Biologically Inspired Cognitive Architectures xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Biologically Inspired Cognitive Architectures

journal homepage: www.elsevier.com/locate/bica

Research article

An urban traffic controller using the MECA cognitive architecture

Ricardo Gudwin[a,*], André Paraense[a], Suelen M. de Paula[a], Eduardo Fróes[a], Wandemberg Gibaut[a], Elisa Castro[a], Vera Figueiredo[a], Klaus Raizer[b]

[a] University of Campinas (UNICAMP), Campinas, SP, Brazil
[b] Ericsson Research Brazil - Indaiatuba, SP, Brazil

ARTICLE INFO

Keywords:
Cognitive architecture
Dual-process theory
Dynamic subsumption
CST

ABSTRACT

In this paper, we present a Cognitive Manager for urban traffic control, built using MECA, the Multipurpose Enhanced Cognitive Architecture, a cognitive architecture developed by our research group and implemented in the Java language. The Cognitive Manager controls a set of traffic lights in a junction of roads based on information collected from sensors installed on the many lanes feeding the junction. We tested our Junction Manager in 4 different test topologies using the SUMO traffic simulator, and with different traffic loads. The junction manager seeks to optimize the average waiting times for all the cars crossing the junction, while at the same time being able to provide preference to special cars (police cars or firefighters), called Smart Cars, and equipped with special devices that grant them special treatment during the phase allocation policies provided by the architecture. Simulation results provide evidence for an enhanced behavior while compared to fixed-time policies.

1. Introduction

The concept of *Smart Cities* appeared during the Smart Growth movement in the late 1990s (Harrison & Donnelly, 2011), where new policies for urban planning were addressed. Among the many kinds of services conceivable within the "smart cities" paradigm, one of paramount importance is urban traffic control. In big cities, increasing traffic intensities are constantly creating huge traffic jams and the standard approach for managing traffic lights (fixed-time controllers) are not being able to maintain the situation under reasonable parameters anymore. Adaptive traffic controllers, which measure current traffic conditions and try to employ dynamic policies, are available but still without a standard approach being employed in real cases. The problem is very complex, and even if the use of computational intelligence can help improving efficiency, in particular cases, there are many reasons to believe that a proper solution might require the use of scalable strategies, which might start by first employing simple local solutions, followed by increasing scale-ups including new capabilities, in order to improve the solution, as new details and strategies become available, and are able to be incorporated in the solution. Following (Kelaidonis et al., 2012), we propose that this boosting technology in dealing with the problem might be the use of *Cognitive Managers* (see Section 3) as a general abstraction in dealing with *Smart Cities* problems.

The general approach is to model the problem as a *System of Systems* (SoS) (Gorod, Sauser, & Boardman, 2008). It is important to differentiate here a *System of Systems* from simply a *System of Subsystems*. Any traditional system can be understood as a system composed of subsystems or even a Family of Systems (FoS). Systems of systems, on the contrary, are not just a larger version of the same old hierarchical structure. A system of systems is "open at the top", "open at the bottom" and "continually evolves - but slowly enough to be stable" (Abbott, 2006). What characterizes an SoS is that new SoS parts are continuously joining and leaving the SoS, establishing a dynamic configuration to it, which easily might become very complex. Our approach proposes that each part joining the SoS should be controlled by a Cognitive Manager, responsible for internally controlling its own subsystem, with its own performance measures, and externally interacting with the SoS, asking and offering collaboration to other peers in the SoS. The Cognitive Manager should be intelligent enough to decide how much to compromise, in its seek for optimization of its performance metrics, in order to collaborate with requests coming from other SoS peers, and also how much and when to ask for collaboration from other peers in order to enhance its own performance indexes. For the construction of such

* Corresponding author.
E-mail addresses: gudwin@dca.fee.unicamp.br (R. Gudwin), paraense@dca.fee.unicamp.br (A. Paraense), suelen@dca.fee.unicamp.br (S.M. de Paula), edufroes@dca.fee.unicamp.br (E. Fróes), wgibaut@dca.fee.unicamp.br (W. Gibaut), ecalhau@dca.fee.unicamp.br (E. Castro), verafeec@dca.fee.unicamp.br (V. Figueiredo), klaus.raizer@ericsson.com (K. Raizer).

https://doi.org/10.1016/j.bica.2018.07.015
Received 4 July 2018; Received in revised form 11 July 2018; Accepted 11 July 2018

Cognitive Managers, we understand that simpler computational intelligence techniques might not be enough. We might require something more robust and eclectic, like a cognitive architecture.

In this paper, we present a first contribution to the subject, describing a small SoS, composed by a set of *Junction Cognitive Managers*, each of them controlling the traffic lights of a single junction in a city topology, a *Car Cognitive Manager*, a simple smartphone application which can turn any car (in principle, special cars, like police or firefighters car) into a *Smart Car*, and a *Coordinator Cognitive Manager*, which receives Smart Car requests for special treatment, identify a possible Junction Cognitive Manager which is closer enough to the Smart Car in order to collaborate with it, and send a reference of it to the Car Cognitive Manager. After that, the Car Cognitive Manager starts talking directly to the Junction Cognitive Manager and, if possible, the Junction Cognitive Manager might collaborate and give priority access to the Smart Car in its control of its traffic lights. In our implementation, both the Car Cognitive Manager and the Coordinator Cognitive Manager were simplified in order to allow our simulations to become easier. The Junction Cognitive Manager, though, was constructed using MECA, the Multipurpose Enhanced Cognitive Architecture (Gudwin et al., 2017, 2018), a cognitive architecture being developed by our research group. Our intention here is not to simply solve the traffic problem, but to show the potential of MECA for building other kinds of Cognitive Managers in other Smart Cities problems and evidence the convenience of using cognitive architectures like MECA to build and manage Systems of Systems. Our experimental results show a decrease in waiting time which is comparable to other computational intelligence techniques (see Section 2), but because we are using a cognitive architecture, it is scalable to include in the future other enhancements, like the communication between neighbor traffic lights.

In Section 2 we introduce the problem of urban traffic control, providing an overview of the state of the art in the field. In Section 3 we introduce some background about the construction of cognitive managers and how they can be useful in building solutions for smart cities technology. In Section 4, we explain how MECA can be used in order to build a cognitive manager, and in Section 5 we describe the details of our cognitive manager for the case of urban traffic control. In Section 6 we present our results and in Section 7 the conclusions.

2. State of the art in urban traffic control

After the first traffic signals controlled by human brains (police officers), 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, Staff, & Albeck, 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 Sydney Co-ordinated Adaptive Traffic System (SCATS) (Sims & Dobinson, 1980) dynamically adjusts common cycle length of a given traffic network (or sub-network) to meet the traffic demand. The SCOOT (Split Cycle and Offset Optimization Technique) 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, Diakaki, Dinopoulou, Kotsialos, & Wang, 2003).

Some examples of applications in real-time traffic-responsive control in major cities are:

- New York City: 7660 (of a total of 12,460) signalized intersections are controlled by a central computer network;
- Toronto: 83% of its signals are controlled by the Main Traffic Signal System (MTSS). 15% also use the SCOOT method.
- Sydney: 3400 traffic signals co-ordinated by the SCATS. Designed and developed by RTA, the system was first introduced in 1963 and progressively developed since then. By October 2010, SCATS was licensed to 33,200 intersections in 144 cities across 24 countries worldwide, including Singapore, Hong Kong, Dublin, Tehran and Minneapolis and Detroit.
- Melbourne: 3200 traffic lights across Victoria, including regional areas such as Geelong and Ballarat, using SCATS. Some 500 intersections also have tram and bus priority.
- Adelaide: 580 sets of coordinated traffic lights throughout the metropolitan region managed by the Adelaide Coordinated Traffic Signal (ACTS) System.

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, Dai, & Zhang, 2012).

Some approaches consider the problem only locally, restricted only to a small number of traffic lights (Sik, Soo, Kwang, & Kug, 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 sub optimal decisions (Viti & 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, Choy, & Cheu, 2006). Hence, how to achieve scalable network-wide optimization remains a challenging problem. Some methods try to produce more optimal solutions considering a given 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, Farges, & Tuffal, 1983), COP (Sen & Head, 1997), ALLONS-D (Porche & Lafortune, 1999), OPAC (Gartner, Pooran, & Andrews, 2002), ADPAS (Kim, Park, & Baek, 2005) and CRONOS (Boillot, Midenet, & Pierrelee, 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, Smith, Lu, & Barlow, 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, Ho, & Fung, 2005; Heydecker, Cai, & Wong, 2007), neuro-fuzzy networks (Choy, Srinivasan, & Cheu, 2003) and reinforcement learning (Cai, Wong, & Heydecker, 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 and 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.

3. Cognitive managers

Kelaidonis et al. (2012) introduced the notion of a *Cognitive Manager*, as a special kind of agent managing a set of physical objects made available at Internet due to a process of virtualization. This virtualization consists in plugging-in physical objects from the real world to the Internet, allowing them to provide information about themselves and in some cases receiving commands meant to cause some change in the object internal state or a mechanical action in the physical world. A Virtual Object (VO) is then the representation of a physical object at the Internet, providing information about it and possibly allowing commands affecting its behavior in the physical world. In this sense, physical objects like cars, buses, trains, traffic lights, doors, ordinary lights, air conditioners, and many other devices might provide information and be (partially) remotely operated through Internet. Also, public spaces like gardens, parking places, stations, libraries, and governmental offices, might provide information such as images and sound, and have themselves be managed in terms of access control, environmental control and resource availability by the Internet.

Even though the existence of VOs brings an interesting scenario, allowing us to grab information from different objects at a city environment, in general the situation starts to become more interesting when many different VOs are mashed up into *Composite Virtual Objects* (CVO) (Kelaidonis et al., 2012), which together provide some sort of functionality, like e.g. the urban traffic control of multiple traffic lights in a city environment. Each junction of streets/roads can be seen as a whole system of VOs, where each traffic light is a VO, each induction loop is also a VO, and all of them together can be seen as a CVO system. One important difference between standard VOs and CVOs is that usually a CVO has a dynamics in time, requiring some sort of control. In this sense, a CVO Manager usually employs control rules opening and closing traffic lights, such that the traffic might flow. These control rules might be fixed, and in such a case a CVO might only offer traffic statistics information, which might be useful for transit authorities. But these control rules might be adaptive, and in this case, the CVO Manager might accept requests for changing its control strategy, in specific cases. Now, considering an SoS scenario, different CVO managers might interoperate, requesting the collaboration of other systems. Imagine for example, a CVO comprising the many systems available in a car, managed by a car CVO manager, and a junction of streets (traffic lights and induction loops on a crossroad), managed by a junction CVO manager. The car CVO manager might require the collaboration of the junction CVO manager to facilitate its passing through the traffic lights. The junction CVO manager might decide to collaborate or not with the car CVO manager, depending on its traffic conditions. In a light traffic situation, (or if the car is an emergency vehicle like an ambulance, or a police car), the junction CVO manager might change the traffic lights in order to facilitate the car crossing the junction with minimal delay. Also, neighbor junction CVO managers might exchange information requesting some sort of collaboration with their neighbors in order to dismiss a traffic jam.

The full specification for the behavior of a CVO manager can be, though, very challenging. To provide the standard properties expected from an SoS (autonomy, belonging, connectivity, diversity and emergence) some sort of enhanced intelligence from the part of CVO managers might be necessary. Following Kelaidonis et al. (2012), we propose that this should be solved by making these managers *cognitive*.

The term *cognitive* here is used in association with a technology that operates inside a complex environment, observes it, makes behavior choices, and receives feedback from it, all the while learning – assembling a data set that will help determine future behaviors based on past and current feedback (Thomas, Dasilva, & Mackenzie, 2005). This behavior might include interacting with other cognitive managers, asking for their help, and also attending requests for help from still other cognitive managers.

An example of such a scenario is presented in Fig. 1. Here, different cognitive managers (CMs) join and leave the *Smart City* SoS. Some of these are continuously managing a set of VOs (e.g. a junction CM managing the different traffic lights and induction loops comprising a junction crossroad), with goals (e.g. the goal of minimizing the average trip time of all cars crossing the junction) and performance indexes (PIs) to meet (e.g. a traffic quality index being measured at each junction). We might have a house CM, providing access control, air conditioned control, illumination control, etc. Others, like car CMs, house CMs or people running their smartphones (which also work as CMs), might appear and disappear as required, entering and leaving the

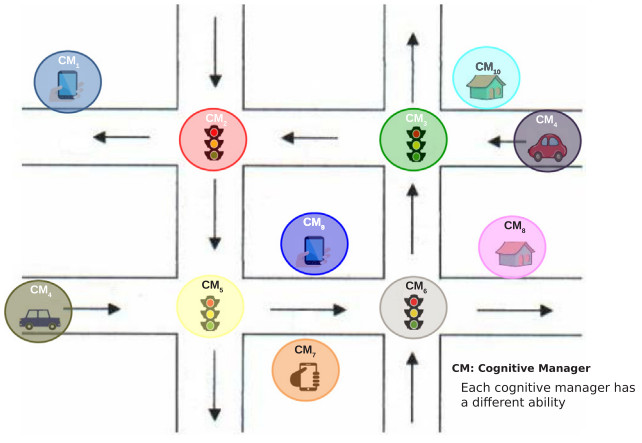


Fig. 1. Examples of cognitive managers.

Smart City SoS and asking for collaboration regarding their own goals.

The construction of a *Cognitive Manager* might require a special strategy in order to provide *cognitive* capabilities to these managers. In our case, we propose that these *cognitive* abilities can be gained with the aid of a *cognitive architecture*, in this case, the MECA Cognitive Architecture (Gudwin, 2017). A cognitive architecture can be viewed as a computational architecture suitable to bring cognitive abilities to an artificial agent. Cognitive architectures (Kotseruba, Gonzalez, & Tsotsos, 2016) are developed using as a source of inspiration different models for cognitive capabilities human beings are known to possess, like perception, memory, internal representation of the external world and the ability to use this internal model to perform simulations predicting future scenarios, making plans of action and executing those plans in order to satisfy the system's goals and/or other motivations.

The cognitive abilities required for a particular cognitive manager might depend on the VOs under their supervision, its goals and motivations, its performance indexes and its possibilities for acting on the real world. In our case, we used the CST toolkit (Paraense, Raizer, de Paula, Rohmer, & Gudwin, 2016) and the MECA cognitive architecture (Gudwin, 2017) to construct a cognitive manager. An overview of the MECA cognitive architecture can be seen in Fig. 2.

4. Designing a cognitive manager with MECA

The MECA Cognitive Architecture provides many features for building cognitive capabilities into a CVO Manager. The exploitation of these features depends on the complexity envisioned for the manager. First of all, it is necessary to analyze the VOs managed by the CVO manager, in order to evaluate the potential information they are able to provide, and the possible commands which can be sent to them. This will allow the definition of the available Sensory and Motor Codelets to be included in MECA, together with the Memory Objects to be created by the Sensory Memory and those used as actuators by the Motor Codelets (see Section 5 for an example).

4.1. Designing the behavioral system

After sensors and actuators are defined, the next step is to build up the Behavior Subsystem. The MECA architecture is split by design into two separate but complementary subsystems: System 1 and System 2,

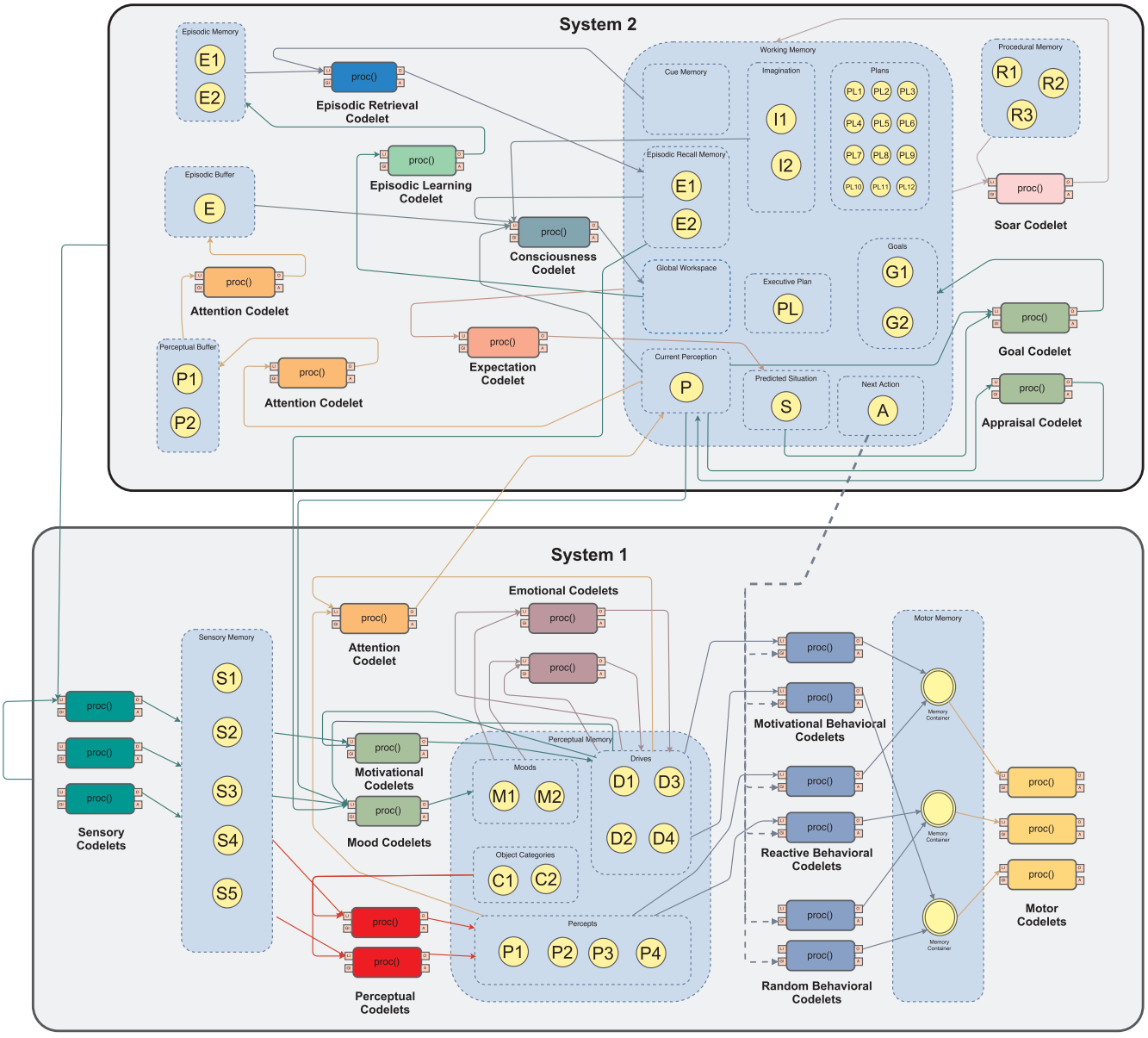


Fig. 2. The MECA cognitive architecture.

following cognitive models for dual-process theories. According to these theories, System 1 is responsible for specialized and automatic behavior, while System 2 is responsible for goal-based, deliberative behaviors. These two subsystems are intertwined, and the deliberations of System 2 are able to interfere with the automatic behaviors at System 1, generating a hybrid overall behavior, partially automatic, partially deliberative. In MECA, System 1 is implemented by a Subsumption architecture (Brooks, 1986), where three different kinds of behaviors are possible:

- Random Behaviors.
- Reactive Behaviors.
- Motivated Behaviors.

In a subsumption architecture, all behaviors are generated in parallel, and they compete with each other in order to have access to the actuators. So, at each time, all these three kinds of behaviors are generated, and evaluated given the actual situation and its context.

In simpler situations, where the CM must provide only a well known automatic behavior, only Reactive Behavioral Codelets are necessary. Reactive Behaviors are those which can be determined by a simple function or algorithmic procedure over the inputs. So, the actuator values are calculated as a function of the sensory inputs. The CM designer must look at the Memory Objects at the Motor Memory and delineate the set of different behaviors which might be able to determine their value.

Different options might be considered. If different behaviors affect the same actuators, Container Memories might be used such that the many behaviors are allowed to compete and only the winner will be able to affect the Motor Codelets.

In some situations, the designer might opt to include Random Behaviors, in cases where only Reactive Behaviors are not enough to generate a solution. In a Reactive Behavior, Codelets inputs are used to verify if the conditions for running the behavior are met. If these conditions are not met, the evaluation of the codelet outputs receives a very low score, such that the behavior is not chosen. In situations where no Reactive Behavior conditions are met, it might be reasonable to have Random Behaviors generating an unexpected output.

This happens also in human behavior. When we don't know what to do, we start doing random moves, trying to move ourselves out of the stuck situation, such that we are able again to decide what to do. The use of Random Behaviors might move the system state to a different condition, where other Reactive Behaviors are suitable, and then the system might recover for example from a repetitive behavior where it got stuck.

Even though Brooks proved that complex behaviors can be generated through the competition of Reactive Behaviors alone, there is a limit on what can be expected from only Reactive (and eventually Random) Behaviors. Reactive behaviors are *blind*. If the conditions for triggering them are met, they are executed, without any evaluation on the possible results to the system situation. If these conditions are well defined, Reactive Behaviors can be quite effective. But the meeting of any performance index is just a casual offspring of a fortunate set of behaviors, chosen by the designer experience on the possible outcomes of the system actions (or by learning, if a learning system is used to set up a set of reactive behaviors). The system, in itself, is not using any performance index for taking decisions. It is just reacting in a prescribed way, following a script given the sensed situation. Reactive Systems might be defined by a simple (or fuzzy) rule system, where rules match a condition into a given action, or similarly, using a neural network (e.g. a multi-layer Perceptron), where a learning system might be used to train this neural network.

During the design of a Cognitive Manager, though, in many cases it is possible to define one or more performance indexes, which the user wants to optimize. In MECA, these performance indexes can be used to determine a Motivated Behavior, giving rise to *Drives*. While in pure Reactive Systems, actions are chosen as a mere function of the sensed inputs, in Motivated Behavior, actions are chosen in order to optimize a *Drive*.

The concept of motivational behavior in CAs has its inspiration in studies about *Human Motivation*, realized by Hull (1943) and Maslow (1943) in Cognitive Psychology. According to Hull's theory of behavior (Hull, 1943), when a motor action is a pre-requisite to optimize the probability of survival of either an individual or a species, we say that there is a state of needness. This need motivates or drives the associated motor action. So, Hull defines a Drive as being a variable used to characterize a need. Drives are used as a measurement of a performance index, which the creature must optimize, to survive. In a living being, a drive might be related to the many needs, such as the need for food, for water, for air, the need to avoid injury, to maintain an optimal temperature, the need to rest, to sleep, and to mate. In an artificial agent, drives are just performance indexes the agent must optimize, being directly associated with the desirable behaviors we want the agent to manifest. They are involved with a desirable future state for the agent, where drives values are minimized or maximized. So, a drive can be seen as the measurement of the agent's success in achieving a purpose. Also, a behavior which is performed to satisfy a drive is said to be a motivational behavior or a motivated behavior.

Following Hull's ideas, Sun (2003) proposed a motivation theory to be used within a Cognitive Architecture. According to Sun (2009), humans, animals or cognitive agents need to follow important criteria in daily activity to survive in any environment: sustainability, purposefulness, focus and adaptivity. Every agent must attend its physiological needs (sustainability), such as hunger, thirst, avoid physical dangers and others. Consequently, its actions must be based on its purposes, instead of a random choice (purposefulness) (Anderson, 2014; Hull, 1943; Sun, 2009). Besides that, it must focus its activities such that its needs and purposes are consistent, persistent and continuous (focus) (Sun, 2009; Toates, 1986). However, there may be temporary or permanent situations in which the agent might enter into a state of urgency (Sloman, 1987; Sun, 2009), while a special behavior might be appropriate. Finally, the agent must be able to adapt its behaviors according to the objectives to get the best results (Sun, 2009). These criteria, facilitate understanding that motivational dynamics in animals or humans are a crucial part of composing a final behavior (Sun, 2009).

Both theories suggest that motivations are used as a reference signal in a feedback control system, guiding and optimizing human behaviors in order to achieve the best results considering its internal needs. Therefore, needs motivate humans and animals to decide the best choice of action such that these internal needs can be suppressed and a desired state reached.

The notion of drive is very important for understanding another critical cognitive capability: emotions (Canamero, 1997). There is an intrinsic relationship between motivations and emotions. The concept of emotions came from cognitive psychology and philosophy, as an alternative way to address the problem of behavior generation (Bates, 1994; Budakova & Dakovski, 2006; Canamero, 1997, 1998; Meyer, 2008; Picard, 1997; Reilly, 1996; Septseault & Nédélec, 2005). There is no consensus about what emotions really are. Different approaches have different views for what they are and how to model them. For example, Ortony, Clore, and Collins (1990) understand 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) and Sloman (2001), in turn, understands emotions as internal "alarms" which give a momentary emphasis to certain groups of signals. Damasio (1994) and Damasio (1999) distinguish between "emotions", which affect the body and "feelings", which are a cognitive introspection of emotion. Other authors have completely different views about what emotions are. For example, to Canamero (1997), emotions work like "amplifiers" for motivations, working as homeostatic processes related to physiological variables.

In MECA, drives are memory objects generated by *Motivational Codelets* in two ways: (i) directly from sensory variables or (ii) derived from other existing drives in the agent. MECA relies on Canamero (1997) model of emotions where emotions are viewed as cognitive distortions on the map of drives, changing for a given time the relative intensity of the different drives on the system, amplifying some drives and decreasing the intensity of others. For that purpose, we include another parameter in our conception of drive: *emotional distortion*, which is a value (positive or negative) which is summed with the *activity level* of a drive, in order to prioritize the motivated behaviors.

Emotional codelets are modulated by *Moods*. Moods are a kind of emotional state which is determined by *Mood Codelets*, based on sensory data acquired from the environment. Depending on being in a *normal* mood, or on an alternate mood like *sleepy*, *worried*, *terrified*, and *in love*, the emotional codelet might determine a cognitive distortion to the drives landscape, making that a different priority is used to select motivated behavior.

We can formalize the concept of drive as:

Definition 4.1. A *drive* d is defined as a tuple $d = \{A, \theta, \delta, p\}$ where:

- A is the *activity level* representing the intensity of the drive, $A \in [0, 1]$
- θ is the *urgency threshold* such that if $A > \theta$ then the drive is in a state or urgency, $\theta \in [0, 1]$.
- δ is the *emotional distortion* which should be added to A in order to compute the intensity of the drive, $\delta \in [-1, 1]$ and $0 \leqslant (A + \delta) \leqslant 1$.
- p is the *priority*, $p \in [0, 0.5]$, which is used to assign a fixed priority while in urgency mode.

Given the definition of *drive* the calculation of the *Eval* value in the Memory Objects (which are generated by the Motivational Behavioral Codelets) is computed in the following way:

Definition 4.2. The calculation of the *Eval* parameter of a Memory Object generated by a Motivational Codelet, should adopt the following criteria:

$$Eval = \begin{cases} 0.5 + p & \text{if } A + \delta > \theta, \\ (A + \delta)/2 & \text{if } A + \delta \leqslant \theta. \end{cases} \quad (1)$$

If we follow Definition 4.2 we will see that while in normal mode, Eval will be a number between 0 and 0.5, while if in urgency mode, Eval will be a number between 0.5 and 1. Using this convention, we have a warranty that while in a state of urgency, the drive with the biggest priority will always be selected by the dynamic subsumption mechanism. This is depicted in Fig. 3.

If one or more performance indexes are available in our design of a CM, we might define a *Drive* for each of the PIs and a corresponding Motivated Behavior which is supposed to increase or decrease the *Drive*, depending if the PI is supposed to be maximized or minimized.

After designing Motivated, Reactive and Random Behaviors in System 1, it is also necessary to design the interference pattern which System 2 is supposed to cause in every behavior in System 1. According to the MECA architecture, System 2 is responsible for generating deliberative plans, which might interfere with System 1 Behaviors, in order to affect the actuators, and consequently the Motor Codelets. It is important to carefully fine tune how System 2 commands interfere with System 1 Behaviors. All the Behavior Codelets in System 1, together with their Local Inputs, have a Global Input. In MECA, this Global Input comes from System 2, and holds the *Next Action* prescribed by the deliberative plans from System 2. This fine tuning is important, because there is the risk of either System 2 or System 1 to systematically win the competition and suppress the effect of the other subsystem. If System 2 always have priority, it might just "turn-off" the effect of System 1, and work as a simple deliberative system. Otherwise, if System 2 is never allowed to win the competition, it is like it doesn't exist in the overall system. A good tune might give preference to System 1 and in exceptional situations allow System 2 to take control.

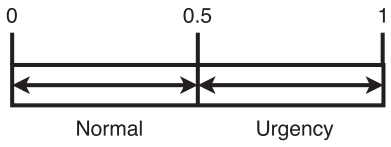


Fig. 3. Value of Eval while in normal or urgency mode.

A good rule in this case might be to use System 1 as the default controller, handling most of the decisions in the determination of actions, while in *normal* operational conditions. System 2 should be actioned only while exceptional conditions are detected, requiring an exceptional intervention. One example of such a duality in a Cognitive Manager might occur while trying a local optimization (managed by System 1 behaviors), when a call for a global optimization comes from another Cognitive Manager communicating with the current CM. This call for collaboration, in being an exceptional situation, should be handled by System 2. System 2 should then analyze its availability for collaboration, based on the current status of its performance indexes, and decide if it is able to collaborate. If its performance indexes allow, it might decide to collaborate, even though at the cost of a decreasing in its performance indexes, with the hope of improving the performance indexes of its solicitor, and with that improving some global (or higher level) performance index. If its performance indexes are in a poor condition (e.g. in a worse condition than those of its solicitor), the system might deny collaboration and System 1 orders should prevail.

4.2. Designing the perception system

After the Behavioral System is designed, there comes the time to design the Perceptual System, preparing the information necessary for the Behavioral System to perform its decision-making. Usually, the information coming from Sensory Codelets refers to *Quality Dimensions* collected from sensors, and related to physical properties of environmental objects. In some situations, these *Quality Dimensions* are enough for supporting decision-making by the Behavioral Codelets. In this case, there is no requirement for further abstractions. This data can be directly used by Behavioral Codelets as they come. But in more complex cases, like those requiring human understanding of complex situations, further abstractions are necessary. The Perception System is the subsystem within a Cognitive Architecture responsible for performing further abstractions, creating more elaborated representations which better describe the environmental situation.

The CM designer might evaluate which concepts might be required, analyzing the requirements of the Behavioral Codelets, and create Perceptual Codelets responsible for creating the representational entities required for further processing. Special care should be dedicated to the entities required by System 2 to make decisions. It is important to remember that MECA uses SOAR (Laird, 2012) as a planning system in System 2. In this sense, the many concepts generated by the Perceptual System will be converted to WMEs in order to be processed by SOAR, and after SOAR makes a decision, the resulting WMEs might be turned into appropriate Memory Objects in order to affect System 1 behaviors.

Also, it is important to define the Attention Codelets, and how these Attention Codelets select important concepts in the Perceptual Memory and copy them to the Working Memory, so that they can be used in System 2.

Depending on the situation, some effort might be necessary in constructing *Episodes* from *Current Perception* and storing them at the Episodic Memory. In this case, the designer might have also to determine how the Episodic Memory is supposed to be cued and how the episodes recalled from the Episodic Memory might be used in the deliberative process.

4.3. Designing the Working and Procedural Memories

In MECA, the design of both the Working Memory and the Procedural Memory are tied together. This occurs because MECA uses SOAR as its planning and execution system. This design starts with defining the format of the *Next Action* which is sent down to System 1 to interfere with the Behavior Codelets. Next, the designer must define the required input collected from System 1 Perceptual Memory and transformed into WMEs within SOAR's input link. Finally, the designer must develop the rules in the Procedural Memory.

5. CMs for urban mobility

The general concept of a Cognitive Manager presented in Section 3 is illustrated now in the context of urban mobility in a smart city.

We already pointed out in Section 3 that a smart city infrastructure can be seen as the construction of an SoS, where different services can be provided by the virtualization of physical objects from the real world (through cyber-physical systems) and their possible interaction in meshes forming CVOs which can be locally controlled by a CVO manager. We already argued that it might be interesting to give cognitive abilities to these managers, turning them into Cognitive Managers, and that these cognitive abilities might be constructed with the aid of cognitive architectures. In our case, we are using MECA as the cognitive architecture to construct these cognitive managers.

To exemplify all these steps, we provide, in this chapter, a simple but quite illustrative case where a set of cognitive managers, constructed with MECA, are put to interact and solve a common problem in urban traffic control, under the general scenario of a Smart City SoS.

5.1. The urban traffic control problem

In the overall SoS scenario, we consider the virtualization of physical assets from the real world into VOs and their management by a CM. We said that some VOs might occupy a fixed location in the real world, like e.g. traffic lights or induction loops, and other VOs might move around, like e.g. cars traveling into the urban environment or people carrying their smartphones. As a general rule, even though the communication of two VOs is possible, in order to provide more flexibility we might assume that only CMs are supposed to interoperate, asking for collaboration and receiving requests for service to be delegated to their VOs. In some particular situations, we might have a CM managing just one single VO. The role of the CM, in this case, is to enhance the VO with cognitive abilities (e.g., with the ability to dialog, integrating it into the SoS scenario of a Smart City).

We are assuming that one major CVO, integrates a controlled junction at an urban topology. In this case, the junction might have multiple traffic lights (each of them having its own VO), and sensors capable of providing information regarding the traffic running on the many lanes crossing the junction. A unique CM, the *Junction Manager*, controls the junction, operating the traffic lights and joining the junction in the Smart City SoS. An example of such a junction can be seen in Fig. 4.

The junction depicted in Fig. 4 has 16 lanes in 4 different edges, being L2–L4, L6–L8, L10–L12 and L14–L16 controlled lanes (because the traffic lights might command a stop on their flow), and L1, L5, L9 and L13 are uncontrolled lanes (because they just receive the flow coming from the other lanes - they do not generate traffic in themselves).

The situation described in Fig. 4 also shows a particular phase in the traffic lights, where L6 and L14 are open to traffic, and all the other lanes are closed. The traffic coming from L6 can either go to L1 or to L5. The traffic coming from L14 can either go to L9 or L13.

The design of just a single junction like this can be quite complicated, where a set of phases must be precisely defined in order to not allow the crossing of traffic flows, and each junction has its specific set

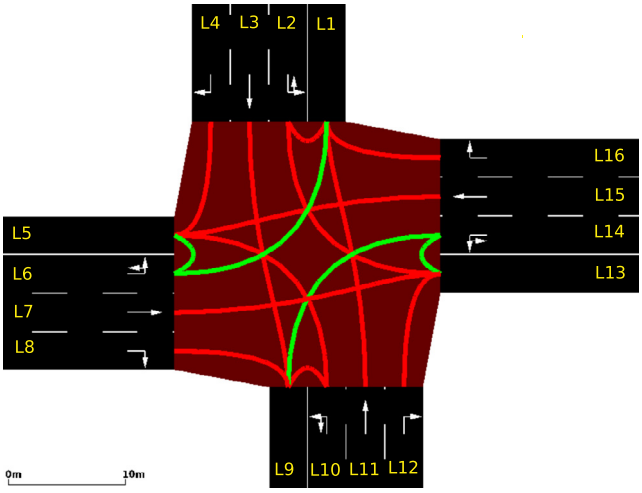


Fig. 4. An example of a junction.

of phases and a typical *sequence* of phases, which must usually be enforced by law. This means that the CM controlling the junction cannot arbitrarily assign traffic lights, or choose random phases at any time. There is also a minimum and maximum time that each phase must attend. Fig. 5 shows a little more elaborated case, with 2 junctions: West and East. Junction West has 4 different phases (as indicated in the figure) and junction East has only 3. The sequence of phases, in each case, must be obeyed.

In our case scenario, a junction cognitive manager locally controls each junction in a given topology, based on sensed information of traffic conditions in all the controlled lanes of the junction, and at each instant of time, decides if the junction should maintain its current phase or change to the next one in the prescribed sequence of phases for the junction, respected the minimum and maximum times allowed for each phase.

Our case also includes a second CM, the SmartCar Cognitive Manager, possibly an ambulance or a police car, which given its urgency in passing through the traffic lights, requests the collaboration of the Junction CM in changing its phase such that it might wait the minimum amount of time for crossing the junction. As soon as the SmartCar CM joins the Smart City SoS, it starts collaborating with a Localization CM, passing to it its current position, and if this position is close enough to the position of a junction controlled by a Junction CM, the Localization CM returns the Junction CM address to the SmartCar, and the SmartCar starts sending its position directly to the Junction CM.

5.2. The Junction Cognitive Manager

The architecture of a Cognitive Manager for a junction CVO,

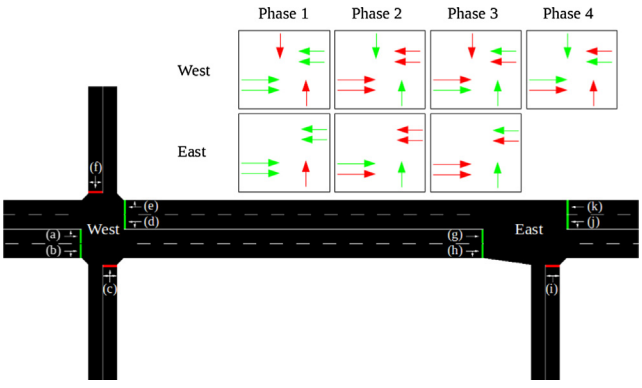


Fig. 5. A signalized small network and its phases.

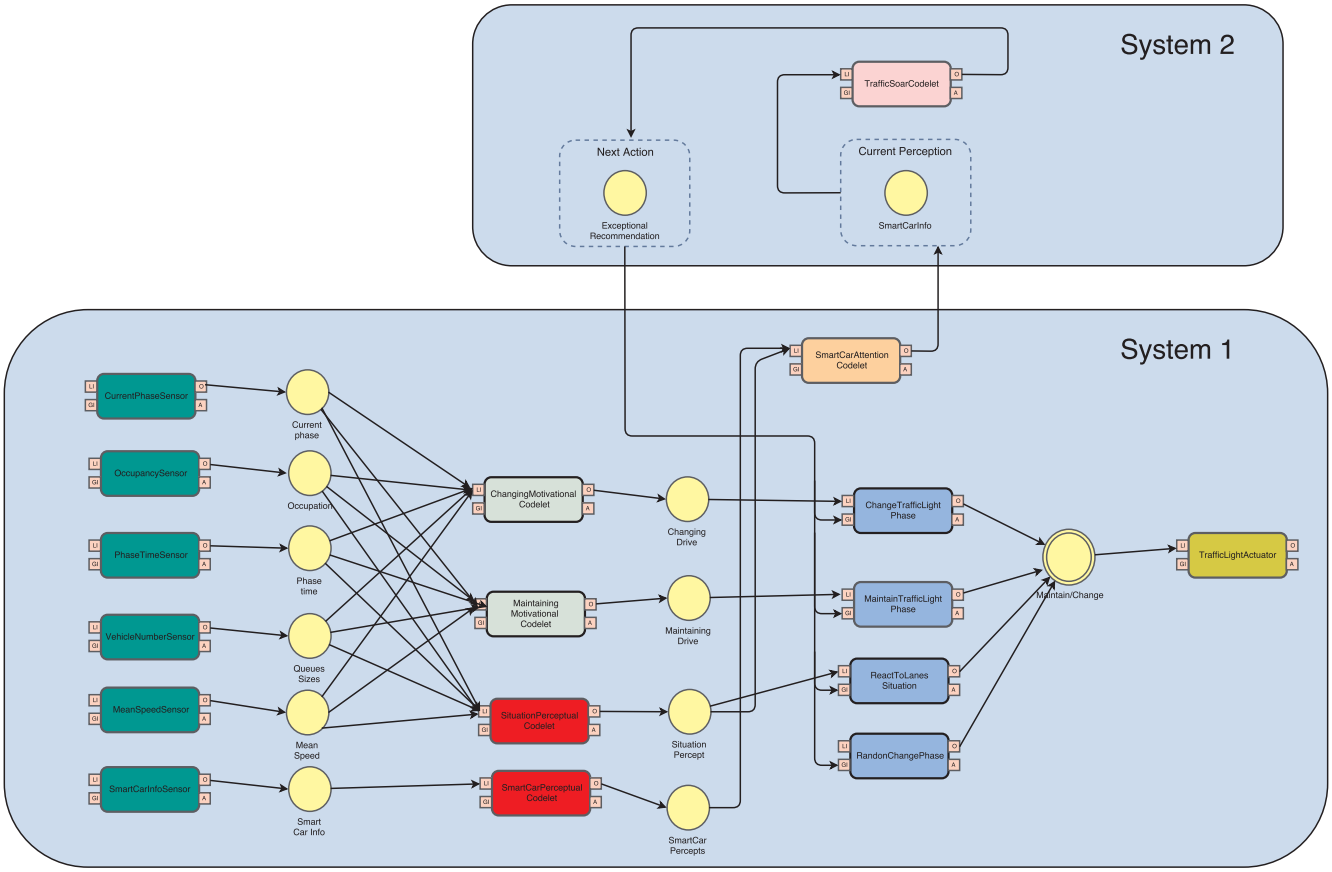


Fig. 6. The junction cognitive manager architecture.

constructed with the aid of MECA can be seen in Fig. 6.

The role of the Junction CM is to decide, at each time instant, if the junction should maintain its current traffic lights phase or change to the next phase. The junction CM is constructed with the aid of MECA, even though not all MECA modules were used. Basically, System 1 is used to provide a decision under normal conditions, where the junction wants to locally minimize the average travel time of all the cars crossing it. For that purpose, we developed a motivational system with two drives: the change drive and the maintaining drive, which from their perspective might evaluate, at each time, the pressure to maintain or to change the current phase, based on the statistics they collect. In our Junction CM, the role of System 2 is to analyze requests from SmartCars and if it is the case, to take the decision to collaborate with the SmartCar CM and change its behavior in order to minimize its travel time. It is important to emphasize that this has a cost, and will most likely make average travel time worse for all other vehicles. System 2 needs to decide what is feasible or not. If it has margins for that, i.e., it is not detecting a critical condition in the traffic flow, it will be more open to collaborate, but if the traffic conditions are critic, then it might decide not to collaborate. This policy, of course, is configurable, so it is possible, in principle, to always honor the request of a SmartCar (what we effectively did in our experiments).

In next subsections, we depict the details of the Junction CM. We follow the design guidelines detailed in Section 4, such that the reader might get an example of these guidelines being instantiated.

5.3. Sensor and actuator codelets

In this work, real world traffic is simulated with SUMO urban traffic simulator (Krajzewicz, Erdmann, Behrisch, & Bieker, 2012). For this purpose, sensor and actuator codelets are socket connections to SUMO using the TraCI protocol to interact with SUMO, acquire the required information and send the control signals to change the traffic lights phases. Depending on the simulation scenario, we will be running many junction CMs, one for each controlled junction in the scenario.

Because our purpose with this case was to generate a proof-of-concept, we decided to use some facilities from SUMO which might provide a better illustration for the potentiality of the approach. In a realistic case scenario, we should be using only input information from inductive loops installed in the junction. This would, however, restrict too much the possibilities of decision-making. In SUMO, we have the possibility of knowing the position of each car in each of the lanes. Even though it is not fully realistic to have this information in a real case (at least while cars are not enforced to provide this information by law), it is not unfeasible to have this information sometime in a near future. Therefore, the following information is collected by each junction:

Occupation: Average of the occupation of all lanes in the junction, ranging from 0% to 100%. The occupation of each lane depends on the length of the lane, the number of cars waiting in the queue and the size of the cars.
Phase Time: The amount of time since the last phase change.
Number of Vehicles: The number of vehicles in all lanes
Average Speed: The average speed of all vehicles crossing the junction.
Smart Car Info: This input receives messages from Smart Cars, informing their current position. If there is no Smart Car in the system, this input remains silent all the time.

There is just one Motor Codelet in the Junction CM. It receives information from a boolean Memory Object commanding the junction to maintain the current phase or to switch to the next phase. The Maintain/Change boolean Memory Object is modeled by a Memory Container (represented in Fig. 6 by a circle with a double line). This is

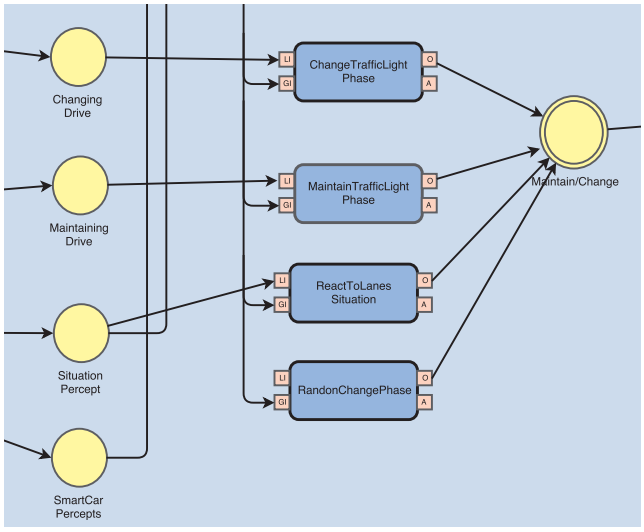


Fig. 7. The behavioral subsystem.

necessary because many different behaviors might propose different actuations. A Memory Container is a special resource provided in CST to allow the Dynamic Subsumption mechanism to choose the proposal with the highest evaluation to be considered and commanded to SUMO. This is performed by the associated Motor Codelet.

5.4. The Behavioral Subsystem

After the definition of sensors and actuators, the next step is to define the many behaviors comprising the Behavioral Subsystem. The repertoire of behaviors in System 1 can be seen in Fig. 7.

The behavioral subsystem provides 4 different behaviors proposing a *maintain/change* decision for the next phase, which might compete with each other:

- 2 Motivational Behaviors: Change Phase and Maintain Phase.
- 1 Reactive Behavior: React To Lanes Situation.
- 1 Random Behavior: Random Change.

5.4.1. Random Behavioral Codelet

In our Junction CM, the role of the Random Behavioral Codelet is to provide a measure of randomness to the decision-making process.

Definition 5.1. The *activation* A_{rand} of the Random Behavioral Codelet is calculated according to the following criteria:

$$A_{rand} = \begin{cases} 1 & \text{if } random.\ next() \geqslant 0.999, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Definition 5.2. The recommendation R_{rand} given by the Random Behavioral Codelet deciding if the Motor Codelet should change or not the current phase is based on its activation according to the following rule:

$$R_{rand} = \begin{cases} True & \text{if } A_{rand} = 1.0, \\ False & \text{otherwise.} \end{cases} \quad (3)$$

The practical effect is that in only 0.1% of the cycles the Random Behavioral Codelet will write to the Motor Codelet determining the change of the current phase. This value was completely arbitrary, and included just to insert some level of randomness in the system.

5.4.2. Reactive Behavioral Codelet

The logic behind the Reactive Behavioral Codelet is that, based on a *Traffic Situation Index* used as input, it should determine a change in the phase.

Definition 5.3. The Reactive Behavioral Codelet's activation is always determined by the traffic situation index TSI (see Definition 5.7) coming from the Situation Perceptual Codelet:

$$A_{react} = \{TSI. \quad (4)$$

Definition 5.4. The recomendation given by the Reactive Behavioral Codelet deciding if Motor Codelet should change or not the current phase is based on the following rule:

$$R_{react} = \begin{cases} True & \text{if } A_{react} \geqslant 0.5, \\ False & \text{otherwise.} \end{cases} \quad (5)$$

The practical effect is that everytime the traffic situation index is greater than 0.5, the Reactive Behavioral Codelet will prescribe the Motor Codelet to change the current phase.

5.4.3. Motivational Behavioral Codelets

Our Junction CM uses two Motivational Behavioral Codelets: the Change Traffic Lights, which always propose a change in the phase, and the Maintain Traffic Lights, which always propose to maintain the current phase. The activation of these codelets is determined by the *Drives* which are used as input to them, given by the Motivational Subsystem.

5.5. The Motivational Subsystem

The main role of the Motivational Subsystem is to calculate the drives to be sent to the Motivational Behavioral Codelets. Our Junction CM uses two drives: the Changing Drive and the Maintaining Drive. This is shown on the top of Fig. 8.

In order to calculate those Drives, our Junction CM has basically two motivational codelets: the Changing Motivational Codelet and the Maintaining Motivational Codelet. These codelets receive the many parameters affecting the junction situation and integrate them into two drives: The Changing Drive and the Maintaining Drive. The definition of a drive implies in the determination of the following parameters

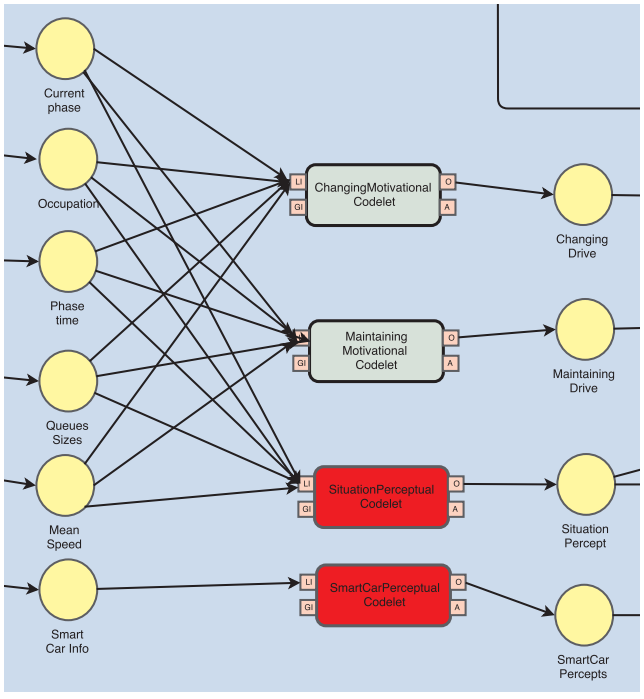


Fig. 8. Motivational and perceptual subsystems.

associated to the drive:

- Activation.
- Urgency Threshold.
- Priority.

In order to compute the drives activation, our motivational codelets employ an heuristics using both the *Phase Time* (θ), the time since the last change in phase and the ΔO, the difference between the average occupancy in all the lanes of the joint, computed in intervals of 8 s. If the *Phase Time* is less than the minimum phase time specified for the junction, the phase must be maintained (by specification). If ΔO is negative (or zero), this means that the traffic conditions are well balanced, and then the Maintaining Drive should be high, such that the phase should be maintained as is. If ΔO is positive, though, this means that the traffic conditions are becoming worse, and there is less and less indications that the phase should be maintained, and so the Maintaining Drive activation should decrease as long as time passes. Finally, if the phase time is greater than the maximum phase time specified for the junction, there is no more reason to maintain the phase, and then the drive should be set to 0. Definition 5.5 below formalizes this.

Definition 5.5. The *activation* A_m of the *drive* generated by the Maintaining Motivational Codelet is calculated by:

$$A_m = \begin{cases} 1 & \text{if } \theta \leqslant \theta_*, \\ 1 & \text{if } \theta_* \leqslant \theta \leqslant \theta^* \text{ and } \Delta O \leqslant 0, \\ \left(1-\frac{\theta}{\theta^*}\right)^4 & \text{if } \theta_* \leqslant \theta \leqslant \theta^* \text{ and } \Delta O > 0, \\ 0 & \text{if } \theta > \theta^* \end{cases} \quad (6)$$

where θ is the current *Phase Time*, θ_* is a constant that defines the *Minimum Phase Time* of the junction (8 s), θ^* is a constant that defines the junction's *Maximum Phase Time* (120 s), and ΔO is the difference between the average occupancy over all lanes in the junction, computed on intervals of 8 s.

The activation of the Changing Drive follows the same basic principles, formalized in Definition 5.6.

Definition 5.6. The *activation* A_c of the *drive* generated by the Changing Motivational Codelet is calculated by:

$$A_c = \begin{cases} 1 & \text{if } \theta > \theta^*, \\ 1 & \text{if } \theta_* \leqslant \theta \leqslant \theta^* \text{ and } \Delta O > 0, \\ 0 & \text{if } \theta_* \leqslant \theta \leqslant \theta^* \text{ and } \Delta O \leqslant 0, \\ 0 & \text{if } \theta \leqslant \theta_* \end{cases} \quad (7)$$

The *Priority* and *Urgency Threshold* of both drives are specified in Table 1.

5.6. The perception subsystem

Perception systems have the role of computing abstractions on the input sensory information, creating derivate knowledge which might be necessary for further processing in the cognitive system. In our Junction CM, we have two different Perceptual Codelets. The first one is the Situation Perceptual Codelet, which was already seen in Fig. 8. This Perception Codelet uses as input the occupation of all the lanes in the junction, the current phase time, the number of vehicles in all the lanes and the average speed in all the lanes and calculates a Traffic Situation Index for the junction. Even though all this information is available, the current implementation uses only the average speed of all the cars crossing the junction. The logic behind that is that the faster the cars are in the controlled lanes, the lower is the traffic situation index, signalizing a low traffic, a situation opposite to a traffic situation with an index closer to 1.0, where traffic is heavy.

Table 1
Drives' parameters.

Drive	Priority	Urgency threshold
Maintaining	0.5	0.967
Changing	0.45	0.3

Definition 5.7. The Situation Perceptual Codelet calculates the traffic situation index TSI based on the following rule:

$$TSI = \left\{1.0 - \frac{min(\overline{S}, S^*)}{S^*} \right. \quad (8)$$

where $\overline{S}$ is the average speed of all the vehicles running in the controlled lanes of the junction and S^* is the maximum speed, arbitrated as 16.66 m/s.

The other Perceptual Codelet is responsible for preparing information to be used by System 2, as shown in the bottom of Fig. 8. It creates a representation for a Smart Car, if one is present.

5.7. The System 2 subsystem

The role of System 2 in our Junction CM is to evaluate collaboration requests from Smart Cars and decide if the CM is able or not to collaborate with the Smart Car and change the traffic lights in a way that could minimize the travel time for the Smart Car, without disturbing too much the other vehicles. From an operational point of view, we are delegating this task to SOAR, as pointed out in Fig. 9.

In the simple scenario we started exploring, we will be dealing with just one situation: the Smart Car is an exceptional situation that must always be attended. The only restrictions, which must be followed, though are related to the minimum and maximum times for staying in a given phase. If the Smart Car arrives before the minimum time in the phase is reached, it will have to wait until this minimum time before it can be switched to the next one. This includes the minimum time holding for all the phases with a red light before one with a green light can be set. Figs. 10–14 present schematic situations being encoded into rules for addressing this situation.

Fig. 10 illustrates the situation covered in Rule 1, in which the Smart Car (drew as an ambulance in the figures) is arriving in a lane where the current phase is enforcing a red light, and the expected time of arrival (given the current Smart Car velocity) in the junction is less than the maximum phase time. Then the rule proposes the phase to change the phase, in order to switch the lane to a green light.

Fig. 11 illustrates the situation covered by Rule 2, in which the Smart Car is arriving in a lane where the current phase is enforcing a red light, but the expected time of arrival is greater than the maximum phase time. In this case, the system proposes to maintain the junction in its current phase.

Fig. 12 illustrates the situation covered by Rule 3, in which the Smart Car is arriving in a lane where the current phase is enforcing now a green light, and the expected time of arrival is less than the maximum phase time. In this case, the rule suggests the system to maintain in the

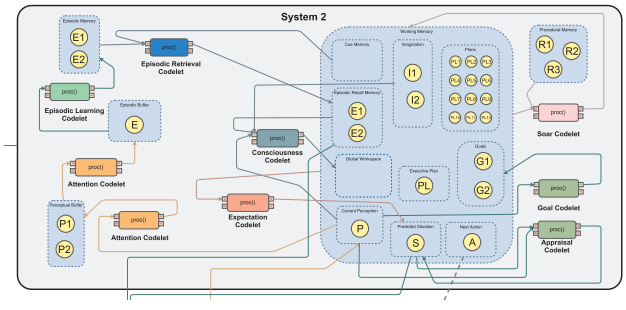
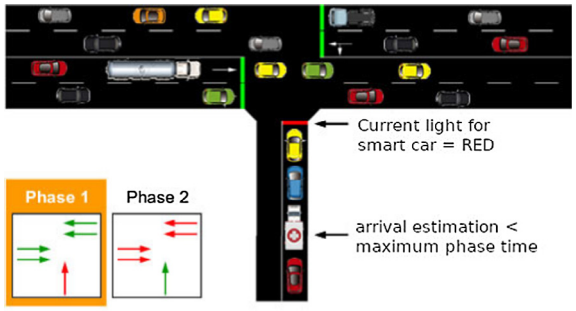
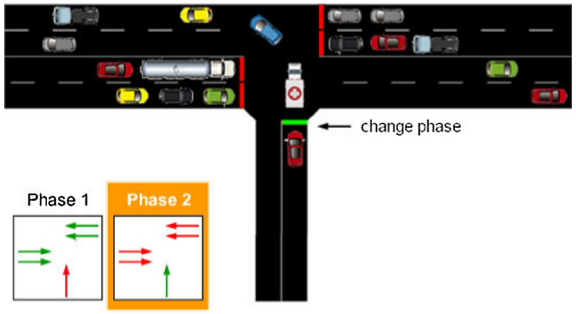


Fig. 9. The System 2 subsystem.

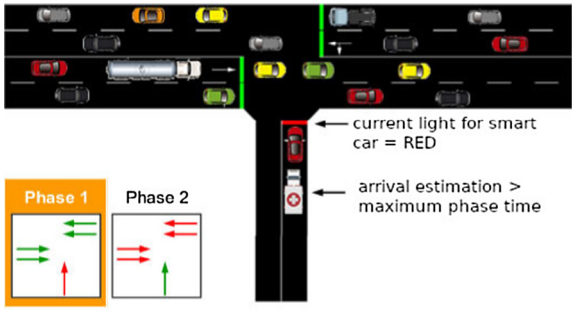


(a) Condition

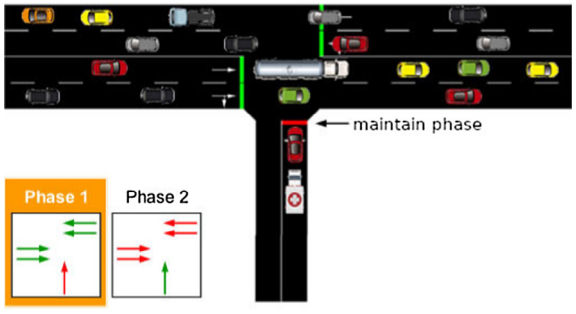


(b) Result

Fig. 10. Schematic situation being covered by Rule 1.

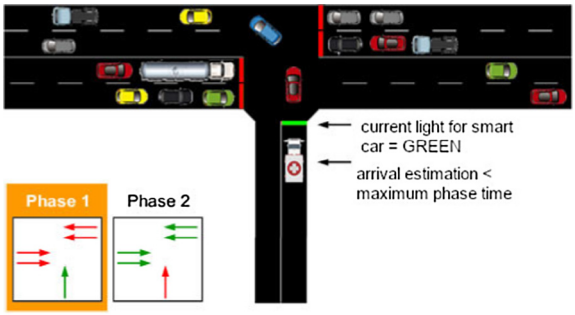


(a) Condition

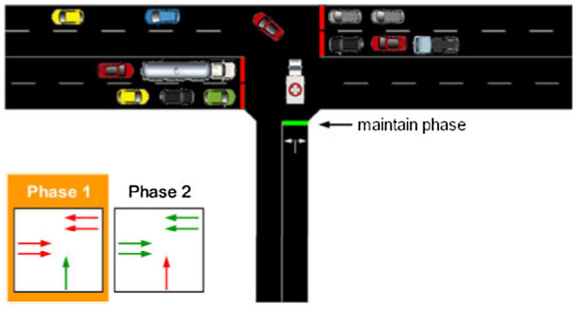


(b) Result

Fig. 11. Schematic situation being covered by Rule 2.

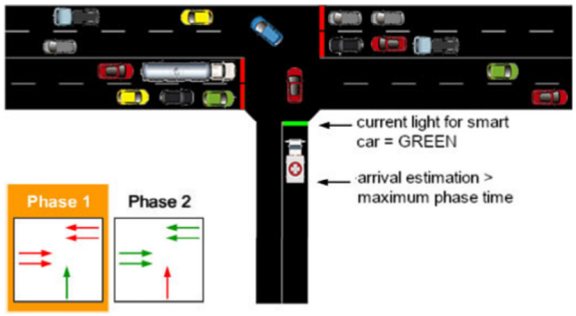


(a) Condition

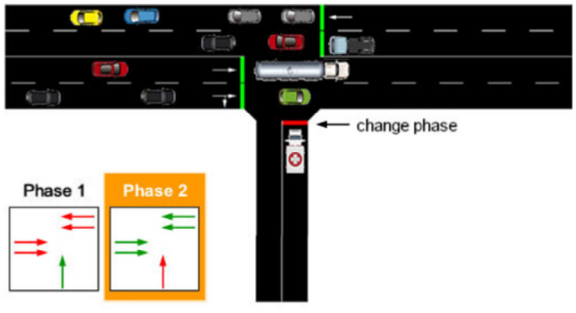


(b) Result

Fig. 12. Schematic situation being covered by Rule 3.



(a) Condition



(b) Result

Fig. 13. Schematic situation being covered by Rule 4.

current phase.

Fig. 13 illustrates the situation covered by Rule 4, in which the Smart Car is arriving in a lane where the current phase is enforcing a green light, but the expected time of arrival is greater than the maximum phase time. In this case, the rule suggests the system to change to the next phase.

Finally, Fig. 14 deals with the case of no Smart Car present in the scenario. In this case, the rule suggests a ⟨*null*⟩ command to be generated. A ⟨*null*⟩ command signalizes the Behavior codelets in System 1 to follow System 1 prescriptions instead of System 2 commands.

5.8. The Smart Car Cognitive Manager

In our simulation studies, the Smart Car Cognitive Manager was not implemented using a Cognitive Architecture, but this condition was

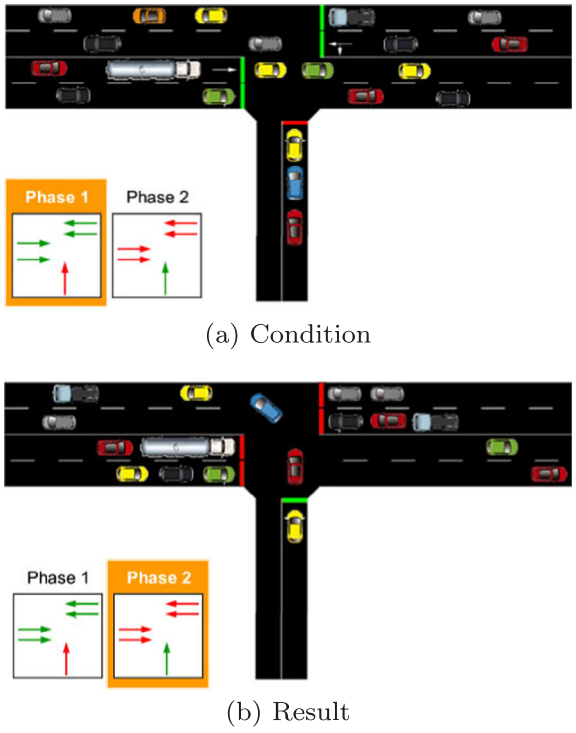


Fig. 14. Schematic situation being covered by Rule 5.

simulated using SUMO's features. In a SUMO simulation, we have labels for all the cars entering and leaving the simulation, information that can be consulted at every time using JTraCI. In order to simulate the Smart Car condition, we analyze the file *.rou.xml which lists all the vehicles supposed to enter into the simulation, and choose arbitrarily one of them to be a Smart Car, noting its ID parameter and providing this ID to the application managing the simulation.

Then, during the simulation run, the application managing the simulation detects if the smart car is already within the simulation, and in the case the car location is under a specific distance to a junction controlled by a Junction CM, it starts sending messages to the Junction CM, simulating the Smart Car sending this message. From the point of view of the Junction CM, it is as if the own Smart Car was sending the message. This message basically informs the car position, and the Junction CM calculates the expected time for the Smart Car to reach the junction. In a real implementation of this concept, the Smart Car CM might contact a Location CM, with the knowledge of all Junctions in the city able to receive a collaboration message, and if the Smart Car is within a range from Junction CM, it gives this information to the Smart Car and the Smart Car starts to send its position to the Junction CM, such that the collaboration is suitable to happen.

6. Simulation results

We made a series of simulations using our Junction CM, in many different scenarios and situations. Basically, the following 3 topologies where simulated:

- Simple T.
- Twin T.
- Corridor.

For each topology, 4 different traffic loads were tested, and each traffic load was run in 10 different simulations. We performed tests with and without a Smart Car. The simulation results are in the next subsections.

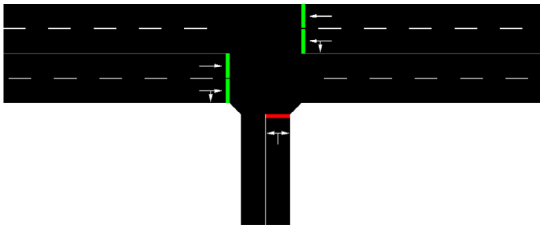


Fig. 15. SimpleT model.

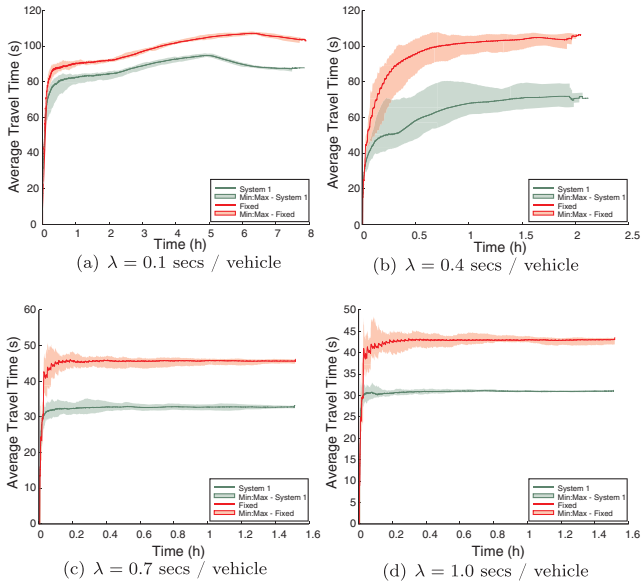


Fig. 16. Simulation results for the Simple T Scenario: The results show the average travel time along time during the simulation for different loads of traffic. For each case, 10 different simulations were performed. The graphics show the standard deviation among these 10 simulations. The simulation was performed using a fixed controller (red curve) and the Junction CM without any SmartCar (green curve).

6.1. The Simple T topology

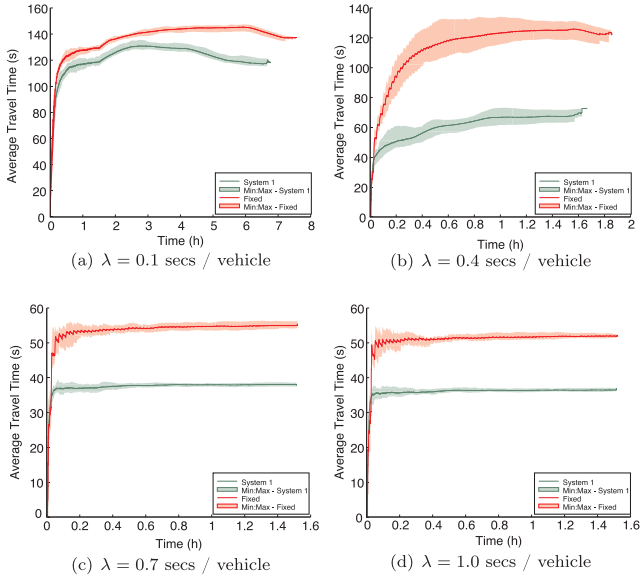
The Simple T topology is illustrated in Fig. 15. It is the simplest case we simulated.

Fig. 16 presents the results for a set of simulations without Smart Car, in 4 different traffic loads, where vehicles were supposed to appear in the simulation, using a Poisson distribution with the following time constants: $\lambda = 0.1$, 0.4, 0.7 and 1.0 s.

Each graphic shows two different cases: a curve in red showing the average of 10 simulations using the fixed controller (without our CM, and using SUMO heuristics), and a curve in green showing the average of 10 simulations using our Junction CM. Both the red and green curves are plotted with a shadow showing the interval from minimum to maximum values obtained within the 10 cases. For this situation, the Junction CM obtained better results than the fixed controller in all the tested traffic patterns, providing an average travel time which is smaller than the one provided by the fixed controller.

We tested different situations with Smart Cars. For each traffic pattern, we randomly selected different cars to promote as Smart Cars and simulated the same traffic again, now considering the chosen car as a Smart Car. The simulations with Smart Cars didn't showed a significant decrease in overall performance due to giving priority to the Smart Cars. In situations of heavy traffic, this priority didn't really improved too much the travel time of these cars, in some cases, as even though the Junction Manager is doing its best in giving priority to the Smart Cars, there are some boundary conditions which cannot be avoided (a minimum and maximum time in each phase). But by

Fig. 17. Twin T model.



(a) $\lambda = 0.1$ secs / vehicle (b) $\lambda = 0.4$ secs / vehicle

(c) $\lambda = 0.7$ secs / vehicle (d) $\lambda = 1.0$ secs / vehicle

Fig. 18. Simulation results for the Twin T scenario.

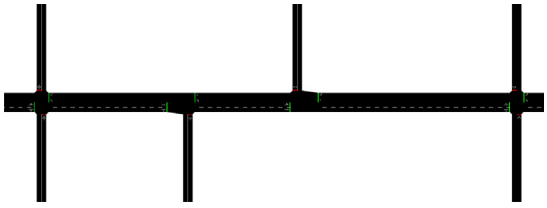
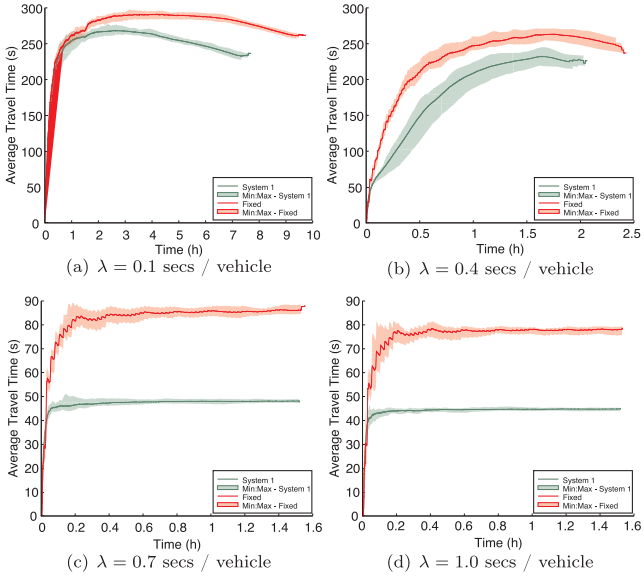


Fig. 19. Corridor model.



(a) $\lambda = 0.1$ secs / vehicle (b) $\lambda = 0.4$ secs / vehicle

(c) $\lambda = 0.7$ secs / vehicle (d) $\lambda = 1.0$ secs / vehicle

Fig. 20. Simulation results for the corridor scenario.

selecting the cars in worst condition and promoting them to Smart Cars really improved their travel time. In lighter traffic conditions, the Smart Cars were able to improve their travel time due to our preference policy in most of the cases.

6.2. The Twin T topology

The Twin T is a topology slightly enhanced while compared to the Simple T. This topology can be viewed in Fig. 17.

Fig. 18 presents the results for a set of simulations without Smart Car, in 4 different traffic loads, similarly to those for the Simple T.

The results for the Twin T are somewhat equivalent of those for the Simple T. For a more heavy traffic situation, the Junction CM was able to provide a substantial decrease in the average travel time. For more light traffic, though, both the Fixed controller and the Junction CM obtained very similar results.

Again, no significant change in performance was detected while running simulations with SmartCars.

6.3. The Corridor topology

The Corridor topology provides a further complex scenario, where now 4 controlled junctions are considered. The Corridor topology can be viewed in Fig. 19.

Fig. 20 presents the results for a set of simulations without Smart Car, in 4 different traffic loads, similarly to those for the Simple T and Twin T. And again, no decrease in performance was detected due to the introduction of Smart Cars.

7. Conclusion

As a proof-of-concept, the results of our simulations show that the Junction CM is able to provide a fair service, providing the collaboration with Smart Cars when this is necessary. The current implementation, though, does not adequately address the interference between neighbor junctions, a topic which requires a further study. Fortunately, our choice of MECA as the cognitive architecture to implement cognitive abilities allows the scalability of our problem solving strategy. It is just a matter of obtaining more inputs (e.g. information from neighbor junctions), followed by further behaviors, either in System 1 or in System 2, in order to enhance the Junction CM abilities.

Even though our simulation cases are very simple, they allow us to validate MECA as a viable cognitive architecture. The next steps in our research might require more complicated topologies and a better study of more sophisticated traffic patterns, which might require some enhancement in our control strategies.

As future work, we also intend to implement learning mechanisms, as e.g. using reinforcement learning techniques (like deep learning), to evaluate different control strategies and allow the system to test different strategies and learn the performance of them.

In the sense of applying MECA to build other kinds of Cognitive Managers, we are currently planning its application to an Industry 4.0 use case: a warehouse automation scenario. Also, its use on Cognitive Networks is being considered.

Acknowledgments:

The authors thank RLAM Innovation Center, Ericsson Telecomunicações S.A. Brazil (Proc. FUNCAMP 4881.2) and CEPID/BRAINN (Proc. FAPESP 2013/07559-3) for supporting this research.

References

Abbott, R. (2006). Open at the top; open at the bottom; and continually (but slowly) evolving. In *System of Systems Engineering, 2006 IEEE/SMC International Conference on* (pp. 6-pp). IEEE.

Anderson, J. R. (2014). *Rules of the mind*. New York and London: Psychology Press.

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

Boillot, F., Midenet, S., & Pierrelee, J. (2006). The real-time urban traffic control system CRONOS: Algorithm and experiments. *Transportation Research Part C: Emerging Technologies, 14*(1), 18–38 https://hal.archives-ouvertes.fr/hal-00505754.

Box, S., & Waterson, B. (2013). An automated signalized junction controller that learns strategies by temporal difference reinforcement learning. *Engineering Applications of Artificial Intelligence, 26*(1), 652–659 http://www.sciencedirect.com/science/article/pii/S0952197612000504.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation, 2*(1), 14–23.

Budakova, D., & Dakovski, L. (2006). Computer model of emotional agents. *Lecture Notes in Computer Science, 4133*, 450.

Cai, C., Wong, C. K., & Heydecker, B. G. (2009). Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies, 17*(5), 456–474 Artificial Intelligence in Transportation Analysis: Approaches, Methods, and Applications. URL < http://www.sciencedirect.com/science/article/pii/S0968090X09000321 > .

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

Canamero, D. (1998). Issues in the design of emotional agents. In *Papers from the 1998 AAAI fall symposium on emotional and intelligent: The tangled knot of cognition* (pp. 49–54).

Choy, M. C., Srinivasan, D., & Cheu, R. (2003). Cooperative, hybrid agent architecture for real-time traffic signal control. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 33*(5), 597–607.

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*. Houghton Mifflin Harcourt.

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.

Gorod, A., Sauser, B., & Boardman, J. (2008). System-of-systems engineering management: A review of modern history and a path forward. *Systems Journal, IEEE, 2*(4), 484–499.

Gudwin, R. R. (2017). *The MECA cognitive architecture*. Tech. Rep. D5, University of Campinas, Campinas-SP, Brazil.

Gudwin, R., Paraense, A., de Paula, S. M., Fróes, E., Gibaut, W., Castro, E., ... Raizer, K. (2018). *Multipurpose enhanced cognitive architecture* < https://github.com/EricssonResearch/meca > .

Gudwin, R., Paraense, A., de Paula, S. M., Fróes, E., Gibaut, W., Castro, E., ... Raizer, K. (2017). The multipurpose enhanced cognitive architecture (MECA). *Biologically Inspired Cognitive Architectures, 22*, 20–34.

Harrison, C., Donnelly, I. A. (2011). A theory of smart cities. In *Proceedings of the 55th annual meeting of the ISSS-2011, Hull, UK* (Vol. 55).

Henry, J. J., Farges, J., & Tuffal, J. (1983). The prodyn real time traffic algorithm. In *IFAC/IFIC/IFORS conference on control in transportation system* (pp. 305–310).

Heung, T. H., Ho, T.-K., & Fung, Y.-F. (2005). Coordinated road-junction traffic control by dynamic programming. *IEEE Transactions on Intelligent Transportation Systems, 6*(3), 341–350.

Heydecker, B., Cai, C., & Wong, C. (2007). Adaptive dynamic control for road traffic signals. In *IEEE international conference on networking, sensing and control, 2007* (pp. 193–198).

Hull, C. (1943). Principles of behavior.

Husch, D., Staff, T., & Albeck, J. (2003). *Synchro 6: Traffic signal software - User guide*. Trafficware, Limited < https://books.google.com.br/books?id=W3EzPQAACAAJ > .

Kelaidonis, D., Somov, A., Foteinos, V., Poulios, G., Stavroulaki, V., Vlacheas, P., ... Giaffreda, R. (2012). Virtualization and cognitive management of real world objects in the internet of things. In *2012 IEEE international conference on green computing and communications (GreenCom)* (pp. 187–194). IEEE.

Kim, C.O., Park, Y., & Baek, J.-G., 2005. In *Proceedings of the international conference on computational science and its applications – ICCSA 2005, Singapore, May 9–12, 2005, Part IV* (pp. 148–160). Berlin, Heidelberg: Springer Berlin Heidelberg, Ch. Optimal Signal Control Using Adaptive Dynamic Programming < https://doi.org/10.1007/11424925_18 > .

Kotseruba, I., Gonzalez, O. J. A., & Tsotsos, J. K. (2016). *A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications*. arXiv preprintarXiv:1610.08602.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal on Advances in Systems and Measurements, 5*(3&4), 128–138.

Laird, J. (2012). *The soar cognitive architecture*. MIT Press.

Lämmer, S., & Helbing, D. (2008). Self-control of traffic lights and vehicle flows in urban road networks.

Luyanda, F., Gettman, D., Head, L., Shelby, S., Bullock, D., & Mirchandani, P. (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. https://doi.org/10.3141/1856-19.

Maslow, A. H. (1943). A theory of human motivation. *Psychological Review, 50*(4), 370.

Meyer, J.-J. (2008). Reasoning about emotional agents. *Artificial Intelligence Preprint Series, 44*.

Mirchandani, P., & Head, L. (2001). A real-time traffic signal control system: Architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies, 9*(6), 415–432 http://www.sciencedirect.com/science/article/pii/S0968090X00000474.

Nakamiti, G. S. (1996). *Distributed artificial intelligence: Architecture, formal specification and application (in portuguese)*, Ph.D. thesis. State University of Campinas (UNICAMP).

Ortony, A., Clore, G. L., & Collins, A. (1990). *The cognitive structure of emotions*. Cambridge University Press.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., & Wang, Y. (2003). Review of road traffic control strategies. In *Proceedings of the IEEE* (pp. 2043–2067).

Paraense, A. L. O., Raizer, K., de Paula, S. M., Rohmer, E., & Gudwin, R. R. (2016). The cognitive systems toolkit and the CST reference cognitive architecture. *Biologically Inspired Cognitive Architectures, 17*, 32–48.

Picard, R. W. (1997). *Affective computing, Vol. 252*. MIT Press Cambridge.

Porche, I., & Lafortune, S. (1999). Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation Systems, 4*(3-4), 209–254 http://dblp.uni-trier.de/db/journals/jits/jits4.html#PorcheL99.

Reilly, W. S. (1996). *Believable social and emotional agents*. Tech. rep., DTIC Document.

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.

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. *International workshop on intelligent virtual agents* (pp. 498). Springer.

Sik, H. Y., Soo, K. J., Kwang, S. J., & Kug, P. C. (1999). Estimation of optimal green time simulation using fuzzy neural network. In *The 1999 IEEE international fuzzy systems conference, FUZZ-IEEE' 99, August* (pp. 761–766). Seoul, South Korea.

Sims, A., & Dobinson, K. (1980). The Sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on Vehicular Technology, 29*(2), 130–137.

Sloman, A. (1987). Motives mechanisms and emotions. *Emotion and Cognition, 1*(3), (Reprinted in M.A. Boden (Ed.), the philosophy of artificial intelligence. Oxford Readings in Philosophy's Series, Oxford University Press 2 (3), 1–24 (1987)).

Sloman, A. (1998). Damasio, descartes, alarms and meta-management. In *1998 IEEE international conference on systems, man, and cybernetics, 1998* (Vol. 3, pp. 2652–2657). IEEE.

Sloman, A., et al. (2001). Beyond shallow models of emotion. *Cognitive Processing, 2*(1), 177–198.

Srinivasan, D., Choy, M. C., & Cheu, R. L. (2006). Neural networks for real-time traffic signal control. *IEEE Transactions on Intelligent Transportation Systems, 7*(3).

Stevanovic, J. (2007). Stochastic Optimization of Traffic Control and Transit Priority Settings in VISSIM. Department of Civil and Environmental Engineering, University of Utah. URLhttps://books.google.com.br/books?id=00uBMQAACAAJ.

Sun, R. (2003). A tutorial on clarion 5.0. Unpublished manuscript.

Sun, R. (2009). Motivational representations within a computational cognitive architecture. *Cognitive Computation, 1*(1), 91–103.

Thomas, R. W., Dasilva, L. A., & Mackenzie, A. B. (2005). Cognitive networks. In *Proceedings of IEEE DySPAN 2005*. Citeseer.

Toates, F. M. (1986). *Motivational systems. No. 4*. CUP Archive.

Viti, F., & van Zuylen, H. J. (2010). A probabilistic model for traffic at actuated control signals. *Transportation Research Part C: Emerging Technologies, 18*(3), 299–310 11th {IFAC} Symposium: The Role of Control < http://www.sciencedirect.com/science/article/pii/S0968090X09000618 > .

Xie, X.-F., Smith, S. F., Lu, L., & Barlow, G. J. (2012). Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies, 24*, 168–189 http://www.sciencedirect.com/science/article/pii/S0968090X12000460.

Zhao, D., Dai, Y., & Zhang, Z. (2012). Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42*(4), 485–494.