

A Multiagent System for Helping Urban Traffic Management

Luis A. García and F. Toledo

Universitat Jaume I
12071 Castellón, Spain
{garcial, toledo}@icc.uji.es

Abstract

This paper describes the MASHGREEN DM prototype following three goals: 1) The identification and implementation of the tasks related to urban traffic control that use deep reasoning mechanisms as main tools for their resolution. The chosen model for explaining urban traffic behaviour is a qualitative model, which includes among its main features their low temporal and spatial computational costs. 2) The definition of a functional architecture with soft real time constraints that integrates the developed tasks. A specialized component named agent, composed by four functional modules executes every task. These modules are the following ones: communication protocols, methods (agent specialization), data space and control. This architecture verifies that the execution performances of every agent are not compromised by the inclusion of new agents in the system, or by the interactions due to the active system agents. 3) The implementation of a prototype in a high performance computational architecture, a *Beowulf* computer system.

1 Introduction

In the last two decades there was a broad interest in the research and development of new methods and tools suited to Intelligent Traffic Systems (ITS). However, the amount of public funds devoted to this action has been continually reduced. The main reason is that a small number of enterprises dominate the market of Urban Traffic Control Systems (UTCS). Current UTCS (including both, software and hardware components) are closed systems in the sense that they belong to the enterprises that sell them. The cities which adopt these systems have their traffic operational enlargement possibilities seriously constrained by high economic cost.

Nevertheless, the great development in the telecommunications and information technologies area, is promoting a change of mentality about the development of new ITS. This change is being used by the European Union to try to open the monopolies to the competence, as much in infrastructure supply as in the provision of services.

In the USA the *Workshop in Support ADVANCED TRAFFIC MANAGEMENT SYSTEMS (ATMS)* held in Florida, May of 1999 (funded by the *US Department of Transportation* and the *Federal Highway Administration*) pointed out the current trends in the development of new ATMS:

- The development of traffic models and the corresponding copyright of the software used to programming them. This development involves the use of quality public software (mainly *Free Software Foundation* software) to programming existing and new traffic models.
- The application of models to deal with lost of significant data to help traffic decision taken. It is important to continue the research on traffic flow and data fusion from different kinds of traffic sensors.
- The development of models and algorithms for dealing with real time characteristics. It must be identified which are the real-time operative constraints related to every task in ATMS.
- The search of models to perform dynamic settings of traffic signals. It is proposed to develop advanced traffic models related to real time traffic surveillance and automatic incident detection.

In order to follow these trends, the MASHGREEN DM prototype (A MultiAgent System for Helping uRban traffic dEcision takEN using Distributed Memory) prototype is developed and described in this paper. The organization of this paper is as follows. Section 2 describes the AI architectures that are being applied to traffic control. An urban traffic deep reasoning mechanism is exposed as a basic tool to deal with the above mentioned trends in ATMS. In section 3, the main features of the MASHGREEN DM prototype are exposed, including the definition of an abstract architecture suited to their agents. In section 4 the implementation and the main results of the execution of this software prototype in a *Beowulf* cluster system is exposed. The paper finishes with the main results achieved.

2 AI Architectures in UTCS

Current UTCS are generally composed of two layers. A first layer uses quantitative models, data and algorithms. A second layer uses intensively the control engineer experience. Given the complexity and the big size of the domain of the urban traffic control problem, the supervision and actions of these

engineers must be continuously provided. Therefore, the two layers UTCS are insufficient for achieving an adequate control [Bell *et al.*, 1991]. This analysis has led to several authors to introduce knowledge based systems (KBS) in the UTCS architecture [Foraste and Scemama, 1986; Bell *et al.*, 1991; Ambrosino *et al.*, 1997].

The prototypes developed up to now have been influenced by two factors: the experience learned in the implementation and evaluation of previous prototypes (considering the technologies used to manage the functionality of the system as well as the knowledge representation used) and the broad availability of faster and cheaper computers (therefore, the responses can be given early and it is possible to add new capabilities to the prototype).

The first KBS applied to traffic control (like the *SAGE* system [Foraste and Scemama, 1986]) have a shallow reasoning mechanism, that is big sets of cause/effect rules implemented in a classical functional architecture of production systems. This kind of knowledge representation is very fragile. The cause/effect rules typically reflect empirical associations derived from experience, rather than a theory of how the device under diagnosis actually works [Jackson, 1990]. Therefore, several authors proposed the integration of a model of urban traffic behaviour in the prototypes, i.e. the integration of a deep reasoning mechanism for urban traffic [Moreno *et al.*, 1993; Ambrosino *et al.*, 1997; Irgens *et al.*, 1997].

The second generation KBS integrate a deep model to explain the behaviour of the system. Besides it, they usually adopt functional collaborative architectures to integrate their components. The Blackboard architecture [Carver and Lesser, 1994] was used in the *Intelligent UTCS* prototype developed under the *DRIVE I & II* European strategic programs [Ambrosino *et al.*, 1997]. Nevertheless, these prototypes were developed and implemented using conventional computers (although the underlying theory on blackboard systems was to use a shared memory multiprocessor computer). This was due because, although there were a broad availability of shared memory parallel computers, the tools needed for programming them were not evolved at the same rate. Due to the great effort to develop better shared memory programming tools, the distributed systems were getting more significance because their lower economic cost, their performances close to the shared memory parallel computers and the wide availability of public domain tools, as MPI, for programming them.

Moreover, the prototypes based on blackboard models have a strong dependency of every component with respect to the rightness and integrity of the information stored in the blackboard, as much as its great generality, what is its strength and weakness, i.e. it is very difficult to forecast the execution sequence of their components. Therefore, it is needed a new functional collaborative architecture to reduce in a significance way the result and data dependency on execution among their main components.

The multiagent architecture fulfills these needs. The *KITS* system [Irgens *et al.*, 1997] implements an architecture called *Supervised Cooperative Agents* built on features of the *IUTCS* prototype. The key point introduced in the *KITS* system is to

split the urban network in different areas. Every area is composed of a set of elements. The set of elements of each area requires a common analysis and operation process, i.e. it is an environment based architecture. Each area is managed by a different agent. The prototype exposed in this paper uses a different approach. Every agent is able to see the whole urban network. A very reliable and fast deep model reasoning is used to understand the evolution of the urban traffic. Communication among agents are performed to share knowledge and control actions. The next subsections expose the deep knowledge reasoning model used and its applications on computing several urban traffic control operations.

2.1 The Deep Reasoning Model

The main idea behind its definition is to model the spatio-temporal fluctuations in traffic levels by defining a discrete qualitative space. Every point with the same density of vehicles define a qualitative region. The spatio-temporal limit that divide two adjacent regions is represented by a straight line. The vertexes of each qualitative region are events which represent changes in the dynamics of the system: the appearance or disappearance of qualitative regions, the change from increase to decrease in the size of a qualitative region and so on. In this way, it is possible to achieve entities with cognitive meaning due to the selection of events as primitive ones that represent the evolution of the system. The urban traffic network is modeled as a set of objects (streets, intersections, traffic lights, inputs, outputs, etc.) that have associated parameters. The evolution of these parameters is described by uni- or bi-directional constraints. This qualitative simulator is broadly explained in [Moreno *et al.*, 1993].

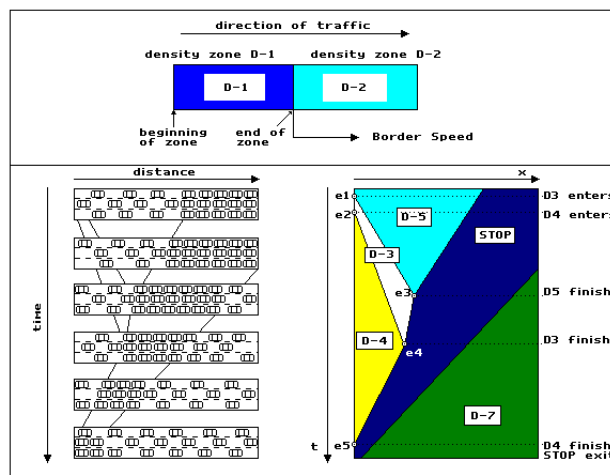


Figure 1: Temporal qualitative representation of a queue generation process due to the change from red to green in its traffic light

The evolution of the urban traffic is achieved by using the qualitative simulator and data preprocessed from sensors. They are both necessary because the current traffic status cannot be obtained just from sensors mainly for two reasons.

First, the sensors usually integrate in periods, therefore current information from sensors is not always available. Secondly, there is a limited number of sensors due to economical and operational restrictions therefore, sensors give only a small part of the information necessary to reconstruct the current traffic status. When sensor data arrives, the actual traffic status is calculated by reconstructing all the events produced during the last temporal interval without real sensor data. The time overhead of the adjustment process between the simulated and the reconstructed traffic status is small due to: (1) the time complexity of the qualitative simulator is small (quadratic on the number of intersections, segments and the length of the simulation interval in the urban network), i.e., the simulator runs much faster than real time; (2) the length of the temporal interval without real sensor data is also small (usually five minutes).

2.2 Application to UTCS Capabilities

The deep reasoning model previously exposed can be applied to several UTCS capabilities: *Preprocessing* and *Monitoring*, *Detection and Diagnosis of Problems*, *Short Term Prediction* and the *Search of Suitable Control Operations*, among others. The function for *Preprocessing* and *Monitoring* translates data from sensors into the qualitative representation described. In this way, a qualitative representation of the system independent of the sensors is obtained, which allow us to use data from several types of sensors. The current state of the system is generated by using qualitative simulation from the previous qualitative state, instead of using direct analysis of sensor data. The qualitative simulation from the previous state will give a whole image of the state of the system assuming that the previous state is correct, the prediction process is correct, and no unpredictable events, i.e. urban traffic incidents, have happened during the simulation. Sensor data is used to verify these facts when they are available.

The function for *Detection and Diagnosis of Problems* analyzes the past and present qualitative states of the system to detect problems. This task should not consider transient problems. On the other hand, a problem belonging to a zone must not be detected as the sum of the problems over the spatial components of the zone. A hierarchical classification of urban traffic problems based on the *General Representation Formalism* [Equator, 1994], an extension of the event calculus of Kowalsky with temporal granularity and continuous change, is used to accomplish this task. The matching between the traffic status reconstructed with real sensor data and the traffic status provided by the qualitative simulation is used to detect incident situations [García, 2000].

The function for *Prediction* predicts which would be the qualitative state of the system at short and medium terms. This task is done by supposing that the previous states are correct, the prediction process is correct, and have not occurred urban traffic incidents during the prediction. The low computational cost of the simulation qualitative process allows this task to be done. The aim is to detect the future problems that will arise if the current control strategy is kept. Once this overall state is computed then the system must look for problems. If there are enough detected problems then the system should search for solutions for minimizing them. This

task does not need to perform a diagnosis of urban incidents process, because future detected problems are caused by bad regulation traffic settings. This behaviour is very important for an UTCS because it should have the capacity to anticipate problems. For example, one of the most common actions is to prevent a primary congestion, which has arisen as a result of the excess of traffic volume, turning into a secondary blockage. This blockage can be due to a growing queue at an intersection located afterward, with the consequent extension of the blockage over other urban network zones.

The last function exposed is the *Search of Suitable Control Operations*. If the detected problems are due to bad regulation traffic settings, then the system tries to compute other traffic settings that minimize these problems. The solution mainly consists of correcting the global effects of these bad control actions, which usually is more complicated than simply to remove the bad control actions. This is because the implementation of local solutions will produce undesirable effects, for example, that these problems will disappear from a zone but will appear in other zones of the urban network. The length reached in a traffic queue is completely determined by two conditions: the traffic status when the traffic light is green, and 2) the time allowed for red light. Therefore, it is necessary to know which are the possible red times for every intersection in the system. There are eight possible values for the red time of every input link to an intersection. The *split* of an intersection is defined as the set formed by the active red time value of every one of its input links. The set formed by the splits of all the intersections is known as a traffic plan. It is impossible to evaluate exhaustively every traffic plan, because its temporal complexity is $\Theta(8^n)$ traffic plans, where n is the number of intersections. The size of this search space is drastically reduced by using two strategies: (1) to sort the set of intersection splits in relation with the red time allowed to its main direction; and (2) to apply a branch and bound method limited to a maximum time for execution. The method consists of the qualitative simulation of the urban traffic by evaluating the profits that it can be obtained by changing the splits of the intersections in relation to their qualitative evolution [García, 2001a]. This method, with lightly modifications, can also be applied to compute other kind of control actions: values for coordination of intersections, and values for the length of the working cycles in groups of intersections [García, 2001b].

3 The MASHGREEN DM Prototype

The MASHGREEN DM prototype is a collaborative software architecture developed for urban traffic management tasks. Its main behaviour features are the following:

- It works with soft real-time deadlines. The execution of its tasks are interrupted at periodic time (except exceptional situations).
- It is software reusable. It can integrates tasks with different computational costs.
- It contains a heterogeneous and distributed computer implementation.

The prototype integrates software agents to perform several tasks: *monitorization* and data preprocessing, *short-medium*

term prediction of urban traffic, bad regulation urban traffic problems identification, urban traffic incident detection, continuous search of best urban traffic plans and cycle lengths, the coordination of the execution of the agents, and a friendly interface to the human operator. The actions allowed for the prototype to urban control are: the automatic and manual execution of the best control actions proposed by the prototype and the punctual modification of the working features of whatever controlled element in the urban network, proposed by the human operator in any moment. Therefore, the main objective of the prototype is to coordinate skills, knowledge, plans and experience among several agents to monitorize, propose and execute control actions that could improve the urban traffic status.

Four design decisions were taken in the development of the prototype: 1) organization and distribution of the data among agents, 2) integration of tasks with different computational costs, 3) implementation of a unique and common temporal reference for every agent of the prototype, and 4) the definition of an agent software architecture to be shared by every agent of the prototype.

3.1 Data Organization and Distribution

There are two kinds of data: static (does not change on execution) and dynamic (change on time). There were two options to organize the data of the prototype: 1) store all the data in the same location (e.g. using a blackboard organization) or, 2) assign different data subsets to different agents, where every agent is responsible of the integrity and correctness of the data that stores. The first option has the advantage that every agent can access to static and dynamic domain data at any time, but it also needs a complex maintenance of the data coherence. The second option overcomes this problem, but it introduces a latency in data request between agents. However, this second option also overcomes the strong centralized component that is the blackboard.

A deep study of the data dependencies among the agents shows that there are a few data which can be accessed simultaneously. Moreover, in these cases, the data access is for reading operations. Therefore, the option chosen is to replicate static data in every agent, while dynamic data are assigned to the agents that make the best use of them.

3.2 Execution Mode

Current trends in AI distributed programming pointed out the high interest in the application of distributed control methods. In these methods every agent knows when and how to perform its work. Nevertheless, the choice of a lightly centralized control has several advantages in this application domain: 1) the task of the agents must be synchronized every time the sensor data arrives to the system, therefore a coordination agent can organize these synchronizations, 2) the inner clocks of the agents must be synchronized to avoid lack of coherence between the real system and the controlled system implemented in the prototype, so a coordination agent can provide the temporal referee for every agent of the prototype, and 3) a coordination agent can be used to monitor the prototype on execution and for storing the configuration parameters of the prototype, so this feature allows us to integrate

fault tolerant mechanism and to perform dynamic adding and deleting of agents. These behaviour features can be also implemented with a distributed control, but it requires a deeper knowledge in the evolution of the execution of the prototype. Therefore, the option chosen is to develop a first prototype to be improved in those undesirable behaviours as result of its later evaluation.

3.3 Common Temporal Reference

The MASHGREEN DM prototype implements a synchronization mechanism to maintain a common temporal clock value for every component of the prototype. This mechanism is based on the following characteristics: 1) the basic temporal unit is the second, 2) the interconnection network transmission time for a single message of whatever size is less than one second, 3) there is an agent, the coordination agent, that contains the right value for all other agents of the prototype, and 4) there is possible to apply communication primitives to perform barrier synchronization and broadcasting.

The applied method is the following:

1. The coordination agent sends at time t_i a broadcast message to request for time synchronization.
2. Every agent receives the request and sends its agreement to the coordination agent. Then, it waits until the reception of the time value message from the coordination agent.
3. The coordination agent send at time t_k the time value message when it has received the agreement message from every agent of the prototype.

Every agent that is in wait state must guarantee response to whatever message received from other agents. The returned message must notifies that the original receiver agent is executing a synchronization barrier. In this way, it is prevented deadlock situations between agents of the prototype. The method exposed has some drawbacks. The final value stored in every agent is t_k or $t_k + 1$ due to the latency in the reception of the time value message sent by the coordination agent. But this is not important due to the application domain features. Another drawback is that there is an unbalanced wasted time on the time waited for the agents until the last of them had arrived to the barrier. Therefore, to avoid these undesirable behaviour, the synchronization method is executed only at the beginning of the execution of the system and every time there were dynamic agents be added or deleted in execution time.

3.4 Agent Architecture

The software architecture of every agent is composed of four modules to deal with communications, methods, data workspace and control. The communication module implements the communication primitives (broadcasting, barrier synchronization, isolated synchronous/asynchronous sending and receiving of messages). The methods module implements two kind of methods: primary methods and secondary methods. Primary methods represent the specialization skills of the agent. Secondary methods implements administrative

work for the agent, which can be inner tasks (coherence maintenance of its data workspace) and outer tasks (knowledge coherence with the beliefs of other agents, synchronization tasks among agents and analysis of alerts). Every data belonging to the knowledge of the agent is stored in the data workspace module that is organized as a blackboard. Finally, the control module takes the decision on the action to be done at any moment depending on actual time, the messages received and the specialization skills of the agent.

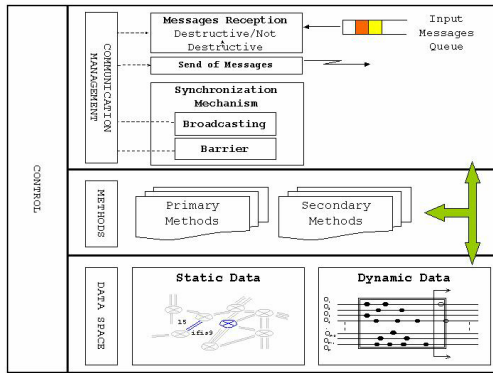


Figure 2: Module decomposition of an agent

The interaction among these modules can be described by using the following formal notation: an agent of the MASHGREEN DM prototype is a tuple of eight components: $A = (Methods, s_0, T, AgId, MsgType, K, Ags, MsgSel)$ where $Methods = PriMethods \cup SecMethods$, $s_0 \in SecMethods$ represents the initialization task of the agent, $T : Z \rightarrow Z$ is a function that returns the common virtual time of the system, $T(t_i) = t_{v_i}$, i.e. the local t_i time in an agent is the t_{v_i} virtual time common to every agent, $AgId$ is the set of possible agents identification, $MsgType$ is the finite set of kind of messages that an agent can receive or send, K is the finite set of communication primitives the agent can use, $Ags : T \rightarrow 2^{AgId}$ is the function that returns the active agents at the common temporal time t_{v_i} , and $MsgSel$ is the function that selects which message to analyze at any moment given the active set of received messages. The $MsgSel$ function defines a total order among the messages received that must fulfill two basic properties: *liveness*, i.e. every received message must be analyzed or effectively deleted, and *deadlock free*, i.e. it must not be possible to cause a deadlock situation between two or more agents. The definition of the components $Ags, T, AgId$ and K must be common to every agent belonging to the same multiagent system.

The formal definition of the *best urban traffic plan* agent is given as an example of the application of the above formal definition of the agents. In this agent the primary method is *calculate_traffic_plan*. The secondary methods are: (1) *startup_agent* = s_0 which prepares the agent to be included in the set of active current agents, (2) *cease_agent* which prepare the agent to be deleted of the set of current active agents, (3) *external_action* to deal with external control actions performed by the human operator, (4) *chg_mode* which set up the way that the results are communicated to other agents, and (5)

monitor which measures and communicates to the coordination agent the main behaviour features in the execution of this agent.

Messages from other agents can be received when the agent is idle or when the agent is executing one of its methods. The *MsgSel* function is defined in both situations. There is no mutual data dependency between this agent and whatever other agent, so this function is deadlock free. The liveness property is also fulfilled because more important messages related with the evolution and control of the traffic status are analyzed earlier, and as result of this analysis, other messages are deleted from the active set of messages received.

4 Implementation Issues

The MASHGREEN DM prototype has been implemented in a *Beowulf* computer network. This kind of computers are composed by low cost hardware components, commonly personal computers. They are programming using distribution free software, commonly *Linux* and public communication libraries. The communication workload is determined only by the application on execution. A unique node of the cluster has communication with the output network from the cluster, so the nodes of the cluster only execute tasks belonging to the cluster. The *Beowulf* cluster used is composed of one server, a biprocessor personal computer at 400 MHz, and 32 working nodes, which are personal computers working at 300 MHz. There are two active interconnection networks in the system, a *Fast Ethernet* one running at 100 Mbits per second, and a *High Performance* one running at 1.2 Gbits per second. The operating system running in the *Beowulf* cluster is *Linux* and the programming language used for programming the agents is *constraint logic programming* extended with *linda* distributed programming model and *tcl/tk* script language interfaces.

The *Interface* agent runs on the server of the cluster because this agent must interact with the human operator of the prototype. The rest of agents implemented run on different working nodes of the cluster. Another working node of the cluster is used to maintain the communications in the *linda* model. The *Fast Ethernet* network is used for performing the communications between the agents of the prototype. This network has enough capacity because the urban traffic domain features, the lower simultaneously number of messages sent, and time needed for its transmission is not a critical feature for the agents. Moreover, the *Fast Ethernet* is a standard network model cheaper than the *High Performance* network.

Every agent of the MASHGREEN DM prototype has been evaluated using isolated and collaborative off-line executions in laboratory tests. Their executions show a stable behaviour working with different input data sets with and without the interaction with the human operator (in [García, 2001a] it is analyzed the *best urban traffic plan* agent and in [García, 2001b] it is analyzed the *best cycle length* agent). The *coordination* agent stores the monitoring messages sent by the rest of agents. These messages are stored in disk for later analysis. Figure 3 shows an example of how this analysis is done.

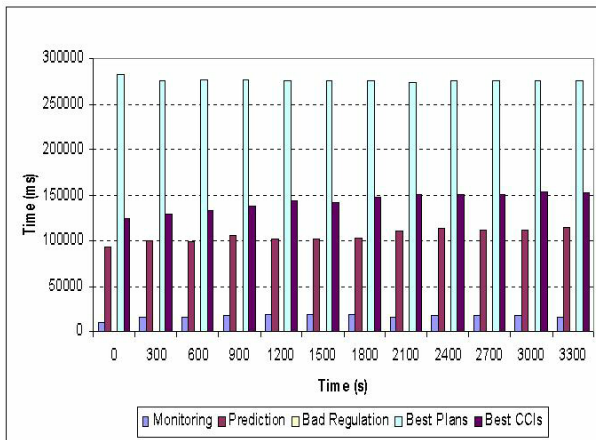


Figure 3: Relationship between the time used to the execution of the primary method of the most relevant agents (vertical axis) with regard to the time available when data from sensors are not available, every 300 seconds (horizontal axis). The graphic shows that the behaviour of every agent is stable

5 Conclusions

Multiagent systems are a relatively new kind of AI architectures applied to UTC that has provided a small number of prototypes up to now. The *KITS* prototype is the most referenced. In *KITS* the urban network is divided in zones. Every zone is associated with an agent that deals with every feature of its area. This is an important difference with the *MASHGREEN DM* prototype because, in the *MASHGREEN DM* prototype every agent has a domain that includes the complete urban network. Therefore, the results provided by its agents have a global view of the urban network. These agents operate by using a deep model of urban traffic which has a very low temporal computational cost. Besides that, the *MASHGREEN DM* prototype provides methods to integrate tasks with different computational cost. Therefore, it is possible to integrate software previously developed. Every agent of the prototype has the same architecture. This architecture is divided in four modules. The first module deals with the definition and implementation of the communication mechanism common to every agent. The second module stores the data needed to obtain its operational objectives. The data distribution among the agents allows us to increment the reliability of the system in the presence of crashed agents. Moreover, this data distribution does not significantly decrease the performances of the system, but it needs the following two conditions to be hold: (1) the agents know where every information is located; and (2) the agents know how the information is represented, i.e. they share the same data structures. The third module stores the methods suitable to be executed. These methods are primary, representing the main operational objectives of the agent, and secondary, representing administrative work for the agent. The last module deals with the control of the agent, i.e. what to do in every moment. The definition of these modules is simple to allow us the easy integration and cooperation between the agents of the prototype. The com-

plex mechanism of their acts are settled by an inner way, depending on how these four modules are implemented and their interactions.

The *MASHGREEN DM* prototype has been implemented in a multiprocessor distributed computer that has three main features: (1) it has a low economic cost; (2) it has a good level of computational performances; and (3) there are stable public domain tools for programming them. The behaviour of this prototype in execution is stable, i.e. the prototype works well in laboratory tests.

Acknowledgments

This work is partly supported by the Spanish CICYT project TAP1999-0590-c02-02 and Bancaixa project P1A98-11.

References

- [Ambrosino *et al.*, 1997] G. Ambrosino, M. Bielli, Boero M., Mastreta M. Application of Artificial Intelligence and Expert Systems to Traffic Control. *Advanced Vehicles and Infrastructure Systems*, John Wiley & Sons, 1997.
- [Bell *et al.*, 1991] M.C. Bell, G. Scemama, L.J. Ibbetson. CLAIRE: An Expert System for Congestion Management. *Advanced Telematics in Road Transport*, Proceedings of the DRIVE conference:596-614, Elsevier, 1991.
- [Carver and Lesser, 1994] N. Carver, V. Lesser. The Evolution of the Blackboard Control Architectures. *Expert Systems with Applications, Special Issue of The Blackboard Paradigm and It's Applications*, 7(1), 1994.
- [Equator, 1994] EQUATOR Final. *ESPRIT p. 2049*, 1994.
- [Foraste and Scemama, 1986] B. Foraste, G. Scemama. Surveillance and congested traffic control in Paris by expert system. *Second IEE Conference on Road Traffic Control*, London, 1986.
- [García, 2000] L. A. García, F. Toledo. Urban Traffic Incident Detection using Qualitative Data and Temporal Intervals. In *Proc. of the IC-AI*, Las Vegas, USA, June 2000.
- [García, 2001a] L. A. García, F. Toledo. An Agent for Calculating the Best Urban Traffic Plan Constrained by Soft Temporal Deadlines. In *Proc. of the SOCO/ISFI-2001*, Paisley, Scotland, June 2001.
- [García, 2001b] L. A. García, F. Toledo. An Agent for Providing the Optimum Cycle Length Value in Urban Traffic Areas Constrained by Soft Temporal Deadlines. In *Proc. of the IEA/AIE-2001*, Budapest, Hungary, June 2001.
- [Irgens *et al.*, 1997] M. Irgens, C. Krogh, H. Traettebers. Model Based Collaboration Architectures for Traffic Control. *Advanced Vehicles and Infrastructure Systems*, John Wiley & Sons, 1997.
- [Jackson, 1990] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley Publishing company, 1990.
- [Moreno *et al.*, 1993] S. Moreno, F. Toledo, F. Rosisch, G. Martin. Qualitative Simulation for Temporal Reasoning in Urban Traffic Control. *Qualitative Reasoning and Decision Technologies*, N. Piera Carrete and M.G. Singh Editors, CIMNE, Barcelona, 1993.