ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

G. Felici, G. Rinaldi, A. Sforza, K. Truemper

A METHODOLOGY FOR TRAFFIC SIGNAL CONTROL BASED ON LOGIC PROGRAMMING

R. 581 Dicembre 2002

Giovanni Felici – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email : felici@iasi.cnr.it.

Giovanni Rinaldi – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email :rinaldi@iasi.cnr.it.

Antonio Sforza – Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II", Napoli, Italy. Email: sforza@unina.it.

Klaus Truemper – University of Texas at Dallas, Box 830688, Richardson, TX 75083-0688, USA,Email: klaus@utdallas.edu.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica, CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

In this paper we present a methodology to design, implement, and operate a traffic signal control system based on logic programming. Such system differs from others presented in the literature and available on the market. The use of logic programming results in very flexible control strategies that can be easily developed by traffic engineers. An important feature of the system is the use of a very efficient logic programming solver, the Leibniz System, that is capable of generating fast solution algorithms for the decision problems associated with traffic signal setting. We outline the main principles of the methodology, that includes a heavy use of a traffic micro simulator that has been developed ad hoc and has proved to be a crucial tool for implementing and testing the control strategies before their implementation. An application to a real case is described, and experimental results are presented.

Key words: Traffic Micro-Simulation, Intelligent Traffic Control, Logic Programming

1. Introduction

Traffic control systems are one of the key factors that affect the mobility in large urban networks. The term *Intelligent Traffic Control* has then been adopted to address the latest generation of traffic control methods, that deploy sophisticated modelling and optimisation tools to try and meet the demand for a more efficient and effective way to manage the movements of a large number of vehicles. Modern technologies allow the implementation of several strategies of traffic signal setting, which differ according to the models and methods which they refer to, the costs of production and maintenance and finally for the effects produced.

A basic distinction is the one between strategies based on predefined plans and traffic actuated strategies. In ther first case, mathematical programming methods may be used to construct predefined plans for the different periods of the day, based on traffic data that has been previosuly detected. In the second case, traffic actuated strategies are constructed on the basis of the current configuration of the traffic flows or on the quantity of vehicles present in the controlled junctions; the plan is then constructed or modified, as the situation on the network changes. More detailed considerations on this subject can be found in Cantarella and Festa (1997) and in particular in Cantarella, Sforza and Viola (1997). The traffic actuated systems are also referred to as systems with *adaptive control*, or with *feedback*. The systems with feedback are more complex and expensive, as they need to reliably acquire and elaborate the information in real time. Although, It seems evident that systems with feedback turn out to be potentially advantageous, being based on a larger quantity of information and being able to optimize the use of the road capacity even when the traffic is irregular or in the presence of unforeseeable events or emergencies.

In general, a traffic signal setting system automatically produces decisions for every signalized junction of the traffic network with respect to the cycle length, the sequence of the phases and the green times. To carry out regulation it is necessary to apply a method which determines the values of the decisional variables on the basis of the network configuration and the traffic conditions. A commonly used class of techniques is composed of models of mathematical programming, consisting of a set of given parameters (the configuration of the network and the flows), a set of decisional variables related to the cycle time, phase length and green times, a set of constraints which relate the assigned parameters to the variables, and finally an objective function that represents the performance parameter to optimize in the system. Several such control methods have been proposed and implemented; amongst others, Chin (1987), Heydecher and Dudgeon (1987), Hisai (1988), Moller (1987), and Smith (1987) present methods based on the creation of fixed plans for the signals. Cantarella et al. (1991a, 1991b) propose an iterative approach for equilibrium network traffic signal setting. Successful examples of adaptive systems are the SCOOT system (Hunt et al. (1981), Luk (1984), Robertson (1986), and Bretherton (1996)), SCATS (Luk (1984)), UTOPIA (Mauro and Di Taranto, (1990)), COP (Head, Mirchandani and Sheppard (1992), Sen and Head (1997)); applications of the same type are described in Chen et al. (1987), Gartner (1983), Yagar and Dion (1996) and Skabardonis (1997).

The method proposed in this paper differs substancially from the ones proposed in the literature as it uses logic programming to implement the control strategy of an actuated and distributed control system. It is based on logical variables (that is, variables that can assume the values *true* or *false*), which describe the state of the traffic and the control decisions, and

on relations of a logical type (conjunction and disjunction of logical variables) between the variables considered. Logic programming represent the decision-making process through relationships that are very close to the reasoning of an expert in traffic, and produce decisions for which one may reconstruct the reasons that led to them. These characteristics make a system of regulation based on logic programming closer to the experience of a traffic expert, jumping the stage of representation through a mathematical model and simplifying the process of analysis and development of the regulation system. The system is characterised by independent control units, each associated with a single intersection in the network. Each control unit receives, as input, the traffic data related to the roads approaching the associated intersection, while neighbouring intersections may also exchange synthetic information on the status of the traffic in their proximity. There is no hierarchical relation between the different control units in the network and there is no central unit to master the control process; each unit thus acts independently according to local data. Data communication between control units and detectors is very limited and low-cost connections can be used. A different control strategy based on a large number of logic statements can be used in each control unit. The solution algorithm that determines the control decision is precompiled off-line according to the logic strategy and is then operated on-line in real time with standard commercial computers. The main duty of this algorithm is to decide the time when to cut the current phase of the signal cycle and to select the next phase.

The paper is organised as follows: in Section 2 the proposed methodology is described; in Section 3 the Logic Programming solver adopted to produce the control decision is briefly outlined; Section 4 presents a on-field application to a single signalised intersection. The results are outlined and discussed in section 5, together with some concluding remarks.

2. Methodology

The aim of this section is to describe the methodology to develop an actuated and distributed traffic control system based on logic programming. We illustrate the main feature of the system, that is, the way logic programming is used to implement the control strategy; then, we describe the complete process of developing such system, that includes a visual simulation tool used to design, implement and test the control programs. Additional information can be found in the related references (Felici et al. 1995a, 1995b, 1996, 1997 and 1999).

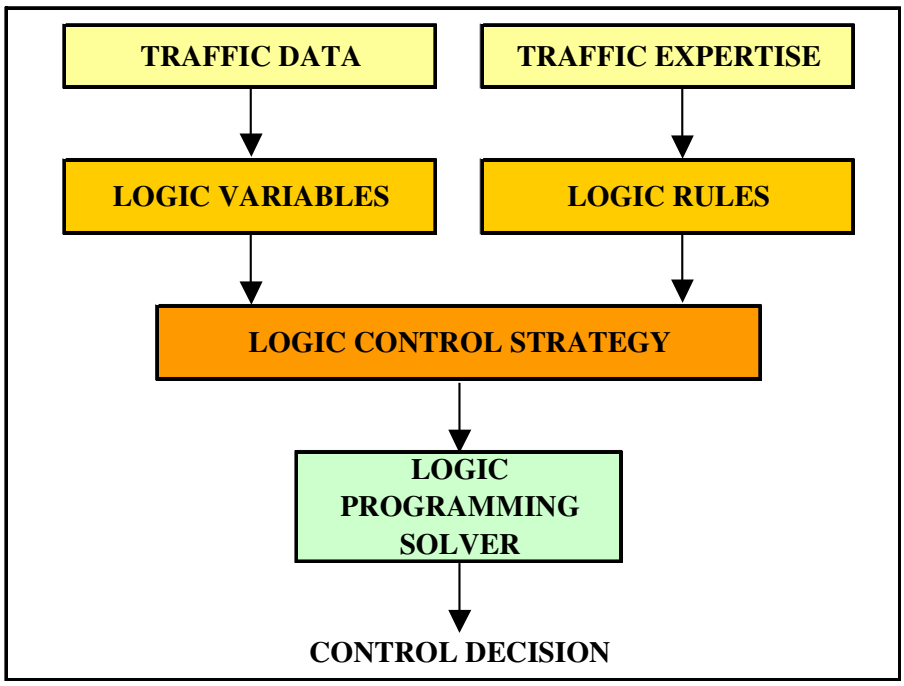
The first step in the formulation of the control strategy is to describe the traffic status via a set of logic variables that, depending on the values detected by the sensors, may assume either the value *true* or the value *false*. A second step is to enrich the description of the traffic status by means of additional logic variables whose values depend on the input variables and on some of the logic statements that compose the control strategy. We call these two types of variables *state variables*. As a third step, the *decision variables* are to be defined. These variables represent the control decisions and are typically associated with the decision of transition from the current phase to another phase of the signal cycle.

By combining state and decision variables a set of *logic statements* is created, using the standard rules of propositional logic; for example, if V_1, V_2, V_3, V_4, and V_5 are state variables

and D_1 and D_2 are decision variables, the following is a valid logic statement:

$$\textit{if} \quad V_1 \textit{ and } V_2 \textit{ and } V_3 \textit{ and not } V_4 \textit{ and } V_5 \quad \textit{then} \quad D_1 \textit{ or } D_2 \,.$$

A set of logic statements such as the one described above forms a logic program representing the *control strategy*. The control strategy is in charge of producing, at regular time intervals, a control decision, i.e., of deciding whether to change the configuration of the signal lights or not. This task is accomplished using theorem proving in propositional logic. A *control unit*, where the logic strategy is put into action, is associated with each signalised



intersection. The general scheme is depicted below in Figure 1.

4.

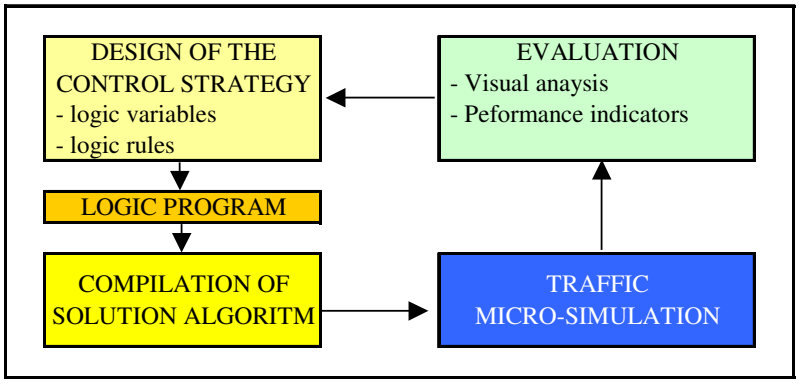
Figure 1: General scheme of the system

The logic strategy is then embedded in a general development system, with the purpose of supporting the traffic engineer in the design, tuning, and maintenance of the control system. The main objectives of the development system are:

- To provide a simple and user-friendly interface which enables a user without expertise in logic programming to develop a control strategy by the proposed system and to efficiently produce decisions based on this strategy;
- To verify the correct functioning of the control system under varying traffic conditions by means of a graphic micro-simulation system;
- To analyse the system's performance by means of simulation, aiming on one hand at tuning the logic model on the traffic flows, and on the other hand at conducting parallel experiments with other control systems in order to evaluate the improvements obtained with the proposed system.

The development system is made of a visual interface that implements a graphic micro-simulation system reproducing the road network, where the signals and the vehicles are visualised. The interaction between the on-screen animation and the traffic control system is automated, as well as the management of the control strategy, which can be easily edited, changed, and recompiled by the user while the simulation is running. Figure 2 represents a general scheme of the development system.
In Figure 3 a simulation session is represented for a grid network with six controlled intersections, showing the road network, the vehicles, the signals in their current state and several performance indicators associated with the different control units. Each lane can be assigned different vehicle generation parameters that can vary as the simulation time progresses. Additional parameters that can be tuned in the generation process are the speed of vehicles and the duration of high traffic and low traffic periods that can be used to emulate a traffic pattern produced by an external signal controlled intersection.

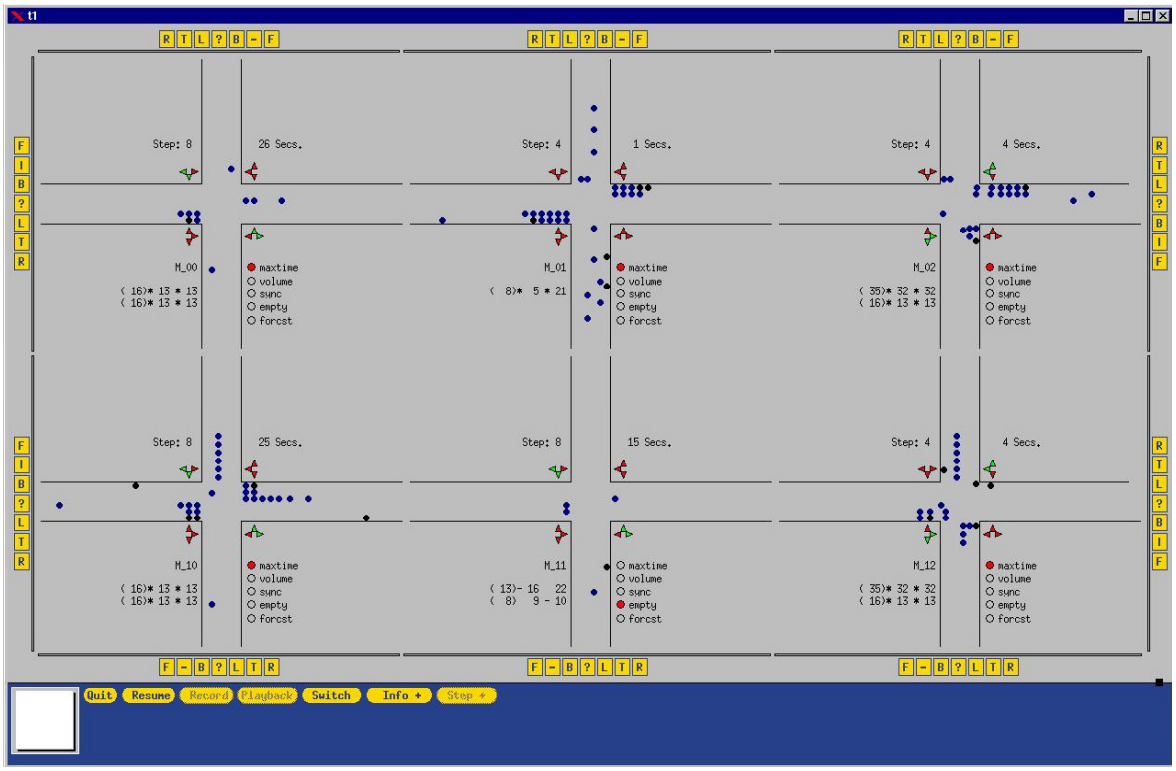
Figure 2: Development



The system

Figure 3 - Grid simulation session 2×3

To provide a complete description of the way a control strategy is designed, implemented, and solved, we make use of a simple example of signal control problem. Let us consider a simple intersection of two roads, where both senses are



allowed on the roads and vehicles can turn in any direction after the signal. The signal cycle is composed of four phases, whose composition is described in Figure 4. The sequence of the phases is 1-2-3-4, but phases 1 and 3 may be skipped from the cycle. The allowed transitions

between the phases are represented on the transition graph of Figure 5.

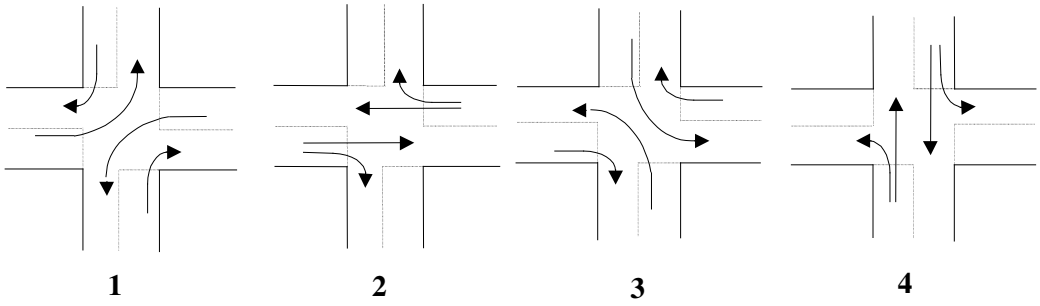


Figure 4: The 4 phases of the cycle

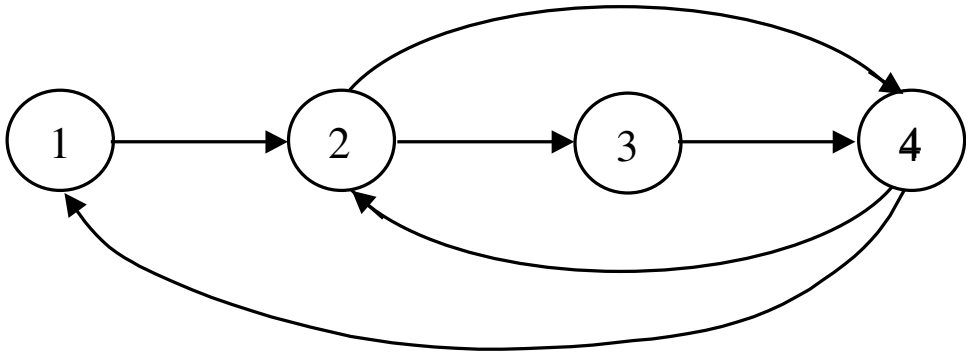


Figure 5: Phase Transition Graph

We also assume that, by means of proper traffic detectors, it is always possible to know whether there are stopped vehicles or moving vehicles in each one of the four accesses of the intersection.

The initial step consists in defining the logic variables and logic predicates that describe the state of the traffic nearby the intersection. A logic variable is a variable that can assume

the value *true* or *false*. A logic predicate is a function defined on one or more finite sets that assumes as well the values true or false.

On the set of the phases {1,2,3,4} we define the predicate *step(i)*, that is equivalent to the four logic variables *step(1)*, *step(2)*, *step(3)*, and *step(4)*, with the following meaning:

- *step(i)* has value true at time t if the signal is in phase i, and false otherwise.

Then, we use the predicate *maxtime(i)* to express the reaching of the maximum time allowed for phase i; assume that a timer is present that counts the time expired in the current phase; then:

- *maxtime(i)* has value true if the maximum time allowed for phase i has expired, and false otherwise.

We now turn to consider some logic predicates that are related with the number of vehicles detected in the proximity of the intersection (say, a 100 meters before the signal stop). We say that an access is "served" by the current phase if the current phase gives green light to that access. Thus we have the following predicates, defined on the set of the phases:

- *empty(i)* has value true if there are no vehicles in the accesses served by phase i, and false otherwise;
- *wait(i)* has value true if there are stopped vehicles in the accesses served by phase i, and false otherwise;
- *cong(i)* has value true if traffic on one of the accesses served by phase i is congested, and false otherwise.

Finally, to represent the control decisions, we define on the set of the phases the predicate:

- *go_a_step(i)*, that has value true if it has been decided to interrupt the current phase and commute to phase i, and false otherwise.

The stage is now set for describing some simple logic relations that express the control strategy. First, we make sure that no phase lasts longer than the its maximum allowed time. The following logic rules accomplish this task for any possible transition described by the graph in Figure 5.

IF step(1) AND maxtime(1) THEN go_to_step(2).
IF step(2) AND maxtime(2) THEN go_to_step(3) OR go_to_step(4).
IF step(3) AND maxtime(3) THEN go_to_step(4).
IF step(4) AND maxtime(4) THEN go_to_step(1) OR go_to_step(2).

Second, we design some logic rules that enforce the commutation of the phase if there are no vehicles served by the current phase and there are vehicles waiting in one of the phases to which transition is allowed. When transition ot more than one phase is possible, the rules below also ensure that only one new phase is proposed.

IF step(1) AND empty(1) AND wait(2) THEN go_to_step(2).
IF step(2) AND empty(2) AND wait(3) THEN go_to_step(3).
IF step(2) AND empty(2) AND NOT wait(3) AND wait(4) THEN go_to_step(4).
IF step(3) AND empty(3) AND wait(4) THEN go_to_step(4).
IF step(4) AND empty(4) AND wait(1) THEN go_to_step(1).
IF step(4) AND empty(4) AND NOT wait(1) AND wait(2) THEN go_to_step(2).

Third, we consider the predicate cong(i), writing a set of logic rules that impose the commutation of the phase when the accesses currently served are not congested, while those that are to be served by one of the next phases are:

IF step(1) AND NOT cong(1) AND cong(2) THEN go_to_step(2).
IF step(2) AND NOT cong(2) AND cong(3) THEN go_to_step(3).
IF step(2) AND NOT cong(2) AND NOT cong(3) AND cong(4) THEN go_to_step(4).
IF step(3) AND NOT cong(3) AND cong(4) THEN go_to_step(4).
IF step(4) AND NOT cong(4) AND cong(1) THEN go_to_step(1).
IF step(4) AND NOT cong(4) AND NOT cong(1) AND cong(2) THEN go_to_step(2).

Finally, we ensure that the transition to phases 2 and 4 is unique, by means of the two following rules:

IF step(2) THEN (NOT go_to_step(3) OR NOT go_to_step(4)).
IF step(4) THEN (NOT go_to_step(1) OR NOT go_to_step(2)).

Combining all the logic rules described above, we obtain a complete logic control strategy (Table 1).

(1)	IF	step(1)	AND	maxtime(1) THEN go_to_step(2)
(2)	IF	step(2)	AND	maxtime(2) THEN go_to_step(3) OR go_to_step(4).
(3)	IF	step(3)	AND	maxtime(3) THEN go_to_step(4).
(4)	IF	step(4)	AND	maxtime(4) THEN go_to_step(1) OR go_to_step(2).
(5)	IF	step(1)	AND	empty(1) AND wait(2) THEN go_to_step(2).
(6)	IF	step(2)	AND	empty(2) AND wait(3) THEN go_to_step(3).
(7)	IF	step(2)	AND	empty(2)
			AND	NOT wait(3) AND wait(4) THEN go_to_step(4).
(8)	IF	step(3)	AND	empty(3) AND wait(4) THEN go_to_step(4).
(9)	IF	step(4)	AND	empty(4) AND wait(1) THEN go_to_step(1).
(10)	IF	step(4)	AND	empty(4)
			AND	NOT wait(1) AND wait(2) THEN go_to_step(2).
(11)	IF	step(1)	AND	NOT cong(1) AND cong(2) THEN go_to_step(2).
(12)	IF	step(2)	AND	NOT cong(2) AND cong(3) THEN go_to_step(3).
(13)	IF	step(2)	AND	NOT cong(2)
			AND	NOT cong(3) AND cong(4) THEN go_to_step(4).
(14)	IF	step(3)	AND	NOT cong(3) AND cong(4) THEN go_to_step(4).
(15)	IF	step(4)	AND	NOT cong(4) AND cong(1) THEN go_to_step(1).
(16)	IF	step(4)	AND	NOT cong(4)
			AND	NOT cong(1) AND cong(2) THEN go_to_step(2).
(17)	IF	step(2)		THEN (NOT go_to_step(3) OR NOT go_to_step(4)).
(18)	IF	step(4)		THEN (NOT go_to_step(1) OR NOT go_to_step(2)).

Table 1 – The complete logic control strategy

The steps to produce the control decision at time t based on the rules described above are:

1. Determine the true/false values of the logic predicates at time t;
2. verify whether any predicate that is associates to a control decision is a theorem of the

system of logic rules or not.

Step 1 is easily dealt by transforming the numerical values detected in the proximity of the intersections into logic values, according to the definition of the predicates.

Solving Step 2 amounts to theorem proving in propositional logic. Such problem has been studied in many contexts, and is typically difficult to solve. We adopt a particular technique to produce the solution, implemented in the Leibniz Solver for Logic Programming (Truemper, 1988), that transform the control strategy and the associated theorem proving problem into a satisfiability problem (or a minimum cost satisfiability problem) and compiles a fast solution algorithm with guaranteed solution performances.

3. The Logic Programming Solver

The control units use the Leibniz System for Logic Programming to produce a control decision. Below, we give an example to clarify the way the system works. Consider the logic formula below:

$$\textit{if}\ (V_1\ \textit{and}\ V_2\ \textit{and}\ V_4\ \textit{and not}\ V_n)\ \textit{then}\ (D_1\ \textit{or not}\ D_2\ \textit{or}\ D_m)$$

where the logic variables $V_1,...,V_n$ represent the traffic status and the decision variables $D_1,...,D_m$ are associated with the control decisions. The types of formulas that can be expressed by the Leibniz System span all the propositional logic, and include formulas such as:

$$V_1\ \textit{or}\ V_2\ \textit{or}\ D_1\ \textit{or not}\ D_2\ \textit{or}\ D_m.$$

When many logic rules as the one described above are present the determination of the set (possibly empty) of decisions implied by the set of rules may require considerable computation time. Several algorithms have been proposed in the literature. While some of these behave extremely well for certain classes of problems, they cannot guarantee short solution times in the general case. The Leibniz System problems produces a tight upper bound on the maximum response time for solving any instance of a given logic programming problem. This characteristic, obtained by means of several results related to the decomposition of combinatorial systems is fundamental for an effective deployment of logic programming in real time applications such as the one here described. A complete description of the Leibniz System goes beyond the scope of this paper; an extensive treatment of the theoretical foundations of this method can be found in Truemper (1998). Below, we provide a brief sketch of this approach. The Leibniz System operates in three main steps:

1. it analyses the combinatorial structure of a given logic program converting in into a satisfiability problem (SAT) or a minimum cost satisfiability problem (MINSAT) instance, using a number of fast algorithms that identify substructures for which the problem is easier to solve;
2. deploys ad-hoc polynomial time algorithms to solve each of the easy identified substructures, and then combines the corresponding solutions using results derived from the theory of matroid decomposition to provide the solution for the complete problem;
3. compiles the above process into an ad hoc algorithm that solves each instance derived

from the given one by fixing variables and/or eliminating clauses, and calculates an upper bound on the running time of such algorithm over all possible instances derived in this way. This algorithm can then be used stand-alone or invoked from a C program by means of library functions.

In our experience, the logic strategies developed for many real and simulated control problems have several hundred logic variables and several hundred logic statements. The Leibniz System has always produced solution algorithms with an upper bound on the solution time below 0.1 seconds on a commercial personal computer, thus making it possible to design control units that produce reliable and timely decisions.

4. The Application

The proposed traffic control system has been applied in a single intersection of a town in the South of Italy (Afragola). Traffic detection is carried out by four cameras that continuously provide information on the volume of traffic. In each one of the four approaches that compose the intersection, vehicles can go straight, turn left or right. This clearly creates many conflicts already within the same green phase, as turning vehicles have to give way to other vehicles coming in the opposite direction. A representation of the intersection is given in Figure 6.

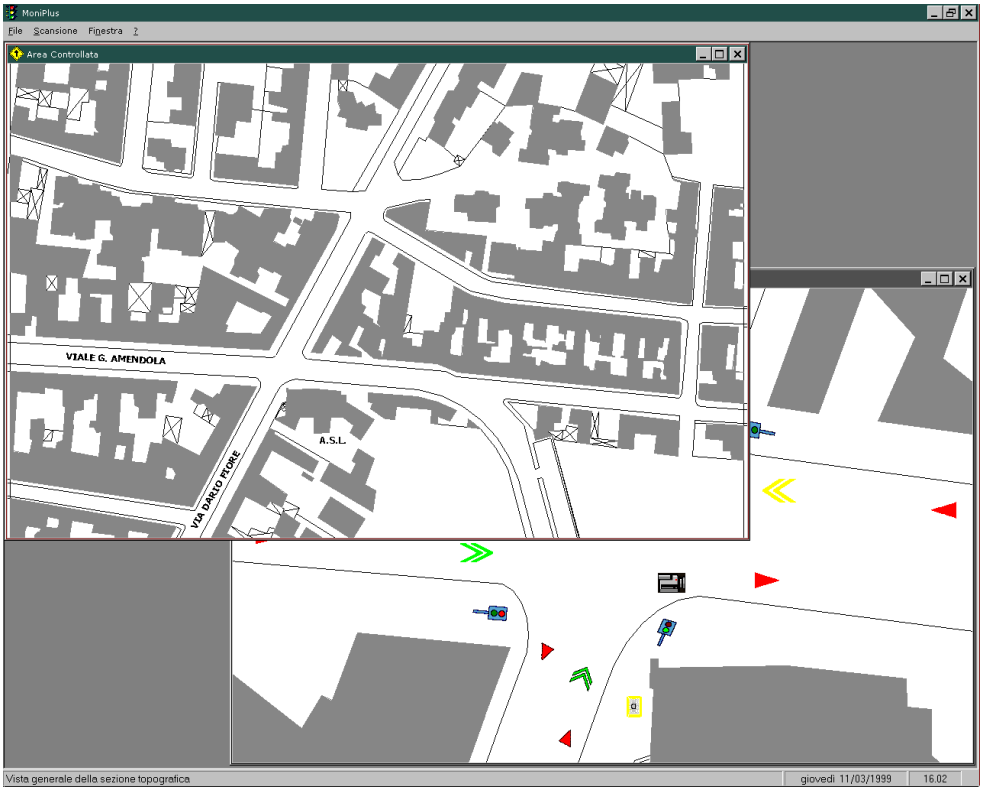


Figure 6: The intersection

The overall system is composed of different modules that exchange information using standard Windows NT interprocess communication protocols:

- *Control Module*: requests traffic data, executes the logic program and produces the control decisions;
- *Monitoring Module*: continuously monitors the performance of the control and the traffic detection system, notifying and recording any change of state in the system;
- *Diagnostic Module*: this module checks the functioning of the cameras and of the detectors. If some errors are detected, such as inconsistency of traffic data or inactivity of the counters, it notifies the other modules;
- *Front-End Module*: this module is in charge of the communication with the signal via an optical fibre connection. Any message that comes from and goes to the signal lights (i.e., «change phase», «change plan», «stop control», etc.) is managed by the Front-End which receives the message from the other modules, sends the message to the signal, waits for the proper answer from the signal, and then returns this answer to the module that originated the message;
- *Database Module*: all the information on the state of the system (the traffic volumes, the diagnosed errors, and the signal setting parameters) are sent from each module to this module, which stores the data for future analysis in a relational database and produces synthetic reports using charts and tables.

The four cameras provide, in real time, counts and occupancy times for 16 presence detectors and 12 queue detectors placed at the intersection.

4.1 The Logic Control Strategy

The control strategy developed for this intersection takes into account a simple plan made of two phases (plan 1), and a more complex plan with 4 phases (plan 2) where two additional asymmetrical phases are added to those of the first plan. Below we describe the logic variables and the logic rules that are involved in the construction of the logic strategy. Logic Variables, Logic Predicates, Logic Rules. The logic variables and predicates are divided into three main categories: *state variables*, that directly translate the detected data into logic values; *traffic variables*, that provide a higher level description of the state of traffic on the intersection; and *decision variables*, that describe the decision that are to be taken by the control system (termination the current phase, select the next phase to switch to). We use the sets *segments*, that identify the 4 road segments composing the intersection, and *positions*, the identify different portions of each segment according to the distance from the signal.

For example, we use the logic predicate *counter* to represent the fact that a vehicle has passed on a counter in the last detection interval. *counter*(s_0,p_1) will have value *true* if the counter in position p_1 on the segment s_0 has seen a car, and false otherwise. In the following, the variables and the predicates are listed, together with a brief description of their meaning.

STATE VARIABLES/PREDICATES

Counter	Current value of counting detectors
Oldcount	Previous value of counting detectors
Zone	Current value of queue detectors
Oldzone	previous value of queue detectors
Camera	Camera status
Phase	Current phase
Expc	Expired percentage of *target time*
Queue2	level of vehicles detected in the portion close to the signal
Arrival2	level of vehicles detected in the portion more distant from the signal
Timemax	Reach of the maximum time allowed for the current phase
plan1	Activation of *plan 1*
plan2	Activation of *plan 2*

TRAFFIC VARIABLES/PREDICATES

Queue	level of vehicles in the portion close to the signal (computed by logic rules)
Arrival	level of vehicles in the portion more distant the signal (computed by logic rules)

DECISION VARIABLES

do_step	next phase proposed
go_step	new phase
go_a_step	decision of interrupting the current phase
Change_plan	decision of changing the current plan (2 or 4 phases)
Change_trg	decision of modifying the *target time*

The control interval adopted in the application is 3 seconds: that is, every 3 seconds the control module queries the front-end module and obtains the traffic values detected since the previous query. Such values are then used to set the True/False values of the logic variables and predicates that describe the status of the traffic: *counter*, *zone*, *maxtime*, *expc*, *camera*, *phase*, *plan1*, and *plan2*. The predicates *queue* and *arrival* are used to represent the state of traffic at a higher level and are those used in the rules that determined the decision variables. The correct setting of these variables is very important for the good behavior of the control system; for this reason, the main effort in determining their values is oriented towards a conservative transformation of the detected data into the logic values, that is able of tolerating anomalous data or errors derived from the detection devices. In particular, we have realized that:

- Vehicles may be erroneously counted by the detectors that are closed to the signal;
- Detectors may be fooled by wrongly parked vehicles, heavy vehicles, anomalous light conditions;
- Cameras may go out of service for significant periods for technical reasons, for some of which assistance is needed.

To overcome these problems, the logic values of queue and arrival are computed in such a way that guarantees a stable and correct, although conservative, behavior in any

circumstance. In the logic strategy three different classes of logic rules are present, that are alternatively used according to the reliability of the detection system:

- Rules that directly compute the logic values of *queue* and *arrival* directly from the detected data;
- Rules that indirectly compute the logic values of *queue* and *arrival* directly from the values of *counter* and *zone*;
- Rules that estimate the logic values of *queue* and *arrival* when one or more cameras are not functioning correctly.

For the plan with 4 phases we have introduced additional logic rules that impose the switching to the asymmetric phases and their termination. The asymmetric phases are activated in the presence of traffic on one of the served segment and absence of traffic in the other one; the phase is interrupted when the a consistent number of vehicles is waiting to be served in the other segments. The change from the 2-phase plan to the 4-phase plan is itself managed by logic rules. The logic variable *change_plan* is set to *true* when the traffic condition suggest so and returned to the control code, that is in charge of practically changing the plan in the signal hardware and managing the transition from the last phase of the old plan and the first one of the new plan.

A new concept that we have introduced in the construction of the control strategy is the *target time*. The *target time* of a phase is a dynamic value that indicates a reasonable value for the duration of that phase. In the control strategy are present different groups of logic rules that are activated according to the percentage of the *target time* that has already expired in the current phase. Its interpretation is the following: the conditions required to change the phase duration when this value is distant from the current *target time* (much smaller of much greater) are more strict, while they become milder when the current duration of a phase is in the proximity of its *target time*. The value of the *target time* of a phase is changed from one cycle to the next based on the actual duration of that phase. The overall effect of this operation is a more stable behavior of the control system, as it avoids abrupt changes in the duration of a phase from one cycle to the other, unless strong variations in the pattern of the traffic approaching the intersection occur. While the decision to terminate the current phase is driven in the logic rules by the current state of the traffic, the use of the *target time* rules creates a smoothing effect on the transitions from cycle to cycle. The *target time* is modified at the end of each phase according to the formula:

$$\mathrm{ntt} = \mathrm{ott} + \frac{(\mathrm{pd} - \mathrm{ott})}{\mathrm{k}}$$

where *ntt* is the new *target time*, *ott* is the current *target time*, *pd* is the duration of the phase (as determined by the logic rules, that may or may not terminate the phase in the proximity of the *target time* depending on the traffic conditions) and *k* is a parameter greater that 1 that determines the sensitivity of the target time to the decision of the logic control strategy.

We omit for brevity a complete description of the logic rules. Below we simply report the main classes of logic rules used to define the control strategy:

- *volume* logic statements: these statements are used to derive the *true* / *false* values of the logic variables that represent traffic levels from the counts provided by the cameras;

- *control* logic statements: these statements associate the traffic levels to the decisions of changing the current phase or changing the structure of the current cycle (2 or 4 phases); they can be subdivided into:
- *maxtime* logic statements, which decide to change phase when the maximum time allowed for the current phase is expired;
- *congestion* logic statements, which decide to change phase when there are many waiting vehicles;
- *empty* logic statements, which decide to change phase when there are few vehicles using the
- current green phase;
- *camera fault* logic statements: as anticipated, the cameras have frequent faults and erratic behaviour. A diagnostic system detects these problems and informs the control unit if a camera is not working properly. Statements of this type produce an estimate of the traffic levels for the cameras that are not working; such an estimate is based on traffic volumes on the other approaches, on phase time, and on historical data.

The above described Logic Program has 104 logic variables and 185 logic statements; the Leibniz system solves any instance of this problem in at most 0.05 seconds on a Pentium II 200 Mhz processor.

The application on field of this method, that had already been extensively tested by simulation, has shown its usefulness in dealing with incorrect or partial traffic data, as it is the case with the counts provided by the cameras. It has also made it possible to adjust the control strategies in real time according to the suggestions of the local traffic experts. No particular problems were encountered in tuning the small set of parameters used, namely those that regulate the mapping of the traffic counts into the logic variables.

5. Results

We have run the controlled intersection with our system and with other two competing systems, and then we have evaluated the system at work in two different ways. First, we have computed some performance indicators based on the traffic volumes detected by the cameras; second, we have used a *floating probe car* that was driven around the intersection in predetermined paths when the three different control systems were running. From the data on fuel consumption, air pollution and travel time collected by the probe car we have derived additional performance indicators.

The two alternative control systems that we have used for the comparison are:

- a fixed plan system, in the following referred to as «MANUAL», where the length of the cycle and the duration of the phases has been determined with the software TRANSYT (Robertson (1986)) on the basis of the traffic volumes of the intersection;
- a commercial adaptive control system, embedded in the equipment installed at the intersection, distributed by SelfSime, in the following referred to as «DYNAMIC». This system adopts as inputs for the adaptive dynamic control the same traffic values

detected by the cameras which are used by the logic control strategy.

As for the first type of performance indicators, we have done the following. We measure the state of the traffic at the intersection in a given time interval with the total occupancy time of the queue detectors. This value directly measures the fluidity of traffic in the detected area; under the same traffic volumes, differences in the total occupancy times in the queues are to be attributed to the action of the control strategies. Then, we compare queue occupancy times derived from the application of the different control strategies only if applied during the same time interval of the day; moreover, we consider time intervals of small dimensions, ranging from approximately 30 to 60 minutes. Finally, we normalise the occupancy values using a measure of the traffic that has used the intersection in the time interval. We derive such measurements from the four count detectors that count the vehicles leaving each one of the four approaches of the intersection. The final performance indicator is then obtained by the ratio between the sum of the queue occupancy time and the traffic volume. Moreover, to make the comparison process more reliable, we compare only those time intervals for which the total traffic volume is similar. From all the experimental sessions run, we have selected only a restricted number of time intervals where data with good quality were available, and then we have constructed the occupancy indicators. For each time interval, we have also averaged the performance indicators of all the homogeneous experimental sessions conducted with the same control strategy. As a result, we have at hand 16 different time intervals, of variable length (30 to 60 minutes) spanning the daylight hours, where the 3 strategies can be compared, as reported in Table 2, where the logic control strategy described in this paper is indicated as "LOGIC".

The numbers of the table show that the LOGIC strategy results, in almost all sessions, in savings of queuing time when compared with the other two methods. We now turn to consider the results obtained with the floating probe car. This car is a specially equipped vehicle that is able to collect a full set of data related with the functioning of the car, that was driven around on 14 different paths around the area where the intersection is located. Here number of stops (Table 3), time spend on the segments (Table 4), fuel consumption and emissions of CO and HC (Table 5) are considered. In the tables we compare the values of these indicators for LOGIC, MANUAL and DYNAMIC. The values of interest are the ones recorded by the car while it was travelling on the 4 segments composing the intersection; these values are computed distinctly for the 4 segments and then averaged. Also in this case the experimental results show a consistent advantage in adopting the proposed method.

16.

TIME INTERVAL		PERFORMANCE INDICATOR		
		MANUAL	LOGIC	DYNAMIC
1	10-11	4,516	4,138	4,396
2	12,15-13	3,649	3,131	3,401
3	14,10-14,45	3,283	2,121	2,323
4	15,10-16	3,238	2,371	2,459
5	16,30-17,03	N.D.	3,181	3,460
6	16,01-20	N.D.	3,814	3,912
7	16,01-17	N.D.	2,871	3,027
8	17,01-18	N.D.	3,876	3,832
9	18,01-19	N.D.	4,116	4,137
10	19,01-20	N.D.	4,094	4,296
11	10,30-16	3,654	3,340	3,505
12	11-12	4,992	4,687	4,970
13	12,01-13	3,354	3,224	3,710
14	13,01-14	3,302	2,885	2,876
15	14,01-15	3,421	2,470	2,601
16	15,.01-16	2,898	2,328	2,457

Table 2: Performance comparison based on detected traffic

STOPS ON SEGMENTS (< 10kmh)			
SEGMENT	DYNAMIC	LOGIC	SAVINGS
1	9,69	12,26	26,5%
2	16,78	5,67	-66,2%
3	6,45	4,00	-38,0%
4	11,58	13,69	18,2%
average	11,13	8,90	-20,0%

Table 3: Performance comparison with probe car: number of stops

TIME ON SEGMENTS			
SEGMENT	DYNAMIC	LOGIC	SAVINGS
1	53,33	64,38	20,7%
2	97,56	43,17	-55,8%
3	50,58	48,50	-4,1%
4	74,00	82,00	10,8%
average	68,87	59,51	-13,6%

Table 4: Performance comparison with probe car: time on segments

SEGMENT	FUEL			HC			CO		
	DYN	LOG	SAV	DYN	LOG	SAV	DYN	LOG	SAV
1	12,51	14,51	16%	2,59	3,04	17%	43,22	53,23	23%
2	17,42	11,99	-31%	3,73	2,44	-34%	68,54	40,05	-42%
3	12,97	12,21	-6%	2,65	2,49	-6%	44,76	41,09	-8%
4	14,67	15,71	7%	3,07	3,25	6%	54,07	57,94	7%
average	14,39	13,60	-5%	3,01	2,81	-7%	52,65	48,07	-9%

Table 5: Performance comparison with probe car: consumption/emissions

References

Bretherton R.D., 1996. Current Development in SCOOT: Version 3, Transportation Research Record, 1554.

Cantarella G.E., Improta G., Sforza A., 1991a. Road Network Signal Setting: Equilibrium Conditions. In Concise Encyclopedia of Traffic & Transportation Systems, 1991., editor M. Papageorgiou. Pergamon Press.

Cantarella G.E., Improta G., Sforza A., 1991b. Iterative Procedure for Equilibrium Traffic Signal Setting, Transp. Res. A 25, 241-249

Chen H., Cohen S.L., Gartner N.H., and Liu C.C., 1987.,Simulation Study of OPAC: A Demand-Responsive Strategy for Traffic Signal Control, in: Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds., Elsevier Science Publishing Co.

Chin S.M., 1987. Theoretical Considerations for Signal Timing Plan Selection in UTCS First Generation Control Systems, in: Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Felici G., Rinaldi G., and Truemper K., 1996. FasTraC: A Decentralized Traffic Control System Based on Logic Programming in: Proceedings of the 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, 30 july-3 august 1996, 216-220.

Felici G., Rinaldi G., and Truemper K. , 1995a. Development of a Decentralized Traffic Control System Based on Logic Programming, in: Proceedings of the fourth International Conference of Advanced Technologies in Transportation Engineering, Capri, Italy, 27-30 june 1995, 573-577.

Felici G., Rinaldi G., Truemper K., 1995b., Controllo decentralizzato del traffico tramite programmazione logica, atti del II Convegno Nazionale del Progetto Finalizzato Trasporto II, CNR, Genova, 29-31 Maggio 1995, pp. 1953-1967.

Felici G., Rinaldi G., Truemper K., 1997., *Un sistema di progettazione di una Rete di Controllo Distribuito del traffico*, atti del III Convegno Nazionale del Progetto Finalizzato Trasporto II, CNR, Taormina, 12-14 Novembre.

Felici G., Rinaldi G., Truemper K., 1999., *Controllo decentralizzato del traffico tramite programmazione logica: Sviluppi ed Applicazioni*, atti del IV Convegno Nazionale del Progetto Finalizzato Trasporti II, CNR, Roma, Italia, 3-5 Novembre 1999.

Felici G., Rinaldi G., Sforza A., Truemper K., 2001. Traffic Control: a Logic Programmino Approach and a Real Application, *Ricerca Operativa*, No 94-95, Franco Angeli Editore, 2000 39-60.

Gartner N.H., 1983. OPAC: A Demand-Responsive Strategy for Traffic Signal Control, Transportation Research Record, 75-81.

Head K.L., Mirchandani P.B., and Sheppard D., 1992. Hierarchical Framework for Real Time Traffic Control, Transportation Research Record, 1324, 82-88.

Heydecker B.G., Dudgeon I.W., 1987. Calculation of Signal Settings to Minimise Delay at a Junction, in: Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Hisai M., 1987. Delay-Minimizing Control and Bandwidth-Maximising Control of Coordinated Traffic Signals by Dynamic Programming, in: Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Hunt P.B., Robertson D.I., Bretherton R.D., and Winton R., 1981. SCOOT-A Traffic Responsive Method of Coordinating Signals, Transportation and Road Research Laboratory Report, No. LR 1014, Crwothorne, Berkshire, England.

Luk J.Y.K., 1984. Two Traffic Responsive Area Traffic Control Methods: SCAT and SCOOT, Traffic Engineering and Control, Vol. 30, 14-20.

Mauro V., Di Taranto D., 1990., UTOPIA, Proceedings of the 6th IFAC / IFIP / IFORS Symposium on Control Computers and Communication in Transportation, Paris, France.

Moller K., 1987. Calculations of Optimum Fixed-Time Signal Programs, in Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Robertson D.I., 1986. Research on TRANSYT and SCOOT Methods of Signal Coordination, ITE Journal, Vol. 56, No. 36-40.

Sen S., Head K.L. , 1997. Controlled Optimization of Phases at an Intersection, Transportation Science, Vol. 31, No. 1 5-17.

Sistema Leibniz, *Leibniz* System for Logic Programming Ver. 4.0, Reference Manual, *Leibniz* Inc., Plano TX, U.S.A.

Skabardonis A., 1996. Determination of Timings in Signal Systems with Traffic-Actuated Controllers, Transportation Research Record, 1554 18-26.

Smith M.J., 1987. Traffic Control and Traffic Assignment ina Signal-Controlled Network with Queueing Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Smith M.J., VanVuren T., Heydecker B.G., and VanVliet D., 1987. The Interactions Between Signal Control Policies and Route Choice, Transportation and Traffic Theory, N. H. Gartner and N.N.M. Wilson, eds.., Elsevier Science Publishing Co.

Truemper K.,1998. Effective Logic Computation, Wiley-Interscience Pub., New York.

Yagar S., Dion F, 1996. Distributed Approach to Real-Time Control of Complex Signalized Networks, Transportation Research Record, 1554 1-8.