order to build up customized architectures. In this work, we were part of the team which designed and implemented the foundations of the Cognitive Systems Toolkit (CST), and collaborated in the development of a neuroscience inspired cognitive architecture (Raizer et al., 2012), with applications to robotics and assistive technology (Raizer et al., 2013b). Moreover, we specifically extended CST by including a GWT-based consciousness subsystem, and applied it in the implementation of a traffic light controller, what constitutes the main focus of this thesis.

Figure 4.2 illustrates the core of the CST toolkit. Basically, the CST is a toolkit for the construction of specialized cognitive architectures. However, all cognitive architectures built with the help of CST share this common core of structures and concepts. Even though different strategies might be chosen for implementing cognitive functions like perception, memory, action selection, etc, these cognitive functions necessarily will be constructed with the help of this common core. We will be referring to this generic kind of cognitive architecture, being constructed using CST, as a CST Architecture. The two basic concepts, fundamental for understanding a CST Architecture, are the concepts of "Memory Object" and "Codelet". Memory objects are any kind of data structure used to store information and/or knowledge. Using a semiotic terminology, we might refer generically to a memory object as being a "sign". A memory object has a type T, and an encoding of information I. This information can be a single measurement, expressed by a number, or a complex data container, which structure is completely defined by the definition of its type T. Codelets are small pieces of non-blocking code, executing a well defined and simple task. The prototype of a codelet is a piece of code which ideally shall be executed continuously and cyclically, time after time, being responsible for the behavior of an independent component of a system running in parallel. The notion of codelet was

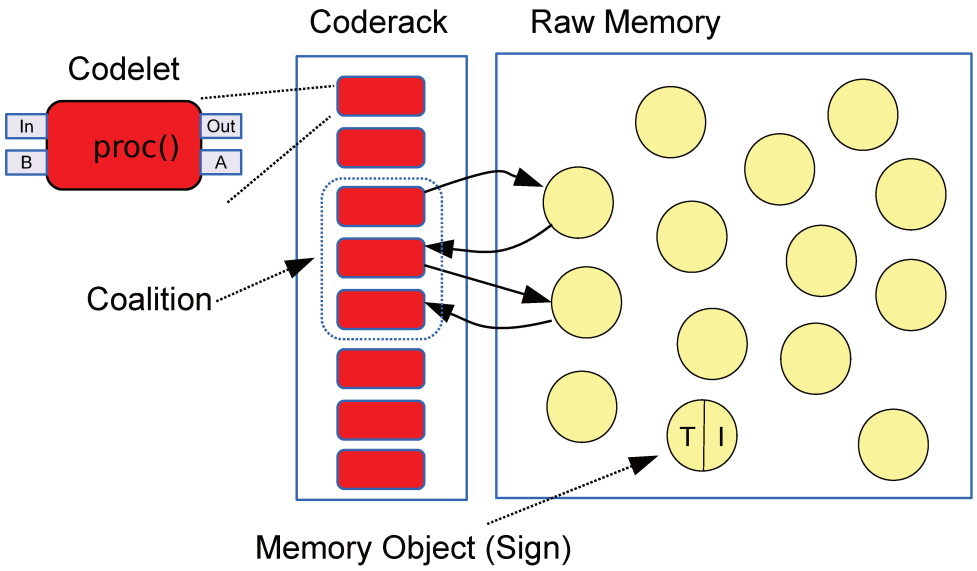


Figure 4.2: The CST Core subsystem

introduced originally by Hofstadter & Mitchell (1994) and further enhanced by Franklin et al. (1998). The CST architecture is "codelet oriented", since all main cognitive functions are implemented as codelets. This means that, from a conceptual point of view, any CST-implemented system is a fully parallel asynchronous multi-agent system, where each agent is modeled by a codelet. CST's codelets are implemented much in the same manner as in the LIDA cognitive architecture (Franklin et al., 2014a) and largely correspond to the special-purpose processes described by Baar's Global Workspace Theory (Baars & Franklin, 2007). Nevertheless, for the system to work, a kind of coordination must exist among codelets, forming coalitions which by means of a coordinated interaction, are able to implement the cognitive functions ascribed to the architecture. This coordination constraint imposes special restrictions while implementing codelets in a serial computer. In a real parallel system, a codelet would simply be called in a loop, being responsible to implement the behavior of a parallel component.

A codelet has two main inputs (which are characterized as *In* and *B* in Figure 4.2), a local input (In) and a global input (B). The local input is used for the codelet to get information from memory objects, which are available at *Raw Memory*. The global input is used for the codelet to get information from the global workspace mechanism (Baars & Franklin, 2007). Information coming from global workspace is variable at each instant

of time, and is usually in the form of a summary, which works as an executive filter to select the most relevant pieces of information available in memory at each time-step. The two outputs of a codelet are a standard output (*Out*), which is used to change or create new information in the Raw Memory, and the activation level (*A*), which indicates the relevance of the information provided at the output. This activation level is also used by the consciousness mechanism in order to select information to be destined to the global workspace.

In order to dig deeper in the details, we invite you to visit our Java implementation of the CST and its Core subsystem, released as open source software, at https://github.com/CST-Group/cst. Figure 4.3 is an Unified Modeling Language (UML) representation of part of the Core subsystem code just referred.

In this implementation, Codelet is an abstract class with three main abstract methods that must be overridden when the class is extended, shown in code below:

```
abstract class Codelet implements Runnable {
    /**
    * This method is used in every Codelet to capture input, broadcast and output MemoryObjects
        * which shall be used in the proc() method.
        * This abstract method must be implemented by the developer. Here, the developer must get the inputs and outputs it needs to perform
              proc.
    */
   abstract void accessMemoryObjects();
   /**
    * This abstract method must be implemented by the developer. Here, the developer must calculate the activation of the codelet before it
          does what it
    * is supposed to do in proc();
    */
   abstract void calculateActivation();
   /**
    * Main Codelet function, to be implemented in each subclass.
    */
   abstract void proc();
}
```

Whenever building new kinds of codelet, in a cognitive architecture built upon CST, the developer only needs to extend the Codelet abstract class and implement these three methods. All the rest is already taken care in the rest of the class implementation, already provided. Indeed, the only thing a developer needs to do in order to build a full CST Architecture is to develop new kinds of codelets or use the ones already provided by CST.

The CST toolkit provides a Reference Architecture, with different kinds of codelets to perform most of the cognitive functions available at a cognitive architecture, as indicated in Figure 4.4.

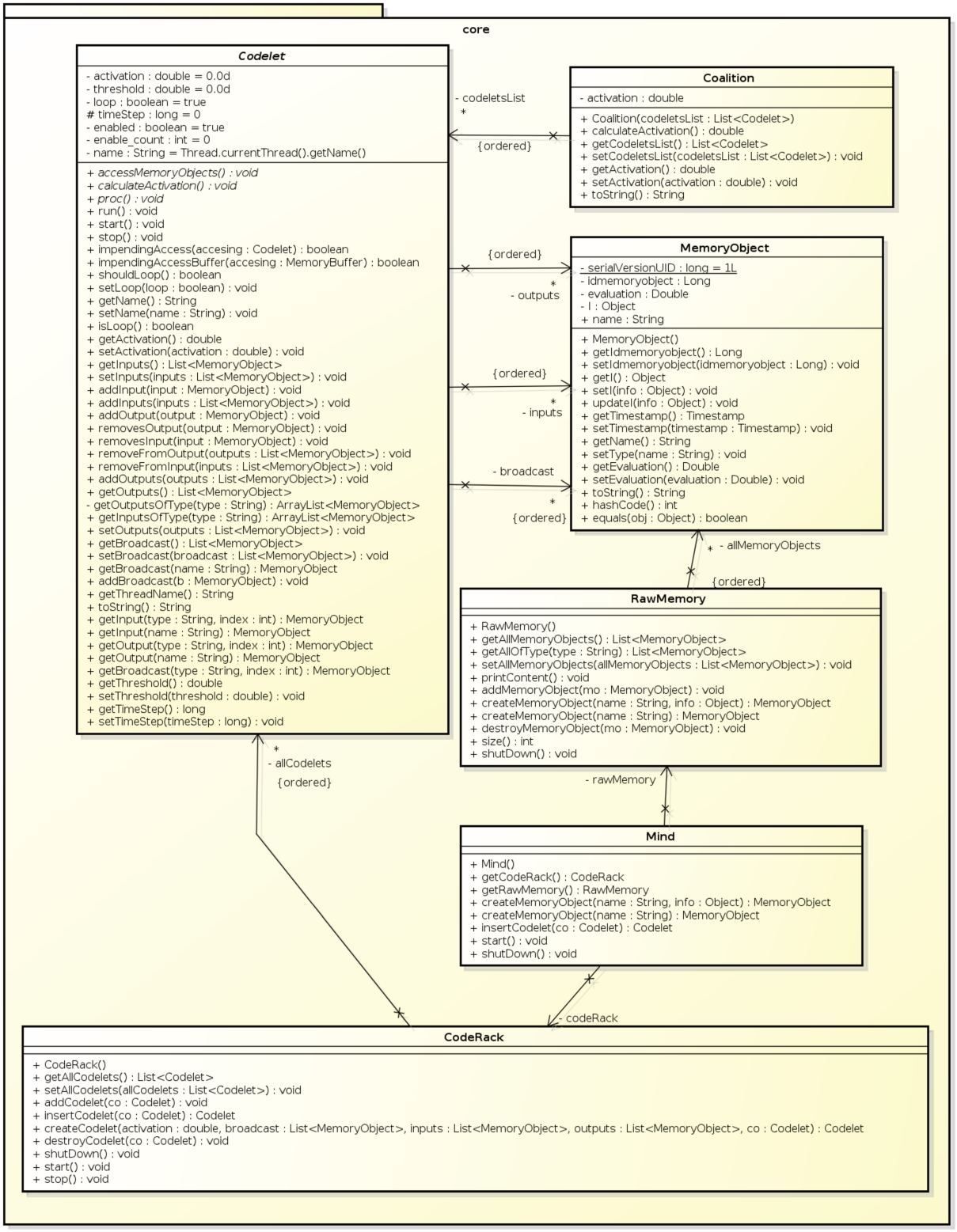


Figure 4.3: Our Java implementation of the CST Core subsystem

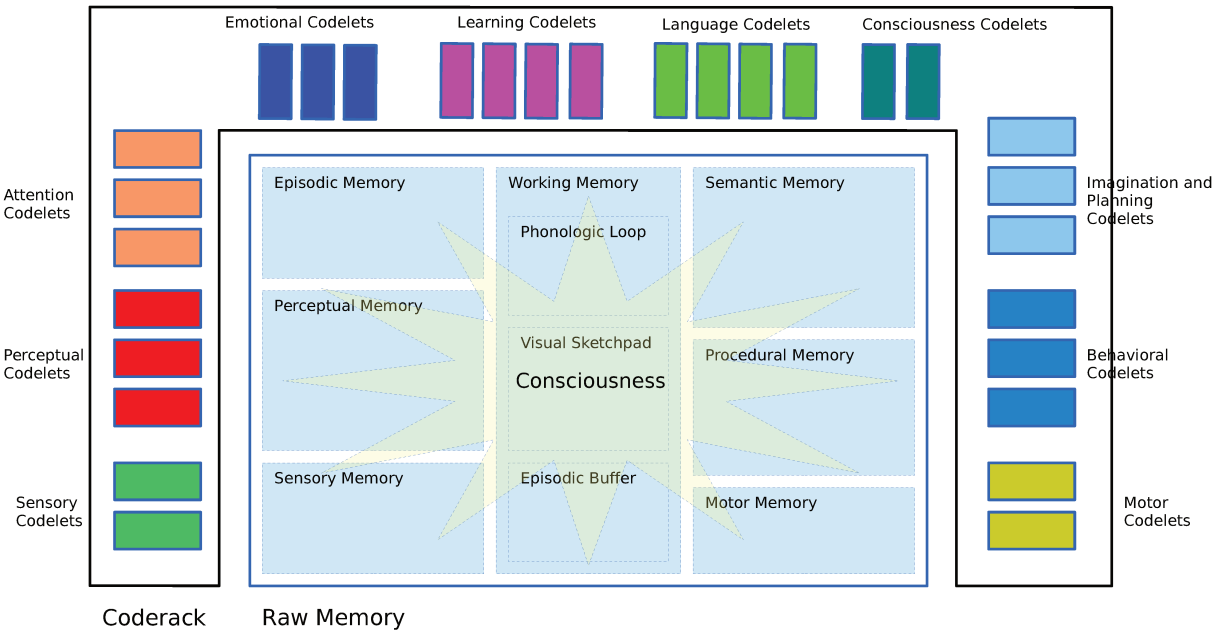


Figure 4.4: The CST Reference Architecture

We call it a Reference Architecture because it is an abstract view of how to organize a set of codelets and a set of memory objects, in order to build a cognitive architecture. Codelets are organized into groups, each group responsible for implementing a cognitive model of some cognitive function. The CST toolkit provides standard implementations for some of these groups. Other codelets will be available in the future, as CST implementation evolves. New types of codelets can be created by the cognitive architect, and easily bound together using CST core functions. Figure 4.4 is a refined view of Figure 4.2, where codelets groups are indicated. Also, memory objects are scattered among many different kinds of memories. The Raw Memory is so split into many different memory systems, which are used to store and access different kinds of knowledge. The following kinds of memories are prescribed by CST Reference Architecture:

Sensory Memory

The *Sensory Memory* is the raw storage of information coming from visual, auditive, olfactory, tactile and other sensory modalities in a time window which generally spans over something as 50-500 ms (Baddeley, 1997). This memory usually comprises uninterpreted data which is used in the first steps of perception. There are at least two kinds of sensory

memory, the *Iconic Memory*, storing visual pattern stimulus and the *Echoic Memory*, storing auditory stimulus, but there might be also other memories for other senses as well, not so widely investigated.

This memory holds Memory Objects carrying direct representations of system sensors. These Memory Objects can be simple numbers or very complex data structures, representing both scalar sensors or n-dimensional images, according to the information provided by a sensor. It might also store temporal sequences of sensor data, which can be used by Perceptual Codelets to create more elaborate percepts. More elaborated or derived representations from direct sensory capture are not stored here, but at the Perceptual Memory. The Memory Objects stored in the Sensory Memory are usually updated by Sensory Codelets.

Perceptual Memory

The *Perceptual Memory* is the memory of categories of things which can be perceived by a Perceptual System. It includes different things, attributes and patterns which can be categorized by a perceptual system. Each instance of a perceptual memory is a representation of a category used during perception.

Perceptual Memory Objects comprises usually some abstraction or high level representation of items of reality. They are usually called in cognitive science as *Percepts*. There might be many different possible ways for representing percepts, like *fuzzy sets*, patterns, objects and other more elaborate representations. Usually, the Perceptual Memory also holds representations for categories of things. Usually, Perceptual Memory is fed by Perceptual Codelets, which collect information from Sensory Memory Objects and provide high-level abstractions of sensory data in terms of percepts. Many other categories of codelets, though, may use Perceptual Memory Objects as a source of information.

Episodic Memory

The *Episodic Memory* (which should not be confused with the Episodic Buffer in Working Memory) is used to store facts particularly contextualized in time and space,

forming *Episodes* which refer to information specific to a particular location and time frame. Episodes are representations for scenes detected from environment, using a higher level abstraction of space-time. We can see an episode as a specific representation for a segment of space-time, where some specific set of objects, and their trajectory in their state space is somewhat represented.

Episodic Memory is a neurocognitive mechanism for accessing time delimited contextualized information that naturally makes part of the human process of decision making, usually enhancing the chances of a successful behavior. This assertion is supported by several human psychological researches which indicate that the knowledge of his/her personal history enhances one's person ability to accomplish several cognitive capabilities in the context of sensing, reasoning and learning (Tulving, 1991; Baddeley, 2000, 2002; Tulving, 2002; Howard et al., 2005; Cabeza et al., 2008).

Episodes may be *State-based Episodes* or *Scene-based Episodes*. State-based episodes encode the episode as sequences of an agent's states (including environmental sensed states). State-based episodes are easier to store, but more difficult to be used by higher-level cognitive functions. In most artificial systems, state-based episodes are the standard approach usually adopted, since its implementation is easier. However, the actual use of state-based episodes in sophisticated applications is difficult, implying restrictions of its applicability to specific kinds of applications. State-based episodes are raw data, as they appear in the Working Memory. There might be a better way of storing this data than just storing everything.

Scene-based Episodes encode a time-space segment as a scene. In this scene, there are objects which were consciously perceived by the agent, and an action, performed by the agent itself or other agents appearing in the scene. Scene-based Episodes can be viewed as interpreted versions of state-based episodes. They are easier to be used by high-level cognitive functions, as they already segment the scene into discrete elements, which are playing their own role in the scene dynamics. At the same time, they are more difficult to be implemented in artificial systems, because they require a process of interpretation of sensory information in order to discover the objects and actions being performed at the

environment.

In addition, episodes can also be *autobiographical* and *non-autobiographical*. Autobiographical episodes are those episodes where the agent itself is performing the action being described in the episode. On the contrary, on non-autobiographical episodes, the subject of the action is another agent. In this case, these actions are being observed by the current agent and memorized as something seen, but not done by the agent itself.

Working Memory

The *Working Memory* is a volatile kind of memory used during perception, reasoning, planning and other cognitive functions. Its capacity in time and space is very short, ranging from 4 to 9 items, and periods up to a few dozen seconds (Miller, 1956; Baddeley et al., 1975; Cowan, 2001). According to Baddeley (1997, 2000), there are at least three subsystems involved in the implementation of a Working Memory, the *Visuo-spatial Sketchpad*, the *Phonological Loop* and the *Episodic Buffer*, coordinated by a *Central Executive* which intermediates between them. Regarding brain localization, the regions related to working memory processes are very overlapping, however recent researches point the prefrontal cortex and basal ganglia as being crucial (Braver et al., 1997; Frank et al., 2001; McNab & Klingberg, 2008).

Visual Sketchpad

The Visual Sketchpad serves the function of integrating spatial, visual and possibly kinesthetic information into a unified representation which may be temporarily stored and manipulated (Baddeley, 1997). In the Visual Sketchpad we usually have Memory Objects representing maps of the environment, or more elaborate representations, where multiple *locations* are connected forming a whole network. These Memory Objects can be used by Planning codelets in order to derive a plan of actions.

Phonologic Loop

The Phonologic Loop comprises a temporary verbal-acoustic storage system which is assumed to be necessary, for example, for the immediate retention of sequences of digits, or words in a phrase (Baddeley, 1997). Despite its name inspired in speech recognition, the Phonologic Loop can store any kind of sequence detected in episodes: sequences of numbers, sequences of words, sequences of situations detected at the environment, etc.

Episodic Buffer

The Episodic Buffer is a limited capacity storage buffer which binds together information from a number of different sources into chunks or episodes, combining information from different modalities into a single multi-faceted code in order to be processed by the Central Executive (Baddeley, 2000).

The Episodic Buffer comprises the detection of episodes at the environment, as they are happening. The information on the Episodic Buffer is an abstract representation of the perceived present. The structures in the Episodic Buffer are the episodes which will later be stored in the Episodic Memory, in a sequence, forming a continuous timeline where we can recover episodes from the past.

Semantic Memory

The *Semantic Memory* is used to record facts of general kind, not contextualized in time and space. Usually, Memory Objects in the Semantic Memory are sentences: strings of characters comprising general facts.

Procedural Memory

The *Procedural Memory* is the memory of actions and behaviors of a system. It is a non-declarative memory which refers to a "how to" kind of information, usually consisting of a record of possible motor and behavioral skills. Typical examples of Memory Objects in the Procedural Memory are behavioral rules.

Motor Memory

The *Motor Memory* is a dual of the Sensory Memory, but now for the system's actuators. Memory Objects in the Motor Memory are usually actuator values, which will be used as parameters by Motor Codelets in order to actuate at the environment.

Global Workspace (Consciousness)

Finally, the Global Workspace, represented as a star labeled *Consciousness* in Figure 4.4 is a virtual kind of memory. Instead of storing its own set of Memory Objects, the Global Workspace is just a collection of references to other Memory Objects stored in the different memories described before. This is the reason it is represented as a star, because it is constantly being changed, by the consciousness mechanism implemented by the Consciousness codelets.

Using the available codelets, different cognitive architectures, using different strategies for modeling different cognitive capabilities, can be composed in order to perform the role necessary to address a specific control problem. These codelets are constructed according to different techniques in intelligent systems, like neural networks, fuzzy systems, evolutionary computation, rule-based systems, Bayesian networks, etc., which are integrated into a whole control and monitoring system.

The definition and choice of a particular cognitive architecture is constructed using a composition of different kinds of codelets, according to the control problem under analysis. Depending on the problem to be addressed, different strategies might be necessary or useful, depending on the problem constraints. Our Reference Architecture groups codelets in the following categories:

Sensory Codelets

Sensory codelets are codelets which are responsible for grabbing information from sensors at the environment, and feeding the corresponding Memory Objects which might hold the sensors values. Depending on the applications (e.g. robotic applications), sensory codelets will be really reading the sensor values and creating a corresponding representa-

tion. In other applications (as e.g. in a video-game or a virtual world), sensory codelets will open sockets to other computer applications and will simulate the acquisition of data from the environment.

Perceptual Codelets

Perceptual codelets are responsible for generating percepts, or high-level representations for things at the environment. Usually they are fed by Memory Objects at the Sensory Memory, and will feed Memory Objects at the Perception Memory.

Attention Codelets

Attention codelets are specialized kinds of codelets which will work as salience detectors for objects, situations, events or episodes happening at the environment which might be important for defining an action strategy, or behavior. Usually they grab information from Memory Objects at the Perceptual Memory, and usually feed Memory Objects at the own Perceptual Memory or the Working Memory.

Emotional Codelets

The concept of emotion, as brought from cognitive psychology and philosophy, was suggested in the literature, as an alternative way of dealing with the problem of behavior generation (Bates et al., 1994; Reilly, 1996; Picard, 1997; Canamero, 1997, 1998; Septseault & Nédélec, 2005; Budakova & Dakovski, 2006; Meyer, 2006).

There is no consensus, though, on what exactly are emotions. Different approaches have different views for what it is and how to model them. For example, Ortony et al. (1998) view emotions as "valenced reactions to events, agents, or objects, with their particular nature being determined by the way in which the eliciting situation is construed". Sloman (1998, 2001) understand emotions as internal "alarms" which give a momentary emphasis to certain groups of signals. Damasio (1994, 1999) make a distinction between "emotions", which affect the body and "feelings", which are a cognitive introspection of an emotion. Other authors may have further different views for what emotions are. For some

of them, emotions work like "amplifiers" for motivations. For others they are homeostatic processes related to physiological variables (Canamero, 1997). Some authors, instead of a single concept of emotion, develop a complex "emotional system", where many different concepts like "motivations", "drives", "impulses", "affections", "needs" and other terms are used to represent different aspects of this emotional system.

Emotional Codelets can be used to derive Cognitive Architectures modeling some of these emotional systems. Particularly, the *Activation* meta-information in codelets, and the *Evaluation* meta-information in Memory Objects can be used to implement these emotional mechanisms.

Learning Codelets

Learning is one of the most important capabilities of cognition. Different cognitive architectures report many different kinds of learning (Lucentini & Gudwin, 2015). For example, SOAR (Laird, 2012) reports 4 different kinds of learning: Chunking, Reinforcement Learning, Semantic Learning and Episodic Learning. The LIDA architecture reports 6 different kinds of learning: Perceptual Learning, Sensory Motor Learning, Spatial Learning, Episodic Learning, Procedural Learning and Attentional Learning (Franklin et al., 2014b). Clarion (Sun, 2003) also cites many different kinds of learning: bottom-up learning (RER - Rule Extraction Refinement, IRL - Independent Rule Learning), top-down learning (Learning Explicit Knowledge, Learning Implicit Knowledge), neural networks learning (Q-Learning, Backpropagation), imitative learning, etc.

Many different learning strategies can be used with CST. Even though CST does not emphasize any particular kind of learning, this learning should be provided by a learning codelet. New kinds of learning can be included, as soon as there are a specific kind of memory object which needs to be learned.

Language Codelets

Language is one of the unique capabilities of human beings, while compared to other cognitive abilities shared with other species of animals (Deacon, 1998). Recently, evidences

that there are two subsystems in the brain responsible for language were discovered (Ardila et al., 2011), one responsible for grounding the meaning of isolated symbols (or words) and the other responsible for what is called *grammatical language*. The study on the simulation of language evolution has brought the attention on the importance of *Language Games* in order to construct the meaning of language in artificial agents (Steels, 2015; Vogt, 2015).

Language codelets are responsible for implementing specific behaviors enabling the agent controlled by a cognitive architecture implemented through CST to engage into interactions to other agents, in order to play a language game. An example of such application is reported in de Paula & Gudwin (2015).

Consciousness Codelets

Even though the topic of machine consciousness is still very controversial in the community (Gamez, 2009), one of the most popular approaches involves the implementation of *Global Workspace Theory* (GWT) from Baars (1988). Baars' theory has been implemented in the LIDA cognitive architecture (Franklin et al., 2012), but also by others (Shanahan, 2006; Dubois et al., 2008). In CST, we would like to provide the cognitive architect with many different possible theories of consciousness. Due to that, the consciousness mechanism is not a built-in mechanism, but a mechanism which is implemented by means of consciousness codelets. It is true that these codelets make use of features provided by CST core, like the global input in codelets, which allow the broadcast required in GWT. The current implementation of CST provides a set of codelets which implements GWT in a way very similar to LIDA, but with some differences. In LIDA, the codelets assumed to be in a coalition are those which trigger at the same time. This is not the same in CST. In CST, codelets are assumed to be in a coalition just if they are coupled together by means of a common memory object. CST implementation of GWT also allows for subtle variations or interpretations of GTW, something which is not available in LIDA.

Imagination and Planning Codelets

Computational Imagination (Setchi et al., 2007) is also a cognitive function described in many works (Chella et al., 2005; Marques & Holland, 2009). In some sense, imagination and planning are bounded together. Imagination and Planning Codelets are codelets which implement different techniques for planning and imagination in a cognitive architecture. Alternatively, it is possible to bind standard approaches to planning, as in Prolog, or SOAR, together with other techniques using CST. In fact, any kind of rule-based system available in Java can be linked to CST and be a part of a cognitive architecture constructed with the aid of the toolkit.

Behavioral Codelets

Planning codelets might make use of different behaviors, implemented through *Behavioral Codelets*. Alternatively, Behavioral Codelets can be used to implement different kinds of action selection mechanisms. Currently, CST provides support for implementing a behavior system like in the Subsumption Architecture (Brooks, 1986, 1991) or using Maes (1989) Behavioral Networks (Tyrrell, 1994).

Motor Codelets

Motor codelets are responsible for picking up Memory Objects from the Motor Memory and transforming them into actuations at the environment. This can be done by simply capturing actuator values and feeding actuators, or by some special protocol interacting with external software or hardware. Usually, these Motor Memory Objects are generated by earlier Behavioral Codelets.

Other Codelets

Other cognitive functions are planned to be implemented in CST. Among others, we intend to extend CST by constructing codelets for implementing meta-cognitive subsystems (Cox, 2005; Sun et al., 2006; Sloman, 2011), theory-of-mind (Sodian & Kristen,

2010), motivational systems (Sun & Wilson, 2011), social cognition (Greenwald & Banaji, 1995; Gallese et al., 2004) and possibly others.

At the time of this publication, the CST had been released as an open source software on Github for only 6 months, and the following kinds of codelets were available, considering the classification in Figure 4.4:

1. Sensory Codelets: sensors are usually very specific and will probably have to be implemented in every application.
   (a) Sensor codelet based on Baars & Gage (2007) concept of Working Memory.
2. Perceptual Codelets
   (a) Perception codelet based on Baars & Gage (2007) concept of Working Memory.
3. Attention Codelets: no types available yet at the toolbox.
4. Emotional Codelets: no types available yet at the toolbox.
5. Learning Codelets
   (a) Learning codelet based on GLAS algorithm, designed and implemented in Raizer (2015) work.
6. Language Codelets: no types available yet at the toolbox.
7. Consciousness Codelets
   (a) Consciousness codelet based on Baars et al. (2013) concept of the Dynamic Global Workspace Theory, designed and implemented in this work.
8. Imagination and Planning Codelets: no types available yet at the toolbox.
9. Behavioral Codelets
   (a) Behaviour Network codelet, based on Maes (1989) work.
   (b) Subsumption architecture codelet, based on Brooks (1991) work.

10. Motor Codelets: Just like sensors, motor codelets are very specific of each application, and will likely have to be implemented for every application.

    (a) Motor codelet based on Baars & Gage (2007) concept of Working Memory.

Even though there are still some kinds of codelets which, despite specified, are not yet available at the toolbox, those available are enough to develop a working cognitive architecture able to control an artificial agent. This work contributed with the design and implementation of a particular consciousness codelet, the so called Spotlight Broadcast Controller Codelet, based on Baars et al. (2013) concept of the Dynamic Global Workspace Theory. A developer requiring this capability, only has to instantiate this codelet as an object inside coderack and the Global Workspace Algorithm (described in section 3.3) will be executed. Considering the activation calculated in the method calculateActivation() of each codelet, and the output of the winner codelet, the conscious one, will be broadcasted and available at the B input of every codelet in the Coderack. Below, we show the final version, by the time of this publication, of the consciousness codelet produced. The whole Java implementation of the CST is available at https://github.com/CST-Group/cst.

```
/*******************************************************************************
 * Copyright (c) 2012 DCA-FEEC-UNICAMP
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the GNU Lesser Public License v3
 * which accompanies this distribution, and is available at
 * http://www.gnu.org/licenses/lgpl.html
 *
 * Contributors:
 *    K. Raizer, A. L. O. Paraense, R. R. Gudwin - initial API and implementation
 ******************************************************************************/
package br.unicamp.cst.consciousness;
import java.util.ArrayList;
import java.util.List;
import br.unicamp.cst.core.entities.CodeRack;
import br.unicamp.cst.core.entities.Codelet;
import br.unicamp.cst.core.entities.MemoryObject;
import br.unicamp.cst.core.exceptions.CodeletActivationBoundsException;
/**
 * @author Andre Paraense
 *
 * A codelet-based implementation of the Global Workspace Theory, originally formulated
 *  in [1988 Baars] Bernard J. Baars. A Cognitive Theory of Consciousness. Cambridge University Press, 1988.
 *
 */
public class SpotlightBroadcastController extends Codelet
{
   private Codelet consciousCodelet;
   /** access to all codelets, so the broadcast can be made*/
   private CodeRack codeRack;
   private double thresholdActivation = 0.9d;
   public SpotlightBroadcastController(CodeRack codeRack)
```

```
{
   this.setName("SpotlightBroadcastController");
   this.codeRack = codeRack;
   consciousCodelet = null;
   this.timeStep = 3001;
}
/* (non-Javadoc)
 * @see br.unicamp.cogsys.core.entities.Codelet#accessMemoryObjects()
 */
@Override
public void accessMemoryObjects()
{
   // nothing
}
/* (non-Javadoc)
 * @see br.unicamp.cogsys.core.entities.Codelet#calculateActivation()
 */
@Override
public void calculateActivation()
{
   try
   {
      setActivation(0.0d);
   } catch (CodeletActivationBoundsException e)
   {
      e.printStackTrace();
   }
}
/* (non-Javadoc)
 * @see br.unicamp.cogsys.core.entities.Codelet#proc()
 */
@Override
public void proc()
{
   if(consciousCodelet!=null)
   {
      if(consciousCodelet.getActivation() < thresholdActivation)
      {
         consciousCodelet = null;
      }
   }
   if(codeRack!=null)
   {
      //first, select the coalition with greater activation to gain consciousness
      List<Codelet> allCodeletsList = codeRack.getAllCodelets();
      if(allCodeletsList!=null)
      {
         for (Codelet codelet: allCodeletsList)
         {
            if(consciousCodelet == null)
            {
               if(codelet.getActivation() > thresholdActivation)
               {
                  consciousCodelet = codelet;
               }
            }else
            {
               if(codelet.getActivation() > consciousCodelet.getActivation())
               {
                  consciousCodelet = codelet;
               }
            }
         }
         //then, broadcast its information to all codelets
         if(consciousCodelet!=null)
         {
```

```
            List<MemoryObject> memoryObjectsToBeBroadcasted = consciousCodelet.getOutputs();
            if(memoryObjectsToBeBroadcasted!=null)
            {
               for (Codelet codelet: allCodeletsList)
               {
                        if(!codelet.getName().equalsIgnoreCase( consciousCodelet.getName() ))
                     codelet.setBroadcast(memoryObjectsToBeBroadcasted);
                  else
                     codelet.setBroadcast(new ArrayList<MemoryObject>());
               }
            }else
            {
               for (Codelet codelet: allCodeletsList)
               {
                  codelet.setBroadcast(new ArrayList<MemoryObject>());
               }
            }
         }else
         {
            for (Codelet codelet: allCodeletsList)
            {
               codelet.setBroadcast(new ArrayList<MemoryObject>());
            }
         }
      }
   }
}
```

In Chapter 5, we present how we designed and implemented the machine consciousness urban traffic signal controller as a CST Architecture, using the new consciousness capability we implemented and made available in the toolkit as a GWT Codelet called Spotlight Broadcast Controller.

Chapter 5

Materials and Methods

After presenting the theoretical background required to build the machine consciousness urban traffic signal controller, as a CST Architecture, we now start describing our experiments in terms of materials and methods. In the next Chapter, we present the experiments results.

In section 5.1, we present the materials used in our experiments. In subsection 5.1.1, we present the traffic simulator used to run our experiments. The simulator chosen is a well known traffic simulator, widely used in the scientific community, called SUMO (Simulation of Urban Mobility). SUMO uses the microscopic approach to traffic simulation, described in section 2.1.3. In subsection 5.1.2 we present the test bed used to run the experiments, composed of five different urban network models. Closing the materials section, in subsection 5.1.3, we present the CST Architecture we built, in order to control the traffic signals in the different network models that compose our test bed. Using the CST, we built two different traffic controllers, one which is an adaptive controller, considering only local conditions around its junction, and the machine consciousness controller, which considers not only the local conditions of its junction but also the global broadcasted information from the conscious junction. We also used a third controller, given by SUMO, which uses a fixed time strategy.

In section 5.2, we present the methods used to test our hypothesis, that is, how we planned the experiments ran with the materials we designed and implemented, in order to validate our scientific hypothesis.

5.1 Materials

5.1.1 The SUMO Traffic Simulator

In this work, we used the SUMO (Simulation of Urban MObility) traffic simulator (Krajzewicz et al., 2012), a computational environment using microscopic simulation of vehicles in order to run our experiments. SUMO is a free and open traffic simulation suite which is available since 2001. It allows modelling of intermodal traffic systems including road vehicles, public transportation and pedestrians.

The SUMO simulation platform offers many features:

- Microscopic simulation - vehicles, pedestrians and public transportation are modeled explicitly;
- Online interaction – control of the simulation items, including traffic signals, through the TraCI interface. TraCI is the short term for "Traffic Control Interface". Giving the access to a running road traffic simulation, it allows to retrieve values of simulated objects and to manipulate their behaviour "on-line" in a SUMO instance. TraCI uses a TCP based client/server architecture to provide access to SUMO. Thereby, SUMO acts as server that is started with additional command-line options: –remote-port <INT> where <INT> is the port SUMO will listen on for incoming connections. The simulation can be started, stopped and advanced step by step. While the simulation is running, much information can be retrieved, both static (e.g. the road network topology) and dynamic (e.g. position and speed of vehicles);
- Time schedules of traffic lights can be imported or generated automatically by SUMO;
- No artificial limitations in network size and number of simulated vehicles;
- Supported map import formats: OpenStreetMap, VISUM, VISSIM, NavTeq.

In this work, we used the tool NETCONVERT, which is part of the SUMO package, to import network models in the OpenStreetMaps format and online interaction through the TraCI API to remotely control the time schedules of traffic lights during simulations.

An example of a linux command line to generate the network model file from the OpenStreetMaps exported file is:

```
$ netconvert --osm-files downtownCampinas.osm.xml -o downtownCampinas.net.xml --tls.join --ramps.guess --ignore-errors --no-turnarounds
```

where:

- *downtownCampinas.osm.xml* - the OpenStreetMaps exported map file
- *downtownCampinas.net.xml* - the network model file to be generated

In order to generate vehicle routes, we used two other auxiliary tools, also part of the SUMO package. The first one is the *randomTrips.py*, a python program which takes the network model as an input and generates random trips for each vehicle. The second one is the *duarouter*, which takes both the network, generated by NETCONVERT, and the random trips, generated by *duarouter* as inputs, and generates all the routes for all the vehicles. In section 5.2, we explain in more details the variables involved in these routes generation, such as time window and density of vehicles in different scenarios.

An example of linux command lines to generate the routes is:

```
$ python randomTrips.py -n manhattan.net.xml -e 5400 -l -L -p 0.1 -s 1000 --fringe-factor 10 -o manhattan.p0.1.1.trips.xml

$ duarouter -n manhattan.net.xml -t manhattan.p0.1.1.trips.xml -o manhattan.p0.1.1.rou.xml --ignore-errors --random
```

where:

- *manhattan.net.xml* - the network model
- 5400 - time window for generating the trips in seconds
- 1000 - the seed for the random generator
- *manhattan.p*0.1.1.*trips.xml* - the trips file to be generated
- *manhattan.p*0.1.1.*rou.xml* - the routes file to be generated

Finally, SUMO runs taking as inputs the network model, generated by NETCONVERT, and the routes, generated by *duarouter*. With an option set, the simulator will write to a file a summary output, containing all the information it generates in each step, such as mean travel time, cars ended and mean waiting time. An example of a linux command line to run a simulation would be:

```
$ sumo -n manhattan.net.xml -r manhattan.p0.1.1.rou.xml --summary-output output.sumo.manhattan.p0.1.1.consc.xml --remote-port 9000
```

where:

- *manhattan.net.xml* - the network model
- *manhattan.p*0.1.1.*rou.xml* - the vehicle routes file
- *output.sumo.manhattan.p*0.1.1.*consc.xml* - the summary output file to be produced
- 9000 - the port to which the simulator will listen for the TraCI interactions

The simulator will append to the output file in batches, usually at each 50 steps of the simulation. At the end of the simulation, it is possible to use the auxiliary program *plot_summary.py* to plot graphs of any of the measurements present in the output summary file over time. For instance, if we want to plot the graph of the mean travel time of the vehicles over time, we could do so by running a linux command such as:

```
$ python plot_summary.py -i output.sumo.manhattan.p0.1.1.consc.xml --labels "Machine Consciousness" -o
      output.sumo.manhattan.p0.1.1.consc.xml.pdf -m meanTravelTime --xlim 0,57600 --ylim 0,10000 --yticks 0,10001,2000,14 --xticks
      0,57601,14400,14 --xtime1 --ygrid --ylabel "Mean Travel Time (s)" --xlabel "time (h)" --adjust .14,.1
```

where:

- *output.sumo.manhattan.p*0.1.1.*consc.xml* - the summary output file produced
- *MachineConsciousness* - the curve label
- *output.sumo.manhattan.p*0.1.1.*consc.xml.pdf* - the plot figure to be generated as a PDF file
- *meanTravelTime* - the measurement present in the output file to be plotted
- 0, 57600 - the x-axis limits

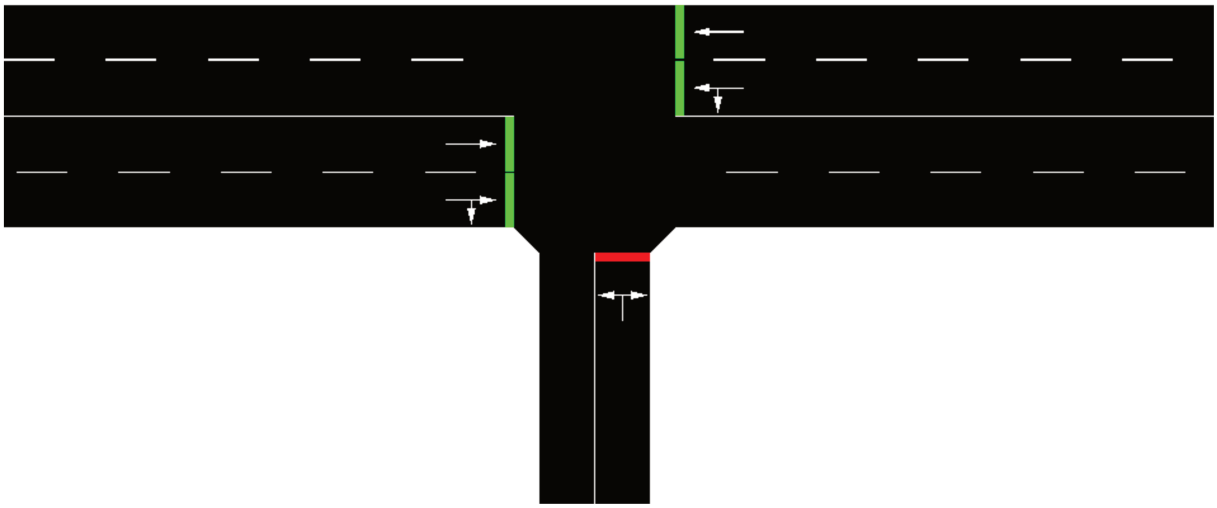


Figure 5.1: Network model “Simple T”, where the experiments took place.

- $0, 10000$ - the y-axis limits
- $0, 57601, 14400, 14$ - the x-axis ticks
- $0, 10001, 2000, 14$ - the y-axis ticks
- $MeanTravelTime(s)$ - the y-axis title
- $time(h)$ - the x-axis title

Indeed, we have used the auxiliary program *plot_summary.py* to plot the graphs later presented in Chapter 6 as the experiments results.

5.1.2 Network Models - Test Bed

Five network models were tested in this work.

The first one is called “Simple T”, which is composed by a single junction linking three main roads, as shown in Figure 5.1.

The second one is called “Twin T”, that stands for the two junctions linking four main roads, as shown in Figure 5.2.

Figure 5.2: Network model “Twin T”, where the experiments took place.

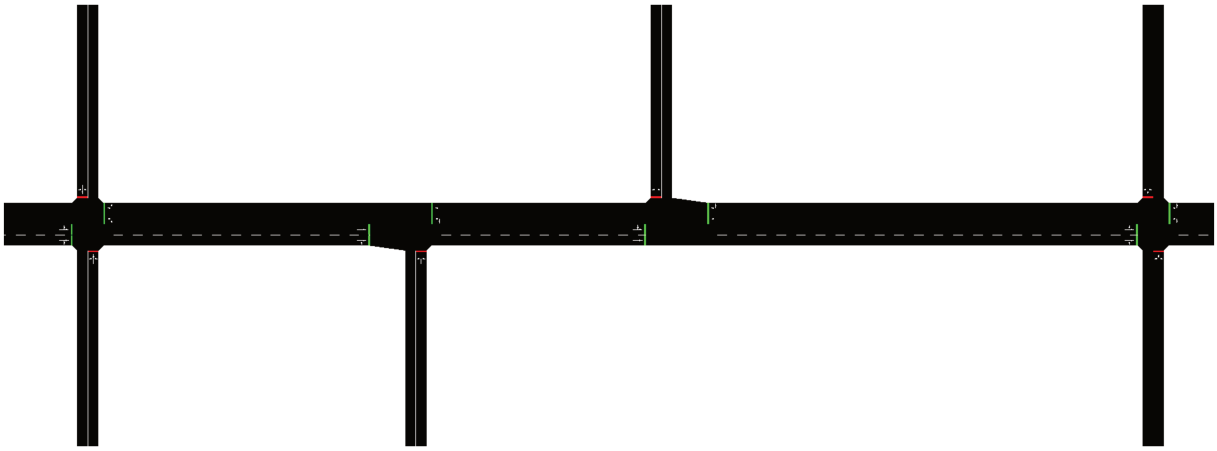


Figure 5.3: Network model “Corridor”, where the experiments took place.

The third model, called “Corridor”, is a little more complex than the “Twin T”, being composed by four junctions connecting many roads, as shown in Figure 5.3.

The fourth model is even more complex, called “Manhattan”, a grid composed by 9 junctions, as shown in Figure 5.4.

The fifth model, called “Downtown Campinas”, is the most complex and the only real network model, representing one whole region of the city of Campinas. This network is composed by 109 junctions, as shown in Figure 5.5.

5.1.3 Our CST Architecture - The Traffic Controllers

We tested three different traffic controllers in our experiments.

The first one, called “Fixed Times”, is the simplest and most common traffic controller. It has a fixed traffic signal plan, independently from the traffic in its controlled lanes,

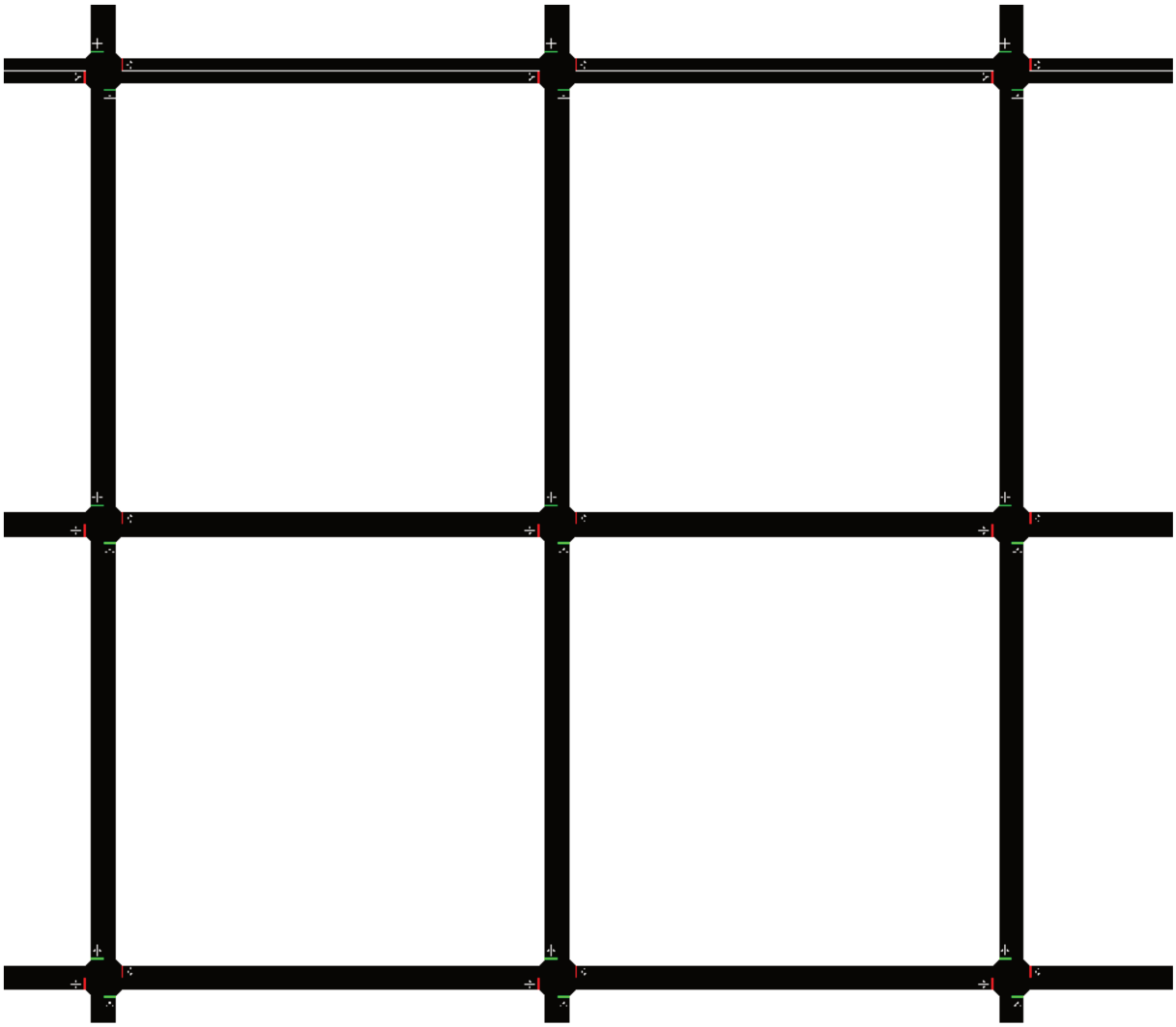


Figure 5.4: Network model “Manhattan”, where the experiments took place.

changing the traffic lights in regular fixed periods called “phases”. As this one is a trivial traffic controller, we did not have to design and implement it, we just used the one provided by SUMO. In this case, the traffic light plans are generated in the absence of any traffic demand information. Hence, no optimizations are undertaken[1]. The idea is just to have the results of this controller to be used as a baseline or reference.

The second and third ones, called “Parallel Reactive” and “Machine Consciousness”, respectively, were designed and implemented using CST. Both of them rely in sensory information, from vehicles traveling in the crossing lanes, in order to choose one phase among n possible phases, with red and green lights for each lane in the junction. These

[1]There was no documentation of SUMO’s heuristic to generate this fixed times plan by the time of this publication, only the code could be found at https://svn.code.sf.net/p/sumo/code/trunk/sumo/src/netbuild/NBOwnTLDef.cpp.

Figure 5.5: Network model “Downtown Campinas”, where the experiments took place.

are the two controllers we want to compare, in order to study the influence of the machine consciousness mechanism.

In order to explain the modeling of the traffic controllers we implemented, we use the simplified network model of Figure 5.6, which is a zoom of the first two junctions from left to right in Figure 5.3, and then explain how this modeling extends to any situation.

In Figure 5.6, we detail the many sensed regions, represented by letters (a) to (k), and the different possible phases, drawn on top right, which can be chosen by the controller in these junctions.

The controllers built with CST can be modeled for this simplified situation as shown in Figure 5.7.

There are four types of codelets in our model:

- Sensory Codelets: for each set of inductive sensors in the region in front of the traffic light, there is one codelet whose output is the pair of vectors X, holding the

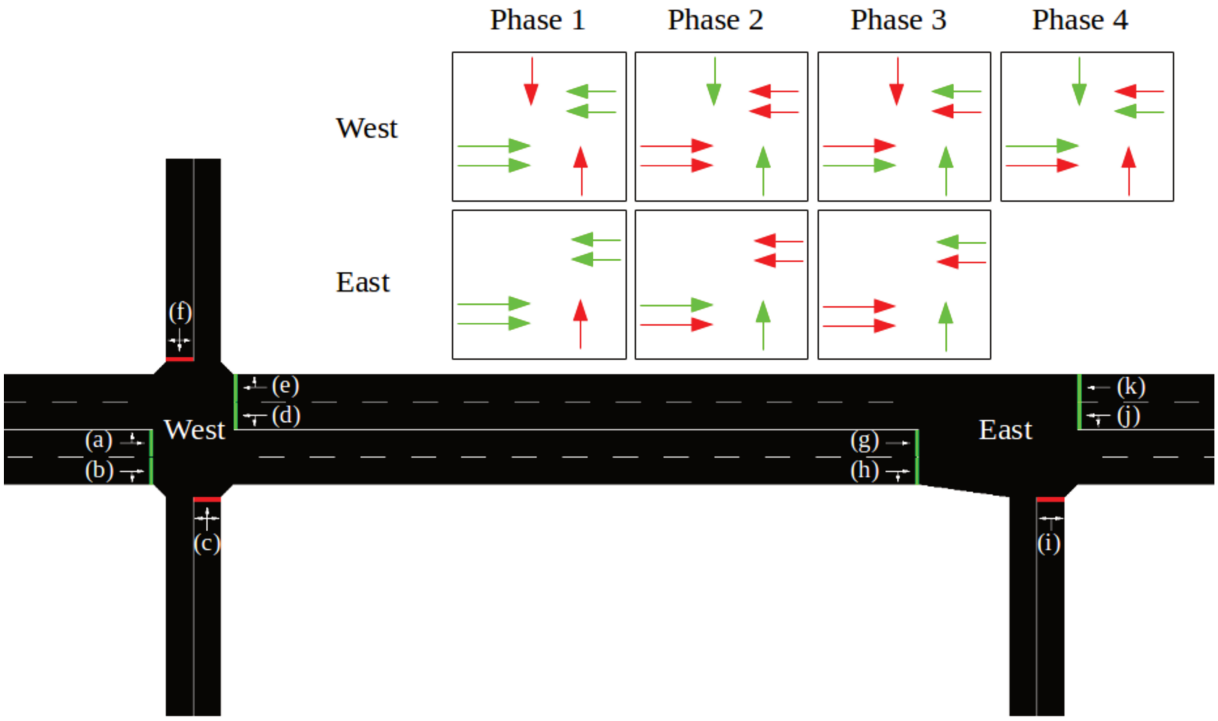


Figure 5.6: Network model of the sensed regions in front of the traffic lights, represented by letters (a) to (k), and the different possibles phases which can be chosen by the controller, drawn on top right, considering junction West and junction East.

position of all vehicles in that region, and V, holding the velocities of each one of the vehicles.

- Behavioral Codelets: for each junction there is one codelet whose inputs are positions and velocities for the vehicles in front of the traffic lights (plus the broadcast from consciousness), and output is one of the traffic lights possible phases for the junction.

- Consciousness Codelet: responsible for selecting the sensory content to gain consciousness and for the broadcast of conscious content among all codelets.

- Motor codelets: for each junction, there is one codelet whose input is a chosen phase for the junction traffic lights and the codelet is responsible for implementing that phase in the real system.

The "Parallel Reactive" controller is automatic or "unconscious", in our analogy to the animal brain. Each sensory codelet captures the vehicles position and velocity in its respective lane continuously, while one behavioral codelet for each junction performs the

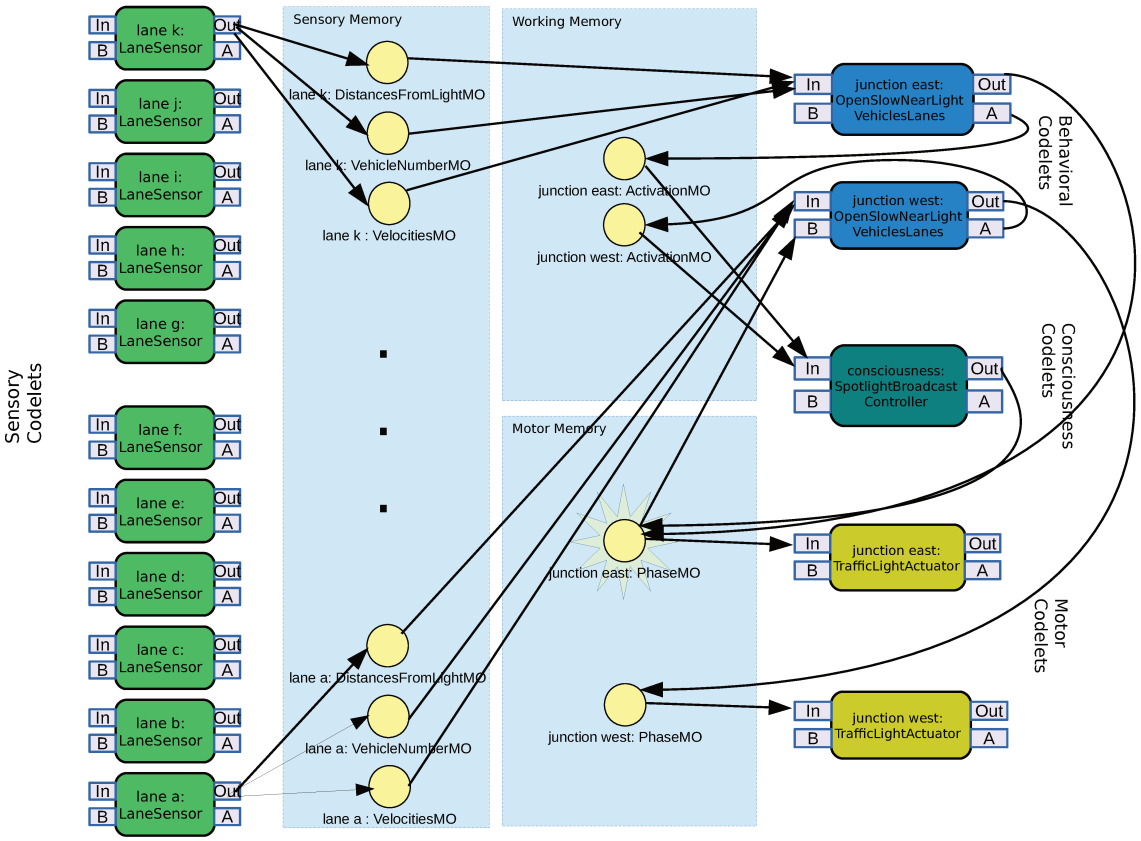


Figure 5.7: Model of the controllers built on top of CST for the simplified Figure 5.6 network model. Four types of codelets were used. Three types were implemented for this specific application - a sensory codelet, a behavioral codelet and a motor codelet. Also, the consciousness codelet provided by CST (also a contribution of this work) was used in the model. In the example shown above, junction East is chosen by the consciousness codelet to gain the spotlight. The consciousness codelet does not change the information of the memory object of the junction East in the motor memory, but just highlights it as the conscious content. The output of the behavioral codelet junction East is then broadcasted to the global input B of the behavioral codelet junction West.

following algorithm, later formally expressed in Algorithm 1:

1. Calculates its level of activation, which is given by equation 5.1, taken from Box & Waterson (2013) work.

$$a(t) = \frac{\sum_{c \in C}(1 - \alpha V_c(t) - \beta X_c(t))}{|C|} \tag{5.1}$$

2. Determines the best phase among the possible ones based on a simple calculation as shown in Table 5.1.

3. Goes back to 1.

In equation 5.1, C is the set of all vehicles monitored by the inductive sensors in one junction; V_c is the vehicle's velocity and X_c is the vehicle's distance to the junction; α and β are constants that can be tuned to adjust the influence in the junction's codelet activation, based on the number of vehicles, velocities and distances from the given junction. The closer to the junction and the slower the vehicles, the greater the codelet's activation representing the junction. According to Box & Waterson (2013), $\alpha = 0.01sm^{-1}$ and $\beta = 0.001m^{-1}$ are values which result in a balance between these influences close to real data, and were, therefore, used in this work.

Finally, based on the chosen phase for each junction, the respective motor codelet modifies the junction traffic lights and the cycle is repeated. In the "Parallel Reactive" case, there is no Consciousness Codelet, and each junction codelet decides its phase based solely on the information they receive from their respective sensory codelets.

In the case of the "Machine Consciousness" Controller, the only difference is the presence of the Consciousness codelet, which does the following, later expressed in Algorithm 2:

1. Defines the junction codelet with greater activation level, which gains access to conscious global workspace while respecting a minimum threshold. If none of the codelets reaches the threshold, the system works unconsciously and global workspace remains empty.

Algorithm 1 Parallel Reactive behavioral codelet

```
1: procedure ACTIVATION(lane)
2:     alpha ← 0.01
3:     beta ← 0.001
4:     vehicleNumber ← lane.getVehicleNumber()
5:     laneActivation ← 0
6:     if vehicleNumber > 0 then
7:         for i = 0; i < vehicleNumber do
8:             at ← 1 − alpha ∗ lane.V[i] − beta ∗ lane.X[i]      ▷ X - array distances
9:             laneActivation ← laneActivation + at               ▷ V - array velocities
10:            i ← i + 1
11:    return laneActivation
12: procedure FINDBESTPHASE(possiblePhases, controlledIncomingLanes)
13:    bestPhase ← −1
14:    bestPhaseValue ← Integer.MIN_VALUE
15:    for i = 0; i < possiblePhases.size() do
16:        p ← possiblePhases[i]
17:        phaseValue ← 0
18:        for j = 0; j < p.lightStates.size() do
19:            ls ← p.lightStates[j]
20:            if ls = GREEN then
21:                cil ← controlledIncomingLanes[j]
22:                phaseValue ← phaseValue + activation(cil)
23:            j ← j + 1
24:        if phaseValue > bestPhaseValue then
25:            bestPhaseValue ← phaseValue
26:            bestPhase ← bestPhase
27:        i ← i + 1
28:    return bestPhase
```

Table 5.1: Action selection in the *Junction East* codelet. In this example, phase number 3 was selected because it gives the best sum of green lanes activations.

1. Heuristic activation for each lane	2. Overall Junction East activation	3. Determination of the best phase - sum of activations of green traffic lights
$AT(l) = \sum_{c \in C}(1 - \alpha V_c(t) - \beta X_c(t))$ (5.2)	$ATJ(j) = \frac{\sum_{i=1 \to n} AT(i)}{n}$ (5.3)	$P_a = \sum_{tl \in G} AT(tl)$ (5.4)
AT(g) = 0.09 AT(h) = 0.3 AT(i) = 0.85 AT(j) = 0.05 AT(k) = 0.13	ATJe = 0.858	$P_1(G, G, R, G, G) = 0.57$ $P_2(G, R, G, R, R) = 0.94$ $P_3(R, R, G, R, G) = 0.98$

2. Broadcasts the sensory information of the conscious codelet.

3. Goes back to 1.

Algorithm 2 Machine Consciousness codelet

```
1:  procedure PROC(consciousCodelet, allCodeletsList)
2:      thresholdActivation ← 0.9
3:      if consciousCodelet then
4:          if consciousCodelet.getActivation() < thresholdActivation then
5:              consciousCodelet ← NULL
6:      for i = 0; i < allCodeletsList.size() do
7:          codelet ← allCodeletsList[i]
8:          if consciousCodelet == NULL then
9:              if codelet.getActivation() > thresholdActivation then
10:                 consciousCodelet ← codelet
11:         Else
12:         if codelet.getActivation() > consciousCodelet.getActivation() then
13:             consciousCodelet ← codelet
14:         i ← i + 1
15:     if consciousCodelet ≠ NULL then
16:         for j = 0; j < allCodeletsList.size() do
17:             codelet ← allCodeletsList[j]
18:             codelet.Broadcast ← consciousCodelet.getOutputs()
19:             j ← j + 1
```

Broadcast information contains details about how critical the situation is in the worst junction in the network controlled lanes. Other junctions receiving the broadcast will decide whether or not to use this information to choose its next phase based on the network topology by following two simple rules, later expressed in Algorithm 3:

1. If one incoming lane of my junction is topologically connected to one incoming lane[2] of the conscious junction that has a red light in its chosen phase, I must close it with a red light.

2. If one incoming lane of my junction is topologically connected to one incoming lane of the conscious junction that has a green light in its chosen phase, or if it is not connected at all, I must open it with a green light.

[2]It is one incoming lane of the first junction topologically connected to another incoming lane of the second junction considering that the path will go through the first junction.

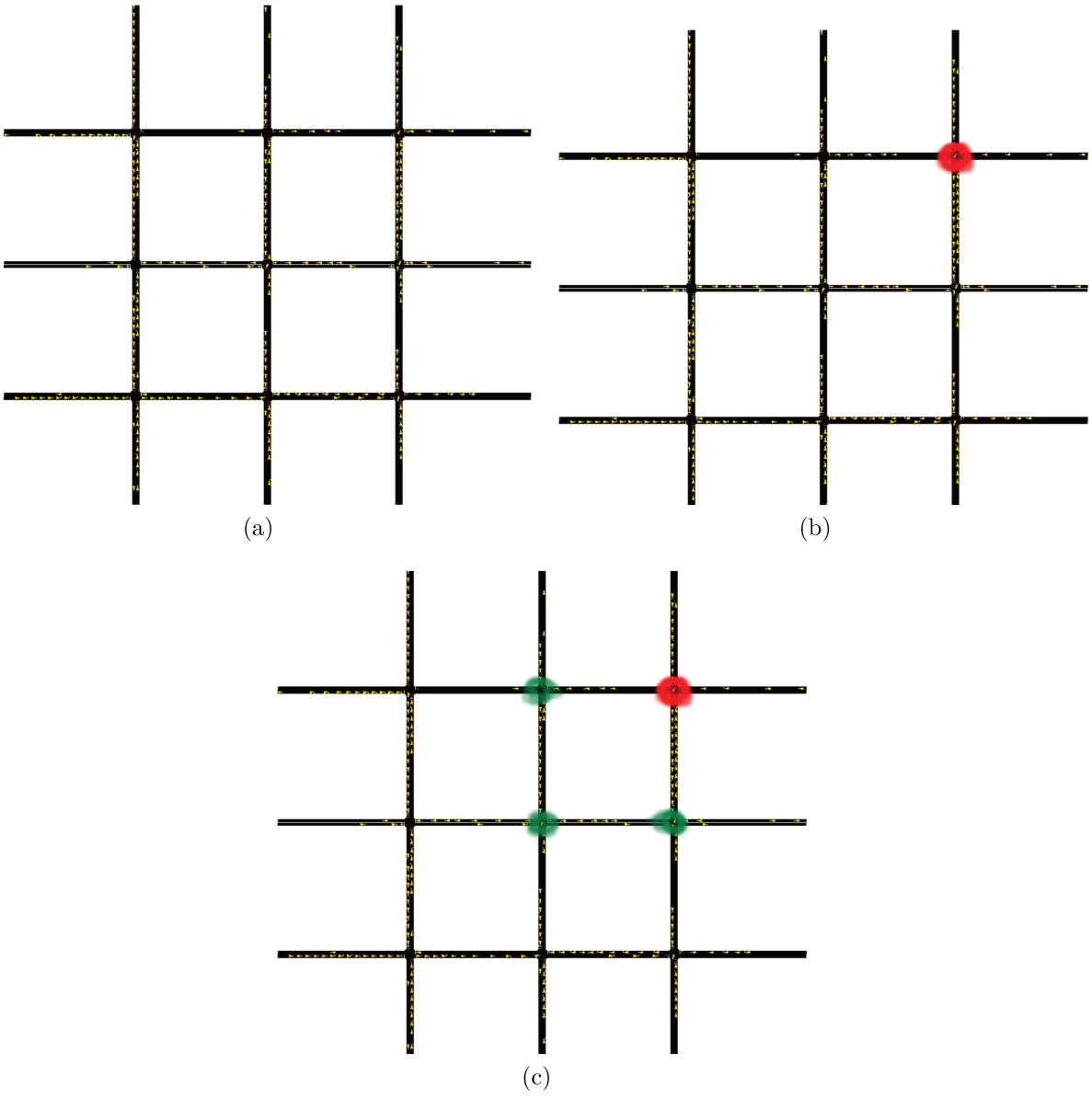


Figure 5.8: Conscious broadcast and interference mechanism. Panel 5.8a shows the parallel controllers acting. In panel 5.8b, the consciousness codelet finds the most critical junction and broadcast the information of its output memory objects (that is, the chosen phase) to all other junctions. Panel 5.8c illustrates an example of which junctions might decide to cooperate with the most critical junction in order to solve the problem it is facing.

Algorithm 3 Broadcast interference rules

```
1: procedure BROADCASTINTERF(controlledLanes, consciousControlledLanes )
2:     for i = 0; i < controlledLanes.size() do
3:         cl ← controlledLanes[i]
4:         if isConnected(cl, consciousControlledLanes) then
5:             if consciousControlledLanes[i].TLS == RED then
6:                 cl.TLS ← RED                          ▷ TLS - Traffic Light State
7:             Else
8:             cl.TLS ← GREEN
9:         Else
10:        cl.TLS ← GREEN
11:        i ← i + 1
```

The hypothesis is that these two simple rules should generate a behaviour similar to dynamic green waves in the network whenever there is a critical situation, helping to solve the conflict as soon as possible, alleviating the situation until the flow becomes normal again.

For the sake of clarity, the example given in this section modeled only two of the four junctions in Figure 5.3. Nevertheless, the controller has the ability to model any urban network given in a digital format that can be read by SUMO, such as the Open Street Maps ".OSM" format (Haklay & Weber, 2008), for instance. In order to do so, in the beginning of the simulation, the controller reads the network model and creates the corresponding codelets: one behavioral codelet for each junction in the model, one motor codelet for each junction, one sensory codelet for each lane controlled by each junction and one singleton consciousness codelet. These codelets are also attached to their corresponding memory objects, following the same architecture principles as the ones given in this section example: outputs of sensory codelets are attached as inputs to behavioral codelets, whose outputs are attached as inputs to motor codelets and the broadcast of the consciousness codelet is attached as an input to the behavioral codelets. The behavioral codelets keep a memory of the topology of the network model surrounding them (not represented in Figure 5.7), within a predefined radius (in this work we used 1 km), so they can answer whether they are connected or not to the conscious junction, which is important in order to interpret the broadcast information and decide whether or not to act upon it, just like Baars' theater audience decide if they want to interact in

the play going on the stage, based on the information broadcasted and in their internal values.

Once again, in order to dig deeper in the details, we invite you to visit our Java implementation of the CST Architecture described, made available as open source software at https://github.com/CST-Group/traffic-signal-control-app. Figure 5.9 shows an UML representation of the three codelets specifically implemented for this application.

In order to represent how a codelet is implemented, we show the code of the simplest one of these three codelets, the motor codelet, below. The whole Java implementation of the Machine Consciousness CST Architecture built to control the traffic signals is available at https://github.com/CST-Group/traffic-signal-control-app.

```
/*******************************************************************************
 * Copyright (c) 2016 DCA-FEEC-UNICAMP
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Apache License, Version 2.0
 * which accompanies this distribution, and is available at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Contributors:
 *    A. L. O. Paraense, R. R. Gudwin - initial implementation
 *******************************************************************************/
package br.unicamp.cst.trafficUnjammer.codeRack.motorCodelets;
import it.polito.appeal.traci.ChangeLightsStateQuery;
import it.polito.appeal.traci.TLState;
import it.polito.appeal.traci.TrafficLight;
import java.io.IOException;
import java.util.List;
import br.unicamp.cst.core.entities.Codelet;
import br.unicamp.cst.core.entities.MemoryObject;
import br.unicamp.cst.core.exceptions.CodeletActivationBoundsException;
import br.unicamp.cst.trafficUnjammer.rawMemory.MemoryObjectTypesTrafficLightController;
/**
 * @author andre
 *
 */
public class TrafficLightActuator extends Codelet
{
   private TrafficLight trafficLight;
   private List<TLState> trafficLightPhases;
   private MemoryObject phaseMO;
   private MemoryObject forcedPhaseMO;
   public TrafficLightActuator(TrafficLight trafficLight, List<TLState> TLStates)
   {
      this.trafficLight = trafficLight;
      this.trafficLightPhases = TLStates;
   }
   /* (non-Javadoc)
    * @see br.unicamp.cogsys.core.entities.Codelet#accessMemoryObjects()
    */
   @Override
   public void accessMemoryObjects()
   {
      int index=0;
      if(phaseMO==null)
         phaseMO = this.getInput( MemoryObjectTypesTrafficLightController.PHASE, index);
```

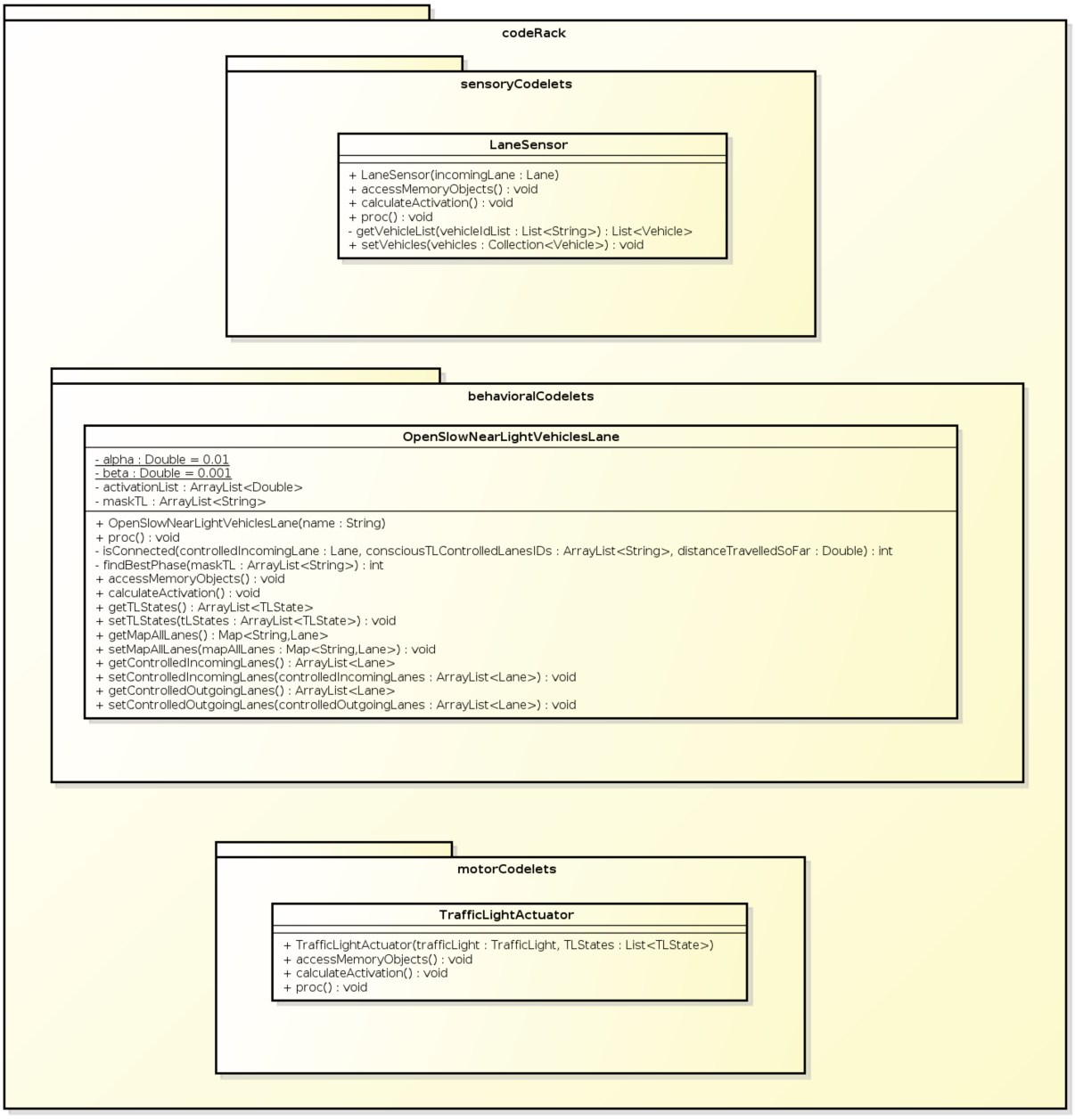


Figure 5.9: Our Java implementation of the codelets implemented in the Machine Consciousness CST Architecture built to control the traffic signals.

```
   if(forcedPhaseMO==null)
      forcedPhaseMO = this.getInput( MemoryObjectTypesTrafficLightController.FORCED_PHASE, index);
}
/* (non-Javadoc)
 * @see br.unicamp.cogsys.core.entities.Codelet#calculateActivation()
 */
@Override
public void calculateActivation()
{
   try
   {
      setActivation(0.0d);
   } catch (CodeletActivationBoundsException e)
   {
      e.printStackTrace();
   }
}
/* (non-Javadoc)
 * @see br.unicamp.cogsys.core.entities.Codelet#proc()
 */
@Override
public void proc()
{
   if(forcedPhaseMO!=null && forcedPhaseMO.getI()!=null && !( (String) forcedPhaseMO.getI()).equalsIgnoreCase("-1"))
   {
      try
      {
         TLState forcedPhase = new TLState((String) forcedPhaseMO.getI());
         ChangeLightsStateQuery lstQ = trafficLight.queryChangeLightsState();
         lstQ.setValue(forcedPhase);
         try
         {
            lstQ.run();
         } catch (IOException e)
         {
            e.printStackTrace();
         }
      }catch(Exception e)
      {
         e.printStackTrace();
      }
   }else
   {
      int phaseIndex = -1;
      if(phaseMO.getI()!=null)
      {
         try
         {
            phaseIndex = Integer.valueOf((String) phaseMO.getI());
         }catch(Exception e)
         {
            e.printStackTrace();
         }
      }
      if(phaseIndex >= 0 && trafficLight != null && trafficLightPhases != null && trafficLightPhases.size() > phaseIndex)
      {
         ChangeLightsStateQuery lstQ = trafficLight.queryChangeLightsState();
         lstQ.setValue(trafficLightPhases.get(phaseIndex));
         try
         {
            lstQ.run();
         } catch (IOException e)
         {
            e.printStackTrace();
         }
      }
```

```
    }
  }
}
```

5.2 Methods

5.2.1 Simulation Scenarios

For each one of the five network models, four simulated scenarios were considered. The first scenario, called "P = 0.1", generates vehicles with random routes in a 0.1 second period during a time window of 5,400 seconds, generating a very high concentration of vehicles coming from different sources of the network model and flowing to different edges. The edges probability of being assigned as the destination of one vehicle trip is weighted by length and number of lanes, so that larger avenues get more cars. The random routes generated are not related or representative of real traffic in principle. The objective of the experiments is to generate different traffic loads and critical traffic situations, so the machine consciousness mechanism can be observed in action. The second scenario, called "P = 0.4" has the same attributes of the first one except for the vehicle generation period, which is 0.4 second, generating a lower traffic outcome. The same method is applied for "P = 0.7" and "P = 1.0". This gives a combination of 20 experimental scenarios, which were run 10 times each, summing up a total of 200 experiments. In each one of these experiments, the simulation was run for each one of three different controllers. Each car is assigned to a random predefined route, containing all the steps it will take from its source, which can be anywhere in the network, to its destination, which can also be anywhere in the network.

Each simulated experiment is run for as long as necessary, until all vehicles generated during the first 5,400 seconds reach their final destination. During the simulation, the mean travel time and the end of the trip of all vehicles are measured and later plotted over time to compare the performance of the three controllers.

5.2.2 Controlled Experiments

The choice of five network models and four concentration scenarios of simulation was intended to provide control groups for the algorithms and controllers being tested, considering the scientific hypothesis we want to validate. For instance, in the case of the Simple T model, as there is only one junction, there is no use for the conscious broadcast, since there are no other junctions to receive it. Hence, it is expected that both controllers, parallel reactive and machine conscious ones, would behave in the same way. In this sense, "Simple T" represents a control group. The same analogy can be applied to scenarios where "P = 1.0" in the network models because, with lower traffic, codelets' activations will rarely reach the conscious threshold, and the parallel reactive and machine consciousness controllers should have similar performances, also representing some sort of control group to evaluate the algorithms' outputs.

In Chapter 6, we will present the results we reached for the aforementioned experiments and our interpretation of the validation of our initial hypothesis.

Chapter 6

Results

After presenting the materials and methods we used to test and validate our scientific hypothesis, it is now time to present the results of our controlled experiments and our interpretation of them. Since we stated our scientific hypothesis back on Chapter 1, it is now appropriate to reinforce our memory of it, before seeing and analyzing the results:

> "The scientific hypothesis of this work is that an artificial mechanism, inspired on some properties and models of consciousness, can bring advantages to automatic processes, such as urban traffic lights control."

The Chapter is organized as follows: the first five sections are dedicated to presenting and analyzing the results of each one of the network models, considering all scenarios and traffic controllers, as explained in section 5.2 - Simple T model in section 6.1, Twin T model in section 6.2, Corridor model in section 6.3, Manhattan model in section 6.4 and Downtown Campinas model in section 6.5. We close the Chapter in section 6.6 explaining how to access the raw data results of the experiments.

6.1 Simple T Model

Figures 6.1 and 6.2 show the results of four simulation experiments considering scenario P = 0.1, presenting mean travel time and end of the trip of all vehicles over time, respectively.

Figures 6.3 and 6.4 show the results of four simulation experiments considering scenario P = 0.4, presenting mean travel time and end of the trip of all vehicles over time, respectively.

Figures 6.5 and 6.6 show the results of four simulation experiments considering scenario P = 0.7, presenting mean travel time and end of the trip of all vehicles over time,

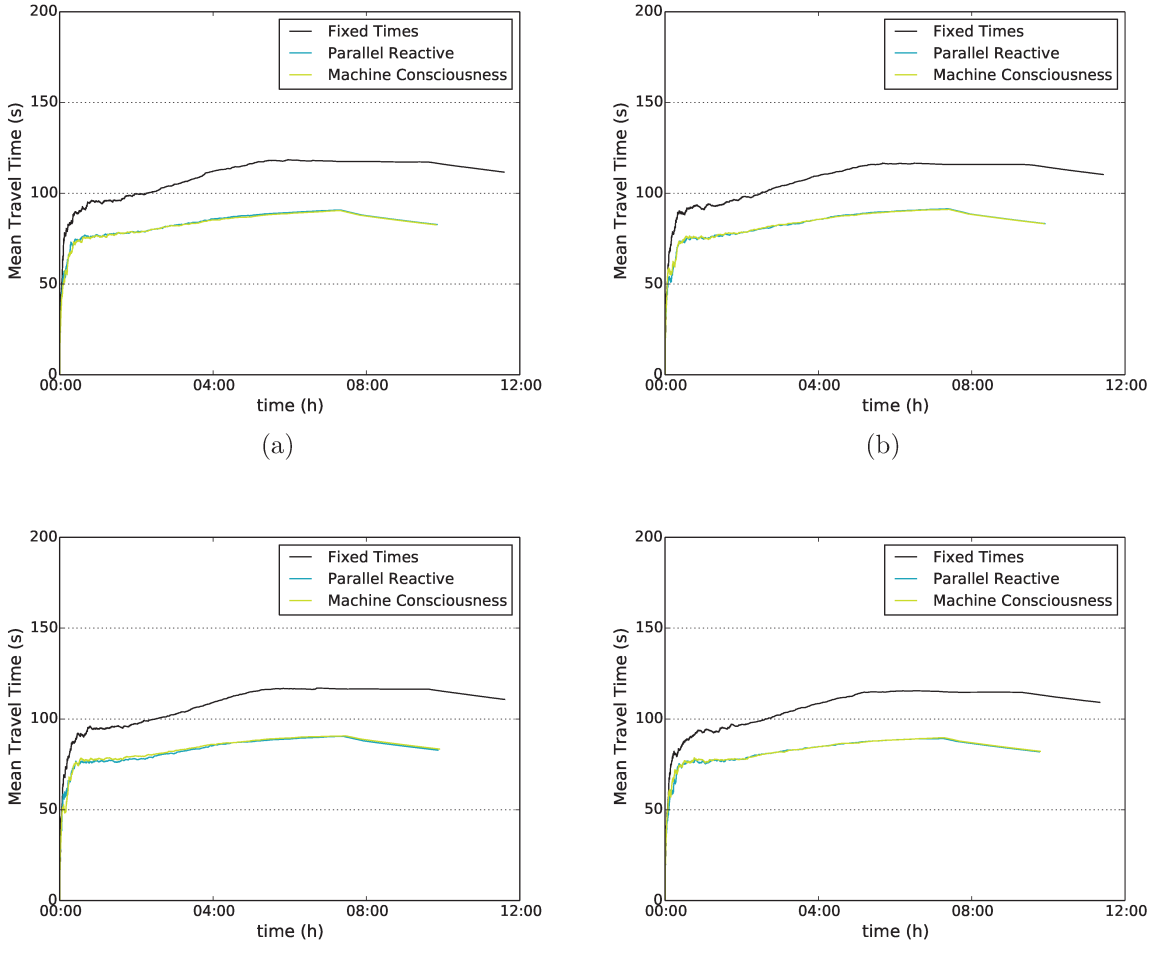
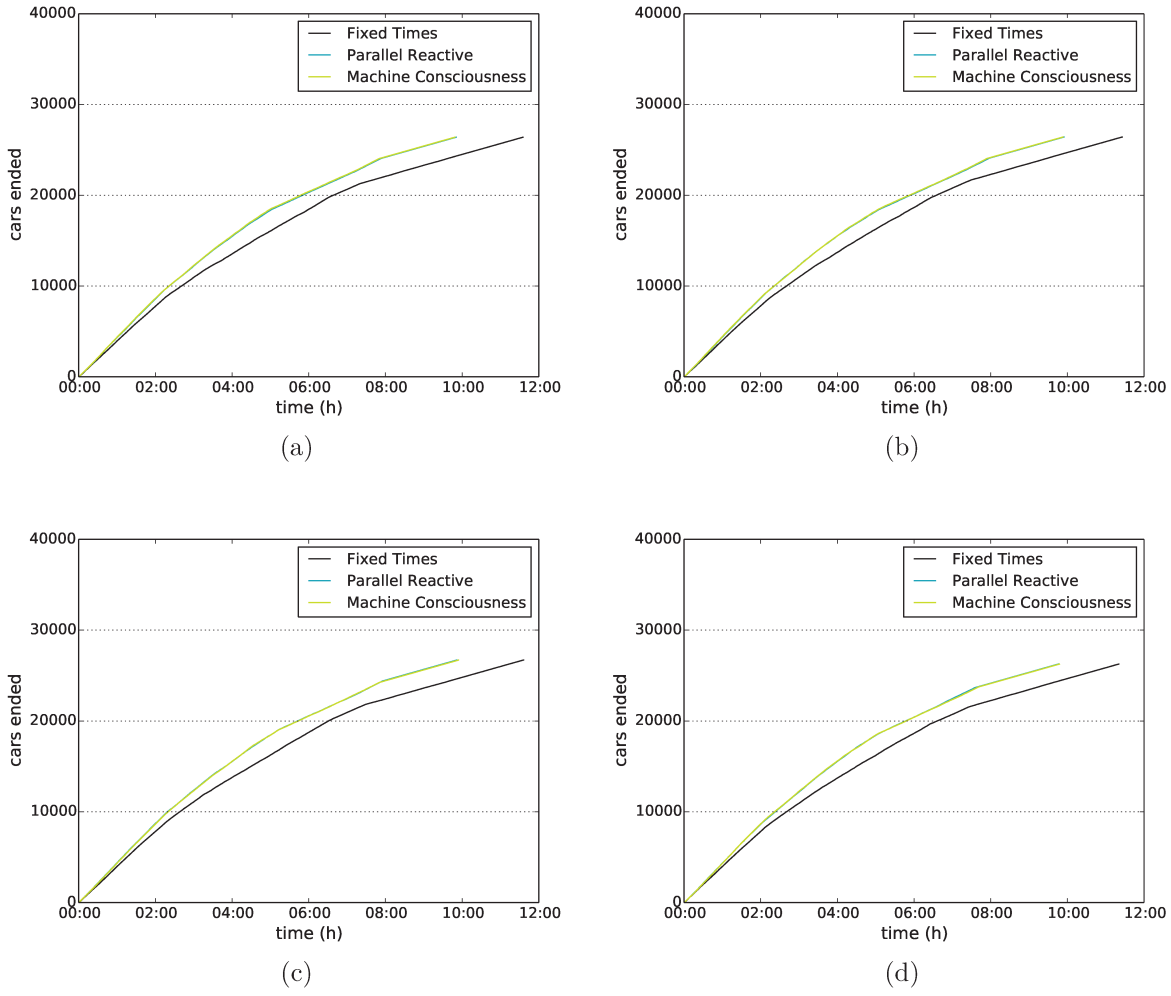


Figure 6.1: Simple T Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.1a, 6.1b, 6.1c and 6.1d represent four distinct simulation experiments.



(a) (b)

(c) (d)

Figure 6.2: Simple T Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.2a, 6.2b, 6.2c and 6.2d represent four distinct simulation experiments.

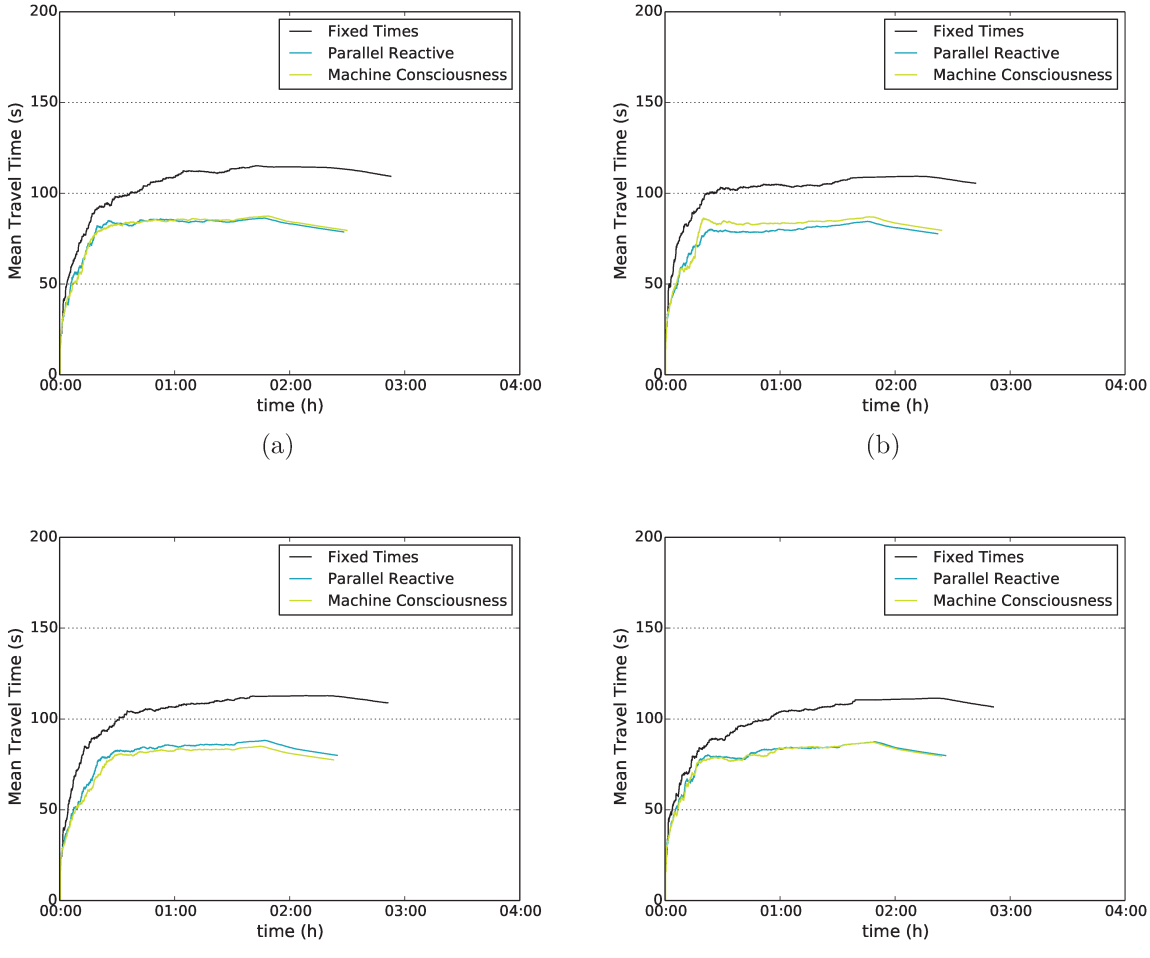


Figure 6.3: Simple T Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.3a, 6.3b, 6.3c and 6.3d represent four distinct simulation experiments.

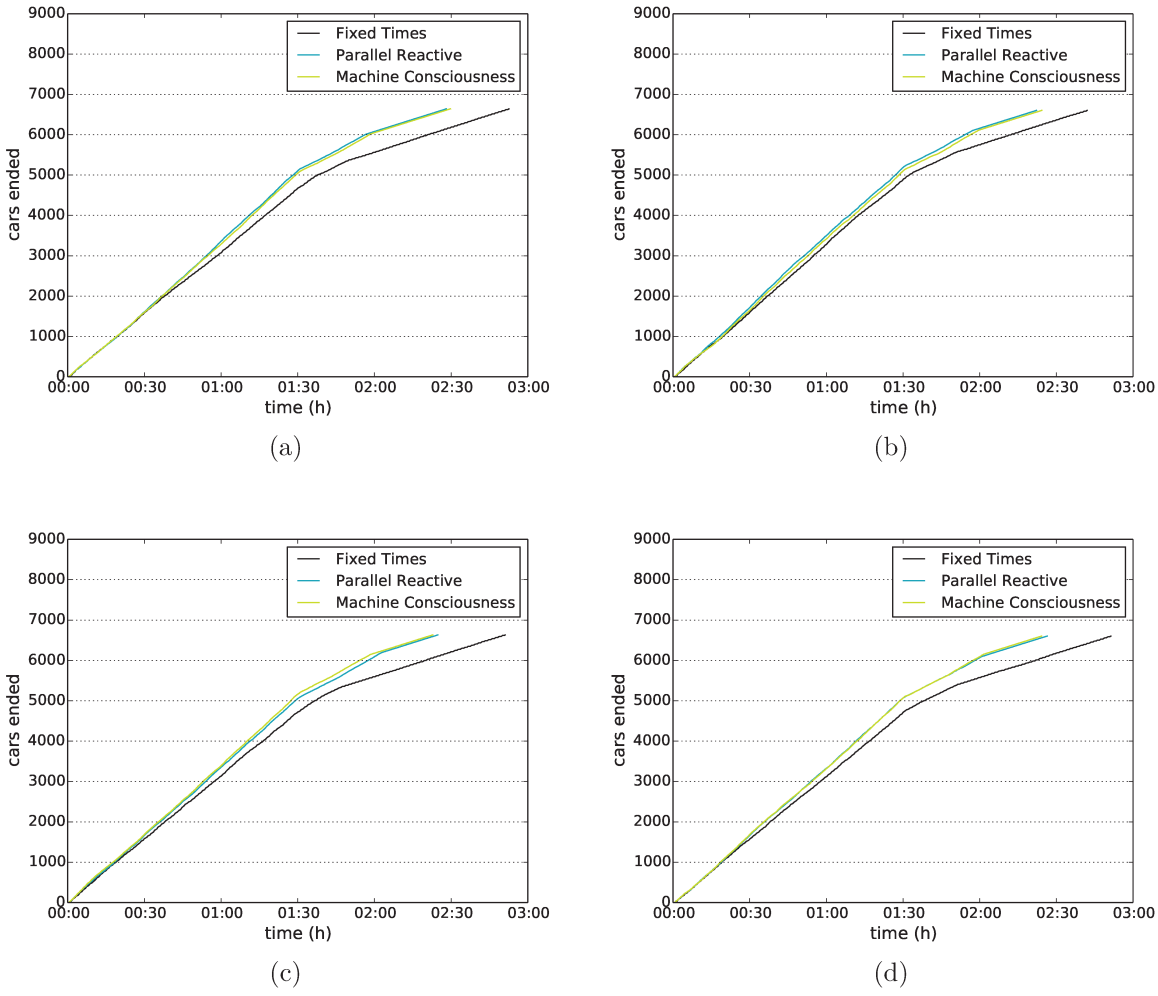


Figure 6.4: Simple T Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.4a, 6.4b, 6.4c and 6.4d represent four distinct simulation experiments.

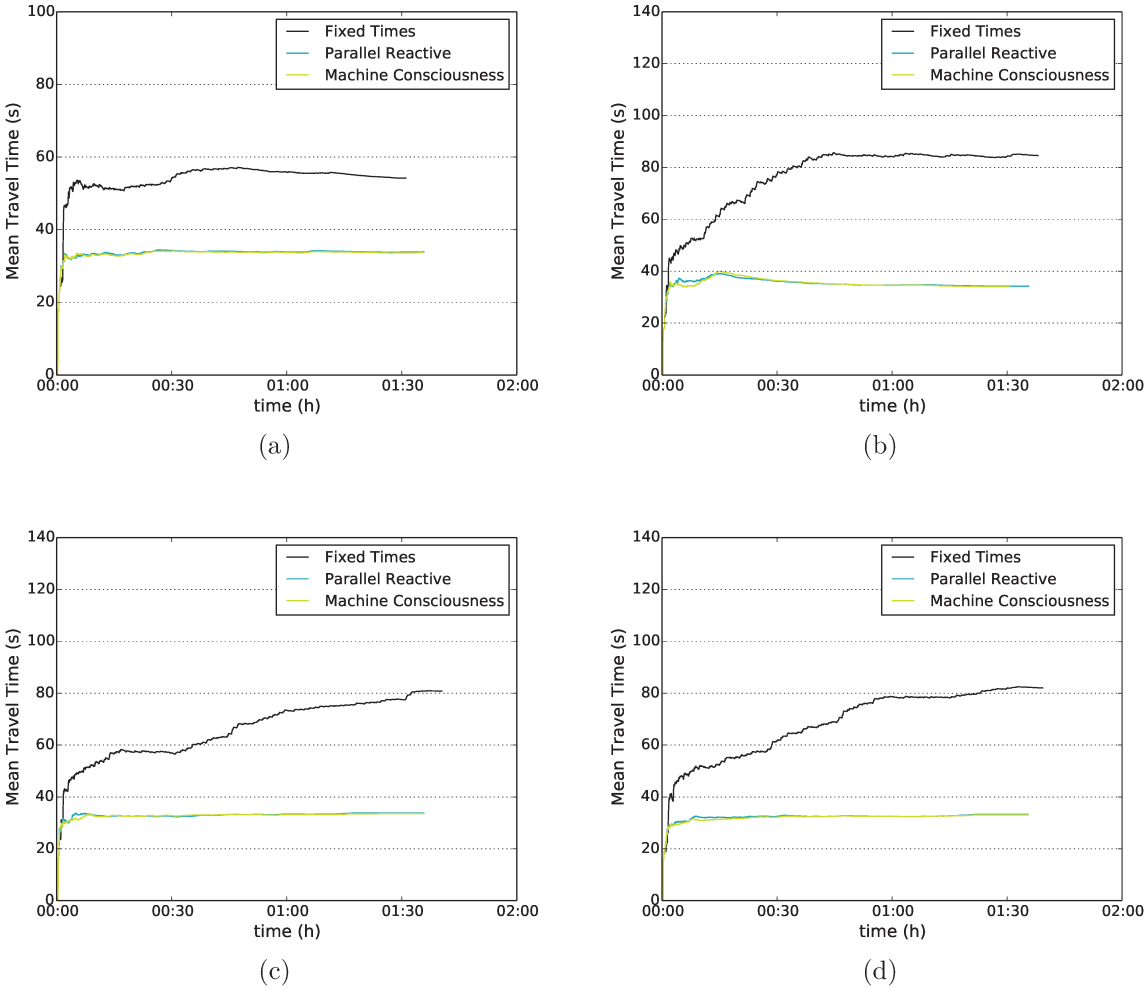


Figure 6.5: Simple T Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.5a, 6.5b, 6.5c and 6.5d represent four distinct simulation experiments.

respectively.

Figures 6.7 and 6.8 show the results of four simulation experiments considering scenario P = 1.0, presenting mean travel time and end of the trip of all vehicles over time, respectively.

In the Simple T model experiments, we found the same results for the Machine Consciousness and Parallel Reactive controllers, which were always better than the Fixed Times controller. This is in line with our predictions and expectations, because as the Simple T model has only one junction, there is no use for the GWT broadcast, since there is no other junction to receive the information of the most critical junction and take advantage of it. This model is therefore a control group for our experiments and the results

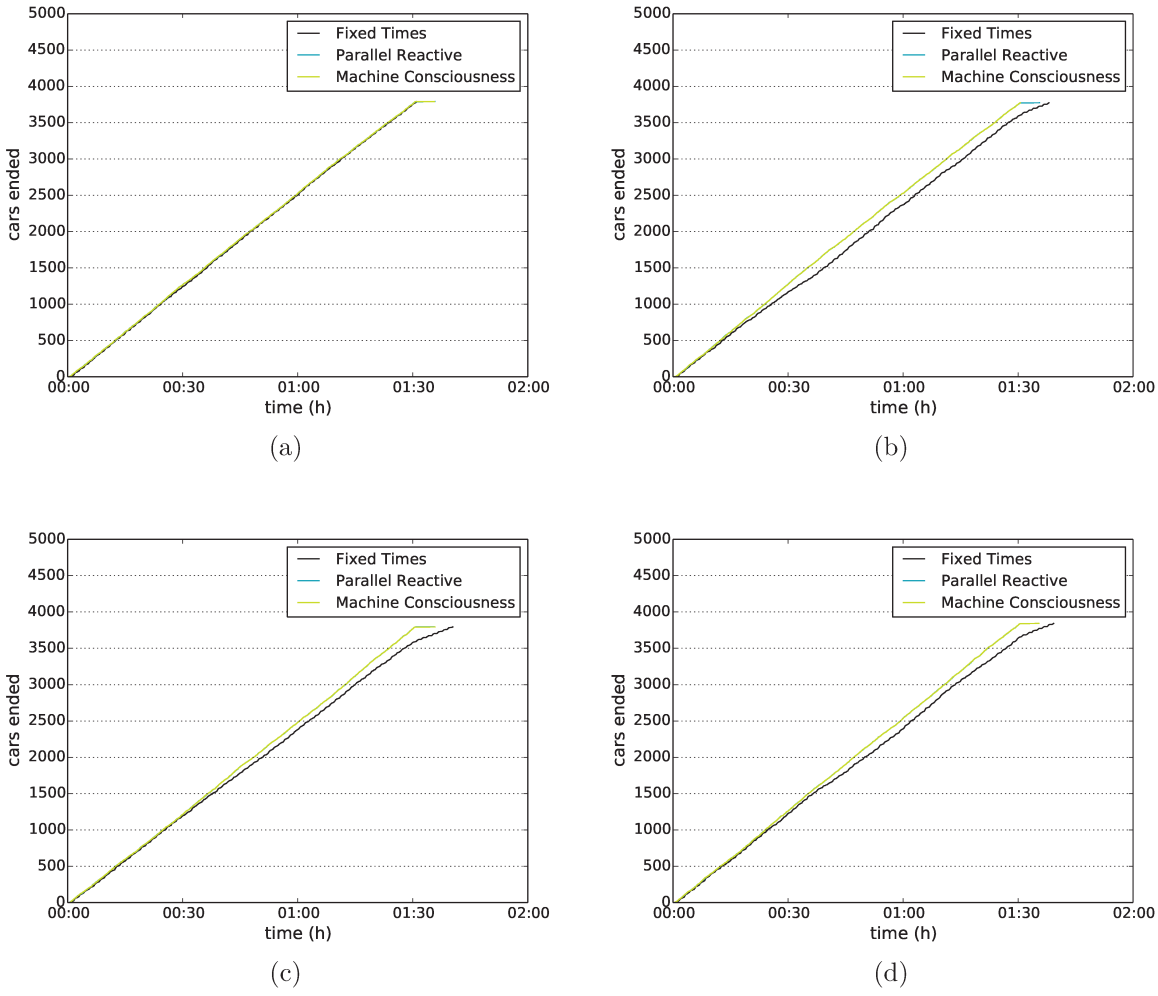


Figure 6.6: Simple T Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.6a, 6.6b, 6.6c and 6.6d represent four distinct simulation experiments.

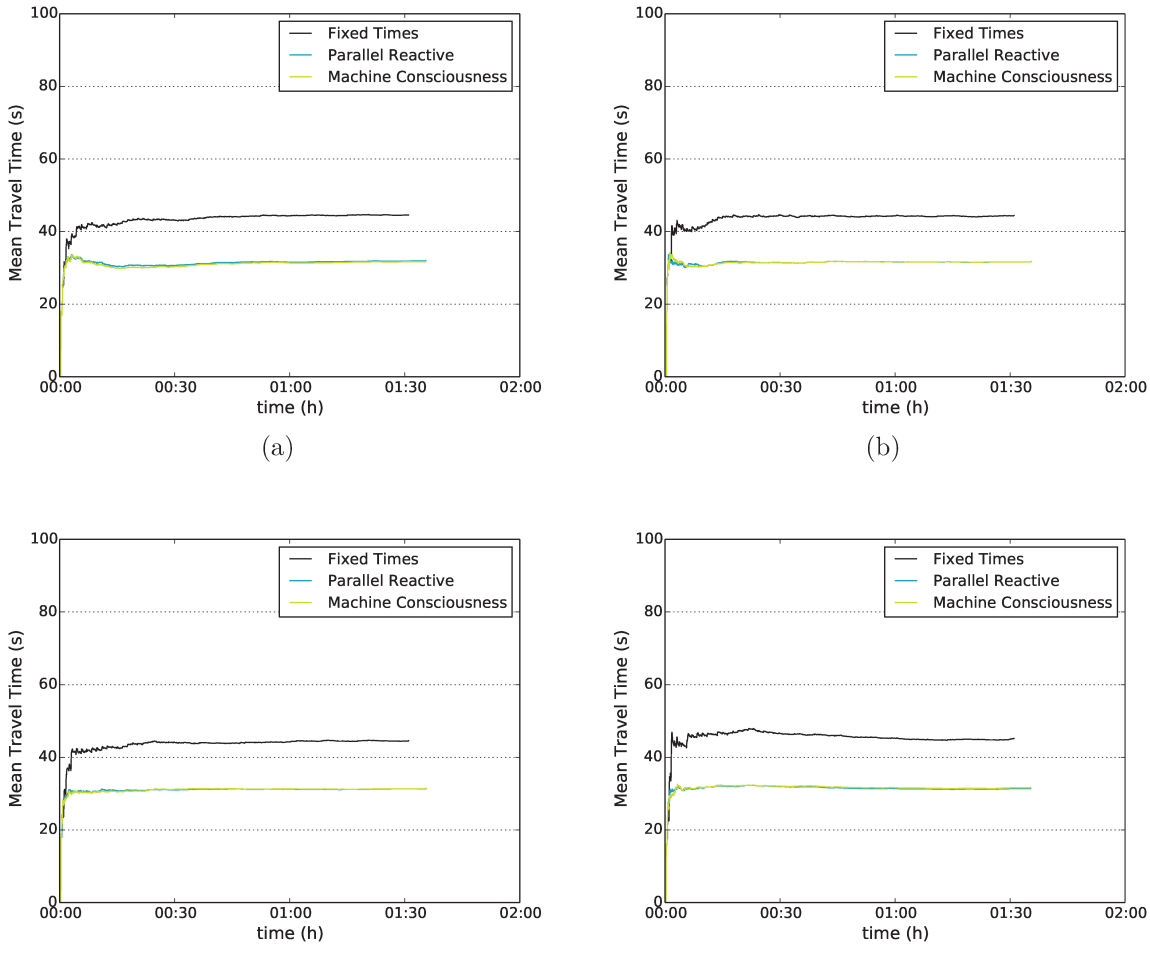


Figure 6.7: Simple T Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.7a, 6.7b, 6.7c and 6.7d represent four distinct simulation experiments.

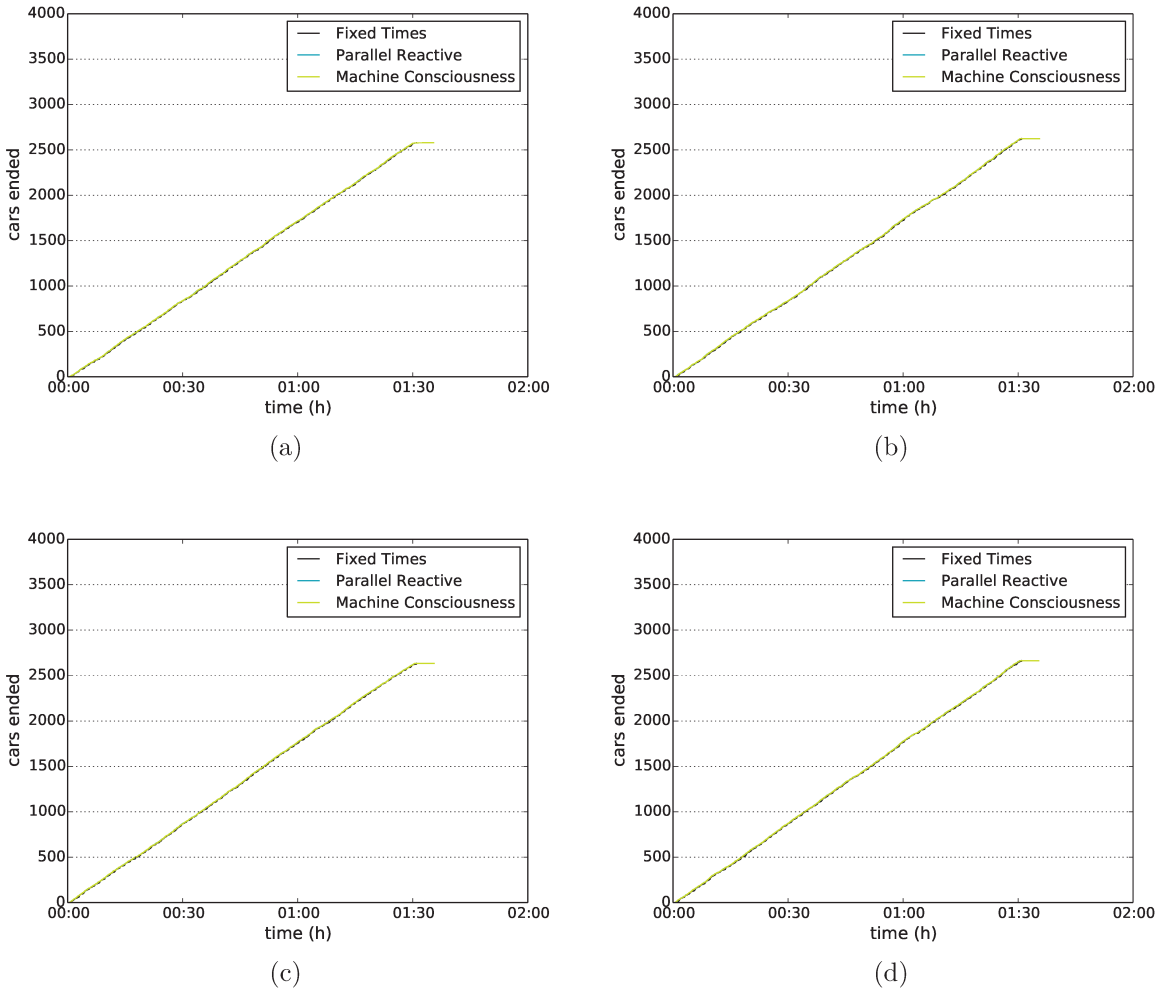


Figure 6.8: Simple T Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.8a, 6.8b, 6.8c and 6.8d represent four distinct simulation experiments.

were consistent with this proposition. In the P=0.1 scenario, the most crowded, we found gains of around 35% in the vehicle's mean travel time, while in the P=1.0 scenario, the least crowded, we found gains of around 25%.

6.2 Twin T Model

Figures 6.9 and 6.10 show the results of four simulation experiments considering scenario P = 0.1, presenting mean travel time and end of the trip of all vehicles over time, respectively. Figures 6.9 and 6.10 represent one of the most interesting results of our experiments. The network model now has two junctions, configuring a system which can take advantage of the machine consciousness mechanism. In the most crowded scenario, P = 0.1, we observe a very interesting output - the Parallel Reactive controller performance is worse than the Fixed Times during most of the time, although ending better. This is actually one of the few scenarios where we observe this kind of output. The explanation for this is the fact that we observe two very close junctions, as can be seen in Figure 5.2, a situation that makes them highly coupled, suffering with a heavy load of traffic, and making decision by themselves, ignoring each other's output, even though their decisions highly affect the inputs of the other. Because the consciousness spotlight mechanism chooses the junction in the worst condition and broadcast this information to the other one, which in turn adapts to help the worst one, we do not find the same bad results in the Machine Consciousness controller, which performs better always. We found gains in the vehicle's mean travel time of around 22%, compared to the Fixed Times, and around 13%, compared to the Parallel Reactive.

Figures 6.11 and 6.12 show the results of four simulation experiments considering scenario P = 0.4, presenting mean travel time and end of the trip of all vehicles over time, respectively. In this scenario, as the traffic load is reduced, we do not observe the same poor performance of the Parallel Reactive controller, as we did in the P=0.1 scenario, but in some occasions its performance gets closer to the Fixed Times, and is always worse than the Machine Consciousness controller. It is important to remember that the Parallel Reactive controller runs the same CST Architecture of the Machine

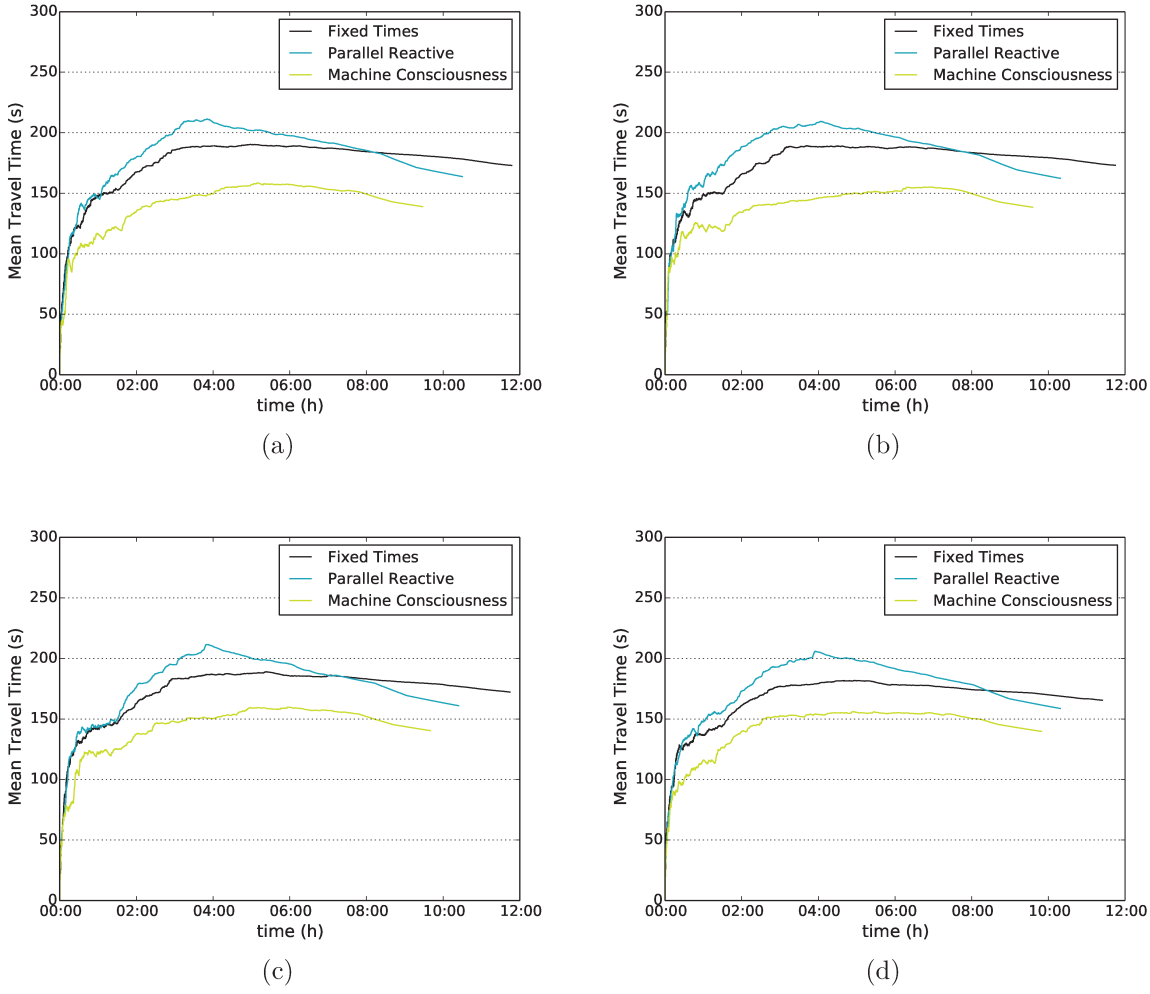
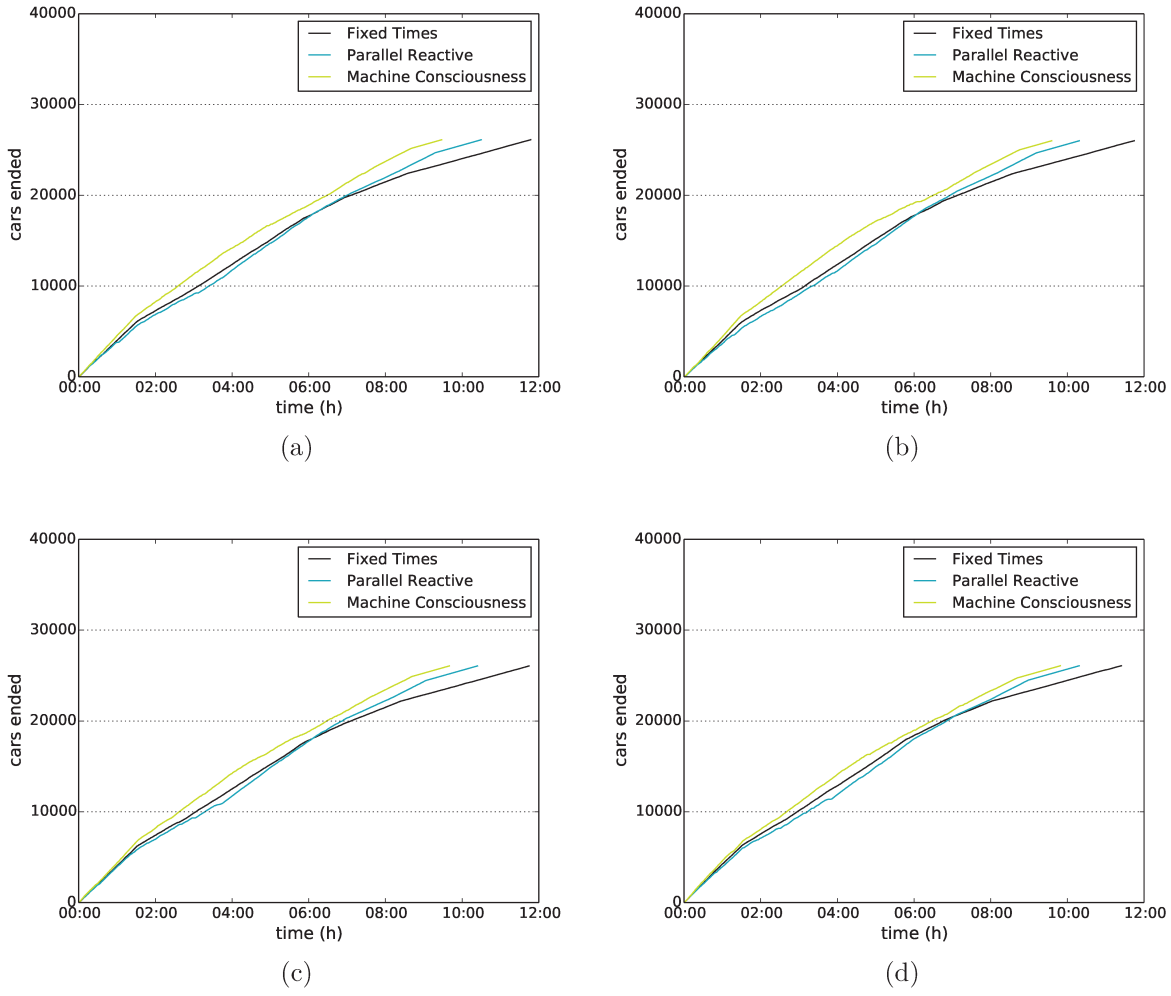


Figure 6.9: Twin T Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.9a, 6.9b, 6.9c and 6.9d represent four distinct simulation experiments.



(a) (b)

(c) (d)

Figure 6.10: Twin T Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.10a, 6.10b, 6.10c and 6.10d represent four distinct simulation experiments.

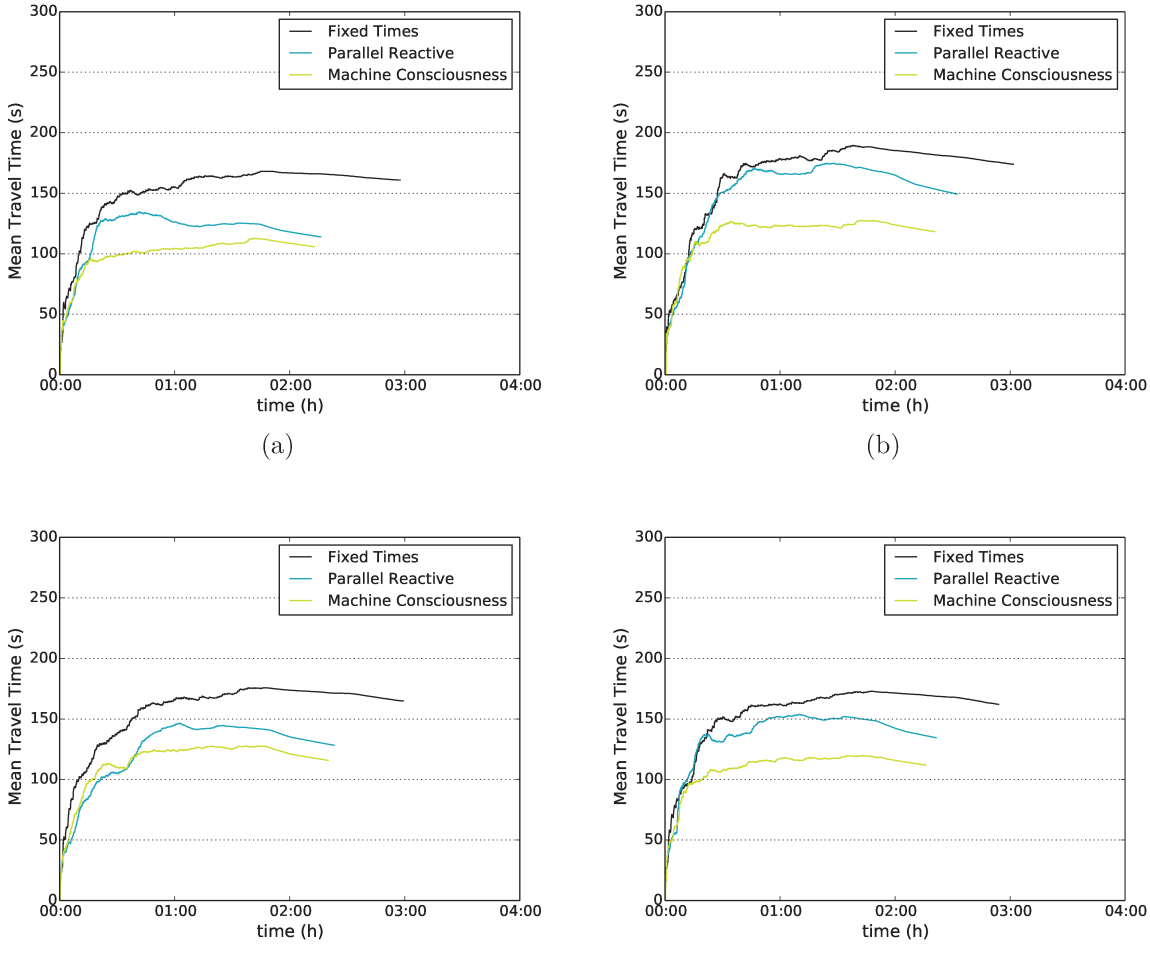


Figure 6.11: Twin T Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.11a, 6.11b, 6.11c and 6.11d represent four distinct simulation experiments.

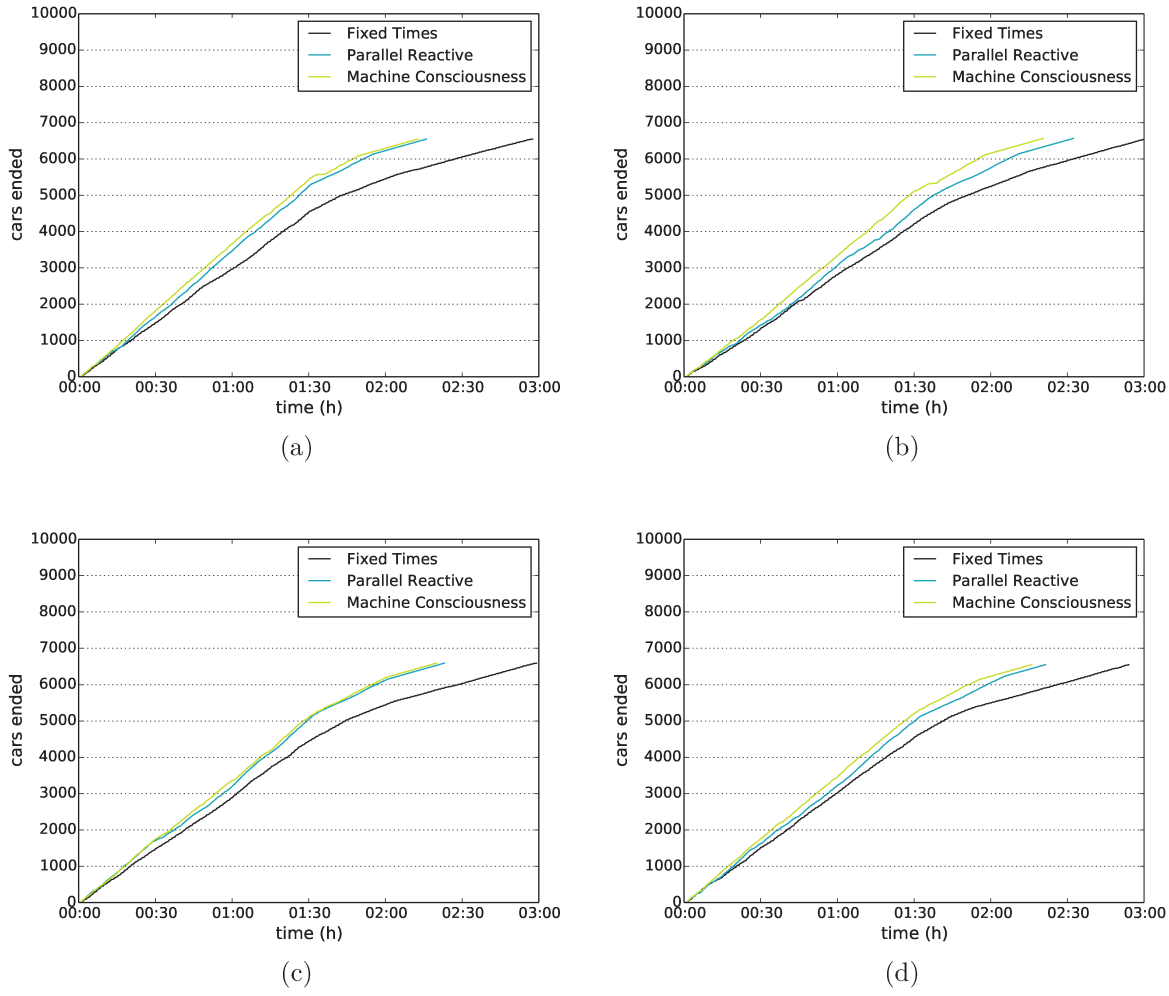


Figure 6.12: Twin T Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.12a, 6.12b, 6.12c and 6.12d represent four distinct simulation experiments.

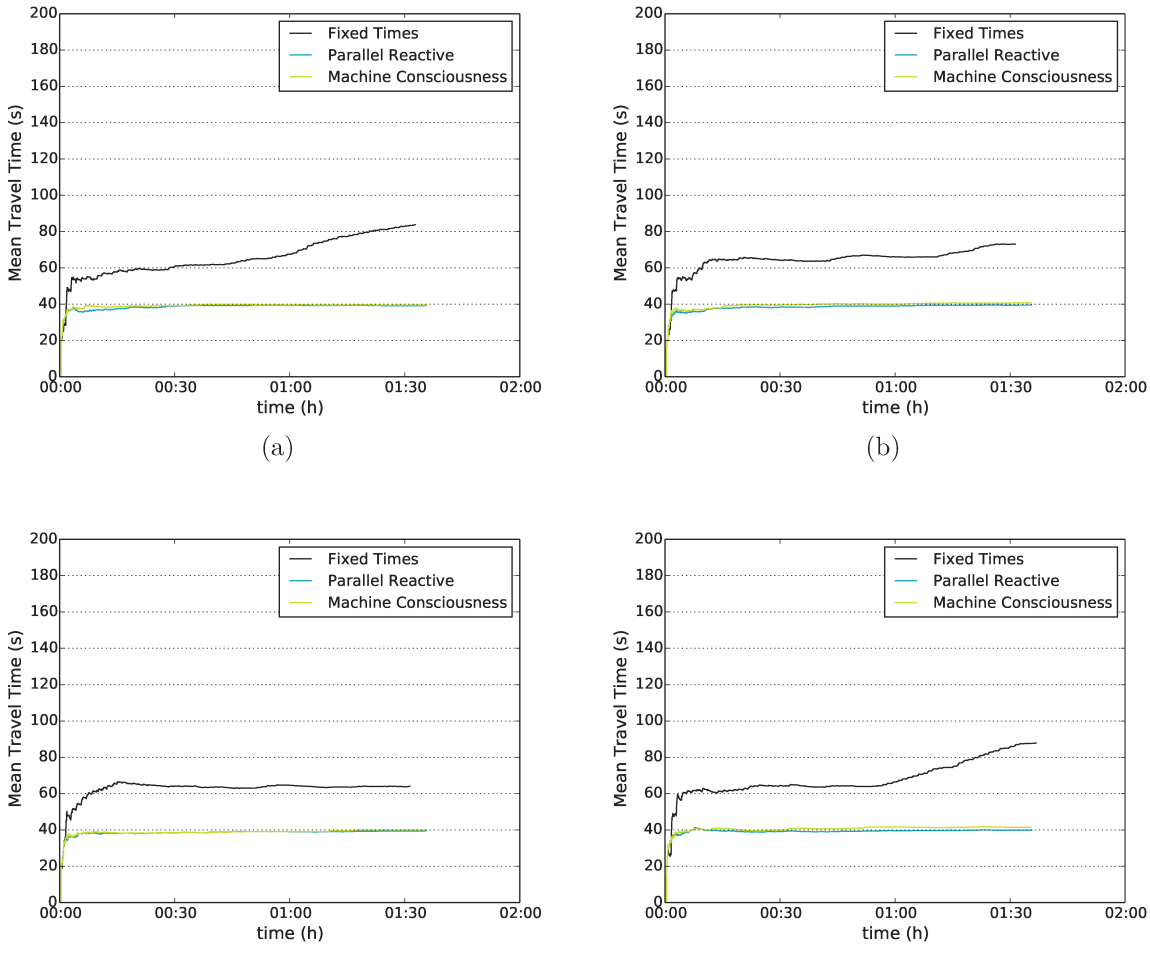
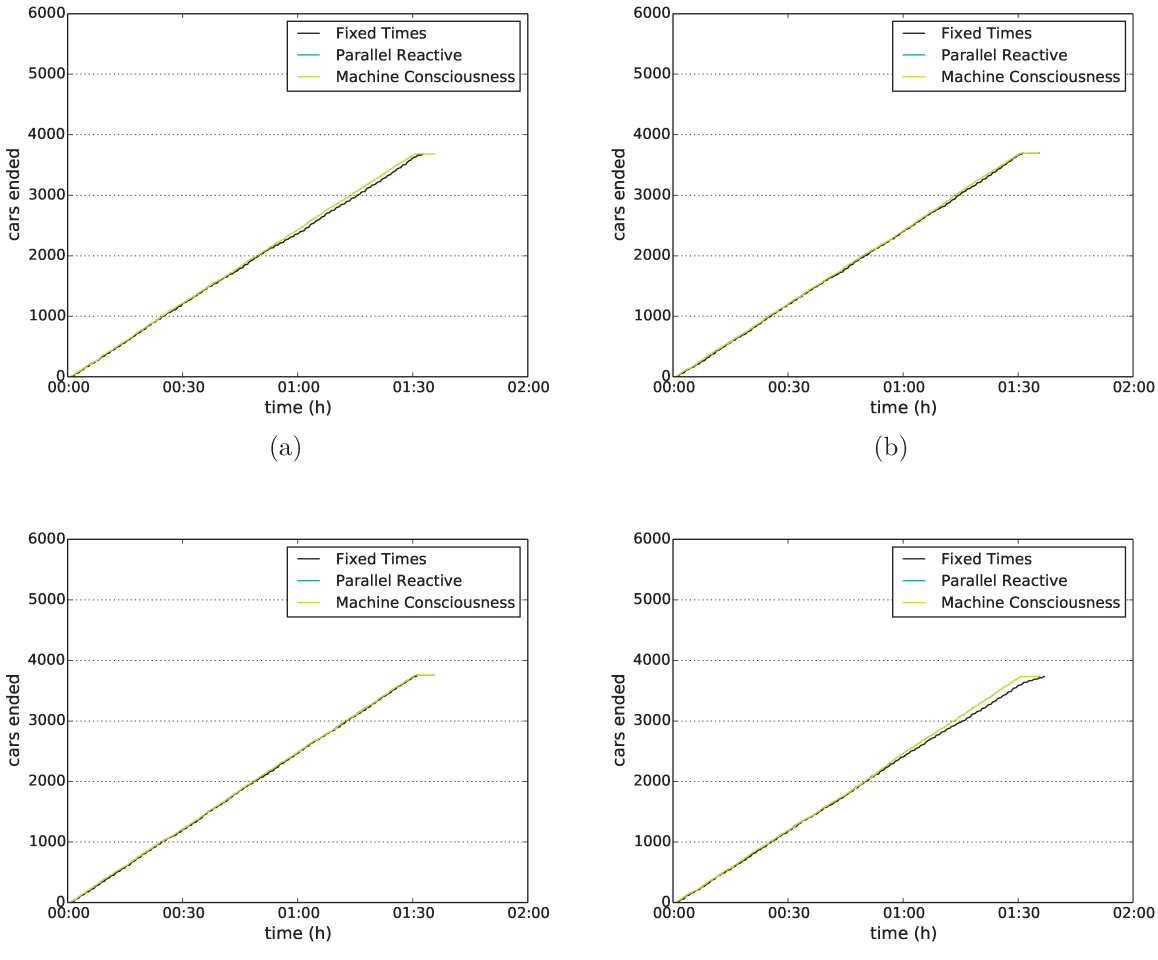


Figure 6.13: Twin T Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.13a, 6.13b, 6.13c and 6.13d represent four distinct simulation experiments.



(a) (b)

(c) (d)

Figure 6.14: Twin T Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.14a, 6.14b, 6.14c and 6.14d represent four distinct simulation experiments.

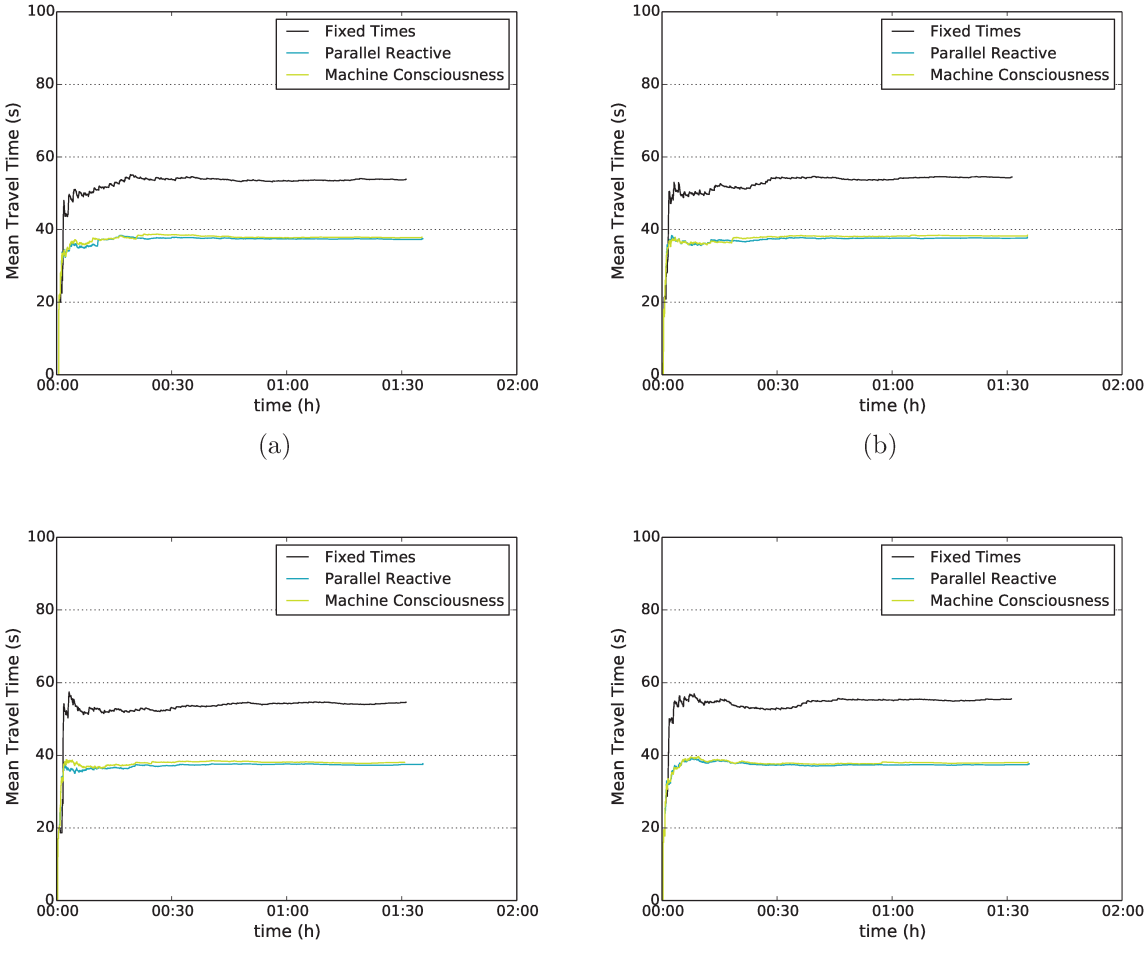
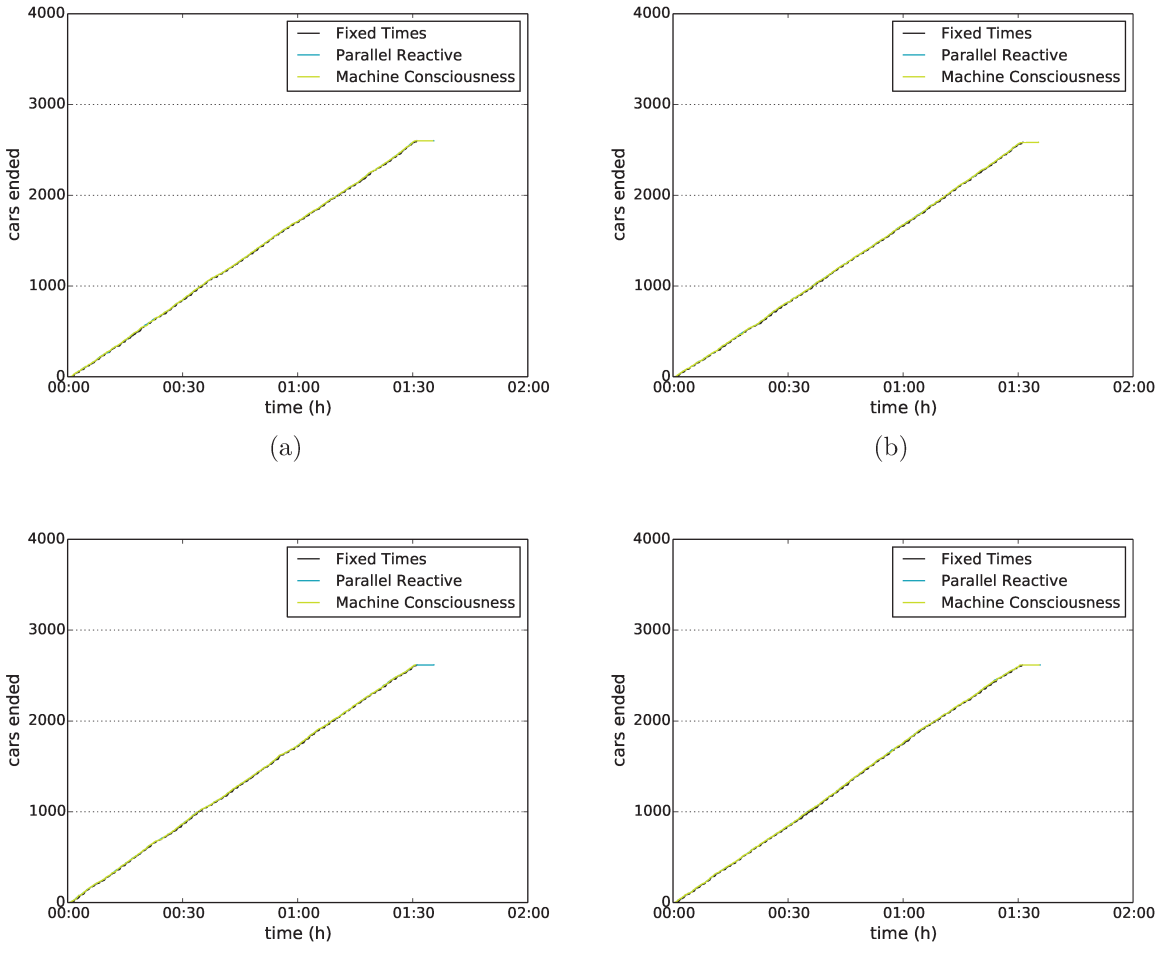


Figure 6.15: Twin T Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.15a, 6.15b, 6.15c and 6.15d represent four distinct simulation experiments.



(a) (b)

(c) (d)

Figure 6.16: Twin T Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.16a, 6.16b, 6.16c and 6.16d represent four distinct simulation experiments.

Consciousness, including the same heuristic in the behavioral codelet, and their only difference is the machine consciousness mechanism, as the consciousness codelet is active in the CodeRack in the case of the Machine Consciousness controller, whether it is not active in the case o f the Parallel Reactive controller. In this scenario, we found gains in the vehicle's mean travel time of around 35% comparing the Machine Consciousness to the Fixed Times, and up to 26% in some simulations, comparing the Machine Consciousness to the Parallel Reactive, but in the latter case most of the times the gains are around 15 %.

Figures 6.13 and 6.14 show the results of four simulation experiments considering scenario P = 0.7, presenting mean travel time and end of the trip of all vehicles over time, respectively. In this scenario, as the traffic load becomes already very small, there is no difference in the performance of the Parallel Reactive controller and the Machine Consciousness controller, mainly because there is no traffic jam problem. Both controllers have very similar performances and the gains observed in vehicle's mean travel time, compared to the Fixed Times controller, are around more than 35%.

Figures 6.15 and 6.16 show the results of four simulation experiments considering scenario P = 1.0, presenting mean travel time and end of the trip of all vehicles over time, respectively. In this scenario, our interpretation is the same as in the P=0.7 scenario. Both controllers have very similar performances and the gains observed in vehicle's mean travel time, compared to the Fixed Times controller, are around 30%.

6.3 Corridor Model

Figures 6.17 and 6.18 show the results of four simulation experiments considering scenario P = 0.1, presenting mean travel time and end of the trip of all vehicles over time, respectively. Corridor model is a little more sophisticated model, consisting of four junctions, where expectations start to grow around how the machine consciousness will be able to help in the overall performance of the system. Indeed, results were very consistent in this model, presenting higher gains in the more crowded P = 1.0 scenario, and smaller gains on the others, until almost no gain is seen on P=0.7 and P = 1.0. In the case of

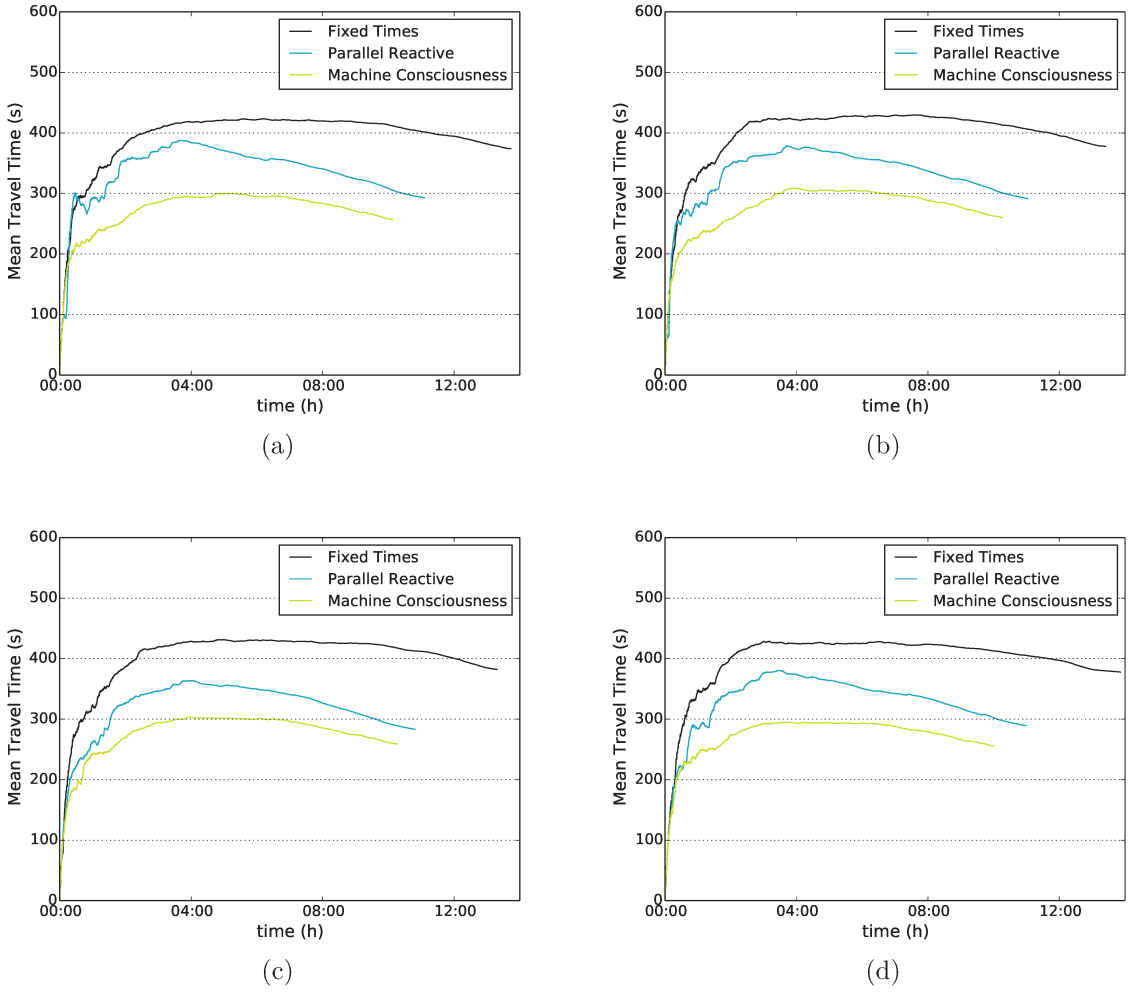


Figure 6.17: Corridor Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.17a, 6.17b, 6.17c and 6.17d represent four distinct simulation experiments.

P=0.1, we observed gains of around 33%, comparing the Machine Consciousness controller with the Fixed Times one, and 13% comparing with the Parallel Reactive controller.

Figures 6.19 and 6.20 show the results of four simulation experiments considering scenario P = 0.4, presenting mean travel time and end of the trip of all vehicles over time, respectively. In the case of P=0.4, we observed gains of around 30%, comparing the Machine Consciousness controller with the Fixed Times one, and 10% comparing with the Parallel Reactive controller.

Figures 6.21 and 6.22 show the results of four simulation experiments considering scenario P = 0.7, presenting mean travel time and end of the trip of all vehicles over time, respectively. In the case of P=0.7, we observed gains of around 80%, comparing the

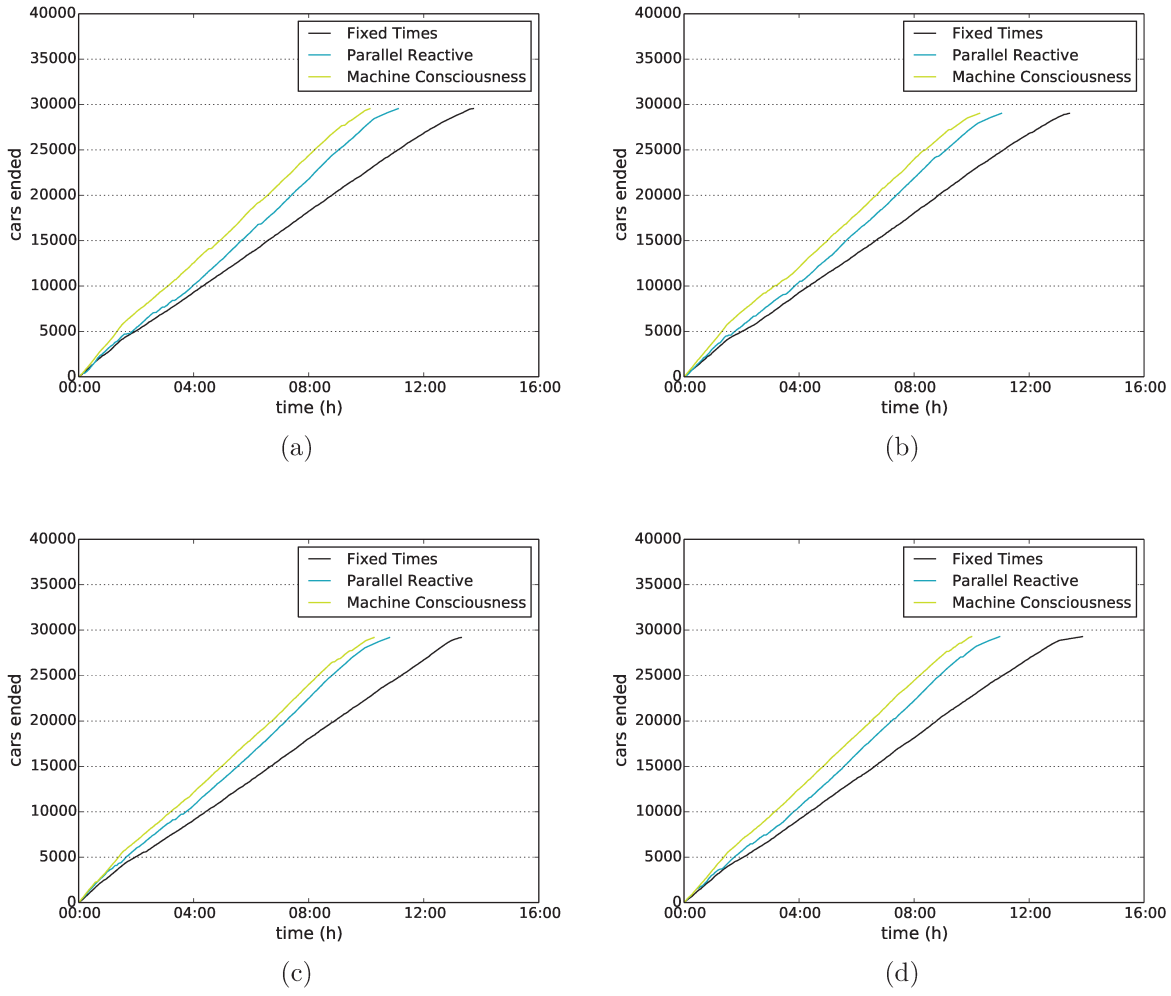


Figure 6.18: Corridor Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.18a, 6.18b, 6.18c and 6.18d represent four distinct simulation experiments.

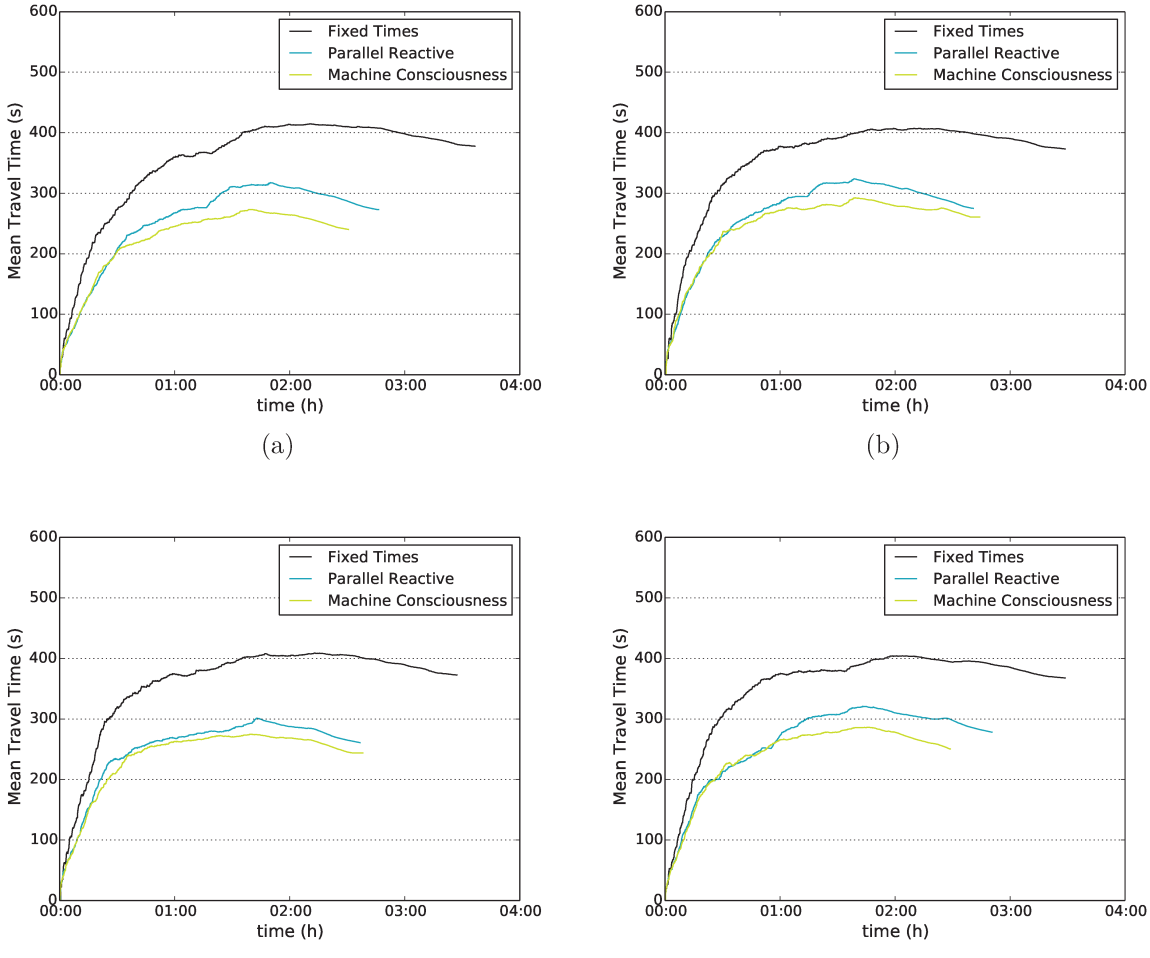


Figure 6.19: Corridor Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.19a, 6.19b, 6.19c and 6.19d represent four distinct simulation experiments.

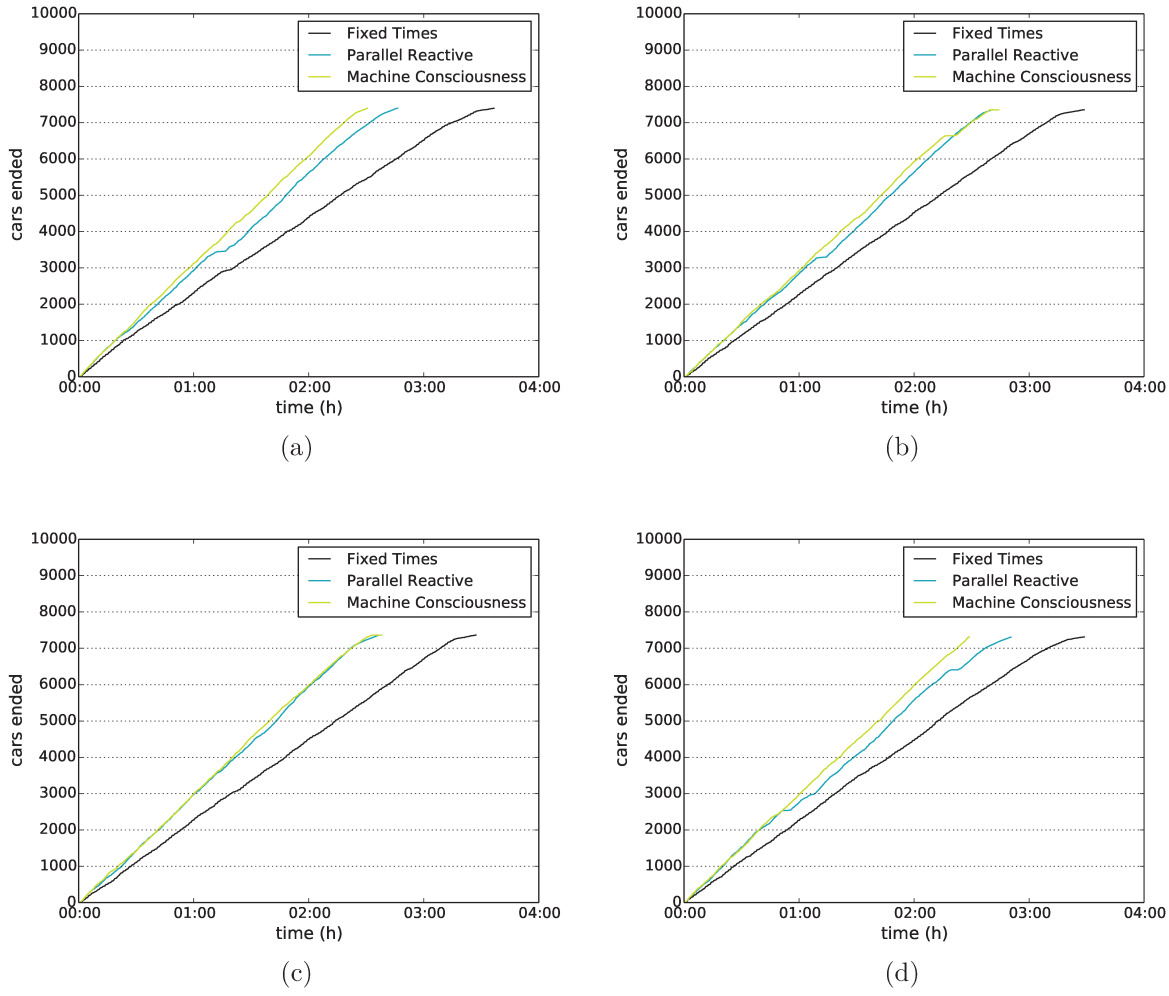


Figure 6.20: Corridor Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.20a, 6.20b, 6.20c and 6.20d represent four distinct simulation experiments.

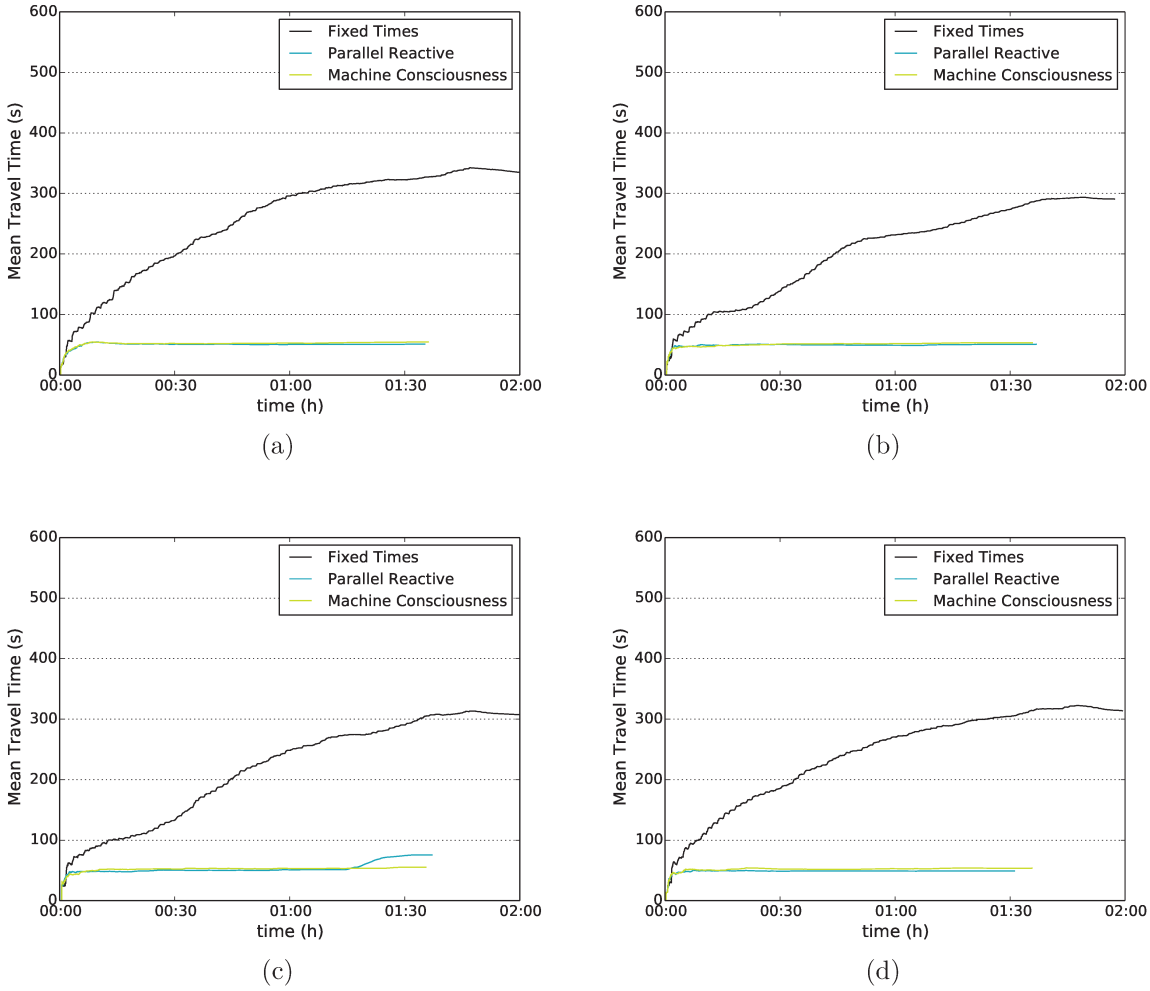
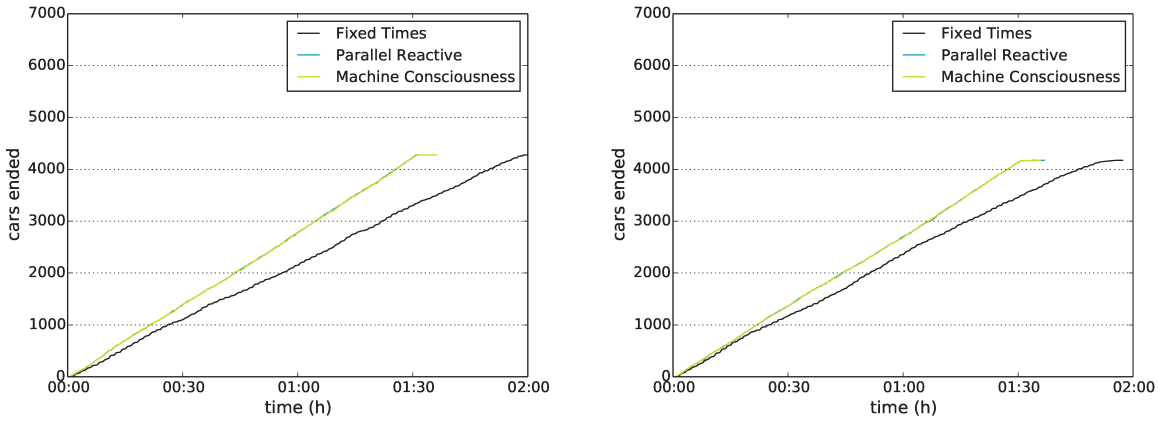


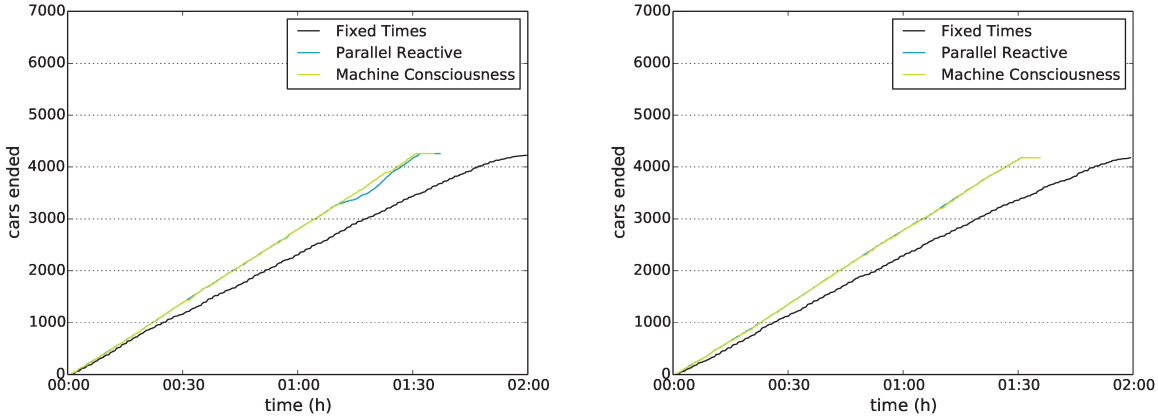
Figure 6.21: Corridor Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.21a, 6.21b, 6.21c and 6.21d represent four distinct simulation experiments.

Machine Consciousness controller with the Fixed Times one, and no gain comparing with the Parallel Reactive controller. The reason is lack of traffic jam, as traffic load becomes small.

Figures 6.23 and 6.24 show the results of four simulation experiments considering scenario P = 1.0, presenting mean travel time and end of the trip of all vehicles over time, respectively. In the case of P=1.0, we observed gains of around 40%, comparing the Machine Consciousness controller with the Fixed Times one, and no gain comparing with the Parallel Reactive controller. The reason is lack of traffic jam, as traffic load becomes small.



(a) (b)



(c) (d)

Figure 6.22: Corridor Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.22a, 6.22b, 6.22c and 6.22d represent four distinct simulation experiments.

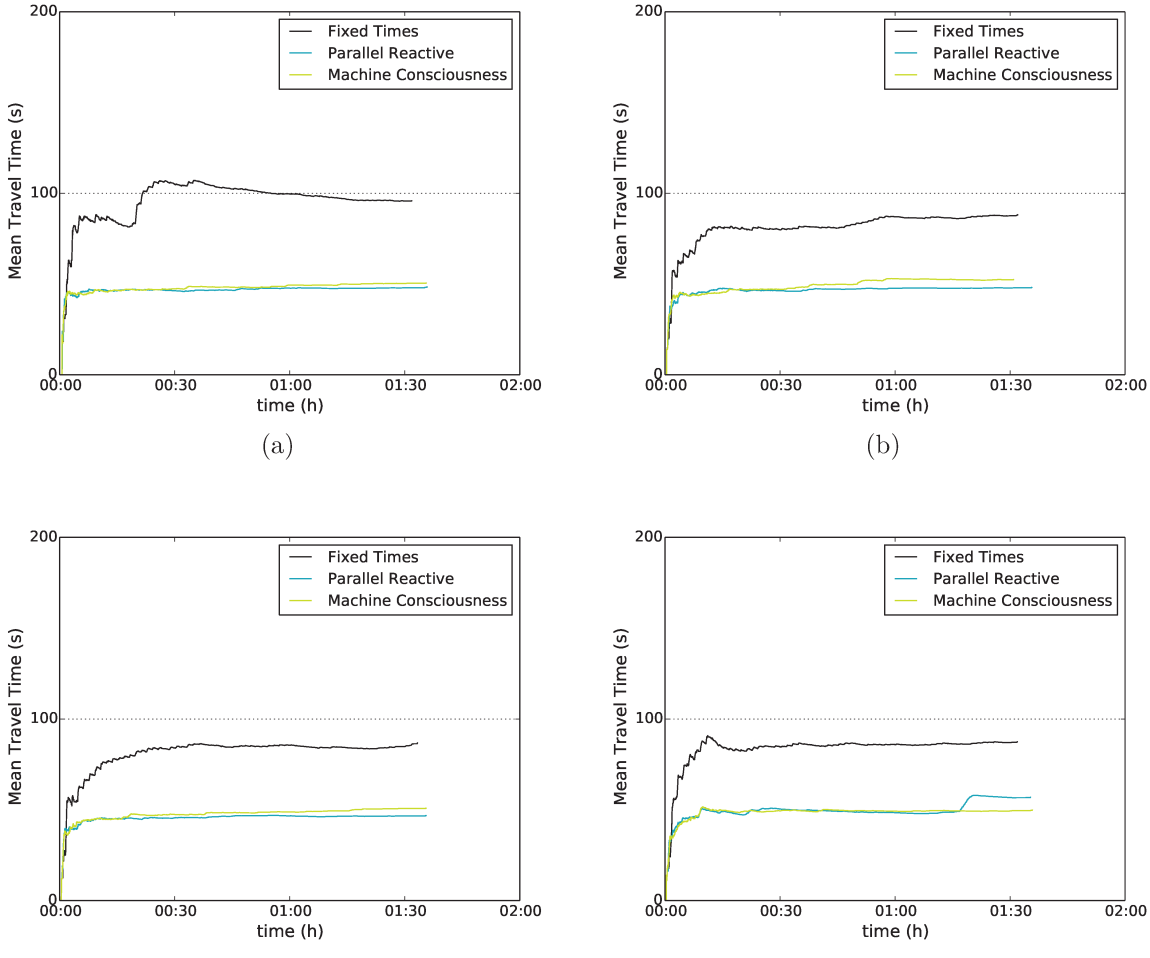
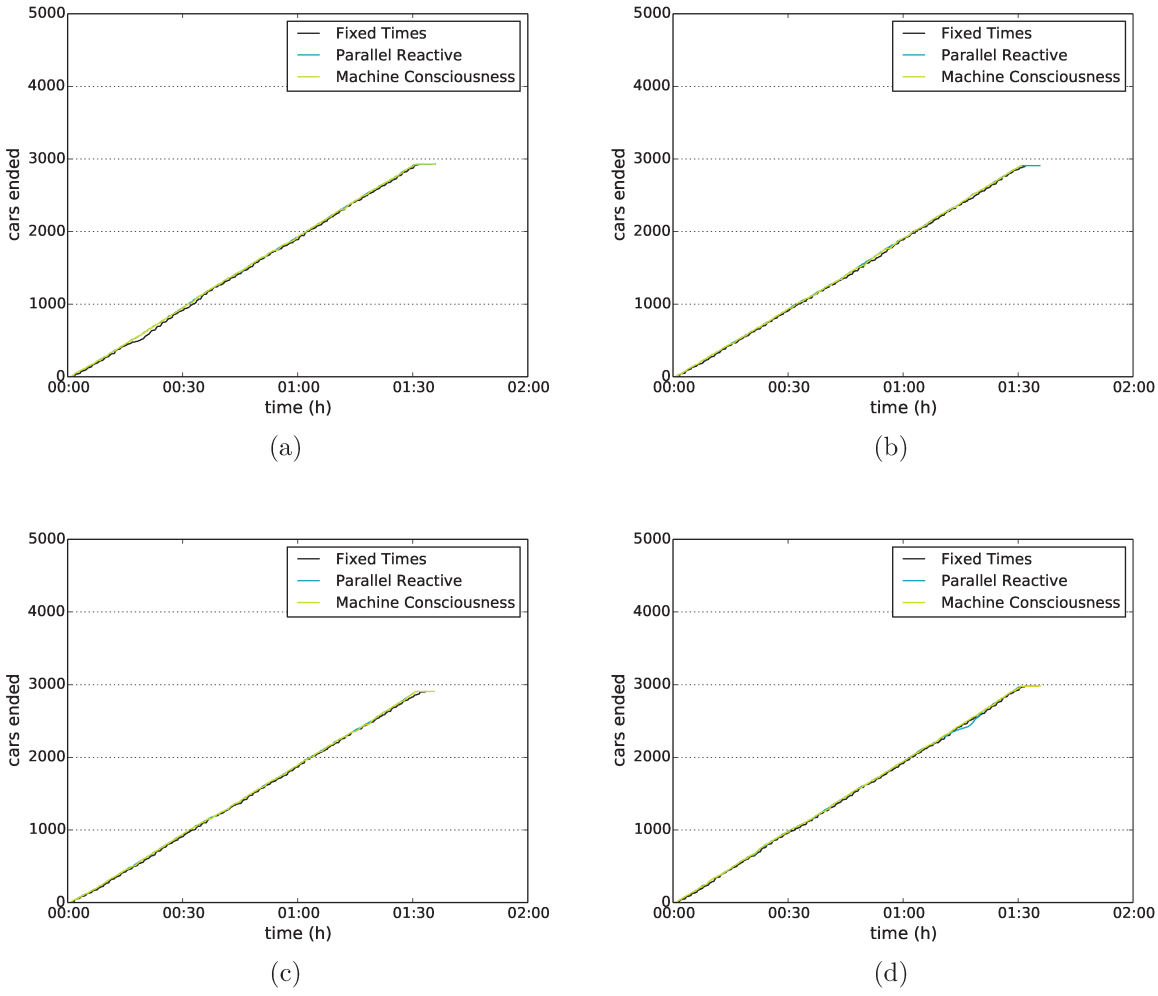


Figure 6.23: Corridor Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.23a, 6.23b, 6.23c and 6.23d represent four distinct simulation experiments.



(a) (b)

(c) (d)

Figure 6.24: Corridor Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.24a, 6.24b, 6.24c and 6.24d represent four distinct simulation experiments.

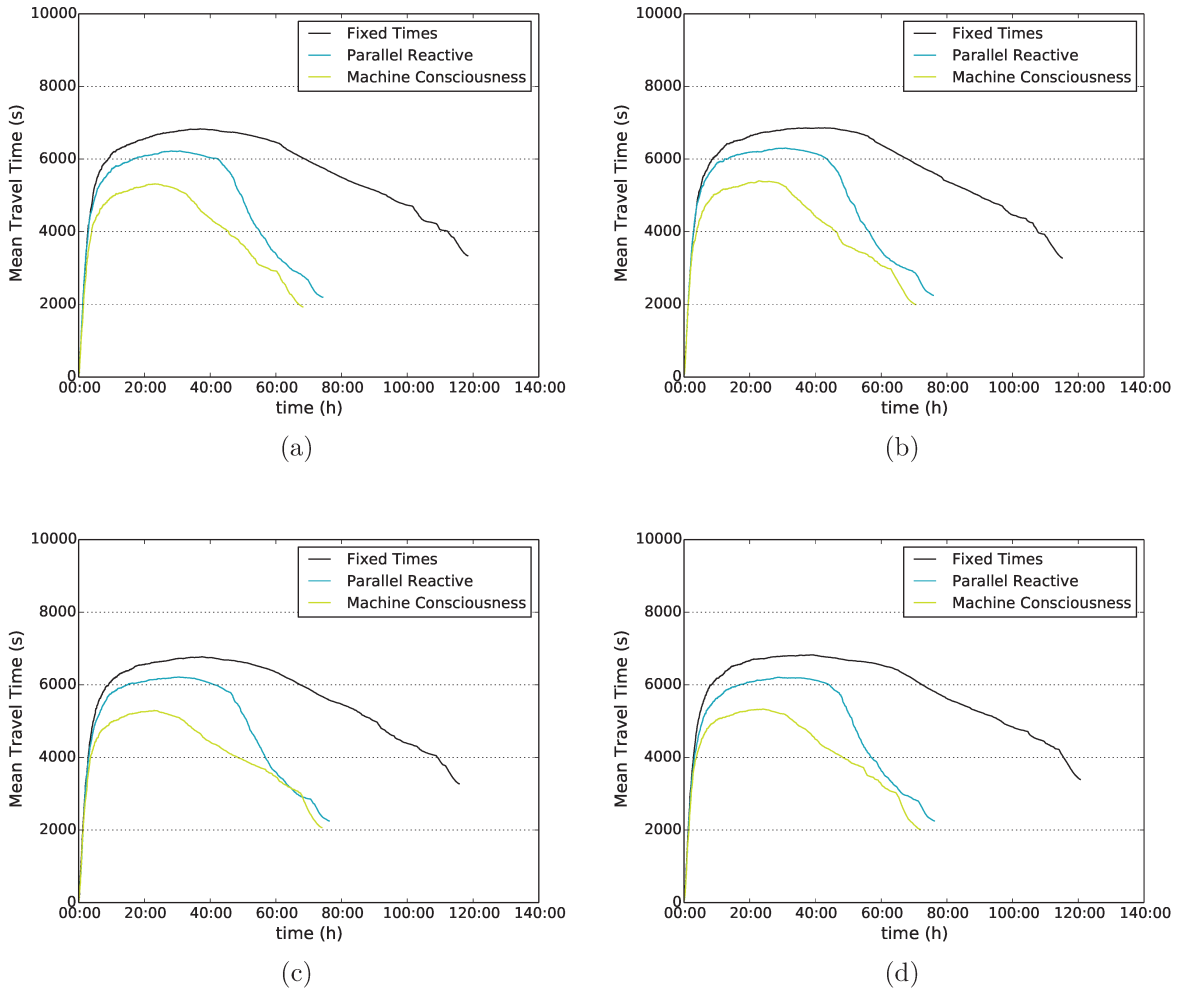


Figure 6.25: Manhattan Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.25a, 6.25b, 6.25c and 6.25d represent four distinct simulation experiments.

6.4 Manhattan Model

Figures 6.25 and 6.26 show the results of four simulation experiments considering scenario P = 0.1, presenting mean travel time and end of the trip of all vehicles over time, respectively. The Manhattan model inserts more challenge in the last step of network models before experimenting our controllers in a real network. Once again, results were consistent with our expectations in scenario P=0.1, where the Machine Consciousness controller performed a little better than the Parallel Reactive. We found gains around 10% comparing the Machine Consciousness controller with the Parallel Reactive one and around 45% comparing the Machine Consciousness controller to the Fixed Times. In the

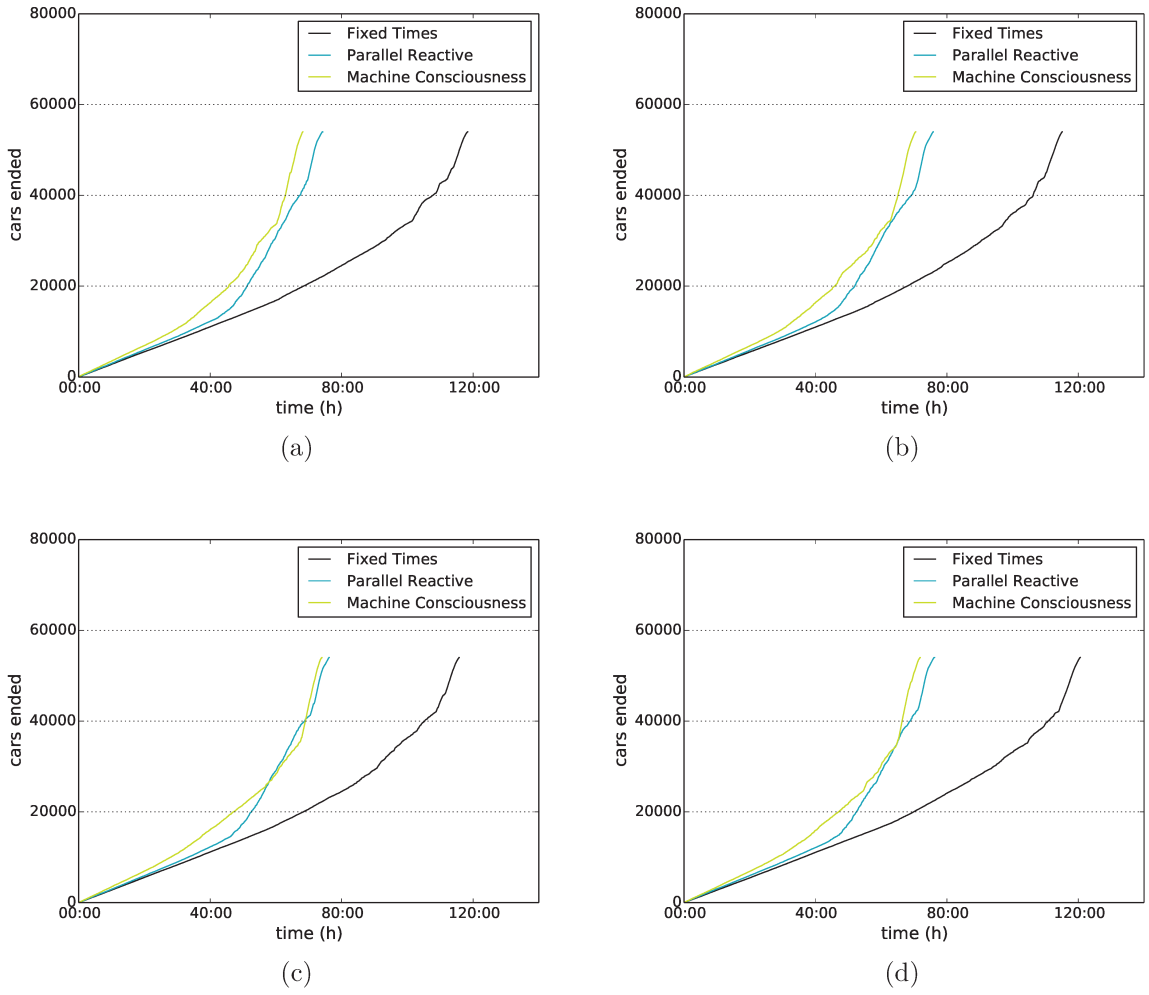


Figure 6.26: Manhattan Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.26a, 6.26b, 6.26c and 6.26d represent four distinct simulation experiments.

Manhanttan Model results, the gap between end of the simulation of the Fixed Times and the other controllers gets bigger. The reason is because the other controllers are more efficient in alleviating the critical situation, and as the cars are generated only in the first 5400 seconds of the simulation and the left to complete their routes, the more efficient the controller, the sooner its simulation will end in time.

Figures 6.27 and 6.28 show the results of four simulation experiments considering scenario P = 0.4, presenting mean travel time and end of the trip of all vehicles over time, respectively.

Figures 6.29 and 6.30 show the results of four simulation experiments considering scenario P = 0.7, presenting mean travel time and end of the trip of all vehicles over time,

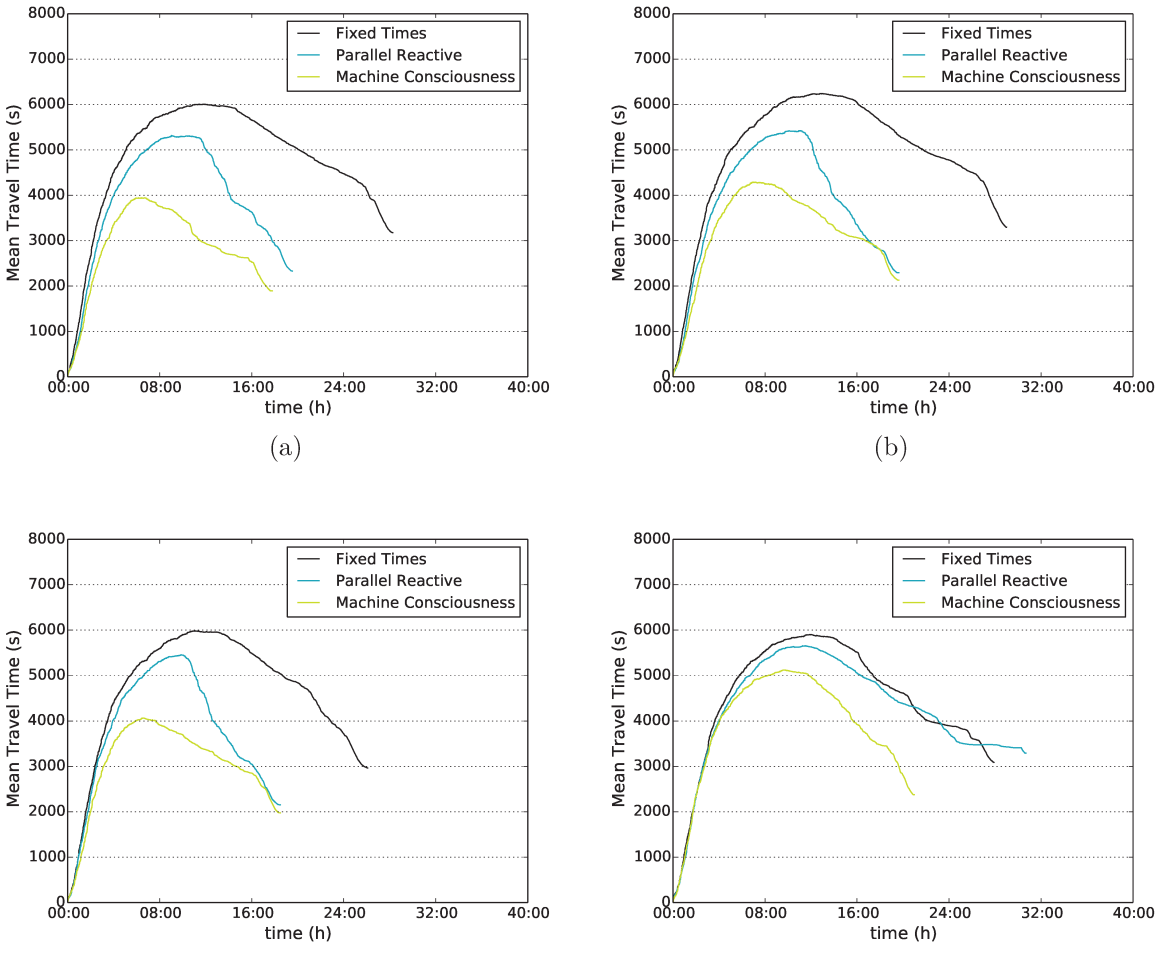


Figure 6.27: Manhattan Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.27a, 6.27b, 6.27c and 6.27d represent four distinct simulation experiments.

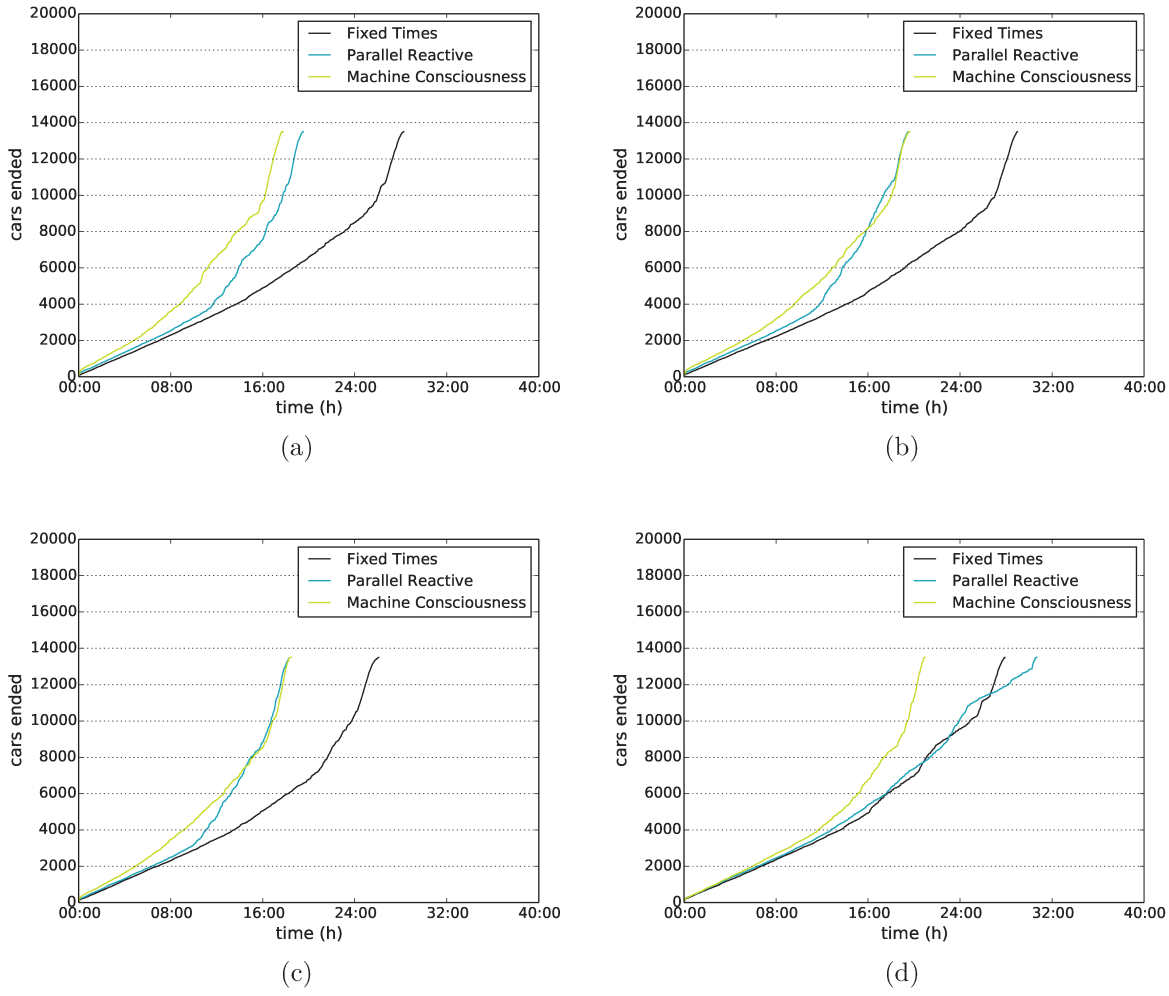


Figure 6.28: Manhattan Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.28a, 6.28b, 6.28c and 6.28d represent four distinct simulation experiments.

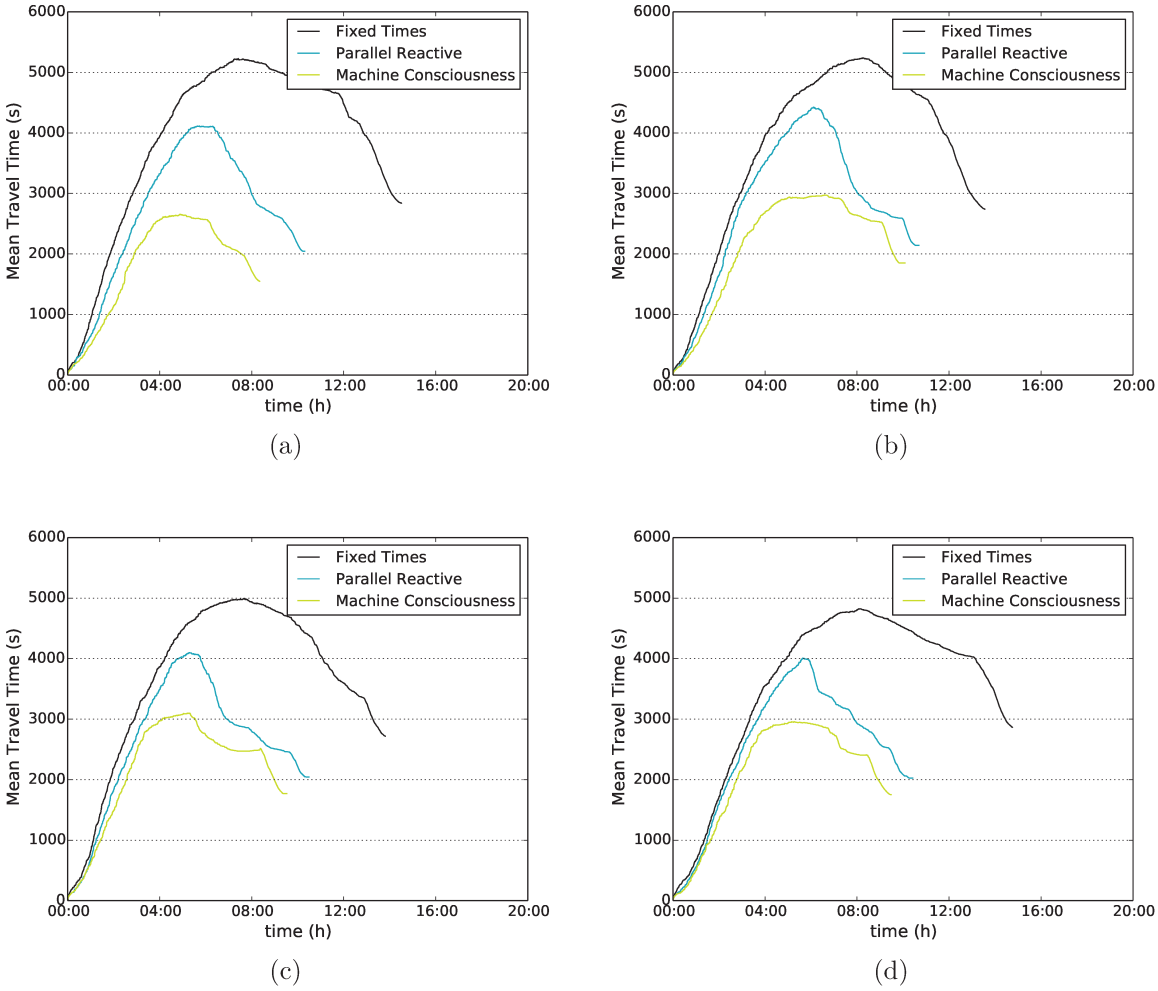


Figure 6.29: Manhattan Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.29a, 6.29b, 6.29c and 6.29d represent four distinct simulation experiments.

respectively.

Figures 6.31 and 6.32 show the results of four simulation experiments considering scenario P = 1.0, presenting mean travel time and end of the trip of all vehicles over time, respectively.

6.5 Downtown Campinas Model

Figures 6.33 and 6.34 show the results of simulation experiments considering scenario P = 0.1, presenting mean travel time and end of the trip of all vehicles over time, respectively. After having experimented with four smaller problems, Downtown Campinas is finally a

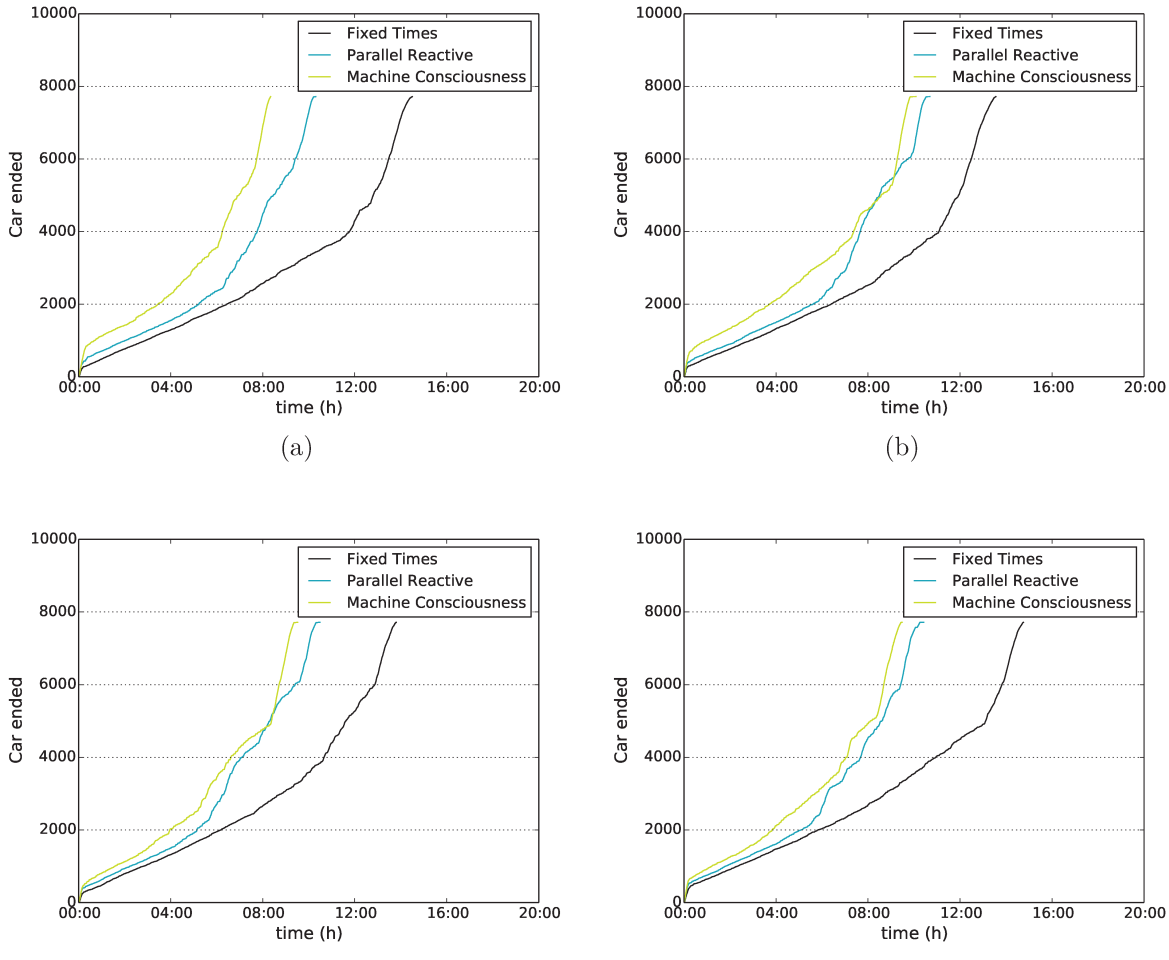


Figure 6.30: Manhattan Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.30a, 6.30b, 6.30c and 6.30d represent four distinct simulation experiments.

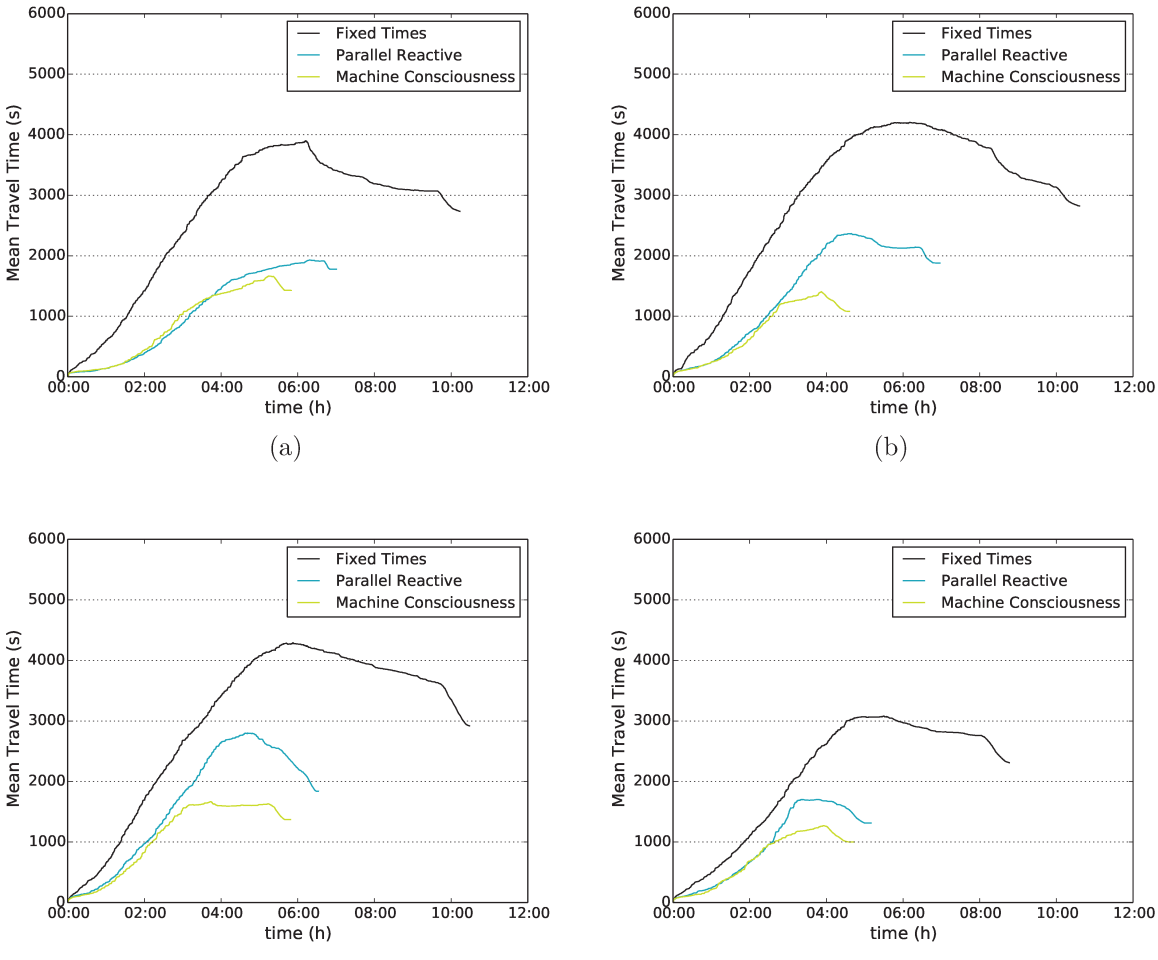


Figure 6.31: Manhattan Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.31a, 6.31b, 6.31c and 6.31d represent four distinct simulation experiments.

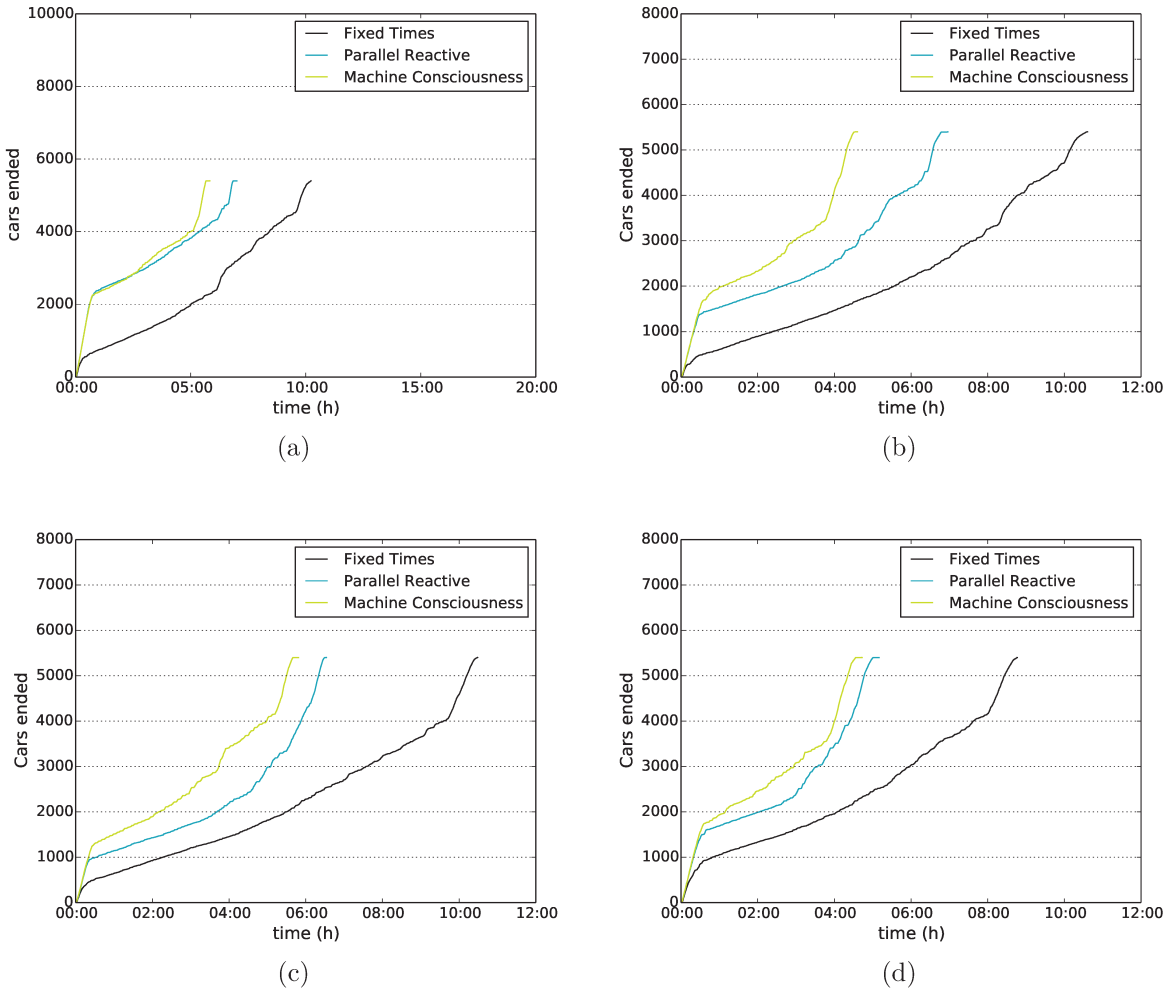
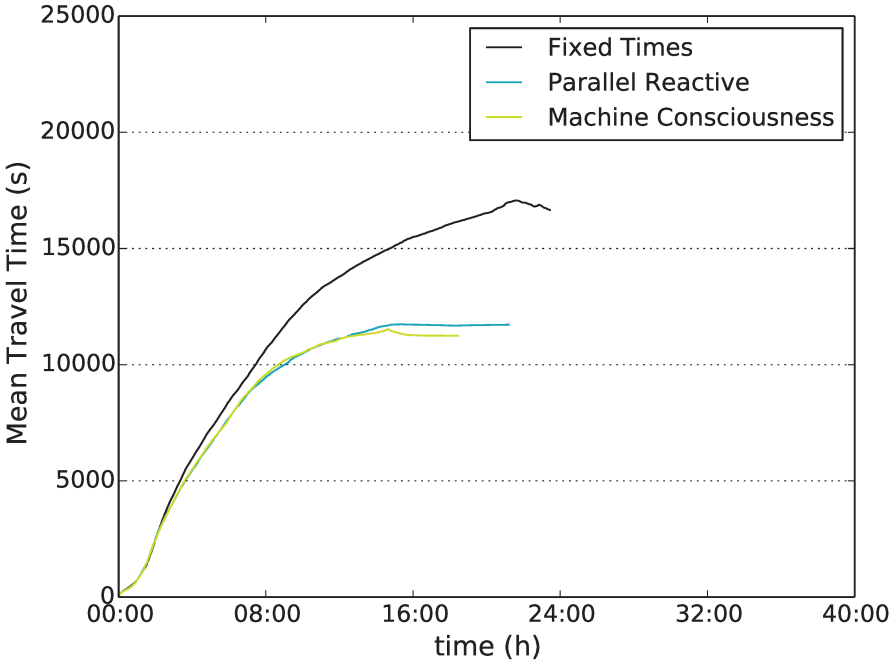


Figure 6.32: Manhattan Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.32a, 6.32b, 6.32c and 6.32d represent four distinct simulation experiments.

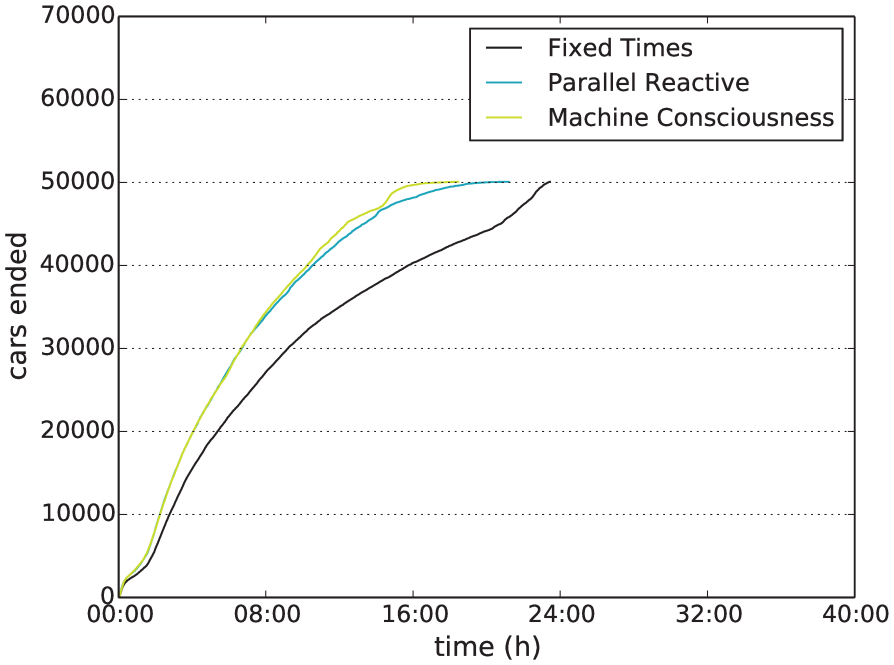
real model, composed by more than 100 junctions. In this model scenario P=0.1 became so crowded with gridlock jams, that the Machine Consciousness and the Parallel Reactive behaved almost the same all the time. This was because there was no much to do in the gridlock situation. Even though the Machine Consciousness mechanism would identify and broadcast the information about the most critical junction in the whole area, not much could be done in order to help this junction. We found gains in the vehicle's mean travel time of around 3%, comparing the Machine Consciousness controller with the Parallel Reactive one, and of around 30% comparing the Machine Consciousness to the Fixed Times controller.

Figures 6.35 and 6.36 show the results of four simulation experiments considering



(a)

Figure 6.33: Downtown Campinas Model in scenario P = 0.1. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis.



(a)

Figure 6.34: Downtown Campinas Model in scenario P = 0.1. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis.

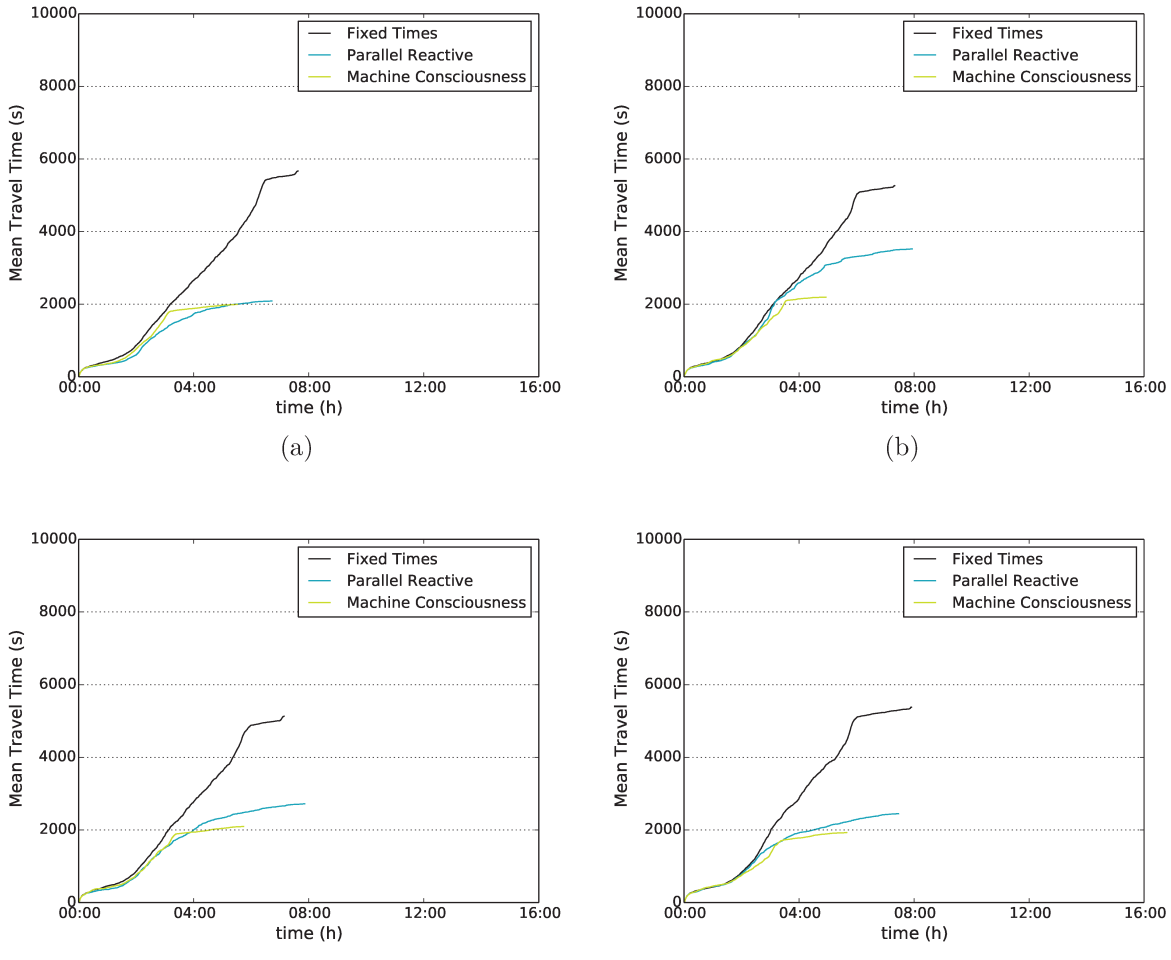


Figure 6.35: Downtown Campinas Model in scenario P = 0.4. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.35a, 6.35b, 6.35c and 6.35d represent four distinct simulation experiments.

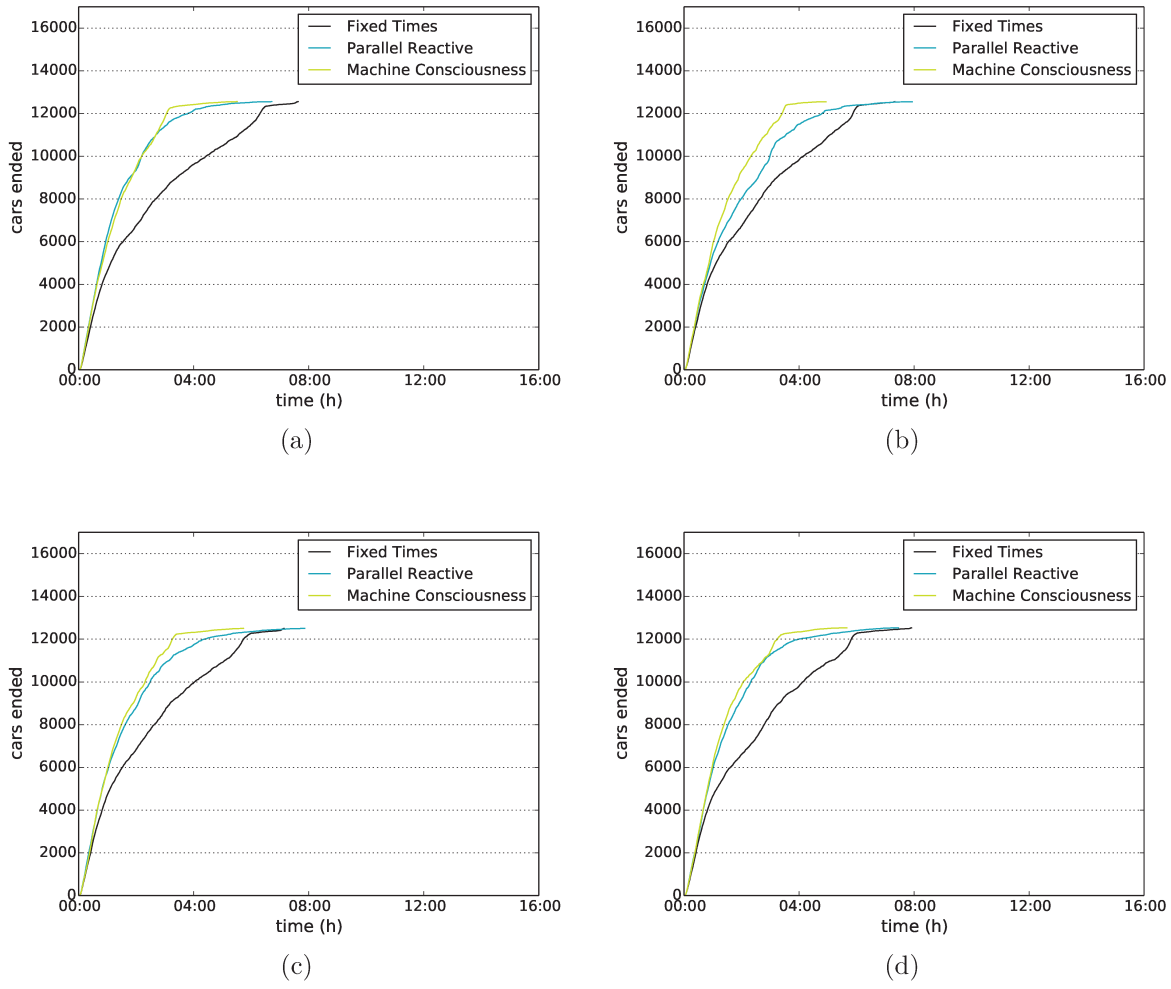
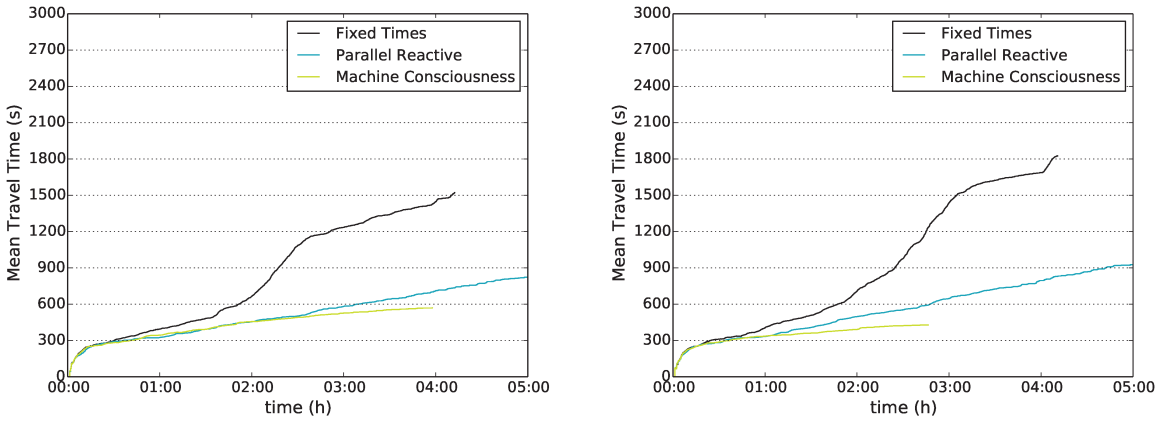
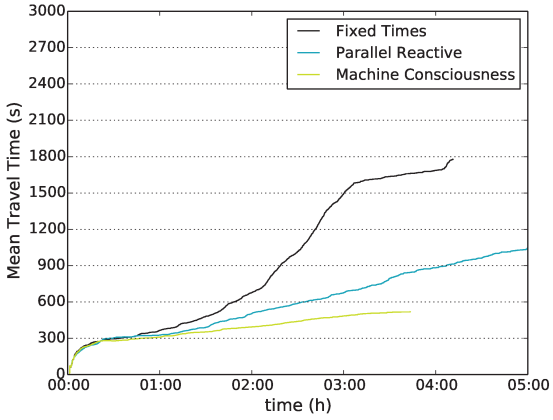


Figure 6.36: Downtown Campinas Model in scenario P = 0.4. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.36a, 6.36b, 6.36c and 6.36d represent four distinct simulation experiments.

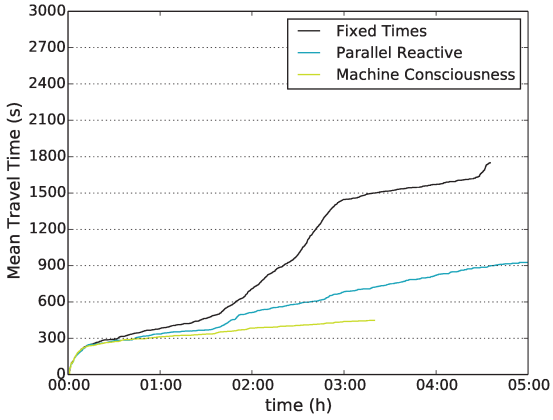


(a)

(b)

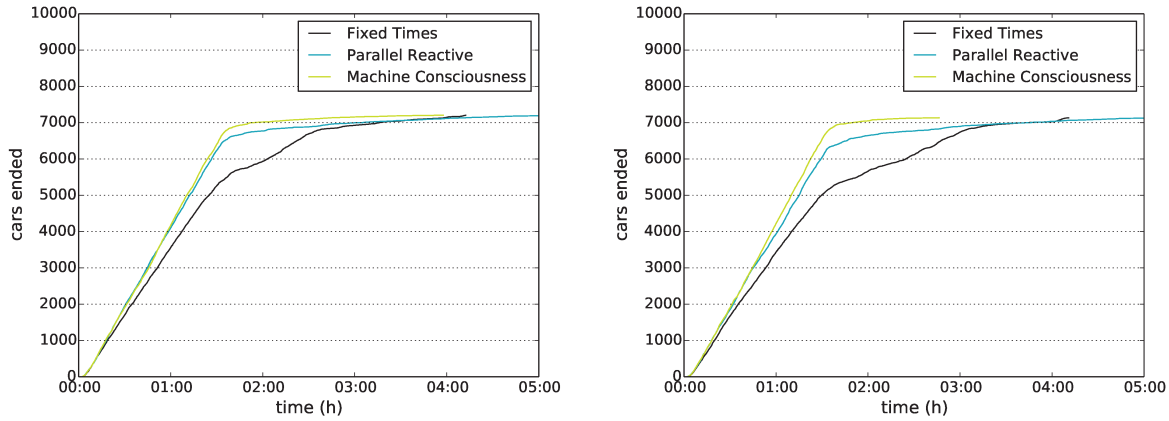


(c)



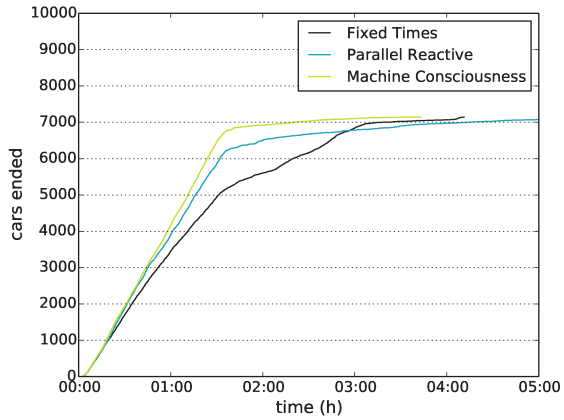
(d)

Figure 6.37: Downtown Campinas Model in scenario P = 0.7. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.37a, 6.37b, 6.37c and 6.37d represent four distinct simulation experiments.

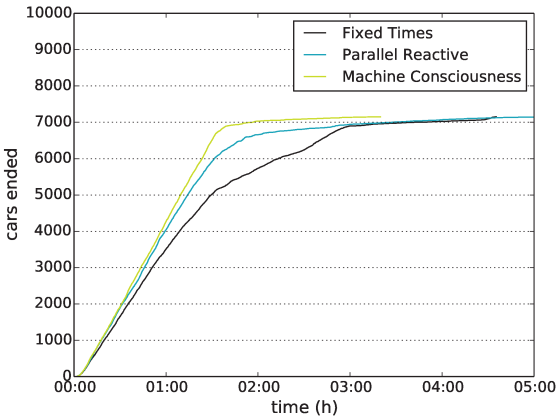


(a)

(b)



(c)



(d)

Figure 6.38: Downtown Campinas Model in scenario P = 0.7. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.38a, 6.38b, 6.38c and 6.38d represent four distinct simulation experiments.

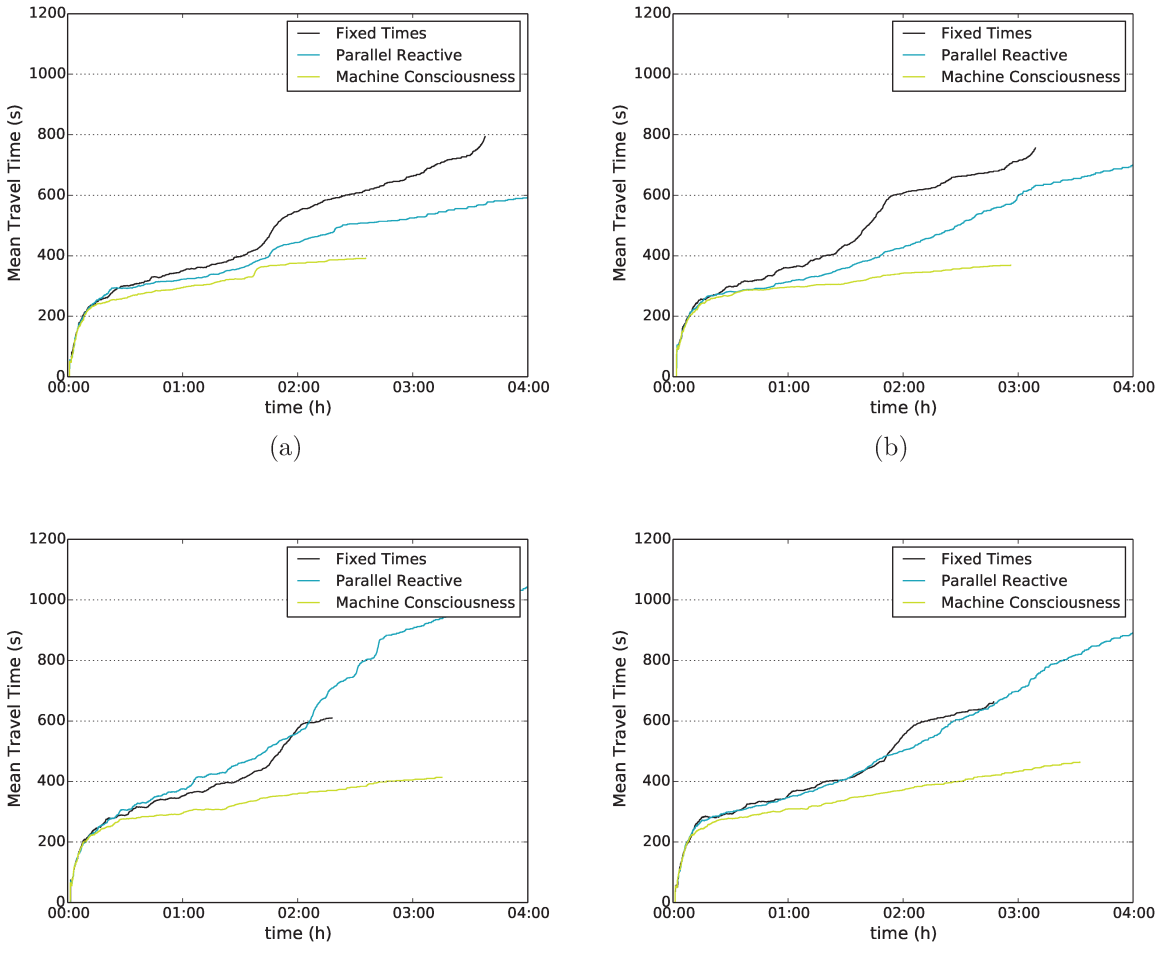


Figure 6.39: Downtown Campinas Model in scenario P = 1.0. Mean travel time is given in seconds in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.39a, 6.39b, 6.39c and 6.39d represent four distinct simulation experiments.

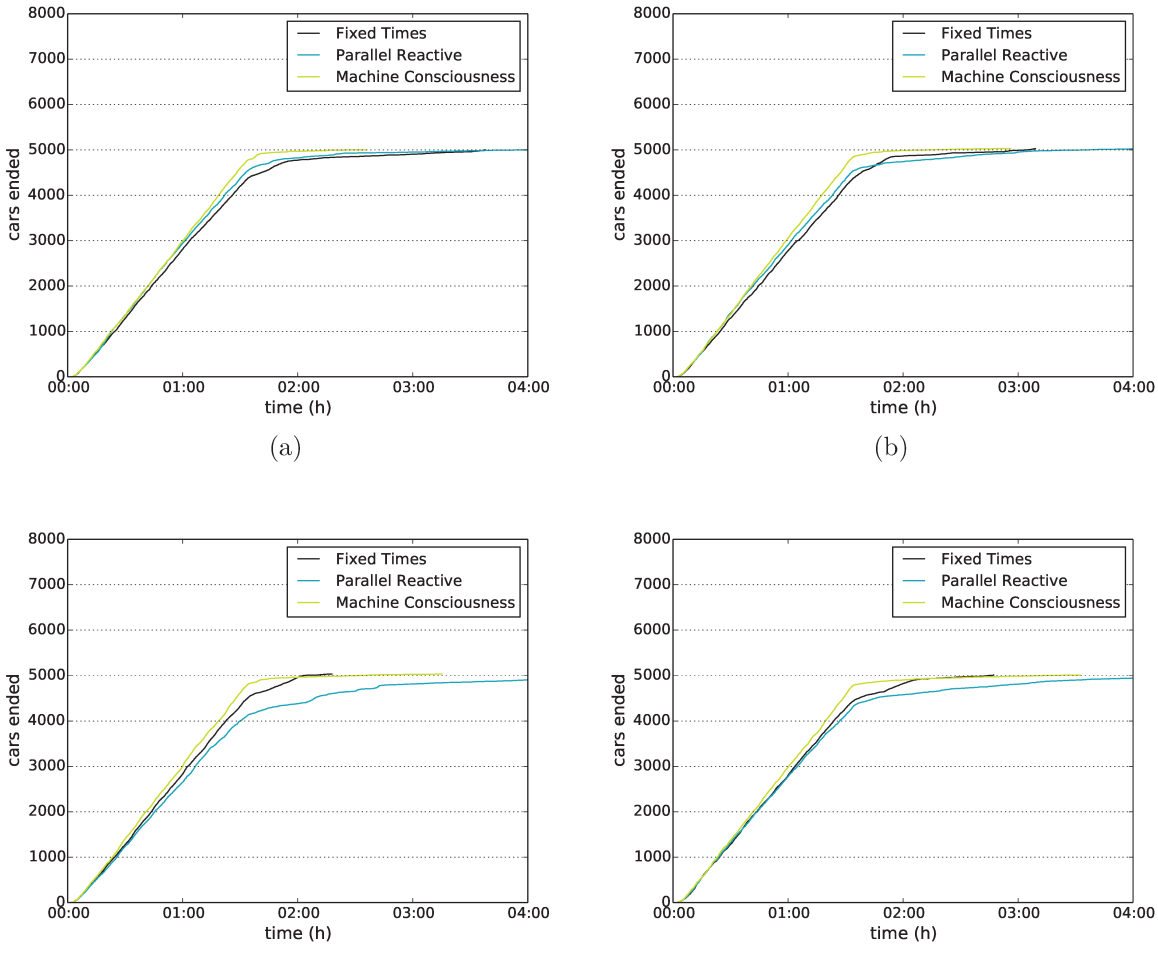


Figure 6.40: Downtown Campinas Model in scenario P = 1.0. Cars ending the trip is given in units in the vertical axis, and simulation time is given in hours in the horizontal axis. Panels 6.40a, 6.40b, 6.40c and 6.40d represent four distinct simulation experiments.

scenario P = 0.4, presenting mean travel time and end of the trip of all vehicles over time, respectively. As the gridlock situation became a little better in the P=0.4 scenario, gains grew. Comparing the Machine Consciousness controller to the Parallel Reactive one, we found gains in the vehicle's mean travel time of around 25%, but reaching more than 40% in some scenarios, while the comparison of the Machine Consciousness to the Fixed Times showed gains of around 65%.

Figures 6.37 and 6.38 show the results of four simulation experiments considering scenario P = 0.7, presenting mean travel time and end of the trip of all vehicles over time, respectively. Even though the traffic load is reduced, the size of the model makes it challenging anyway. In this case, we found the Parallel Reactive controller behaving badly sometimes, getting closer in performance to the Fixed Times. The Machine Consciousness controller performed always better. We found gains of around 20% comparing the Machine Consciousness to the Parallel Reactive controller and of around 40% comparing to the Fixed Times.

Figures 6.39 and 6.40 show the results of four simulation experiments considering scenario P = 1.0, presenting mean travel time and end of the trip of all vehicles over time, respectively. The Machine Consciousness controller performed always better. We found gains of around 30% comparing the Machine Consciousness to the Parallel Reactive controller and of around 50% comparing to the Fixed Times.

6.6 Experiments Raw Data

In this Chapter, we presented only four results for each scenario. Because the results of the many experiments were coherent, this was a way of summing up and delivering the message more directly. If you want to analyze more results or if you want to take a look in more details, we have made the experiments raw data available at:
https://github.com/CST-Group/traffic-signal-control-app-experiments.

In this raw data repository, you will find the following structure and content:

1. /bin (runnable .jar of the CST Architecture)

2. /experiments
    (a) corridor
        i. networkModel (urban network model structure)
        ii. scripts (shell scripts used to run the experiments)
        iii. summaryOutputs
            A. fixed (results for the Fixed Times controller)
            B. machineConsciousness (results for the Machine Consciousness controller)
            C. parallelReactive (results for the Parallel Reactive controller)
        iv. vehicleRoutesInputs (scenarios of the simulations - P=0.1,0.4,0.7 and 1.0)
    (b) downtownCampinas (idem)
    (c) manhattan (idem)
    (d) simpleT (idem)
    (e) twinT (idem)

The results achieved in this work showed the feasibility of the proposed system, and brought evidence fulfilling the scientific hypothesis stated.

In Chapter 7, we close the thesis restating our main findings, their limitations and the possibility of extending these findings in future works.

Chapter 7

Conclusion and Future Work

7.1 Main Findings

A consistent gain in performance with the "Machine Consciousness" traffic signal controller during all simulation time, throughout different simulated scenarios, could be observed in the results of Chapter 6. By the end of the simulation, this gain can range from around 10% to more than 20% in specific times of the simulated scenarios, when compared to the "Parallel Reactive" controller without the artificial consciousness mechanism.. Due to the stochastic nature of simulated experiments, sometimes a smaller gain in performance is observed, as in Figure 6.17c, which shows a gain around 7%, but even in this case it is a relevant and consistent gain throughout the simulation.

As expected, there was no relevant difference in the behaviour of the "Parallel Reactive" and the "Machine Consciousness" controllers, in the scenarios represented in Figure 6.1, 6.5 and 6.21, which is an evidence that the presence of the GWT mechanism not only brings gains in performance in stress situations, but also does not interfere in situations where the network is under a normal flow, resembling the way consciousness interferes in the automatic unconscious processes in animal brain. In the cases of Figure 6.29a, 6.29b, 6.29c and 6.29d, due to the complexity of the network model, even the concentration P = 0.7 was enough to produce lower stress situations. We also found out that the more complex the model and the more critical the situation, the better was the gain produced by the machine consciousness mechanism, until it reached a gridlock situation, and in Figure 6.33, when the gains are highly reduced.

7.2 Publications

This work has made contributions to the state of the art in cognitive architectures and machine consciousness mechanisms. These contributions are documented in the present text, and also in a number of international congress articles, journal papers, a submitted patent and registered software, as follows:

1. Raizer, K., Paraense, A. L. O. and Gudwin, R. R. (2011). A cognitive neuroscience inspired codelet-based cognitive architecture for the control of artificial creatures with incremental levels of machine consciousness, Symposium Proceedings at the AISB11 Convention. Machine Consciousness 2011: Self, Integration and Explanation, UK Society for the Study of Artificial Intelligence and Simulation of Behaviour, York, United Kingdom (Raizer et al., 2011).

2. Raizer, K., Paraense, A. L. O. and Gudwin, R. R. (2012). A cognitive architecture with incremental levels of machine consciousness inspired by cognitive neuroscience, International Journal of Machine Consciousness (Raizer et al., 2012).

3. Raizer, K., Rohmer, E., Paraense, A. L. O. and Gudwin, R. R (2013). Registered Software: Intelligent Software Agent Applied to Assistive Technology (Raizer et al., 2013a).

4. Raizer, K., Rohmer, E., Paraense, A. L. O., Gudwin, R. R. and Cardozo, E. (2013). Pending patent: Method for the development of a suggestion agent with behavior network for assistive technology (Raizer et al., 2013c).

5. Raizer, K., Rohmer, E., Paraense, A. L. O. and Gudwin, R. R. (2013). Effects of behavior network as a suggestion system to assist BCI users, IEEE 2013 Symposium on Computational Intelligence in Rehabilitation and Assistive Technologies (CIRAT) (Raizer et al., 2013b).

6. Gudwin, R. R., Paraense, A. L. O., Raizer, K. (2015). A Cognitive Systems Toolkit (CST), with machine consciousness capabilities, released as Open Source under LGPL licence at `https://github.com/CST-Group/cst` (Gudwin et al., 2015).

7. Paraense, A. L .O., Raizer, K. and Gudwin, R.R. (2015). A CST Machine Consciousness Traffic Signal Control Application released as Open Source under Apache licence at
https://github.com/CST-Group/traffic-signal-control-app (Paraense et al., 2015).

8. Paraense, A. L. O., Raizer, K. and Gudwin, R.R. (2016). A machine consciousness approach to urban traffic control, Biologically Inspired Cognitive Architectures, Volume 15, January 2016, Pages 61-73, ISSN 2212-683X,
http://dx.doi.org/10.1016/j.bica.2015.10.001 (Paraense et al., 2016)

7.3 Limitations and Future work

The next steps for improving and testing our hypothesis are the following: working with varying conscious thresholds for codelet activation - which could be necessary for a less specialized controller - applying different heuristics in the unconscious automatic codelets, and working with more complex traffic networks and scenarios, which should include more real networks and data, to evaluate if these results are scalable.

The next step for evolving our consciousness model is to implement automatization and deautomatization of behaviours, as novel situations become frequent and are stored in long term memory, becoming accessible without conscious interference. For instance, imagine that the heuristic used in this work, that was readily available to unconscious automatic codelets, would have to be learned somewhere in the history of the controller, as the agent became more and more experienced. It could also be deautomatized, if environmental changes somehow turned it into a non functional strategy as time went by.

One true limitation for running the experiments was hardware: having to run a conceptually parallel cognitive architecture in a serial computer surely compromises the results. Even though the Java Virtual Machine implements virtual threads, and the current computers have more than one core in the CPU, in the end of the day, we are still limited and locked into a seriality.

As more future work, CST architectures and the machine consciousness technology produced in this work should be applied to other fields, such as manufacturing, construction, agriculture, mining, education, etc.

7.4 Conclusion

This work produced evidence to support the hypothesis that an artificial consciousness mechanism, which serially broadcasts content to automatic processes, can bring advantages to the global task performed by such a society of parallel agents working together for a common goal. A consistent gain in performance with the "Machine Consciousness" traffic signal controller during all simulation time, throughout different simulated scenarios, could be observed, ranging from around 10% to more than 20%, when compared to the "Parallel Reactive" controller without the artificial consciousness mechanism.

Based on the results of the experiments, it would be worth it to apply the technology to big cities with high traffic. However, it is important to point out that the infrastructure of the city must allow actuators to adaptively control the traffic lights and sensors to gather the necessary information about position and velocity of vehicles that the model expects.

Bibliography

Anderson, J.R.; et al. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060. ISSN 0033-295X. doi:10.1037/0033-295X.111.4.1036.

Ardila, Alfredo; et al. (2011). There are two different language systems in the brain. *Journal of Behavioral and Brain Science*, 1(02):23.

Baars, Bernard J. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.

Baars, Bernard J.; Franklin, S. (2009). Consciousness is computational: the lida model of global workspace theory. *International Journal of Machine Consciousness*, 01(0101):23. doi:10.1142/S1793843009000050.

Baars, Bernard J.; Franklin, S.; Ramsoy, T. Z. (2013). Global workspace dynamics: Cortical "binding and propagation" enables conscious contents. *Frontiers in Psychology*, 4(200):22. doi:10.3389/fpsyg.2013.00200.

Baars, Bernard J.; Franklin, Stan (2007). An architectural model of conscious and unconscious brain functions: Global workspace theory and ida. *Neural Networks*, 20:955–961. doi:10.1016/j.neunet.2007.09.013.

Baars, Bernard J.; Gage, Nicole M. (2007). *Cognition, Brain, and Consciousness*. Elsevier, 1 edition.

Baddeley, Alan (1997). *Human Memory - Theory and Practice*. Psychology Press, Taylor and Francis Group, revised edition edition.

Baddeley, Alan (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423.

Baddeley, Alan (2002). Is working memory still working? *European Psychologist*, 7(2):85–97.

Baddeley, Alan; Thomson, Nell; Buchanan, Mary (1975). Word length and the structure of short-term memory. *Journal Of Verbal Learning And Verbal Behavior*, 14(6):575–589.

Bates, Joseph; et al. (1994). The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125.

Bogner, Myles Brandon (1999). *Realizing "Consciousness" In Software Agents*. Ph.D. thesis, The University of Memphis.

Boillot, F.; Midenet, S.; Pierrelee, Jc (2006). The real-time urban traffic control system CRONOS: Algorithm and experiments. *Transportation Research Part C: Emerging Technologies*, Vol14,Issue1:p18–38.

Box, Simon; Waterson, Ben (2013). An automated signalized junction controller that learns strategies by temporal difference reinforcement learning. *Engineering Applications of Artificial Intelligence*, 26(1):652 – 659. ISSN 0952-1976. doi:10.1016/j.engappai.2012.02.013.

Braver, T S; et al. (1997). A parametric study of prefrontal cortex involvement in human working memory. *NeuroImage*, (1):49–62.

Brockfeld, E.; et al. (2001). Optimizing traffic lights in a cellular automaton model for city traffic. *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat.*, 64(5).

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, 2(1):14–23.

Brooks, R.A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1-31-3):139–159. doi:10.1098/rsta.2003.1257.

Budakova, D.; Dakovski, L. (2006). *Computer Model of Emotional Agents*, volume 4133. Springer Berlin / Heidelberg.

Cabeza, Roberto; et al. (2008). The parietal cortex and episodic memory: an attentional account. *Nature Reviews Neuroscience*, 9(8):613–625.

Cai, Chen; Wong, Chi Kwong; Heydecker, Benjamin G. (2009). Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 17(5):456 – 474. ISSN 0968-090X. doi:http://dx.doi.org/10.1016/j.trc.2009.04.005. Artificial Intelligence in Transportation Analysis: Approaches, Methods, and Applications.

Canamero, Dolores (1997). A hormonal model of emotions for behavior control. *VUB AI-Lab Memo*, 2006.

Canamero, Dolores (1998). Issues in the design of emotional agents. In *Emotional and Intelligent: The Tangled Knot of Cognition. Papers from the 1998 AAAI Fall Symposium*, pages 49–54.

Castro, Leandro Nunes de (2006). *Fundamentals of Natural Computing (Chapman & Hall/Crc Computer and Information Sciences)*. Chapman & Hall/CRC. ISBN 1584886439.

Cavanna, Andrea Eugenio; Nani, Andrea (2014). *Consciousness: Theories in Neuroscience and Philosophy of Mind*. Springer-Verlag Berlin Heidelberg.

Chalmers, David J. (1995). Facing up to the problem of consciousness. *Journal of Consciousness Studies*, 2(3):200–219.

Chalmers, David J. (1996). *The Conscious Mind: In Search of a Fundamental Theory*. Oxford University Press.

Chella, Antonio; Frixione, Marcello; Gaglio, Salvatore (2005). Planning by imagination in cicerobot, a robot for museum tours. In *Proceedings of the AISB 2005 symposium on next generation approaches to machine consciousness: imagination, development, intersubjectivity, and embodiment*, pages 40–49.

Choy, Min Chee; Srinivasan, D.; Cheu, R.L. (2003). Cooperative, hybrid agent architecture for real-time traffic signal control. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(5):597–607. ISSN 1083-4427. doi:10.1109/TSMCA.2003.817394.

Cowan, N (2001). The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–114; discussion 114–85.

Cox, Michael T (2005). Metacognition in computation: A selected research review. *Artificial intelligence*, 169(2):104–141.

Crick, Francis; Koch, Christof (2003). A framework for consciousness. *Nature Neuroscience*, 6(2):119–126.

da Silva, Ricardo Capitanio Martins (2009). *Análise da Arquitetura Baars-Franklin de Consciência Artificial Aplicada a uma Criatura Virtual*. Ph.D. thesis, Universidade Estadual de Campinas.

Damasio, A. R. (1994). *O Erro de Descartes: Emoção, Razão e Cérebro Humano*. Europa-América.

Damasio, A. R. (1999). *The Feeling of What Happens: Body and Emotion in the Making of Consciousness*. New York: Harcourt.

de Medeiros, Valério A. S. (2006). *Urbis Brasiliae ou Sobre Cidades do Brasil*. Ph.D. thesis, Universidade de Brasília.

de Paula, Suelen M; Gudwin, Ricardo R (2015). Evolving conceptual spaces for symbol grounding in language games. *Biologically Inspired Cognitive Architectures*, 14:73–85.

Deacon, Terrence W (1998). *The symbolic species: The co-evolution of language and the brain*. WW Norton & Company.

Dehaene, S.; Naccache, L. (2001). Towards a cognitive permanence of consciousness: Basic evidence and a workspace framework. *Cognition*, (79):1–37.

Dennett, Daniel C. (1991). *Consciousness Explained*. Little, Brown and Co.

Drescher, Gary L. (1991). *Made-up minds: A constructivist approach to artificial intelligence*. MIT Press.

Dubois, Daniel (2007). *Constructing an Agent Equipped with an Artificial Consciousness: Application to an Intelligent Tutoring System*. Ph.D. thesis, Université du Québec à Montréal.

Dubois, Daniel; et al. (2008). Cognitive tutoring system with" consciousness". In *Intelligent Tutoring Systems*, pages 803–806. Springer.

Edelman, Gerald M.; Gally, Joseph A.; Baars, Bernard J. (2011). Biology of consciousness. *Frontiers in Psychology*, 2(4).

Fabbro, Franco; et al. (2015). Evolutionary aspects of self- and world consciousness in vertebrates. *Frontiers in Human Neuroscience*, 9(157). ISSN 1662-5161. doi:10.3389/fnhum.2015.00157.

Frank, Michael J; Loughry, Bryan; OReilly, R C (2001). Interactions between frontal cortex and basal ganglia in working memory: a computational model. *Cognitive, Affective and Behavioral Neuroscience*, 1(2):137–60.

Franklin, S.; Kelemen, A.; McCauley, Lee (1998). Ida: a cognitive agent architecture. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2646–2651 vol.3. ISSN 1062-922X. doi:10.1109/ICSMC.1998.725059.

Franklin, S.; et al. (2014a). Lida: A systems-level architecture for cognition, emotion, and learning. *Autonomous Mental Development, IEEE Transactions on*, 6(1):19–41. ISSN 1943-0604. doi:10.1109/TAMD.2013.2277589.

Franklin, Stan; Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pages 21–35.

Franklin, Stan; et al. (2012). Global workspace theory, its lida model and the underlying neuroscience. *Biologically Inspired Cognitive Architectures*, 1:32–43.

Franklin, Stan; et al. (2014b). Lida: A systems-level architecture for cognition, emotion, and learning. *Autonomous Mental Development, IEEE Transactions on*, 6(1):19–41.

Gallese, Vittorio; Keysers, Christian; Rizzolatti, Giacomo (2004). A unifying view of the basis of social cognition. *Trends in cognitive sciences*, 8(9):396–403.

Gamez, David (2009). The potential for consciousness of artificial systems. *International Journal of Machine Consciousness*, 1(02):213–223.

Gartner, N.; Pooran, F.; Andrews, C. (2002). Optimized policies for adaptive control strategy in real-time traffic adaptive control systems: Implementation and field testing. *Transportation Research Record: Journal of the Transportation Research Board*, 1811(1):148–156.

George, Dileep (2008). *How the Brain Might Work: a Hierarchical and Temporal Model for Learning and Recognition*. Ph.D. thesis, Stanford University.

Greenwald, Anthony G; Banaji, Mahzarin R (1995). Implicit social cognition: attitudes, self-esteem, and stereotypes. *Psychological review*, 102(1):4.

Guberinic, Slobodan; Senborn, Gordana; Lazic, Bratislav (2008). *Optimal Traffic Control: Urban Intersections*. CRC Press.

Gudwin, R. R.; Paraense, A. L. O.; Raizer, K. (2015). CST: Cognitive Systems Toolkit library. https://github.com/CST-Group/cst.

Haklay, Mordechai (Muki); Weber, Patrick (2008). Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18. ISSN 1536-1268. doi:10.1109/MPRV.2008.80.

Hawkins, Jeff (2004). *On Intelligence*. Times Books.

Henry, J. J.; Farges, J.L.; Tuffal, J. (1983). The prodyn real time traffic algorithm. In *IFAC/IFIC/IFORS Conference on Control in Transportation System*, page 305–310.

Heung, Tsin Hing; Ho, Tin-Kin; Fung, Yu-Fai (2005). Coordinated road-junction traffic control by dynamic programming. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):341–350. ISSN 1524-9050. doi:10.1109/TITS.2005.853713.

Heydecker, B.G.; Cai, Chen; Wong, C.K. (2007). Adaptive dynamic control for road traffic signals. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, pages 193–198. doi:10.1109/ICNSC.2007.372775.

Hofstadter, D.; Mitchell, M. (1994). *The Copycat Project: A Model of Mental Fluidity and Analogy-making*, pages 205–268. BasicBooks, 10, East 53rd Street, New York, NY 10022-5299.

Howard, Marc W; et al. (2005). The temporal context model in spatial navigation and relational learning: toward a common explanation of medial temporal lobe function across domains. *Psychological Review*, 112(1):75–116.

Husch, D.; Staff, Trafficware; Albeck, J. (2003). *Synchro 6: Traffic Signal Software - User Guide*. Trafficware, Limited. ISBN 9780974290317.

Kanerva, P. (1988). *Sparse distributed memory*. The MIT Press.

Kim, Chang Ouk; Park, Yunsun; Baek, Jun-Geol (2005). *Computational Science and Its Applications – ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part IV*, chapter Optimal Signal Control Using Adaptive Dynamic Programming, pages 148–160. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-32309-9. doi:10.1007/11424925_18.

Knospe, W; et al. (2002). A realistic two-lane traffic model for highway traffic. *Journal of Physics A: Mathematical and General*, 35(15):3369–3388.

Koch, Christof (2012). *Consciousness: Confessions of a Romantic Reductionist*. The MIT Press.

Krajzewicz, Daniel; et al. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138.

Kutz, Myer, editor (2004). *Handbook of Transportation Engineering*. McGraw-Hill.

Laird, J.E. (2008). Extending the soar cognitive architecture. In *Proceeding of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, pages 224–235. IOS Press, Amsterdam, The Netherlands, The Netherlands. ISBN 978-1-58603-833-5.

Laird, John (2012). *The Soar cognitive architecture*. MIT Press.

Langley, Pat; Laird, John E.; Rogers, Seth (2009). Cognitive architectures: Research issues and challenges. *Cogn. Syst. Res.*, 10(2):141–160. ISSN 1389-0417. doi:10.1016/j.cogsys.2006.07.004.

Lighthill, M. J.; Whitham, G. B. (1955). On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, pages 317–345.

Lucentini, Danilo Fernando; Gudwin, Ricardo Ribeiro (2015). A comparison among cognitive architectures: A theoretical analysis. *Procedia Computer Science*, 71:56 – 61. ISSN 1877-0509. doi:http://dx.doi.org/10.1016/j.procs.2015.12.198. 6th Annual International Conference on Biologically Inspired Cognitive Architectures, {BICA} 2015, 6-8 November Lyon, France.

Lucky, Robert W. (2009). Engineering achievements: The two lists. *IEEE Spectrum*, 46(11):23–23.

Luyanda, Felipe; et al. (2003). ACS-Lite Algorithmic Architecture: Applying Adaptive Control System Technology to Closed-Loop Traffic Signal Control Systems. *Transportation Research Record: Journal of the Transportation Research Board*, 1856(-1):175–184. doi:10.3141/1856-19.

Lämmer, Stefan; Helbing, Dirk (2008). Self-control of traffic lights and vehicle flows in urban road networks.

Maes, Pattie (1989). How to do the right thing. *Connection Science*, 1(3):291–323. doi:10.1080/09540098908915643.

Maroto, Joaquín; et al. (2006). Real-time traffic simulation with a microscopic model. *IEEE Transactions on Intelligent Transportation Systems*.

Marques, Hugo Gravato; Holland, Owen (2009). Architectures for functional imagination. *Neurocomputing*, 72(4):743–759.

McNab, Fiona; Klingberg, Torkel (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, 11(1):103–107.

Meyer, J. J. C. (2006). Reasoning about emotional agents. *Int. J. Intell. Syst*, 21(6):601–619.

Miller, George (1956). The magical number 7, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63.

Mirchandani, Pitu; Head, Larry (2001). A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, 9(6):415 – 432. ISSN 0968-090X. doi:http://dx.doi.org/10.1016/S0968-090X(00)00047-4.

Mountcastle, V. (1978). An organizing principle for cerebral function: the unit model and the distributed system. In G. Edelman; V. Mountcastle, editors, *The Mindful Brain*. MIT Press, Cambridge, Mass.

Nagel, Kai; Schreckenberg, Michael (1992). A cellular automaton model for freeway traffic. *J. Phys. I France*, 2(12):2221–2229. doi:10.1051/jp1:1992277.

Nakamiti, Gilberto Shigueo (1996). *Distributed Artificial Intelligence: Architecture, Formal Specification and Application (In Portuguese)*. Ph.D. thesis, State University of Campinas (UNICAMP).

Negatu, Aregahegn Seifu (2006). *Cognitively Inspired Decision Making for Software Agents: Integrated Mechanism for Action Selection, Expectation, Automatization and Non-Routine Problem Solving*. Ph.D. thesis, The University of Memphis.

Nii, H Penny (1986). The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38.

Ortony, A.; Clore, G.; Collins, A. (1998). *The Cognitive Structure of Emotions*. Cambridge Univ. Press.

Papageorgiou, Markos; et al. (2003). Review of road traffic control strategies. In *In Proceedings of the IEEE*, pages 2043–2067.

Paraense, A. L .O.; Raizer, K.; Gudwin, R.R (2015). Machine consciousness traffic signal control application. https://github.com/CST-Group/traffic-signal-control-app.

Paraense, André Luis O.; Raizer, Klaus; Gudwin, Ricardo R. (2016). A machine consciousness approach to urban traffic control. *Biologically Inspired Cognitive Architectures*, 15:61 – 73. ISSN 2212-683X. doi:http://dx.doi.org/10.1016/j.bica.2015.10.001.

Picard, R. W. (1997). *Affective Computing*. MIT Press, Cambridge.

Porche, Isaac; Lafortune, Stéphane (1999). Adaptive look-ahead optimization of traffic signals. *J. Intellig. Transport. Systems*, 4(3-4):209–254.

Raizer, K. (2015). *Executive Functions for Learning and Decision-Making in a Bio-Inspired Cognitive Architecture*. Ph.D. thesis, University of Campinas.

Raizer, K.; Paraense, André L. O.; Gudwin, Ricardo R. (2011). A cognitive neuroscience-inspired codelet-based cognitive architecture for the control of artificial creatures with incremental levels of machine consciousness. In *AISB - Artificial Intelligence and Simulation of Behaviour*, volume 1, pages 43–50. UK Society for the Study of Artificial Intelligence and Simulation of Behaviour, York, United Kingdom.

Raizer, K.; Paraense, André L. O.; Gudwin, Ricardo R. (2012). A cognitive architecture with incremental levels of machine consciousness inspired by cognitive neuroscience. *International Journal of Machine Consciousness*, 04(02):335–352. doi: 10.1142/S1793843012400197.

Raizer, K.; et al. (2013a). Agente de software inteligente aplicado a tecnologia assistiva. Registered software ag as:: INPI - Instituto Nacional da Propriedade Industrial. Funding agencies: CAPES; CNPQ; FAPESP.

Raizer, K.; et al. (2013b). Effects of behavior network as a suggestion system to assist bci users. In *2013 IEEE Symposium Series on Computational Intelligence*, volume 1, pages 40–47. IEEE SSCI.

Raizer, K.; et al. (2013c). Pending patent number br10201302310: Método de sugestões baseado em uma rede de comportamentos para tecnologia assistiva.

Reilly, W. S. N. (1996). *Believable Social and Emotional Agents*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Richards, Paul I. (1959). Shock waves on the highway. *Operations Research*, 4(1):42–51.

Robertson, D. I. (1969). Transyt - a traffic network study tool. *Report No TRRL-LR-253 (Transport and Road Research Laboratory, Crowthorne)*.

Robertson, D. I.; Bretherton, R. D. (1991). Optimizing networks of traffic signals in real time-the SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1):11 –15. ISSN 0018-9545. doi:10.1109/25.69966.

Samsonovich, A. (2012a). Comparative table of cognitive architectures. BICA Society. Http://bicasociety.org/cogarch/architectures.htm.

Samsonovich, Alexei (2012b). Comparative table of cognitive architectures. http://bicasociety.org/cogarch/architectures.htm.

Samsonovich, Alexei V. (2012c). On a roadmap for the {BICA} challenge. *Biologically Inspired Cognitive Architectures*, 1:100 – 107. ISSN 2212-683X. doi:http://dx.doi.org/10.1016/j.bica.2012.05.002.

Searle, John (1980). Minds, brains and programs. *The Behavioral and Brain Sciences*, 3:417–457.

Searle, John R. (1997). *The Mystery of Consciousness*. The New York Review of Books.

Sen, S.; Head, K. (1997). Controlled optimization of phases at an intersection. *Transportation Science*, 31(1):5–17.

Septseault, C; Nédélec, A. (2005). A model of an embodied emotional agent. In *IVA*, page 498.

Setchi, Rossitza; Lagos, Nikolaos; Froud, Danny (2007). Computational imagination: research agenda. In *AI 2007: Advances in Artificial Intelligence*, pages 387–393. Springer.

Shanahan, Murray (2006). A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and Cognition*, 15(2):433 – 449. ISSN 1053-8100. doi:http://dx.doi.org/10.1016/j.concog.2005.11.005.

Sik, Hong You; et al. (1999). Estimation of optimal green time simulation using fuzzy neural network. In *The 1999 IEEE International Fuzzy Systems Conference, FUZZ-IEEE'99*, pages II–761–II–766. Seoul, South Korea.

Sims, A.G.; Dobinson, K.W. (1980). The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on Vehicular Technology*, 29(2):130 – 137. doi:10.1109/T-VT.1980.23833.

Sloman, A. (1998). Damasio, descartes, alarms, and meta-management. In *In Proc. of the IEEE International Conf. on Systems, Man, and Cybernetics*, pages 2652–2657.

Sloman, A. (2001). Beyond shallow models of emotion. In *Cognitive Processing*, volume 2, pages 177–198.

Sloman, Aaron (2011). Varieties of meta-cognition in natural and artificial systems. *Metareasoning: Thinking about thinking*, pages 307–23.

Sodian, Beate; Kristen, Susanne (2010). Theory of mind. In *Towards a theory of thinking*, pages 189–201. Springer.

Srinivasan, Dipti; Choy, Min Chee; Cheu, Ruey Long (2006). Neural networks for real-time traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 7(3).

Steels, Luc (2015). *The talking heads experiment*. Computational Models of Language Evolution. Language Science Press.

Stevanovic, J. (2007). *Stochastic Optimization of Traffic Control and Transit Priority Settings in VISSIM*. Department of Civil and Environmental Engineering, University of Utah.

Sun, R (2003). A tutorial on clarion 5.0. http://www.cogsci.rpi.edu/public_html/rsun/sun.tutorial.pdf.

Sun, Ron (2006). *Cognition and Multi-Agent Interaction*, chapter The CLARION cognitive architecture: Extending cognitive modeling to social simulation. Cambridge University Press.

Sun, Ron; Wilson, Nick (2011). Motivational processes within the perception–action cycle. In *Perception-Action Cycle*, pages 449–472. Springer.

Sun, Ron; Zhang, Xi; Mathews, Robert (2006). Modeling meta-cognition in a cognitive architecture. *Cognitive Systems Research*, 7(4):327–338.

Sánchez-Medina, Javier J.; Galán-Moreno, Manuel J.; Rubio-Royo, Enrique (2010). Traffic signal optimization in "la almozara" district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):132–141.

Tononi, Giulio (2008). Consciousness as integrated information: a provisional manifesto. *The Biological Bulletin*, 215(3):216–242.

Tulving, E. (1991). Concepts of human memory. In L. Squire; G. Lynch; N.M. Weinberger; J.L. McGaugh, editors, *Memory: Organization and locus of change*, pages 3–32. Oxford Univ. press.

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, 53:1–25.

Tyrrell, T. (1994). An evaluation of maes's bottom-up mechanism for behavior selection. *Adaptive Behavior*, 2(4):307–348. doi:10.1177/105971239400200401.

Viti, Francesco; van Zuylen, Henk J. (2010). A probabilistic model for traffic at actuated control signals. *Transportation Research Part C: Emerging Technologies*, 18(3):299 – 310. ISSN 0968-090X. doi:http://dx.doi.org/10.1016/j.trc.2009.05.003. 11th {IFAC} Symposium: The Role of Control.

Vogt, Paul (2015). How mobile robots can self-organise a vocabulary. *Computational Models of Language Evolution*.

Wahle, Joachim; et al. (2002). A microscopic simulator for freeway traffic. *Networks and Spatial Economics*.

Xie, Xiao-Feng; et al. (2012). Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies*, 24:168 – 189. ISSN 0968-090X. doi: http://dx.doi.org/10.1016/j.trc.2012.03.004.

Zanin, Rodrigo Bruno (2004). *Metodologia Automática Para Extração de Cruzamentos de Rodovias em Imagens de Alta Resolução*. Master's thesis, Unesp.

Zhao, Dongbin; Dai, Yujie; Zhang, Zhen (2012). Computational intelligence in urban traffic signal control: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):485–494. ISSN 1094-6977. doi:10.1109/TSMCC.2011.2161577.