International Journal of Automotive Technology, Vol. 3, No. 3, pp. 89–94 (2002)

1229–9138/2002/003–01

ADAPTIVE, REAL-TIME TRAFFIC CONTROL MANAGEMENT

G. NAKAMITI[1)*] and R. FREITAS[2)]

[1)]Rua Alaro Faria de Barros, 1371, casa 22, 13098-393-Campinas-SP-Brazil
[2)]Paulista State University, Brazil

(Received 27 February 2001; Revised 6 March 2002)

ABSTRACT–This paper presents an architecture for distributed control systems and its underlying methodological framework. Ideas and concepts of distributed systems, artificial intelligence, and soft computing are merged into a unique architecture to provide cooperation, flexibility, and adaptability required by knowledge processing in intelligent control systems. The distinguished features of the architecture include a local problem solving capability to handle the specific requirements of each part of the system, an evolutionary case-based mechanism to improve performance and optimize controls, the use of linguistic variables as means for information aggregation, and fuzzy set theory to provide local control. A distributed traffic control system application is discussed to provide the details of the architecture, and to emphasize its usefulness. The performance of the distributed control system is compared with conventional control approaches under a variety of traffic situations.

KEY WORDS : Urban traffic control, Intelligent control, Computational intelligence

1. INTRODUCTION

Constructing traffic control systems capable of flexible, autonomous, and intelligent behavior is an inherently interdisciplinary and challenging task. Intelligent traffic management systems are designed to perform well under significant uncertainties in the system and environment for extended periods of time, and they must be able to compensate for significant system changes without too much external intervention. They evolve from conventional control and decision systems by adding concepts and techniques from artificial intelligence, real-time computing, and computational intelligence, the main fields that join system and control theory to construct intelligent traffic management systems.

The use of knowledge-based processing is of utmost importance when an application encounter a sizeable number of the members of a large set of related situations, each requiring the selection of one response from a large set of possible responses. In such application, it may be more desirable to automate the process of constructing the response than to enter all the possible responses into the computer in advance.

An intelligent traffic management system requires real-time computing. The issue of having software run in real time when computation must include knowledge-based reasoning means that performance and competence must be combined to deal with applications. On one hand, the majority of control systems are real time in the sense that the programs must run quickly enough. Knowledge-based processing, on the other hand, is directed specifically toward applications, for which no conventional algorithm is available, so that computation times are usually highly variable and unpredictable. Control system's physical and functional structures becomes important design decisions to manage the computational complexity and the implied real-time trade-offs. Distributed control architectures with support to efficient communication and cooperation among agents become essential to provide adaptability, naturalness, specialization, reliability, autonomy, and to handle resource limitations. Real-time, knowledge based control systems must agree with these requirements by trading-off between processing power, response time, data space, inattention, and degradation.

This paper presents an intelligent distributed traffic management system whose basic components are autonomous agents that cooperate *via* communication to achieve common goals. From the methodological point of view, the architecture supports evolutionary case-based reasoning for learning and adaptation, and processing of imprecise information via fuzzy set theory in real-time. A distributed traffic network with sixteen traffic lights in a six intersections network, implemented based on real data from downtown Campinas (Brazil), is included to illustrate the usefulness of the architecture

*Corresponding author. e-mail: nakamiti@bestway.com.br

and its underlying intelligence. Simulation results and a comparison with conventional traffic control schemes are also discussed, together with a set of final remarks to conclude the paper.

2. MANAGING THE URBAN TRAFFIC NETWORK

2.1. The Architecture

The Intelligent Distributed Control System Architecture consists of a network of cooperating, autonomous nodes which are able to take their own decisions, and control local devices. To achieve this, they must know the system's structure and state, besides global goals and constraints. Nodes have to communicate with one another to share information.

Each node is able to reason about the network structure because it knows its detailed neighborhood and overall network structure. Moreover, it is capable to reason about the global problem, and may cooperate to reach a solution once the application context is known. The node's knowledge, represented in the upper side of Figure 1, provides the necessary information to cooperate with its similes (for a more detailed description of the architecture, see (Nakamiti *et al.*, 2000)).

The Local Problem Solver (LPS) is the basic processing unit of the node. It is responsible for handling ordinary situations, *i.e.*, well-understood situations to which heuristics or rules could be established. The LPS basic processing scheme is as follows:

- Receive_from_network_medium(information);
- Receive_from_local_devices(information);
- Update_system_view;
- Inform_neighbor_nodes;
- if not_able (Take_decision (view,rules_or_heuristics))
  Send_CBM(situation);
  Receive_from_CBM(decision);
- Perform(decision);
- Send_neighbors(decision);
- Send_CBM(decision_performance)

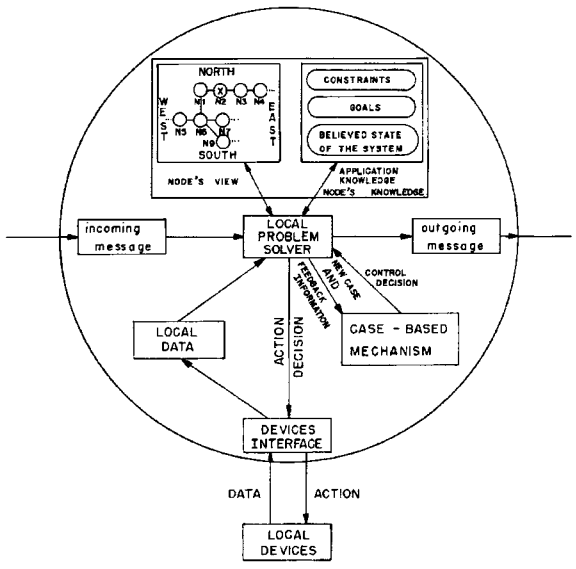


Figure 1. Structure of a node.

An evolutive fuzzy Case-Based Mechanism (Nakamiti and Gomide, 1994) is available to help the LPS in complex and dynamic environments, where all possible situations can not be modeled by the system. Its basic processing scheme is the following:

- Receive_from_LPS(situation);
- Look_for_similar_past_cases(situation);
- Select_successful_cases;
- Combine_actions;
- Send_LPS(actions);
- Receive_from_LPS(performance);
- Decide_if_store(situation,actions,performance)

2.2. Distributed Traffic Control

The fuzzy distributed traffic-light control system consists of a processor situated at each intersection, communicating with its neighbor processors, and deciding the settings of its local traffic-lights. There are sensors measuring the traffic flow entering each intersection; these sensors send data to local processors. The aim is to optimize traffic flow, reducing the average delay of vehicles and average queue lengths, and thus improving the traffic quality. The strategy used to achieve performance requirements consists in changing the green time (and consequently the cycle length[1]) according to the vehicles arrivals and queue lengths.

Most of the research in fuzzy traffic control is restricted to a single intersection (Papis and Mamdani, 1977). describes a controller for a two one-way streets intersection, which decides the green time extension for each way (Nakatsuyama *et al.*, 1984). Presents a control system for two consecutive intersections of an arterial one-way road. Many other papers, such as (Favilla *et al.*, 1993; Hoyer and Jumar, 1994; Kelsey *et al.*, 1993;

[1]The *cycle length* is the time period required for one complete sequence of signal indications in a traffic light. The cycle length is usually divided into a number of phases, each phase being a part of the time cycle allocated to one or more traffic or pedestrian movements. The *green phase* is a particular state that provides a green light (right of way) for a particular direction. The green time represents the time duration of the green phase.

Skowronski and Shaw, 1993), present fuzzy systems for a multi-way single intersection. The control problem for a network of intersections still is an important issue in the field of traffic engineering.

To study this problem, we have modeled part of Campinas-SP (Brazil) downtown area, measured the average arrivals and queue lengths for several intersections, and developed a traffic-light control system for the network based on the distributed architecture.

2.2.1. Local problem solver

The Local Problem Solver has the purpose of controlling the traffic flows at its corresponding intersection. Basically, it compares the incoming traffic and queue lengths of the related intersections, besides its own. Based on this information, it decides to extend or not the current green phase, and informs its neighbors about its decision. This basic approach is used whenever the traffic flows are similar to the traffic flows used to settle the traffic lights, according to (Wohl and Martin, 1967) In this case, the Local Problem Solver makes minor adjustments to pre-defined settings. It uses membership functions for the vehicles' arrivals, queues, and green time extensions.

When the traffic flows vary from the expected ones, or when there is an unexpected situation (such as an accident, a game, or a parade), the strategy above may not be reasonable. In these cases, a Case-Based Mechanism helps the Local Problem Solver to find an appropriate solution.

2.2.2. The case-based mechanism

The CBM's main issues are: to identify and to retrieve similar cases, to combine the selected cases, generating a decision, and to manage the case-base.

To identify and to retrieve the most similar cases, the CBM searches a tree-structured case-base. It handles the experiences' structures, which are divided into an attributes' part, an actions' part, a results' part, and a complementary information' part. Each of these parts is composed by attributes' values, which specify, for example, a cars' queue size (for more details, see (Nakamiti and Gomide, 1994)).

In order to perform the match, let us consider the following attributes parts for the current situation and a stored experience:

Current situation's (CS) attributes:

4	X	1	5	2	?	3	2	X	3	5	3

Stored experience's (SE) attributes:

5	1	1	3	?	3	X	0	X	5	1	3

Very few positions match perfectly, although these experiences may be very similar. So, we have to compare each attribute and establish a degree of proximity (Nakamiti *et al.*, 2000) between the current situation and the stored experiences.

Obviously, instead of testing all the experience-base for proximity, we firstly select a specific sub-tree or a small group of sub-trees, which stores the most similar cases. This is done by applying this same match above to the group of attributes that classify the experience as a "traffic-jam" or "(almost) empty streets", for example.

The test for proximity is performed only to the experiences belonging to the selected sub-tree(s), and eventually to correlated experiences accessed by pointer from them. For each experience, we have the calculated degree of proximity and the stored degree of failure/success. These values permit us to select, for example, the two most similar or most successful experiences.

After retrieving the most similar experiences from their attributes part, the CBM combines, for example, the actions part of the two most similar or most successful experiences through genetic algorithms[2]. There are three possible strategies:

– Take the actions by generic features. For illustration, take one's eyes (size, shape and color), the other's nose, and so on. This is a good strategy for inter-related or fine-grained actions.

Actions part of experience 1:

2	1	4	X	5	?	3	1	X	1	5	6

Actions part of experience 2:

6	X	X	3	1	4	X	2	1	6	2	2

Resulting actions part:

2	1	4	3	1	4	X	2	1	1	5	6

– Take the actions individually, "gene by gene". When both actions are the same, we just take their value. With different actions, we choose one of them, avoiding 'X's and '?'s.

Actions part of experience 1:

2	1	4	X	5	?	3	1	X	1	5	6

Actions part of experience 2:

6	X	X	3	1	4	X	2	1	6	2	2

Resulting actions part:

6	1	4	3	5	4	3	1	1	1	2	2

[2]Experiences may also be taken at random with very low probability, to provide more variability and avoid local maxima.

– Take the actions individually, choosing whatever value between the two of the selected experiences.

Actions part of experience 1:

2	1	4	X	5	?	3	1	X	1	5	6

Actions part of experience 2:

6	X	X	3	1	4	X	2	1	6	2	2

Resulting actions part:

4	1	4	3	5	4	3	1	1	4	2	3

The degree of failure/success may be used as a complementary but decisive information. If it has a definitive importance, the most successful experience as a whole will be taken. If it has no importance, any combination may be achieved.

These mechanisms permit us to work from the best and most similar experiences we have stored, combining them prior of a possible adaptation. Nevertheless, mechanisms like these, although improving old experiences, avoid significantly different actions, because they only adapt previous experiences, and may lead to local maxima. Some values or combinations of values may be missing as the combinations go on. To surpass this limitation, we use a probability as low as 0.04 to change at random the values of the actions or generic actions, depending on the strategy employed.

After generating the decision, the CBM sends it to the LPS, and awaits the results.

The case's results' part is filled by watching the effectiveness of the actions taken. For example, if a medicine taken to lower a patient's fever from 106°F to 98°F, and it has lowered the fever to 105°F, we may consider this a very bad result. A good result would had be achieved, for example, if the fever had lowered to 100°F. After achieving all the results, we are able to calculate the degree of failure/success from them (Nakamiti *et al.*, 2000), mapping the values of the linguistic variables into numbers, as for example: (very bad = 1, bad = 2, regular = 3, good = 4, very good = 5, excellent = 6).

Standard experiences (*i.e.* experiences with standard degrees of failure/success) are stored in the experience-base only if there are few experiences of that kind (in their subtrees) or for a little period of time. Top experiences are always stored and remain in the experience-base for a long time.

The CBM main tasks are: to identify and to retrieve similar cases, to combine the selected cases, to generate decision, and to manage the case-base. Information on past cases is codified into slots of attributes, actions and results, as in a DNA-chain, and stored in the case-base. This information is codified through linguistic variables. The retrieval is performed from the new situation's attributes, which are received from the Local Problem Solver. Similar situations are retrieved and combined through genetic algorithms (Goldberg, 1989). As a result, a new decision for the new situation is generated and applied. (For more details, see (Nakamiti and Gomide, 1994)).

After the decision application by the Local Problem Solver, the Case-Based Mechanism observes its appropriateness, verifying the resulting delays and queues. The new case and its result are included into the case base, allowing better decisions over time.

3. RESULTS

For comparison purposes, a simultaneous, a progressive, and the Intelligent Distributed Control System (IDCS) were implemented. In a simultaneous system, all signals along a street show the same color at the same time. In a progressive system, successive intersections have a common cycle length, but the timing of one signal related to the next is arranged to permit continuous movement of vehicles through the system. We simulated a 6 intersections network composed by 18 traffic-lights, as illustrated by Figure 2.

The data used correspond to real observations of arrival rates in downtown Campinas, on Thursdays,

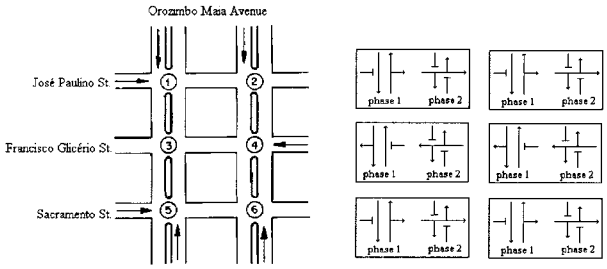
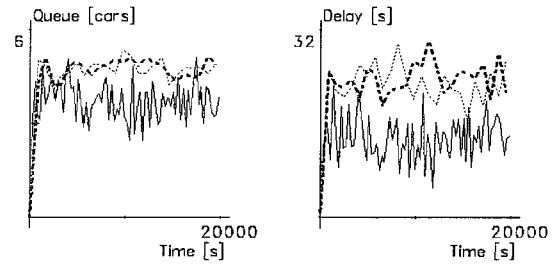


Figure 2. Intersections layout.



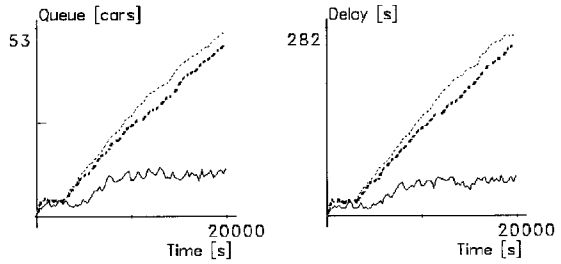
Approach	Av. Queue	Av. Delay	Max. Delay
IDCS	3.4	13.2	24.4
Simultaneous	4.5	23.3	31.8
Progressive	4.5	23.2	30.0

Figure 3. Comparative results – (1).

during the rush time. The cycle length and green time for the simultaneous and progressive systems also correspond to real observations. They were computed according to (Wohl and Martin, 1967) and adjusted on-line by traffic engineers to improve the system performance. Results for other traffic situations may also be obtained in (Nakamit *et al.*, 2000).

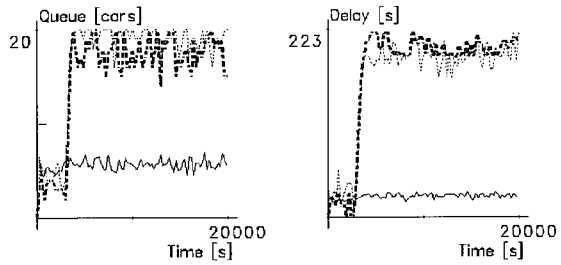
In the first 20,000 seconds simulation, illustrated by Figure 3, we kept the same arrival rates as observed on the intersections. It shows the behavior for the three approaches under these conditions. The Intelligent Distributed Control System (IDCS) presented better performance than simultaneous and progressive systems, demonstrating that its agents can achieve coordination. In fact, its performance was about 24% better regarding to queue lengths (3.4 against 4.5 vehicles), and about 43% better regarding to average delay (13.2 against 23.2 seconds).

In the second simulation, illustrated by Figure 4, we increased the vehicles arrival rates by 30% after 3,000 seconds. The Intelligent Distributed Control System was capable to adapt to the environment changes, and presented average queue lengths of 8.9 vehicles and 38.4



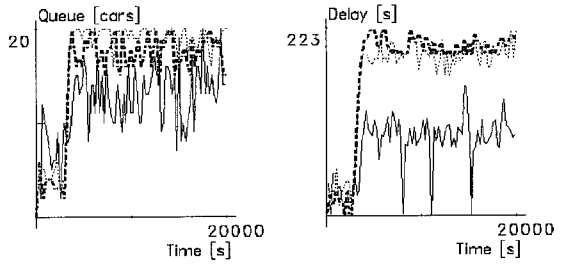
Approach	Av. Queue	Av. Delay	Max. Delay
IDCS	8.9	38.4	64.6
Simultaneous	27.1	143.1	281.7
Progressive	24.3	127.2	259.0

Figure 4. Comparative results – (2).



Approach	Av. Queue	Av. Delay	Max. Delay
IDCS	5.7	24.9	31.4
Simultaneous	16.4	166.3	220.0
Progressive	15.1	172.4	223.0

Figure 5. Comparative results – (3).



Approach	Av. Queue	Av. Delay	Max. Delay
IDCS	12.1	84.2	153.0
Simultaneous	16.4	166.3	220.0
Progressive	15.1	172.4	223.0

Figure 6. Comparative results – (4).

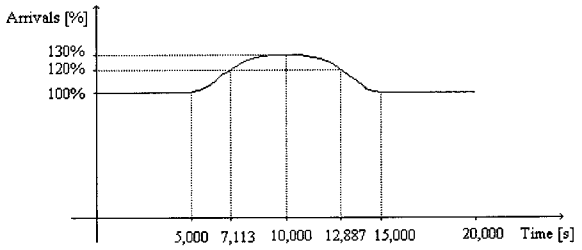
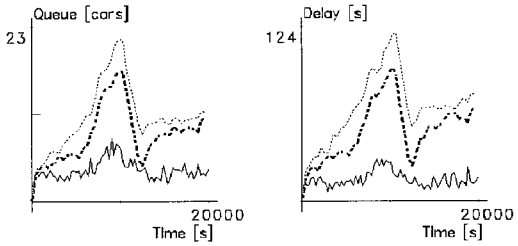


Figure 7. Changing traffic income.

seconds of average delay. The conventional approaches could not handle the changes, and queue lengths and delays increased continuously. The average performance for the IDCS was about 63% better regarding to vehicles queue lengths and 70% better regarding to delays.

Figure 5 shows the system behavior in face of a simulated traffic accident at the 3,000th second. This was the average results for the whole system. The results at the intersection in which the accident occurred are shown in Figure 6. In the first case, the IDCS performance was about 62% and 85% better regarding to queue lenths and delays, respectively. In the latter case, the IDCS



Approach	Av. Queue	Av. Delay	Max. Delay
IDCS	4.0	15.5	33.8
Simultaneous	10.9	65.7	123.4
Progressive	8.7	50.0	101.3

Figure 8. Comparative results – (5).

performance was about 20% regarding to queue lengths (5.7 against 15.1 vehicles) and 50% better regarding to the average delays (82.4 against 166.3 seconds).

For the next simulations, we provided two different settings for the conventional approaches, corresponding to 100% and 120% of the base arrival rates, following (Wohl and Martin, 1967). We changed the arrival rates as illustrated by Figure 7.

Between 7,113 and 12,887 seconds, the conventional approaches detected the arrival rates change, and were set to 120% of the base arrival rates (Wohl and Martin, 1967). In the other periods, they were set to 100% of the base rates. Figure 8 illustrates the simulation results. The IDCS performance was about 54% better than conventional approaches regarding to queue lengths (4.0 against 8.7 vehicles) and 69% better regarding to average delays (15.5 against 50.0 seconds).

4. CONCLUSION

The incorporation of fuzzy sets concepts into intelligent distributed systems can increase their flexibility and adaptability, leading to an effective approach to develop complex real-world systems. This paper presented a traffic control system built upon a distributed architecture that merges fuzzy logic and artificial intelligence techniques into a unique architecture. The distributed traffic control system presented has shown to achieve coordination and adaptation to environment changes. The performance obtained by the system was superior to conventional approaches, even when the traffic conditions were the same as used to settle the conventional traffic-lights, according to well-known traffic engineering principles.

REFERENCES

Favilla, J., Machion, A. and Gomide, F. (1993). Fuzzy Traffic control: Adaptative strategies, *Second IEEE International Conference on Fuzzy Systems*, San Francisco, CA, 506–511.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publ. Co..

Hoyer, R. and Jumar, U. (1994). Fuzzy control of traffic-lights, *Third IEEE International Conference on Fuzzy Systems*, Orlando, FL, 1526–1531.

Kelsey, R., Bisset, K. and Jamshidi, M. (1993). A simulation environment for fuzzy control of traffic systems, *XII IFAC-World Congress*, Sydney, Australia, 553–556.

Nakamiti, G., Gonçalves, R. and Gomide, F. (2000). Knowldge processing in control systems, *in Knowledge Engineering: Systems, Techniques and Applications*, Leondes, C. (Ed.), **2**, Chapter 16, Academic Press.

Nakamiti, G. and Gomide, F. (1994). An evolutive fuzzy mechanism based on past experiences, *Second European Congress on Intelligent Techniques and Soft Computing -EUFIT 94*, Aachen, Germany, 1211–1217.

Nakatsuyama, M., Nagahashi, H. and Nishizura, N. (1984). Fuzzy logic controller for a traffic junction in the one-way arterial road, *IX IFAC - World Congress*, Budapest, Hungary, 13–18.

Papis, C. and Mamdani, E. (1977). A fuzzy logic controller for a traffic junction, *IEEE Trans. Syst.*, Man, and Cybern., **7**, 707–717.

Skowronski, W. and Shaw, L. (1993). Self-learning fuzzy traffic controller for a traffic junction, I. *European Congress on Intelligent Techniques and Soft Computing - EUFIT 93*, Aachen, Germany, 751–761.

Wohl, M. and Martin, B. (1967). *Traffic System Analysis for Engineers and Planners,* McGraw-Hill Book Co..