A Demand-Responsive Traffic Control System for Urban Areas

Author: Rafel Grau Mariani

Ph.D. Thesis

Director: Prof. Jaume Barceló i Bugeda
Departament of Statistics and Operational Research
Faculty of Computer Science
Universitat Politècnica de Catalunya

Barcelona, July 1994

UPC
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament d'Estadística i Investigació Operativa

SECCIÓ D'INFORMÀTICA

Edifici FIB
Pau Gargallo, 5
08028 Barcelona
Tel. (93) 401 69 48
Fax (93) 401 70 40
E-mail: eio@eio.upc.es

D. JAIME BARCELO BUGEDA, CATEDRATICO DEL DEPARTAMENTO DE ESTADISTICA E INVESTIGACION OPERATIVA DE LA UNIVERSITAT POLITECNICA DE CATALUNYA.

CERTIFICA: Que la memoria titulada: " A Demand-Responsive Traffic Control System for Urban Areas", ha sido realizada bajo su dirección en el Departamento de Estadística e Investigación Operativa de la Universitat Politécnica de Catalunya por D. Rafael Grau Mariani, y constituye su Tesis para optar al grado de Doctor en Informática.

Y para que conste, en cumplimiento de la legislación vigente, la presenta ante la Facultad de Informática de la Universitat Politécnica de Catalunya.

Barcelona, 8 de Septiembre de 1994

El Director

[illegible]

Fdo. Jaime Barceló Bugeda

A Demand-Responsive Traffic Control System for Urban Areas

Author: Rafel Grau Mariani

Ph.D. Thesis

Director: Prof. Jaume Barceló i Bugeda
Departament of Statistics and Operational Research
Faculty of Computer Science
Universitat Politècnica de Catalunya

Barcelona, July 1994

Acknowledgements

To Jaume Barceló, my supervisor, for his support in my research and his forbearance when some other project became urgent. He introduced me to the world of Urban Traffic Control.

To the staff in the Transportation Research group of the Department of Statistics at the Universitat Politècnica de Catalunya, those who are currently there or have passed through during recent years: Sergi Benedito, Jordi Casas, Esteve Codina, Pilar Egea, J.L. Ferrer, David Garcia, Olga Llima, Sergi Maymí, Lidia Montero, and Enrique Palma. Without the collaboration of all of them this work would hardly have been feasible. Sergi Benedito programmed a first version of a predictor of vehicle arrivals in CARS V1, and prepared the tests with AIMSUN1. Pilar Egea and Olga Llima programmed the CARSedi graphical editor, a precursor of TEDI. David Garcia, Olga Llima, and Sergi Maymí programmed and contributed many useful comments relating to the TEDI graphical editor. J.L. Ferrer agreed to design AIMSUN2 in the GETRAM environment. Jordi Casas helped to design the messaging protocol between AIMSUN2 and CARS V2, and programmed the AIMSUN2 part.

To Panos Michalopoulos, Eil Kwon, Ping Yi, Xiao Liu and Yorgos Stephanedes of the Civil and Mineral Engineering Dept. (University of Minnesota), and to Bernhard Friedrich of the Fachgebiet Verkehrstechnik und Verkehrsplanung, Technische Universität München, for their useful comments and, particularly, for the feeling of doing something exciting.

To Montse, without whose support I would not have had the willpower to accomplish the — most of times — terribly boring act of writing technical-scientific documentation, or this thesis would be much shorter, or... anyway, life would have been different.

Barcelona, July 1994

Rafel Grau

To YOU.

Contents

Structure of this document

The goal of this Ph.D. Thesis is the design, development and testing of a 'well-engineered system' aimed at the demand-responsive traffic control of urban areas. Because of the author's background — software engineering —, a 'well-engineered system' means a system that is both efficient at performing its task in a wide range of conditions and also that has been built from the robust design's and ease-of-use's points of view. The work has proceed in four steps and, accordingly, this document has been divided in four parts that are briefly described here.

PART I is, mainly, an introduction to the subject. It introduces the concept of demand-responsiveness in traffic control, providing a historical perspective from traditional fixed-control systems to current modern demand-responsive systems. The part concludes with a study of areas where research and improvement is needed. All of these areas are considered in the following parts, where some solutions are proposed.

PART II presents the design, development and preliminary tests of a demand-responsive traffic control system, CARS V1, intended to operate in signalized urban areas: networks, arterials, and isolated intersections. A graphical user interface, X-Windows and DecWindows compliant, allows the user to specify network characteristics in a friendly and intuitive manner, without the need to be acquainted with the actual modeling. The system features an underlying simulation system and a prediction model based on real-time measured conditions, implements a centralized approach based on small variations, and has flexible detector positioning-and-number requirements.

PART III presents the design of a simulation environment, named GETRAM, that solves the difficulties in testing CARS that were discussed at the end of the previous part. We have seen that these difficulties are inherent in the traffic engineer having to use diverse models in order to analyze a traffic network, and that there is the need for a system to solve it. GETRAM provides a unified framework integrating various types of traffic models and tools for traffic analysis, sharing a DataBase, a graphical editor and a module for results presentation. The traffic network can be partitioned into views, hierarchically organized polygons in the real world, so that, for example, a simulation model applies only to one of these restricted areas. Network states produced by one model can be used as a starting point by another model. In order to ease the task of integrating a new model or analysis tool, a library of object-based high-level functions provide a view-aware access to the DataBase, maintaining consistency. Included in this Ph.D. thesis are the design of the whole environment — DataBase, GETRAM API, and graphical editor — and the development of the DataBase and GETRAM API.

PART IV starts from the demand-responsive traffic control system developed in PART II, CARS V1, and improves it in various respects. First, in order to ease the task of testing the system, it is integrated into the traffic modeling and analysis environment described in PART III. Second, certain parts that directly influence to the effectiveness of the system, such as control timing, adaptive control logics and communication with the controllers, are revised or totally redone. A suite of tests has been applied to the resulting system, CARS V2, in the four scenarios described in PART II. Finally, to further testing the system and taking advantage of the fact of having real-world data available, it is compared against a vehicle-actuated control at an isolated junction.

PART I
Introduction

This part introduces the concept of demand-responsiveness in traffic control, providing a historical perspective from traditional fixed-control systems to current modern demand-responsive systems. Finally, the part concludes with a study of areas where research and improvement is needed. All of these areas will be considered in the following parts, and some solutions will be proposed.

Demand-responsive traffic control systems change signal timings in response to traffic flow variations measured or estimated in terms of traffic volume and queue length. The effectiveness of these third-generation control systems over traditional first and second-generation systems depends on the accuracy of the modeling of the vehicles' state and behaviour, prediction models, the method of finding the optimum policy, and the frequency with which a change in policy is considered.

1. Historical perspective: previous generations

The key element in urban traffic systems to date has been the system that links traffic signal timings at the different intersections. The development of computer-aided traffic signal control systems has evolved through a number of generations. A classification that has become quite common was fixed by UTCS. It distinguishes between three generations, although some authors (Yagar 1991) include an additional generation between the first and second generations, as described below.

1.1 1st generation

The first-generation software uses prestored timing plans computed off-line and based on previously measured traffic data. The system can store a number of timing plans per sub-area. A sub-area is a group of local controllers operating in the same mode and switching simultaneously from one timing plan to another. Timing plan selection of a desired set of timing parameters (cycle, split, offset) may be based on time of day, automatic response to traffic conditions, or operator choice. The mode of timing plan selection is determined by the operator. Four modes of area control are available:

1. Standby. Controllers are transferred from computer to local backup system control.
2. Time-of-Day. Timing plans are automatically selected on a time-of-day and day-of-week basis. Change to the selected plan is accommodated with a time resolution of 15 minutes for each sub-area in the system. This mode may be selected by the operator on a system basis (all sub-areas), or on an individual sub-area basis.
3. Traffic-Responsive. This mode automatically matches the timing plan best suited for existing traffic conditions. Very recent volume and occupancy data are compared with volume and occupancy characteristics of the various stored patterns. Each history for a sub-area of controllers is unique to a set of control parameters (controller timing plans). The control plan related to the best data-history match is selected from the available timing plans.
4. Manual. This mode is used primarily for handling abnormal traffic conditions not provided for when operating under the Time-of-Day and Traffic-Responsive modes. The manual mode may be used during checkout of traffic-responsive control patterns or to provide rapid response to traffic conditions resulting from accidents, concerts, or sport events.

In order to deal with oversaturation, the Critical Intersection Control is introduced. Critical intersections which saturate frequently may have their split adjusted to be directly proportional to the traffic demand for the green time computed for their associated links. The cycle splits for these intersections are then changed on a once-per-cycle basis as a function of the values of queues and volume on these links. This critical intersection control is used to finely tune the split allocated to each approach to the intersection, based upon fluctuations in local traffic demand. Thus, intersection control works in conjunction with, and supplements, area control operation.

1.2 1.5th generation

In 1.5th-generation systems, a library of timing plans is calculated/updated automatically, based on detected traffic volumes, using an off-line model. The system selects the one it thinks is best suited to traffic conditions at the time, usually based on current measurements on volume and occupancy. These plans offer considerable flexibility; for example, the Los Angeles Automatic Traffic Surveillance and control (ATSAC) system (Rowe 1988) supplements its 1.5th generation plans with critical intersection control to adjust phase times on a cycle-by-cycle basis.

1.3 2nd generation

The second-generation control strategy is a real-time, on-line system that computes and implements signal plans based on surveillance data and predicted changes. The system retains the selection of prestored timing plans through the manual or time-of-day modes, and the hardware failure detection functions of the first-generation UTCS software and hardware. The optimization process is repeated at 5 minute intervals. New timing plans cannot be implemented more often than every 10 minutes to avoid developing control parameters based on conditions which exist during signal timing transition.

In general, 2nd-generation traffic signal systems have not been as effective as anticipated. Some of the reasons for this are the transition between two control plans, which can introduce traffic disturbances and negative effects, and the reliance on predictions. In some cases, the predictions were heavily weighted to measurements made during the most recent signal cycle. This did not work very well (Yagar 1991) because traffic demand is quite variable from cycle to cycle. In other cases, the predictions used detector data that were evened out over, for example, a 15-minute period, hindering the responsiveness of the system.

2. 3rd generation

Third-generation UTCS systems implement and evaluate a fully responsive, on-line traffic control system. Like the second generation, it computes control plans to minimize a network-wide objective, using for input predicted traffic conditions. However, the period after which timing plans are revised is shorter (less than 5 minutes), and cycle length of each controller is permitted to vary from cycle to cycle. The rest of this historical perspective will focus on these third-generation systems.

Development of third-generation, demand-responsive traffic control systems began in the early 1960's. The first systems failed mainly due to the use of off-line concepts and inadequate predictions. SCOOT (Hunt et al. 1982) represents the first clear success, soon followed by SCATS (Lowrie 1982). Both systems are designed to operate on a network following a centralized approach, where the controller associated with each junction sends detected measures to and receives control changes from a central computer. Local processing at each controller generally reduces, as in SCOOT, to data filtering and translation between a control phase and its corresponding traffic light state. SCATS complements these functions with vehicle actuation which adapts the current split plan, i.e., a phase can be terminated earlier or skipped if there seem to be no vehicles waiting. PRODYN (Henry 1983) was initially developed as a centralized system for urban networks, but decentralized versions followed and are discussed later in this section. UTOPIA (Donati et al. 1984) is a centralized system that follows an interesting approach by decomposing the tasks between a central computer and local processing at each junction; it will be discussed further in following sections.

While these systems provide better performance than traditional fixed-control systems on traffic conditions ranging from light to moderate, their effectiveness is reported to decrease when approaching congestion (Robertson 1987). This is due to their vehicle modeling, detector-positioning requirements, and control logic. A more detailed discussion is available elsewhere (Grau and Barceló 1992c). The CARS system (Barceló, Grau, Egea and Benedito 1991) (Grau and Barceló 1992a,b,c) has been designed to overcome these shortcomings. It provides a more accurate, congestion-oriented vehicle modeling, improved control logics and flexible positioning-and-number detector requirements. CARS's centralized approach helps to coordinate junctions and provides sustained effectiveness in a wide range of traffic conditions.

In contrast to these network systems, other demand-responsive systems have been specially aimed at isolated intersection control, such as TOL (Bang 1976), OPAC (Gartner et al. 1983), MOVA (Vincent and Young 1986), SAST (Lin et al. 1988). This restricted approach determines control logic — usually binary choice — and vehicle modeling.

Decentralized systems differ from isolated-junction systems in that they consider the junction control in a network context. Therefore, coordination between adjacent junctions is addressed. OPAC considers some degree of coordination with the downstream intersections, and can thus be considered as a primitive decentralized system; it is discussed further in this section. PUGWA (Schlabbach 1988) varies control parameters in oscillations of up to ±15% from a fixed-control plan. OFSET (List and Pond 1989) can follow two strategies, either a look-up table of fixed-control plan approach, or a set of decision rules. All these systems will be discussed further in following sections.

2.1 Systems for isolated intersections

In contrast to centralized and decentralized systems, which focus on all or part of the network, some demand-responsive systems have been specially aimed at isolated intersection control. The term 'vehicle-actuated control', used almost solely in isolated intersections, infers some form of control strategy which is directly dependent on measured vehicle activity. In contrast, the term 'traffic-actuated control' is used when data of all traffic is available to calculate the signal phase on-line.

Vehicle-actuated control was prior to any other form of responsive control; the special features of actuated controllers — at least the first ones — can be 'built into' the hardware, without requiring microprocessor-based controllers. They are generally classified into three types: semiactuated, actuated, and volume-density. Modern controllers are simply referred to as 'actuated' controllers, with settings to determine the level of responsiveness. There are two widespread types of actuated control hardware: NEMA and Type 170/179 controllers. NEMA specifications cover both hardware and functionality.

The underlying premise in semiactuated control is that there is a 'main street' that should have the green as much as possible, and a 'side street' that should be given only enough green to service the relatively low and somewhat unpredictable demand that occurs. A side-street detector (point or area) is used to identify the arrival of a vehicle. For semiactuated operation, it is essential to set a minimum green for the main street and an initial interval, a vehicle interval, and a maximum interval for the side street.

In full actuated mode, all phases and approaches have detectors and operate in the actuated mode. The underlying concept is that the competing demands are equally important, and that there is no structured arrival pattern on any approach to take advantage of. The timing principles are esentially the same as for the semiactuated, but the maximum green time does not begin timing until there is a call on the conflicting phase.

The volume-density controller is an advanced form of actuated controller which keeps track of the number of arrivals, and reduces the allowable gap according to certain rules. Also common is a variable initial interval based upon actuation on the phase during its nongreen period.

Often, vehicle-actuated control is not considered as a third-generation system. This is because of its simple logic and also because it is activated by the vehicles without considering the global traffic in the intersection. More recent control systems for isolated intersections include traffic-actuated features. They require microprocessors and implement more sofisticated, global strategies. Examples of traffic-actuated control include TOL (Bang 1976), SAST (Lin et al. 1987) and MOVA (Vincent and Young 1986). Their basic approach features binary-based control logic, taken from Miller's (1963) proposed algorithm for adjusting signal timings in small time intervals based on the trade-off between extending the current green duration and terminating it immediately.

MOVA (Vincent and Young 1986). The MOVA (Microprocessor Optimized Vehicle Activation) methodology was originally developed in England to provide traffic control strategies for isolated intersections. It has led to control strategies that are substantial improvements over the conventional vehicle actuation signal systems used throughout England. MOVA is structured to optimize the green timings for the critical traffic movements on a phase-by-phase basis to reduce delays and stops. It can accommodate both saturated conditions and traffic flow at considerably less than full saturation rates. MOVA is based on a microprocessor that includes four primary software programs: Input Routine, Strategy Programs, Output Routine, and Archive Program. Implementation of a completely new control strategy requires revising the Strategy Program only. All programs are written in Fortran. MOVA detectors, installed in each approaching lane at 40 and 100 meters from the stop line, are connected directly into the MOVA computer unit containing the four primary programs. The local signal controller is fitted with a standard Urban Traffic Control (UTC) interface that allows the MOVA computer to force the traffic signal to change phases. To address the needs of isolated intersection control, the algorithms have included only phase split and cycle length adjustment. In order to expand the MOVA into a network-coordinated signal system, an additional algorithm would need to be developed and evaluated to address the offset.

Algorithms posterior to Miller's enhance the binary choice approach by using a more complex objective function, as in **TOL** (Bang 1976), or by adding several levels in the decision-making process, as in **SAST** (Lin et al. 1987). The first three levels are based on simple decision rules and the last level requires signal optimization. The information needed to reach a timing decision in each step includes queue lengths at the beginning of a step and the expected numbers of vehicle arrivals at the stop lines in each of several steps in the future (Lin et al. 1987). Parallel to this augment in decision complexity, the interval of time discretization has been lowered from 10 secs in Miller's method to 2 secs in SAST, as sufficient computing power has become available.

We have been testing several variations of the binary choice strategy applied to isolated intersections (e.g. as used in MOVA and TOL), and we have identified the following problems in all of them:

- A tendency to produce frequent terminations of green and, in general, unstable policies.
- Difficulties in coordinating adjacent intersections. In fact, to the best of our knowledge, binary choice has only been used in isolated junctions. Augmenting the time horizon along which control changes are tested does not seem to improve results.
- The need to plan in each time interval. This restricts real-time flexibility and the length of the time interval as the size of the network grows.

2.2 Centralized systems

Under a centralized approach, a single instantiation of the control strategy operates on all signals. Local processing at each controller is generally limited to data filtering and translation of control phase commands to corresponding traffic light states. Nevertheless, 'central' systems may feature some hierarchical decomposition, with local controllers having a limited degree of autonomy. For example, SCATS (described below) features vehicle actuation which adapts the current split plan; i.e., a phase can be terminated earlier or skipped if no vehicles are waiting.

Examples of centralized systems include SCOOT, SCATS, PRODYN, ACTS, and UTOPIA.

2.2.1 SCOOT

SCOOT (Hunt et al. 1981), which stands for 'Split, Cycle and Offset Optimization Technique', is aimed at the centralized control of urban networks. In SCOOT, the user partitions the network in sub-areas sharing a common cycle, in order to maintain coordination between signals. The cycle of each sub-area can vary in increments of a few seconds at intervals of not less than 2.5 minutes, to attempt to allow the most heavily loaded junction to operate at a maximum degree of saturation of 90 per cent. The cycle time of a sub-area may be changed between single and double cycle operation if this is expected to produce a net saving in delay.

Aa well as this, a kind of binary choice approach is used a few seconds before each stage change, to estimate whether it is better to make the change earlier or later, always with the restriction of a common cycle within the sub-area. An offset optimizer operates on each junction at each cycle, where the cyclic flow profile information obtained from the entrance detector is used to estimate whether or not an alteration to the offset will improve the traffic progression on those sections which are immediately upstream or downstream of the junction.

To cope with congestion, SCOOT employs congestion offsets that are different from those available for routine minimization of delays and stops in a link. The congestion offset has to be set to maximize capacity and avoid link blockage when the upstream intersection is green to the critical approach. These pre-specified congestion offsets are automatically implemented when queues cover the upstream detector.

In addition, 'gating' and 'action at a distance' facilities are available. These facilities allow restriction of the inflow traffic into a sensitive area upstream of a bottleneck to prevent the buildup of a long queue or congestion. Vehicles are redistributed to more acceptable roads. In addition, 'gating' can be set up so as to increase green to 'gated' links downstream of the bottleneck and clear the queues. This 'flow gating' procedure is currently applied manually. However, expert systems may be used for this purpose in the longer term.

Moreover, feedback of signal status into SCOOT for demand-dependent stages is available. Simulation studies have suggested that such feedback can lead to delay reduction of 5-10% for a typical intersection, as a result of better distribution of the available green times. Higher reductions are obtained when the intersection is oversaturated and when the frequency of the demand dependent stage differs substantially from the one initially assumed.

Additionally, in a SCOOT system a link can use the congestion information from another link: a link can be specified as 'supplier of congestion information' to another link, so that the traffic signals at an intersection can be affected by the blocking back ocurring in an upstream link.

Furthermore, SCOOT can model the queue buildup when SCOOT is overriden by an emergency plan (e.g., a fire engine). Thus, the optimizer can dissipate the queue once it resumes control.

Finally, a particular node can be favored by using a 'weighting' facility: the stops and delays of the route to be favored can be multiplied by large weighting factors in the objective function, so as to avoid large values for stops and delays on this route at the optimal signal settings. This 'weighting' procedure — or the above mentioned 'gating' facility — is currently being applied manually. However, expert systems may be used for this purpose in the longer term.

The effectiveness of SCOOT has been assessed by major trials in five cities in the UK and in three North American cities (Oxnard, Red Deer and Toronto). In most cases, comparison was made against a good standard of fixed time coordination. SCOOT achieved an average delay saving of about 12%. The tests also suggested that SCOOT is more effective when traffic demand is unpredictable and approaches an intersection's capacity. To date, it has been implemented in the urban traffic control systems of over 40 cities in the UK and other countries.

In the absence of field evaluation directly comparing SCOOT with other demand-responsive systems, SCOOT can be evaluated on the basis of its reported difficulties in near-saturation conditions, and from the following first-principles examination of the method:

- Although it is a centralized system, the simulation model and optimizers do not take advantage of the knowledge of the network state. Except for the congestion offsets, the model does not consider the state upstream of the entrance detector to each link. This seriously restricts its utilization both in very light and in near-congested traffic conditions, particularly in very short links (Robertson 1987).
- In light traffic conditions, the use of a slowly varying cycle can restrict the responsiveness of the system when fast green changes may be needed to accommodate scarce and unexpected platoons.
- In fast-changing conditions (e.g., sudden transition from light to dense conditions), the use of a common cycle may slow the responsiveness of the system. More importantly however, since the system can not 'see' the traffic conditions upstream of the entrance detector in each link, vehicle arrival predictors (cyclic flow profiles) cannot adapt to fast-varying arrivals.
- In near-saturation conditions, spill-back avoidance requires sudden changes in offset, split and cycle at the affected junction. This seems to be incompatible with the use of common cycle sub-areas.

2.2.2 SCATS

SCATS (Lowrie 1982), which stands for 'Sydney Co-ordinated Adaptive Traffic System', is a centralized system for urban networks developed in Australia. Similarly to SCOOT, SCATS divides an area into smaller sub-areas of several intersections sharing a common cycle time. Each sub-area contains one critical intersection, for which the sub-area's green split plans, internal and external offset plans, and cycle lengths are selected. In contrast to SCOOT, detectors in SCATS are located at the stoplines and perform the dual functions of providing traffic flow data — vehicles exiting the link — and local data for vehicle actuation. The user has to specify four 'background' green split plans for each junction, five internal offset plans which determine the offsets within the sub-area, and five external offset plans for linking adjacent sub-areas.

The sub-area cycle length is a function of the highest degree of saturation measured in the sub-area during the previous cycle. The change in cycle length is restricted to ±6 secs, but occasionally can double or triple the cycle length. As the author points out, "The four green split plans for each intersection specify the proportion of the cycle to be allocated to each phase usually as a percentage of cycle length./.../The split plan also specifies the normal sequence of phases, which may vary between plans, and a number of options for each phase which control the transfer of unused time (as a result of vehicle-actuated operation) between phases and ensure that nominated phases are included in or excluded from the sequence as a function of cycle length. Phases which may not be terminated by local vehicle actuation are nominated in the split plans./../ Once per cycle, a 'split plan vote' is calculated. Two votes for the same plan in any three consecutive cycles result in the selection of that plan." Local processing in the controller of each junction allows vehicle actuation which adapts the current split plan, i.e., a phase can be terminated earlier or skipped if there seem to be no vehicles waiting.

Referring to the internal offsets, the author says: "Once per cycle, an 'offset plan vote' is calculated. Four votes for the same plan in any five consecutive cycles result in the selection of that plan." As for the external offsets, "each cycle, a link vote is calculated for each sub-area which has possible linkages defined./.../ An option exists to force an immediate linkage between two sub-areas if the measured flow /.../ exceeds a preset value./.../ When a linkage occurs, the nominal cycle length of the two subsystems is set to the higher of the two cycles previously operating..."

Results from evaluations in Sydney, Australia indicated that it was of similar performance to TRANSYT in journey time, but was 9% better in stops. SCATS has been implemented in the urban traffic control systems of several cities worldwide. The algorithm has demonstrated its ability to coordinate large arterial and grid networks, and is currently being evaluated in Oakland County, Michigan.

A first-principles examination of the method suggests that:

- The criteria used for most plan changes rely on the degree of saturation function, which relates the number of vehicles exiting a link to the available green time. Queue length or space free measurements are ignored. This allows some vehicle-actuated control, but restricts system operation in medium to congested conditions, where queues play an increasingly important role.
- Specification of the split and offset plans seems to require an extensive knowledge of the traffic conditions in the network. As the author notes: "Typically, two of the offset plans define offsets which are optimum for highly directional flow patterns such as those experienced during the morning and evening peak periods."
- Although the vehicle-actuated operation can give an advantage in certain conditions, it is not of much help in coordinating closely located intersections, especially near congestion.
- The use of relatively flexible pieces of fixed-control plans may restrict the 'responsiveness' of the system, classifying the current traffic conditions between the operator-specified plans.
- The detector location at stoplines rather than at link entrances enables some vehicle-actuated tactics, but hampers the monitoring of platoon progression and this results in an inherent imprecision at both the offset and split plan selection.

2.2.3 ACTS

ACTS, which stands for 'Adaptive Control of Traffic Signals', is a centralized system for urban networks that was first developed in 1983 (Gartner, Kaltenbach et al. 1983) and subsequently refined and modified (Kaltenbach et al. 1986). Like OPAC, the system follows the open-loop-feedback methodology and a rolling time horizon to consider changes in the signal plan. In fact, as Kaltenbach notes, the use of a rolling horizon had already been suggested by Van Zijverden and Kwakernaak (1968). However, what first distinguishes ACTS from the previously reviewed systems is the use of a centralized network simulation model, although a relatively simple one. The vehicles are modeled as moving at constant speed and accumulate vertically at the end of each link. The links are divided into equally spaced blocks, and vehicles move from one block to the next in each time interval. The model also accounts for the free space in the destination links, although measured as the total link space minus the space that the vehicles occupy, that is, as if the vehicles where all grouped from the end of that link.

Also innovative, the 1983 version includes the use of planning in a subnetwork around each junction. This fixed-size subnetwork will be discussed in following sections. The 1986 implementation focuses on smaller networks, and no subnetwork planning is mentioned. To our knowledge, neither of these implementations has been tested in real conditions. The detectors are intended to be located in the upstream part of a link, although the network modeling should permit greater flexibility than the previously described systems.

The 1983 control logic approach proposes a Branch&Bound procedure to identify optimal admissible control sequences at the network nodes over the planning horizon. Although the method is said to assure a global optimum, it has high computing needs. This led to the 1986 version, which uses the traditional fixed-time concepts of cycle, offset, and splits. Each time interval, usually 5 secs, the optimization procedure tests small changes alternatively in the splits and in the offsets of each junction, along a fixed time horizon. A common cycle length is updated if it differs significantly from a moving average of cycle durations recently implemented.

We have implemented the ACTS network traffic model as described in the 1983 and 1986 reports, extended it for complex geometries, and experimented with the 1986 control strategies. Our experience shows that:

- Although ACTS is a network model, the vehicles are still moved at constant speed and accumulated vertically at the end of the link. Because of this, the measure of free space contains an intrinsic inaccuracy. This produces poor results in near-congested conditions, when the importance of queues and accurate free space grows. The use of vertical queues hinders the determination of progression timing in arterials.
- The use of the traditional concepts of cycle, offset, and splits complicates the adaptive logic. For example, it is not clear if a common cycle should be used, when to update it, or if alternating the split and offset changes is more suitable than using both at the same time.

2.2.4 PRODYN-H

PRODYN (Henry 1983) was initially developed as a centralized system for urban networks. In the author's words: "PRODYN uses the decomposition coordination techniques to convert the large initial optimization problem into several smaller problems which are solved by dynamic programming, and then solves the global problem using a two-level iterative calculation structure. The decomposition is spatial; each smaller problem is assigned to an intersection." The mathematical model represents the vehicles as moving at a constant speed and accumulating vertically at the end of the links. The model considers a restricted form for free space, and the queue in a link is required to be less than a constant value, supposedly the link total vehicular storage. Time is discretized in intervals of generally 5 secs. PRODYN resulting policies are acyclic, a common cycle length not being required. The system needs a supervisor node to coordinate the junctions, and therefore is centralized. Each intersection can solve its smaller problem in local processing, the supervisor providing for the upstream arrivals and control policy testing. Changes in control policy are tested along a time horizon.

Decentralized approaches were tested later (Barriere et al. 1986) and they are discussed hereafter, in the section devoted to decentralized systems.

We have not explicitly implemented the PRODYN-like approach, but have worked on similar traffic models and, in our experience, constant speed and vertical queues cannot be expected to deal with near-congested conditions, nor with coordinating arterials when queues are long and greater accuracy in free space is needed to avoid spill-backs.

2.2.5 UTOPIA

UTOPIA (Donati et al. 1984) is a centralized system for urban networks where a central computer governs control policies calculated at each local controller — junction. Although it is a centralized system, it introduces some interesting decentralized strategies.

Each local controller contains a microcomputer which is able to collect data from detectors, process them, and exchange information with adjacent local controllers and with the central computer. A detector is needed on each approach lane to a junction, and is connected to the appropiate local controller.

UTOPIA was designed to give absolute priority to selected public transport lines at every intersection, except when conflicts or interference between these vehicles could occur. Public vehicle detectors are installed for identifying priority vehicles and display succesive arrival times in waiting stations.

Urban traffic is described according to three dynamic models:

- The global model considers the network as constituted by storage units. The model relates the number of vehicles present in the i-th time interval to the number of vehicles which during the same time interval can move to the successive storage, depending on the average travel speed and the saturation flow. Results translate into reference rules for the local controllers.
- The local model describes every intersection with traffic lights as a set of links. Its state is given by the vector of arrivals at the intersection itself of the vehicles already in the

link, assuming constant speed and vertical queues. The state in the next time interval depends on the state at the adjacent junctions. Changes in control policy are tested over a few minutes (typically, 2 or 3 minutes) according to decisions given as reference by the global model and to the priority of certain public vehicles.

- The public traffic model is deterministic as regards the vehicle generation and the routes of each trip. Travel time is stochastic and depends on the free travel time, waiting time at station, and lost time at intersections.

UTOPIA strategy integrates public and private transport management and, in general, its task partition where a central computer makes reference plans to the local controllers seems more advantagous than PRODYN-H in congested conditions. However, the authors do not give details about the actual features of these plans.

Nevertheless, the system requires more communication line installation than any other approach: between detectors and local controllers, local controllers and the central computer, and between adjacent controllers. Public transport identification requires additional detectors.

In the global model, there seems to be no provision for predicting future traffic states; reference rules from the local controllers are generated according to the current state. In the local model, vehicles are assumed to move at constant speed and accumulate in vertical queues; this probably hampers the effectiveness of coordinating adjacent intersections or operations in congested conditions.

2.3 Decentralized systems

Decentralized systems consider individual control in a network context. Therefore, some coordination between adjacent junctions is addressed. Examples include OPAC, PRODYN, PUGWA and OFSET.

2.3.1 OPAC

OPAC (Gartner 1982), which stands for 'Optimization Policies for Autoadaptive Control', was developed initially for isolated junctions with simple geometry, and uses an open-loop-feedback strategy. It can be considered as a primitive decentralized system because a small degree of coordination with downstream intersections has been addressed. Each time interval (usually 5 secs), the timing plan is examined along a time horizon, the best timing being choosen among all the possibilities along the time horizon without requiring a fixed cycle time. The evaluation is made according to a simple traffic model which moves the vehicles at a constant speed and accumulates them in a vertical queue at the end of the link. OPAC uses the concept of the 'rolling horizon', where although the best timing has been evaluated along a time horizon, only a small part of it (the 'head') is implemented, until the next time interval in which the timing plan is reconsidered; see Figure 2.1. The time length of the head corresponds to the distance from the link entrance detector to the stopline. Therefore, OPAC requires the detector to be located, at least, at 5 secs travel distance from the stopline (then r = 1). The 'Rolling Horizon Method', as implemented in OPAC, presents a close relationship between timing and control logic, summarized as follows:

0 - Determine the length of the time horizon, k, and the rolling period, r.
1 - Obtain the flow for the next r intervals ('head') from the detectors.
Predict the flow for the next k - r intervals ('tail').
2 - Calculate the optimum control policy along the time horizon.
3 - Implement the control only during the rolling period ('head').
4 - Move the time horizon r units to obtain the new horizon.
Repeat steps 1 to 4.

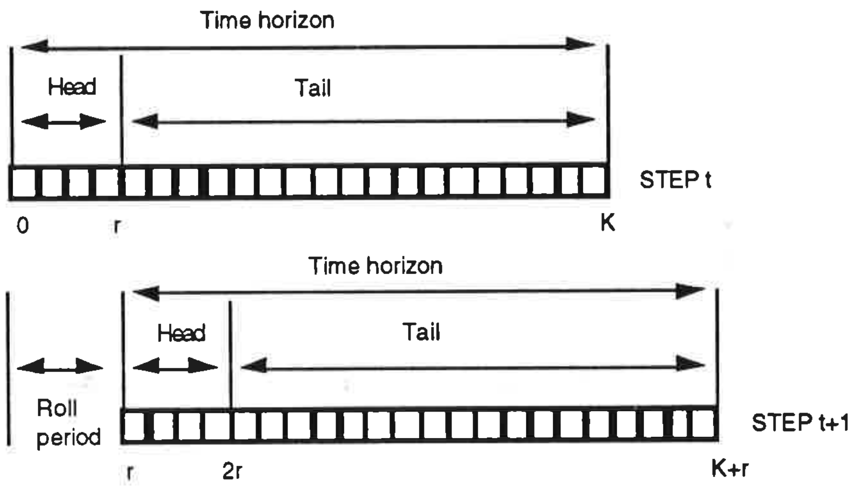


Fig. 2.1 Rolling horizon as implemented in OPAC.

A recent improvement considers a special treatment for congestion conditions: the optimization algorithm may be overridden by special signal plans in the presence of congestion, as defined by a user-specified occupancy threshold. OPAC uses a special 'congestion override' facility, since the optimization is intended for operation under normal (undersaturated) conditions. During periods of congestion, measures of effectiveness such as queue length might be preferable to stops and delay, used as optimization criteria under undersaturated conditions. Moreover, during periods of congestion, queues may become extremely long, extending over the upstream detectors. As a result, traffic demand will not be accurately measured by OPAC. If the occupancy of the detectors associated with one of the major phases exceeds the threshold, the switching decision for that phase is ignored and the phase is allowed to time to its maximum. When the occupancies for that phase fall below the threshold, the switching decisions from the optimization algorithm are implemented.

The OPAC-RT Version 2.0 system is composed of three major subsystems: the Data Acquisition and Control Subsystem (Eight Phase Dual Ring Digital Controller Unit, Conflict Monitor, etc.). The detectors are located well upstream of the stopline on all approaches to an intersection. Additional detectors are placed at the confluences of driveways with the links. The optimization is intended for operation under normal (undersaturated) conditions. To date, research on OPAC has concentrated on isolated intersection control and little has been done towards implementing the method in a network context.

Some tests have been conducted on OPAC by the FHWA. Overall delays were reduced in the range of 4% to 16%, while the percentage of vehicle stops was mixed. The main limitations of OPAC are: the system to date cannot coordinate sets of intersections; it does not address congestion problems (saturated conditions); it is not compatible with standard controllers; and detectors are further upstream than current US practice. OPAC has not yet been implemented in the field and further research and modification are needed to produce an operational version.

Our experience in implementing and testing by simulation the 1982 version of OPAC shows that, because of vehicle modeling restrictions and decentralized policy, coordination problems arise when it is used in arteries or networks, especially in near-congested conditions. Our conclusions from experimental results are that:

- When applied to more than one junction, OPAC acts as a decentralized system but a certain degree of coordination with the downstream junctions can be achieved if their fixed-control policy is known and the traffic conditions remain light.
- The simple traffic model, which supposes vehicles traveling at constant speed and accumulating at the link end, is reported to have problems in moderate to heavy congestion when queues are long. The model does not account for the free space in the destination links. This can become more important when applying OPAC to networks.
- The approach used in OPAC, i.e., testing every possible control policy, is only feasible with a very simple (fast) model, whose assumptions quickly become unrealistic when approaching saturation conditions.
- In general, a decentralized approach in networks is reported to be ill-suited when approaching congested conditions, especially when the congestion is already well established (OECD 1981).

2.3.2 PRODYN-D

As said previously, PRODYN (Henry 1983) was initially developed as a centralized system for urban networks, but decentralized approaches, PRODYN-D1 and PRODYN-D2, were tested later (Barriere et al. 1986). The optimization procedure is executed, for each intersection, through an adapted forward dynamic programming algorithm looking for optimum control on a rolling horizon.

PRODYN-D1 proposes a decentralized control where each junction does not require information from adjacent junctions. Link entrance detectors provide measures that update the arrival predictors. Although a lot of care seems to have been taken in the arrival predictors, the authors acknowledge the difficulties in coordinating arterials. Consequently, PRODYN-D2, another decentralized system, requires an exchange of information between adjacent junctions: upstream junctions provide for the vehicle arrivals. Both decentralized systems model vehicles as moving at constant speed and forming vertical queues. Free space in the link exits is not considered. PRODYN-D2 is reported to achieve some coordination in arterials, but the authors do not specify under which traffic conditions. A third version, PRODYN-D2 with pseudo-horizontal queues, is reported to have been developed as a 'saturation model'. However, it cannot compute coordination parameters. In any event, the authors do not provide information on this version.

PRODYN has been shown slightly less efficient than fixed time plans regarding total presence time when perturbation occurs and, particularly, when congestion progresses upstream from downstream intersections (Henry 1989). However, a new procedure is planned to be incorporated into PRODYN. This procedure defines an upper coordination level controlling the algorithm under oversaturated conditions, when the short range link delay criterion is not sufficient to ensure good network control.

Several on-site tests have shown that PRODYN is more efficient than fixed time with a significant decrease of over 10% in delay. However, the limitations of this algorithm are: high-capacity communication links are needed; the system is not fault-tolerant; and complex computations are necessary. PRODYN has been applied in only one city (Niort) to date. A commercial version is under development by a European consortium in the DRIVE program.

2.3.3 PUGWA

Several demand-responsive traffic control systems have been developed in Germany during recent years. However, documentation about their operation internals is not readily available or has not been translated.

PUGWA (Schlabbach 1988), which stands for 'Pulsating Green Wave', is a control method which varies split, cycle time and offset by a modified gap/density control within a coordinated signal system network. In the author's words, "In a two-stage system the sum of green time could be divided into a fixed, unalterable part for each of the two approaches and a varying part — the latter assigned by traffic-actuation, whether by seconds to the main and secondary road (respectively, left-turning and oncoming traffic) or as a whole to a congested approach or lane. In this way the cycle time is varying although the mean value in the macroscopic view is fixed." PUGWA can vary cycle time in oscillations up to ±15 per cent relative to the average value, and green times at the approach to each junction can be modified within these limits according to a vehicle-actuated policy. Thus, the system is based on a fixed-control plan which has to be previously specified for the whole network under study. A microprocessor at each junction provides computing support, receiving vehicle measures from loop detectors located 30 m before the stopline. In this context, the system is decentralized.

Responsiveness of the system is restricted to 15% variations from a fixed-control plan. Tests of the PUGWA system have shown an improvement over a totally fixed control. However, we believe that its conservative approach limits its operation in a wide range of traffic conditions. As the system is based simply on incremental adjustments to a fixed-control plan, its performance is restricted by the suitability of the underlying fixed plan. This restriction may be alleviated by operator intervention, but this in itself raises staffing and control problems. Moreover, local variations at each intersection do not consider coordination with adjacent intersections, although this limitation may be mitigated by the restriction on variations from the fixed plan.

2.3.4 OFSET

OFSET (List and Pond 1989) is a decentralized system where the controller of each junction shares information with the adjacent controllers without a central computer. The shared communication consists of approach volumes and signal parameters — cycle length and offset. A cycle is established where signals periodically send data to one another. Each junction interprets this as a "signature pattern" in the subnetwork constituted by itself and the adjacent junctions. The authors describe two alternative demand-responsive strategies to operate at each junction: the first one is based upon a look-up table of fixed-control policies and the second uses a set of decision rules.

In the look-up table approach, the subnetwork signature pattern is used by each controller to identify a matching pre-specified operating mode — cycle length and offset. Splits are set by the demand-actuated software inherent to the controller. The interval between change points has to be carefully assigned in order to allow responsiveness to volume fluctuations, yet long enough to ensure that transitions are completed before new ones begin.

The second approach uses a set of decision rules based on the traditional concepts of cycle, offset and splits. Each signal computes a desired cycle length based on its own approach volumes. This is compared to the maximum desired cycle length among all contiguous junctions, which is obtained from the signature pattern. If the ratio between the two is near one, the maximum is selected. If the ration is nearer to two, half the maximum is selected. The offset can vary in incremental, small adjustments between pre-specified values, either in absolute terms or relative to the offsets on adjacent signals. The routine searches for a value that minimizes total intersection delay, where input volumes are estimated from a moving average over a past set of cycles. As for the splits, green time is shared proportionally, based on approach volumes.

OFSET includes some interesting concepts, but is hampered by certain operational problems. In the look-up table approach, the system behaves very much as a fixed-control switch, the transition time between plans limiting its responsiveness. Vehicle actuation features acting on the green time duration do not consider coordination between adjacent intersections. Fixed-control plans have to be pre-specified for the network, this raising periodic maintenance issues. The other approach, a set of decision rules in each intersection, relies on the traditional concepts of cycle, offset and splits. Although the method has not been fully explored, it seems that the strategy at each junction is that of accomplishing a common cycle with other intersections in the network. This alone would limit the responsiveness of the whole system. Moreover, its split adjustment, which is based on approach volumes, requires accumulated measurements over a certain period of time, further limiting responsiveness.

3. First experiences

Having reviewed the literature on demand-responsive systems, we implemented the 1982 code of OPAC (Gartner 1982) and began experimenting with it, as stated in **2.3.1**. Our experiences corroborated the results given by Gartner for an individual junction (evaluated by OPAC itself), but it was obvious that extension to a network of OPAC-controlled junctions would require extensive modification of the code. Furthermore, even in the case of fixed-control downstream junctions, coordination simply could not work if queues downstream were large, since OPAC assumes that vehicles travelling from junction to junction always take the same time.

In order to overcome some of these restrictions, we moved to an ACTS-based model (Kaltenbach et al. 1986), discussed in **2.2.3**. In ACTS, links between nodes are partitioned into an integral number of segments of length $V * \partial$ called blocks; V corresponds to an average speed of vehicles in free motion over a link, and ∂ is a time discretization step size. In this model, vehicles move in free flow until the end of the link and then may disperse into a number of destination links, or queue vertically. A <u>queue discharge</u> model is added in order to limit the flow of vehicles in the first seconds after the signal light turns to green. To model the vehicles' movement within the junction, we <u>added</u> the concept of the '<u>bridge link</u>' that exists within the junction between an origin and its destination link, and a queue. Also, to account for some variability in the speed of the vehicles, we apply a degree of dispersion in the movement of vehicles from block to block. We tested this model by microscopic simulation at some isolated junctions, applying a <u>binary choice</u> as an autoadaptive strategy. Figures 3.1, 3.3, and 3.4 show the geometry of the three junctions where the model was applied, and Figures 3.2, 3.4, and 3.6 their respective modeling in links and bridge links. As regards turning percentages, at the first junction vehicles present equal probability of turning or continuing straight on; at the second junction, turning ratios are 1:5 for the horizontal entrance, and 1:4 for the vertical. Detectors at each entrance are placed 50 m from the street entrance. Vehicles are assumed to have an average length of 4.8 m and to move at 23 m/s in free flow. All junctions have two phases, the first one for the horizontal approach(es) and the second for the vertical(s).

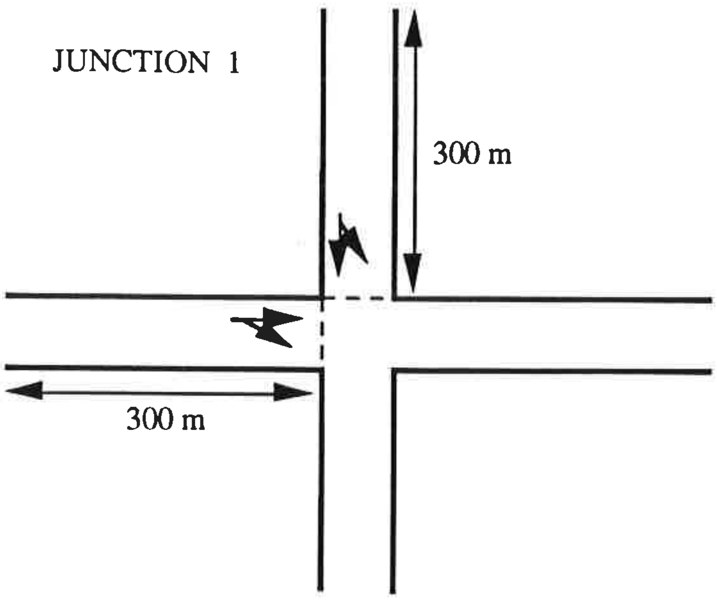


Fig. 3.1 Geometry of the first junction used in these first tests.

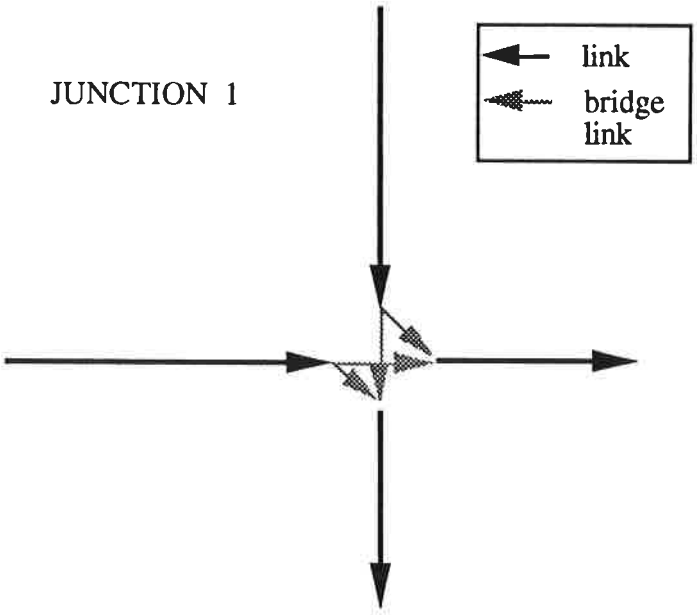


Fig. 3.2 Modeling in links of the first junction (Figure 3.1).

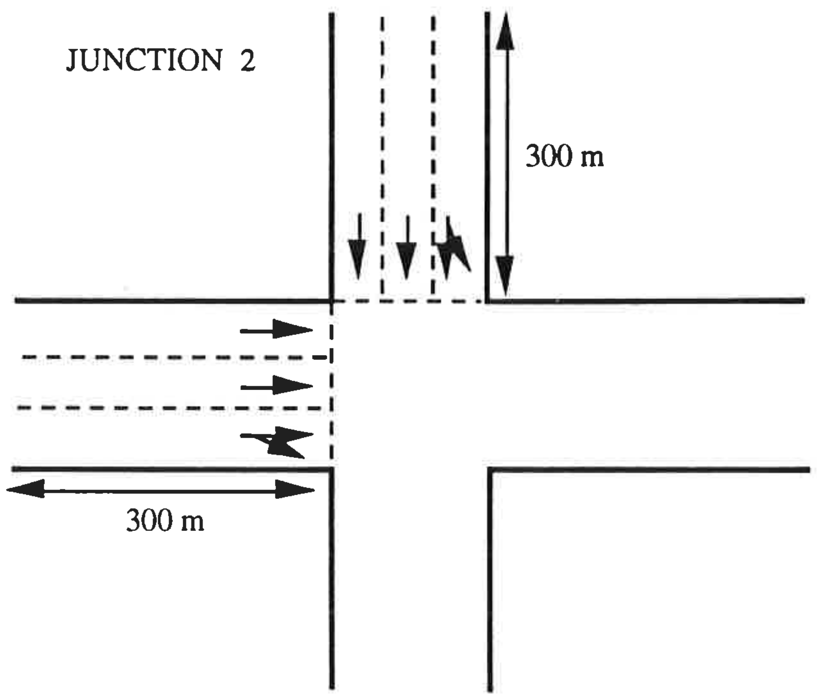


Fig. 3.3 Geometry of the second junction used in these first tests.

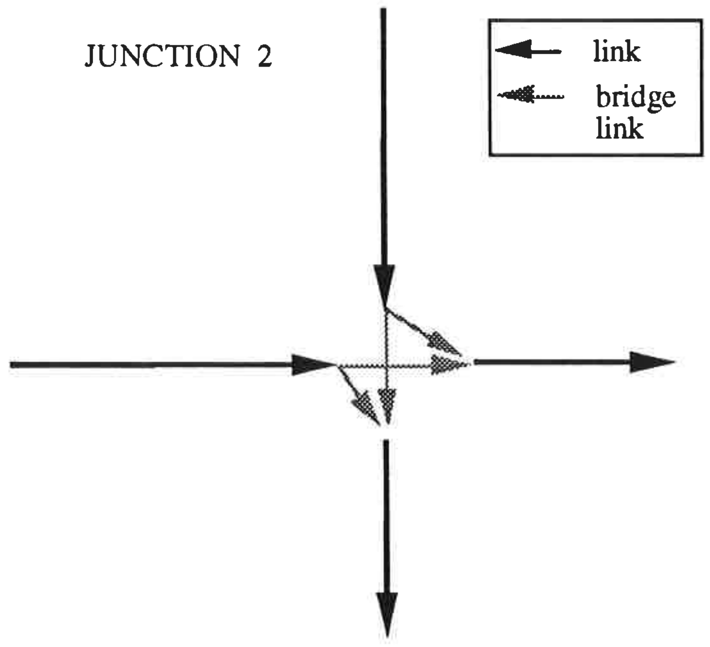


Fig. 3.4 Modeling in links of the second junction (Figure 3.3).

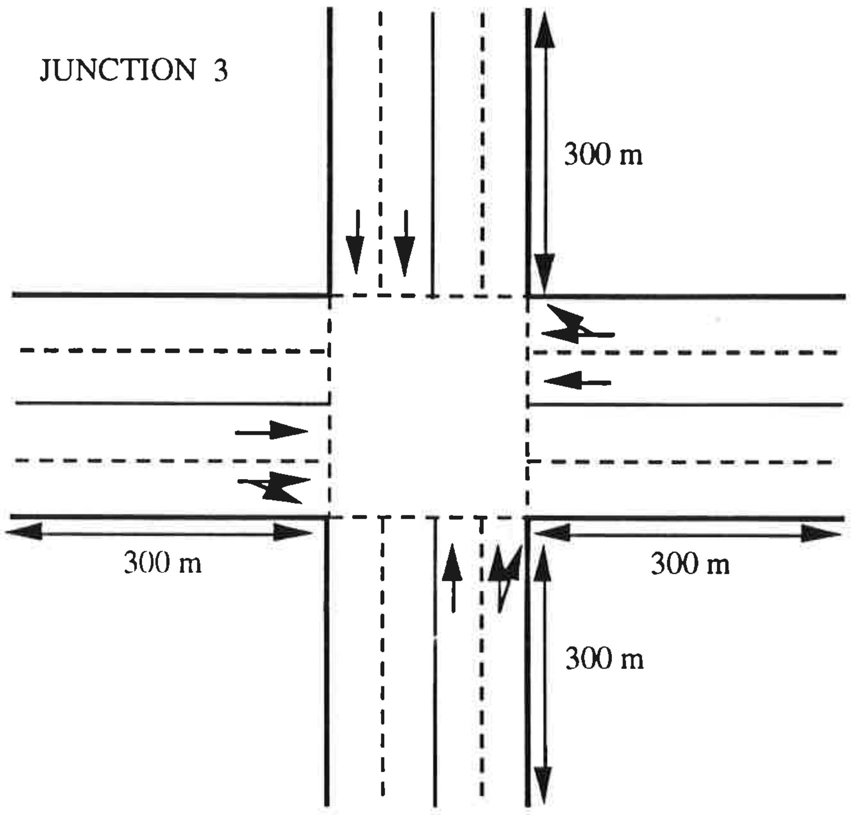


Fig. 3.5 Geometry of the third junction used in these first tests.

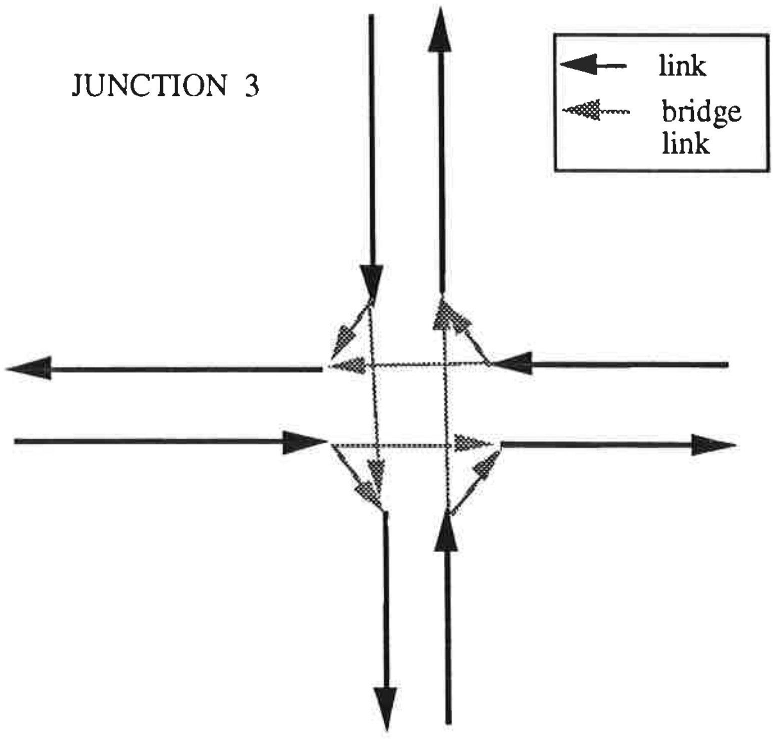


Fig. 3.6 Modeling in links of the third junction (Figure 3.5).

We tested these three junctions with the AIMSUN (Barceló, Ferrer and Montero 1989) microscopic simulator, which had been previously validated. First, the following CARS parameters had to be adjusted, supposing an entrance flow of 500 vehicles per hour and lane, and $\partial = 2$ seconds.

- Free-flow speed in each approach: In ACTS, vehicles are either moving at free-flow speed or stopped, so free-flow speed has to be less than in reality. A value of maxspeed / 1.1 has been found satisfactory for all three junctions.
- Platoon dispersion: More platoon dispersion reduces the autoadaptiveness of the control strategy, averaging the flows of vehicles. For the first junction, satisfactory values are 0.1 forward dispersion and 0.1 backward dispersion. For the second and third junctions, 0.1 forwards and 0.2 backwards are satisfactory.
- Queue discharge: This specifies the proportion of the maximum exiting flow that is allowed to exit in the time intervals after the traffic light turns to green. For the first junction, we take the following values: 0.01, 0.03, 0.05, 0.1, 0.1, 0.2, 0.3, 0.4, 0.6, and 0.7. For the second and third junctions the values are: 0.1, 0.2, 0.4, 0.6, and 0.8.

Varying the entrance flows from 500 veh/h*lane to 800 veh/h*lane needs to adjust the free-flow speed to maxspeed / 1.2, and if 200 veh/h*lane then maxspeed / 1.0. Queue discharge has also to be adjusted. Therefore, this suggests that all three parameters depend, at least, on the number of lanes and the entrance flows.

Once the parameters have been adjusted, we make a first attempt to compare the effectiveness of the autoadaptive control against a fixed control timed according to Webster and Cobbe (1966). Supposing a constant entrance flow of 500 veh/h*lane, and a saturation flow of 2,500 veh/h, fixed-control timings are: 10 secs of green time (plus 2 secs amber time) both for the first phase and also for the second phase. Simulating 30 minutes produces the results shown in Figure 3.7. Autoadaptive control gives better results in all three cases, particularly in the first junction — total delay decreases from 9:17 to 3:27. This is due to the fact that, when there is only one lane per approach, wider gaps between vehicles are produced.

	JUNCTION 1		JUNCTION 2		JUNCTION 3	
	fixed control	auto-adapt.	fixed control	auto-adapt.	fixed control	auto-adapt.
Total delay	9 : 17	3 : 27	11 : 41	10 : 53	16 : 50	15 : 36
t stopped / t total	0.1342	0.0764	0.0656	0.0584	0.0862	0.0821
t travel / (veh * km)	1 : 29	1 : 09	1 : 03	1 : 00	1 : 08	1 : 07
t delay / (veh * km)	0 : 29	0 : 12	0 : 08	0 : 06	0 : 11	0 : 10
t stopped / veh	0 : 05	0 : 02	0 : 03	0 : 02	0 : 03	0 : 02
Average speed	44.02	56.15	59.12	61.79	54.90	56.2
nb.stops / (veh * km)	1.6	0.5	0.7	0.5	0.8	0.7

Fig. 3.7 Results obtained by microscopic simulation comparing a fixed-control and a demand-responsive strategy. Entrance flows are constant.

In a second set of tests, instead of constant flows, entrance flows are supposed to vary as indicated in Figure 3.8, maintaining, in average, the previous entrance flow. This assumption is intended to add more realism to the experiment. Results, in Figure 3.9, show that the advantage of autoadaptive over fixed control is still greater in this case — a 30% improvement. In fact, Webster's method of calculating signal timings asumes constant flows — which can prove to be an unrealistic assumption in many cases. In contrast, autoadaptive control is more flexible, being effective in a wider range of traffic conditions.

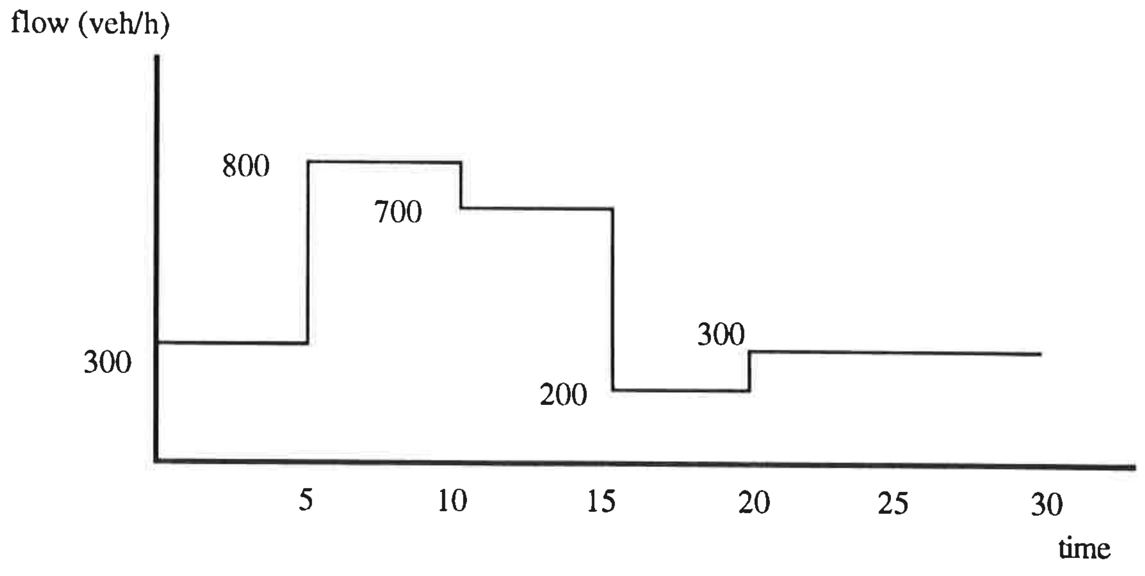


Fig. 3.8 Entrance flow per lane in the second experiment.

	JUNCTION 1		JUNCTION 2		JUNCTION 3	
	fixed control	auto-adapt.	fixed control	auto-adapt.	fixed control	auto-adapt.
Total delay	10 : 23	3 : 18	10 : 50	7 : 34	18 : 52	13 : 27
t stopped / t total	0.1452	0.0718	0.0725	0.0528	0.0914	0.0755
t travel / (veh * km)	1 : 31	1 : 11	1 : 04	0 : 91	1 : 09	1 : 05
t delay / (veh * km)	0 : 29	0 : 15	0 : 08	0 : 06	0 : 11	0 : 09
t stopped / veh	0 : 06	0 : 02	0 : 03	0 : 02	0 : 04	0 : 02
Average speed	42.76	54.43	58.20	61.11	53.20	56.4
nb.stops / (veh * km)	1.7	0.6	0.7	0.5	0.9	0.7

Fig. 3.9 Results obtained by microscopic simulation comparing a fixed-control and a demand-responsive strategy. Entrance flows are variable.

The third set of tests is similar to the previous one, but this time the autoadaptive control has been adjusted for 300 veh/h*lane. As Figure 3.10 shows, results are still advantageous for the autoadaptive control, but the advantage is less than with the right parameter adjustment.

	JUNCTION 1		JUNCTION 2		JUNCTION 3	
	fixed control	auto-adapt.	fixed control	auto-adapt.	fixed control	auto-adapt.
Total delay	10 : 23	5 : 20	10 : 50	9 : 58	18 : 52	16 : 03
t stopped / t total	0.1452	0.8903	0.0725	0.0697	0.0914	0.0853
t travel / (veh * km)	1 : 31	1 : 19	1 : 04	1 : 01	1 : 09	1 : 07
t delay / (veh * km)	0 : 29	0 : 21	0 : 08	0 : 07	0 : 11	0 : 10
t stopped / veh	0 : 06	0 : 04	0 : 03	0 : 03	0 : 04	0 : 03
Average speed	42.76	50.17	58.20	60.05	53.20	54.4
nb.stops / (veh * km)	1.7	0.9	0.7	0.7	0.9	0.8

Fig. 3.10 Similar to Figure 3.9 but with the autoadaptive control adjusted for lower flows.

The fourth set of tests consists of a series of experiments, each one increasing the entrance flow further from what was used to calculate Webster's fixed-control timings, that suppose an entrance flow of 300 veh/h, giving 7 secs (plus 2 secs amber time) for each phase. We make four tests for each type of control and junction, entrance flows varying from 300 veh/h in the first one to 500, 700, and 1,000 in the others. The results, shown in Figure 3.11, show that the advantage of autoadaptive control fades as flow augments.

		JUNCTION 1		JUNCTION 2		JUNCTION 3	
		fixed control	auto-adapt.	fixed control	auto-adapt.	fixed control	auto-adapt.
ENTRANCE FLOWS (veh/h)	**300**	2 : 26	0 : 48	4 : 11	3 : 04	8 : 18	6 : 23
	500	4: 33	2 : 01	5 : 25	4 : 01	8 : 10	7 : 05
	700	5 : 23	4 : 55	5 : 29	5: 05	14 : 57	14 : 23
	1,000	8 : 49	6 : 22	6 : 57	6 : 01	17 : 32	17 : 09

Fig. 3.10 Results of varying entrance flows. Obtained by microscopic simulation.

Applying our autoadaptive control to a simple network, such as the four-junction arterial shown in Figures 3.11 and 3.12, points to other restrictions of our model. We calculate two fixed-control settings, the first one corresponding to light traffic conditions, the second to heavy conditions — this experiment is repeated in PART II — this being one of the four networks that are tested in that part. Comparing these fixed-control timings with our autoadaptive control produces the results in Figure 3.13. We see that, in light traffic conditions, autoadaptive control produces better results, but the advantage (5%) is less than in the single-junction case. In heavy traffic conditions, our autoadaptive control turns out to be disadvantageous: either the traffic model with vertical queues and constant speed or the binary-choice control, or both of them, should somehow be changed.

- A binary-choice control method produces acyclic signal timings, i.e., does not require a common cycle length. Traditionally, networks have been operated by using a common cycle length, and the first 3rd-generation control systems — SCOOT, SCATS — still followed this strategy. Newer systems, such as PRODYN and OPAC are acyclic, and this is considered as a promising trend, since removing the common-cycle-length restriction can, potentially, help to produce a more autoadaptive control. We adhere to this trend and in subsequent parts of this document, other acyclic control networks will be proposed.
- With regard to the traffic model, these first experiences suggest that our ACTS-based model has to be adjusted for the geometry and average traffic conditions in each

network. Should traffic conditions vary significantly from the average then performance decreases and the advantage of installing a demand-responsive control system becomes dubious. Also importantly, although our ACTS-based model can be applied to networks, coordination between junctions is hard to achieve as traffic conditions approach congestion and queues grow. In the next part, we will propose a totally different traffic model that, as the tests show, solves all these problems.

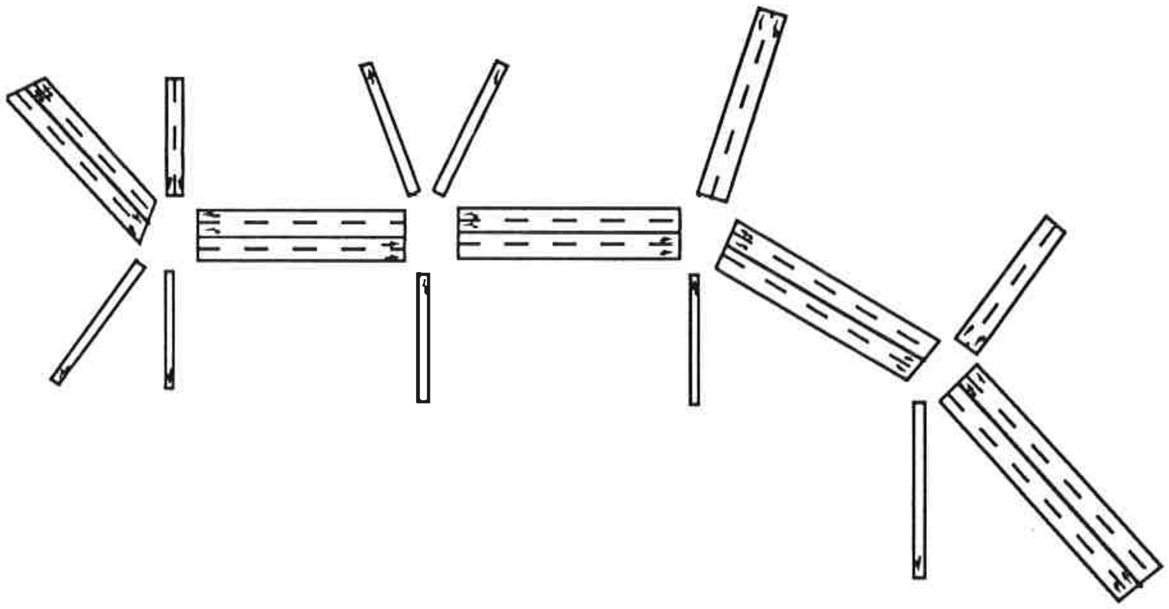


Fig. 3.11 Geometry of the arterial used in these first tests.

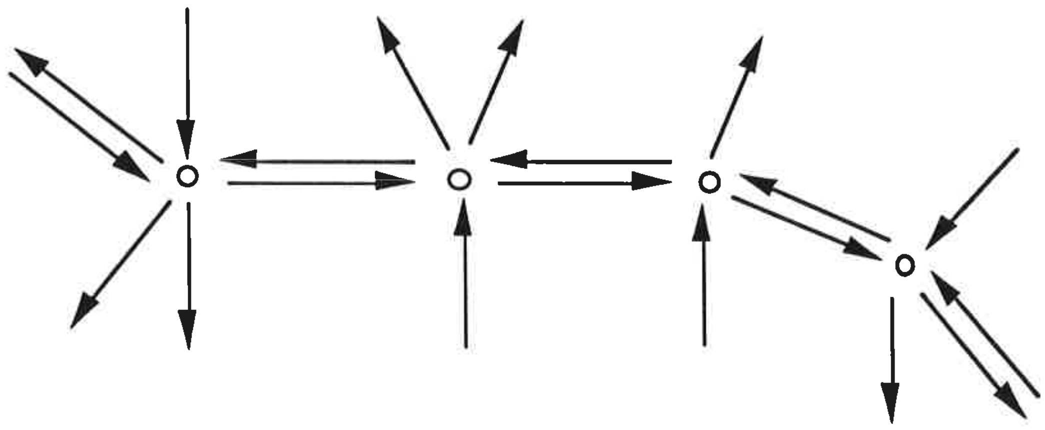


Fig. 3.12 Modeling in links of the arterial in Figure 3.11.
Bridge links are not shown.

ARTERIAL	LIGHT CONDITIONS		HEAVY CONDITIONS	
	fixed control	auto-adapt.	fixed control	auto-adapt.
t stopped / t total	0.4022	0.3806	0.4501	0.4665
t delay / (veh * km)	3 : 40	3 : 06	4 : 25	4 : 40

Fig. 3.13 Some results on the arterial in Figure 3.11.

4. Areas where research and improvement are needed

To date, 3rd-generation traffic control systems have rarely been applied, and mostly at isolated junctions or in very small networks. This unduly limited success may stem partly from political problems, institutional friction, investment in present systems, technical inertia or lack of expertise of the personnel that have to install and apply them, and economic issues. Related to economic considerations is the fact that it is sometimes hard to justify the cost, in installation and personnel training, of a demand-responsive system, when the reliability of such a system in some traffic conditions is not assured or, at least, is no better than the already installed traditional fixed-control system. From the above historical perspective, we have identified the following areas where research and development is clearly needed. All of them will be considered in our proposed systems, CARS V1 and V2.

- More flexibility in detection requirements. Current systems accept few types of detectors and controllers, usually being limited to a single type. Still more important, they require the detectors to be positioned at fixed locations. For these reasons, it is usually impossible to make good use of existing investments in detection, and installation costs rise.
- More effectiveness in near-congestion traffic conditions. When approaching congestion, most demand-responsive systems encounter serious difficulties and have to be replaced by a more traditional strategy. This requires better modeling of traffic behaviour in those conditions (long queues, spill-backs and so on), and finding new adaptive strategies capable of coping with the sudden changes that are required to deal effectively with such queues. Traditional 3rd-generation systems, such as SCOOT and SCATS, use a common cycle and this hinders these sudden changes.
- Improving the ease of use of the software. Most systems were designed long ago, and they lack user-friendliness by today's standards. Network geometry and control parameters have to be specified in terms of extensive data, system-dependent, and esoteric concepts. In order to specify them, the user is usually faced with no more than an ASCII file editor. This raises training costs, and increases the risk of faulty data.

5. References

Bang, K.L. (1976). "Optimal control of isolated traffic signals.", *Traffic Eng. Control*, 17(7), pp. 288-292.

Barriere, J.F., J.L. Farges, J.J. Henry (1986). "Decentralization versus hierarchy in optimal traffic control.", *Control in Transportation Systems, 5th IFAC/IFIP/IFORS Conference*, Vienna, 8-11, 1986, pp. 321-326.

Donati, F., V. Mauro, G. Roncolini and M.Vallauri (1984). "A hierarchical-decentralized traffic light control system. The first realization: Progetto Torino.", *IFAC 9th World Congress*, Vol. II, 11G/A-1.

Gartner, N.H. (1982). "Demand-responsive decentralized urban traffic control: Part I: Single-intersection policies.", Office of University Research, US Department of Transportation, Rept. DOT-RSPA-DPB-50-81-24.

Gartner, N.H., M.H. Kaltenbach et al.(1983). "Demand-responsive decentralized urban traffic control: Part II: Network extensions.", Office of University Research, US Department of Transportation, Rept. DOT-OST-P-34-85-009, pp. 53-97.

Henry, J.J, J.L. Farges and J. Tuffal (1983). "The PRODYN real time traffic algorithm.", *Control in Transportation Systems, 4th IFAC/IFIP/IFORS Conference*, Baden-Baden, April 20-22, 1983, pp. 307-312.

Henry, J.J (1989). "PRODYN tests and future experiments on ZELT.", *VNIS'89: Vehicle Navigation and Information Systems, IEEE Conference*, Toronto, September, 1989.

Hunt, P.B., D.I. Robertson, R.D. Bretherton and R.I. Winston (1981). "SCOOT — A traffic-responsive method of co-ordinating signals.", *TRRL Report* LR1014, Transport and Road Research Laboratory, Crowthorne, 1981.

Hunt, P.B. et al. (1982). "The Scoot on-line traffic signal optimization technique.", *Traffic Eng. Control*, 23(4), pp. 190-192.

Kaltenbach, M., J.P. Dussault, and G. Marquis (1986). "Adaptive control of traffic signals.", Université de Montréal, Centre de Recherche sur les Transports, Publication #469.

Kessaci, A., J. Farges and J. Henry (1989). "On-line estimation of turning movements and saturation flows in PRODYN.", *6th IFAC/IFIP/IFORS Conference on Control in Transportation Systems*, Paris, 1989.

Kessaci, A., J. Farges and J. Henry (1991). "Turning movement ratios estimation using inductive loop detectors and information from route guidance systems.", Recherche Transports Sécurité, No. 6, Feb. 1991, pp. 39-44.

Lin, F.B, N. Wang, and S. Vijayakumar (1987). "Development of an intelligent adaptive signal control logic.", *Proc. of the Eng. Foundation Conf.*, June 1987, pp. 257-279.

Lin, F.B. and S. Vijayakumar (1988). "Adaptive signal control at isolated intersections.", *ASCE Journal of Transport Engineering*, Vol. 114, No 5, pp. 555-573.

List, G. and J. Pond (1989). "A strategy for real-time control of demand-sensitive signal networks.", *AATT Conference*, February 1989, pp. 367-372.

Lovrie, P.R. (1982). "The Sydney co-ordinated adaptive traffic system — principles, methodology, algorithms.", *Proc. IEE Int. Conf. Road Traffic Signalling*, 1982, pp. 67-70.

Miller, A.J. (1963). "A computer control system for traffic network." *Proc., 2nd Int. Symp. on Theory of Road Traffic Flow*, London, U.K., pp. 201-220.

OECD (1981). "Traffic control in saturated conditions.", *Road Research*, Organization for Economic Co-operation and Development, Jan. 1981.

Robertson, D.I. (1974). "Cyclic flow profiles.", *Traffic Eng. Control*, June 1974, 15(14), pp. 640-641.

Robertson, G.D. (1987). "Handling congestion with SCOOT.", *Traffic Eng. Control*, April 1987, pp. 228-230.

Rowe, E. (1988). "The Los Angeles ATSAC system.", *Management and Control of Urban Traffic Systems*. US Engineering Foundation Press, New York, pp. 9-17.

Schlabbach, K. (1988). "Pulsating green waves — the Dramstadt experiment.", *Traffic Eng. Control*, July/August 1988, pp. 392-397.

Vincent, R.A., and C.P. Young. (1986). "Self-optimizing traffic signal control using microprocessors — the TRRL 'MOVA' strategy for isolated intersections.", *Traffic Eng. Control*, 27(7/8), pp. 385-387.

Webster, F.V. and B.M. Cobbe (1966). "Traffic signals.", *Road Research Tech. Paper*, No. 56, London: Her Majesty's Stationery Office, pp. 55-60.

Yagar, S. (1991). "Traffic Control Systems: Trends.", *Concise Encyclopedia of Traffic & Transportation Systems*, Editor Markos Papageorgiou, Pergamon Press, pp. 536-541.

PART II

Towards a new demand-responsive control system: CARS V1

This part presents a demand-responsive traffic control system, CARS V1, intended to operate in signalized urban areas: networks, arterials, and isolated intersections. A graphical user interface, X-Windows and DecWindows compliant, allows the user to specify network characteristics in a friendly and intuitive manner, without the need to be acquainted with the actual modeling. The system features an underlying simulation system and a prediction model based on real-time measured conditions, implements a centralized approach based on small variations, and has flexible detector positioning-and-number requirements.

1. A new approach

In the previous part we reviewed third-generation traffic control systems, implemented an OPAC-like and an ACTS-based system, and experimented with them. These two systems represent two current trends, the first one for isolated junctions and the second for networks. Our experiences show certain difficulties that are intrinsic to the simple traffic modeling employed in these systems, and in order to solve them, here we propose a new approach to third-generation demand-responsive traffic control. Our proposed system, CARS (Barceló, Grau, Egea and Benedito 1991), which stands for "Control Autoadaptativo para Redes Semaforizadas", is a third-generation demand-responsive traffic control system for networks, arterials and isolated intersections that relates to the other systems in that:

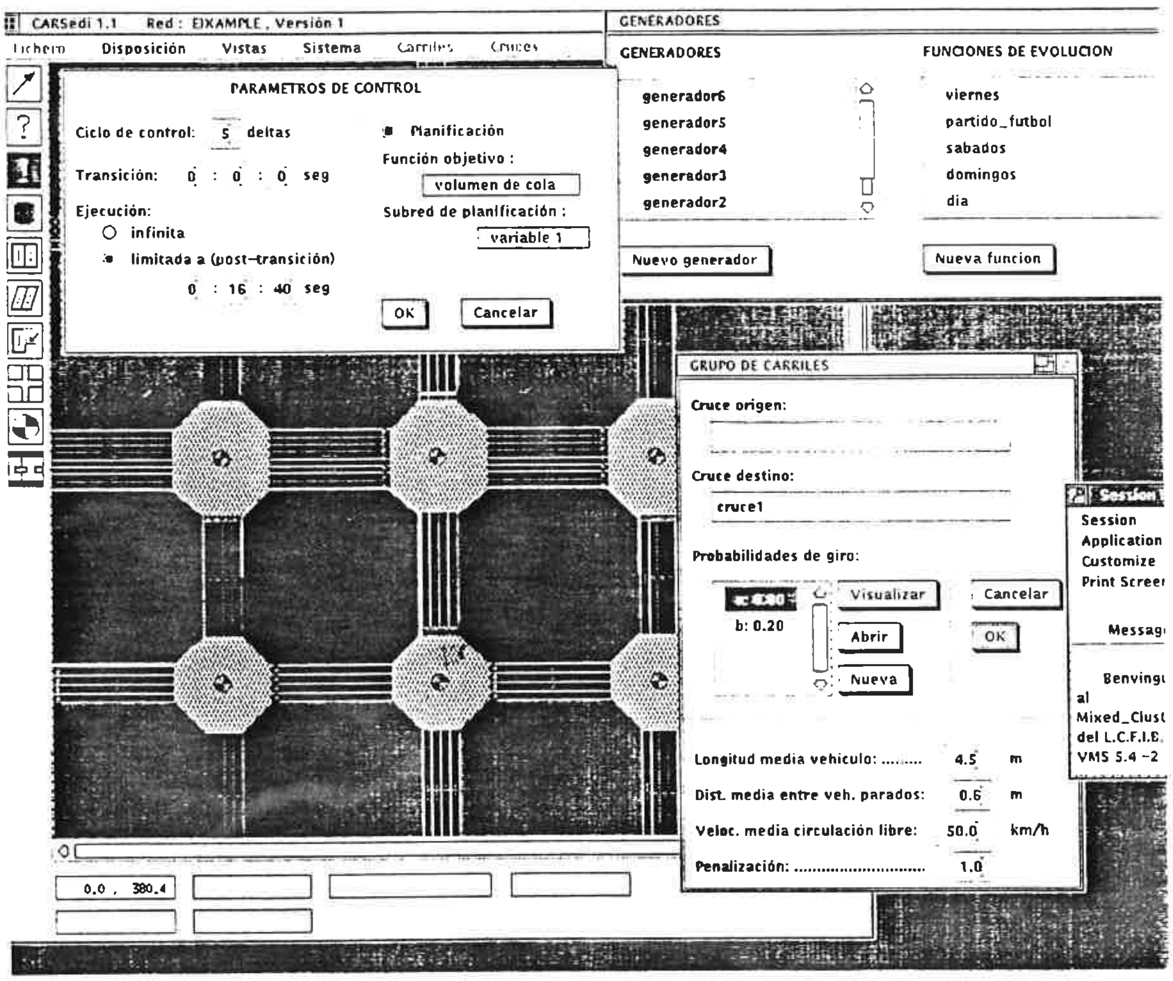
- It shares with OPAC, ACTS, and PRODYN the idea of a rolling horizon to test the policy changes.
- It uses the ACTS and SCOOT approach to an adaptive centralized control based on small variations.
- As in ACTS, a network simulation model is used to evaluate the effectiveness of the control changes.
- It maintains a cyclic flow profile, similar to SCOOT's, to help to predic vehicle arrivals and free space in the internal links.

CARS is intended to be used in densely signalized urban areas and follows a centralized, congestion-oriented approach. It incorporates the following new elements:

- An underlying simulation model, PACKSIM (Grau and Barceló 1992), that deals with packets of vehicles moving according to an adhoc 'packet-following' model and stopping in horizontal queues. Forbidden zones in the intersections as well as mergings and give-ways are carefully modeled. Special emphasis has been put on queue modeling in order to deal with congested situations, accurately evaluate free space, and find a progressive signalization in the arterials.
- A network state forecast is performed before considering changes in control policy, and the system does not need to plan the control policy in each time interval.
- The autoadaptive algorithm tests control changes in variable-sized subnetworks. The size of the subnetworks depends on the traffic conditions, and it is efficiently reduced by the prediction method.
- It accepts an arbitrary number and positioning of detectors. Because this affects the performance of the system, certain rules are recommended in order to assure the effectiveness of the demand-responsive control.

The system includes a graphical user interface named CARSedi, X-Windows and DecWindows compliant that allows the user to specify network characteristics in a friendly and intuitive way; see Figure 1.1. The links are automatically generated, so there is no need for the user to know the actual modeling. Finally, the whole system is coded in ANSI C, thus being highly portable.

For reasons that will become clear later in this part, CARS is restricted to urban signalized areas, preferably with a high density of signal lights and short links (a maximum link length of 400 m is recommended). Ramp-meterings are not supported.



PARAMETROS DE CONTROL

Ciclo de control: 5 deltas

Transición: 0 : 0 : 0 seg

Ejecución:
○ infinita
● limitada a (post–transición)
0 : 16 : 40 seg

● Planificación

Función objetivo : volumen de cola

Subred de planificación : variable 1

GENERADORES

GENERADORES: generador6, generador5, generador4, generador3, generador2

FUNCIONES DE EVOLUCION: viernes, partido_futbol, sabados, domingos, dia

GRUPO DE CARRILES

Cruce origen:

Cruce destino: cruce1

Probabilidades de giro: a: 0.80, b: 0.20

Longitud media vehiculo: .... 4.5 m

Dist. media entre veh. parados: 0.6 m

Veloc. media circulación libre: 50.0 km/h

Penalización: 1.0

0.0 , 380.4

Benvingut al Mixed_Clust del L.C.F.I.E. VMS 5.4 -2

Fig. 1.1 The CARSedi user interface.

2. Network description

Following the trend toward friendly user interfaces, CARSedi allows the description of the network by using real-based objects such as lanes, junctions, detectors and controllers in a graphical multi-windowed environment. The user is faced with a realistic object-oriented network representation where the mouse is used to select, create, examine, and modify objects without the need to remember commands or identifiers. Eight types of objects are distinguished:

- Background objects. Used to represent blocks, squares and so on. They are optional, and are not used by the CARS control system, but serve to clarify network representation, providing visual clues to position the other objects, as shown in Figure 2.1.
- Sections. Groups of contiguous lanes in the same direction, with rectangular or polygonal geometry. On exiting a section, the vehicles either leave the network or turn into another sections. The turnings take place within a junction area, and the origin lanes and destination section in the turning must be specified, as Figures 2.2 and 2.3 show.
- Junctions. Bidimensional nodes whose area is defined by the space between the sections entering and exiting the junction. Three types of junctions are distinguished according to the control policy applied: unsignalized, fixed signalized and demand-responsive signalized. Unsignalized junctions are modeled as having only one stage, in which all turnings have right of way. A list of priorities can be specified when give-ways between turnings arise. In the signalized junctions, a stage-based approach is applied, in which the cycle of the junction is divided into stages, each of which can have a set of turnings with right of way. We distinguish two stage types: phases and interphase periods. Give-ways are specified as lists of priorities between turnings. See Figures 2.4, 2.5, 2.6, and 2.7.
- Progression arterials. Used for demand-responsive control, a progression arterial is a set of contiguous junctions along which a progressive green wave is desired; see Figure 2.8. While they can be used to modulate control, progression arterials restrict the demand-responsiveness of the overall system and can therefore produce drawbacks if badly defined.
- Controllers. Each controller is associated with a junction to which it sends the phase endings, and with a set of detectors from which it periodically receives traffic measures; see Figures 2.9 and 2.10. The controllers can be adapted to match the real-world connections with the control system, and the same with the detectors connected with each controller.
- Detectors. These can be installed on any position and number of lanes in a section; see Figures 2.11 and 2.12. An arbitrary number of detectors are allowed for each section. The detectors must be connected to a controller to which traffic measures are periodically sent. CARS assumes that the detectors are capable of measuring both the number of vehicles and the average speed in each time interval. CARS has been designed to be able to work with a wide variety of traffic detector technologies, ranging from the conventional magnetic loop detection to video-detection.
- Generators. Used to replace missing detected data while testing the system without being connected either to a real-network or a microscopic simulator. Each detector can be associated with a generator, which will be used when testing the system. The vehicles are

generated according to arrivals (uniform, Poisson, or others) and speed distribution (uniform, normal, etc.). See Figure 2.13.

• Evolution functions. Associated with the generators, and provide periodic volume variations. For example, a 24-hour evolution function can vary the parameters used to generate vehicles on a daily basis; see Figure 2.14.

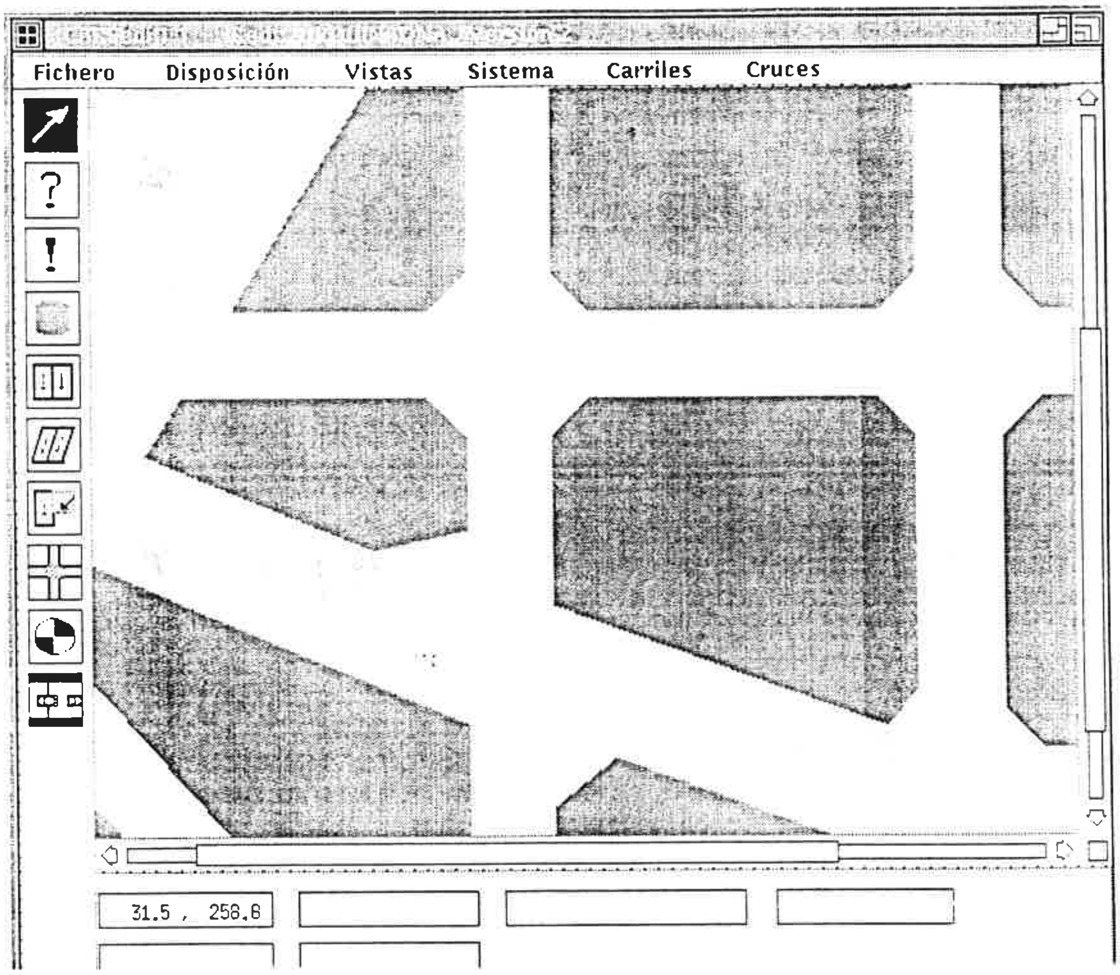


Fig. 2.1 Blocks displayed in CARSedi before adding sections and other network objects.

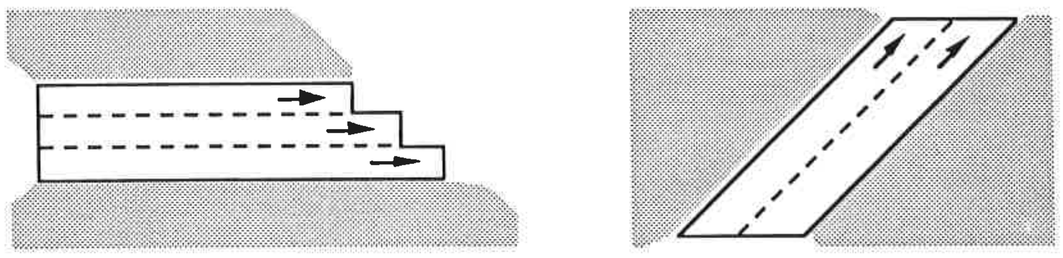
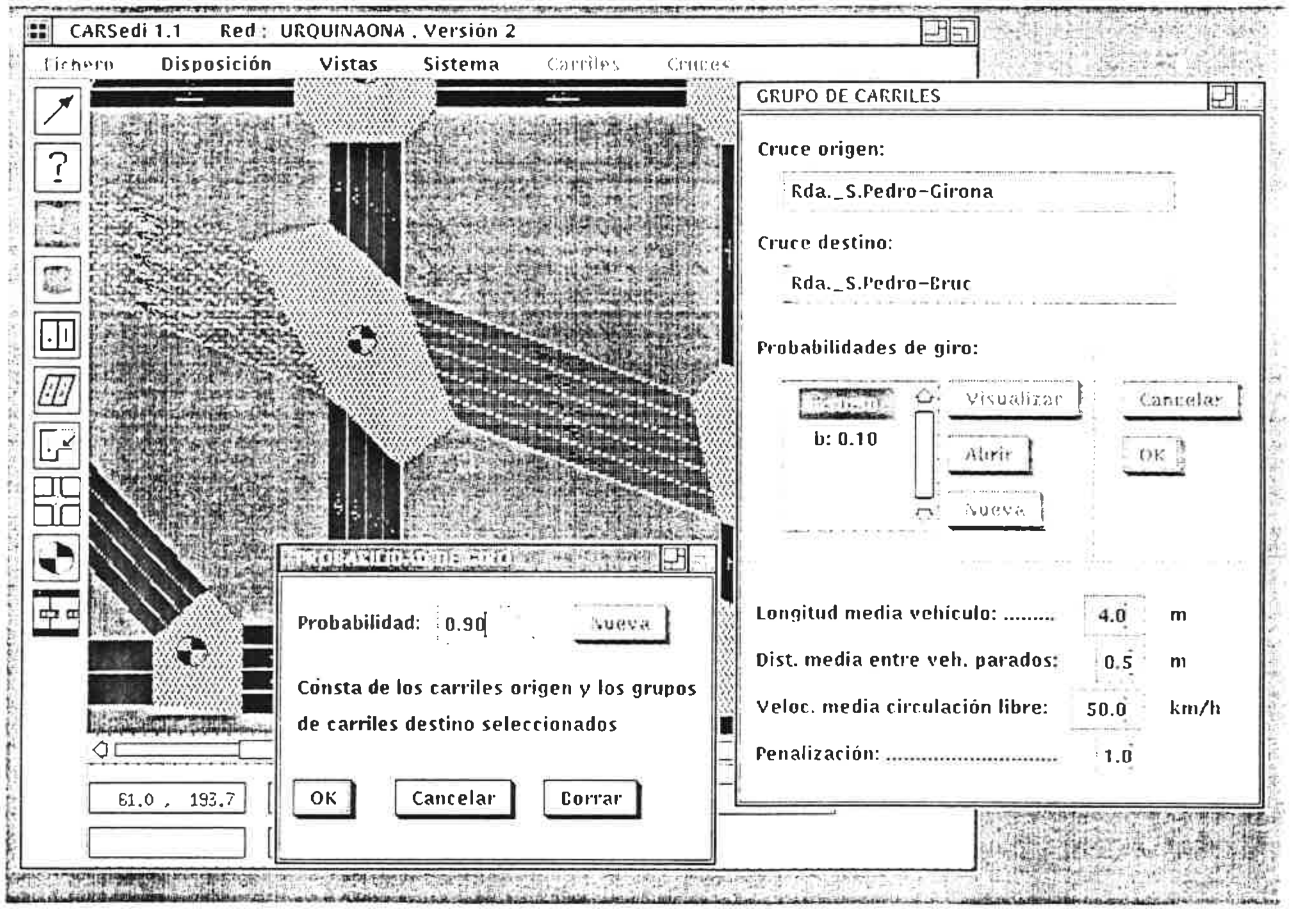


Fig. 2.2 Two sections flanked by blocks.



Probabilidad: 0.90

Consta de los carriles origen y los grupos de carriles destino seleccionados

Cruce origen: Rda._S.Pedro-Girona

Cruce destino: Rda._S.Pedro-Bruc

Probabilidades de giro: b: 0.10

Longitud media vehículo:	4.0	m
Dist. media entre veh. parados:	0.5	m
Veloc. media circulación libre:	50.0	km/h
Penalización:	1.0	

Fig. 2.3 Specification of turnings at the end of a section, in CARSedi.

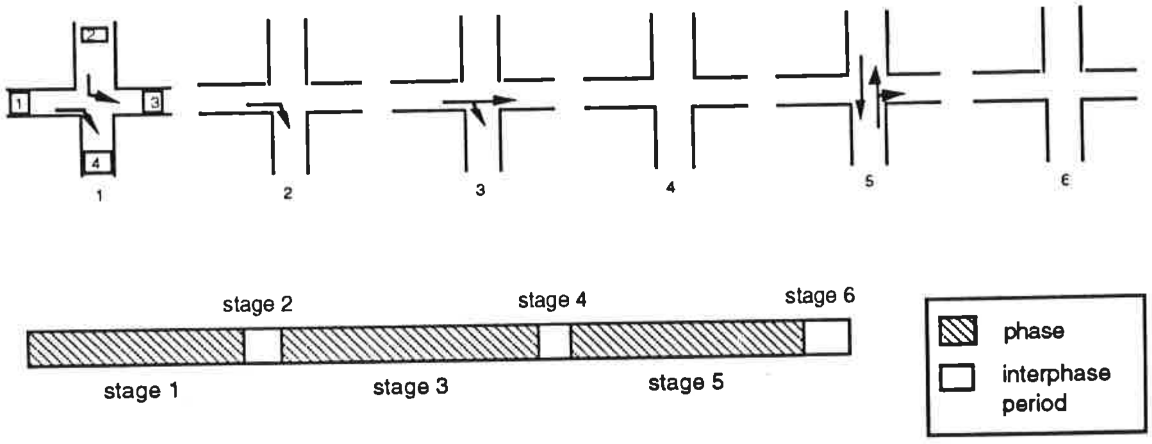
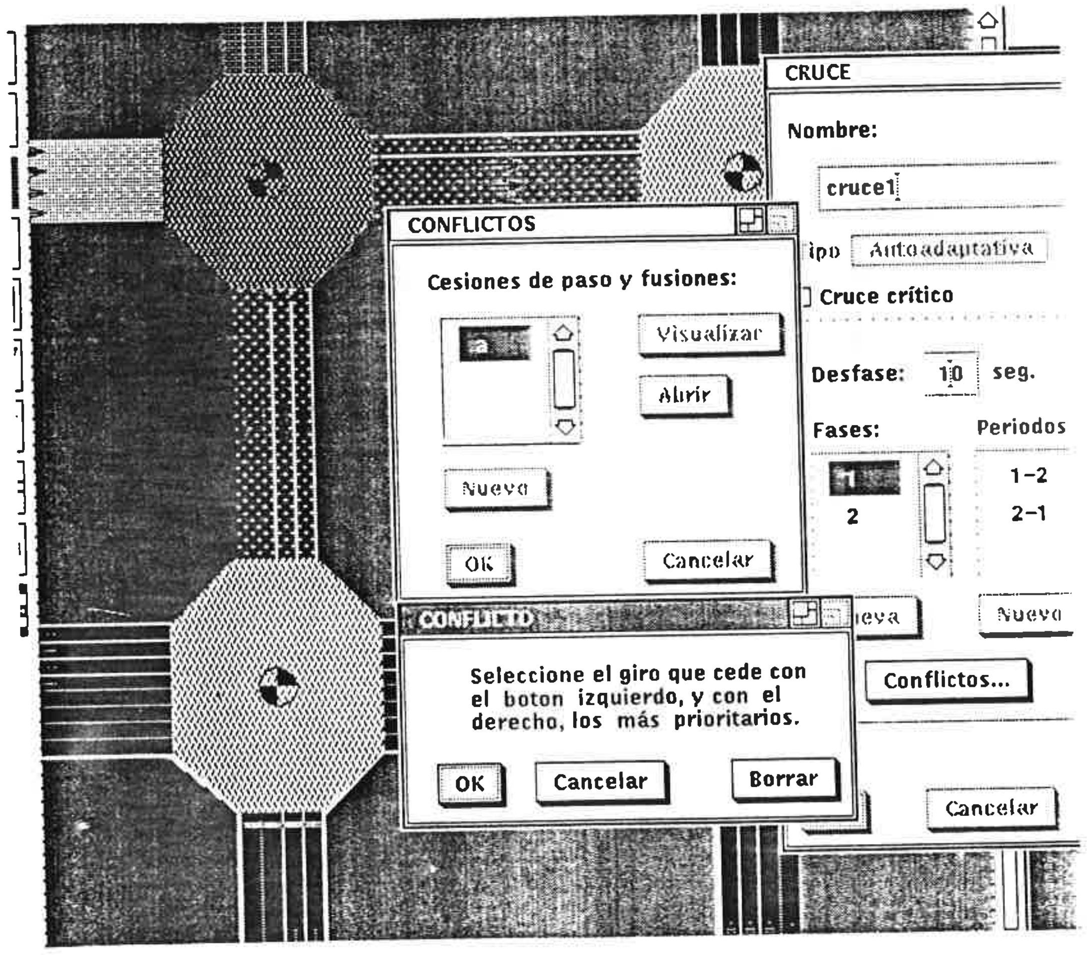


Fig. 2.4 Example of sequence of rights of way at a junction (top) and its modeling in stages (below).



Nombre: cruce1

Tipo: Autoadaptativa

Cruce crítico

Desfase: 10 seg.

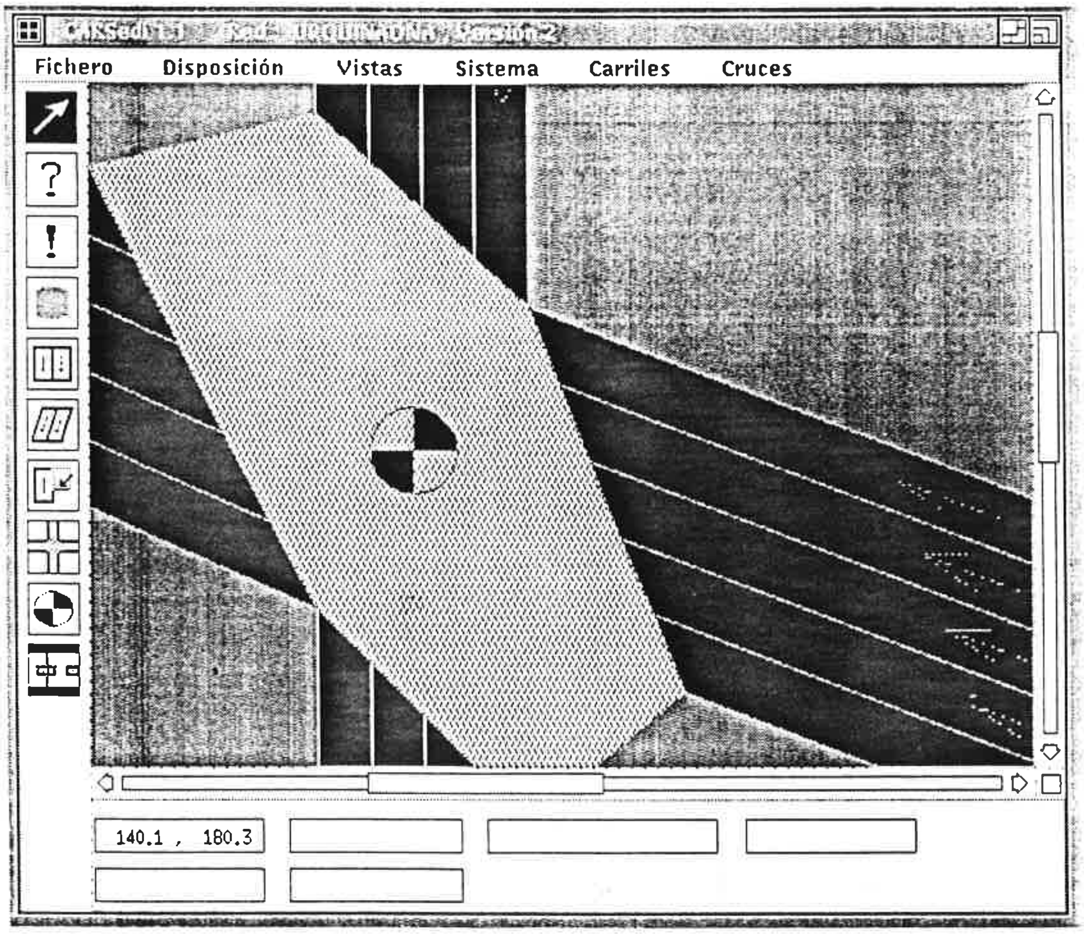
Fases: 1, 2

Periodos: 1-2, 2-1

Cesiones de paso y fusiones:

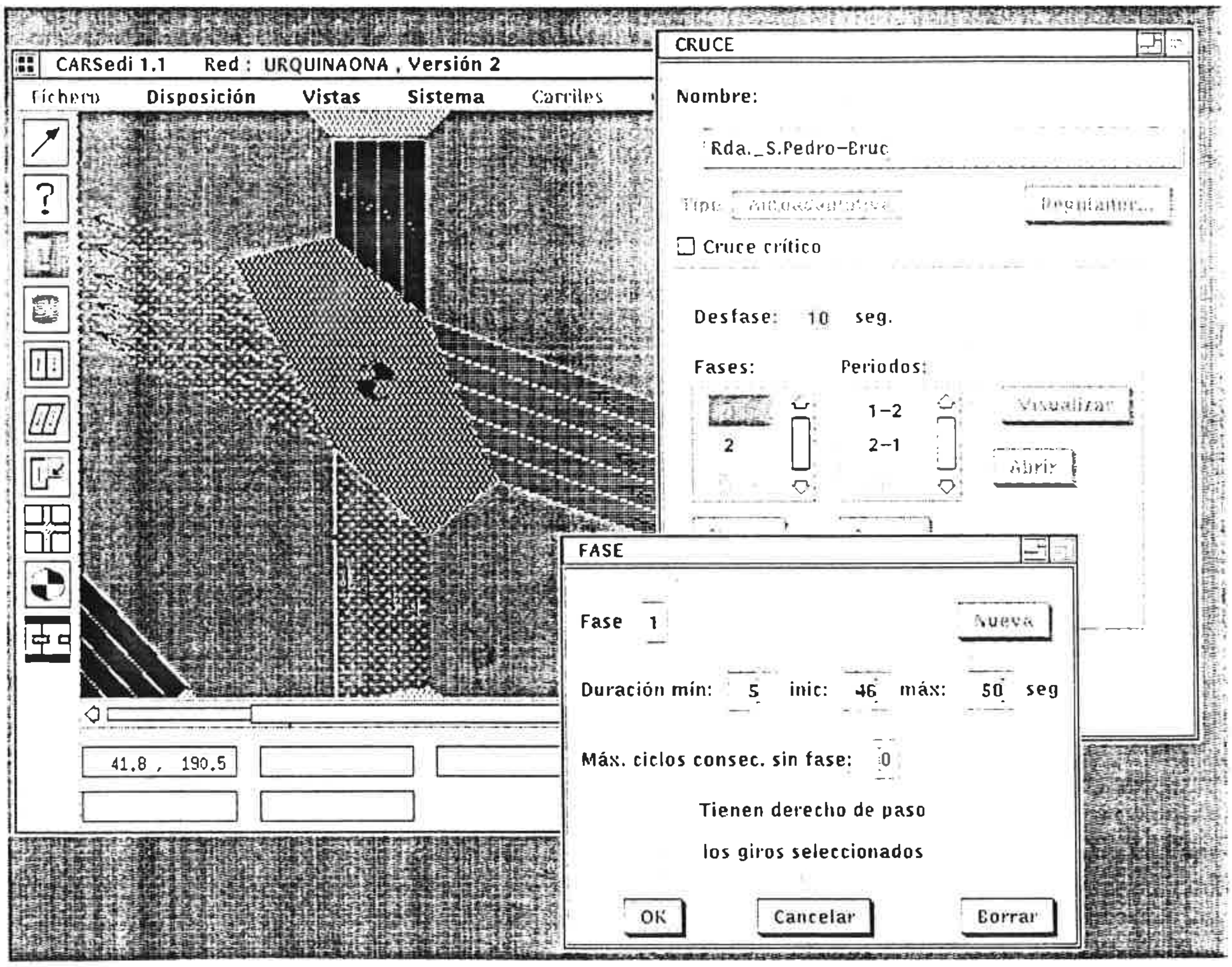
Seleccione el giro que cede con el boton izquierdo, y con el derecho, los más prioritarios.

Fig. 2.5 List of priorities between turnings in CARSedi.



140.1 , 180.3

Fig. 2.6 A junction as displayed in CARSedi.



Nombre: Rda._S.Pedro-Bruc

☐ Cruce crítico

Desfase: 10 seg.

Fases:	Periodos:
1	1-2
2	2-1

Fase 1

Duración mín: 5 inic: 46 máx: 50 seg

Máx. ciclos consec. sin fase: 0

Tienen derecho de paso los giros seleccionados

41.8 , 190.5

Fig. 2.7 Rights of way specification in CARSedi.

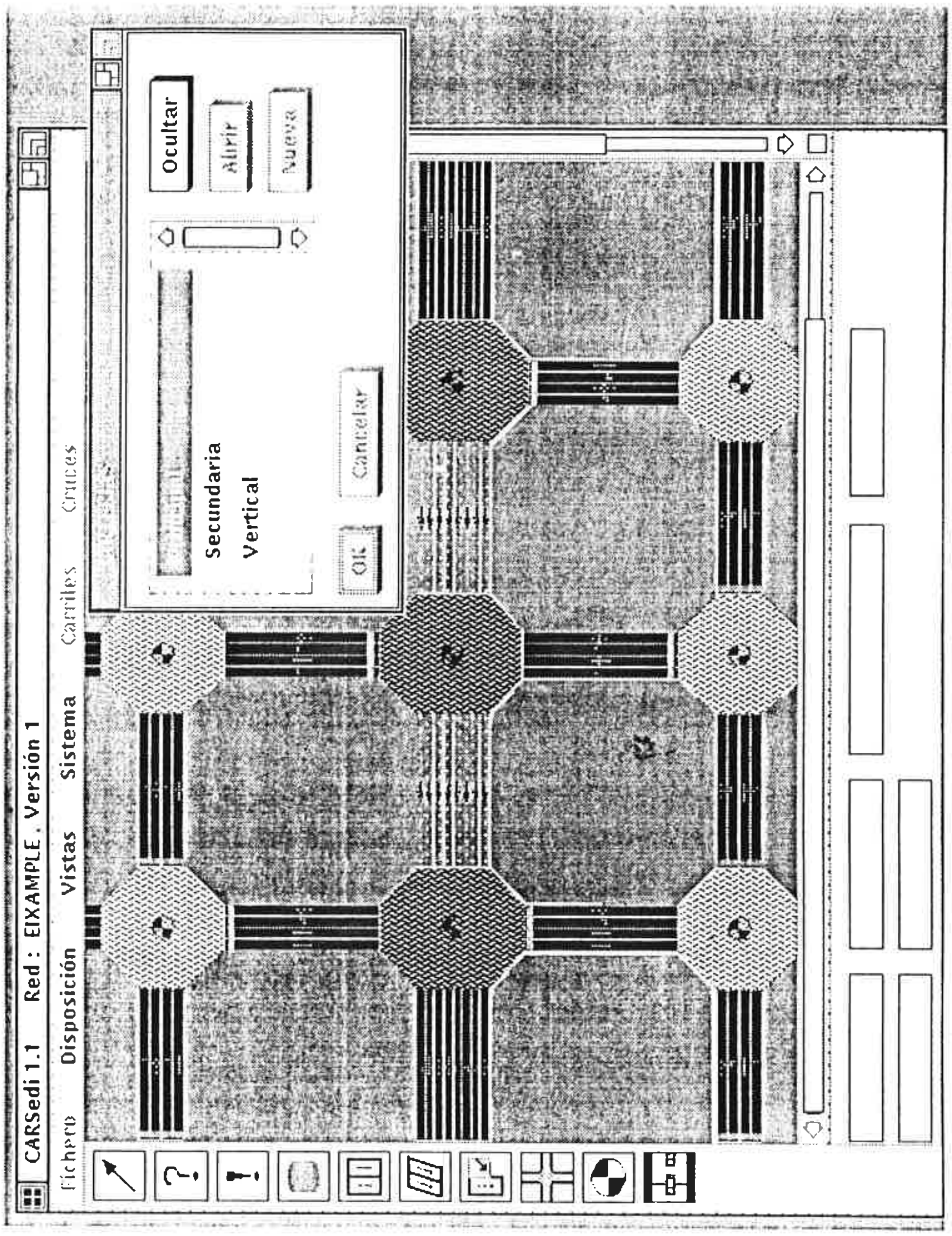
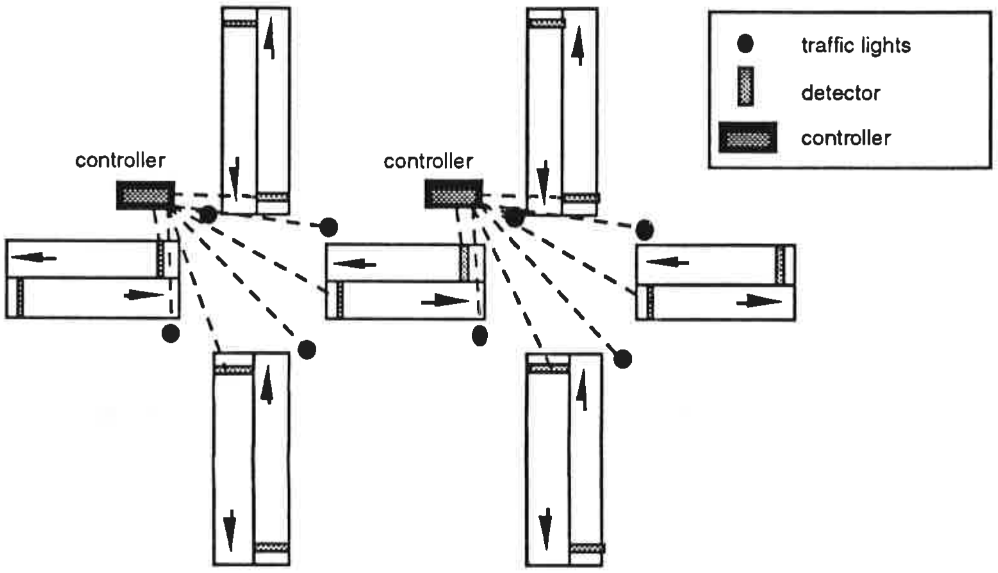
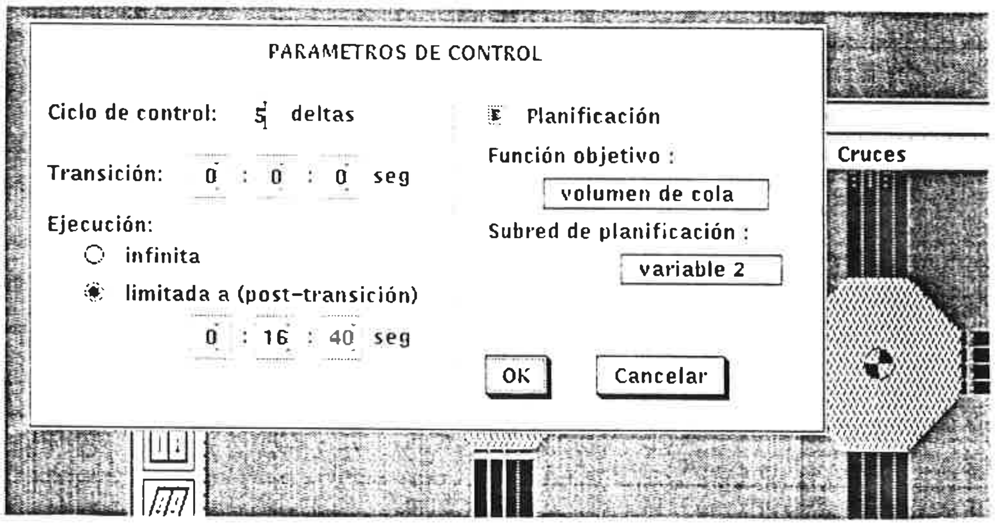


Fig. 2.8 Arterial representation in CARSedi.




PARAMETROS DE CONTROL

Ciclo de control: 5 deltas

Transición: 0 : 0 : 0 seg

Ejecución:

○ infinita

[✓ limitada a (post-transición)]

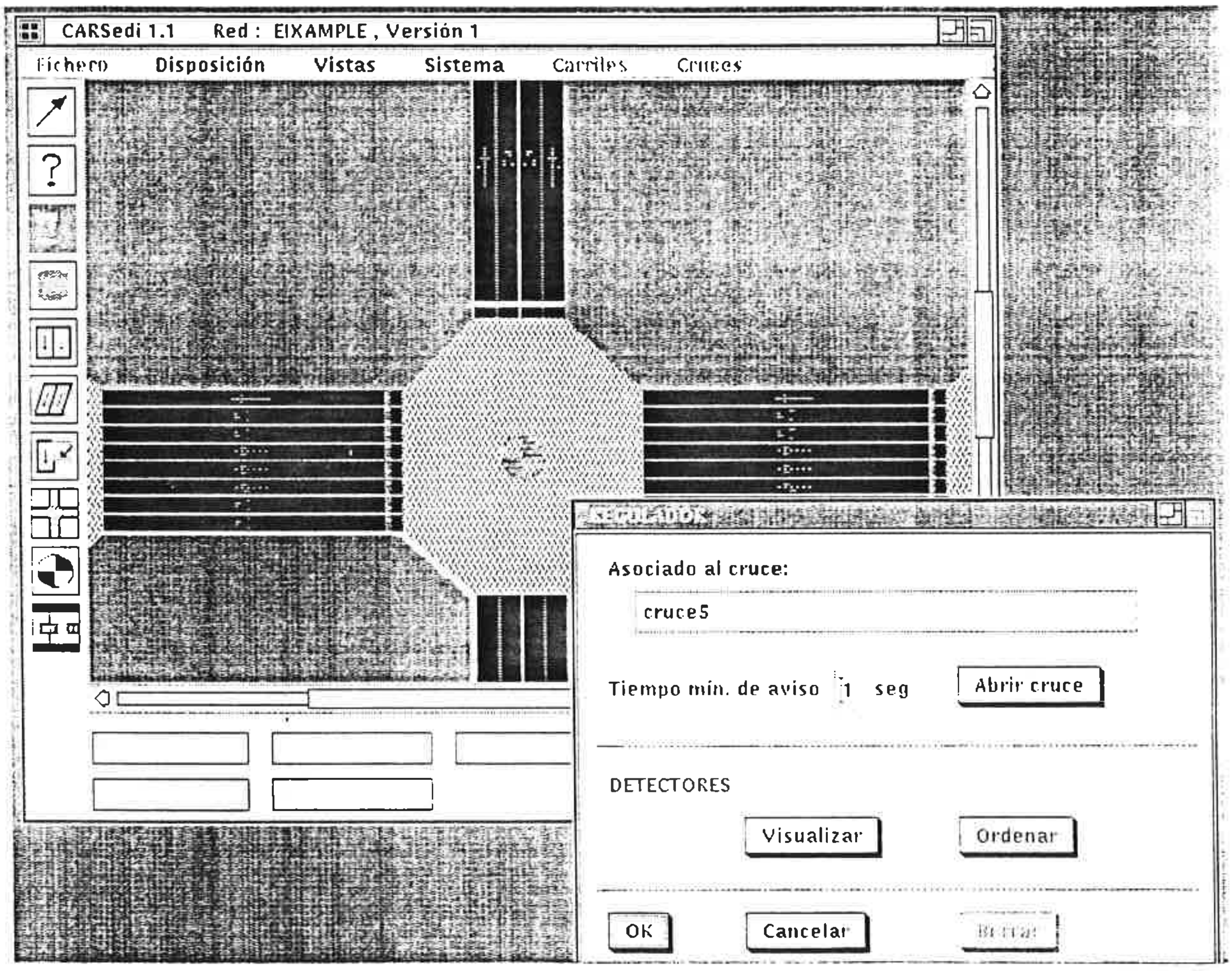
0 : 16 : 40 seg

[✓ Planificación]

Función objetivo : volumen de cola

Subred de planificación : variable 2

Fig. 2.9 Example of devices connected to a controller (top). The figure below shows the CARSedi dialog box where the user specifies control timing parameters.



Asociado al cruce: cruce5

Tiempo mín. de aviso 1 seg

Fig. 2.10 Controller representation in CARSedi, and its associated dialog box.

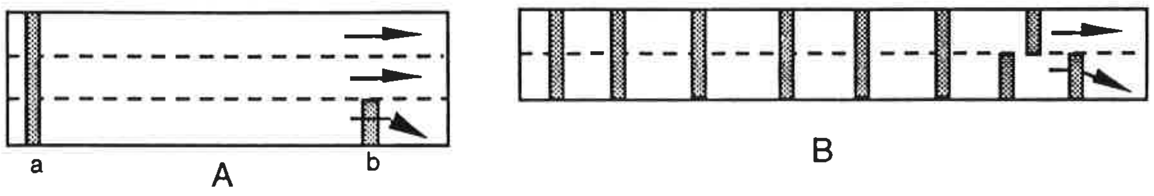
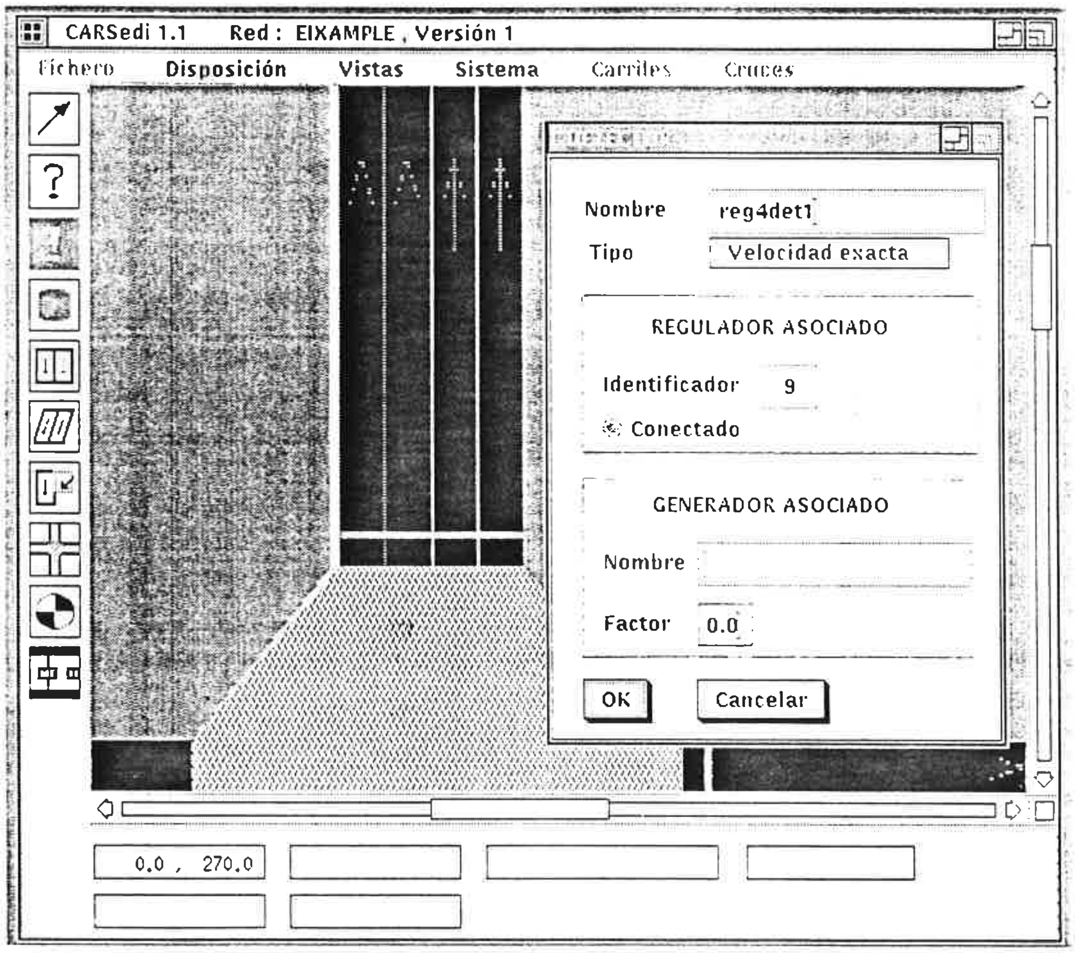


Fig. 2.11 Example of detector positioning. Case B may need video-detection in order to be feasible.



Nombre reg4det1
Tipo Velocidad exacta

REGULADOR ASOCIADO

Identificador 9

Conectado

GENERADOR ASOCIADO

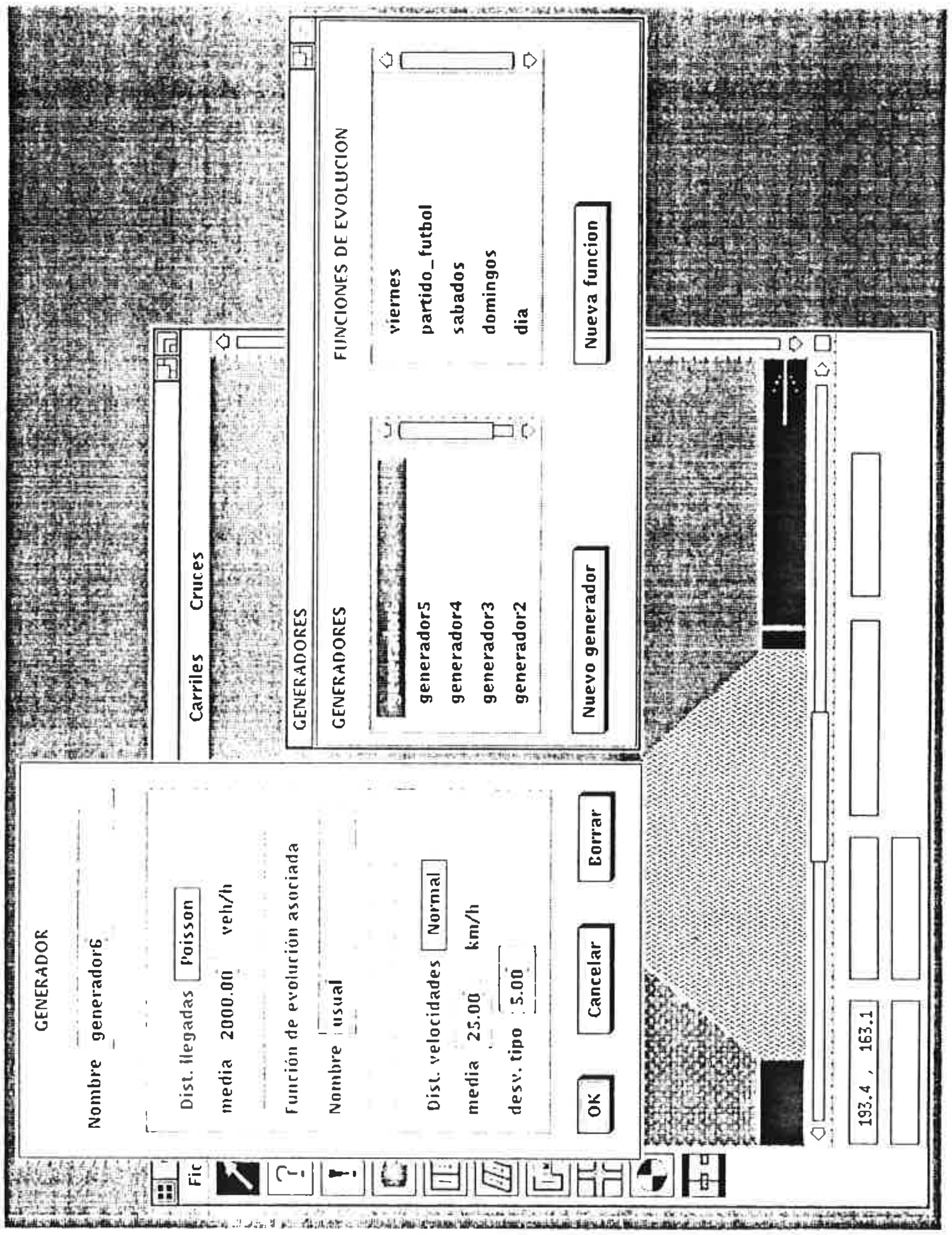
Nombre

Factor 0.0

OK Cancelar

0.0 , 270.0

Fig. 2.12 Detector representation in CARSedi, and its associated dialog box.


GENERADOR

Nombre generador6

Dist. llegadas Poisson
media 2000.00 veh/h

Función de evolución asociada
Nombre usual

Dist. velocidades Normal
media 25.00 km/h
desv. tipo 5.00

OK Cancelar Borrar

GENERADORES

GENERADORES
generador5
generador4
generador3
generador2

Nuevo generador

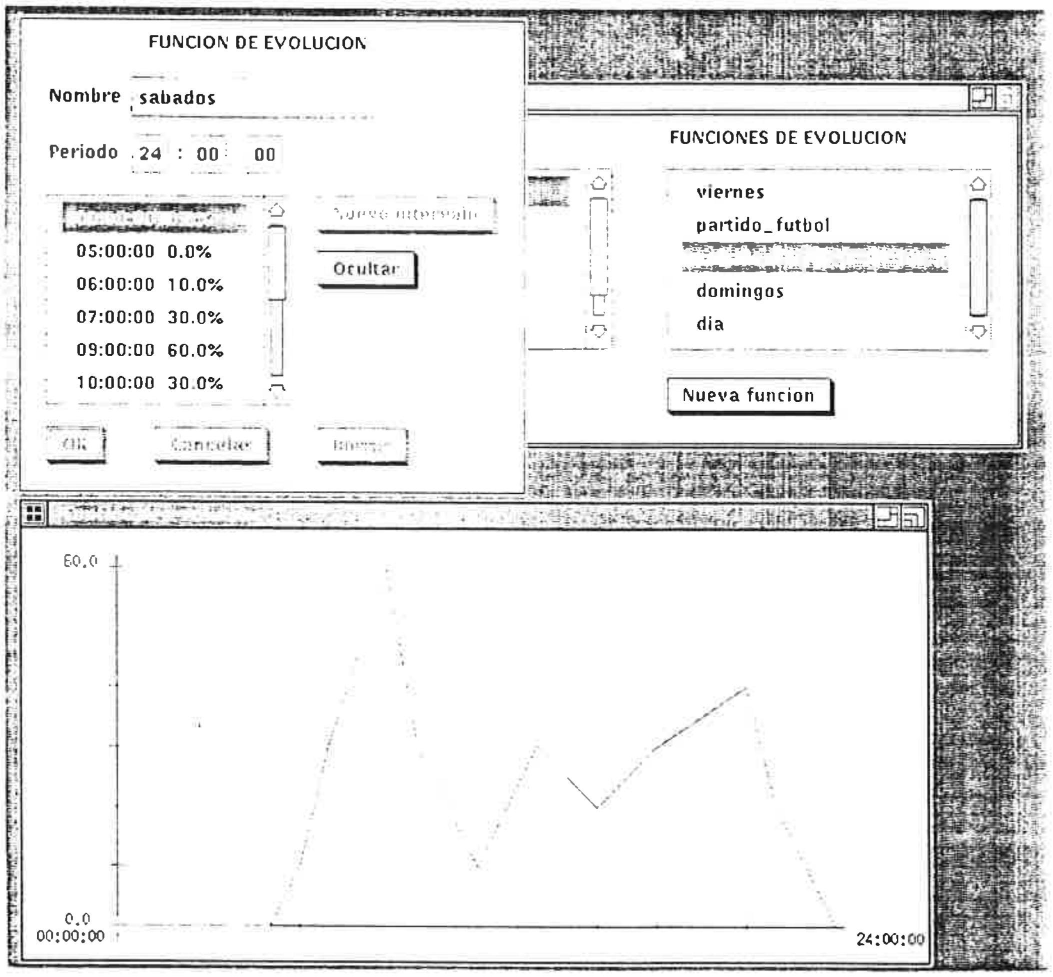
FUNCIONES DE EVOLUCION
viernes
partido_futbol
sabados
domingos
dia

Nueva funcion

Carriles Cruces

193.4 , 163.1

Fig. 2.13 Generator list box and a generator dialog box in CARSedi.



Nombre sabados

Periodo 24 : 00 00

```
05:00:00  0.0%
06:00:00  10.0%
07:00:00  30.0%
09:00:00  60.0%
10:00:00  30.0%
```

```
viernes
partido_futbol
domingos
dia
```

Fig. 2.14 List box of evolution functions, and associated dialog boxes in CARSedi.

3. Sequence of tasks

As real-time control, the responsiveness of the system depends not only on the logical results of computations but also on the time it takes to compute them. In this case, the controlled system is a traffic network and the controller system, CARS, interacts with its environment on the basis of information from the detectors on the streets and the traffic light changes at the intersections. As will be explained, CARS does not rely on the concepts of cycle, split and offset but calculates acyclic settings. CARS sends a signal to the controller of the junction each time a phase has to end.

The temporal sequence of tasks has been designed to assure the correct signal timing in all conditions while allowing flexible hardware requirements, adapting to diverse controllers and computing platforms. Time is discretized in intervals of delta seconds. The system loops in the system control cycle (SCC) of scycle deltas duration, in which the following series of tasks is performed:

1. Updating the state of the vehicles in the network using the detector measurements corresponding to the last SCC.
2. Sending to the controllers the traffic light changes corresponding to the next scycle deltas.
3. Stepping ahead: forecasting the network state at the end of the current SCC.
4. Planning: scheduling the phase endings at the junctions for the duration of the next SCC.

A careful analysis of the timing constraints is necessary. We distinguish the system internal tasks and the environment-related tasks. The former group includes part of Task 1, i.e., updating the state of the vehicles, and Tasks 3 and 4. Of these, only the fourth task can be considered to have variable computing needs from SCC to SCC, due to the variable-size subnetworks used in the planning, and is bounded considering the more time-consuming possibility during the first SCC. The environmental-related group of tasks contains the so-called controller's functions, which are presented in detail in a later section. These functions are classified into initialization functions (CONNECT CONTROLLER, CONNECT DETECTOR, SET CONTROLLERS, and SYNCHRONIZE CONTROLLERS), which take place prior to the first SCC and have no time constraint, and SCC-functions (RESET DETECTORS and CHANGE JUNCTION), whose timing requirements will be analized susequently.

The RESET DETECTORS function carries out the processing of the detector measurements, receiving the measures corresponding to the last scycles deltas. This is part of Task 1, and is periodic in the sense that it must be executed once each scycle deltas. The existence of this task is known at the outset and may be scheduled statically, this is, its computing requirements can be evaluated before the full operation.

The CHANGE JUNCTION function sends signals to a junction to end the current phase and proceed to the next possibly through an interphase period. This is more dynamical in the sense that it can happen at each delta of the SCC, and for any number of junctions. Again, this can be bounded to the worst case, and thus the computing requirements are evaluated statically. The function returns control immediately and local processing at each junction is required to accomplish the appropriate light changes. This assures safe lighting at each junction when debugging the system.

The design of the controller's functions, requiring some local processing on the controller associated with each junction, turns them from hard to soft real-time tasks, that is, they retain some disminished value after its deadline.

CARS can act as a conventional fixed control system (Figures 3.1 and 3.2) or as demand-responsive (Figures 3.3 and 3.4); in the first case, Tasks 3 and 4 are omitted. In order to clarify ideas, let us explain the more complicated, fourth case (Figure 3.4). At point (1), controllers send the detectors measurements corresponding to the last four deltas. These measurements are used by CARS to update its representation of the traffic state in the network, so that it more accurately matches the real-world state at time (1). This update process is called 'feedback'. Then, the system estimates the traffic state at point (2) —this is called 'stepping ahead'. From this estimated state at (2), the system considers a set of alternative control plans for the near future —this is called 'planning', and the near future is usually up to two minutes from point (2). A 'best' control plan is choosed and the new control timings are implemented during, at least, four deltas —the length of the next SCC.

The length of the delta time interval and the SCC are user-specified parameters, the values most used in the tests being delta = 2 seconds and scycle = 5 deltas. Note that, because the planning process can extend on more than one time interval, a reduced delta of 1 or 2 seconds can be used in medium-size networks without requiring a powerful computing platform. Small time intervals allow more exact modeling, feedback, and phase changes. In comparison, most demand-responsive control systems currently in use are reported to use 5-sec intervals and are restricted to a specific computing platform.

Approximately every five minutes, a special SCC called the 'maintenance cycle' takes place, where the stepping ahead and planning processes are substituted by:

- Recalculating the parameters of the predictors of external arrivals.
- Updating the turning movement ratios where sufficient detector measurements are available.
- Reconsidering the planning subnetworks according to the traffic conditions around each demand-responsive junction.

Similarly, it is convenient to enter a transition period when starting the system before the actual planning begins. During this period the system acts as fixed-control, while detectors

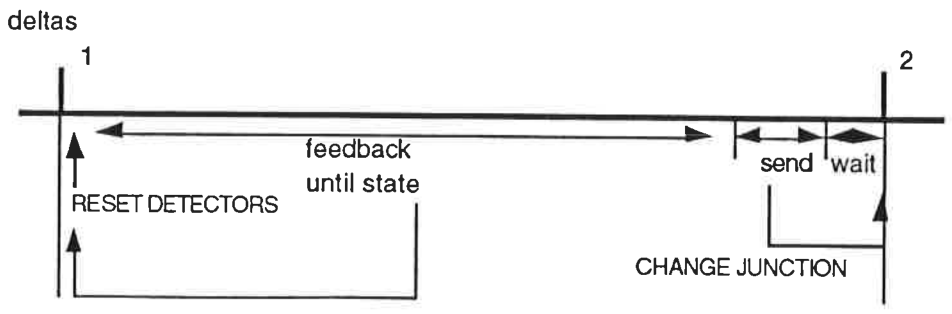


Fig. 3.1 SCC of 1 delta, without planning.

provide measurements to initialize the various predictors. A transition period of 2 minutes has proved to be satisfactory in the tests; by that time almost all the predictors are already initialized and the remaining have substitution mechanisms to provide convenient predictions until their complete start.

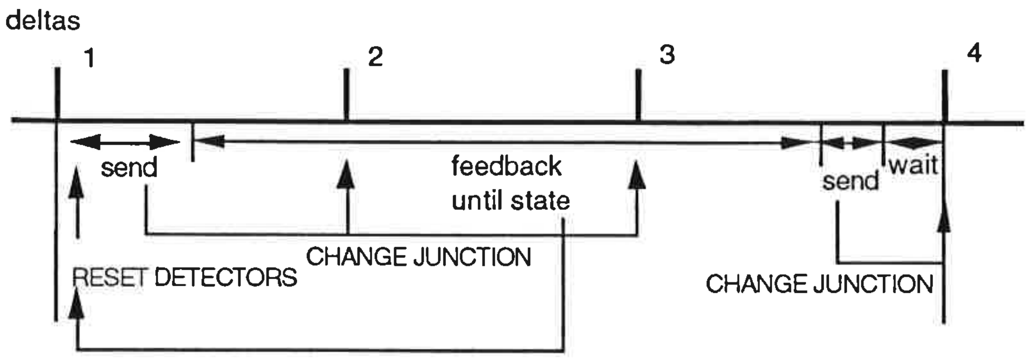


Fig. 3.2 SCC of 3 deltas, without planning.

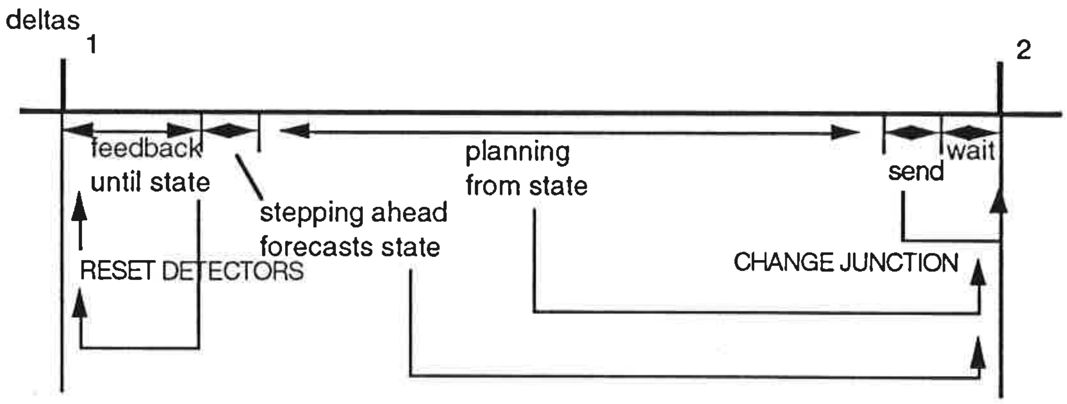


Fig. 3.3 SCC of 1 delta, with planning.

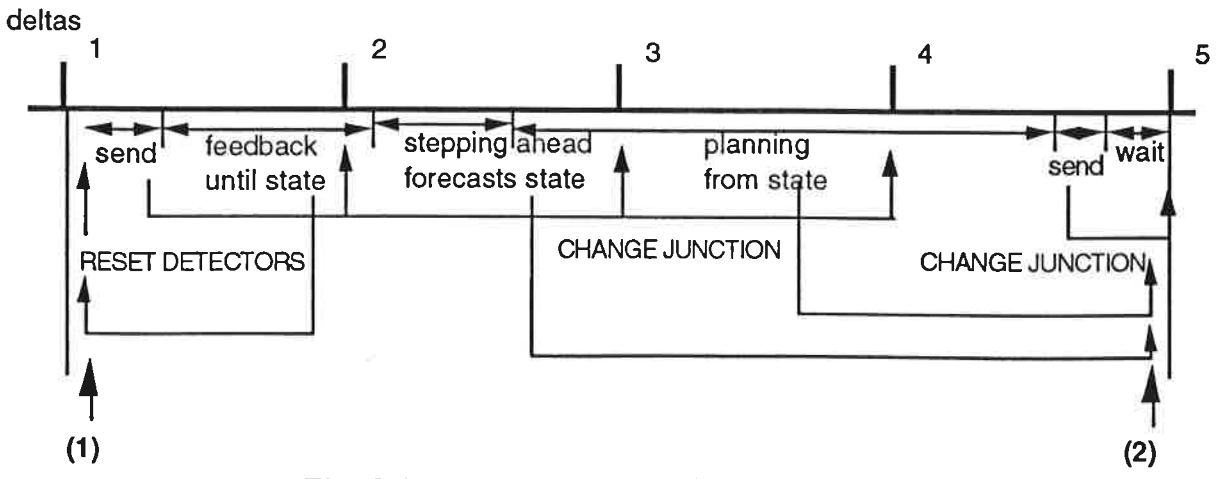


Fig. 3.4 SCC of 4 deltas, with planning.

4. Simulation model

So far, traffic models employed in 3rd-generation traffic control systems have been severely constrained by the available computing power and the real-time operation requirements. In most systems, a mathematical model is used to represent and short-term predict the vehicles' state. The way queues of vehicles are modeled can give an idea of the degree of accuracy of the models:

- Vertical queues. The vehicles move along the sections at a constant speed and, if exit is not possible, accumulate vertically at the end of the section. Examples of such systems are DYPIC (Robertson and Bretherton 1974), PRODYN (Henry et al. 1983), OPAC (Gartner 1983), ACTS (Kaltenbach et al. 1986), and SAST (Lin et al. 1988). Of these systems, only ACTS can be considered to employ a simulation model.
- Horizontal queues. Although still moving at a constant speed, the stopped vehicles accumulate in a horizontal queue. The queue is always at the end of the section, and shortens at its back end as vehicles discharge. An example of such systems is SCOOT (Hunt, Robertson et al. 1981), where the vehicles are generated independently in each section, using cyclic flow profiles, and move according to a platoon dispersion model.
- N groups of stopped vehicles. The vehicles move in groups ('packets') along the section, according to a simplified car-following algorithm. A packet can join other packets or can subdivide, and thus contains a variable number of vehicles. This allows for a more realistic queue modeling, i.e., queues of stopped vehicles propagating backwards, and more than one group of stopped vehicles when in congested situations the queue cannot discharge completely during the green time. To the best of our knowledge we are the first to introduce such a simulation model, known as PACKSIM (Grau and Barceló 1992) — for 'packet simulator'.

The simulation model PACKSIM is used in CARS in order to represent the vehicles in the network, short-term predict the vehicles' state, and present a comprehensible graphical output. Special emphasis has been put on queue modeling in order to deal with congested situations, accurately evaluate the objective function and find a progressive signalization in the arterials.

To evaluate the control changes, the system needs to advance the traffic state along a time horizon. Hence, the PACKSIM simulator needs to model the average behavior of the vehicles. For instance, at the end of a section, the groups of vehicles have to divide according to the turning percentages. The model can be considered as mesoscopic, because it uses a simple car-following algorithm and is not based on the macroscopic concepts of flow, density and speed.

To use the model in a demand-responsive control system, it must fulfill the following requirements:

- It must be accurate enough to advance (predict) the state approximately one minute in the future, the time horizon required to evaluate changes in the control policy for the junctions.
- Emphasis on queue modeling. The objective function used to evaluate changes in

control policy is based on the number of vehicles stopped. Accurate queue modeling is also needed for the evaluation of progressive signalization in the arterials.

- The model has to allow for a flexible feedback system: none, one or many detectors in each section, on all or some lanes. Such a system can be more easily used the more realistic the vehicle modeling is. This feature will become more important when the use of advanced video-detection systems (Michalopoulos et al. 1989) become more widespread.
- Suitable computing requirements, as is obvious in a real-time operation.

4.1 Additional objects

PACKSIM uses all the objects introduced in **2.** and adds four more types of objects: links, bridge links, junctions, and packets of vehicles. Links model a path of vehicles within a section. We distinguish as many links as there are possible turns at the end of the section; see Figure 4.1. The vehicles move (turn) from one section to another through the bridge links, i.e., bridge links model the path of vehicles within the junction; see Figure 4.2. Entering a section, the vehicles disperse throughout the links according to the turning percentages. At the end of a link, vehicles enter the bridge link into the next section.

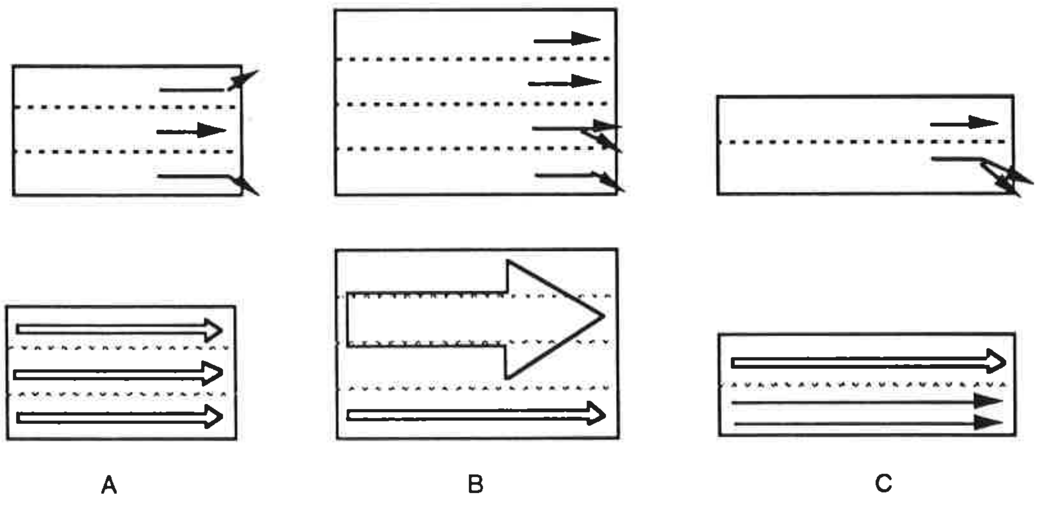


Fig. 4.1 Examples of how sections (top) model as links (below).

The vehicles move grouped in packets through the links and bridge links. A packet contains a floating number of vehicles. The packets are generated with the feedback of detector measurements, merge and split along the links, and vary their speed according to a simple car-following algorithm which will be presented hereafter. Figures 4.3, 4.4, 4.5, and 4.6 are photographs showing PACKSIM on various networks. The idea of grouping vehicles in packets has already been used in CONTRAM (Leonard and Gower 1982), a traffic assignment model that can deal with time-varying traffic conditions in a network. However, CONTRAM's packets are much simpler because they have a fixed number of vehicles, move at constant speed, and accumulate vertically at the end of the link, thus being a vertical-queue model.

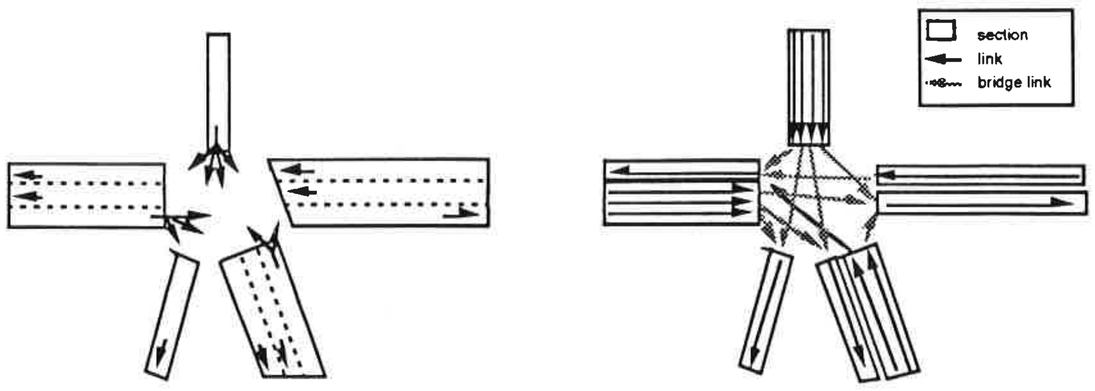


Fig. 4.2 Example of junction (left) and its modeling in links (right).

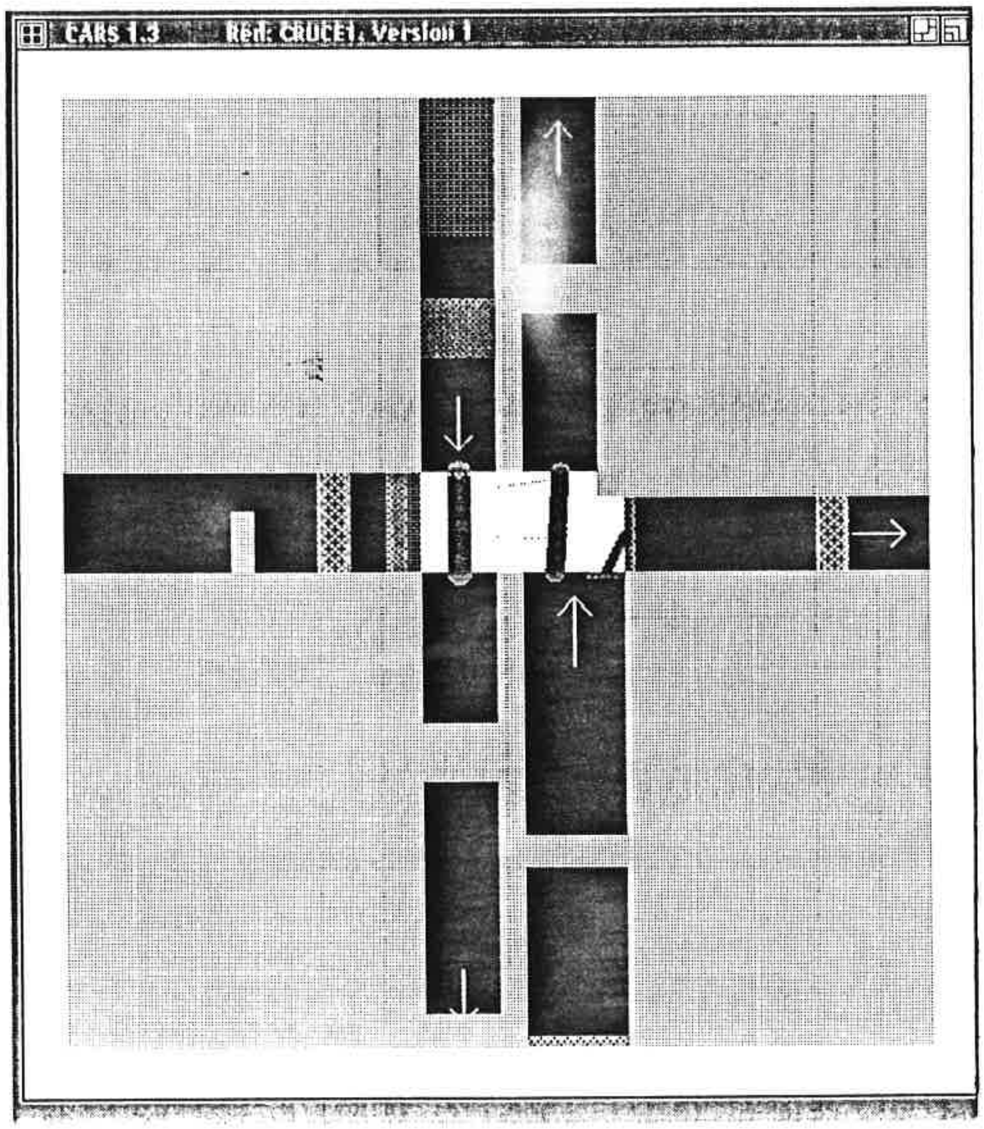


Fig. 4.3 The PACKSIM simulator on a junction.

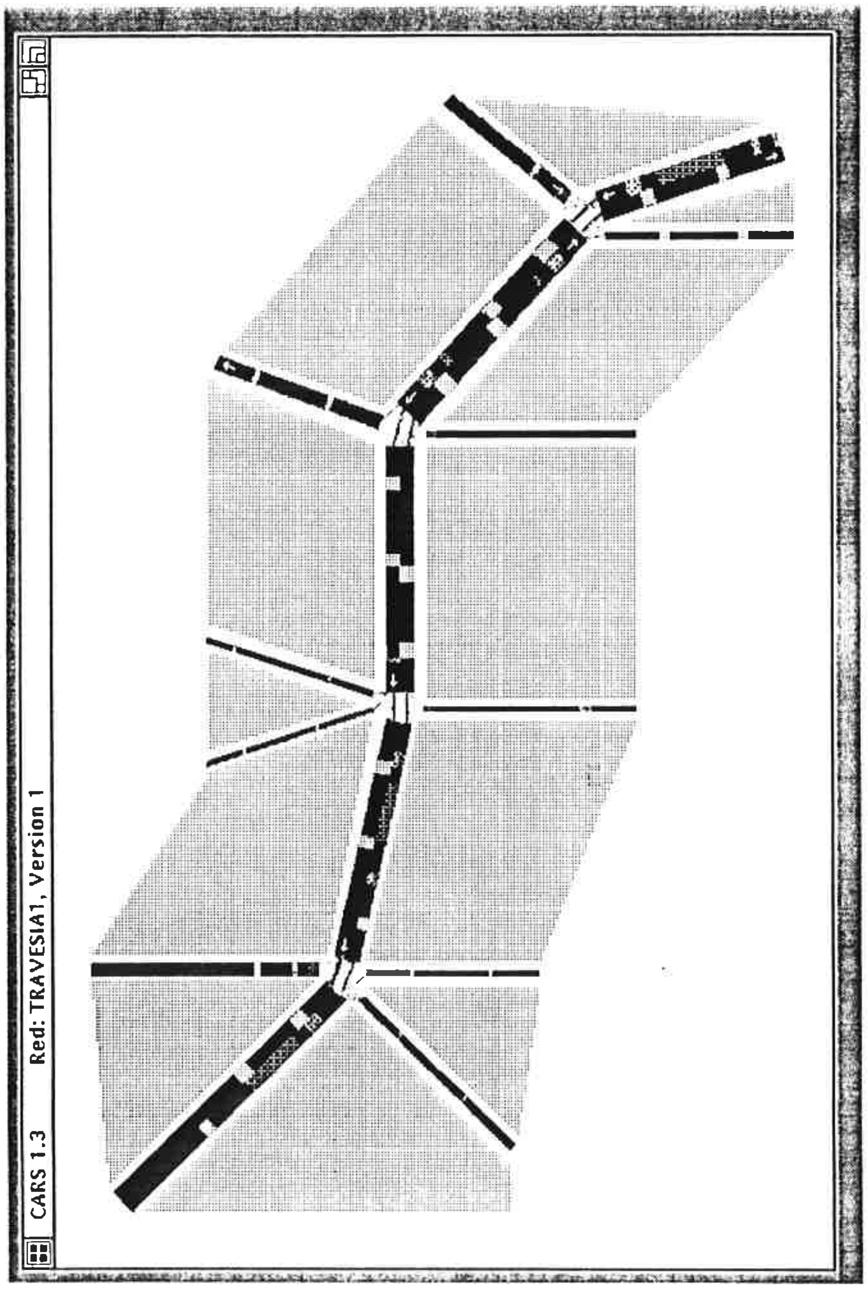


Fig. 4.4 The PACKSIM simulator on an arterial.

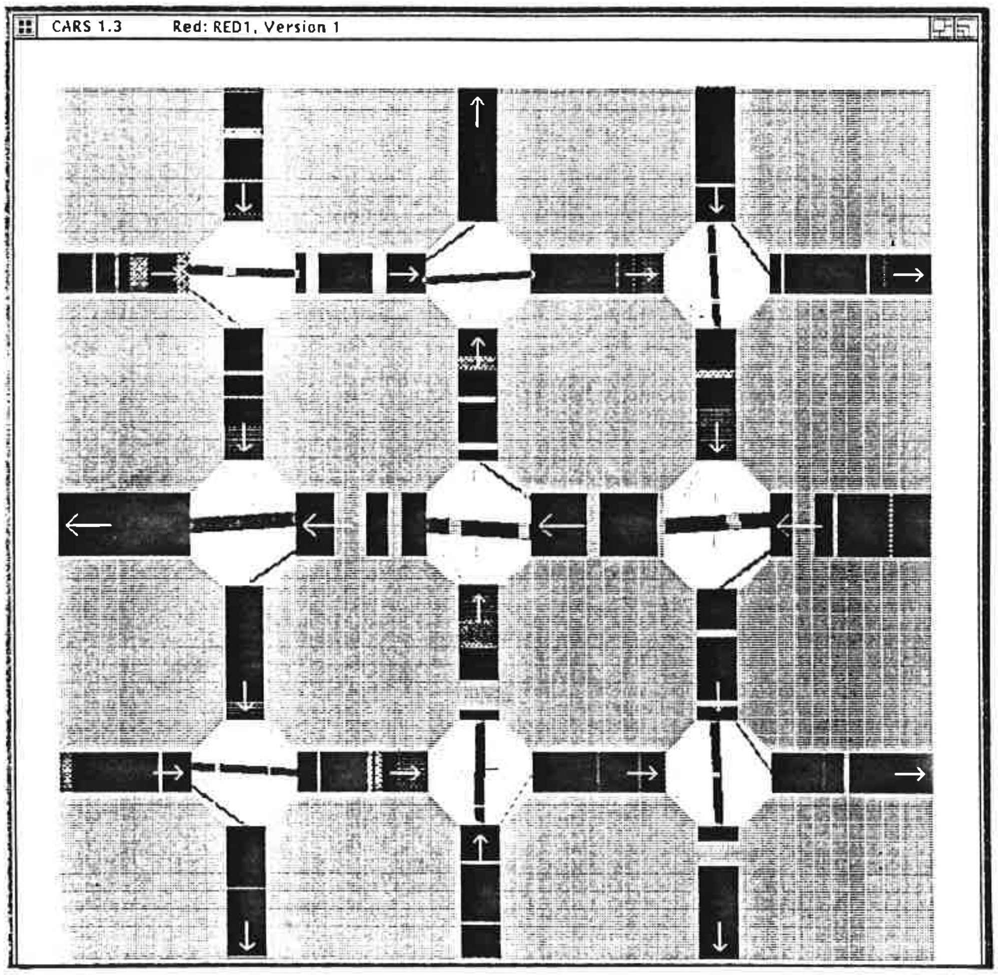


Fig. 4.5 The PACKSIM simulator on a network.

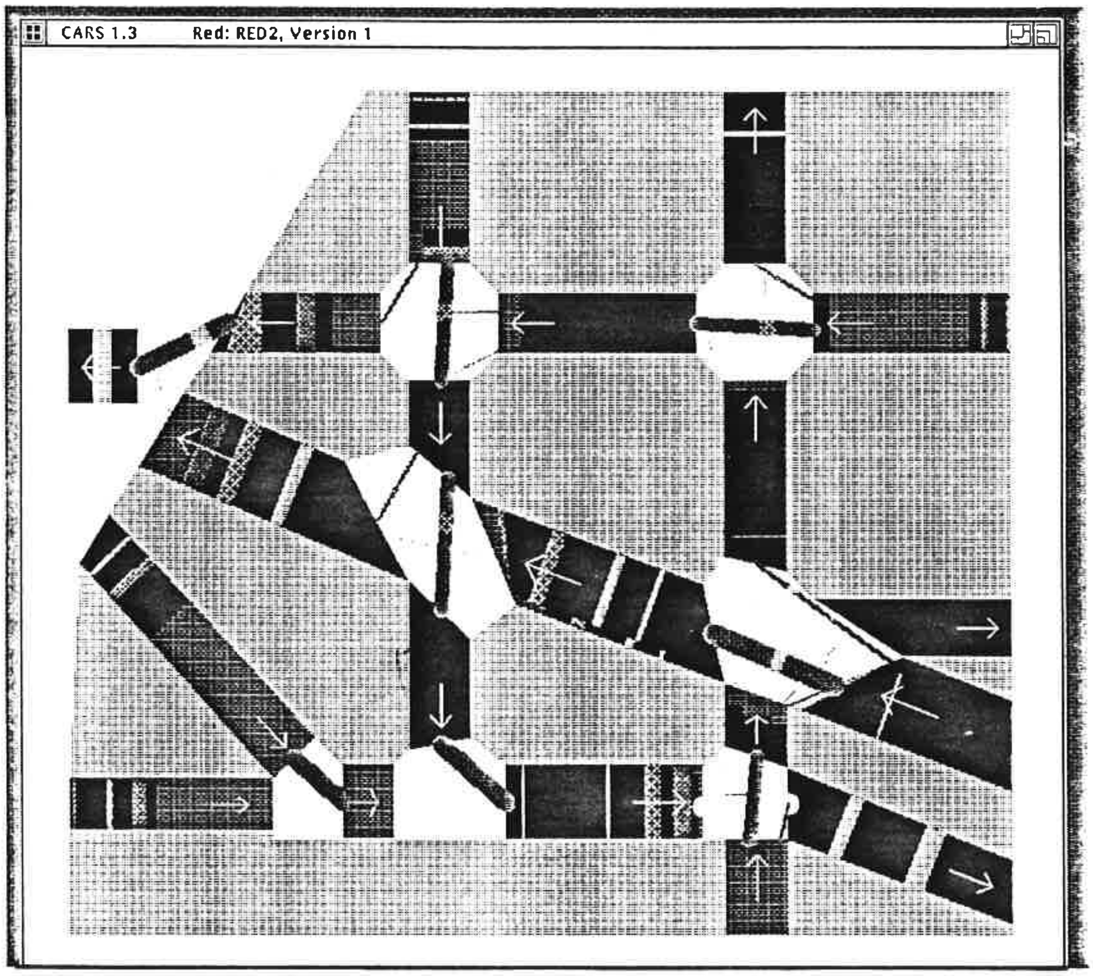


Fig. 4.6 The PACKSIM simulator on a network.

4.2 Updating steps

Advancing the vehicles' position requires two steps to allow the transfer of vehicles from one section to another.

• <u>Step 1</u>

For each section and link, we move first the packets of vehicles in the bridge link towards an 'entrance container' in the destination section and, second, the packets in the link towards the bridge link and the entrance buffer. The right of way of this link, if existing during this Δt, is reconsidered to account for the give-ways and the available free space in the destination section.

The entrance container of a section serves as a deposit between this step and the next. It is characterized by: number of vehicles, mean speed and desired position. The first packet of vehicles entering the section determines the desired position.

The sections are sorted so that, when updating a section, the destination sections will already have been updated and the available space known. Unfortunately, in a network such a sorting is not always possible, and then we need to, temporarily, predict the available space.

After updating the vehicles, we consider the exit blocks. A link has its exit blocked if, having right of way, there are vehicles which cannot exit. This is due to give-ways or to lack of available space in the destination section. If the exit is blocked, then the link increases its desired width, as will be explained shortly.

The first step contains the following actions:

```
for each section
        predict the available space in the next $\Delta t$ time
endfor
for each section
        for each link
                move packets in the bridge link
                move packets in the link
        endfor
        evaluate available space
        modify desired widths of the links according to the blocked exits
endfor
```

• <u>Step 2</u>

For each section, the contents of the entrance buffer is distributed among the links. The distribution is made proportionally to the turning percentages at the end of the section, but we consider also the possible entrance blockage in some links. In each link, the packet so constituted is made to advance towards the desired position.

Once the updating of vehicles is finished, we revise the width of the links in the sections where the link's desired widths have changed. This happens as a consequence of blockages and disblockages in the vehicle exit. Therefore, the following actions need to be performed:

```
for each section
    for each link
        take a percentage of vehicles from the entrance container
        move the packet towards the desired position
    endfor
endfor
```

4.3 Movement of packets

We group the vehicles in packets in order to achieve a reasonable computing speed while maintaining accuracy in average conditions. The packets move along the links and bridge links. A packet contains a real number of vehicles moving at a given speed and interactions with the other packets according to a simple car-following algorithm. The length of a packet, Lp, depends on the number of vehicles it contains, nVehp, and the average length of a vehicle in the link, Lvl:

$$Lp = nVehp * Lvl$$

The average length of a vehicle in a link is proportional to the average length of a vehicle in the section, Lvs, and inversely proportional to the width of the link measured in number of lanes, Wl. To speed up the process, at this moment we also add the distance between stopped vehicles, Ds:

$$Lvl = (Lvs + Ds) / Wl$$

This approximation is more accurate as the link gets more congested and it is in these conditions when we need a more precise evaluation of the free space available.

4.3.1 Packet-following algorithm

The decision about which car-following algorithm to use is conditioned by the following factors:

- it has to allow relatively long intervals of time Δt between updates (1-5 secs) in order to reduce computing time and to adapt to the detectors' timings.
- it should tend to merge packets under certain conditions of proximity. The fewer packets in a link, the less computing time required to move them, and this means more time to consider changes in the control policy.
- we are interested in it being most accurate in the short sections (max. 400 m) where application of the demand-responsive system is recommended. Furthermore, in order to correct inaccuracies, feedback from the detectors is expected at least in some sections.
- computational efficiency.

With these conditions in mind, we have been testing algorithms with the AIMSUN

(Barceló, Ferrer and Montero 1989) microscopic simulator on areas with short sections (50-300 m) between the junctions. Several algorithms have shown a good compromise between the required capabilities, and we describe here the one that has produced the best results.

We suppose that, in free conditions, a packet accelerates at a constant rate, Ap, specified by the user, until reaching the maximum speed allowed in the section. If the packet reaches an obstacle — the preceding packet or the end of the section — in less than the 'influence time', then its speed is averaged with the speed of the obstacle. The algorithm is:

```
        /* speed increment in free movement */
Vp = minimum(Vp + Ap * $\Delta$t, Vsmax)
        /* predecessor or end-of-section influences */
if there is a preceding packet then
                /* if influenced by the predecessor, averages its speed */
        if Vp * Ti > Dpp then
                Vp = $c_1$ * Vp + $c_2$ * $Vp_{predecessor}$
        endif
else if Vp * Ti > Dpes then
                /* if it has no right of way, reduce speed */
        if no right of way then
                Vp = $c_1$ * Vp
        else if Vp > Vtmax then
                        /* there is a maximum turning speed */
                Vp = $c_1$ * Vp + $c_2$ * Vtmax
        endif
endif
```

where,	Vp: speed of the packet.
	Vsmax: maximum speed in the section.
	Vtmax: maximum turning speed.
	Ap: constant acceleration of the packet.
	Dpp: distance between packets.
	Dpes: distance from a packet to the end of the section.
	Δt: interval of time between two vehicles' updating.
	Ti: influence time.
	$c_1 + c_2 = 1.0$ and $c_1, c_2 >= 0$.

So far, with $\Delta t = 2$ secs, we have found that the values $c_1 = 0.5$ and $c_2 = 1 - c_1 = 0.5$ give good results.

In the bridge links, the algorithm increments the speed limited only by the maximum turning speed, Vtmax, since we assume that the packets do not stop until reaching the entrance container in the destination section. The algorithm reduces then to the speed increment in free movement:

$$Vp = \text{minimum}(Vp + Ap * \Delta t, Vtmax)$$

Once the speed has been obtained, the desired position, Xdes, of the packet in the link or bridge link is calculated as:

$$X_{des} = X_{old} + Vp * \Delta t$$

following which the packet tries to advance to this position. If in the course of this movement the packet reaches the back end of the predecessor packet, then it joins it and adopts its speed.

4.3.2 Merging and splitting of packets

While moving along links and bridge links, packets of vehicles can produce merging and splitting phenomena. This can happen also during the feedback due to corrections discussed in **5**. The lack of free memory of packets can also produce mergings, as described in **4.3.4**.

As a packet moves, if it reaches the back end of the preceding packet, then they merge into a single packet. The resulting packet length is the addition of both and takes the speed of the first one (Figure 4.7).

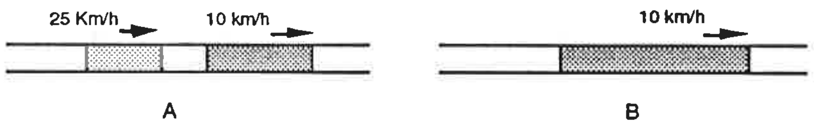


Fig. 4.7 A packet reaching its predecessor (A) and joining it (B).

If the preceding packet is stopped, the merging needs to be dealt with more carefully, as it influences the objective function. In this case, the merging is done by means of a 'gradual stopping'.

The splitting of a packet in two can happen when a queue discharges and also when the new position of the packet lies between the link and the bridge link. The former case happens when a stopped packet gradually starts to move, generating new packets from the front end (Figure 4.8). The user specifies the number of vehicles starting from stopped situation, per lane and delta, nVehDis_lane (usually around 1.0). From this parameter, the number of vehicles, nVehDisl, in each one of the new discharge packets is calculated as Wl * nVehDis_lane, being Wl the link width —in number of lanes as will be explained in **4.5**.

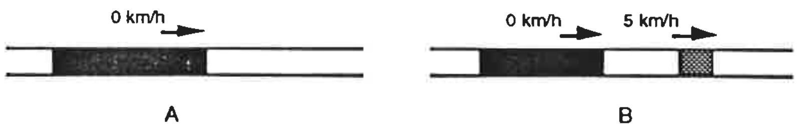


Fig. 4.8 A stopped packet (A) and its gradual start (B).

Each link and bridge link has a maximum number of packets they can use and this is considered when updating a packet: the division or generation of packets is only possible if there are available packets in the memory; otherwise, either the whole packet moves or the discharged packet joins its predecessor. Because this will happen when a lot of small packets exist in the section, the distances between them are likely to be rather small and the inaccuracy produced by this 'hard joining' slight.

4.3.3 Gradual stopping

The case of a packet having to stop because of reaching either a stopped packet or the end of the link with no right of way requires special care, as it affects the objective function. In both cases, instantaneous stopping produces more inaccuracy the bigger the packet is. A gradual stopping is preferable: the packet gradually stops from its front end, while the rest reduce speed (Figure 4.9).

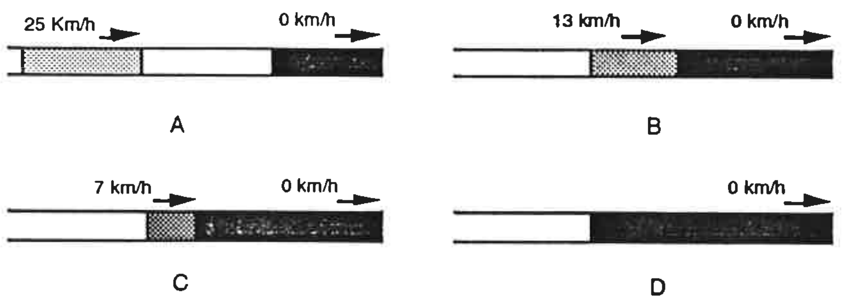


Fig. 4.9 Gradual stopping when reaching a stopped packet.

In this situation, the number of vehicles stopping in each Δt time interval depends on the desired position of the packet, the portion of packet which stops being that which would superpose on the preceding stopped packet. This position grows at a slower pace as the packet-following algorithm reduces the speed of the packet. Therefore, the packet stops at a progressively slower pace until the whole packet has joined the stopped predecessor.

Movement in the bridge links is simplified, as the packets do not stop until reaching the entrance container in the destination section.

4.3.4 Memory of packets

For each section and associated bridge links, the model manages a list of free packets. This list provides packets for the generation and subdivision of packets, and receives packets from mergings or because of packets arriving at the end of a bridge link. The fact that the link and its bridge link share the same packet list reduces the amount of operations needed to move a packet from the link to the bridge link. Also, the decentralized management of the free packets helps to reduce computing requirements when a change in control policy has to be considered on a subnetwork.

The maximum number of packets available is controlled independently for each link and for each bridge link. The calculation uses a heuristic based on the queue discharge:

given nVehDisl, the maximum number of vehicles discharged —in one delta— from a stopped packet in a link, we calculate the number of packets with nVehDisl vehicles that can be positioned along the link, one beside the other. This number is then divided by 1.5 in the links (but not in the bridge links), because of the distance between packets on the link and the mergings (packets with less than nVehDisl tend to merge quite early unless there is a reasonable distance between them). On the bridge links, one more often finds small packets due to small turning percentages and, therefore, the calculated number is not divided. Finally, a minimum number of available packets is required in order to assure validity the event of extremely short links and bridge links: two in links and one in bridge links. The calculation in the links is:

$$\text{max\_packets} = \text{maximum}(2,\ Ll / (1.5 * nVehDisl * Lvl))$$

In the bridge links:

$$\text{max\_packets} = \text{maximum}(1,\ Lb / (nVehDisl * Lvb))$$

where, Ll: link length.
Lb: bridge length.
nVehDisl: number of vehicles discharged in the gradual start in the link, in one delta. This is a parameter, usually around 1.0 * Wl.

4.4 Free space within a section

The free space for vehicles to enter inside a section, SP, is calculated after the first step of the packet updating and it is measured in number of vehicles that can enter that section. A fluidity coefficient is applied in order to reduce the free space when the entrance to a link is blocked. The algorithm is:

```
accum_blocked = 0.; accum_notblocked = 0.;  SP = 0.
for each link in the section
            /* this condition changes for the predicted free space */
    if there are packets then
            LA = length up to last packet
    else
            LA = Ll
    endif
            /* limitation due to the Δt interval and the maximum speed */
    LA = minimum(LA, Vsmax * Δt)
            /* free space in the link without considering the blockages */
    link.SP = LA / Lvl
            /* accumulates the available space */
    SP = SP + link.SP
            /* accumulates for the fluidity coefficient */
    if link.SP < 0. then
```

```
            accum_blocked = acum_blocked + Wl * Pt
        else
            accum_notblocked = acum_notblocked + Wl * Pt
        endif
    endfor

        /* calculates a fluidity coefficient in the section entrance */
    μ = 1. - accum_blocked / (accum_blocked + accum_notblocked)
        /* applies the fluidity coefficient to the available space in each link. It will
        * be used in the second step of the updating */
    for each link of the section
        link.SP = μ * link.SP
    endfor

        /* applies the fluidity coefficient and substracts the vehicles which are
         * already in the entrance container */
    SP = μ * SP - container_veh
```

where,	LA: length available in a link
	Pt: turning ratio of a link in a section
	μ: fluidity coefficient

The evaluation of the free space accounts, partially, for the dynamic behavior of at least some drivers when varying their initial turning decision if the entrance to the destination section is blocked. This gives better results in highly connected networks.

The sections are sorted in such a way that, when executing the first updating step in a section, all the destination sections have already been updated and the free space is known. Unfortunately, such sorting is not always possible, in which case we need to predict the free space temporally. After the first updating step, the predictions are replaced by the exact values (according to our model).

The prediction of the available space is based on the algorithm used for the exact evaluation, where the already marked block changes to:

```
if there are packets and the last one is stopped then
        LA = length up to last packet
else
        LA = Ll
endif
```

Thus, the algorithm assumes that if the last packet in the link is stopped, its back end will not start moving in the next Δt interval. If the last packet is moving, then it assumes than there will be unlimited available space. Because of this inaccuracy, the free space in a section can, momentarily, take a negative value.

Great care is required in the details. For example, if a portion of the packet has entered the next section, the rest of the packet does not stop but only slows down. In the next Δt interval, the packet will stop if no other portion can exit. This accounts for the inaccuracies in this prediction of free space. Ultimately, these errors will be less important with a short Δt time interval.

4.5 Link width

The width of a link —capacity— is measured in terms of the number of lanes it represents. In general, it will be a fractional number. As said earlier, a section has as many links as possible turnings at its exit. Within a section, we say that two links intersect when they share a lane. A link can intersect above and/or below. Figure 4.10 shows a section and its link modeling, where the link 2 intersects below with link 1 and above with link 3.

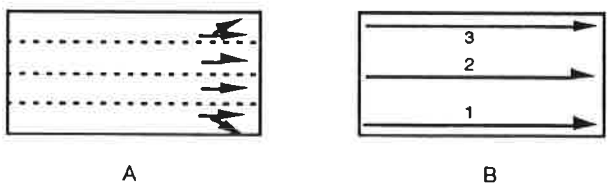


Fig. 4.10 A section (A) and its link modeling (B).

The width of a link is divided into two terms: the first one is fixed and belongs to the lanes where the link does not intersect with other links; the second term, which is variable, is divided into two parts, one for the lower intersection and the other for the upper intersection (if these intersections exist). As a result, the width of a link with intersections can vary between certain limits, restricted by the minimum width that a link can occupy in a lane, which is taken as:

min_width = minimum(1./Ninterlane, 1./6.)

where Ninterlane is the number of links which intersect in this lane. For example, in Figure 4.10, the width of Link 1 could vary between 0.166 and 0.834 and that of Link 2 between 2.336 and 3.638. This results from the fact that, in the first lane, the minimum width of both Links 1 and 2 is 1/6 = 0.166, and this restricts the maximum of Link 1 to 1 - 0.166 = 0.834.

Besides the current link width, we distinguish the width that the link 'desires', Wldes, related to the percentage (turning percentage) of vehicles in this link:

Wldes = Pt * Nlanes

where Nlanes is the number of lanes in the section.

The idea is, in the link intersections, to favour the links with greater turning percentages, i.e., containing more vehicles: that is, to augment their width against the width of adjacent links. The resulting problem can be formulated as the following minimization:

$$[\text{MIN}] \sum_{i:1..N} (f_i + a_i + b_i - \text{Wldes}_i)^2$$

with the constraints:

$$\sum_{i:1..N} (f_i + a_i + b_i) = \text{Nlanes}$$

and a bunch of constraints deriving from the fact that a part of each link cannot be moved from a certain lane:

$a_i + b_j = 1$, in lanes where two links have a variable width.

$\sum_{k:1..L} (f_k + b_k) = 1$, in lanes where l links are totally included in the same lane.

...

bounds:

$a_{min} <= a_i <= a_{max}$

$b_{min} <= b_i <= b_{max}$, $i= 1 .. N$

where, f_i: fixed width.

a_i: width in the lower intersection.

b_i: width in the upper intersection.

Wldes: desired width.

N: number of links in the section.

Nlanes: number of lanes in the section.

The problem has a small dimension and the quadratic objective function could be approximated by the linear $| Fi + Ai + Bi - Di |$. However, it should be solved for each section and, due to the treatment of the blocked exits, in each Δt time interval. We have, therefore, preferred an approximation where each link in the section makes a 'width request' to the intersection lane (upper and/or lower). In the event of a link intersecting on both sides, the width request on each side is:

width_request = (Wldes - f) / 2.

After receiving all the width requests, each intersecting lane shares its width (1.0) among the links, in an approximately proportional way. A link requesting a negative width may receive the minimum width or even augment its width if there are not enough positive width requests in that lane.

The link width, Wl, determines both the nVehDisl vehicles discharged from the queue and the vehicle length in the link, Lvl:

nVehDisl = Wl * nVehDis_lane

Lvl = Lvs / Wl

When advancing a time interval with these new values, the packets in a link that 'narrows' contain fewer vehicles but continue to occupy the same length. This effect is particularly useful when a link exit is blocked.

4.6 Turning speed

Modeling the behavior of the vehicles when turning into another section and when inside the junction is particularly important in densely signalized areas with short sections. Also, it makes for a more accurate give-way modeling and spill-back detection.

None of the previously mentioned systems (OPAC, ACTS, SAST, PRODYN, SCOOT, ...) model explicitly the behavior of the vehicles inside a junction. In general, in the macroscopic models, the vehicle goes from the end of a section directly to the destination section, ignoring the area inside the junction. So far, only microscopic simulators have attempted this level of detail. For example, in SIMRO (Chin 1985), a microscopic simulator for roundabouts, the user specifies the radius of each turn, and the maximum turning speed, Vtmax, is calculated as:

$$Vtmax = \sqrt{B0 + B1 * r}$$

where r is the turning radius and B0, B1 are constants evaluated by regression.

The AIMSUN microscopic simulator evaluates a maximum link exit speed using the angle that the link has with the bridge link. The entrance speed to the link is evaluated similarly. Inside the bridge link, the vehicles use a car-following algorithm to vary their speed. The user specifies the turning angle and the coefficients c1 and c2 of the linear function used to calculate the speed depending on the angle.

In PACKSIM, we use bridge links to model the internal space in a junction. To reduce computing time, in the bridge link the vehicles use a simplified packet-following algorithm, as has been discussed in **4.3.1.** We are concerned with reducing the amount of data the user has to specify. For this reason, we do not employ the SIMRO equation based on the turning radius: the evaluation of the real turning radius is not easy for the user. Also, the AIMSUN approach, although more automatizable, is not convenient due to the simplified packet-following algorithm that we use in the bridge links. Our approach in PACKSIM is to suppose a maximum turning speed as $Vtmax = f(ß,\partial,l)$

where	ß:	angle (0^o, 180^o) between the origin link and the destination section.
	∂:	angle (0^o, 180^o) between the origin link and the bridge link.
	l:	length of the bridge link.

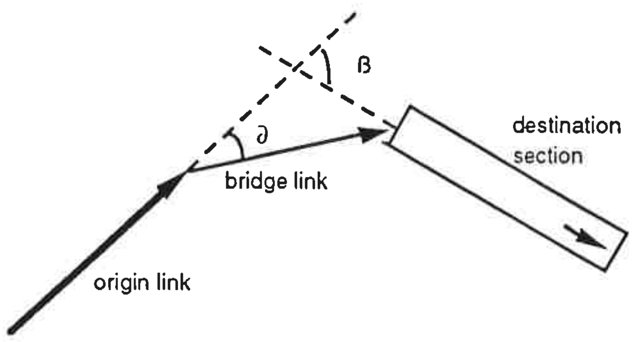


Fig. 4.11 Notation used in turnings.

as shown in Figure 4.11. Then, we consider $ß' = | ß - \partial |$, and suppose that it is possible to express the maximum turning speed as Vtmax = f(ß') + g(l), f() being a decreasing function which is approximated with n linear functions:

$$f(ß')=\begin{cases} vmax, & \text{if } ß' <= ß'min \\ f1(ß') = k11 * ß' + k21, & \text{if } ß'min < ß' <= ß'1 \\ f2(ß') = k12 * ß' + k22, & \text{if } ß'1 < ß' <= ß'2 \\ \ldots\ldots & \\ fn(ß') = k1n * ß' + k2n, & \text{if } ß'n < ß' <= ß'max \\ vmin, & \text{if } ß' > ß'max \end{cases}$$

where ßmin: minimum angle. If the angle is smaller, then the turning speed is not restricted (vmax).

ßmax: maximum angle. If the angle is bigger, then the turning speed is minimum (vmin > 0).

ß1, ß2, ...ßn: optional intermediate angles.

The user specifies (ßmin, vmax), (ßmax, vmin), and, optionally, n intermediate coordinates, from which the coefficients k11, k21, k12, ... k2n are evaluated (Figure 4.12).

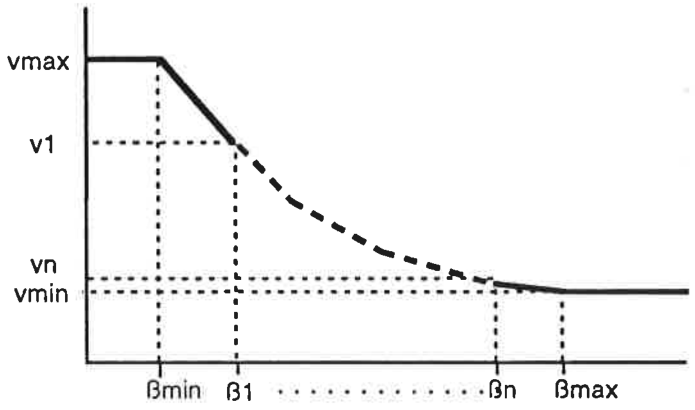


Fig. 4.12 Example of f() function.

The distance function, g(l) is expressed as g(l) = k3 * l, where k3 > 0 is a coefficient specified by the user. Finally, the maximum turning speed is:

$$Vtmax = minimum(Vtmax, vmax_{transition})$$

where $vmax_{transition}$ is a limit on the transition speed between the origin and destination sections, due to the maximum speed in these sections.

$$vmax_{transition} = (Vsmax_{origin} * Nlanes_{origin} + Vsmax_{dest} * Nlanes_{dest}) / (Nlanes_{origin} + Nlanes_{dest})$$

4.7 Queue propagation between sections

In congested situations, there is a need to model the upward propagation of queues between sections due to the lack of available free space in the destination section. When vehicles at the end of a section cannot turn to exit, they accumulate upwards in the section until, eventually, they block the entire entrance to the section. A demand-responsive system should recognize when this phenomenon is going to happen (or is currently happening) and consequently change the control.

We distinguish two types of queue propagation between sections: at the entrance to a section and at the exit from a link.

4.7.1 Queue propagation at the entrance to a section

The entrance of vehicles into a section accounts, through the fluidity coefficient, for the lack of free space:

$$\mu = 1. - \sum_{\text{entrance blocked}} (Wl * Pt) \; / \sum_{\text{all links}} (Wl * Pt)$$

This coefficient is used in the algorithm to reduce the free space, SP, when there is no free space for vehicles to enter certain links. Only in the event of none of the links having free space, is free space in a section null. This is a rather optimistic consideration of the dynamic behaviour of drivers and the connectivity of the network, but it could easily be changed to accommodate different behaviors. Furthermore, in most sections, there will be a feedback to correct the inaccuracies of the model.

In the second updating step, the vehicles in the entrance container of a section disperse into the links:

```
SP_initial = SP_current + nVeh_container
if SP_initial > 0. then
        nVeh_enter = nVeh_container * (link.SP_initial / SP_initial)
else
        nVeh_enter = 0.
endif
```

Figure 4.13 shows a example of a section where the third (upper) link has its entrance blocked. This does not impede the entrance of vehicles into the other links, but only reduces it (in the figure, the dark packets are stopped).

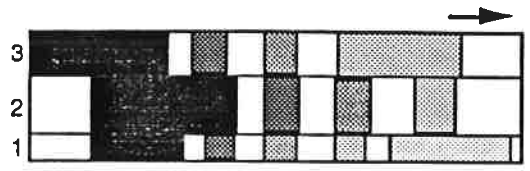


Fig. 4.13 Section where the upper lane has its entrance blocked.

4.7.2 Queue propagation at the exit from a section

A link has its exit blocked if, while having right of way, there are vehicles which cannot exit. This can happen either because of give-ways or due to the lack of free space in the destination section.

As a first effect in the real world, there is an accumulation of vehicles from the end towards the beginning of these lanes, occupying all the possible origin lanes of the blocked turn. We model this phenomenon by increasing the desired width, Wldes, of the link concerned; hence, it will tend to increase its width: stopped packets decrease their length while containing the same number of vehicles. This width increase is made by decreasing the width of other links that intersect in the lanes concerned. From then on, the model continues the updating of packets using the new queue discharge, vehicle length and detector percentage.

The width increase is made gradually:

$$\text{Wldes} = \text{Wldes} + \text{Pt} * \text{Nlanes}$$

where Nlanes: the number of lanes the section contains,
Pt: turning percentage of a link in the section.

When the exit of the link is again possible, the desired width is made to gradually decrease towards the initial width:

$$\text{Wldes} = \text{maximum}(\text{Pt} * \text{Nlanes},\ \text{Wldes} - \text{Pt} * \text{Nlanes})$$

By gradually varying the desired width, we try to avoid the sudden width changes which could arise when the exit blocks and unblocks fast and continuosly.

4.8 Junctions

As stated above, we distinguish three types of junctions according to the applied control policy: unsignalized, fixed signalized and demand-responsive signalized. Unsignalized junctions are modeled with one stage in which all streams have right of way. The user specifies a list of priorities when give-ways between streams arise.

In signalized junctions, a stage-based approach is applied, in which the cycle of the junction is divided into stages and interstages, each one having a particular set of streams with right of way. "A stream of traffic at a signal-controlled junction comprises either one or several adjacent lanes of traffic that behave as a single queue, independent of the behaviour of traffic in any other adjacent lane" (Allsop 1991). In our modeling, a stream is associated with a possible turn at the end of a section.

The control of a junction is specified by the user as:

- the offset of the cycle, in number of seconds in relation to the time the simulation starts.
- the cycle as a sequence of stages and interstages.

For each stage and interstage, the user specifies initial duration and groups of streams with right of way. The difference between a stage and an interstage is that, in demand-responsive control, the stage has a variable duration and consequently the user has to specify its minimum and maximum duration. The duration of the amber period and the end of a green period can be included at the end of the stage or at the beginning of the interstage period: the first case implies that the vehicles will use the amber period, unlike the second.

4.8.1 Critical junctions and progression arterials

For the purpose of demand-responsive control, the user can specify critical junctions and progression arterials. A junction can both be critical and pertain to one or more progression arterials. The junctions are sorted from higher to lower priority:

1- Critical junctions: the tuning of their control has priority over all the others.
2- Progression arterials: the system seeks maximal-bandwidth control over these arterials.
3- Other junctions.

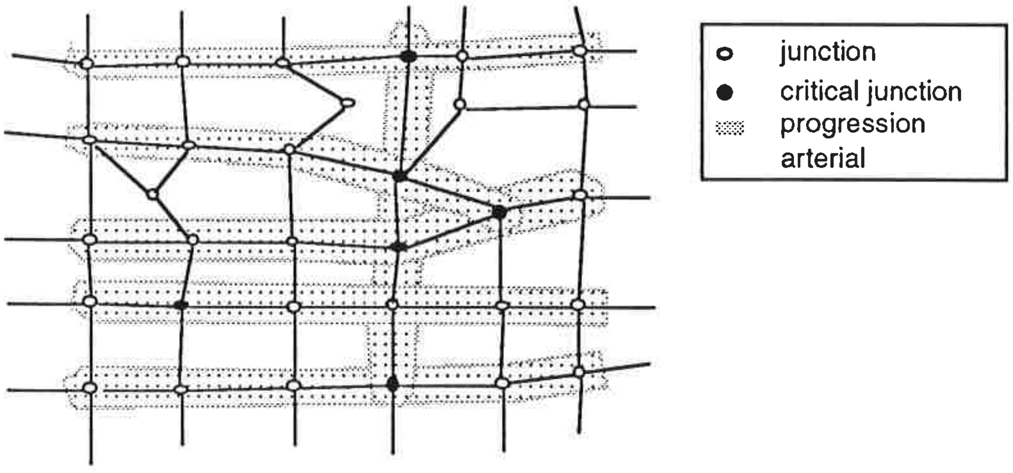


Fig. 4.14 Example of critical junctions and arterials specified on a network.

Misuse of such sets of junctions can lead to a poor performance from the demand-responsive policies. Therefore, as with fixed-time control systems, they should be used with care and, preferably, by an experienced traffic engineer.

4.9 Give-ways

A give-way occurs when the vehicles exiting a section have to give way to the vehicles exiting other sections. All sections concerned in a give-way must belong to the same junction. Give-ways are regarded as priority problems between turns (streams). For each turn, the user can specify a list of turns with higher priority. These lists are then used to sort the entrance sections to a junction before the simulation starts. Hence, when updating the vehicles in a section, the vehicles which have exited the higher-priority sections are already known. The right of way of a turn is cancelled (in that Δt) if the number of vehicles having left the higher-priority sections is larger than a given proportion of the maximum number of vehicles which could, in free conditions, exit from the give-way link:

$$nVeh\_prio > prop * Vsmax * \Delta t / Lvl$$

where: nVeh_prio: number of vehicles that have exited the higher-priority sections.
prop: proportion of give-way, specified by the user.
Lvl: average length of a vehicle in the link.
Vsmax: maximum speed in the section.

Because of grouping the vehicles in packets, the count of vehicles exiting high-priority links needs to be evaluated with some care. An exponential filter is used to obtain a more representative count:

$$n\_veh\_prio = (1. - \beta) * nVeh\_prio + f * Nveh\_prio_{old}$$

Figure 4.15 shows a junction where a merging follows a give-way: Turn 2-1 gives way to Turns 4-1 and 3-1 and merges with them in Section 1. Therefore, the user has to specify Turns 4-1 and 3-1 in the list as higher-priority turns than 2-1. In that way, the vehicles in Sections 3 and 4 will be updated before the vehicles in Section 2.

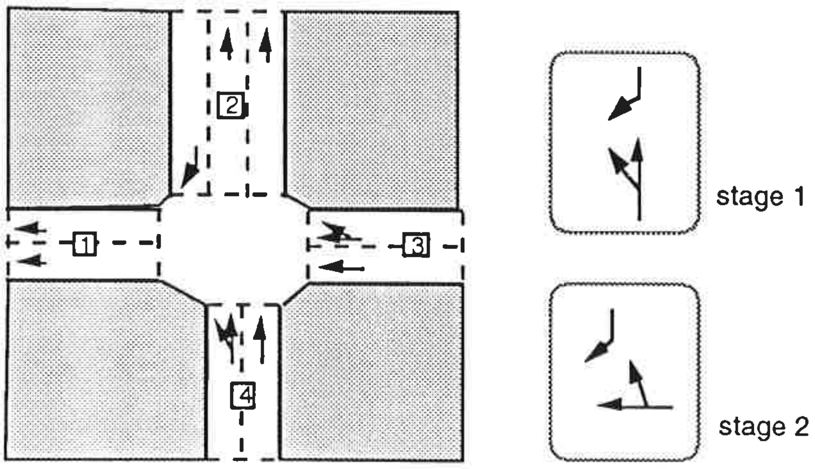


Fig. 4.15 Example of give-way.

5. Feedback

The feedback process takes as inputs the network state at the beginning of an SCC and the detector measures in each 1 .. scycle delta, outputing the state at the end of the SCC; see Figure 5.1. For each delta interval, we compare the model's measurements to the real-world measurements and then correct the packets: this may cause the generation of new packets, or the extension, partition or deletion of existing packets.

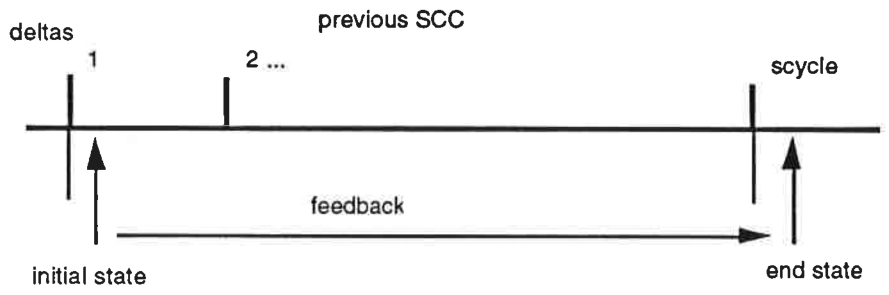


Fig. 5.1 The feedback carries out the transition between two states, distanced scycle deltas from the beginning of the previous SCC to the beginning of the current SCC.

This is done by advancing the simulation model scycle deltas, and in each delta the following actions are performed for each section and link:

- Move the packets measuring the vehicles that pass through each detector. These are called 'model measurements'.
- Compare the real-world detector measurements ($nbvehicles_R$, $speed_R$) with the model measurements ($nbvehicles_M$, $speed_M$), revising the packets accordingly.
- Update the control policy state.

Before revising the packets, the real-world measurements pass through a data-cleaning process which checks for network consistency. The measurements that do not pass this test are discarded and the revision does not take place for that detector.

When the relative difference between a real-world measurements and a model detector measurement is significant, packets in the affected links are revised. The revision takes place from the position of the detector in the link, first_pos, to last_pos evaluated as follows:

```
if next detector exists then
        last_pos = minimum(position next detector, first_pos + speed * delta)
else
        last_pos = link length
endif
```

The revision of packets is intended to add or remove vehicles between these two positions, so as to closely match the real-world measurements, see Figure 5.2. To add vehicles, free spaces between the two positions are filled; this may cause the creation of new packets, or the joining or extension of existing ones. Removing vehicles between these two positions can reduce or completely remove existing packets.

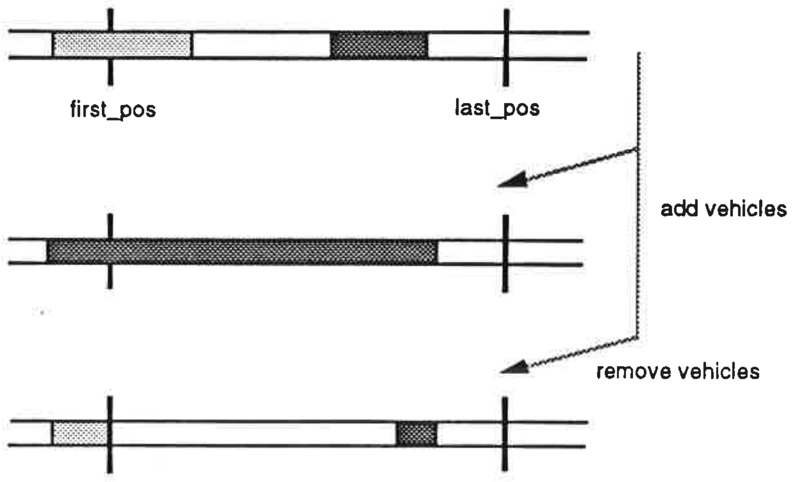


Fig. 5.2 Adding or removing vehicles between two positions in the feedback.

5.1 Detectors and links

CARS allows a section to have an arbritary number of detectors, each one being positioned on the entire section width or only on some lanes of the section. Because PACKSIM partitions the section width in one or more links of dynamically varying width, this means that a detector can be positioned on more than one link, covering all or part of their widths; see Figure 5.3. We deal with these situations by assigning to each link a fraction of the detectors positioned on it, link_width / detector_width, where both widths are measured in number of lanes, and the link width is a floating number. For example, the fractions in Figure 5.4 result from the cases in Figure 5.3. These fractions are initialized at the beginning of the execution and need to be recalculated for a section each time the link widths vary. Then, for each link, the measurements provided by the detectors are multiplied by these fractions.

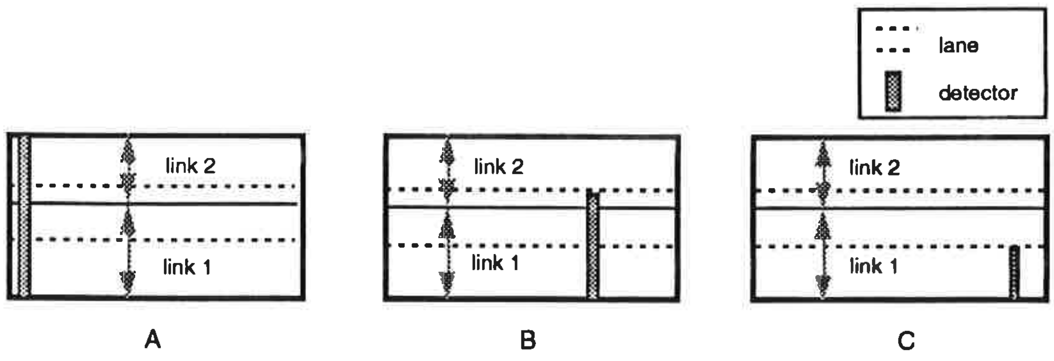


Fig. 5.3 Examples of detector positioning on a three-lane section modeled with two links. Case A: positioned on the whole section width and two links. Case B: positioned on one link and a fraction of the other link. Case C: positioned on a fraction of link.

	section A		section B		section C	
	link 1	link 2	link 1	link 2	link 1	link 2
fraction	0.4	0.6	0.8	0.7	1.6	none

Fig. 5.4 Detector fractions resulting from the cases in Figure 5.3.

5.2 Updating the predictors

The detector measurements described in the feedback process are also used to update the various types of predictors that are needed in the stepping ahead and planning processes. When available, real-world measurements are used; otherwise, model measurements. Vehicle arrival predictors are updated with data from a section-wide detector near the beginning of the section. Free space predictors are updated after the vehicle revision, taking the space available to admit new vehicles at the beginning of the section.

5.3 Updating the turning ratios

Turning movement ratios may vary significantly in time, partly due to changing traffic conditions, partly because of varying origin-destination of journeys. Usually, they are evaluated over long periods of time and are assumed to be invariant. But demand-responsive systems require a more accurate description of the traffic inside the intersection, particularly the disturbance caused by turning movements. This seems to be very important to effectively prevent spill-backs.

The approach taken by PRODYN (Kessaci et al. 1989) to estimate turning movement ratios uses Kalman filters. Although the authors report good results (Kessaci et al. 1992) we have chosen a different method which is made simpler by taking into account the traffic light information: by observing the vehicle arrivals at a section (see Figure 5.5), it becomes quite clear that they correlate with the sequence of stages in the upstream junction (the stage sequence is shown below the x-axis).

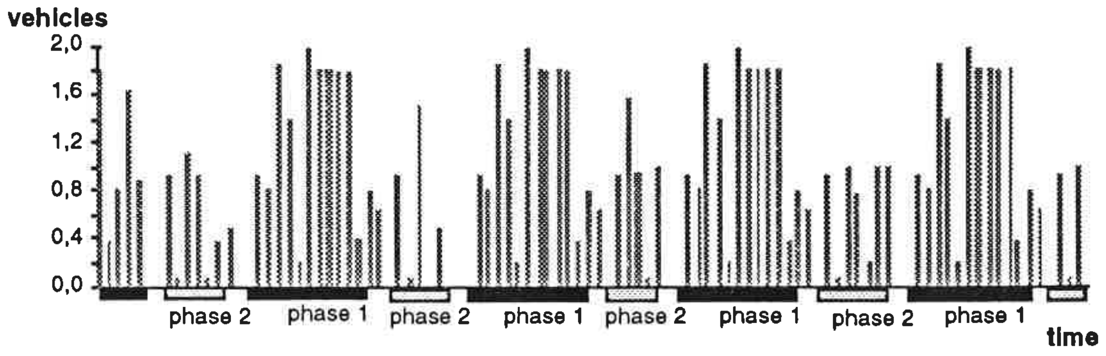


Fig. 5.5 Temporal evolution of vehicles arriving to a section in the AIMSUN microscopic simulator.

In order to evaluate the turning movement ratios of the vehicles exiting a section, a section-wide detector at the beginning of the destination sections is required; see Figure 5.6. Our approach is to classify the vehicles measured by this first detector according to the stage sequence in the upstream junction, affected by an offset; see Figure 5.7. This offset is due to the delay experienced by the vehicles when travelling through the bridge link, from the origin section to the destination section, and is evaluated using an estimated average turning speed. Of course, these accumulative measurements can only be used to update the parameters if the vehicles arriving at the destination section do not come from more than one origin section at once.

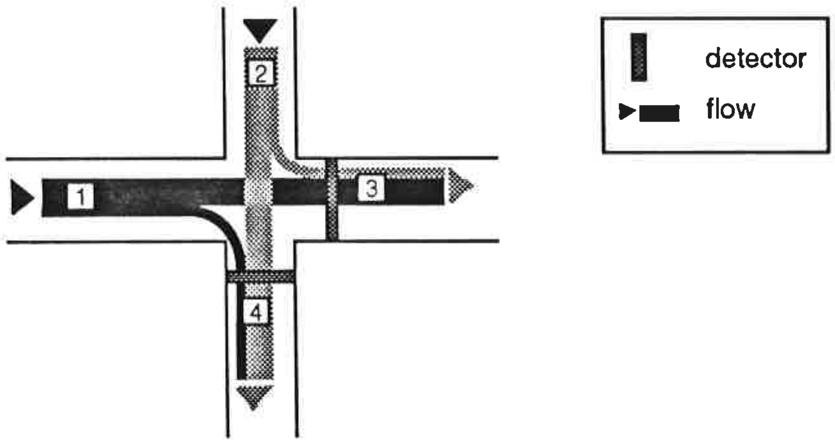


Fig. 5.6 Turning percentage supervision using detectors in the destination sections.

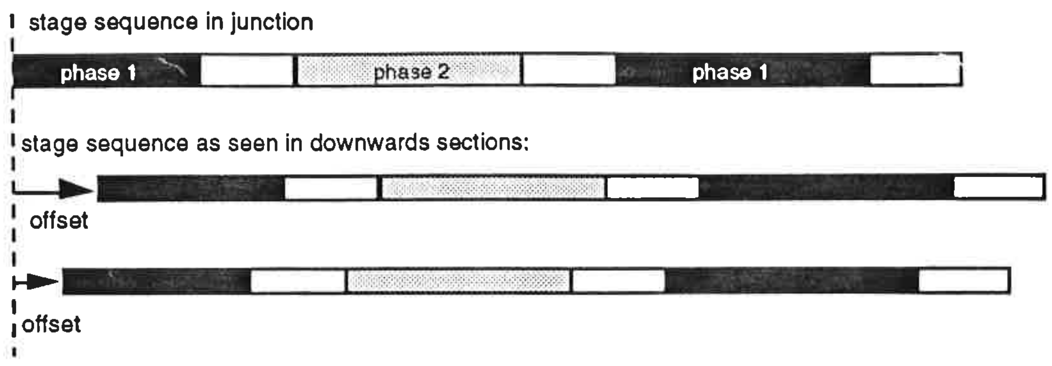

stage sequence in junction
phase 1
phase 2
phase 1
stage sequence as seen in downwards sections:
offset
offset

Fig. 5.7 Displaced stage sequences used for the classification of vehicle arrivals.

Vehicle arrivals are classified in this way, accumulated along several stage sequences, and recovered during the maintenance cycle to update the turning ratios, p_i, according to an exponential filter:

$$p_i = (1 - ß) * \text{measurement}_i + ß * \text{p\_old}_i$$

where a ß = 0.6 is generally used. In the event of detector failure and, subsequently, a lack of measurements to update certain turning percentages, turning ratios are slowly returned to their initial values:

$$p_i = (1 - ß) * \text{p\_init}_i + ß * \text{p\_old}_i$$

where a ß = 0.9 is used.

6. Stepping ahead

Before considering changes in control policy, the network state at the end of the current SCC is forecasted. This is done by advancing the simulation model scycle deltas, using vehicle arrival predictions at the network entrances and free space predictions at the network exits. The stepping ahead process is needed each time before the planning process begins, due to real-time considerations; see Figure 6.1. As the planning proposes and tests changes in control policy, a change cannot be made effective until the end of the planning process, so the first scycles deltas from the beginning of the current SCC should be repeated invariantly during each control policy test. The stepping ahead process takes these first cycles, thus alleviating computing needs later in the planning.

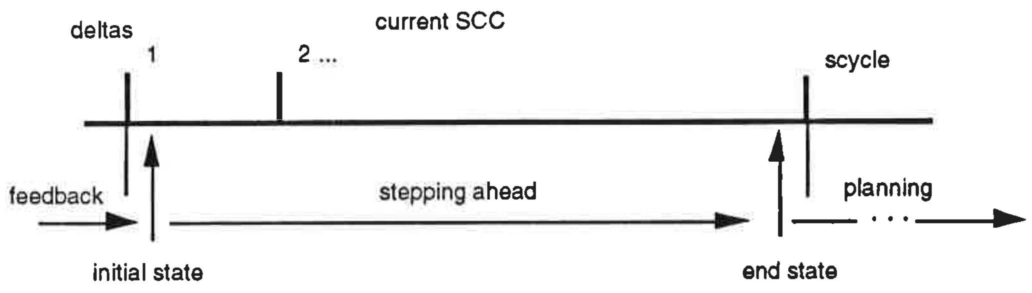


Fig. 6.1 The stepping ahead advances the state from the beginning to the end of the current SCC.

6.1 Prediction of external arrivals

In order to advance the network state scycle deltas, vehicle arrivals at the network entrances need to be predicted. These are called 'external arrivals' to distinguish them from other predictions that will be needed in the planning. The predictor has to provide vehicle arrivals as (nbvehicles, speed) tuples for the scycle deltas of the current control cycle. Because of the parameters which have been generally used (delta = 2 secs and scycle = 5 deltas, 10 secs), this is a short-term predictor.

A method to predict the number of vehicles has been selected (Stephanedes, Michalopoulos and Plum 1981) because of the satisfactory results reported, and its independence of historical information. The method is made to adapt to varying conditions by recalculating the prediction parameters at each maintenance cycle if there is a significant difference between the predicted and the real-detected measure. We consider continuous arrivals at five-minutes intervals in order to average the effects of an upstream signalized junction.

Due to the packet-following simulation model, the prediction of speeds is less important for the planning, and therefore a moving average of the last five minutes is used.

6.2 Prediction of external free space

For the same reason as for the external arrivals, there is a need to predict the free space in the sections exiting the network. The free space in a section is measured as the number of vehicles that can enter into that section (see Figure 6.2), but the measurement also accounts for the vehicles' speed, as discussed in **4.4**. Clearly, free space in an exiting section can only be evaluated if there is a detector positioned on the section; otherwise, a section exiting the network is not modeled.

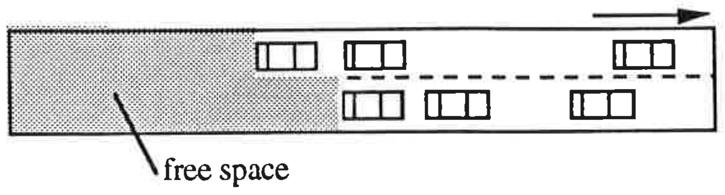


Fig. 6.2 Evaluation of free space: from the beginning of the section to the first vehicles.

A moving average of the last five minutes has been used with satisfactory results, the five-minute intervals is intended to average the effects of a possible downstream signalization while still providing a fast enough response to changes. In the event of a blockage existing that extends over more than one minute, an emergency mechanism immediately updates the free space predictor, so that the next planning process can start considering corrective action.

7. Planning

The planning process considers changing the control policy in order to minimize a certain objective function. For each demand-responsive junction, and starting from the state at the end of the stepping ahead, the process evaluates the effects of a change in control along a time horizon; see Figure 7.1. To reduce computer requirements, the planning is carried out on a subnetwork around the junction, the size of the subnetwork being variable, adapting to the current traffic conditions. Network-aware predictors provide for vehicle arrivals at the subnetwork entrances and the free space in the subnetwork exits. The control policy is configurable — progression arterials, critical junctions — as well as the objective function.

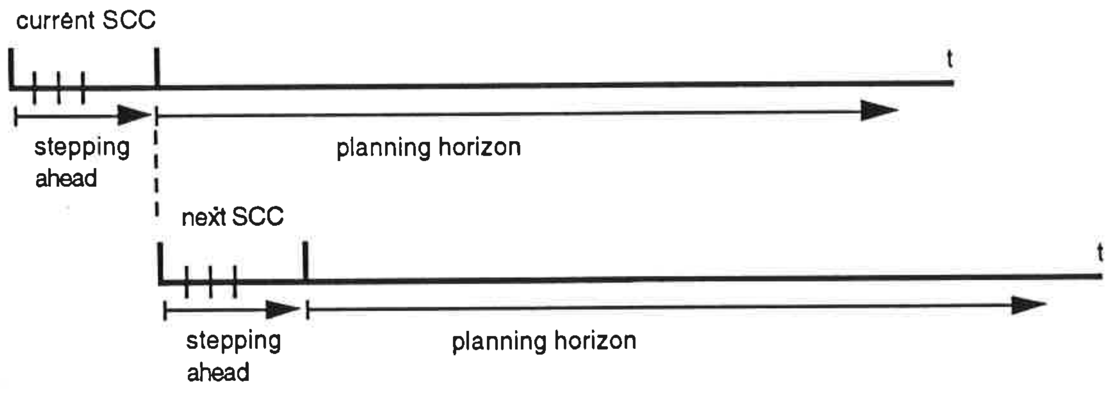


Fig. 7.1 Comparing the planning horizons between two consecutive SCC, they can be regarded as rolling in time (rolling horizon).

7.1 Adaptive Control Logics

Our approach to adaptive control in a junction is to construct, every several time intervals, a set of control policies that are small variations on current control timings, then to test their effectiveness by simulation, and choose the best timings. This simulation-based optimization is done separately in each junction. Besides, we have chosen to implement only acyclic control policies, i.e., we do not consider a common cycle length between junctions. The acyclic approach is also followed by OPAC and PRODYN, and is said, theoretically, to be capable of providing greater effectiveness and autoadaptiveness.

This section discusses how to construct the set of control policies around the current control timings, i.e., which control timings will be tested by simulation. The approach to this issue differs from one system to another: OPAC tests all feasible control policies, ACTS tests a small subset, and TOL and SAST follow the binary choice approach that consists of testing two timings — the trade-off between extending the current green duration or terminating it inmediately.

After our experiences with OPAC and an ACTS-based model discussed in PART I, we started to experiment with binary choice methods, such as the one in Figure 7.2, applied to PACKSIM in isolated junctions. We have been trying to reduce the decision-making process used in TOL and SAST by providing a supposedly more realistic simulation model, PACKSIM, longer planning horizons, and more accurate vehicle arrival predictors. Throughout the testing we have experienced the already reported binary choice tendency to produce frequent terminations of green and unestable control policies. This, coupled with the real-time restriction of needing to plan in each time interval —inherent to the binary choice method — and the difficulties to coordinate adjacent junctions in a network, has brought us to consider another approach, which is discussed below. We emphasize the real-time considerations as highly significant, in that such an approach restricts the flexibility of the system, requiring a rougher time discretization (generally 5-secs time intervals for bigger networks than a single intersection) and this brings inaccuracy both in the model and the resulting control policy, moreover limiting the minimum length of time of an interphase stage.

The approach that has finally been adopted in CARS V1 comes as a result of the following considerations:

- The previous testing of binary choice algorithms shows difficulty in coordinating junctions in a network, and imposes very limiting real-time requirements, as seen previously.
- The use of PACKSIM, a quite realistic but more computing-hungry simulation model, in CARS restricts the number of feasible changes that can be tested in each planning process, therefore making unfeasible an OPAC-like approach of testing every possible control policy, or an ACTS Branch&Bound procedure. On the other hand, PACKSIM allows the functioning of CARS in near-saturation conditions, and features flexible detector requirements and network-wide capabilities.
- The desire to find a simple but still capable algorithm.
- The search for a scalable method, this is, one that operates satisfactorily both in isolated intersections and in small and medium-sized networks.

The algorithm tests small changes in the control policy of each junction, and is based on the 1986 version of ACTS, in that it considers a tendency to phase increase or decrease, but departs from it in that:

- The time horizon is dynamically calculated each time as the current junction cycle.
- Only the first variable phase in the time horizon is adjusted.
- There is no need to consider offsets and common junction cycles to achieve coordination.
- It improves computational efficiency by considering a tendency to stage increase or decrease.

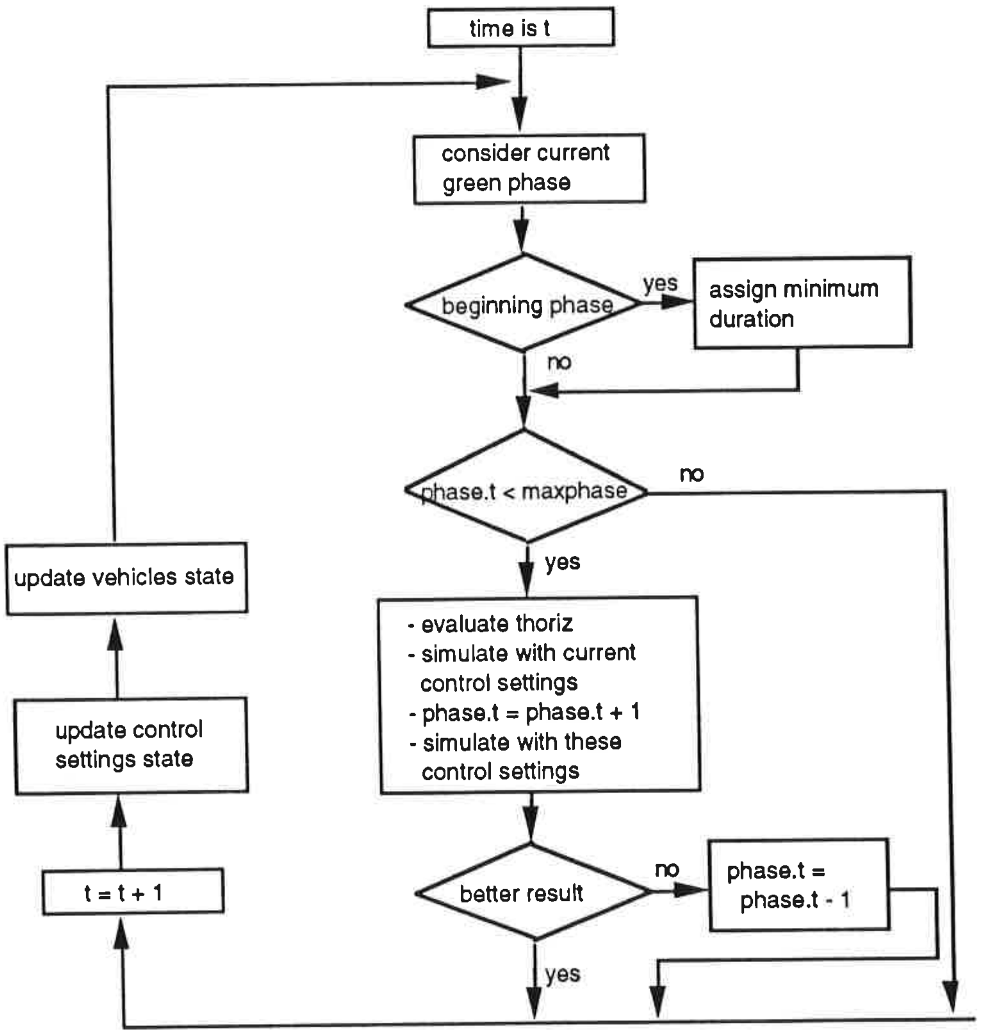


Fig. 7.2 Example of binary choice algorithm.

The algorithm description, formalized in pseudocode, is shown in Figures 7.3A and 7.3B. As will be seen in the following sections, the algorithm is associated with the use of planning subnetworks, various types of objective functions, vehicle arrivals and free space predictions, and the PACKSIM simulation model. The resulting control timings can be influenced by declaring critical junctions and progression arterials.

The use of a tendency to phase increase or decrease means that in most cases only two tests are needed per junction: one with the current control timings, and the other with the tendency-modified control timings. The use of a dynamically calculated time horizon is found to improve the overall control policy, as will be seen below.

We stress the simplicity of the algorithm, in that it obviates the use of the traditional concepts of offset, cycle, and split, and the adhoc decision-making found in some binary approach methods. Nevertheless, tests show that the system succeeds in coordinating adjacent junctions thanks to the careful combination of simple but powerful algorithms.

```
evaluate thoriz
find the first modifiable phase in thoriz
if phase can be modified  then
    simulate with current control settings
else
    exit
endif

first_iteration = TRUE
action = junction.trend

loop
  case
      action = DECREMENT:

          if not feasible  then
              if first_iteration  then
                  first_iteration = FALSE
                  loop again
              else
                  exit
              endif
          endif

          decrement phase in one delta
          simulate with these control settings
          if better result  then
              junction.trend = DECREMENT
          else
              recover original phase
              if first_iteration  then
                  first_iteration = FALSE
                  action = INCREMENT
              else
                  exit
              endif
          endif
```

Fig. 7.3A First part of the algorithm used in CARS V1 to find and test control policies in a junction.

```
    action = INCREMENT:
        if not feasible then
            if first_iteration then
                first_iteration = FALSE
                loop again
            else
                exit
            endif
        endif

        increment phase in one delta
        simulate with these control settings
        if better result then
            junction.trend = INCREMENT
        else
            recover original phase
            if first_iteration then
                first_iteration = FALSE
                action = DECREMENT
            else
                exit
            endif
        endif
    endcase
endloop
```

Fig. 7.3B Second and last part of the algorithm used in CARS V1 to find and test control policies in a junction.

An interesting issue in this algorithm is the dynamic evaluation of the time horizon along which the control policies are tested. All the reviewed rolling-horizon systems use a fixed time horizon: OPAC, ACTS, PRODYN. The length of this horizon is given by a compromise between effectivity and computational demands. However, we have found that using a time horizon dependent on the current cycle length in each junction regularly gives better results —independently of the type of objective function that is used to evaluate the performance of the system. Thus, cycle length is calculated each time before planning a junction. Figure 7.4 compares fixed-length rolling horizons with two variable-length strategies: cycle length and double cycle length, the former resulting in a consistently better performance. Therefore, we take, in each junction, a rolling horizon equal to the current cycle length.

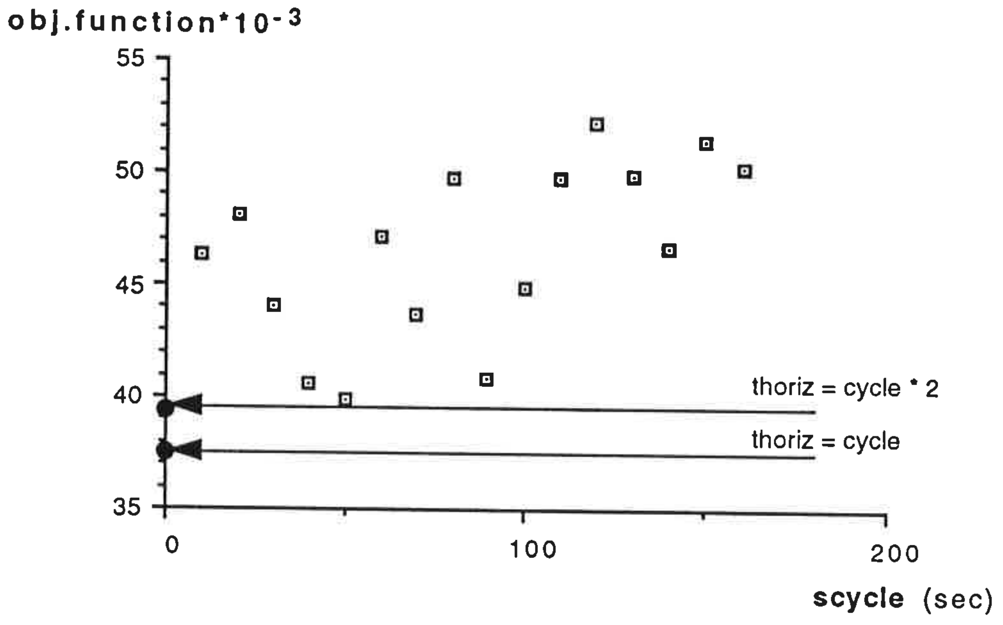


Fig. 7.4 Objective function results obtained with various fixed time horizons in a nine-junction network. The two points on the vertical axis correspond to dynamic time horizon evaluations, where 'cycle' means the cycle length in each junction. The other points correspond to fixed-length time horizons. The objective function here is the accumulated number of stopped vehicles.

7.2 Planning subnetwork

The planning task considers changing the control policy in each junction of the network. To linearize the required computing time, the evaluation of a control change is carried out on a subnetwork around the junction. Vehicle arrivals in the subnetwork and free space in the exiting sections must be predicted in each delta of the time horizon. Network-aware predictors that take advantage of knowledge of the control in the origin junctions supply vehicle arrivals and free space in the internal sections to the subnetwork; entrance and exiting sections to the network use the same predictors that were already used in the stepping ahead task, discussed in **6.1** and **6.2**. The network-aware predictors help reduce the size of the planning subnetwork around each junction. Furthermore, the subnetworks extend or reduce throughout the execution to match the current traffic conditions around the junction.

Maximum fixed subnetwork

As an initial approach to the problem, the changes in control policy in a junction can be evaluated on the entire network. Clearly, this is not feasible on large networks because it

Minimum fixed subnetwork

In an attempt for ways to reduce the planning subnetwork around a junction, we examined the ACTS approach (1983 version). ACTS considers a subnetwork around each demand-responsive junction as: the sections entering the junction and each contiguous junction, see Figure 7.5. A moving average predictor is used to predict arrivals at the entrances to this subnetwork.

In order to reduce the size of this subnetwork, special predictors have been developed that mimic the effect the adjacent junctions have upon the vehicles, i.e., the platoons formed at the beginning of each phase, and the cyclically-varying free space. These predictors adapt automatically as the control policy in these junctions change. The planning subnetwork can then reduce to the sections entering the junction, as shown in Figure 7.6. As a consequence, computing time is lineal O(n).

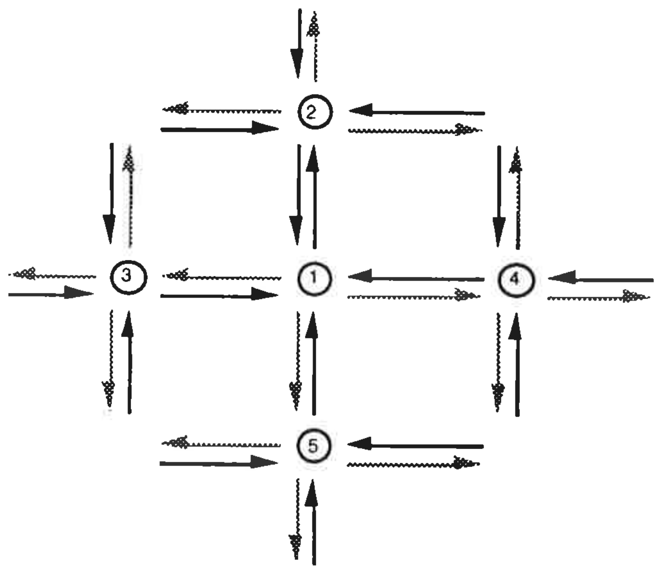


Fig. 7.5 Planning subnetwork around Junction 1, as used in ACTS. The grey links, although existing, are not considered.

Our experiences with this subnetwork show that, although the resulting control policies are less effective than with the maximum fixed network, coordination between junctions can still be achieved, and the computing time reduces enormously.

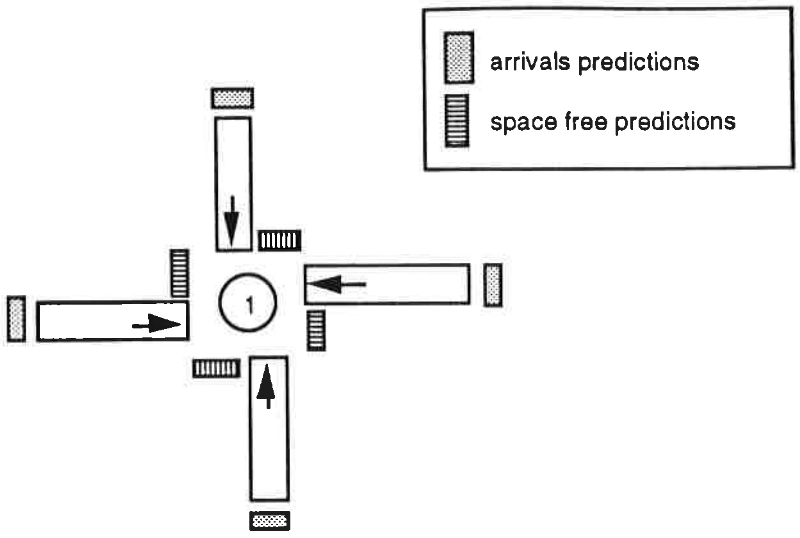


Fig. 7.6 Minimum fixed subnetwork around a junction.

Variable subnetworks

It is desirable to reduce the computing time needed for a maximum fixed size subnetwork while retaining its effectiveness. Closer examination of the influence area around the junction where vehicles are affected by a change in control policy reveals that two parts can be distinguished, the first fixed and resulting from the geometry of the network around the junction, the second variable and depending on:

- the time horizon during along which the change is evaluated.
- the network state during this time horizon (traffic conditions).
- the control policies in the network.

This variable part suggests that the influence area around a junction might need to be reconsidered each time before planning. Furthermore, dependence on neighboring control policies rules out an exact area as it would need to be changed slightly when these demand-responsive control policies change, this requiring an excesive computing time. Instead, a sort of 'average influence area' can be considered which is dependent on the time horizon and the average speed in the sections of the network, the last term accounting for both the traffic state and the control policies around the junction. This kind of subnetwork is reconsidered during the maintenance cycle, i.e., approx. every five minutes. Therefore, the variable subnetwork around a junction J_i contains all the sections from which a vehicle moving at average speed arrives at J_i in the time horizon; see Figure 7.7. The effect of the sections exiting from J_i is substituted by the predictions of free space. Vehicle arrivals in all the sections entering the subnetwork are supplied by the predictions of external or internal arrivals.

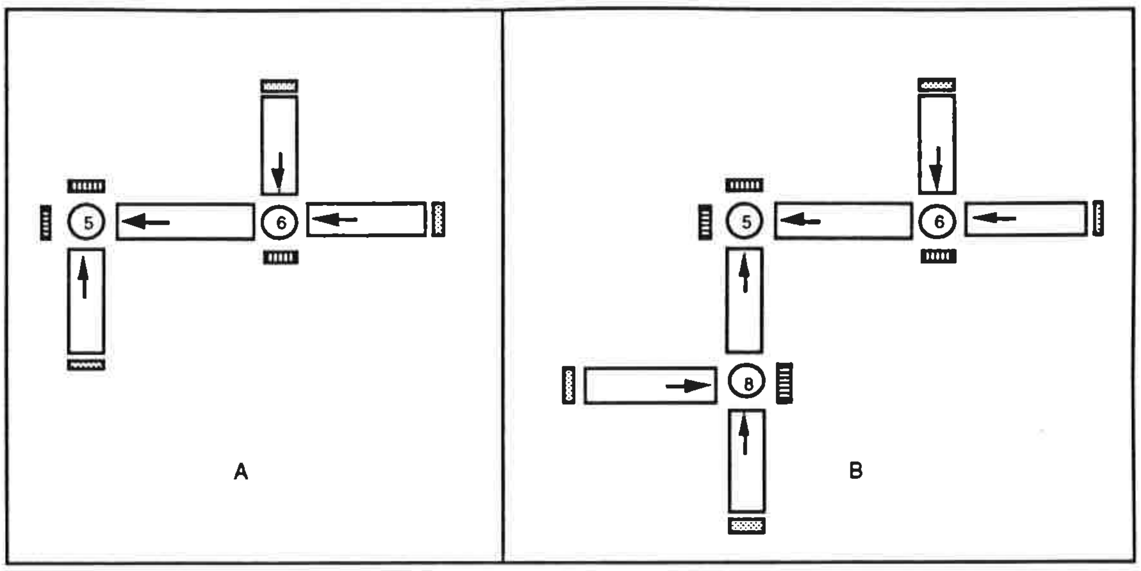


Fig. 7.7 Two examples (A and B) of planning subnetworks around Junction 5 at different times, as used with the variable-size subnetwork method.

A question that needs to be addressed when using variable-size planning subnetworks concerns its use in a real-time system. More than fast algorithms, a real-time system needs predictable computing time requirements. Therefore, to avoid timing failures, the variable-size subnetworks are initially set to their maximum size, taking as average speed the maximum allowed speed in the sections and also taking as the time horizon the maximum cycle length in the junction. Thus, computing time requirements are checked at the beginning, assuring a timing-safe operation.

The subnetworks can vary their size at each maintenance cycle, from the initial maximum size to the smallest size corresponding to near-congested conditions, where the average speed is very low. In these congested conditions, the subnetwork is the same as the minimum fixed-size subnetwork.

Nevertheless, given a high average speed or a long time horizon, the subnetwork can contain a considerable number of junctions or even the whole network. With the purpose of further reducing the planning subnetwork, we now focus on the duration of the SCC, scycle. In fact, as scycle gets smaller, changes in control policy in a junction are reconsidered more often, therefore raising an opportunity to further reduce the average influence area. Inaccuracies could be compensated, at least partially, by the greater number of times a change is reconsidered. Thus, instead of the time horizon thoriz, we take a time tsub that depends on both thoriz and scycle. On the basis of several experiments, two measurements seem to produce the best results:

$$\text{tsub}_1 = \text{minimum(scycle, thoriz)} \quad \text{and} \quad \text{tsub}_2 = \text{minimum}(2 * \text{scycle, thoriz})$$

Comparison

We implemented the above planning subnetworks in CARS, and tested their performance on small and medium-sized networks, evaluated with the internal PACKSIM simulator, obtaining similar results across all networks. Figure 7.8 shows the results obtained on a six-junction network, where the entrance flows vary from 100 to 1,000 veh/h per lane. Fixed control is compared to demand-responsive strategies: minimum fixed, maximum fixed, variable1 ($tsub_1$), and variable2 ($tsub_2$) subnetworks. A clearer representation is shown in Figure 7.9, where results from Figure 7.8 have been subtracted from those obtained with maximum fixed subnetworks. In all cases, demand-responsive strategies give better results than fixed control. Also, as expected, the maximum fixed subnetwork compares favorably with the minimum fixed subnetwork. But variable2 subnetworks consistently give still better results. This may be due to the fact that, with maximum fixed subnetworks, the vehicles are simulated through junctions that can be affected by a posterior change in their control policy during the current SCC. In variable subnetworks, however, these junctions may become entrance junctions to the network and their entrance flows be provided by predictions of external or internal arrivals which might give, hopefully, on average, more accurately predicted traffic conditions. As will be seen later, tests with the AIMSUN microscopic traffic simulator confirm these results.

Another fact that is revealed by these last figures: the smaller the scycle, the better the result. This is logical, because as scycle diminishes, so the number of times that a planning has been performed during the same length of time increases, thus resulting in a change in control policy being reconsidered more times before it is finally implemented. In general, a smaller scycle allows more demand-responsiveness, and this would require more computing power if it were not through the variable planning subnetworks; a small scycle is automatically associated to a minimum planning subnetwork and, inversely, longer scycles result in bigger planning subnetworks. Finally, the decision of which scycle length is the more suitable for operating in a given network could be given primarily by the controller's operating requirements and capabilities, more than by the available hardware power. A smaller scycle requires more strict timing constraints and an increased bandwidth communication between the controllers and the CARS system.

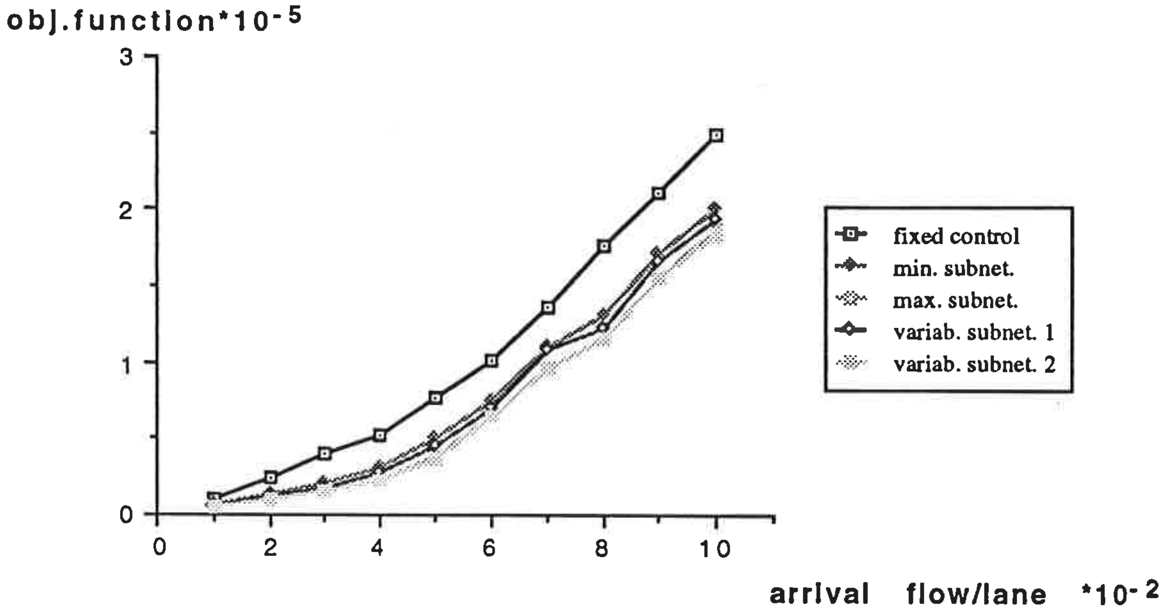


Fig. 7.8 Comparison of planning subnetworks in a six-junction network. The results of fixed control are presented as reference.

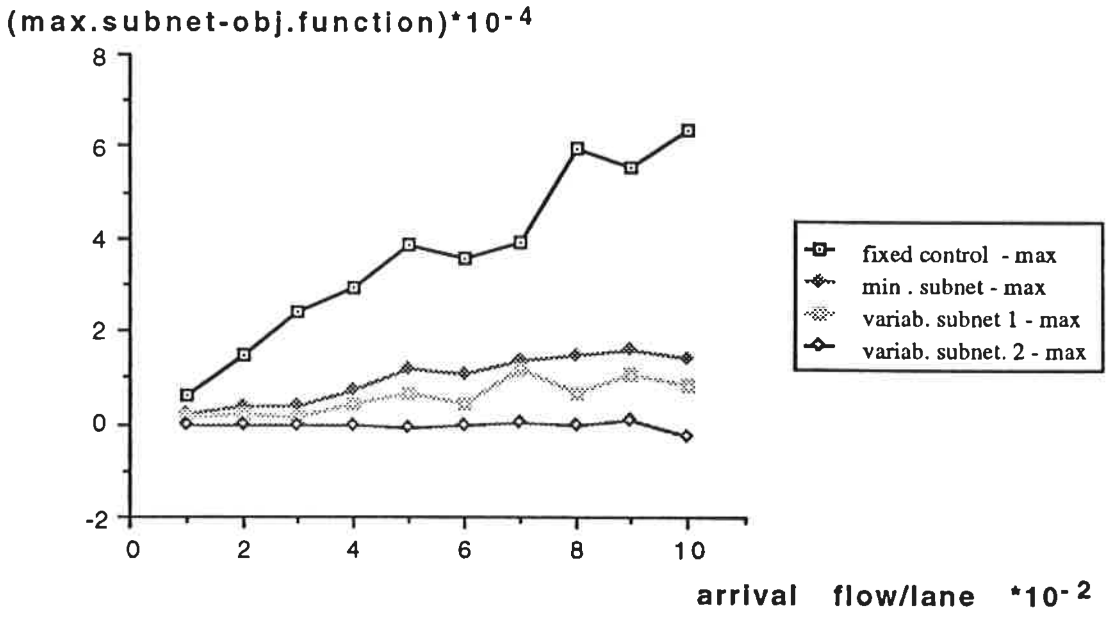


Fig. 7.9 Comparison of planning subnetworks, where the results have been subtracted from the ones obtained from the maximum fixed subnetworks.

7.3 Planning order

When planning the junctions in a network, the question arises of which order should be followed, and whether it may affect the result. To our knowledge, the only system which has been concerned with this issue is the 1983 version of ACTS. The solution that was adopted there consists of alternating 'coordination nodes' (junctions whose control is fixed during the current planning task) in the next planning, the previous coordinated nodes are planned and the others become coordination nodes. The purpose of this is to ensure that, when planning a junction, all its adjacent junctions do not change their control policy during that same planning task. This is reported to help coordinate junctions.

We have been experimenting with coordination nodes together with various types of planning subnetworks, predictors and control logic. We cannot confirm the ACTS results. Our experience shows that coordination nodes do not improve results in CARS. This difference with ACTS may be explained by our use of different methods in other areas while already performing these tests: acyclic settings, minimum and variable planning subnetworks.... Figure 7.10 shows a comparison between two 'coordination node' strategies and without them (planning all the junctions at the same time). This comparison has been performed on several arterials and networks with similar results.

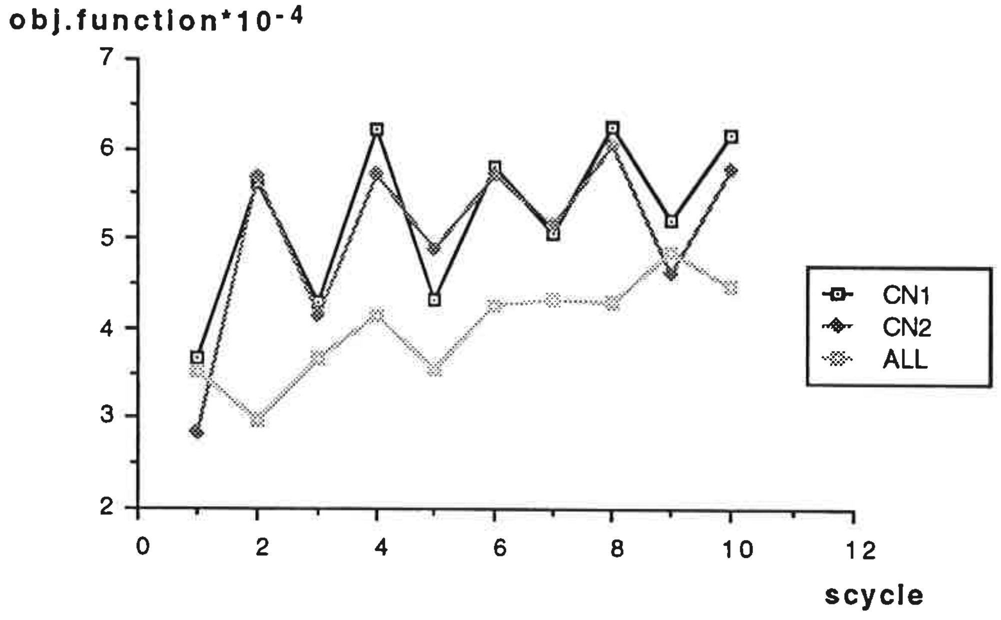


Fig. 7.10 Comparison of two coordination node strategies (CN1 and CN2, the second with reversed order), and without coordination nodes (ALL) in an arterial.

When planning a junction, control policies in adjacent junctions clearly exert an influence, either because they are part of the subnetwork, or because they influence the vehicle arrivals and space free predictors. The junction which is first planned could be said to 'impose' its control policy on the remaining junctions. After some testing we have arrived at the following satisfactory planning order:

1. Critical junctions.
2. Arterials with order alternation, that is, reversing the order from one planning to the next.
3. Remaining junctions.

7.4 Prediction of internal arrivals

We call to the vehicles entering an internal section of the network, that is, a section whose upstream junction pertains to the network, 'internal arrivals'. In contrast, 'external arrivals' are vehicles entering the network. During the planning process, predictions of internal arrivals are needed in order to supply vehicles to the entrances of planning subnetworks. These predictions must supply a (nb.vehicles, speed) tuple for each one of thoriz deltas in the time horizon, where typically delta = 2 secs. Because CARS requires thoriz to be equal to the junction cycle, usually in the range from 20 to 150 secs, these are short-term predictions. We require them to:

- Preserve the platoons' shape in order to achieve coordination between the upstream and the downstream junctions.
- Accept an upstream demand-responsive junction, that may change control timings.
- Help stabilize the changes in control policy from one SCC to the next SCC, and at the same time, provide enough accurateness to reduce the size of the planning subnetwork. This amounts to requiring a good average platoon prediction for the next thoriz deltas.
- Small computing and storage requirements.

We examine first the predictors employed in other systems. To our knowledge, SCOOT and PRODYN-D1 are the only systems whose predictors consider the platooning effect caused by an upstream junction.

SCOOT uses the data from the detector located at the section entrance to build a cyclic flow profile (CFP). This profile is used to provide the flow of vehicles moving along the section. The SCOOT papers do not detail how the CFP is constructed, at least in the readily available papers. To find this, we have to go back in time, to when Robertson (1974) explains, "(the data from the detected vehicles) is analysed to give the average flow rate during each part of the cycle time of the upstream signals. /.../ The base time is measured from the start of the cycle of the upstream signals. A useful convention is to make time zero equal to the start of the main road green. Alternatively, time zero can be made to correspond to an arbritrary datum used for all signals in the network. The base time is divided into periods of 6 sec. The length of the base is one cycle of the upstream signals. The ordinate is average flow expressed as an hourly rate. The average can be taken over any number of cycles, but 10 to 20 would be typical." Supposing that this is the kind of CFP used in SCOOT, it provides an averaged number of vehicles entering the link each 6 secs, based on the cycle of the upstream junction.

In PRODYN-D1 (Barriere et al. 1986), a completely decentralized system, each junction is controlled without requiring information from the adjacent junctions. Three methods are devised to predict vehicle arrivals:

a) A model method, which seeks in the last 50 time intervals (a time interval of 5 sec) a rectangular and cyclic signal defined by 5 parameters.
b) A recursive least squares method applied to an autoregressive model of order N.
c) A probability model stating that the flow on the detector at k is statistically dependent on the flow at k-1, ... k-4.

However, the authors recognize that, in tests on PRODYN-D1, none of these methods succeed in coordinating an arterial. Although the authors do not describe the implementation in sufficient detail, method (a) seems to search for only one cyclic bump, when in most cases several green phases give several bumps. Method (b) is, in reality, a Kalman filter which, as a least squares method, is well known to be non-robust to outlying observations, and in the Kalman filter this could make the whole process unstable (Guttman and Peña 1984). Because traffic is not a regular phenomenon, spill-backs, parking vehicles and so on can produce outliers. Moreover, although it is recommendable to apply a Kalman filter on stationary series, the authors do not mention any previous transformation to make the series stationary. The third method, (c), supposes the flow to be statistically dependent on the last four measurements, but the authors do not provide enough information or tests to judge it, nor any mention of its robustness. The next version, PRODYN-D2, included an information exchange between adjacent junctions to substitute these predictors.

Like SCOOT, CARS follows a centralized approach. Therefore, when considering the vehicle arrivals in a section, the upstream control policy is already known. The effect of this control policy on the vehicles arriving in a section has been analyzed in the AIMSUN microscopic simulator on some networks, with regard to both the number of vehicles (Figure 7.11) and their speeds (Figure 7.12). These figures show the flow of arrivals in PACKSIM, after receiving the feedback of the detector measurements. This is an important issue; because of the flexible detector requirements, we are more interested in the prediction of arrivals in the PACKSIM model plus feedback than in the arrivals of real vehicles. Of course, if each section has an entrance detector, then both arrivals will coincide after the feedback process.

Similarly to the already discussed prediction of turning movement ratios, vehicle arrivals are classified according to the current stage in the upstream junction affected by an offset due to the travel time inside the junction's area (Figures 7.13 and 7.14). Thus, contrary to SCOOT, we use an upstream cycle affected by an offset and we do not consider 6-sec. intervals inside each stage, but accumulate vehicles during the whole stage. Instead of averaging the last 10 to 20 CFPs, the predictor of arrivals in one stage is updated according to an exponential filter:

$$\hat{a}(t) = (1 - \beta) * a(t) + \beta * \hat{a}(t - 1)$$

where, $\hat{a}(t)$ is the new predictor.

â(t - 1) is the previous prcdictor.

a(t) is the average number of vehicles entering the section during that stage.

ß is the filter parameter. In general, we take ß=0.5 as a compromise between white noise elimination and excesive mitigation of the variations.

The parameter ß assumes a fixed value in the tests. Because of the satisfactory results, we have not considered the possibility of a reevaluation after a certain period. Figure 7.14 shows two examples of arrivals' prediction along a time horizon.

The same method is used for predicting the vehicles' speed, the only difference being that a(t) is the speed averaged according to the number of vehicles.

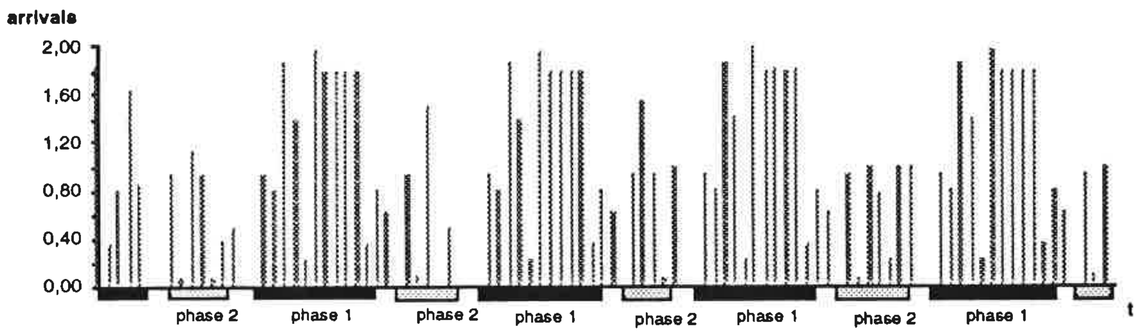


Fig. 7.11 Temporal evolution of number of vehicles entering a section, at 2-sec. intervals, according to PACKSIM. Phases in the upstream junction are shown below the x-axis.

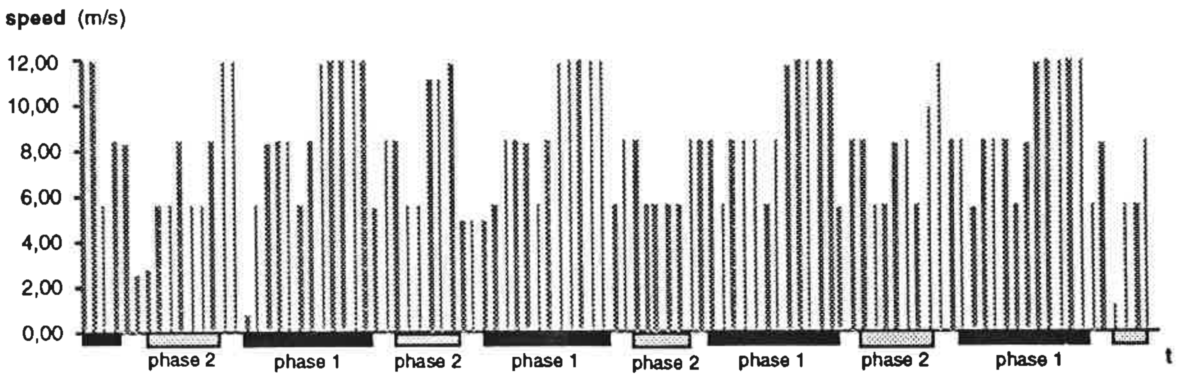


Fig. 7.12 Temporal evolution of speeds of the vehicles entering a section, at 2-sec. intervals, according to PACKSIM. Phases in the upstream junction are shown below the x-axis.

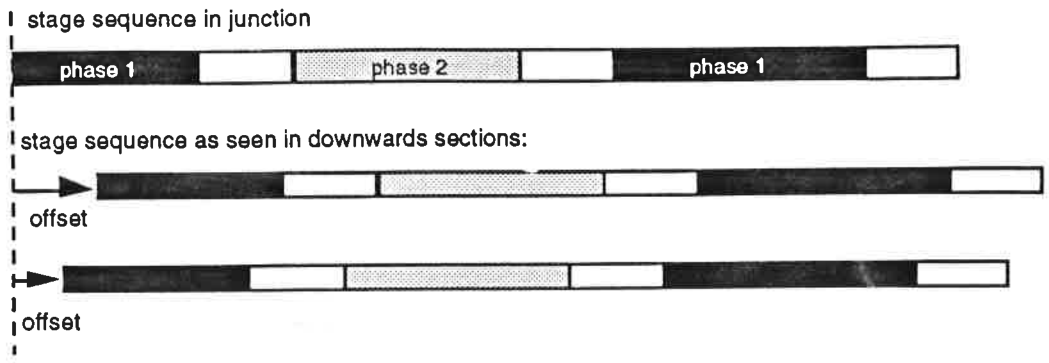

stage sequence in junction
phase 1
phase 2
phase 1
stage sequence as seen in downwards sections:
offset
offset

Fig. 7.13 Displaced stage sequences used for the classification of vehicle arrivals.

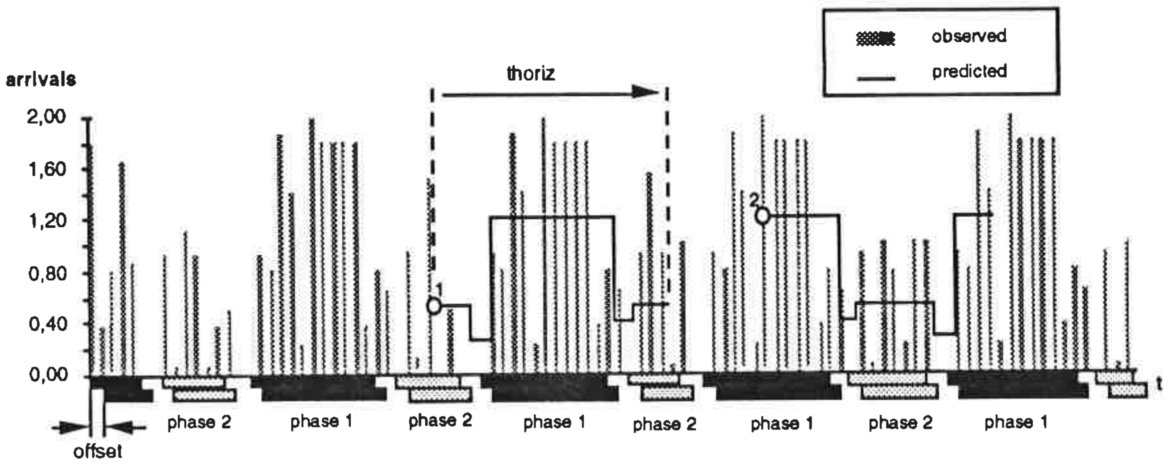


Fig. 7.14 Two examples of predicted vehicle arrivals during a junction cycle, superposed on the PACKSIM temporal evolution of number of vehicles entering a section, at 2-sec. intervals.

One could argue that SCOOT's CFP would classify the vehicles more exactly were it not for the exponential filter. Because of the truth in this, we call CARS's method the CFP-R ('R' for rough). But we note that the purpose of our arrival predictor is slightly different from SCOOT. In order to explain this point, let us consider what happens when increasing the length of a stage in the upstream junction. This would be a problem in the CFP if it needed to be immediately updated (which is the new profile in that stage?), but this is not the case, since the CFP is averaged over the last 10 to 20 cycles, thus restricting the responsive coordination with the upstream demand-responsive junction. In CFP-R the predictor automatically takes the new stage length and assumes the same flow rate which is a constant along that stage. This seems reasonable, since an increase in stage length is expected to occur as a consequence of more demand. Moreover, when used in CARS, this rough CFP-R helps in coordinating adjacent junctions.

7.5 Prediction of internal free space

We call free space in a section which is internal to the network 'internal free space'. The free space in a section is measured as the number of vehicles that can enter that section, taking into account the speed of the already existing vehicles, as stated previously. This prediction is needed in the sections exiting the planning subnetwork but that are not network exits; otherwise, they are supplied by the predictors of external free space. The predictor has to provide the free space measurement for each one of thoriz deltas in the time horizon, where typically delta = 2 secs., and thoriz=10 to 150 secs. (equal to a junction cycle). Thus, this is a short-term predictor. We require it to:

- Preserve the free space cyclic variations in order to help in coordinating the upstream and downstream junctions.
- Accept both an upstream and a downstream demand-responsive junction, varying control policy.
- Help stabilize changes in control policy and accurateness to reduce the size of the planning subnetwork. This amounts to requiring a good average free space prediction for the next thoriz deltas.
- Small computing and storage requirements.

We examine first the predictors employed in other systems. To our knowledge, only ACTS and PRODYN consider the free space in a section. But they measure it as the total space minus the space that the vehicles occupy. This means, in effect, reckoning all the vehicles grouped at the section exit. This is too simple a measurement, especially when conditions get congested. Also, although they use this measurement in their traffic model, none of these systems tell anything about predicting it while planning, in contrast to the explanations regarding arrival predictors.

Similarly to the prediction of internal arrivals, because of the flexible detector requirements, we are more interested in predicting the free space in the PACKSIM model plus feedback than in real data provided by detector measurements. Here this is more important because, generally, detectors do not provide a direct measurement of free space.

The analysis of samples of such free space measurements shows a clear periodicity. Although this could be expected to correspond to the downstream junction, we have found the PACKSIM measurements to correlate far more with the upstream junction cycle, as is shown in Figure 7.15. The method, then, is similar to the prediction of internal arrivals, classifying the free space measurements according to the current stage in the upstream junction affected by an offset due to the travel time inside the junction's area. The predictor of internal free space in one stage is updated according to an exponential filter:

$$\hat{a}(t) = (1 - \beta) * a(t) + \beta * \hat{a}(t - 1)$$

where: $\hat{a}(t)$ is the new predictor.
$\hat{a}(t - 1)$ is the previous predictor.
$a(t)$ is the average free space during that stage.

ß is the filter parameter. In general, we take the fixed value ß=0.5 as a compromise between white noise elimination and excessive mitigation of the variations.

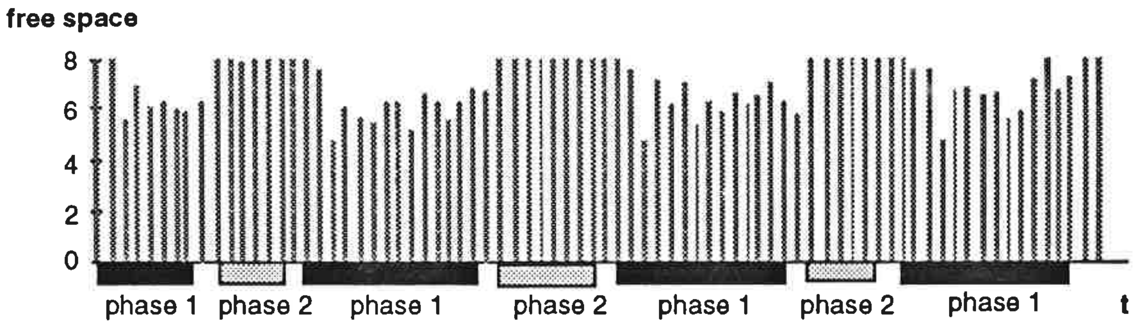


Fig. 7.15 Temporal evolution of free space in a section, at 2-sec. intervals, according to PACKSIM. Phases in the upstream junction are shown below the x-axis.

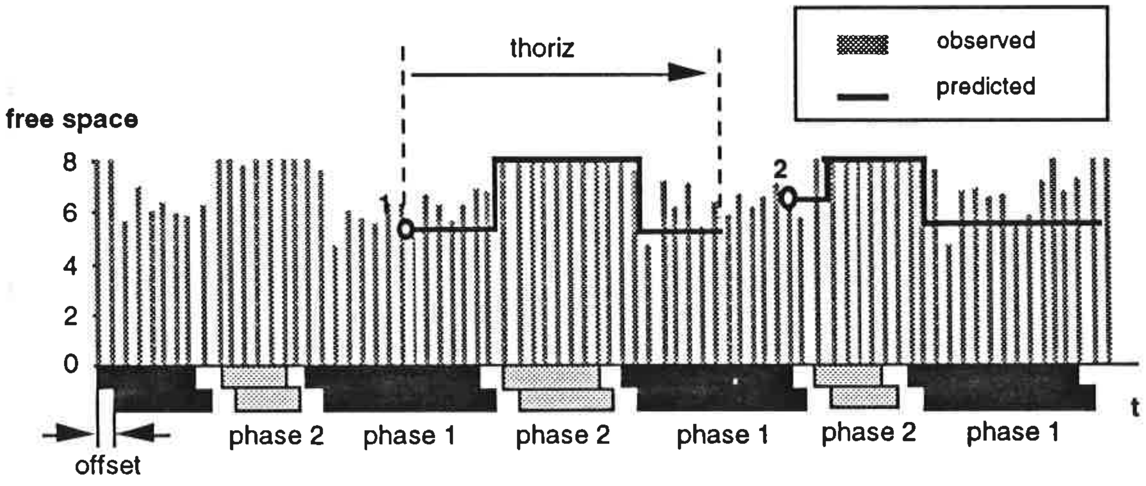


Fig. 7.16 Two examples of predicted free space during a junction cycle, superposed on the PACKSIM temporal evolution of free space in a section, at 2-sec. intervals.

Figure 7.16 shows two examples of free space prediction along a time horizon. This cyclic predictor has shown satisfactory results in the tests, as will be discussed later. Because of its structure, the predictor automatically accounts for a change in stage duration in the upstream junction.

8. Objective function

The objective function is the criterion used to evaluate and compare changes in control policy over the current control settings. Multiple types of objective functions have been used throughout the short history of these systems, as will be shown in the next section. As a decision criteria, it becomes evident that the choice of objective function influences the effectiveness of the system and the range of traffic conditions to which it can be applied.

Related to objective functions, congestion measurements are used to evaluate the degree of congestion in a traffic network. In the field of demand-responsive control, congestion measurements have often been taken to be synonymous with objective functions. However, although the aim in a demand-responsive system is usually to avoid congestion whenever possible, in some cases it may be to minimize the emission of pollutants or give priority to a certain direction of movement in the network. In the last few years there has been a surge of interest in congestion measures because of their use in expert systems for traffic control (DRIVE V1015 1990a, b).

8.1 Approach in other systems

A historical perspective of demand-responsive systems being available in PART I, only the details concerning the objective function will be presented here.

SCOOT's objective function (Hunt et al. 1981, 1982) is composed of delay, number of stops and queuing, although an exact definition is said to be part of the secrecy of the system (DRIVE V1015 1990). However, it has been reported (Robertson 1982) that the system has many similarities in structure and objectives to TRANSYT (Vincent, Mitchell and Robertson 1980), both using the same signal setting concepts and platoon dispersion model. In TRANSYT signal timings may be set to minimize a weighted sum of total delay (i.e., the sum of average queues) and total number of stops/starts:

$$PI = \sum_{i=1}^{N} \left(W \,.\, w_i d_i + \frac{K}{100} - k_i s_i \right)$$

where, N : number of links.
W : overall cost per average pcu of delay.
K : overall cost per 100 pcu stops.
w_i : delay weighting on link i.
d_i : delay on link i.
k_i : stop weighting on link i.
s_i : number of stops on link i.

This coincides with what has been outlined about the function used in SCOOT, therefore it seems very likely that both systems use a similar function.

In **SCATS** (Lowrie 1982), the criterion used for most of the plan changes relies on the degree of saturation function, which relates the number of vehicles exiting a link with the available green time.

In **OPAC** (Gartner 1982), **ACTS** (Kaltenbach et al. 1986) and **PRODYN** (Henry et al. 1983) (Barriere et al. 1986), the objective function considers the accumulated sum of vertical queues in the links over a time horizon.

UTOPIA (Donati et al. 1984) models the traffic network on two levels: global and local. In the global model, the network is constituted by storage units and the objective function depends on the number of vehicles, average speed, and saturation flow on each link. The local model is applied to each junction, and the objective function accumulates the effects of the traffic lights and the vehicles over a time horizon.

PUGWA (Schlabbach 1988) is a decentralized system that cannot be considered to have an objective function. Green times are varied according to a vehicle-actuated policy: a modified gap/traffic density control and a non-directional consideration of occupancy. The system is restricted to 15% variations from a fixed-control plan.

OFSET (List and Pond 1989) is a decentralized system which presents a choice of two approaches. In the first one, a look-up table is used to find the best match between fixed-control plans and current traffic conditions — therefore, no objective function is employed. The second approach uses a set of decision rules where the desired cycle length and green times are based on approach volumes, and offsets are varied between pre-specified values minimizing intersection delay. The authors give no further details to determine the exact nature of the objective function.

8.2 CARS approach

The objective function is used during planning to evaluate control policy changes. Several functions have been tested in a number of networks under a wide range of traffic conditions. In order to make these tests easier to perform, CARS allows the user to choose the objective function: number of stopped vehicles, queue length, queue volume factor, average speed factor, and exiting volume — all them are presented and compared here, together with other promising functions that have been discarded. The evaluation of these objective functions benefits from the realistic simulation model which allows an accurate measurement of volumes, speeds and queues.

As a way to give priority to certain sections, a penalty factor can be specified for each section; this factor multiplies the objective function value in that section. For each delta time interval in the planning horizon, the function is evaluated on the sections pertaining to the planning subnetwork, and the accumulated value is used for comparison.

Among the objective functions that have been tested, we distinguish:

8.2.1 Number of stopped vehicles

The number of stopped vehicles is evaluated in the planning as:

$$\frac{\sum_{\text{deltas}} \sum_{\text{sections}} \left(\text{penalty} * \sum_{\text{links}} \text{nb stopped veh.}\right)}{\text{thoriz}}$$

In order to compare different control changes, the final result is divided by the number of deltas in the planning horizon (thoriz). CARS uses a thoriz equal to the junction cycle.

8.2.2 Queue length

Queue length is evaluated in the planning as:

$$\frac{\sum\limits_{\text{deltas}} \sum\limits_{\text{sections}} \left(\text{penalty} * \sum\limits_{\text{links}} \text{nb stopped veh.} * \text{veh.length}\right)}{\text{thoriz}}$$

8.2.3 Queue volume factor

Queue volume (Taylor, Bell and Geary 1987) is defined as the area beneath the curve representing the back of the queue in the time-distance diagram, as shown in Figure 8.1. Instead of plain queue volume, we measure the ratio of the queue volume to the maximum possible queue volume.

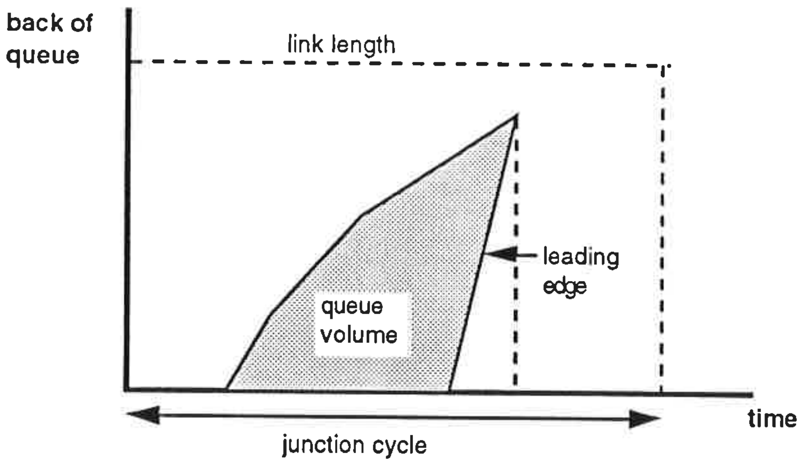


Fig. 8.1 Time-distance diagram showing queue evolution in a section.

According to the definition, queue volume is highly correlated with the maximum queue length observed during the cycle. A linear relationship is found between queue length and queue volume until the maximum queue is equal to the link length when queue volume continues to increase (DRIVE V1015 1990a). This is advantageous since it provides a time dimension reflecting the proportion of the cycle when the queue was at or close to the maximum. It has been shown (DRIVE V1015 1990a) that queue volume is also highly correlated to journey time or delay, traffic density, and vehicle time of occupancy on the link.

Queue volume (Qv) has reportedly been measured (Taylor, Bell and Geary 1987) by taking advantage of the correlation with traffic density and vehicle time of occupancy (occ), in a relation as follows:

$$Qv = \alpha + \sum_i \left(\beta_i \frac{occ_i}{flow_i}\right)$$

where i refers to the ith interval into which the junction cycle is divided (in a manner of cyclic flow profile). An additional detector is required in links longer than 80 m. Furthermore, coefficients are said to be probably dependent on each link.

In CARS, evaluation of queue volume has been made directly on the PACKSIM simulation model, thanks to the accurate queue dynamics. Each delta time interval, the number of stopped vehicles in the link is accumulated. Queue volume is evaluated at the end of the junction cycle.

A sharp distinction has been reported (Taylor, Bell and Geary 1987) between queue volumes at low and high levels of congestion, with very sudden jumps from congested to non-congested conditions, making the prediction of forthcoming congestion virtually impossible. However, queue volume as measured in PACKSIM does not display this behavior, as shown in Figures 8.2 and 8.3.

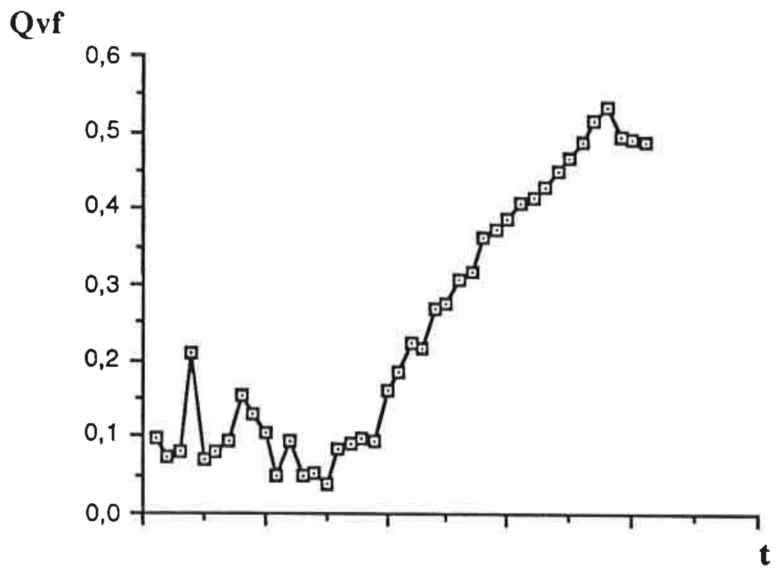


Fig. 8.2 Queue volume factor (Qvf) evolution from light to heavier flow. Measurements were collected at the end of each junction cycle.

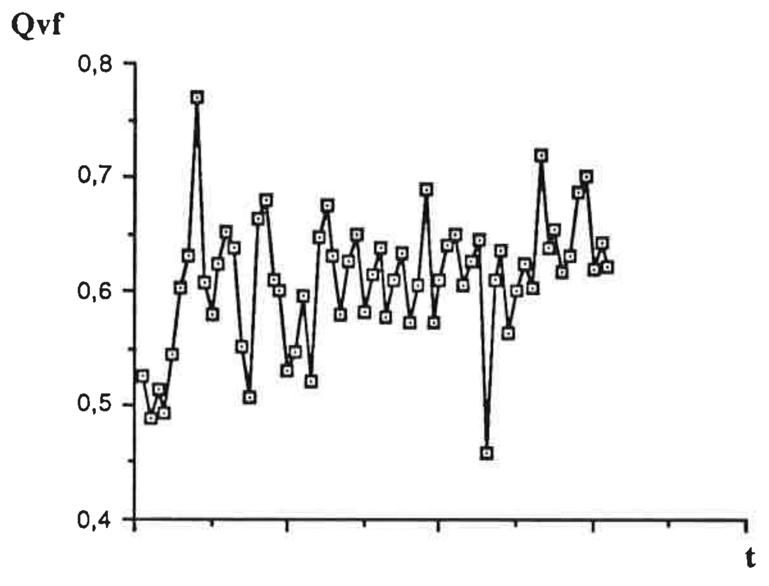


Fig. 8.3 Queue volume factor evolution in congested conditions.

8.2.4 Comfortable capacity factor

Webster and Cobbe (1966) defined capacity as the amount of traffic that can pass through a signal-controlled intersection. This depends on the green time available to traffic, the cycle time and the saturation flow of the stop line:

$$Cp = \frac{g\,S}{C}$$

where: Cp : capacity (vehicles / hour).
g : effective green time (seconds).
C : intersection cycle time (seconds).
S : saturation flow (vehicles / hour).

Webster and Cobbe assumed that traffic arrives randomly at the stop line and that once the signals turn green traffic discharges throughout the green period at a constant rate — the saturation flow. Delay is calculated as:

$$d = \frac{C\,(1 - z)^2}{2\,(1 - zx)} + \frac{x^2}{2q(1 - x)} - 0.65\left(\frac{C}{q^2}\right)^{\frac{1}{3}} x^{(2 + 5z)}$$

where: d: average delay per vehicle on the link.
C: cycle time.
z: proportion of the cycle which is effectively green for the phase under consideration.

x: degree of saturation.
q: flow.

The relationship shows that once the traffic flow approaches a 90 % degree of saturation, the delay per vehicle increases rapidly, as shown in Figure 8.4. Therefore, it was assumed that 90% of the capacity would be an appropriate measurement beyond which the delay increases rapidly with small measurements in flow. This assumption is also made by TRANSYT and SCOOT. Comfortable capacity (Cf) is defined thus:

$$Cf = 0.9\ Cp$$

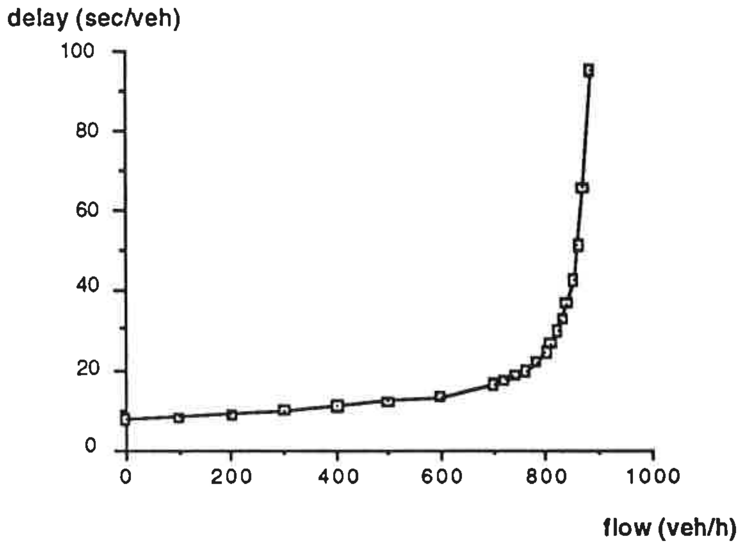


Fig. 8.4 Delay/flow curve as result of Webster and Cobbe's formula.

A potential congestion measurement, called comfortable capacity factor (CCF), was proposed (DRIVE V1015 1992a) considering the flow relationship between comfortable capacity and the actual flow, thus:

$$CCF = \frac{q}{Cf}$$

In a later experiment (DRIVE V1015 1990b) using the CCF, values in a link were plotted at five-minute intervals. The results show "a smooth CCF profile in uncongested links, whereas for congested links the CCF profile fluctuates wildly." The authors note that "this is probably a sign of the link exceeding a threshold of equilibrium, beyond which any minor incident on the link will adversely affect the flow along it, causing extreme variations in the CCF." The study proposes to recognise congestion on a link by the shape of the graph of the CCF plotted against time. This would not need the accurate measurement of saturation flow, since the absolute value of the CCF would not be critical.

We have rejected this measurement because of the difficulty in determining the degree of variation in CCF profile that indicates congestion. Furthermore, even if this could be solved, demand-responsive control needs to evaluate the effect of a change in control policy over a reduced time horizon (usually no more than one and a half minutes in CARS), whereas the CCF profile should be examined for variations over a much longer period.

8.2.5 Average speed factor

Average speed in a section has been reported to decrease when traffic gets congested (Bonvallet and Robin-Prevallee 1987). Specifically, when the number of vehicles increases, average speed decreases and flows increase slightly — or diminish in the case of oversaturation. Speed can then vary in a factor from 1 to 5, while corresponding flows may vary by less than 20%. Thus, average speed promises to be a good choice for measuring and predicting congestion.

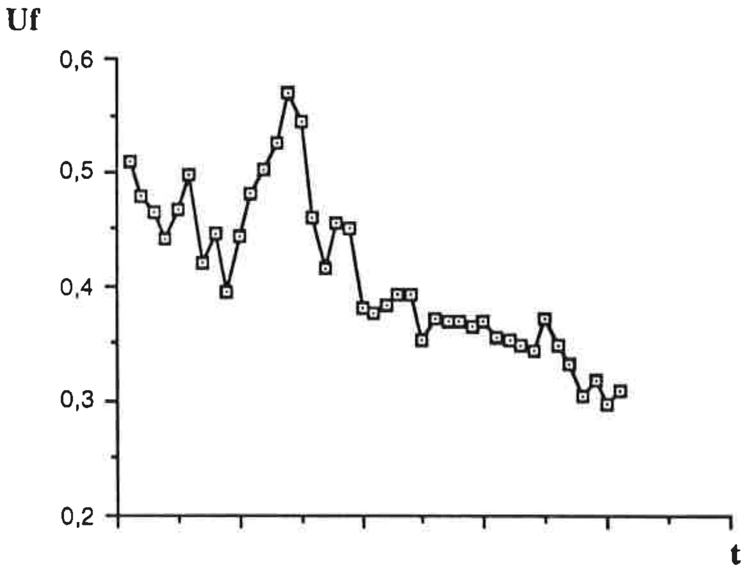


Fig. 8.5 Average speed factor (Uf) evolution from light to heavier flow. Measures were collected at the end of each junction cycle.

Average speed can be evaluated by using the correlation between travel time and vehicle occupancy time (Gault 1981), or from an estimate of the number of vehicles in circulation (Bonvallet and Robin-Prevallee 1987). In CARS, average speed is evaluated on the PACKSIM simulation model thanks to the packet-following algorithm which varies speeds according to the state of preceding vehicles. But to use it an objective function, we need a relative measurement, such as the factor average_speed / desired_speed in each section — known as the speed factor — since this provides a measurement of the quality of the service. Figures 8.5 and 8.6 show samples of speed factors showing consistent behavior according to the traffic conditions.

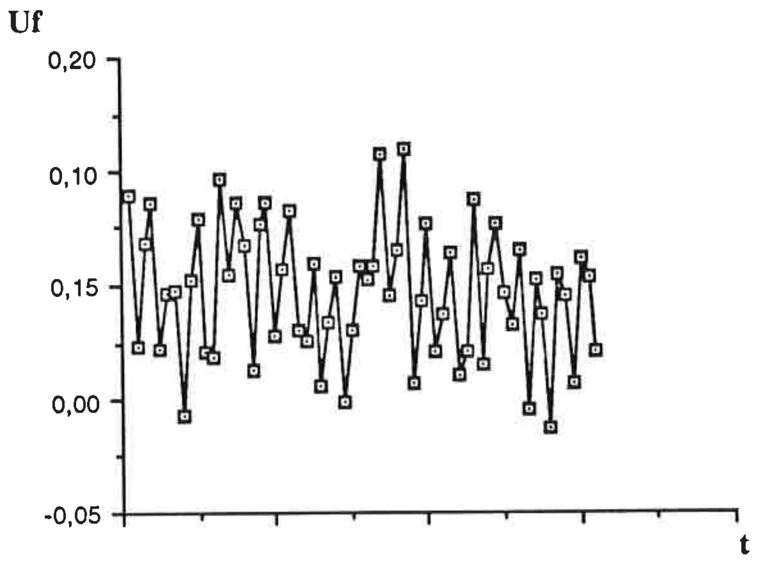


Fig. 8.6 Average speed factor evolution in congested conditions.

8.2.6 Exiting volume

The exiting volume evaluates the number of vehicles exiting each section in the planning, evaluated as:

$$\frac{\sum_{\text{deltas}} \sum_{\text{sections}} \left(\text{penalty} * \sum_{\text{links}} \text{nb exiting veh.}\right)}{\text{thoriz}}$$

8.2.7 Comparison

Comparing objective functions can be a sticky issue: the question arises of which reference function should arbitrate. Even taking this reference value from field tests or from microscopic simulation, the question remains of which criterion is the best to evaluate the results. A combination of several objective functions could be used, although the solution adopted will never be more than a convention.

Furthermore, objective functions can hardly be compared because, by definition, each one is set for a particular goal, characterizing an aspect of traffic behavior. For example, improving average speed in the network may have a negative effect on queue lengths. The goal we pursue is somewhat vague: ‘improvement in traffic conditions’, ‘to avoid congestion’, etc..

Fortunately, we have observed that employing a certain objective function in the planning decision helps reduce the value in the remaining functions. This has led us to the following strategy:

- All the objective functions are evaluated and accumulated throughout the simulation run, that is, in during the feedback task.
- The comparison is made by selecting a particular objective function to be used in the planning decision process.
- At the end of the simulation run, we take the accumulated queue length function as a reference value.

Following this method, we have carried out tests on the junction, arterial, and two networks that were shown in Figures 4.3, 4.4, 4.5, and 4.6, obtaining similar results in a wide range of traffic conditions. Figure 8.7 shows the results on the nine-junction network (figure 4.5), while Figures 8.8 and 8.9 show graphical representations. In view of these results, queue volume has been choosed as the best objective function: it becomes preferable as conditions get more congested. Under such conditions it seems that queue volume takes more advantage of the remaining free space. In light traffic conditions, queue volume results are similar to the ones obtained with number of stopped vehicles and queue lengths.

entrance flow/lane \ objective functions	fixed control	stopped vehicles	queue length	queue volume	average speed	exit volume
100	5741	10897	5836	5847	8918	6332
200	13052	25116	12414	12316	18432	12972
300	17973	40470	18533	18264	24597	19073
400	27840	53005	28042	27923	39557	28964
500	45051	76619	44610	44649	68302	49809
600	72602	100479	71783	69050	91721	74093
700	115961	135845	116090	108105	124704	120701
800	135832	175656	134073	122737	170850	159126
900	194879	210469	188725	165821	203895	200908
1000	221497	250113	214830	194370	243171	227061

Fig. 8.7 Comparison of objective functions on a network.
The values correspond to accumulated queue lengths.

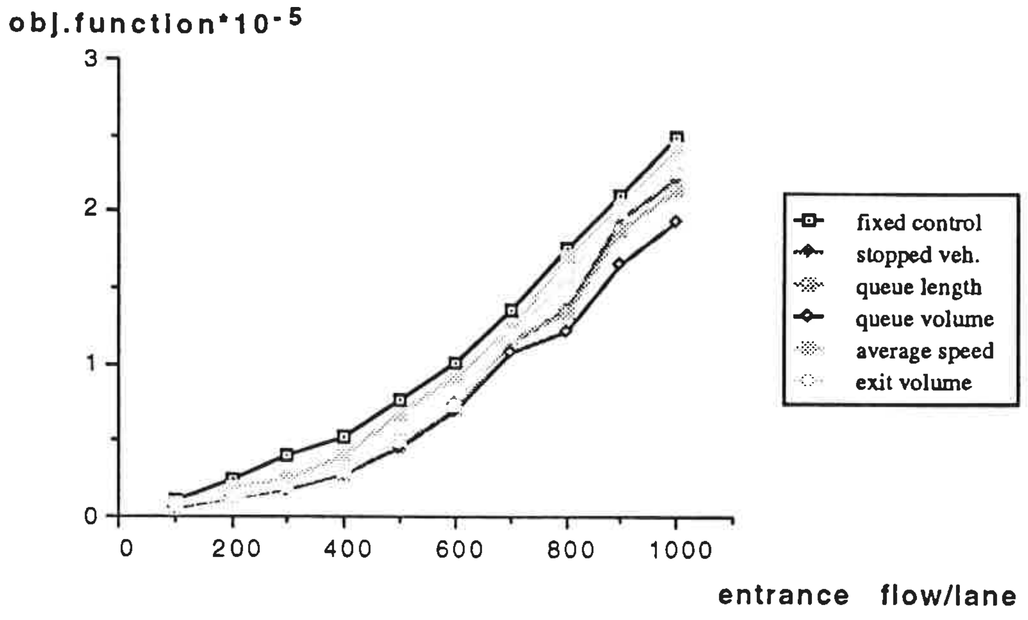


Fig. 8.8 Graphical representation of Figure 8.7 data.

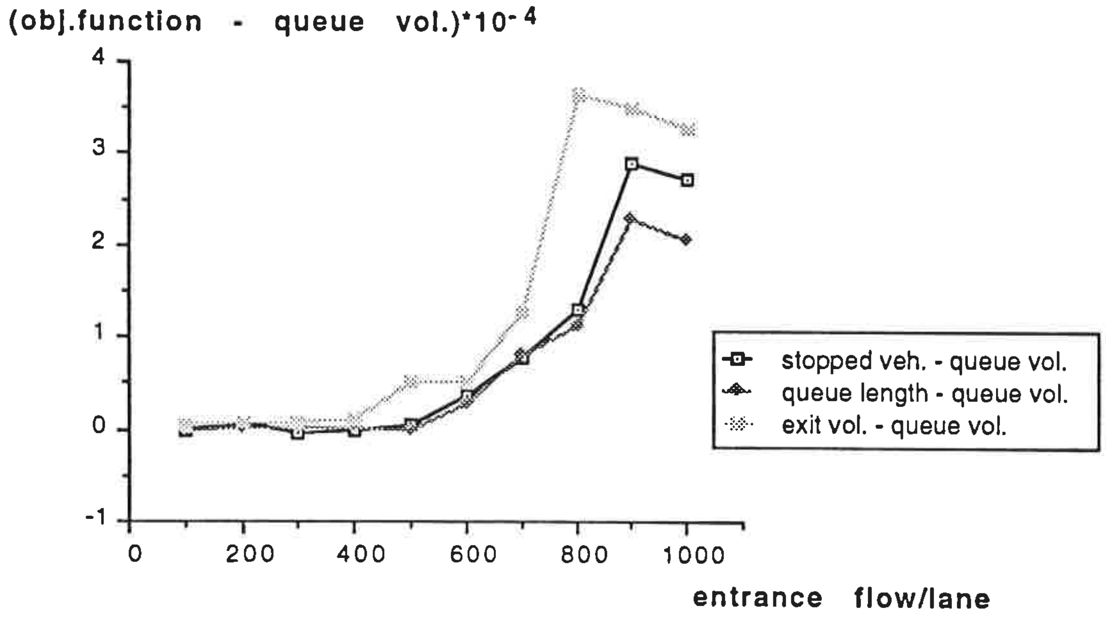


Fig. 8.9 Data from Figure 8.8 minus the corresponding queue volume.

9. Communication with controllers

The controller is the logical entity used in CARS V1 to interact with the real world. It is associated with a junction — in CARS V2 it can be associated with more than one junction — and can be connected to an arbitrary number of detectors from which it obtains traffic measurements; see Figure 9.1. CARS V1 requires real-world controllers to have the following functionality:

- To receive, from CARS, signals to end the current phase, converting them into the appropriate signals for the traffic lights on the junction approaches. Because of the delta time interval used throughout the system, phase changes can only be sent each delta seconds, delta being the time resolution of the system. All signalized junctions controlled by CARS — both fixed and demand-responsive junctions — have to be connected to a controller and receive phase endings.
- To send, to CARS, sets of measurements ($nbvehicles_R$, $speed_R$) from the detectors, each measurement corresponding to delta seconds. When requested — once during each system control cycle — the controller must send to CARS the accumulated measurements corresponding to the last scycle delta intervals.

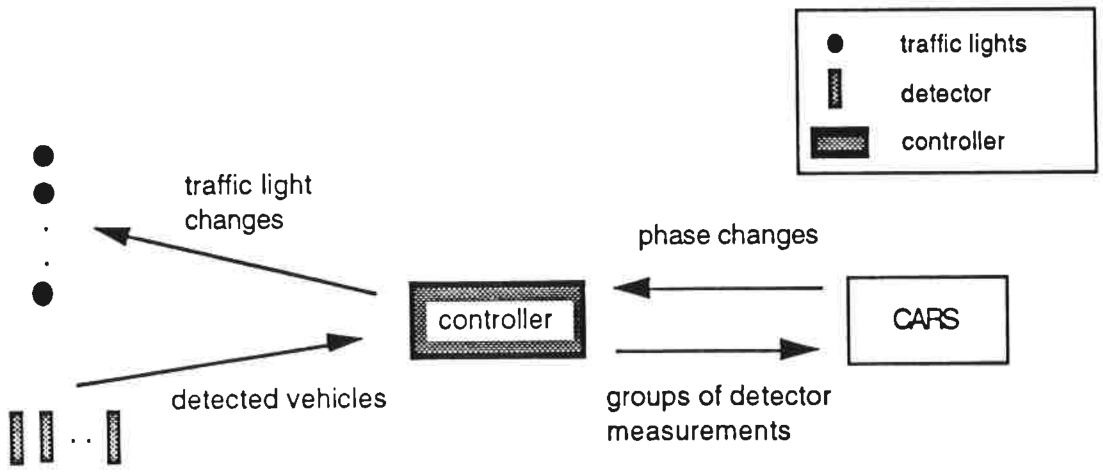


Fig. 9.1 Interaction between a controller and the CARS system.

CARS allows the controller to have connections with detectors which are not positioned on the entrance or exit sections of the associated junction. Thus, connections are not tied to the model and can be driven by economic and physical considerations. CARS V1 interacts with the controllers through a few, clearly defined functions which are specified in **APPENDIX 1** and summarized here.

• **Initialization functions**

CONNECT CONTROLLER
: Connects a controller to the CARS system, providing information about time intervals and initial state.

CONNECT DETECTOR
: Connects a detector to the CARS system, providing information about the data communication.

SET CONTROLLERS
: Checks that all controllers have reached the required initial state.

• **Synchronization functions**

SYNCHRONIZE CONTROLLERS
: Resets the clock in all the controllers, thus synchronizing them.

• **Data exchange functions**

RESET DETECTORS
: Indicates to the controllers that the last scycle measurements from the detectors must be sent to CARS.

CHANGE JUNCTION
: Indicates to the controller the moment when the associated junction has to finish the current phase.

9.1 Requirements

The modeling approach in CARS V1 imposes certain requirements on the traffic lights and detectors:

- Time discretization. Time is discretized at delta seconds intervals, which means that the system can send phase changes only each delta seconds. The value of the parameter delta must be in the range from 1 to 5 secs.; a value of 2 secs. has been used in most of the experiments.
- Detector positioning. A section-wide detector is required on all entrance sections to the network. In the remaining sections, detector positioning and number is arbitrary, although it is recommended to position at least one detector in the sections where the traffic flow is greater or that are difficult to model because of accidents or parking infringements. In the absence of detector measurements in one section, CARS cannot correct the PACKSIM model on that section during the feedback; because of the accurate modeling, this may be permissible on secondary sections in the network.

9.2 Fault tolerance

Fault tolerance in this context means how the CARS system reacts to a failing controller, traffic light, or detector. A failure in the computer where the CARS system is run is, by now, not considered. Therefore, failures are detected either directly in the error codes returned by controllers' functions, or by inspecting the detector measurements received from controllers. In these cases, warning messages are sent to the operator's console and, meanwhile, the system tries, whenever possible, to replace the unavailable detector measurements or to suppose the most plausible state in the junction's signalization.

9.2.1 Phase-changing failure

Recognized as an error code returned by the CHANGE JUNCTION function. In the case of a demand-responsive junction, the system assumes that, from then on, control switches to fixed. However, the system does not stop sending phase changes with the CHANGE JUNCTION function and, on returning successfully, control is again considered as demand-responsive. Warning messages are sent to the operator's console each time the CHANGE JUNCTION returns with error.

9.2.2 Detector failure

When a detector failure occurs, treatment depends on whether the detector is the first one in an entrance section to the network. If this is the case, then the system can hardly function without the detector, and the following action is evaluated, in this order, to find a temporary solution:

1. If a vehicle generator has been specified to substitute the detector in case of failure, then use it. An evolution function can be associated with the vehicle generator in order to model daily variations.
2. If the detector had been sending correct measures for at least five minutes, then CARS uses the predictor of external arrivals on that approach in order to replace detector measurements.

In the other detectors, a failure causes the system to ignore the detector, and the simulation model is supposed to give, at least temporarily, a better approximation than could be provided by any generator or predictor.

In both cases, an error in a CONNECT DETECTOR function results in the detector being definitively catalogued as useless. On the contrary, an error in the RESET DETECTORS function, or when cleaning the measurements, starts the substitution mechanisms only during that SCC.

9.2.3 General controller failure

The execution cannot progress if the initialization functions CONNECT CONTROLLERS or SET CONTROLLERS return error. On the other hand, if the failure comes from the SYNCHRONIZE CONTROLLERS function, then the system takes the following action:

- Phase changes: The previously demand-responsive junctions switch to fixed signalization, and the system proceeds as in a phase change failure.
- Detector measurements: If available, the substitution mechanisms activate, as in a detector failure.
- Synchronization frequency: The calling frequency to the SYNCHRONIZE CONTROLLERS function increases to one call per minute until returning successfully and the situation can be normalized.

10. Tests

CARS V1 has been tested in various scenarios, including isolated intersections, arterials, and networks. We have used the AIMSUN (Barceló, Ferrer and Montero 1989) microscopic traffic simulator as a substitute for the real world: AIMSUN sends detector counts to CARS and receives stage (phase) endings; see Figure 10.1. Previously, AIMSUN had been calibrated using real-world data.

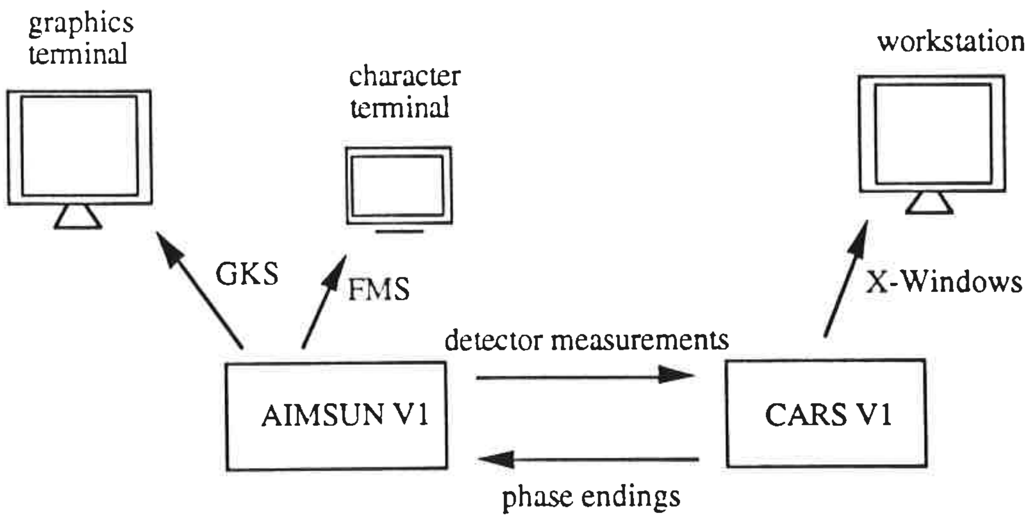


Fig. 10.1 Testing environment for CARS V1.

10.1 Testing scenarios

We have tested CARS V1 in four scenarios, namely, an isolated junction, a four-junction arterial, a nine-junction network, and an eight-junction network. They are described hereafter, including their geometry, turning percentages, and two sets of vehicle arrivals at the network entrances, one representing light-traffic conditions, and the other for heavy-traffic conditions.

Junction

The first scenario is a junction with 3 entrance sections and 3 exit sections; see Figure 10.2. Section lengths and turning probabilities are described in Figure 10.3. The junction has two phases, the first one for the horizontal approaches, and the second for the vertical. We suppose interphase periods of 4 secs. duration. The maximum speed in all the sections is 50 km/h.

The first set of vehicle arrivals, representing light traffic conditions, varies around the average value of 400 veh/h*lane (Figure 10.4). A Webster-based fixed-control plan gives a cycle length of 36 secs., 14 secs. for each phase.

The second set of vehicle arrivals, for heavy conditions, has the average value of 800 veh/h*lane (Figure 10.5). A Webster-based fixed-control plan gives a cycle length of 90 secs., 41 secs. for each phase.

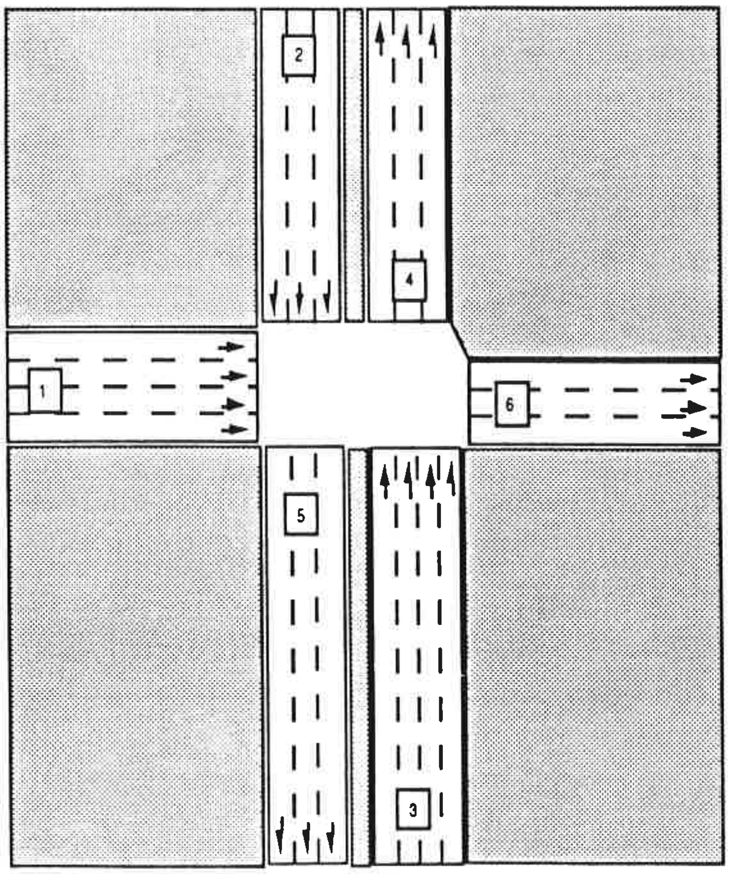


Fig. 10.2 Isolated junction used to test CARS V1.

sections	length (meters)
1	42
2	45
3	57
4	45
5	57
6	36

A

origin sections	turning probab.	destination sections
1	0.375	4
	0.375	6
	0.25	5
2	1.0	5
3	0.625	4
	0.375	6

B

Fig. 10.3 Section lengths (A) and turning probabilities (B) corresponding to the isolated junction in Figure 10.2.

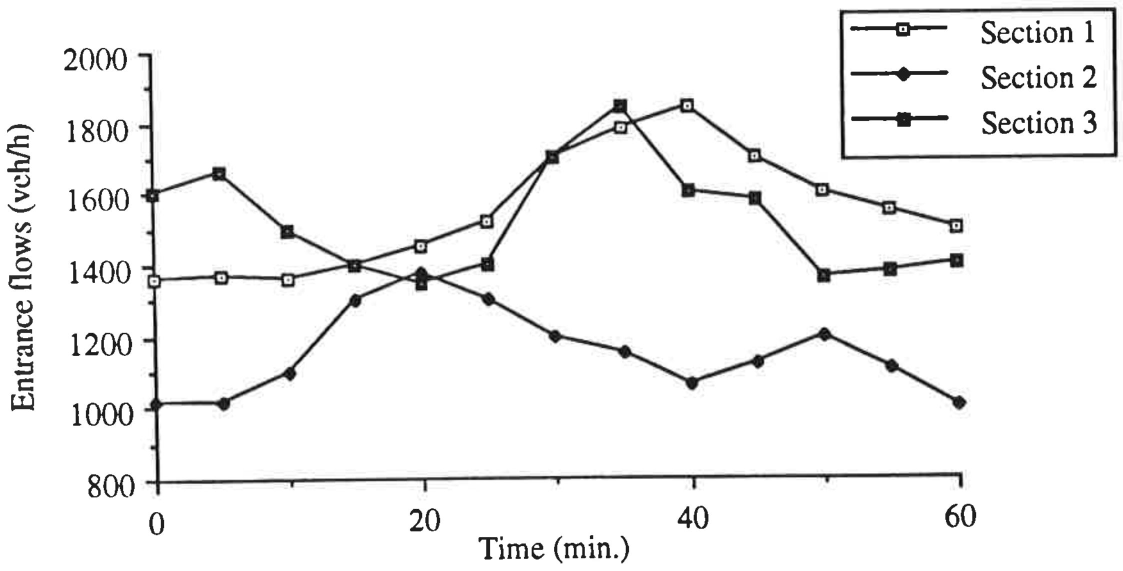


Fig.10.4 Vehicle arrivals in light traffic conditions at the junction in Figure 10.2.

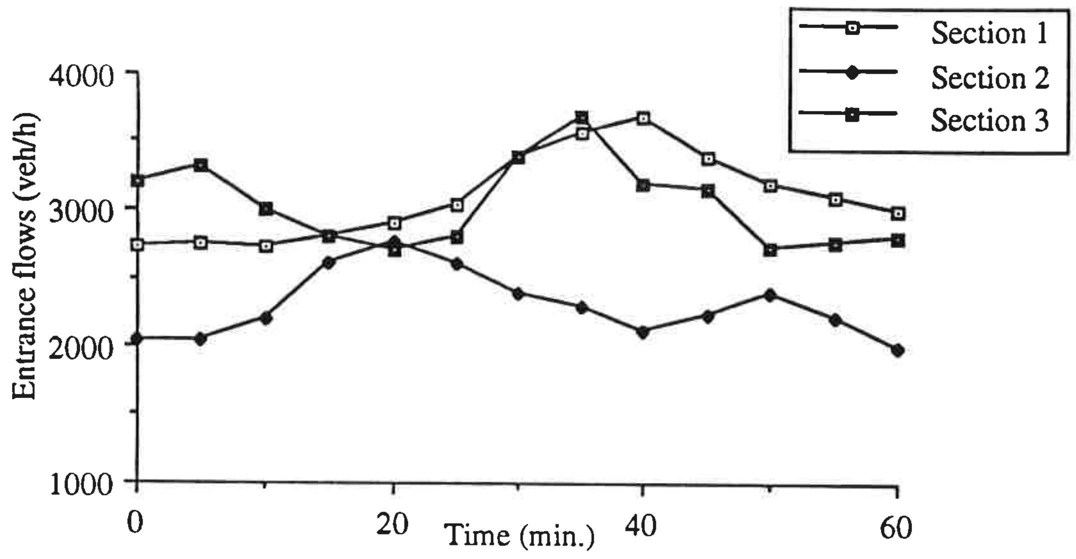


Fig.10.5 Vehicle arrivals in heavy traffic conditions, at the junction in Figure 10.2.

Arterial

The second scenario is a four-junction arterial with 6 entrance sections and 8 exit sections; see Figure 10.6. Section lengths and turning probabilities are described in Figures 10.7 and 10.8, respectively. All junctions have two phases, the first one for the horizontal approaches, and the second for the vertical. We suppose interphase periods of 4 secs. duration. The maximum speed in all sections is 50 km/h.

The first set of vehicle arrivals, representing light traffic conditions (Figure 10.9), varies around the average value of 610 veh/h*lane in Sections 1 and 19, and 600 veh/h*lane in the vertical approaches. A fixed-control plan based on Webster and Little-Morgan gives a cycle length of 90 secs., 41 secs. for each phase, and zero offset in all the junctions.

The second set of vehicle arrivals, for heavy conditions (Figure 10.10), has the average value of 800 veh/h*lane in Sections 1 and 19, and 600 veh/h*lane in the vertical approaches. A fixed-control plan based on Webster and Little-Morgan gives a cycle length of 90 secs., 47 secs. and 35 secs. for the first and second phase, respectively, and zero offset in all the junctions.

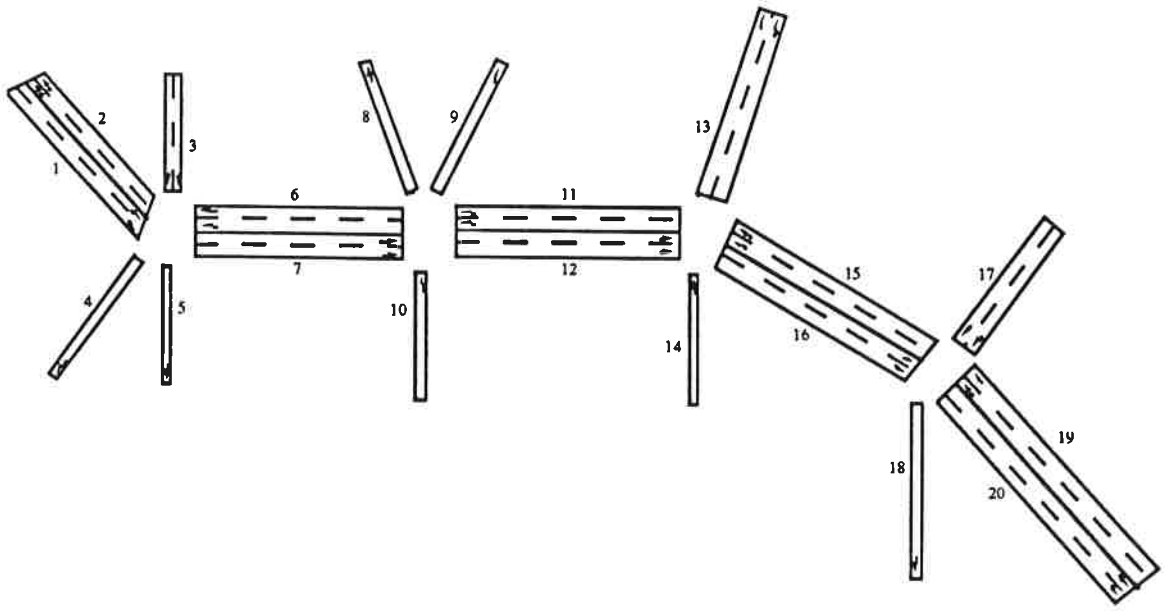


Fig. 10.6 Four-junction arterial used to test CARS V1.

sections	length (meters)
1	136.
2	136.
3	101.5
4	96.25
5	78.75
6	112.
7	112.
8	78.75
9	78.75
10	96.25

sections	length (meters)
11	113.75
12	113.75
13	78.75
14	94.5
15	112.87
16	112.87
17	70.
18	84.
19	85.75
20	85.75

Fig. 10.7 Section lengths corresponding to the arterial in Figure 10.6.

origin sections	turning probab.	destination sections
1	0.1	4
	0.1	5
	0.8	7
3	0.3	2
	0.1	4
	0.1	5
	0.5	7
6	1.0	2
7	1.0	12

origin sections	turning probab.	destination sections
10	0.4	12
	0.1	9
	0.1	8
	0.4	6
11	0.1	9
	0.1	8
	0.8	6
12	1.0	16
14	0.4	16

origin sections	turning probab.	destination sections
	0.2	13
	0.4	11
15	0.3	13
	0.7	11
16	0.3	18
	0.7	20
17	0.3	15
	0.2	18
	0.5	20

Fig. 10.8 Turning probabilities corresponding to the arterial in Figure 10.6.

TIME	Section 1	Section 3	Section 10	Section 17	Section 19
0:00	1050	1300	600	1240	1120
0:05	1120	1200	580	1230	1030
0:10	1100	1400	630	1400	1080
0:15	1200	1230	690	1210	1100
0:20	1350	1140	650	1280	1220
0:25	1400	1050	630	1050	1330
0:30	1320	1200	620	1390	1210
0:35	1140	1400	530	1210	1300
0:40	1300	1200	510	1150	1380
0:45	1280	1160	550	1200	1400
0:50	1140	1270	600	1280	1200
0:55	1090	1240	630	1220	1180
1:00	1040	1280	620	1240	1230

Fig.10.9 Vehicle arrivals in light traffic conditions, in the arterial in Figure 10.6.

TIME	Section 1	Section 3	Section 10	Section 17	Section 19
0:00	1360	1300	600	1240	1400
0:05	1400	1200	580	1230	1430
0:10	1380	1400	630	1400	1570
0:15	1450	1230	690	1210	1600
0:20	1570	1140	650	1280	1700
0:25	1630	1050	630	1050	1800
0:30	1700	1200	620	1390	1840
0:35	1750	1400	530	1210	1790
0:40	1810	1200	510	1150	1600
0:45	1840	1160	550	1200	1400
0:50	1740	1270	600	1280	1470
0:55	1720	1240	630	1220	1500
1:00	1660	1280	620	1240	1540

Fig.10.10 Vehicle arrivals in heavy traffic conditions, in the arterial in Figure 10.6.

Network1

The third scenario is a nine-junction network with a total of 24 sections, 6 entrance sections and 6 exit sections; see Figure 10.11. It is based on a part of the 'Eixample' in Barcelona, between C/ València and C/ Consell de Cent, and between C/ Pau Claris and C/ Bruc. All sections have a length of 67 meters. As regards turning probabilities, the general rule is: vehicles have a 0.8 probability of continuing straight on and 0.2 of turning. The exceptions are Section 17, where the probability of continuing straight on is 0.95, and Section 18, where the probability is 0.9. All junctions have two phases, the first one for the horizontal approaches, and the second for the vertical, with interphase periods of 4 secs. duration. The maximum speed in all the sections is 50 km/h.

The first set of vehicle arrivals, representing light traffic conditions (Figure 10.12), has average values varying from 400 to 600 veh/h*lane, depending on the section. The fixed-control plan was supplied by the local authorities; see Figure 10.13.

The second set of vehicle arrivals, for heavy conditions (Figure 10.14), has average values varying from 700 to 800 veh/h*lane, depending on the section. A fixed-control plan based on Webster and Little-Morgan gives a cycle length of 90 secs., 44 secs. and 38 secs. for the first and second phase, respectively. The offsets are: 10 secs. in Junction 1, 20 secs. in Junction 2, 30 secs. in Junction 3, 20 secs. in Junction 4, 10 secs. in Junction 5, 0 secs. in Junction 6, 80 secs. in Junction 7, 0 secs. in Junction 8, and 10 secs. in Junction 9.

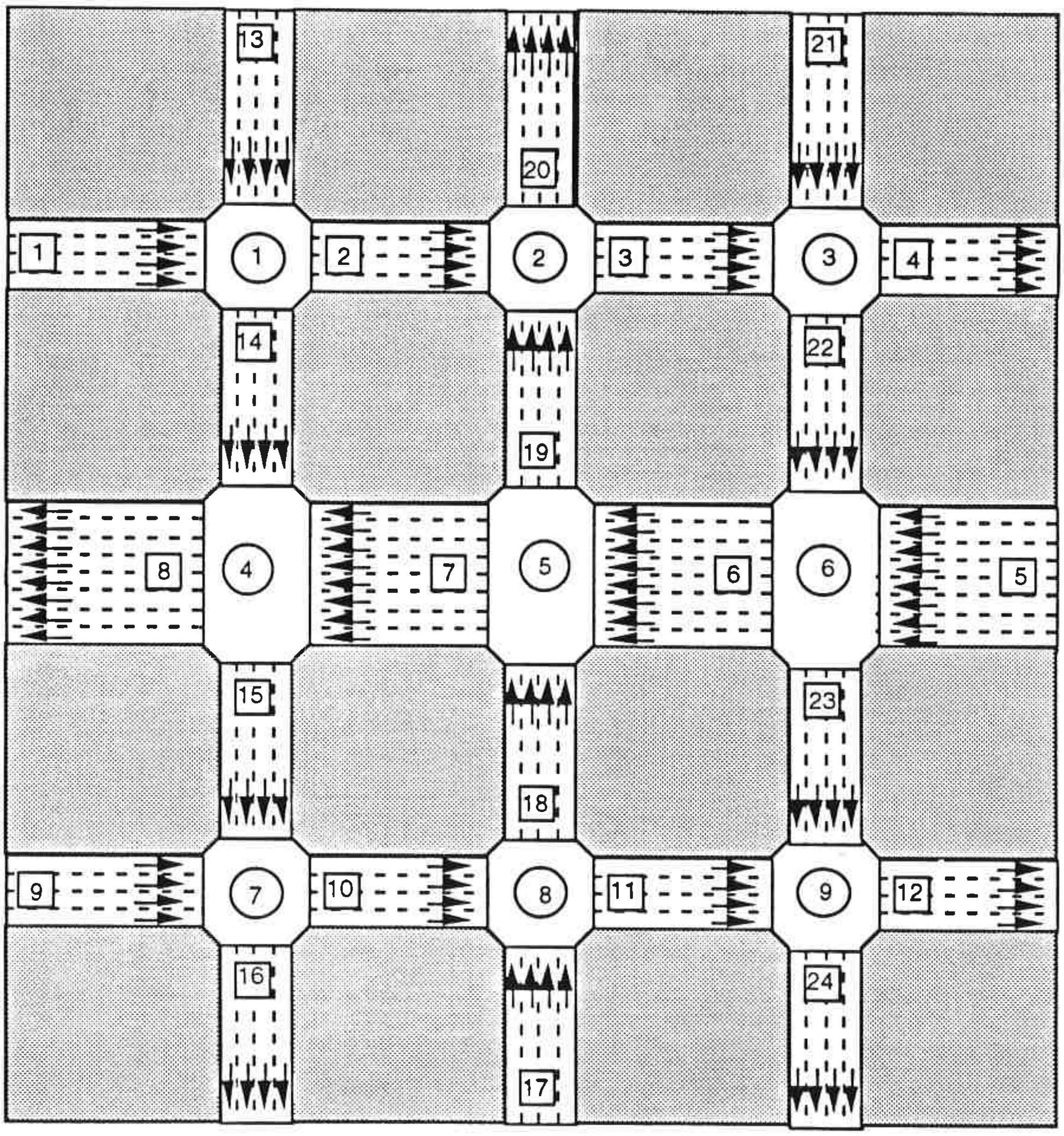


Fig. 10.11 Nine-junction network, named network1, used to test CARS V1.

TIME	Section 1	Section 5	Section 9	Section 13	Section 17	Section 21
0:00	1600	4800	1240	2100	2000	2000
0:05	1360	4850	1300	2300	2100	2070
0:10	1400	4720	1200	2160	2260	1800
0:15	1480	5200	1500	2020	2000	1820
0:20	1600	5520	1800	1900	1950	1640
0:25	1800	5300	2000	1700	1800	1530
0:30	1840	5250	1900	1730	1700	1600
0:35	1680	5020	1700	1850	2000	1630
0:40	1640	4860	1600	2130	2130	1700
0:45	1440	4540	1540	2300	2210	1780
0:50	1680	4200	1510	2200	1900	1830
0:55	1640	4420	1400	1870	2000	1800
1:00	1280	4500	1570	2000	1800	1670

Fig.10.12 Vehicle arrivals in light traffic conditions, in network1 in Figure 10.11.

junction	phase 1	phase 2	offset	cycle
1	40	42	10	90
2	40	42	0	90
3	43	39	40	90
4	39	43	40	90
5	40	42	10	90
6	39	43	40	90
7	28	24	2	60
8	28	24	0	60
9	28	24	42	60

Fig.10.13 Control plan in light traffic conditions, in network1 in Figure 10.11.

TIME	Section 1	Section 5	Section 9	Section 13	Section 17	Section 21
0:00	1600	4800	1240	2100	2000	2000
0:05	1360	4850	1300	2300	2100	2070
0:10	1400	4720	1200	2160	2260	1800
0:15	1480	5200	1500	2020	2000	1820
0:20	1600	5520	1800	1900	1950	1640
0:25	1800	5300	2000	1700	1800	1530
0:30	1840	5250	1900	1730	1700	1600
0:35	1680	5020	1700	1850	2000	1630
0:40	1640	4860	1600	2130	2130	1700
0:45	1440	4540	1540	2300	2210	1780
0:50	1680	4200	1510	2200	1900	1830
0:55	1640	4420	1400	1870	2000	1800
1:00	1280	4500	1570	2000	1800	1670

Fig.10.14 Vehicle arrivals in heavy traffic conditions, in network1 in Figure 10.11.

Network2

The fourth scenario is an eight-junction network with a total of 19 sections, 6 entrance sections and 5 exit sections; see Figure 10.15. It is based on a part of the 'Eixample' in Barcelona, between C/ Trafalgar, C/ Roger de Llúria, and Ronda de Sant Pere. Section lengths and turning probabilities are described in Figures 10.16 and 10.17, respectively. All junctions have two phases, the first one for the horizontal approaches, and the second for the vertical, with interphase periods of 4 secs. duration. The maximum speed in all the sections is 50 km/h.

The first set of vehicle arrivals, representing light traffic conditions (Figure 10.18), has average values varying from 250 to 600 veh/h*lane, depending on the section. A fixed-control plan based on Webster and Little-Morgan is shown in Figure 10.19.

The second set of vehicle arrivals, for heavy conditions (Figure 10.20), has average values varying from 400 to 700 veh/h*lane, depending on the section. A fixed-control plan based on Webster and Little-Morgan is described in Figure 10.21.

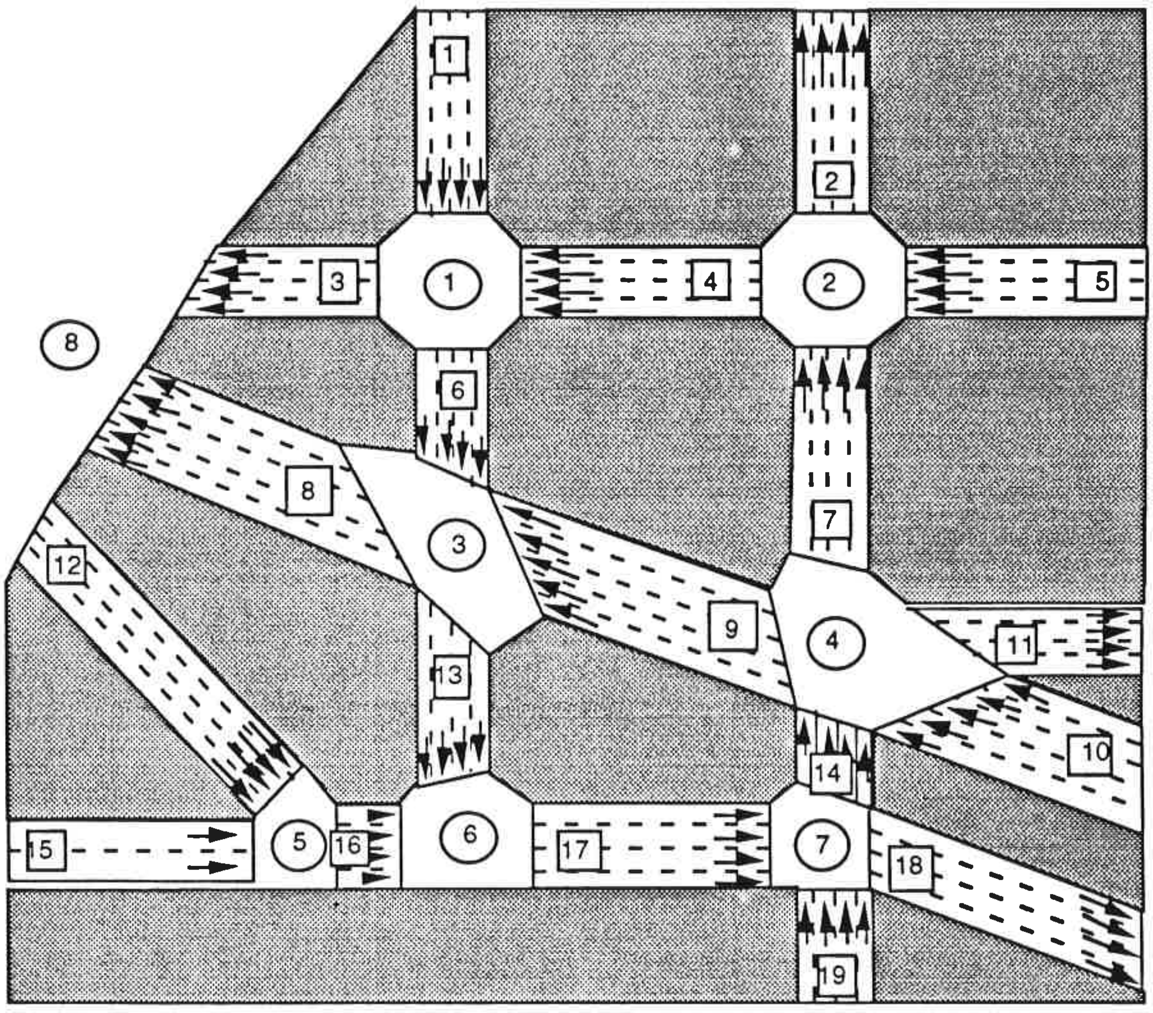


Fig. 10.15 Eight-junction network, named network2, used to test CARS V1.

sections	length (meters)
1	68.
2	68
3	62.
4	81.
5	79.
6	36.
7	73.
8	94.
9	95.
10	71.
11	61.
12	118.
13	53.
14	25.
15	82.
16	21.
17	80.
18	96.
19	38.

Fig. 10.16 Section lengths corresponding to network2 in Figure 10.15.

origin sections	turning probab.	destination sections
1	0.2	3
	0.8	6
4	0.9	3
	0.1	6
5	0.2	2
	0.8	4
6	0.2	8
	0.8	13
7	0.9	2
	0.1	4
9	0.9	8
	0.1	13
10	0.1	7
	0.9	9
14	0.2	9
	0.6	7
	0.2	11
17	0.9	18
	0.1	14
19	0.2	18
	0.8	14

Fig. 10.17 Turning probabilities corresponding to network2 in Figure 10.15.

TIME	Section 1	Section 5	Section 10	Section 12	Section 15	Section 19
0:00	900	1400	2125	900	270	2100
0:05	850	1460	2300	800	300	2040
0:10	1000	1400	2420	920	255	2100
0:15	1050	1320	2700	750	310	2500
0:20	1150	1300	2875	680	320	2200
0:25	1000	1190	2500	700	290	2600
0:30	1034	1200	2300	800	270	2500
0:35	1070	1400	2150	720	310	2400
0:40	1000	1460	2400	860	345	2670
0:45	940	1600	2300	800	330	2760
0:50	920	1610	2320	730	300	2400
0:55	980	1500	2500	810	280	2340
1:00	1010	1380	2450	700	290	2500

Fig.10.18 Vehicle arrivals in light-traffic conditions, in network2 in Figure 10.15.

junction	phase 1	phase 2	offset	cycle
1	48	34	8	90
2	29	53	57	90
3	46	36	10	90
4	45	37	30	90
5	48	34	8	90
6	29	53	57	90
7	38	44	20	90
8	48	34	50	90

Fig.10.19 Control plan in light traffic conditions, in network2 in Figure 10.15.

TIME	Section 1	Section 5	Section 10	Section 12	Section 15	Section 19
0:00	1600	2200	3300	1840	700	1200
0:05	1650	2100	3100	1700	595	1190
0:10	1720	2220	2975	1750	720	1450
0:15	1620	2300	3060	1600	805	1300
0:20	1530	2230	3400	1470	700	1560
0:25	1630	1960	3700	1360	630	1610
0:30	1700	1870	3800	1400	600	1500
0:35	1810	1990	4025	1700	670	1470
0:40	1900	2000	3900	1600	780	1429
0:45	1920	2400	3500	1580	703	1250
0:50	2030	2530	3400	1500	640	1295
0:55	2070	2480	3700	1530	690	1240
1:00	1976	2170	3450	1600	720	1420

Fig.10.20 Vehicle arrivals in heavy traffic conditions, in network2 in Figure 10.16.

junction	phase 1	phase 2	offset	cycle
1	41	41	69	90
2	52	30	58	90
3	52	30	13	90
4	53	29	0	90
5	38	44	18	90
6	48	34	65	90
7	60	22	77	90
8	34	48	78	90

Fig.10.21 Control plan in heavy traffic conditions, in network2 in Figure 10.15.

10.2 Preliminary adjustments

The first step in testing CARS is to adjust its internal simulation model, PACKSIM, to field conditions. For this purpose, certain local parameters in each section need to be specified:

- Average vehicle length.
- Average distance between stopped vehicles.
- Maximum speed.

These are taken, directly, from their real-world counterparts; in this case, from the scenarios specification in AIMSUN (vehicle modalities, vehicle lengths, etc.). Next, we adjust the PACKSIM global parameters:

a) Number of vehicles starting from stopped per lane and second.
b) Average acceleration (constant) of a vehicle in free conditions.
c) Influence time of an obstacle.

These global parameters can also be measured directly in the real world. In this case, they have been adjusted by visually examining the vehicles' behavior in AIMSUN, without resorting to examining the exact value of these parameters in the AIMSUN code. The values are: a) 0.5, b) 10., c) 3., and, as expected, they are independent of the testing scenario and traffic conditions. Their independence of traffic conditions will be showed in the results hereafter.

In contrast, parameters in the ACTS-based model reviewed in I.3 showed an annoying dependence on the scenario's geometry and had to be focused to certain traffic conditions — the farther current traffic conditions were from these ones, the worse the results.

10.3 First results

Only a few results are presented here, mainly to illustrate certain disadvantages of CARS V1 that will be corrected in PART IV, where a larger suite of tests is included.

Once the simulation model had been adjusted, we tested CARS V1 against fixed-control strategies on the four scenarios above — the isolated junction, the arterial, and the two networks. The tests included both light and congested conditions, each with the corresponding fixed-control timings as described in 10.1. In both cases, CARS showed a 5-30% improvement over the fixed-control timings, as shown in Figure 10.22.

In contrast, results using the ACTS-based model (I.3) showed some improvements on the isolated junctions and the arterial in light traffic conditions, but it turned out to be worse than fixed-control timings in heavy traffic conditions. We believe that the more accurate modeling of queues used in CARS (PACKSIM) helps to explain the advantage when conditions near congestion and queues grow. As for the worse results in light traffic conditions, we recall that in the ACTS-based model we chosen a binary-choice autoadaptive logic, whereas the CARS V1 strategy, described in 7.1, considers increasing or decreasing the duration of the current phase. The fact that the binary-choice strategy turns out to be better in isolated junctions when operating in light traffic conditions can be explained by the system's ability to change phases fast enough to take advantage of the wider gaps between vehicles; the CARS V1 strategy cannot change timings so fast. This problem will be addressed in PART IV. However, the CARS V1 control strategy shows better results than the binary choice in all the other cases — in congested conditions, and when coordination between junctions is needed.

	control	delay in light cond.	deviation from fixed control	delay in congested conditions	deviation from fixed control
isolated junction	fixed	145		280	
	responsive	136	-6.2 %	246	-12.1 %
arterial	fixed	192		265	
	responsive	185	-3.6 %	207	-21.9 %
network 1	fixed	129		229	
	responsive	88	-31.8 %	208	-9.2 %
network 2	fixed	140		174	
	responsive	100	-28.6	162	-6.9 %

Fig 10.22 Comparison of results in the four scenarios.
Delay was measured per vehicle and km.

Another problem in CARS V1 is that, particularly in heavy traffic conditions, if the autoadaptive strategy starts from control timings that are far removed from the optimal timings for those conditions, then the system may fall into a local optimum where it does not progress towards the global optimal timings and, as a result, may produce worse results than a fixed-control optimal strategy (Figure 10.23).

As regards the computational feasibility of CARS V1, networks with nine junctions were processed in real time by using only 20% of the available CPU time in a VAXStation 3200 rated at 3 MIPS, so that, given the current availability of more powerful computers, CARS V1 could easily be applied to bigger networks. As will be seen in PART IV, one way of solving CARS V1's current problems increases these computing requirements considerably.

	control	delay in congested conditions	deviation from fixed control
isolated junction	fixed	280	+2.5 %
	responsive	287	
arterial	fixed	265	-0.37 %
	responsive	264	
network 1	fixed	229	-5.67 %
	responsive	216	
network 2	fixed	174	-6.32 %
	responsive	163	

Fig 10.23 Comparison of results in the four scenarios in heavy traffic conditions, where the autoadaptive strategy starts from light-traffic optimal timings.

11. Conclusions and required improvements

This part has presented a first version of a demand-responsive traffic control system, CARS V1, that includes some innovative aspects such as a congestion-oriented simulation model, flexible control timing and detector positioning requirements, and control logics. The first tests with the AIMSUN microscopic traffic simulator confirm the advantage in heavy-traffic conditions and also show two points to ameliorate: effectiveness in light traffic conditions — this seems to require a faster autoadaptiveness —, and the capacity to start from control timings far removed from optimal. These two points will be considered in PART IV, where a new version of the system, CARS V2, will be proposed and a complete suite of tests will be included.

However, before ameliorating CARS V1, there is another, more fundamental area that needs to be improved, and this is the environment in which to test CARS. Currently, each test with the AIMSUN V1-CARS V1 systems requires a lot of effort and is prone to errors:

- Creation of a new network has to be done in both systems, each one requiring the user to specify the network in a different way, with a different format, and using different concepts. For example, AIMSUN V1 control lights are not stage-based but phase-based, so the communication of phase endings from CARS V1 to AIMSUN V1 requires additional files that specify the correspondence between a stage in CARS and the control lights' state in AIMSUN.
- Equally, almost any change has to be done separately in both systems.

- CARS V1 provides a graphical interface, CARSedi, to specify and modify the network, but AIMSUN V1's interface for the same purpose is not graphical and it is difficult to detect errors.
- AIMSUN V1 only runs on VAX VMS systems, such as, at the time of this testing, the VAXStation 3200. Tests on this workstation are painfully slow — both systems, AIMSUN V1 and CARS V1, run concurrently.

The emergence of much faster Unix workstations offers the chance to use a better testing environment where more tests can be performed.

Some of these problems are inherent in the traffic engineer having to deal with more than one traffic model, and wanting to share data between them. In the next part, PART III, a new environment — that introduces some innovative ideas — will be proposed that solves all the previous testing difficulties. However, the environment does not limit its applicability to the AIMSUN-CARS testing problem; it has been designed as a general tool for traffic analysis and modeling, capable of supporting other traffic models and different levels of detail.

12. References

Allsop, R.E. (1991). "Signal control at individual junctions: stage-based approach.", *Concise Encyclopedia of Traffic and Transportation Systems*, Editor Markus Papageorgiou, Pergamon Press, pp. 478-483.

Bang, K-L. (1976). "Optimal control of isolated traffic signals.", *Traffic Eng. Control*, 17(7), pp. 288-292.

Barceló, J., J. Ferrer, and L. Montero (1989). *AIMSUN. Descripción del sistema*, Department of Statistics and Operational Research, Faculty of Computer Science, Universidad Politècnica de Catalunya.

Barceló, J., J. Ferrer, and L. Montero (1989). *AIMSUN: Advanced Interactive Microscopic Simulator for Urban Networks. Vol I: System Description, and Vol II: User's Manual*, Department of Statistics and Operational Research, Faculty of Computer Science, Universidad Politècnica de Catalunya.

Barceló, J., R. Grau, P. Egea and S. Benedito (1991). "CARS: A demand-responsive traffic control system.", *Applications of advanced technologies in transportation, Proceedings of the 2nd International Conference*, Minneapolis, pp. 91-95.

Barriere, J.F., J.L. Farges, J.J. Henry (1986). "Decentralization versus hierarchy in optimal traffic control.", *Control in Transportation Systems, 5th IFAC/IFIP/IFORS Conference*, Vienna, 8-11, 1986, pp. 321-326.

Bonvallet, F. and Y. Robin-Prevallee (1987). "Mise au point d'un indicateur permanent des conditions de circulation en Ile-de-France.", *TEC* n84/85, Sept.-Oct.-Nov.-Dec.

Bonvallet, F. and Y. Robin-Prevallee (1989). *Mederation du trafic urbain: Approches nouvelles - gestion dinamique du trafic - les indicateurs de niveau de service*, Centre d'études des Transports Urbains / Ecole Nationale des Ponts et Chaussées.

Chin, H.C. (1985). "SIMRO: a model to simulate traffic at roundabouts.", *Traffic Eng. Control*, Vol. 26, No. 3, pp. 109-113.

Donati, F., V. Mauro, G. Roncolini and M. Vallauri (1984). "A hierarchical-decentralized traffic light control system. The first realization: Progetto Torino.", *IFAC 9th World Congress*, Vol. II, 11G/A-1.

DRIVE V1015 (1990a). "Delineate System Indicators.", Work Package 6, prepared by Nottingham University for the DRIVE I, V1015 project, *Artificial Intelligence for Traffic Control*, pp. 28-29, 48.

DRIVE V1015 (1990b). "Strategies for treating congestion.", Work Package 8, section prepared by Nottingham and Leeds Universities for the DRIVE I, V1015 project, *Artificial Intelligence for Traffic Control*, pp. 2.

Gartner, N.H. (1982). *Demand-responsive decentralized urban traffic control: Part I: Single-intersection policies*, Office of University Research, US Department of Transportation, Rept. DOT-RSPA-DPB-50-81-24.

Gartner, N.H., M.H. Kaltenbach et al.(1983). *Demand-responsive decentralized urban traffic control: Part II: Network extensions*, Office of University Research, US Department of Transportation, Rept. DOT-OST-P-34- 85-009, pp. 53-97.

Gault (1981). "An on-line measure of delay in road traffic computer system.", *Traffic Eng. Control*, 22(7).

Grau, R. and J. Barceló (1992a). *PACKSIM: An experience in using traffic simulation in a demand-responsive traffic control system*, Research Report 92/05, Department of Statistics and Operational Research, Faculty of Computer Science, Universidad Politècnica de Catalunya.

Grau, R. and J. Barceló (1992b). *CARS: An experience in demand-responsive traffic control*, Internal Report, Department of Statistics and Operational Research, Faculty of Computer Science, Universidad Politècnica de Catalunya.

Grau, R. and J. Barceló (1992c). *A review of objective functions in the CARS demand-responsive traffic control system*, Research Report 93/01, Department of Statistics and Operational Research, Faculty of Computer Science, Universidad Politècnica de Catalunya.

Henry, J.J, J.L. Farges and J. Tuffal (1983). "The PRODYN real time traffic algorithm.", *Control in Transportation Systems, 4th IFAC/IFIP/IFORS Conference*, Baden-Baden, April 20-22, 1983, pp. 307-312.

Hunt, P.B., D.I. Robertson, R.D Bretherton and R.I. Winston (1981). "SCOOT—A traffic-responsive method of co-ordinating signals.", *TRRL Report* LR1014, Transport and Road, Research Laboratory, Crowthorne, 1981.

Hunt, P.B. et al. (1982). "The Scoot on-line traffic signal optimization technique.", *Traffic Eng. Control*, 23(4), pp. 190-192.

Kaltenbach, M., J.P. Dussault, and G. Marquis (1986). *Adaptive control of traffic signals*, Université de Montréal, Centre de recherche sur les transports, Publication #469.

Kessaci, A., J. Farges and J. Henry (1989). "On-line estimation of turning movements and saturation flows in PRODYN.", *6th IFAC/IFIP/IFORS Conference on Control in Transportation Systems*, Paris, 1989.

Kessaci, A., J. Farges and J. Henry (1991). "Turning movement ratios estimation using inductive loop detectors and information from route guidance systems.", *Recherche Transports Sécurité*, No. 6, Feb. 1991, pp. 39-44.

Leonard, D.R., P. Gower (1982), *User guide to CONTRAM version 4.* Transport and Road Research Laboratory, Supplementary Report No. 735. TRRL, Crowthorne, UK.

Lin, F.B., N. Wang, and S. Vijayakumar (1987). "Development of an intelligent adaptive signal control logic.", *Proc. of the Eng. Foundation Conf.*, June 1987, pp. 257-279.

Lin, F.B. and S. Vijayakumar (1988). "Adaptive signal control at isolated intersections.", *ASCE Journal of Transport Engineering*, Vol. 114, No. 5, pp. 555-573.

List, G. and J. Pond (1989). "A strategy for real-time control of demand-sensitive signal networks.", *AATT Conference*, Feb. 1989, pp. 367-372.

Lovrie, P.R. (1982). "The Sydney co-ordinated adaptive traffic system—principles, methodology, algorithms.", *Proc. IEE Int. Conf. Road Traffic Signalling*, 1982, pp. 67-70.

Michalopoulos, P.G., R. Fitch, B. Wolf (1989). "Development and evaluation of a breadboard video imaging system for wide area detection.", *Transportation Research Record 1225*, Transp. Res. Board, Nat. Res. Council, Washington, D.C, pp. 140-149.

Miller, A.J. (1963). "A computer control system for traffic network." *Proc., 2nd Int. Symp. on Theory of Road Traffic Flow*, London, U.K., pp. 201-220.

Morgan, J.T. and J.D.C. Little (1964), "Synchronizing traffic signals for maximal bandwidth.", *Operational Research*, Vol. 12, No. 6, pp. 896-912.

OECD (1981), "Traffic control in saturated conditions.", *Road Research*, Organization for Economic Co-operation and Development, Jan. 1981.

Robertson, D.I. and R.D. Bretherton (1974). "Optimum control of an intersection for any known sequence of vehicle arrivals.", *Proceedings of the Second IFAC/IFIP/IFORS Symposium on Traffic Control and Transportation Systems*, North-Holland, Amsterdam, 3-17.

Robertson, D.I. (1974). "Cyclic flow profiles.", *Traffic Eng. Control*, June 1974, 15(14), pp. 640-641.

Robertson, G.D. (1987). "Handling congestion with SCOOT.", *Traffic Eng. Control*, April 1987, pp. 228-230.

Stephanedes, Y.J., P.G. Michalopoulos, and Plum, R.A. (1981). "Improved estimation of traffic flow for real-time control.", *Transportation Research Record*, 795, pp. 28-39.

Taylor, I.G., C. Margaret and G. Geary (1987). "Queue volume: a measure of congestion.", *Traff. Eng. Control*, Vol. 28, pp. 582-585.

Vincent, R.A., A.I. Mitchell and D.I. Robertson (1980). "User Guide to TRANSYT Version 8.", *TRRL Report LR888*, Transport and Road Research Laboratory, Crowthorne, 1980.

Vincent, R.A., and C.P. Young (1986). "Self-optimizing traffic signal control using microprocessors—the TRRL 'MOVA' strategy for isolated intersections.", *Traffic Eng. Control*, 27(7/8), pp. 385-387.

Webster, F.V. and B.M. Cobbe (1966). "Traffic signals.", *Road Research Tech. Paper*, No. 56, London: Her Majesty's Stationery Office, pp. 55-60.

PART III
A simulation environment: GETRAM

This part presents a simulation environment, named GETRAM. It solves the difficulties in testing CARS that were discussed at the end of the previous part. We have seen that these difficulties are inherent in the traffic engineer having to use diverse models in order to analyze a traffic network, and that there is the need for a system to solve it. GETRAM provides a unified framework integrating various types of traffic models and tools for traffic analysis, sharing a DataBase, a graphical editor and a module for results presentation. The traffic network can be partitioned into views, hierarchically organized polygons in the real world, so that, for example, a simulation model applies only to one of these restricted areas. Network states produced by one model can be used as a starting point by another model. In order to ease the task of integrating a new model or analysis tool, a library of object-based high-level functions provide a view-aware access to the DataBase, maintaining consistency. Included in this Ph.D. thesis are the design of the whole environment — DataBase, GETRAM API, and graphical editor — and the development of the DataBase and GETRAM API.

As discussed at the end of the last part, certain difficulties are encountered when testing CARS V1 with the AIMSUN microscopic simulator. These are due, mainly, to differences in concepts and entities between the two systems. Creation and updating of traffic networks are tedious because they have to be done separately in each model, and because of the lack of friendly tools in AIMSUN. Interfacing both systems is also more complicated than it would be if they shared a common description of the traffic network.

This is a problem the traffic engineer is usually faced with when having to deal with more than one traffic model. We have identified the lack of an appropriate tool to solve it, and thus the desire to merely solve our problem with CARS V1 has turned into the more ambitious project of providing a system able to solve the more general problem.

GETRAM, which stands for Generic Environment for TRaffic Analysis and Modeling, is a software environment designed to provide support for multiple modeling and analysis approaches — sometimes alternative, but more often complementary — such as traffic assignment and traffic simulation models ranging from macroscopic to microscopic. Although all these models have been available for some years and from various sources, the user is usually faced with incompatible systems:

- Traffic network coding is a cumbersome task, due to both the actual data typing — or mouse operation — and testing the data correctness of each model. Furthermore, the traffic network has different, and usually incompatible, specifications for each model.
- Network objects and, more generally, the conceptual model, tend to differ significantly from model to model, for example, from an assignment model to a microscopic model.
- Data sharing between models becomes a matter of retyping the whole sets of data: control plans, network states, and so on.
- Results obtained with one traffic model cannot be readily accepted as an initial state from which another model can start (again, retyping is the rule). More importantly, because of differences in the conceptual model, data has to be transformed.

The effect of all this burden of work obliges most traffic engineers to restrict themselves to one or, in favourable conditions, two traffic models. GETRAM seeks to solve this situation by providing a comprehensive simulation framework where the network under study is specified once and various types of traffic models can be applied, sharing results in such a way that a simulation model can start from the results of an assignment model — or any other simulation model.

Design and development of GETRAM was carried out within the scope of the TRYS project, funded by the Dirección General de Tráfico. The project started in mid-1991 and lasted until the end of 1993. In its current state, GETRAM contains a graphical editor to specify the network geometry and traffic parameters, a static assignment model, a microscopic simulation model — AIMSUN2 (Ferrer and Barceló 1993), and CARS V2. Included in this Ph.D. thesis are the design of the whole environment (DataBase, GETRAM API, and graphical editor) and the development of the DataBase and GETRAM API. Future developments will build the results presentation and complete the library of functions to allow storage in an INGRES relational DataBase.

1. Historical perspective

Transportation planning systems began to be developed several decades ago. Currently, they can provide capabilities such as travel demand forecasting models, ranging from the classical traffic assignment or the classical four-step models (trip generation, distribution, modal choice, assignment) to the implementation of multimodal equilibration procedures. Moreover, graphical interfaces for network edition and results viewing are increasingly being included. This is the case of EMME/2 (INRO 1992), URBAN/SYS (The Urban Analysis Group 1993), and TRIPS (MVA Systematica 1993), which will be described in more detail. While they can perform their task admirably, they are not able to provide support for other types of models, namely, simulation models, that are increasingly being requested by traffic engineers. This is mainly due to the description of the network solely with the detail required by the assignment model. Nor would it be possible to share certain types of results. From the software point of view, these systems present a multi-module structure, where different modules are called from a menu interface, and data is stored in particular file formats. SATURN (Van Vliet and Hall 1991) incorporates a macroscopic simulation model, but its purpose is mainly to ameliorate the accuracy of the assignment, and moreover, there is no provision for incorporating a more detailed model or sharing results.

TRAF (MacTrans 1989) contains assignment and various types of simulation models, but it lacks integration in that network building is done through model-dependent editors, results cannot be reused or shared, and furthermore the models are not interactive. ASTERIX's (DRIVE V1054 1992) more modern design still does not provide full integration, and each model deals with a particular DataBase.

1.1 EMME/2

EMME/2 (INRO 1992) is a transportation planning system that includes an assignment model, a wide variety of travel demand forecasting methods and a set of associated graphical tools for network and matrix editing and results displaying; see Figure 1.1. The EMME/2 package consists of 49 modules accessing data stored in a common data bank. The modules are subdivided into the following groups: Utilities, Network Editor, Matrix Editor, Function Editor, Assignment Procedures, and Results.

Many modules give a rich functionality to the system but overload the system operation by requiring frequent module switching, for example, to fully specify a network or to view results. Moreover, this switching is done through a rather awkward character-based menu. EMME/2 does not include a simulator model, nor is the network specified with more detail than is strictly required by the traffic assignment model, that is, the description of the network topology. Control and give-ways in the nodes are not considered, thus it cannot analyze changes in these policies directly, but would have to do this indirectly, by penalizing the turning movements.

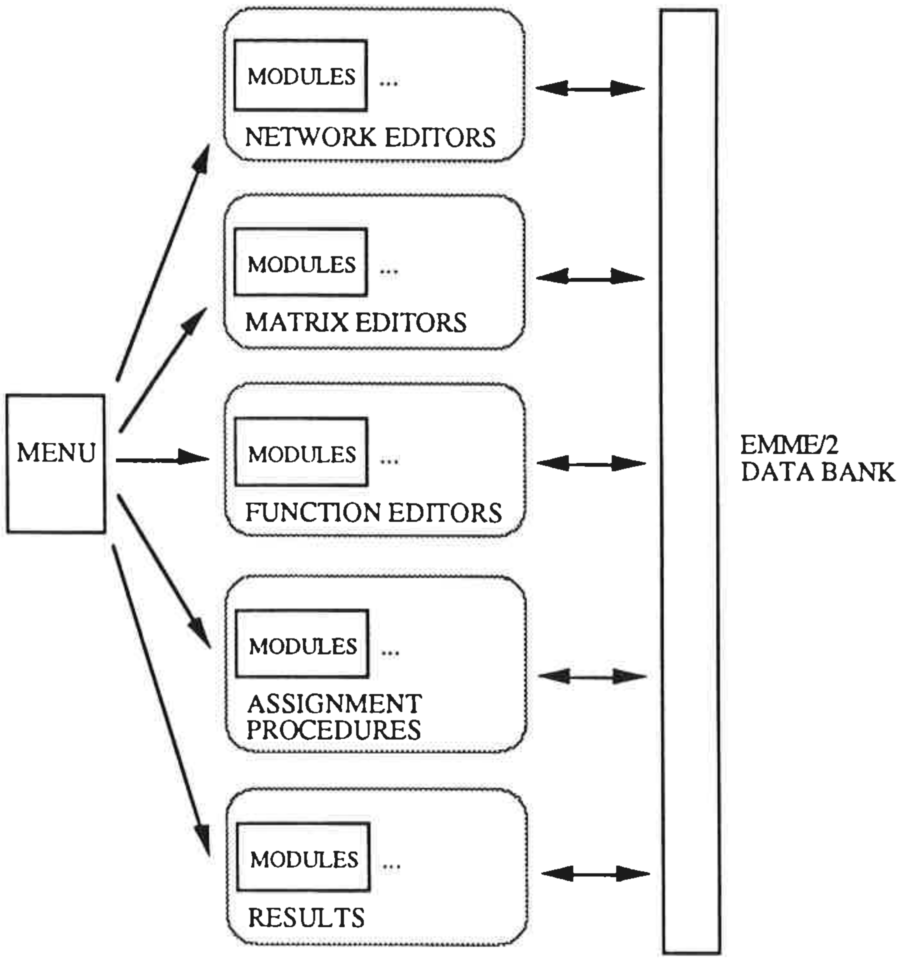


Fig. 1.1 EMME/2 package structure.

1.2 URBAN-SYS

URBAN-SYS (The Urban Analysis Group 1993) is a transportation planning tool designed to aid planners in the planning process at a regional level. Similarly to EMME/2, the system is a package of modules (see Figure 1.2) with the main objective of performing a traffic assignment — a set of modules, TRANPLAN, allows a choice between several traffic assignment models. The system offers network building and sub-network analysis, several options for network loading and includes efficient path search algorithms. TRANPLAN data is stored in a particular file format, and its specification is made easier through two graphical interfaces, TNIS and HNIS, offering several network editing features that permit the user to visually inspect and change geometry and related parameters.

As in EMME/2, the system does not include a simulation model, and network building is at the level of detail required by the assignment models — the user can not expect to analyze a control policy in the junctions.

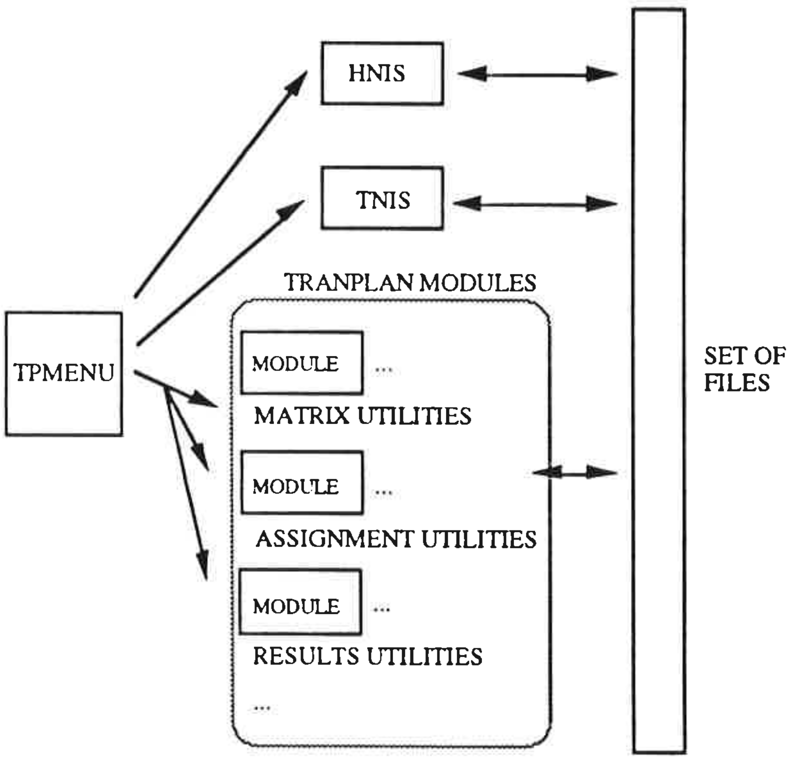


Fig. 1.2 TRANPLAN package structure.

1.3 TRIPS

TRIPS (MVA Systematica 1993), similarly to EMME/2 and TRANPLAN, is a package of programs for transportation planning; see Figure 1.3. But TRIPS's traffic assignment model includes some dynamical features, thus intersection analysis can be performed: the model obtains the flow profiles at an intersection, depending on the control policy. A graphical interface, MVGRAF, allows the user to interactively edit the network, extract subnetworks, and view results. TRIPS data is stored in a particular file format.

Although the system can be used for intersection analysis, the accuracy of the model cannot rival a more specific macroscopic or microscopic model.

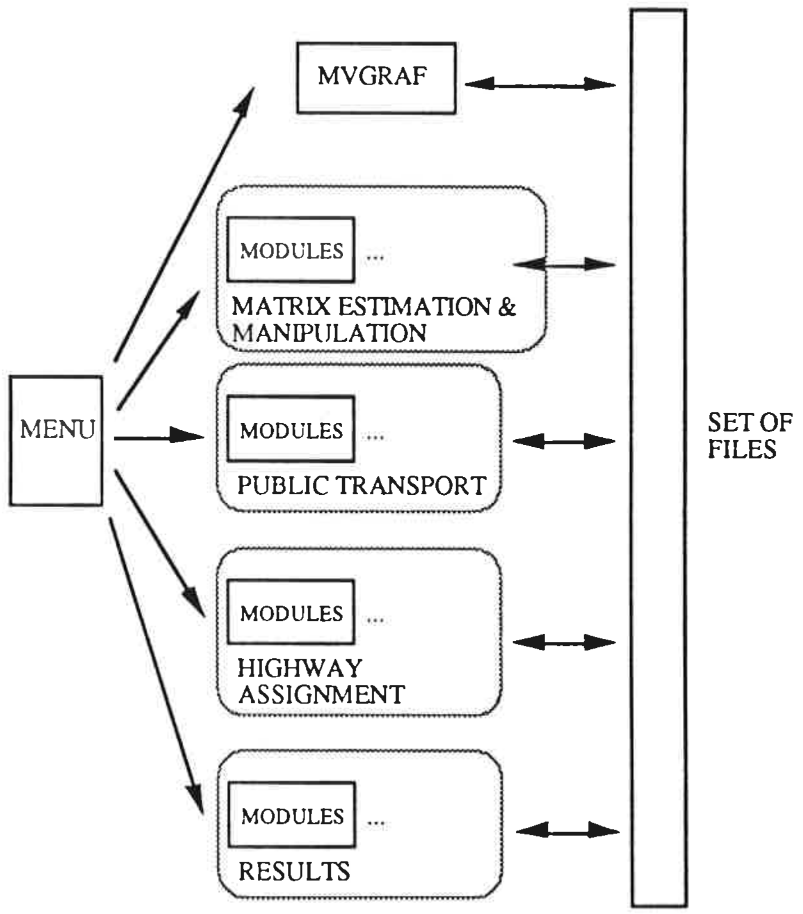


Fig. 1.3 TRIPS package structure.

1.4 SATURN

SATURN (Van Vliet and Hall 1991), which stands for 'Simulation and Assignment of Traffic to Urban Road Networks', is a suite of computer programs around a traffic assignment model, SATASS. The system is closely linked with the ME2 model in order to estimate O/D matrices directly from traffic counts. For better assignment accuracy, a macroscopic simulation model, SATSIM, can be used in a closed loop with the assignment model: the flows generated by the assignment model can be passed to the simulation model, which in turn re-calculates junction delays for re-input to the assignment; see Figure 1.4. SATSIM can also be used to simulate an individual junction, for example, to assess the control policy as if it were an isolated junction. SATSIM models vehicles using cyclic flow profiles (Robertson 1974) derived at a link entrance and exit, and each junction's exit.

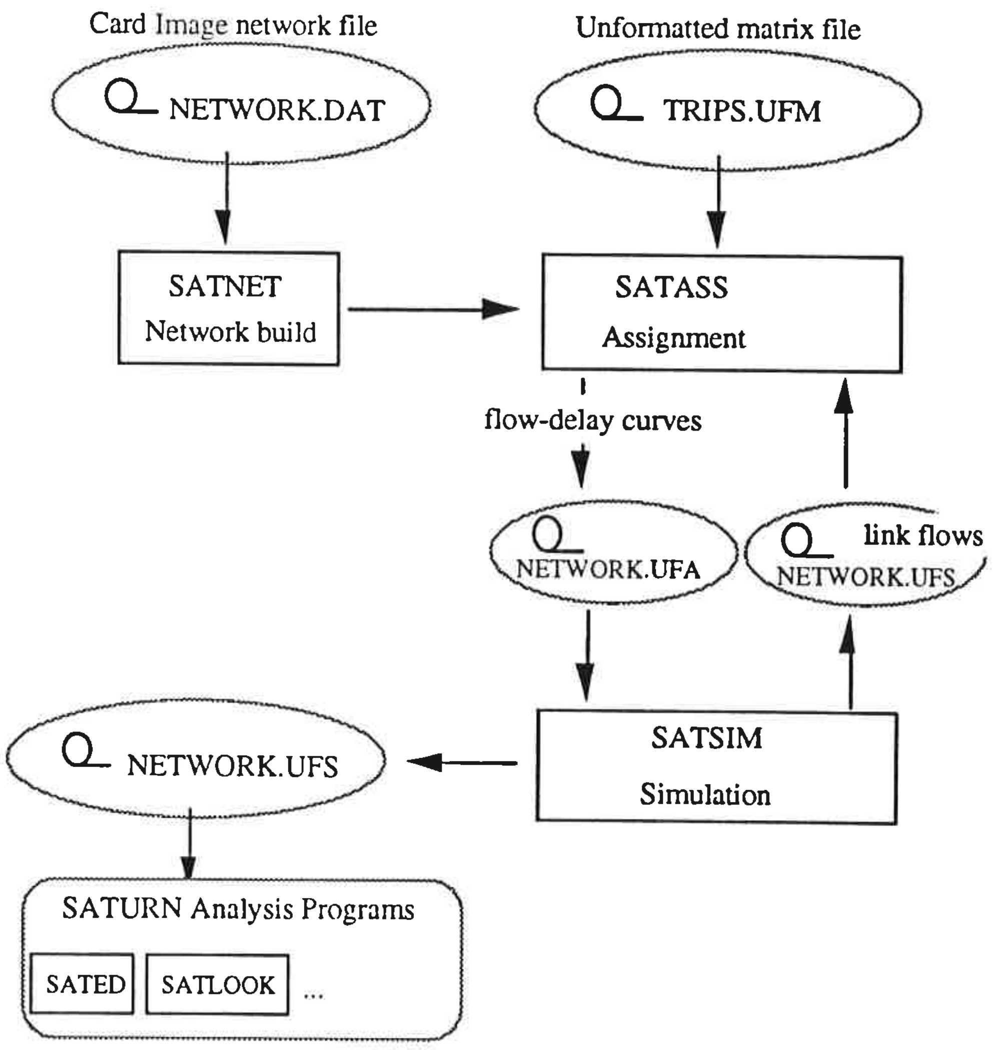


Fig. 1.4 Running the basic SATURN model.

A network can be coded at two levels of detail:

- As an 'inner' or 'simulation' network in which considerable junction-based data in addition to road-based data must be provided by the user.
- As a 'buffer' network, normally surrounding the simulation network, which is coded in the more conventional sense of only requiring data to describe the roads as opposed to the junctions.

Although the assignment and simulation models are highly integrated, main purpose of the simulation model is to ameliorate the assignment accuracy. When used independently of the assignment, SATSIM can only simulate individual junctions, and no interactive output is provided. There seems to be no provision to start SATSIM from the results of a previous simulation run.

1.5 TRAF

The TRAF (MacTrans 1989) system presently consists of five major model components: 1. ROADSIM, a microscopic simulation model of traffic on a two-lane, two-way rural road; 2. NETSIM, a microscopic simulation model of urban traffic; 3. FREFLO, a macroscopic freeway simulation model; 4. NETFLO, a collection of macroscopic simulation models which deal with two different levels of detail for traffic streets; and 5. TRAFFIC, an equilibrium traffic assignment model. Input data for each one of these models is stored in ASCII files, in TRF format. The integration between models comes from a basic common TRF format, although extended then by each particular model. Input data can be generated by a model-dependent editor; for example, input data for NETSIM is usually generated by the character-based NEDIT; see Figure 1.5. Similarly, model-dependent graphical tools exist, in the GTRAF package, for reviewing the input data and analyzing the TRAF simulation results. For example, SNETG provides the means for reviewing the NETSIM input data and analyzing the output results, while ANETG provides an animated view. Both SNETG and ANETG need a previous pass, to convert output files from sequential to direct-access files, as shown in Figure 1.5.

From our point of view, although TRAF presents the advantage of being not-proprietary software containing various types of simulation models, its design is rather old, featuring non-interactive models that need additional software to provide any visualization. More importantly, it does not provide mechanisms to simulate only a restricted area of the network, nor to start from the results of a previous run.

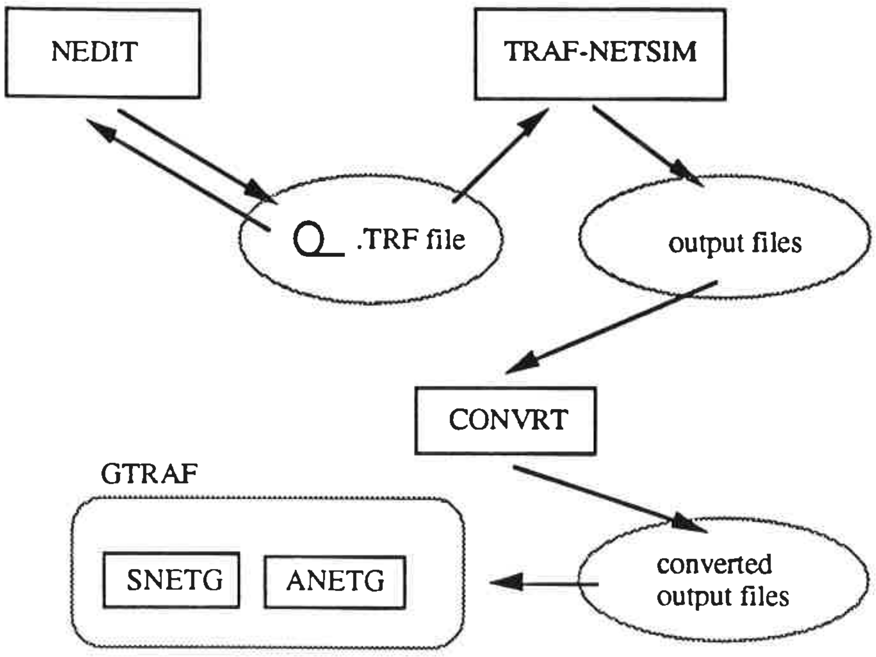


Fig. 1.5 Sequence of operations in a NETSIM execution within the TRAF environment.

1.6 ASTERIX

ASTERIX (DRIVE V1054 1992), which stands for 'A Simulation Tool for Evaluating Road Informati-X', is a software simulation environment embedding two assignment and macroscopic simulation models (SATURN and CONTRAM) and a microscopic simulation model (SITRA-B+). Being a modern software environment, all editors are graphical, and data is stored in relational DataBases. As shown in Figure 1.6, geometry and traffic parameters common to all three models are specified using a graphical editor, and stored in a 'Common Area' DataBase. In order to be used by a specific model, data must be transferred and recodified in a model-dependent DataBase. A 'Window Extraction' feature allows the transfer of only a restricted area (window) of the network. Then, additional data must be fed into these DataBases — for example, with a stream and movement editor in the CONTRAM area. To avoid modifying any of the three traffic models, data has to be transferred to a model-dependent file format before assignment or simulation can proceed. Similarly, results are transferred from file format to the model-dependent DataBase.

One advantage of this approach is the preservation of the original form of each assignment/simulation model. However, the data transfer process from the Common Area to the input of a model is impaired by the large number of steps involved. Results are stored back in their corresponding area, and thus are not available to the other models.

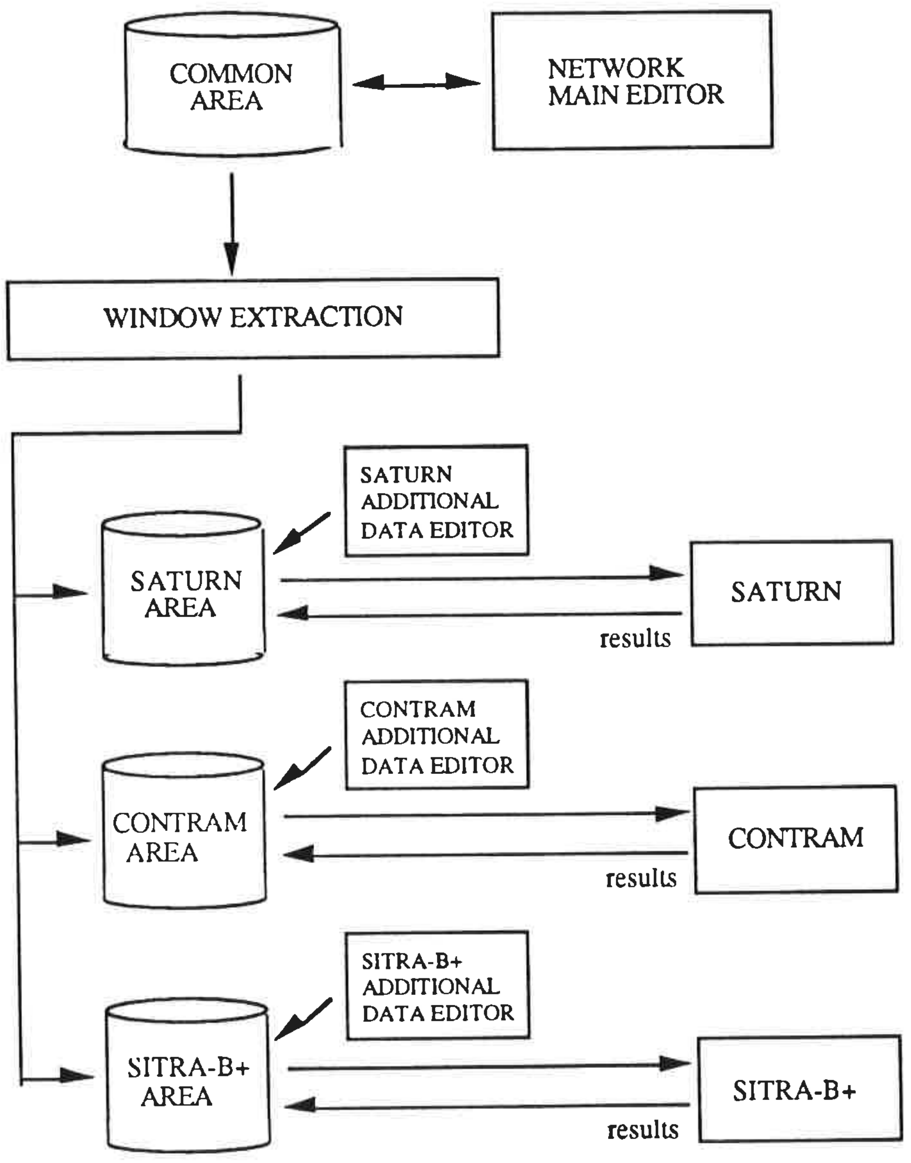


Fig. 1.6 Modules and DataBase structure in ASTERIX.

2. GETRAM approach

GETRAM's goal is to provide a framework in which various types of traffic models and analysis tools can be closely integrated, sharing a DataBase, a graphical editor and a results presentation. The DataBase can be either a relational system (INGRES) or, for portability purposes, ASCII files. Automatic recoding of a network from one format to another is provided. Subnetwork analysis and modeling is performed on the same DataBase instead of creating a new reduced one, and in any case, a model can, in any event, start from previous results of another model on the whole or part of the network.

The system supports both urban and interurban roads, which means that elements such as, for example, lateral lanes, entrance and exit ramps, junctions, and ramp meterings can be specified. The geometry of the links is specified at the microscopic level, but the ease of use of the editor makes it as fast as specifying unidimensional links in the previous systems, and nodes can be created automatically. Although this helps, data entry, especially the network geometry, is one of the hardest tasks in transportation planning. Therefore, the editor accepts, as a background, a graphical description of the network area, so that sections and nodes can be built subsequently in the foreground with that reference. This obviates the need to digitalize maps.

On a regional level of analysis, our approach provides an interface to any traffic assignment model using input data in a format similar, for instance, to the EMME/2 DATA BANK.

GETRAM's approach can be summarized as follows:

- It is designed to support the levels of detail required by regional, intermediate and local areas, with increasing modeling detail. For the regional level, an interface to EMME/2 or similar is provided.
- A simulator can start from the result of the traffic assignment or another simulation run.
- Once the network has been specified, the user can define a hierarchical tree of views so that a traffic model can be restricted to one of these views.
- It exports a user-friendly graphical interface; see Figure 2.1. A great effort has been made with regard to the ease of use: the user specifies the network employing realistic concepts such as sections, junctions, ramp meterings, etc., instead of constructing a wire-shaped image — links and unidimensional nodes — the final look of the network is very close to the reality. To this end, the editor accepts a background image that serves as a reference and avoids the need for a previous digitalization.
- It provides a common network editor to the assignment, macroscopic and microscopic simulators. The editor supports the geometries found in both urban and interurban roads.
- It gives a recommended 'Style guide' to the user interface of the simulators: the look of the window and dialogs. This style is reinforced by providing a skeleton window and a set of ready-to-use functions and Dialog Boxes which will facilitate the interaction with the user and access to the Data Base.
- It provides a common result presentation module.

- It accepts both static and dynamic assignment models.
- Simulators can start from a totally user-defined network state, thus allowing them to run in the absence of assignment results.
- A library of high-level, object-based application programming functions, named TDFunctions, is supplied to ease the task of integrating a new model within the environment, and, in general, to access any data. The TDFunctions allow the reading and manipulation of objects in the network, as well as restricting the view to a sub-area. Results storage and control plans are also accessed with these functions.

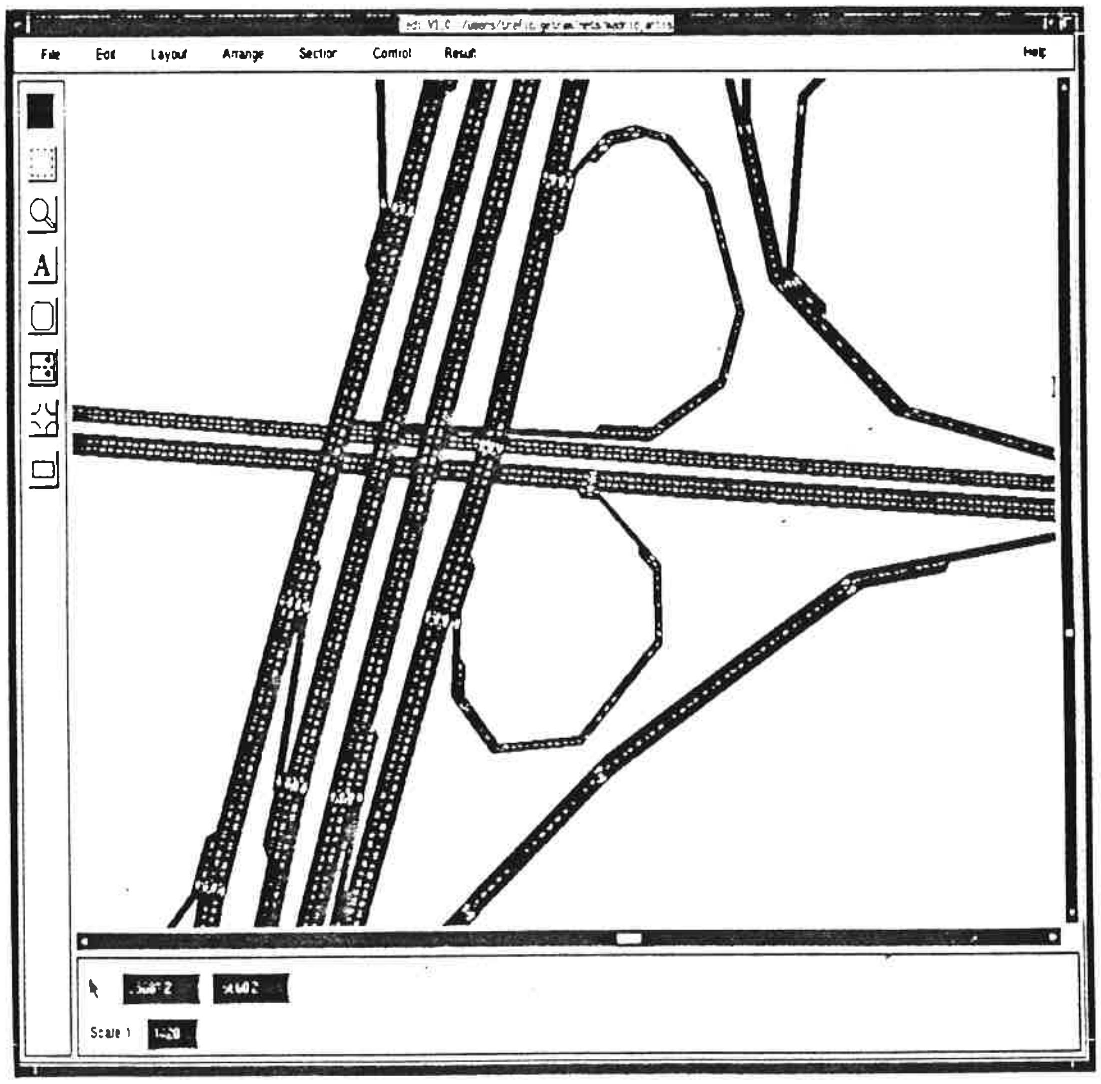


Fig. 2.1 A snapshot of GETRAM's graphical editor, showing a freeway area.

3. Targeted methodology

The approach adopted in GETRAM assumes that different levels of detail are necessary depending on the extension of the area and the required analysis. For example, for a regional area, an assignment model such as EMME/2 is often the best solution. Usually, as the area reduces, more detail is required and thus models range from macroscopic to mesoscopic and microscopic; see Figure 3.1.

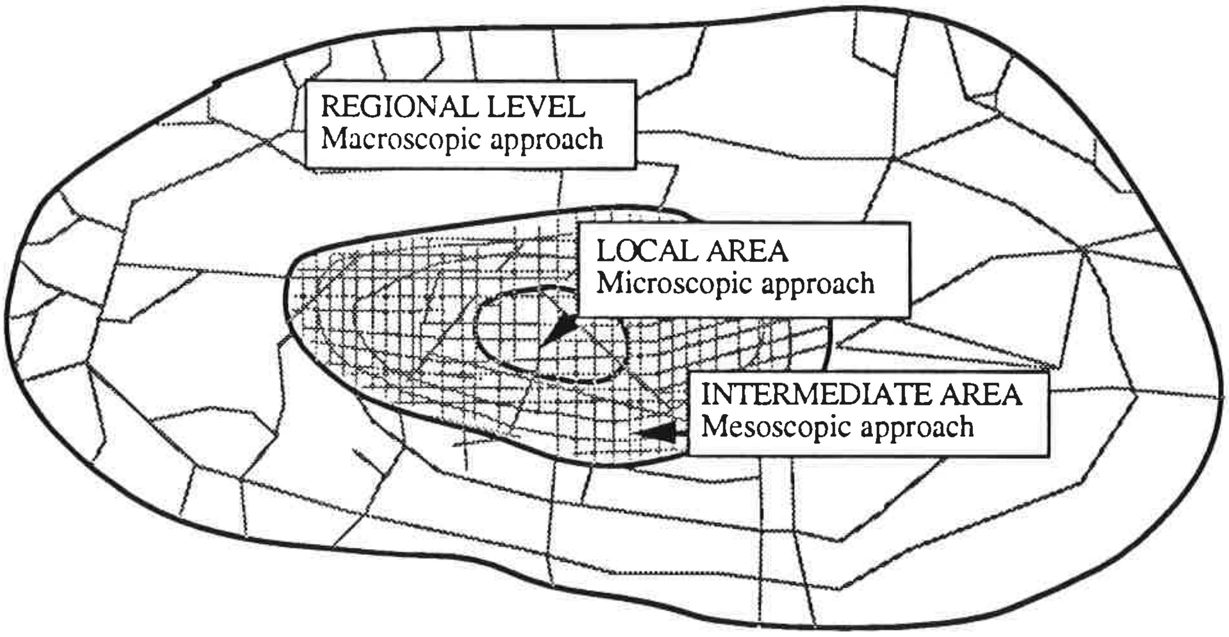


Fig. 3.1 Depending on the size of the network and the objectives, different modeling approaches are used.

The system has been designed to support the following methodology:

1- Specify the network geometry, traffic parameters and traffic demand (O/D matrix). Instead of requiring a previous network digitalization, the emphasis has been placed on accepting a graphical data exchange format, DXF, that acts as a reference background on which network edition builds up. DXF format is readily available from most government agencies or GIS vendors.

2- Specify a control plan for the intersections and ramp meterings.

3- Use an assignment model with an O/D matrix and a control plan on the network to obtain a state or sequence of states: turning percentages, entrance flows to the sections, and an approximate traffic state within sections. As already stated before, this task is performed through the EMME/2 system, or by using a compatible assignment model.

4- Apply a macroscopic simulation model on a subnetwork using the assignment results. Analyze incidents impact.

5- When more accuracy is needed, apply a mesoscopic or a microscopic simulation model, using the results of the assignment or a previous simulation.
6- Change the control plan or network characteristics and go back to 4 or 5.
7- Go back to 3 when changes may affect the assignment results considerably.

However, this does not mean that the system is tied to this methodology. Simulators are able to operate independently of the assignment results, the environment could be used with assignment models only, or macroscopic simulation models can be obviated— as it is now the case — by a sole microscopic model.

4. Global scheme

GETRAM (see Figure 4.1) will offer a series of common modules that can be used by any assignment/simulation model in the environment:

- A common DataBase — either ASCII files or a relational system— accessed through a library of high-level Application Programming Interface (API) functions that assure data consistency. This library facilitates both the access to the data and its geometrical display. It also provides view capabilities to restrict the analysis to a sub-area of the network.
- A common graphical network editor, that operates with microscopic detail, but with facilities to perform a fast specification where less accuracy is needed. Control plans, network states and view management are included.
- A common graphical results presentation, that takes data from both the Database and the results of a given model. Due to the considerable amount of data, network states and control plans are stored in files to avoid storage inefficiencies that arise in a relational DataBase.

All modules in GETRAM access the DataBase — and control plans, results, etc.— through the library of API functions, known as TDFunctions. As for the assignment models, data is converted to EMME/2 format, and this allows the use of all the matrix utilities and assignment tools provided by EMME/2. Assignment models other that those in EMME/2 can access to the Data Bank and produce their particular results. Simulation modules take as input a network view, and can store back the network states at the end of the execution, or at intervals during it, so that another simulator can start from one of them. For example, results from a macroscopic simulator can be used as a starting point for a microscopic simulator.

Since some networks might be available in EMME/2 format, an interface will be built so that those networks can be introduced into the DataBase — where the graphical editor can be used to improve geometrical accuracy, specify junctions, etc. — and on the basis of the network states produced in the assignment —EMME/2 or another assignment model— simulation models could be applied to some sub-areas.

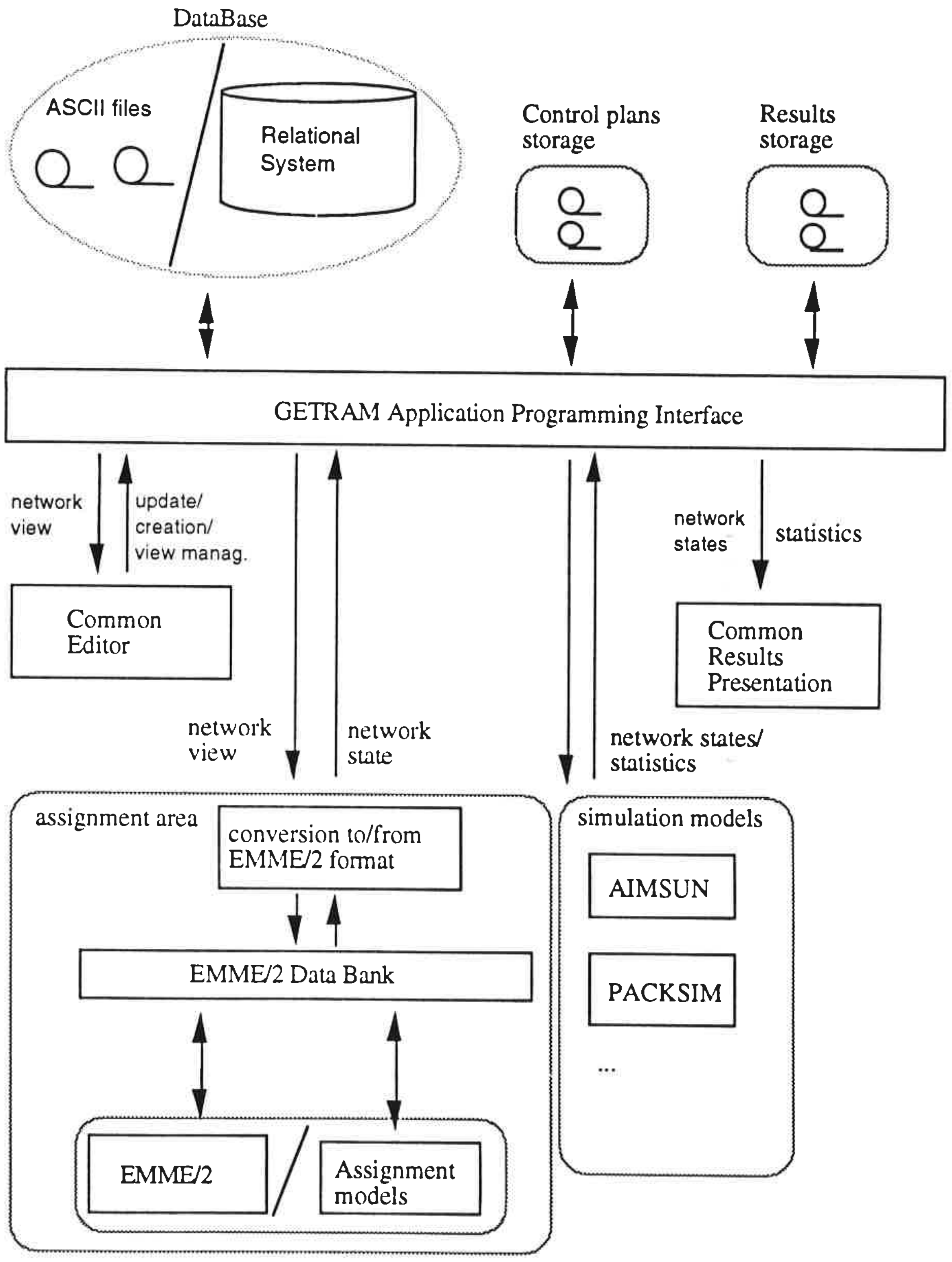


Fig. 4.1 Global schema of the GETRAM environment.

5. Basic objects

For the purpose of specifying the network under study — network geometry and movements, controller connections and so on — we distinguish certain basic objects that act as building blocks upon which more elaborate concepts — control plans, network states — are built. Basic objects are sections, nodes, controllers, classes of vehicles, and background objects (blocks), and they are all stored in the DataBase.

5.1 Sections

A section is a group of contiguous lanes where vehicles move in the same direction, as shown in Figure 5.1. The partition of the traffic network into sections is usually given by the physical boundaries of the area and the existence of turning movements. In an urban network, a section closely corresponds to the road from one junction to the next. In a freeway area, a section can be the part of the road between two ramps. Whichever the case, at the end of the section, vehicles may have to take certain lanes in order to be able to turn into another section. However, since turnings occur in the transition between sections, they are modeled within the node, which will be explained hereafter.

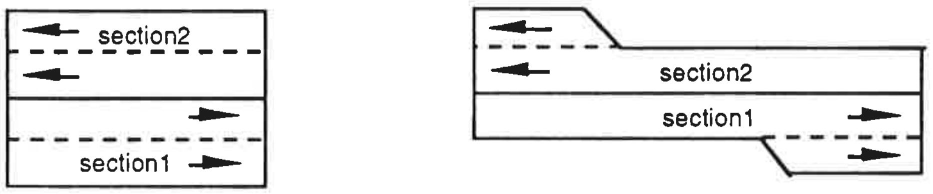


Fig. 5.1 Examples of sections.

We distinguish the entrance, exit, and right and left sides of a section, according to the vehicles' movement, as shown in Figure 5.2.

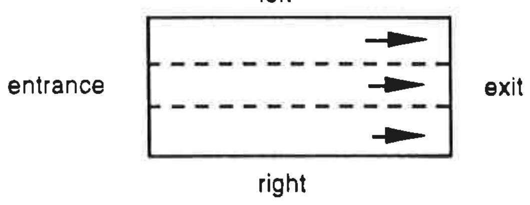


Fig. 5.2 Notation used in sections.

• Main and lateral lanes

Lanes in a section are classified as main lanes or lateral lanes. In a main lane, vehicles can enter from other sections and exit to other sections: it is both an entrance and exiting lane.

A lateral lane is positioned either at the right or the left side of the section, and is either an entrance or an exiting lane; see Figure 5.3. In an urban area, lateral lanes model turning storages at the end of a section before a junction; see Figure 5.4. In a freeway area, lateral lanes are used to model entrance and exiting ramps — acceleration and deceleration lanes, as well as lane gains and losses, as will be seen later when introducing the node. While a section cannot exist without one or more main lanes, lateral lanes are optional. A section cannot have more than one lateral lane per side.

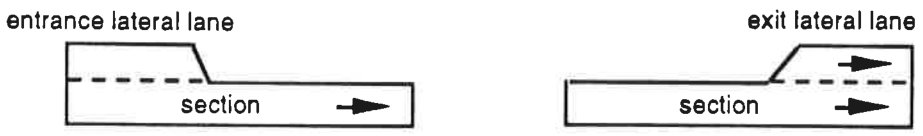


Fig. 5.3 Types of lateral lanes.

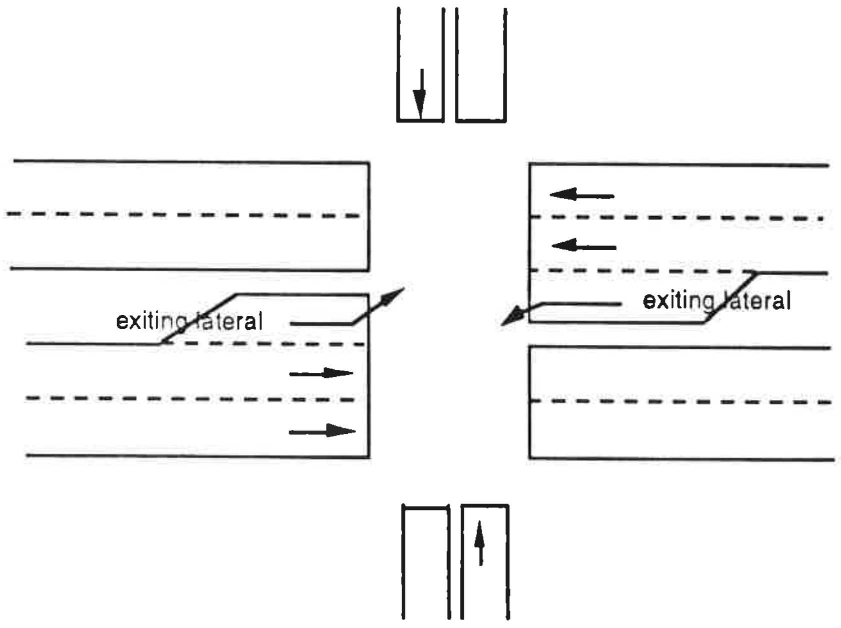


Fig. 5.4 Lateral lanes used for left turnings in a junction.

• **Lane identification**

Each lane in the section is identified with a number in the range 1..nblanes, from the right to the left of the section, where lateral lanes are also counted; see Figure 5.5.

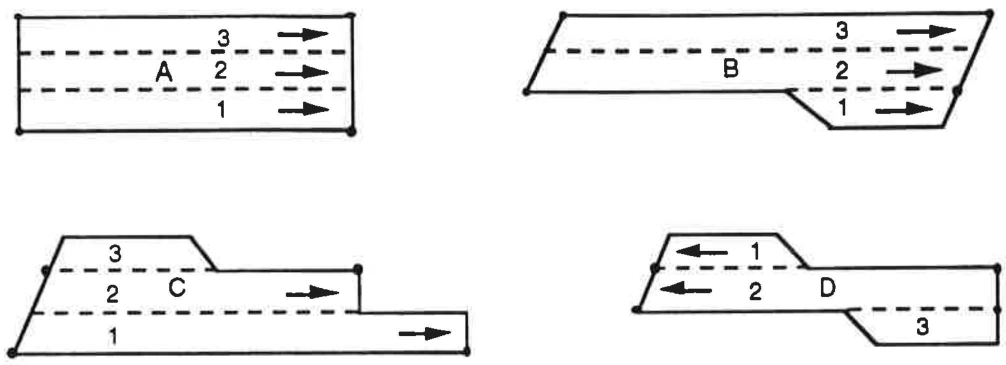


Fig. 5.5 Lane identification in a section.

• **Geometry specification**

Great care must be taken to describe the section geometry with the accuracy required in some traffic models while, at the same time, limiting the data storage requirements and, still more important, reducing the number of operations needed when translating or changing its geometry. Because of this, a section is specified, mainly, by the four coordinates corresponding to the four corners of the set of main lanes; see Figure 5.6. A 'stair exit' geometry, found in some junctions, can be specified on the exiting side; see Figure 5.7. A library of high-level functions deals with these details and gives the user the coordinates with the required detail.

Lateral lanes are characterized by their width and length, as shown in Figure 5.8. Again, it is the task of given functions to evaluate the exact geometry, accounting for the entrance and exiting slopes in the section's main lanes and stair exit.

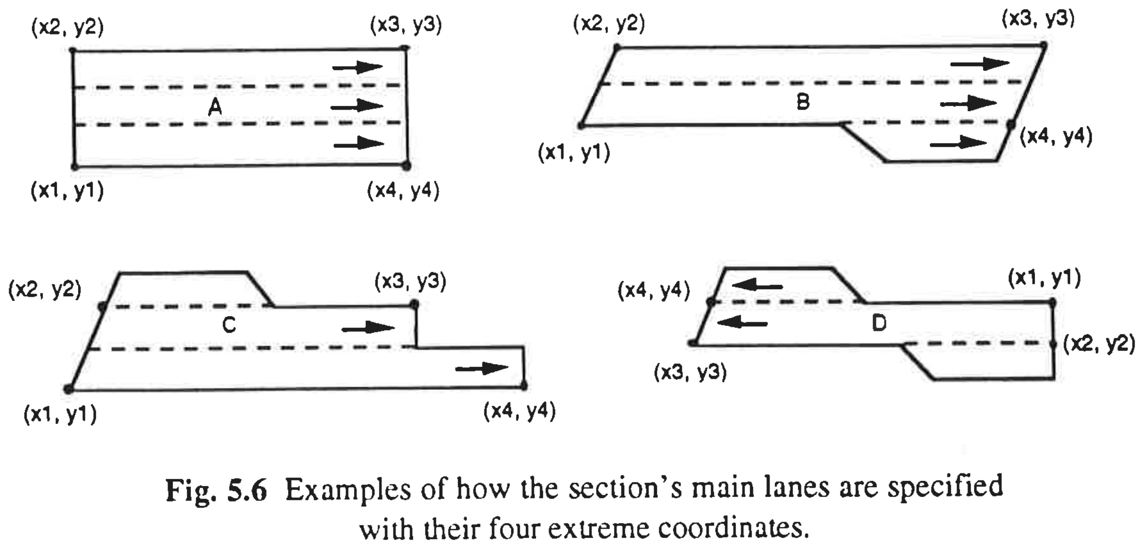


Fig. 5.6 Examples of how the section's main lanes are specified with their four extreme coordinates.

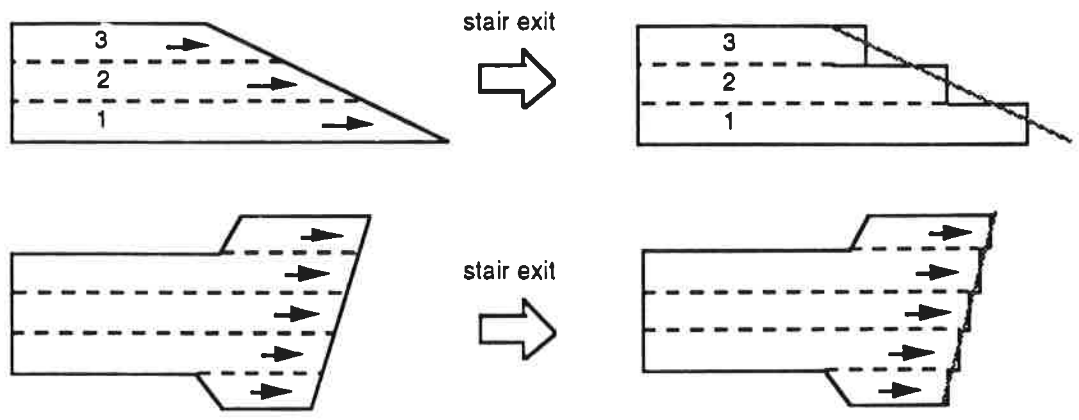


Fig. 5.7 An 'stair exit' shape can be specified for the section's exiting side.

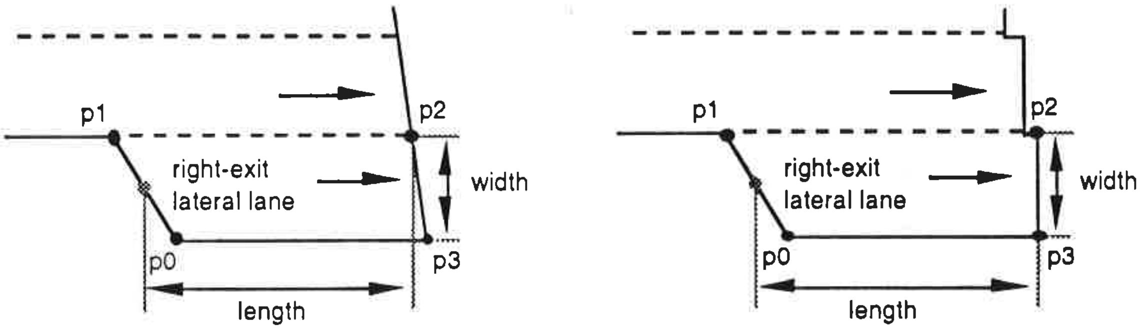


Fig. 5.8 The geometry of a lateral lane is specified by its width and length.

• Reserved lanes

Reserved lanes in a section are lanes where only one class of vehicles is allowed. A section can have only one set of reserved lanes, and it must be positioned either on the right or on the left side of the section. Reserved lanes can include both main and lateral lanes but must in all cases be contiguous. The most obvious example is bus lanes. Sometimes these lanes can also be used by taxis, and this can be specified by defining the 'public transport' class of vehicles to which the 'public bus' and 'taxi' vehicle modalities pertain. Classes of vehicles and vehicle modalities are explained hereafter.

• Detectors

Multiple types of detectors exist: pressure, magnetic, loop, video, and others. Measuring capabilities vary: count, presence, speed, occupancy, and queue length. Explicit distinctions are made between inductance loops, pneumatic tubes, coaxial cables, tape-switches, and queue-length detectors. New detector types can be added by specifying their measuring capabilities.

A detector can be positioned at any point in a section, and its width can extend to more than one lane. A section can have no, one, or multiple detectors. The only restriction is that a detector cannot be shared between sections. Figure 5.9 shows some positioning examples. In Section A, Detector b is used to detect right-turning vehicles. Section B shows multiple detectors on a section, this being possible in practice with a video detection system.

The detector's position in a section is specified by the range of lanes it ocupies, the distance to the first main lane's coordinate, and the length; see Figure 5.10.

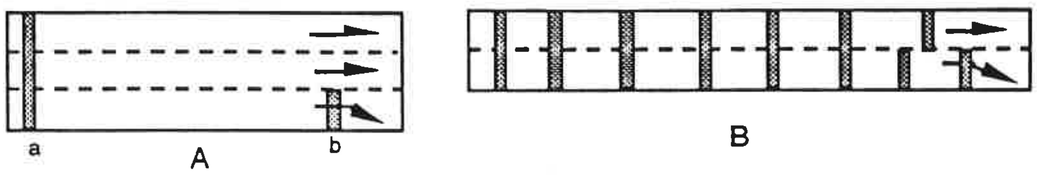


Fig. 5.9 Examples of detector positioning in a section. In Section A, Detector b is used to detect right-turning vehicles. Multiple detectors as in (B) are feasible using videodetection.

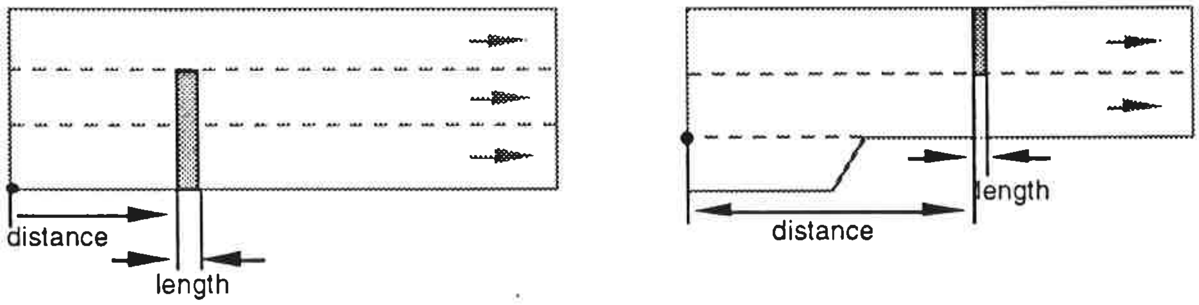


Fig. 5.10 Examples of how the detector's position is specified.

Figure 5.11 shows an example of detector positioning in a ramp metering, showing a notation which is often used:

- queue detector: to detect a queue before it blocks the upstream section.
- check-in detector: to detect vehicles waiting.
- check-out detector: to adjust the green time to the desired platoon length (can be directly connected to the ramp metering)
- merge detector: to detect vehicles waiting to merge, and thus inhibit the green time in the ramp metering.
- main line detector: to detect the vehicles in the freeway and thus adapt the control policy.

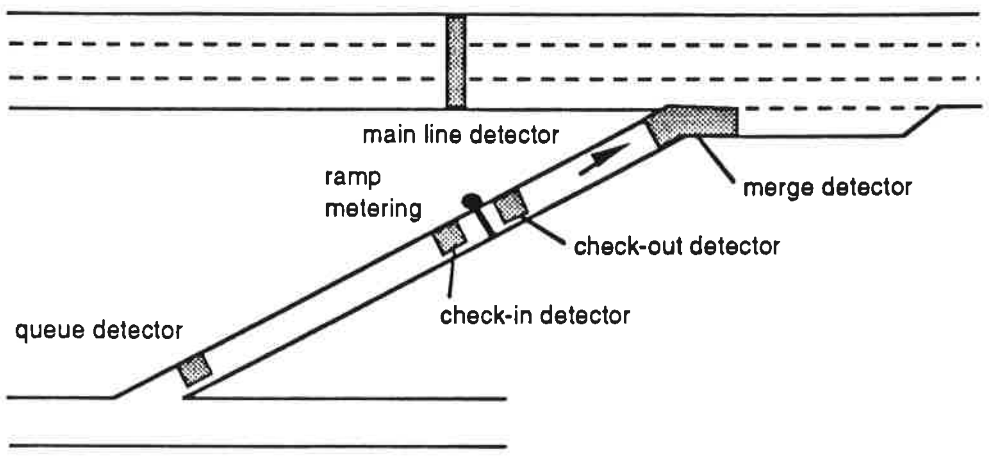


Fig. 5.11 Example of detector positioning in a ramp metering.

• **Metering**

Meterings are used on entrance ramps to a freeway in order to limit the flow of vehicles entering, assuring a not-congested flow on the freeway itself. Figure 5.12 shows an example in a single-lane section. Ramp metering can have several implementations: as a normal semaphor, flashing light, etc.. We distinguish two types:

- Green-time metering. Usually implemented as a semaphor, can be characterized with a cycle time and a green time duration. This is being implemented mainly in Europe.
- Flow metering. It controls the maximum flow of vehicles that can pass through the signal. An important parameter in a flow metering is the number of vehicles that are allowed to pass each time (normally one, but occasionally two). This is assumed to be fixed throughout the history of the device.

A metering can be positioned at any position along the section, all exit lanes are affected, and only one metering per section is allowed.

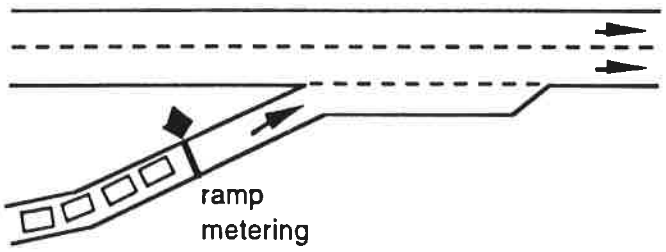


Fig. 5.12 Metering in a lane.

As will be seen in the junctions, we distinguish three types of control policies:

- uncontrolled policy: the metering device is assumed to be disconnected.
- fixed policy: a maximum flow of vehicles is allowed to pass.
- autoadaptive policy: the initial maximum flow of vehicles can vary between two values.

In some situations, a selective metering is used, i.e., an adjacent lane with no metering where only certain selected classes of vehicles (for example, public vehicles) are allowed to pass. This is modeled by considering the unmetered lane as a different section. Figure 5.13 shows an example: vehicles that are not metered turn from section 1 to section 3.

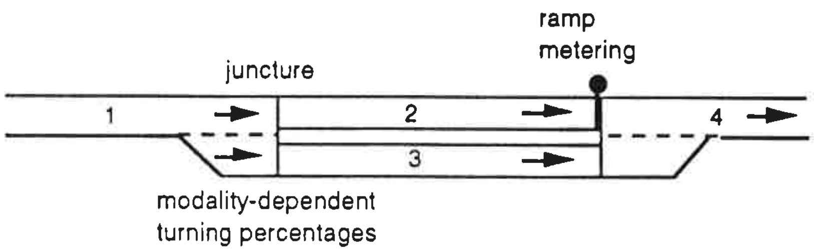


Fig. 5.13 Modeling of a selective metering.

• **Variable message signs**

Variable message signs (VMSs) are used to inform drivers of regulations or instructions that are applicable only during certain periods of the day or under certain traffic conditions (Doughty 1982). First implementations allowed only for the on-off switching of a single message. Currently they are more flexible and allow for the selective display of a number of messages.

We model a VMS as being located in a given position in a section (see Figure 5.14) and having a set of feasible messages. The currently displayed message will be specified in the control plan.

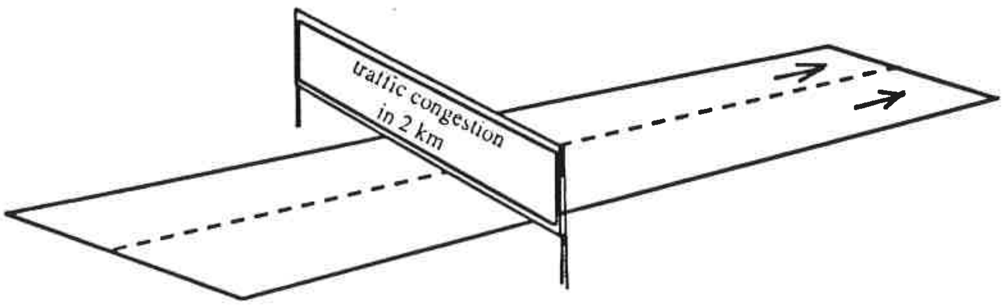


Fig. 5.14 A VMS in a section.

5.2 Nodes

A node is a point or an area in the network where vehicles change their direction and/or disperse in other directions. Hence, a node has one or more origin sections and one or more destination sections. We distinguish two types of node: joint nodes (junctures) and junctions. The difference between them is that, while in a junction there is a space between the origin and destination sections, in the joint node there is no space between these two sections. In a joint node, the number of origin lanes equals the number of destination lanes. Junctions are often found in arterials and streets; joint nodes, in roads and highways.

The generic concept of node, including both junctions and junctures, is useful for assigning a common identifier, and characterizing both views and origin/destination zones.

• Joint nodes

A joint node, also called juncture, is a type of node which differs from the other type, the junction, in that entrance and exiting sections are directly connected, i.e., the origin lanes of a turning are directly connected to the destination lanes. Hence, when a juncture is completed, the number of intervening origin lanes equals the number of destination lanes.

Junctures are often found in freeway areas, see Figure 5.15, to model bifurcations (A), exiting ramps (B), mergings (C), and entrance ramps (D). Figures 5.16 show how various situations can be modeled.

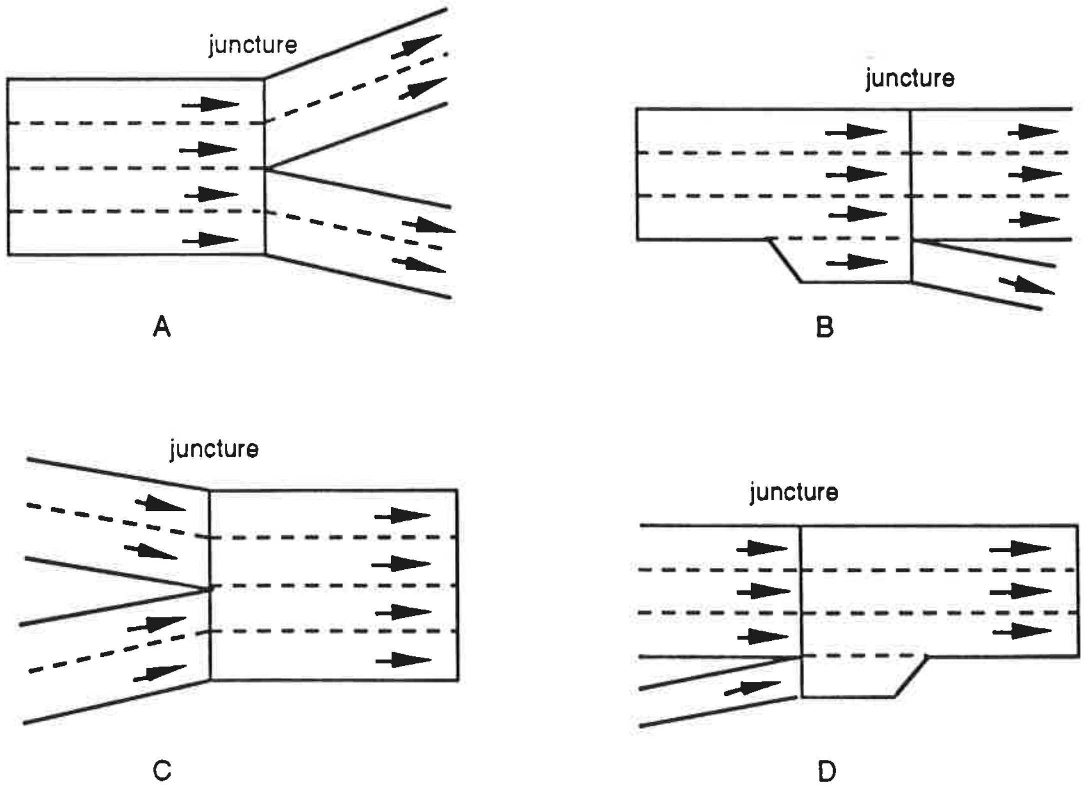
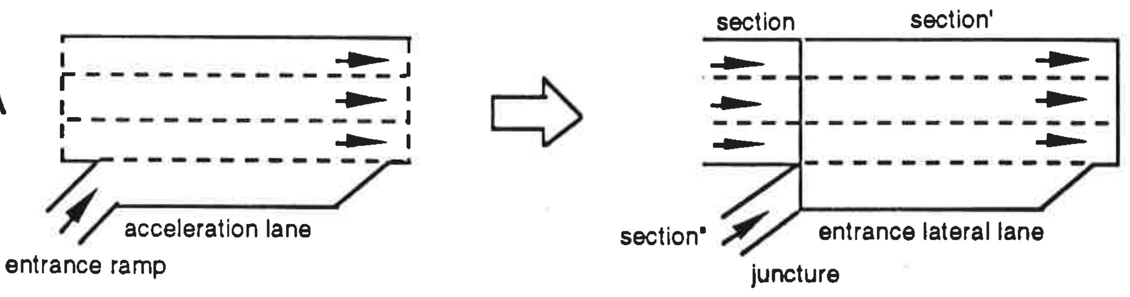
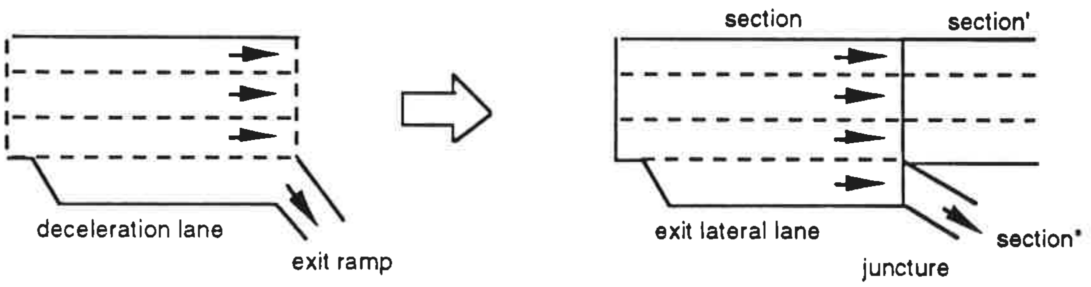


Fig. 5.15 Examples of junctures

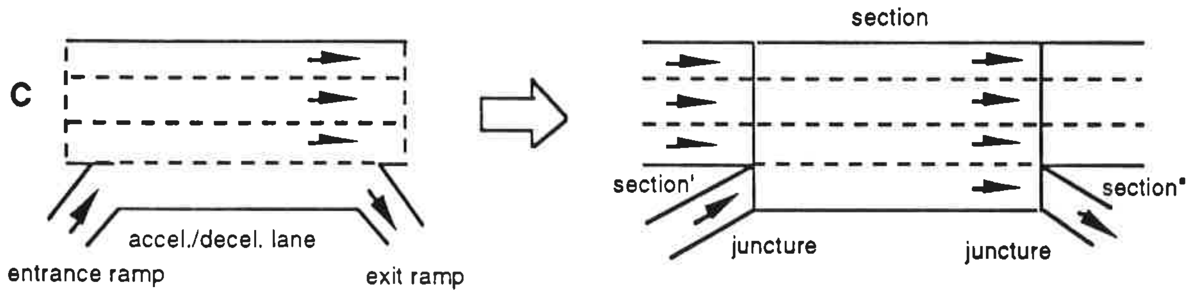
A


acceleration lane
entrance ramp
section
section'
section"
entrance lateral lane
juncture

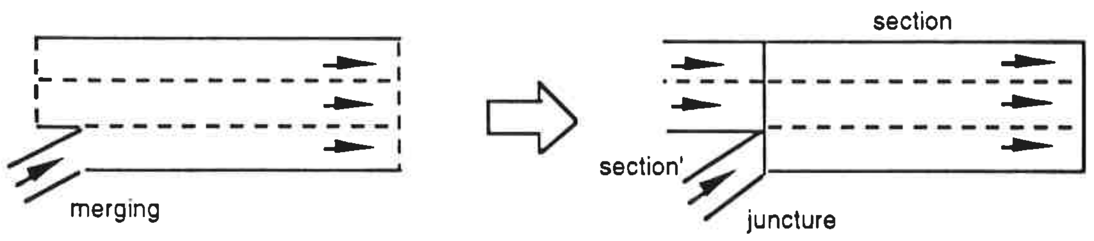
B


deceleration lane
exit ramp
section
section'
exit lateral lane
section"
juncture

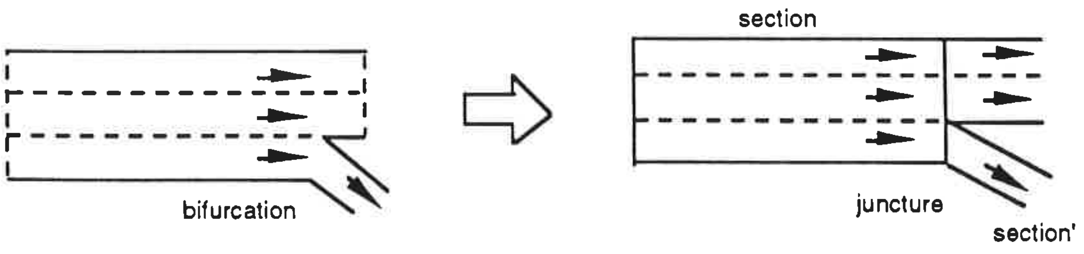
C


accel./decel. lane
entrance ramp
exit ramp
section
section'
section"
juncture
juncture

D


merging
section
section'
juncture

E


bifurcation
section
juncture
section'

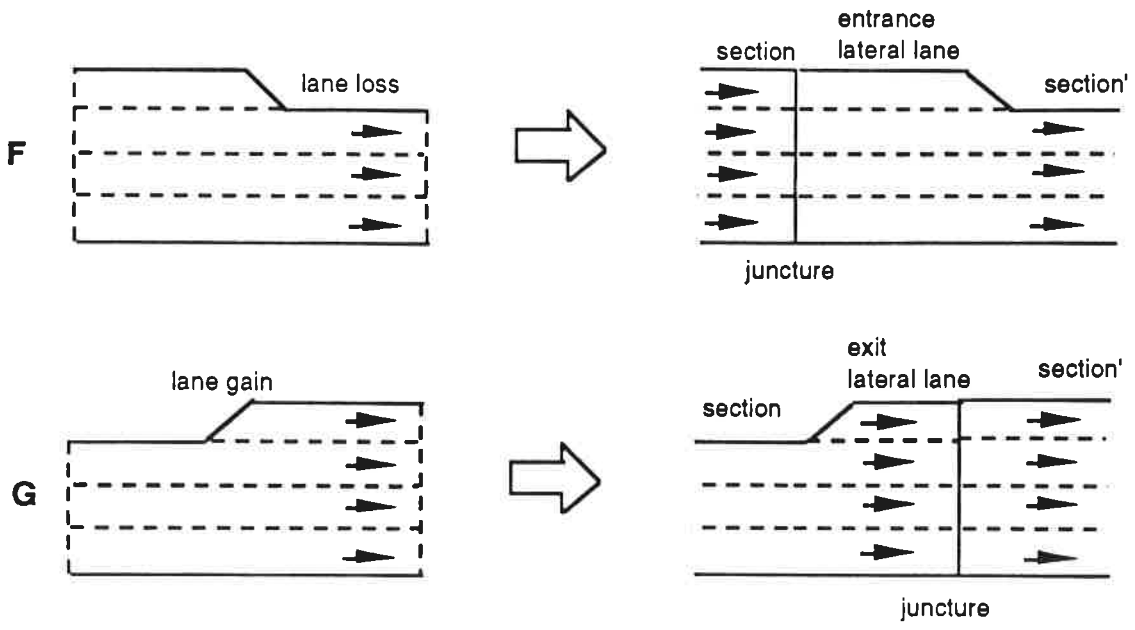


Fig. 5.16 Examples of modeling roads with sections.

• Junctions

As stated above, a junction is a type of node where there is a space between the origin and destination sections, as was shown in Figure 5.4. Junctions are often found in arterials and streets.

Depending on the control policy applied, we distinguish three types of junctions: unsignalized, fixed controlled, and demand-responsive controlled. Turnings occur within a junction and the user can specify give-ways between them. Conflicts between turnings arising in unsignalized junctions can be solved by imposing give-ways. In signalized junctions a sequence of stages is designed to avoid conflicts, but give-ways can also be specified.

Signal lights at a junction are connected to a controller that sends signal light changes according to a pretimed (fixed controlled) or demand-responsive strategy. Controllers will be explained hereafter.

• Turnings

Vehicles turn to pass from one section to the next. Turnings occur within a node, be it a joint node or a junction. In the first case, a joint node, there is a direct connection between the two sections, from certain lanes in the origin section to certain other lanes in the destination section — origin and destination lanes are given by the node geometry. Instead, in a junction, vehicles enter first into the junction area before entering the destination section (by any of its entrance lanes), see Figure 5.17. Therefore, at the turning of a junction, origin lanes are not clear nor are vehicles restricted to entering the destination section by a lane subset, and therefore both origin and destination lanes have to be specified.

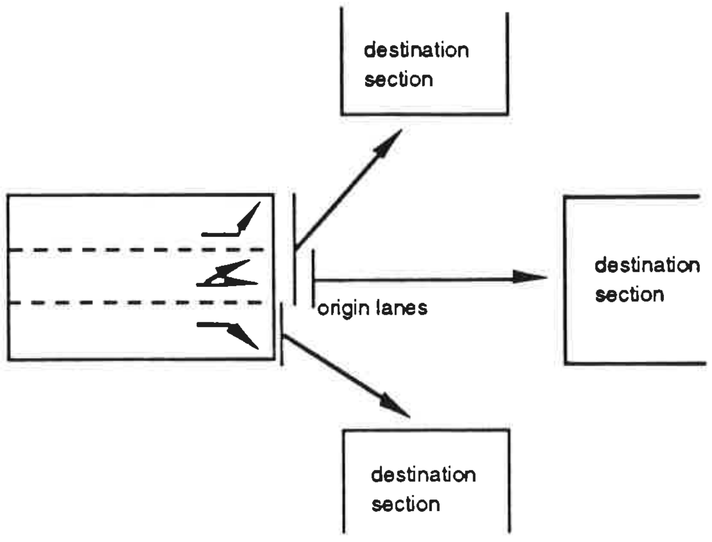


Fig. 5.17 Notation used in turnings.

- **Give-ways**

A give-way is a conflict between turnings where one of them has less priority than the others. Give-ways occur within a junction, usually unsignalized. They are specified as lists of higher-priority turnings, to which the lower-priority turning must give way. Figure 5.18 shows a junction where the vehicles turning from Section 2 to Section 1 have to give way to the vehicles turning from 4 into 1 and from 3 into 1. Therefore, the user would specify the turnings 4-1 and 3-1 as having more priority than 2-1.

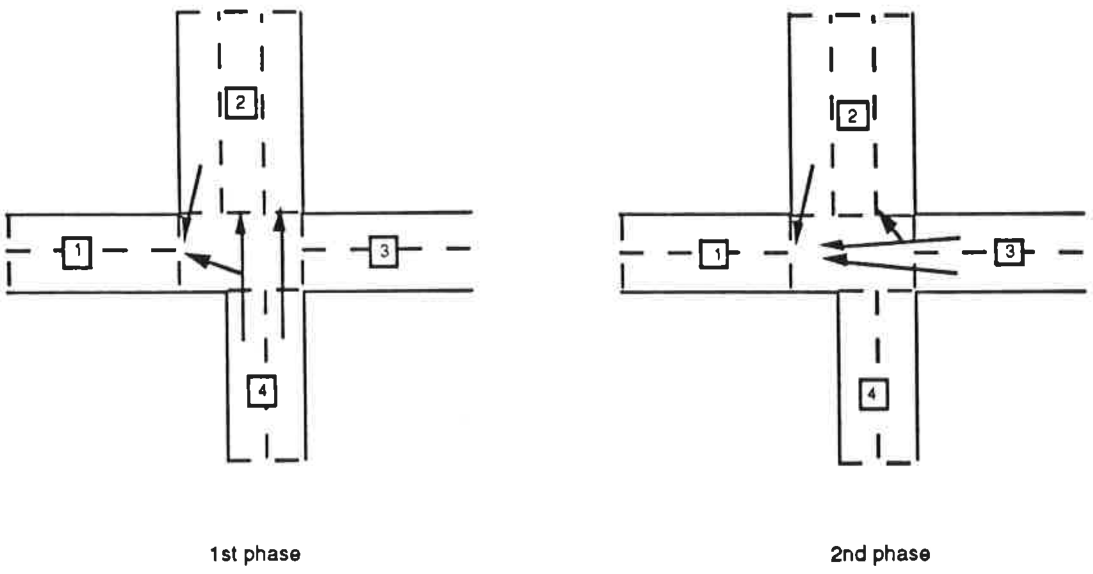


Fig. 5.18 Junction with give-ways.

• **Sequences of stages**

In signalized junctions — fixed or demand-responsive controlled — a stage-based approach is applied, in which the cycle of the junction is divided into stages, each one having a particular set of turnings with right of way. Among these stages, we distinguish between phases and interphase periods. Figure 5.19 shows an example of rights of way sequence, and Figure 5.20 shows its stage modeling. In demand-responsive control, the duration of a phase is allowed to vary between minimum and maximum values.

Progression arterials, each one constituted by a list of contiguous junctions, can be specified. This might be used by a signal-plan algorithm to find a sort of maximal-bandwidth control on these paths.

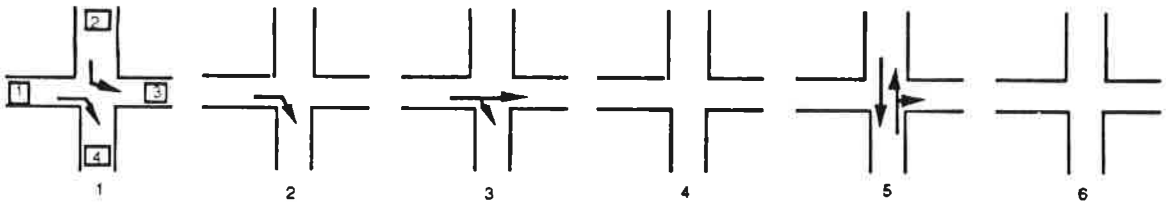


Fig. 5.19 Example of sequence of rights of way in a junction.

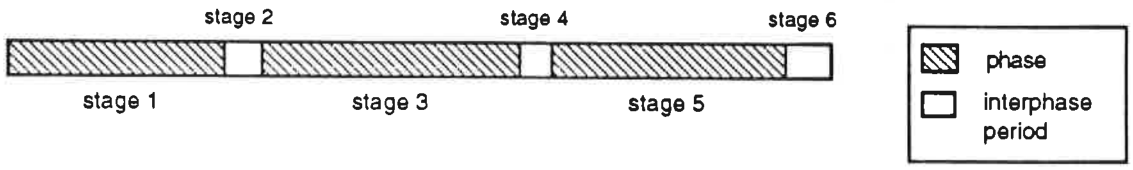


Fig. 5.20 Stage modeling in the junction in Figure 5.18.

5.3 Controller

Devices such as semaphors, meterings, detectors, and VMSs are usually connected, in the real world, to controllers. Besides sending and receiving electrical signals to and from the devices, depending on the purpose, controllers store data and communicate with a control center.

In our modeling, controllers are optional: a simulator can bypass them and work directly with junctions, meterings and detectors. Nevertheless, controllers offer a way to represent real-world connections or to group devices conveniently in the network. A controller can be connected to:

- n junctions: the controller sends the phase endings.
- m meterings: the controller sends the metering policy.
- p detectors: the controller receives measurements from the detectors.
- q VMSs: the controller sends the message to display.

Each connection is characterized by three fields:

- Type: junction, metering, or detector.
- Connection identifier, within a given type of device in the controller.
- Activate/Disactivate flag.

Figure 5.21 shows an example of connections in an urban area, while Figure 5.22 shows connections in a ramp metering. There is no restriction on the physical location of the devices connected to a controller. For example, a controller connected to a junction could be connected to detectors that are not positioned in entrance or exit sections to this junction.

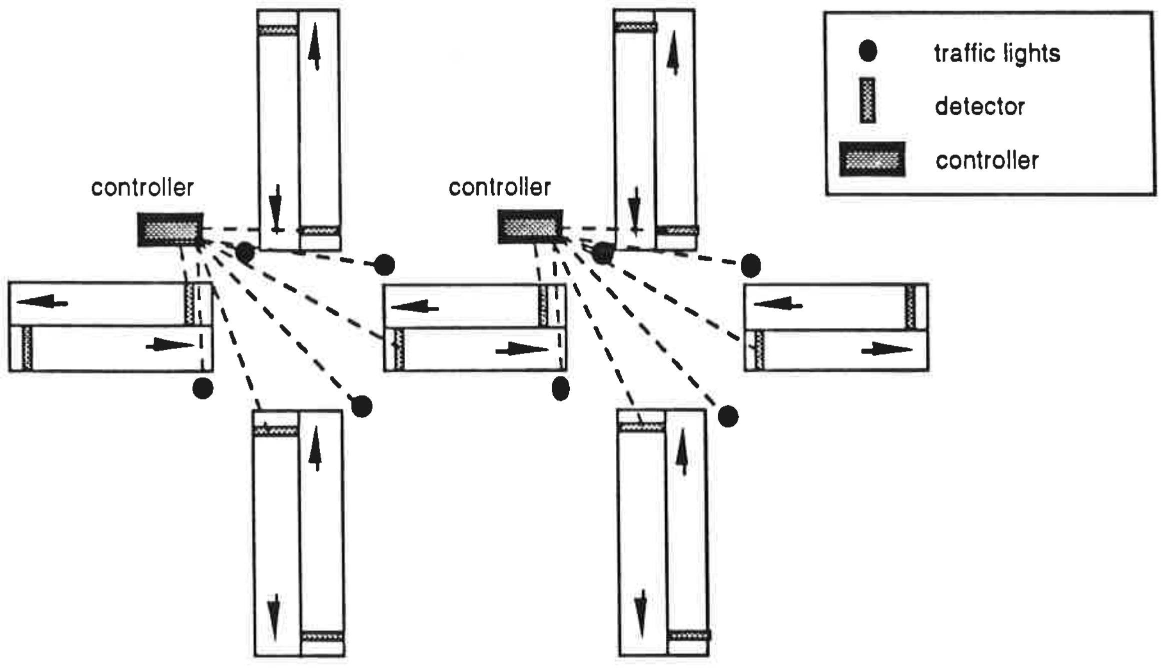


Fig. 5.21 Example of controller connections in an urban area.

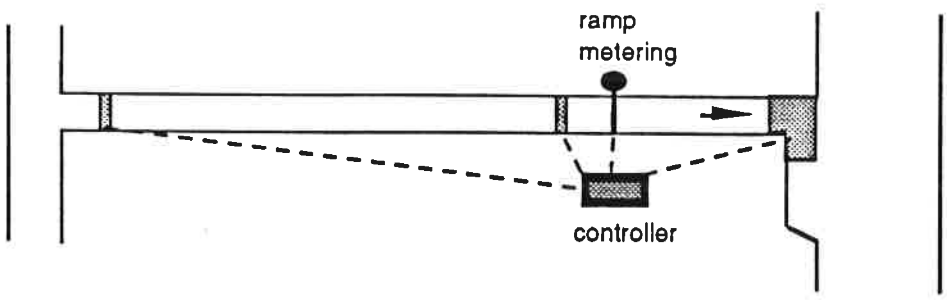


Fig. 5.22 Example of controller connections in a ramp metering.

5.4 Classes of vehicle modalities

Classes of vehicle modalities are used to provide a flexible way to relate selective authorization in certain lanes of the network and generic vehicle characteristics to the actual vehicle modalities used in the simulations.

For example, the user might have to specify, in some sections, lanes that can be used only by public transport. This can be done by declaring the vehicle class 'public transport' and assigning selective authorization to this class in those lanes. Then, before starting a simulation, vehicle modalities are specified and three of them — long buses, short buses, and taxis — could be declared as pertaining to the 'public transport' class. Subsequently, another simulation could be started by using only long buses as public transport. This is a more convenient mechanism than having to re-specify, in a large network, all the selective authorizations that are affected.

5.5 Background objects

Background objects are used to write texts and represent the framework of complementary equipments — squares, blocks of buildings — in which the traffic network is embedded. They may be convenient to clarify the network view, adding reference points which aid recognition of the area to be modeled; see Figure 5.23.

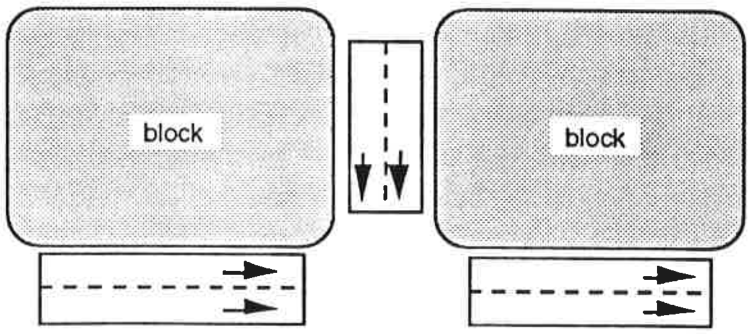


Fig. 5.23 Blocks of buildings can be displayed together with the modeling elements.

6. View restriction

A view is a polygonal area of the real - world. GETRAM allows the user to specify a view hierarchy, i.e., a tree of views where a view contains its lower-level subviews. This has been found to be convenient when analyzing large networks, and applying the regional-intermediate-local strategy introduced in **4**: on the basis of the results from an assignment model, a sub-area may need a more detailed analysis using a macroscopic model, and, starting from those results, a microscopic model in a smaller sub-area.

A maximum of ten levels are allowed in the view hierarchy. Each view is characterized by a polygon, and contains all the objects (sections, nodes, controllers, etc.) within. A view can be totally or partially superposed on another view at the same hierarchy level. Figure 6.1 shows an example of a network with views.

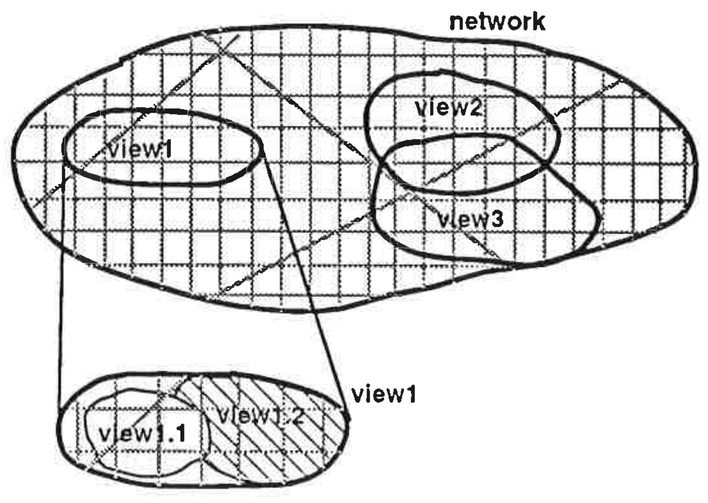


Fig. 6.1 A network where some restricted views have been specified. Views 1.1 and 1.2 are subviews of view 1.

An important point in this view management is that control plans and network states can be specified at a view level, see Figure 6.2 —as it will be seen, network states are stored within results containers. This allows, for example, specifying a control plan for the whole network and then, as a subarea analysis demands it, modify the control plan in that subarea and save it within that restricted view for use by a more detailed model. Similarly, a network state from a higher-level view can be modified and saved within a current view restriction. The main advantage of this is that control plans and network states can be organized more clearly according to the sequence of analysis.

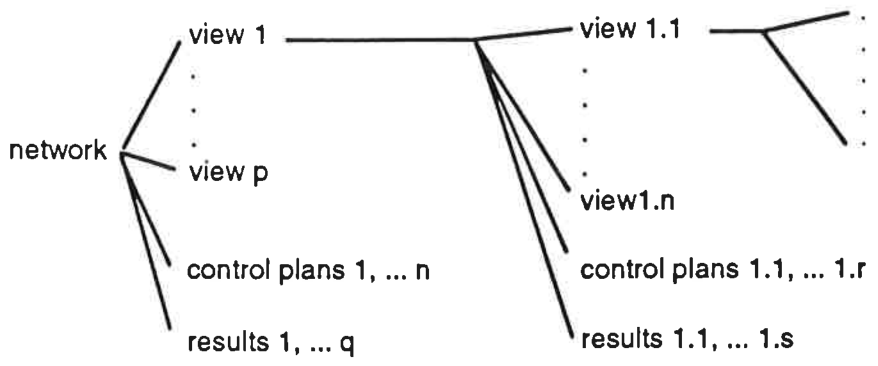


Fig. 6.1 Hierarchy of views in a network. Network states are stored in results containers.

7. Control plans

A control plan specifies the control parameters applied to each junction, metering, and VMS in an area — either the whole network or a restricted view. Then, a simulation model considering a restricted view of the network can use either a control plan specified at this same view, or at a higher level in the hierarchical tree of views.

Parameters in a junction are:

- Control type: unsignalized, fixed control, or demand-responsive control.
- Critical junction flag: a critical junction could be considered differently in a demand-responsive model.
- Absolute offset: used to set the beginning of the sequence of stages, and therefore synchronize adjacent junctions.
- Sequence of stages: number of stages and, for each one, type — interphase period or phase — initial, minimum and maximum durations.

Also specified in a control plan are high-priority routes. A route is a list of sections, in contiguous or non-contiguous order. A priority level from 1 to 10 can be specified for each route, 1 being the lowest and 10 the highest priority. A section can belong to more than one route. The use of these high-priority routes can vary among the models and systems connected to GETRAM, depending on the flexibility of their control methods. As will be seen in the next part, CARS V2 uses this concept to implement progression arterials, i.e., routes where a progressive signalization is desired.

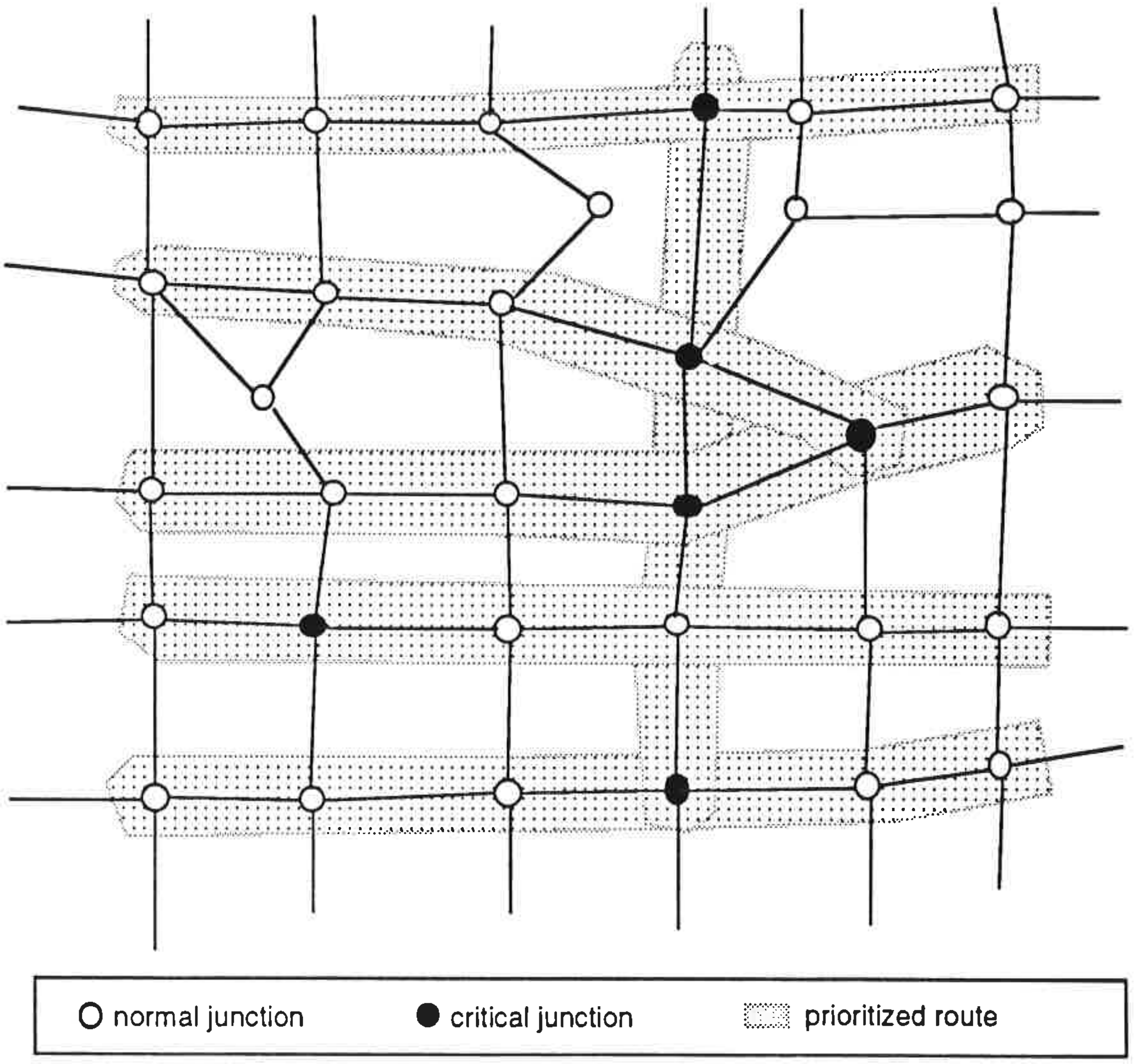


Fig. 7.1 Example of critical junctions and high-priority routes.

Parameters in a metering depend on whether it is a green-time or a flow metering. In the first case, a green-time metering, the parameters are:

- Control type: unsignalized, fixed control, or demand-responsive control.
- Cycle time: The duration of a sequence of green-time and red-time — a cycle in the control.
- Green-time duration: initial, minimum and maximum.

In a flow metering, the following parameters must be specified:

- Control type: unsignalized, fixed control, demand-responsive control.
- Flow metering: initial, minimum and maximum flows allowed.

The only parameter to be specified in a VMS is the identifier of the displayed message — a zero identifier meaning that no message has to be displayed.

An important issue is how to deal with changes in the network. When the network configuration changes — for example, a new junction is added to the network—then the previously stored control plans may be incomplete or inconsistent. The usual approach to this problem requires the user to modify all the control plans that had been specified — independently of whether they are stored in a DataBase. Our approach is to signal inconsistencies at the time a control plan is open, requiring the user to supply consistent values for those junctions and meterings that have changed.

8. Results containers

Results from assignment or simulation models are stored in results containers. Among the results that a model can produce, network states — i.e., the state of traffic in the network at certain intervals during the execution — can be used by other models as a starting state.

A results container is characterized as having been produced in a network area — the whole network of a restricted view — the control plan, and the vehicle modalities that have been used. As said before, results containers can include network states produced in that network area. A subsequent execution of a traffic model in that area can use network states stored in that results container or use other ones produced in a higher-level view.

A results container consists of three parts:

- Header: contains the number of seconds since the beginning of results generation (in relation to the reference time 00:00:00), the name of the control plan that was used during the execution, and a time offset applied to all the junctions' offsets. All this information is optional.
- Vehicle modalities: Specified as types of vehicles with certain physical measurements:
  - length
  - width
  - mean desired speed
  - maximum acceleration
  - maximum deceleration

  Also specified is a set of classes, as in **5.5**, to which each vehicle modality belongs. For example, a 'bus' vehicle modality can have selective authorization in some lanes because of belonging to the 'public transport' class. Another example, the 'guided taxi' modality belongs to the classes 'public transport' and 'guided'.

  There also exists a library of vehicle modalities used in the common editor in order to easy the creation of new vehicle modalities in the network, or store them for future use in other networks. The library is independent of the network being edited.
- Network states: Usually generated as the result of a previous run of an assignment or simulation model, each network state contains a time offset, and a description of the traffic state in each section in the network view. The time offset is measured in relation to the reference time 00:00:00, also used in control plans to measure the absolute offset in the cycle of a junction. The traffic state in a section is characterized by the following averaged measurements:

- Per each previous turning and vehicle modality, the entrance flow.
- Per each previous turning, vehicle modality, and turning: the average speed, total number of vehicles, number of vehicles stopped, and turning percentage.

The user is recommended to name network states after the style 'hh:mm:ss' where hh, mm, and ss express the time in hours, minutes and seconds since the beginning of the execution in which this state has been generated. This can make these files easier to handle, although an application is free to assign other names which could state particular situations during the execution, such as 'atypical_congestion'. The network state includes redundant information like section identifiers, number of vehicle modalities, turnings, entrance points, etc., to help solve inconsistencies which could be produced when the network view has been slightly modified after the generation of results.

As in the control plans, changes in the network are dealt with by signaling inconsistencies at the time a network state is open, requiring the user to supply consistent values for those sections.

9. GETRAM's API: TDFunctions

GETRAM's Application Programming Interface, TDFunctions is essentially a set of high-level access functions to the DataBase, although there are also a considerable amount of functions to aid the programmer in the task of displaying, graphically, the objects in the network. A complete description is available elsewhere (Grau 1994), therefore the following document only outlines them.

Modules in the GETRAM environment should access the DataBase through these functions providing the following advantages:

- The ease the task of a graphical editor from the DataBase, while still being convenient to the other modules. The objects are open, manipulated, and closed, following a fairly close match with the dialog boxes usually provided by more recent editors. Thus, the functions could be termed object-and-user-dialog based. Some functions are provided to specifically address the requirements of geometrical representation.
- To maintain the consistency of the objects in the DataBase. Generally, only one object can be open at a time (the only exception being background elements), and has to be closed after manipulation. Three closing modes exist: cancel, update, and delete. When closing to update, the functions check for consistency and the object is updated if consistency is maintained. In either case, the functions automatically execute cross-references between objects.
- To provide a shortcut for applications only wishing to read data. In this case, opening and closing an object is not necessary.
- Format-independent DataBase access: A graphical editor or a model using these functions does not need to know in which format the DataBase is stored —either ASCII files or a relational system.

- Easy-to-use view restriction. Once a view restriction is set up, a module only 'sees' the objects in that sub-area, so that a simulation model reading the network does not need to act differently depending on wether it is the whole network or a sub-area: control plans, turnings and network states are automatically passed to the user conveniently. Similarly, a module cannot modify an object that does not pertain to the current view.

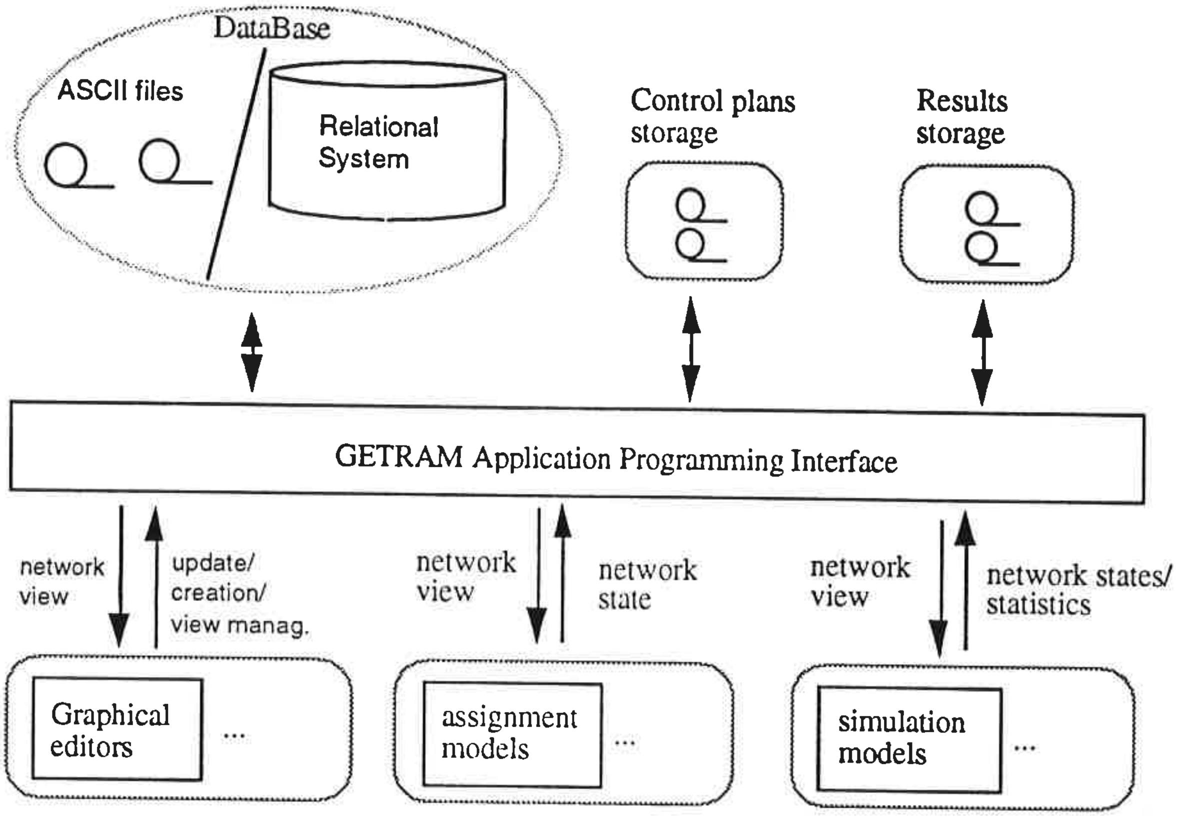


Fig. 9.1 The TDFunctions module is intended to provide a convenient access to the network representation through the other modules.

TDFunctions distinguish the following objects: networks, background objects (blocks), classes of vehicle modalities, sections, nodes, controllers, control plans, results containers, and network states. An object has to be previously open before any update operation can take place on it. Also, some interrelations exist between objects, as shown in Figure 9.2; when opening an object, the higher-level object(s) have to be already open:

- The network must already be open before accessing or updating any of its objects (sections, nodes, etc.). Only one network can be open at a time.
- Within a network, we distinguish between background objects and network objects. Blocks are background objects, while classes of vehicle modalities, sections, nodes, and controllers are network objects. Only one background element can be open at a time. Equally, only one network object can be open at the same time. Although more elaborate object hierarchies have been considered, such as the ramp meterings and

detectors being subobjects of the section, the current approach is the most suitable as regards database consistency.

- Control plans and results containers are a special kind of object because, as has been seen, they are not actually stored in the Network DataBase but in associated files. This is due to the extreme storage inefficiency that would occur in a relational system. However, consistency in these objects is checked when opening them. Therefore, the network has to be previously open. Network states are subobjects of a results container. It is recommended to avoid updating the network while a control plan or a results container is open.

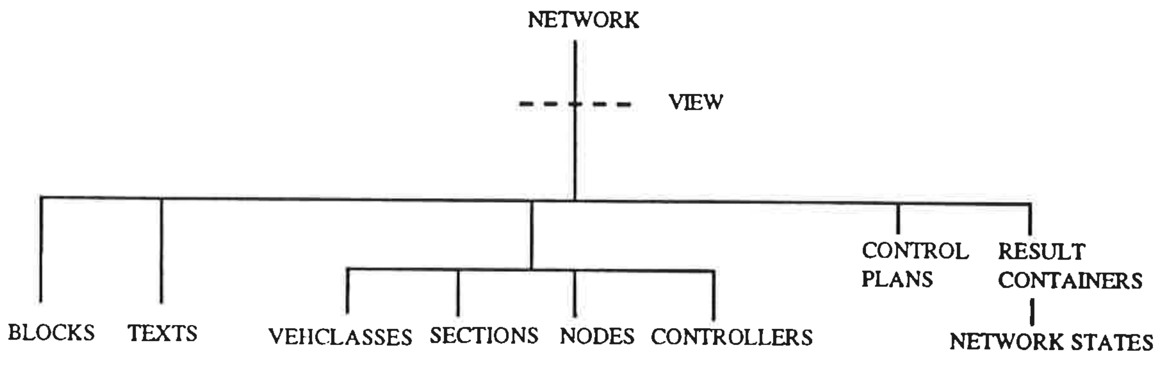


Fig. 9.2 Hierarchy of network objects.

Programmers using the TDFunctions have to include the "TDFun.h" file, which, in turn, includes the following files:

- TDTypes.h: Data types used by the function parameters.
- TDFunctions.h: Function prototypes.
- TDErrors.h: Symbol definition of error codes returned by the functions. A function is provided to obtain a string describing the error, ready for output on screen.
- TDGlobal.h: Global variables, mainly for error reporting.

Given an open network and a restricted view, the READ functions on the network objects refer to the objects contained in the current view. That is, the READ NEXT functions actually read the next object in that view. In contrast, when creating new objects or modifying them, the whole network has to be considered for any possible incompatibility (identifiers, etc.). The TDFunctions manage these details transparently.

For example, the following code reads all the sections in the network 'Net_example', changes their capacity and the maximum allowed speed according to the number of lanes, and creates selective authorization in the right-most lane for the 'public transport' vehicle class.

```
ushort error, idsection;
Bool first;
TDSection *TDsection;

    /* open the network, without specifying its storage format */
if (error = TDOpenNetwork("Net_example", 0))
                                        goto LERROR;

first = TRUE;
end = FALSE;
while (! end) {
    if (first) {
                    /* read first section */
            error = TDReadFirstSection(&TDsection);
            first = FALSE;
    } else {
                    /* read next block */
            error = TDReadNextSection(&TDsection);
    }
    if (error == SectionUnk) {
                            break;
    } else if (error) {
                    goto LERROR;
    }

            /* opens the section before update can take place */
    if (error = TDOpenSection(&TDsection->idsection))
                                                goto LERROR;

            /* set the flags to modify the required fields */
    TDsection->flags = SetSMaxSpeed | SetSCapacity;
            /* initializes the required fields */
    if (TDsection->nblanes == 1) {
            TDsection->maximum_speed = 50.;
    } else {
            TDsection->maximum_speed = 60.;
    }
    TDsection->capacity = TDsection->nblanes * 1800.;

            /* call the function to modify these fields */
    if (error = TDModifySection(&TDsection))
                                goto LERROR;
            /* create selective vehicle permission for the 'public transport' class in the
             *  right-most lane
             */
        if (error = TDCreateSelVehPer(1, 1, "public transport"))
                                                    goto LERROR;

            /* closes the section in update mode */
        if (error = TDCloseSection(1))
                                goto LERROR;
    }

        /* closes the network, in update mode */
    if (error = TDCloseNetwork(1))
                                goto LERROR;

    ............

        /* error treatment */
LERROR:
        /* retrieve error message in TDMessage[] global variable */
    TDTextMessage(error);
        /* prints the message on screen */
    printf("ERROR- %s\n", TDMessage);
```

10. Network editor

Included in this Ph.D. thesis is the design of the Traffic Network Editor, TEDI for short, which makes fairly complete use of the TDFunctions and provides the user with a friendly, mouse-oriented interface. TEDI has been programmed by David Garcia and Olga Llima in the X-Windows and OSF/Motif environment and is therefore quite portable across UNIX platforms. A programmer's description being available elsewhere (Grau 1993), only a brief introduction is given here.

TEDI presents the user with a realistic view of the network; see Figure 10.1. In comparison, most other network editors for traffic modeling deal with links (unidimensional segments) and nodes (a point). Although traffic engineers may be used to these simplifications, we believe that TEDI's approach has some clear advantages:

- It provides a better geometrical accuracy in sections and nodes. Objects in the network are built following, very closely, their description in 5. Moreover, a modern visual interface makes this almost as fast as it would be in a link-node model.
- If a DXF graphical-format file of the area is available, then TEDI can load it as a background, so that digitizing maps becomes unnecessary; see Figure 10.2. When unavailable, then there is still the possibility of obtaining an aerial photograph that can be processed by a CAD system and outputting it in DXF format.

- In TEDI, all edition capabilities — section geometry, turning movements, junction signal setting, and so on — are available in the same window, without the need to switch to another module or to change the view of the network. Network states and control plans are also specified within the main window.
- Dialog boxes follow a cascade approach, extending as the level of detail in an object is required; see Figure 10.3.
- By using TDFunctions, the user does not need to know the storage format of the network being edited.

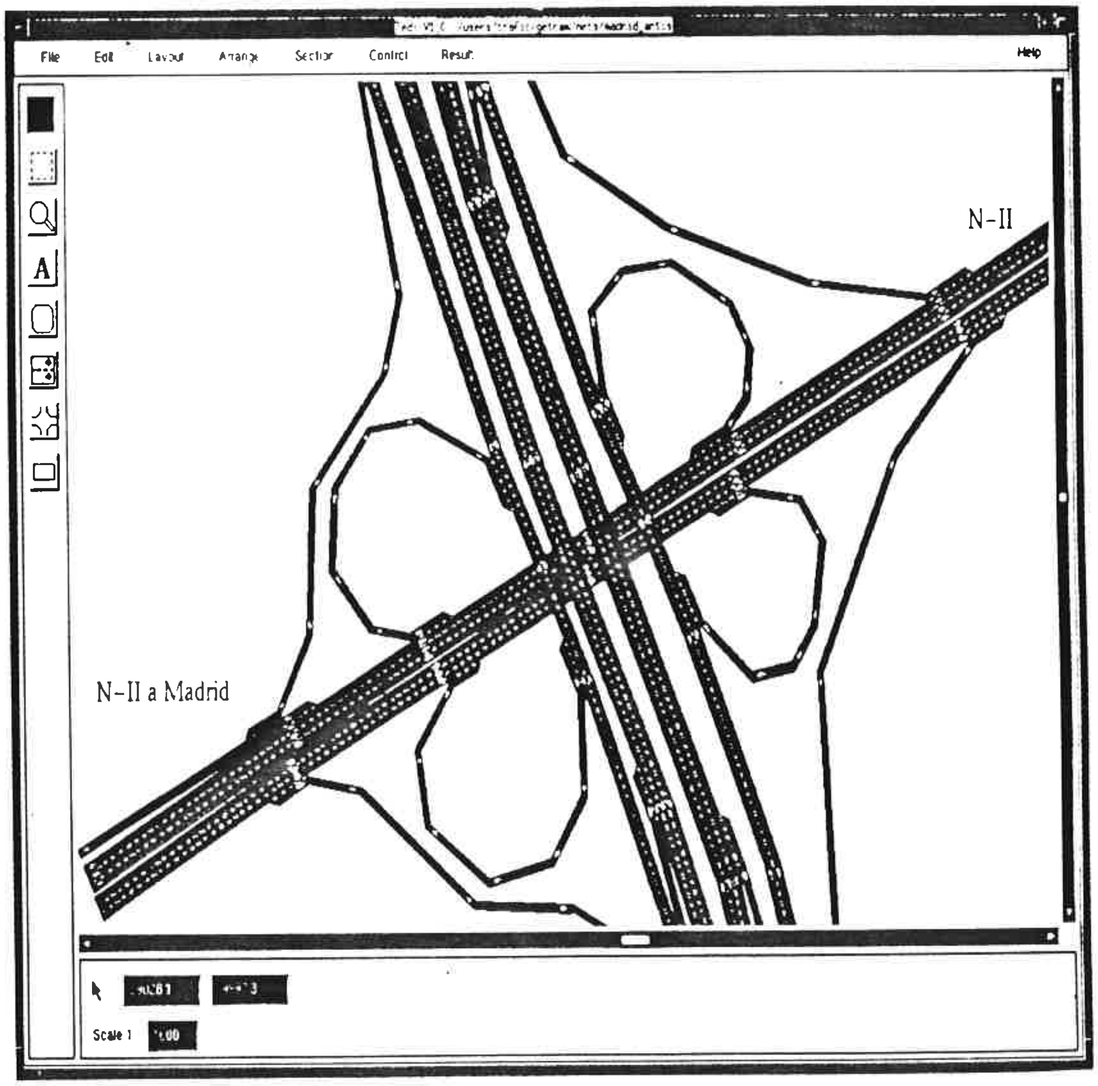


Fig. 10.1 TEDI showing a freeway area.

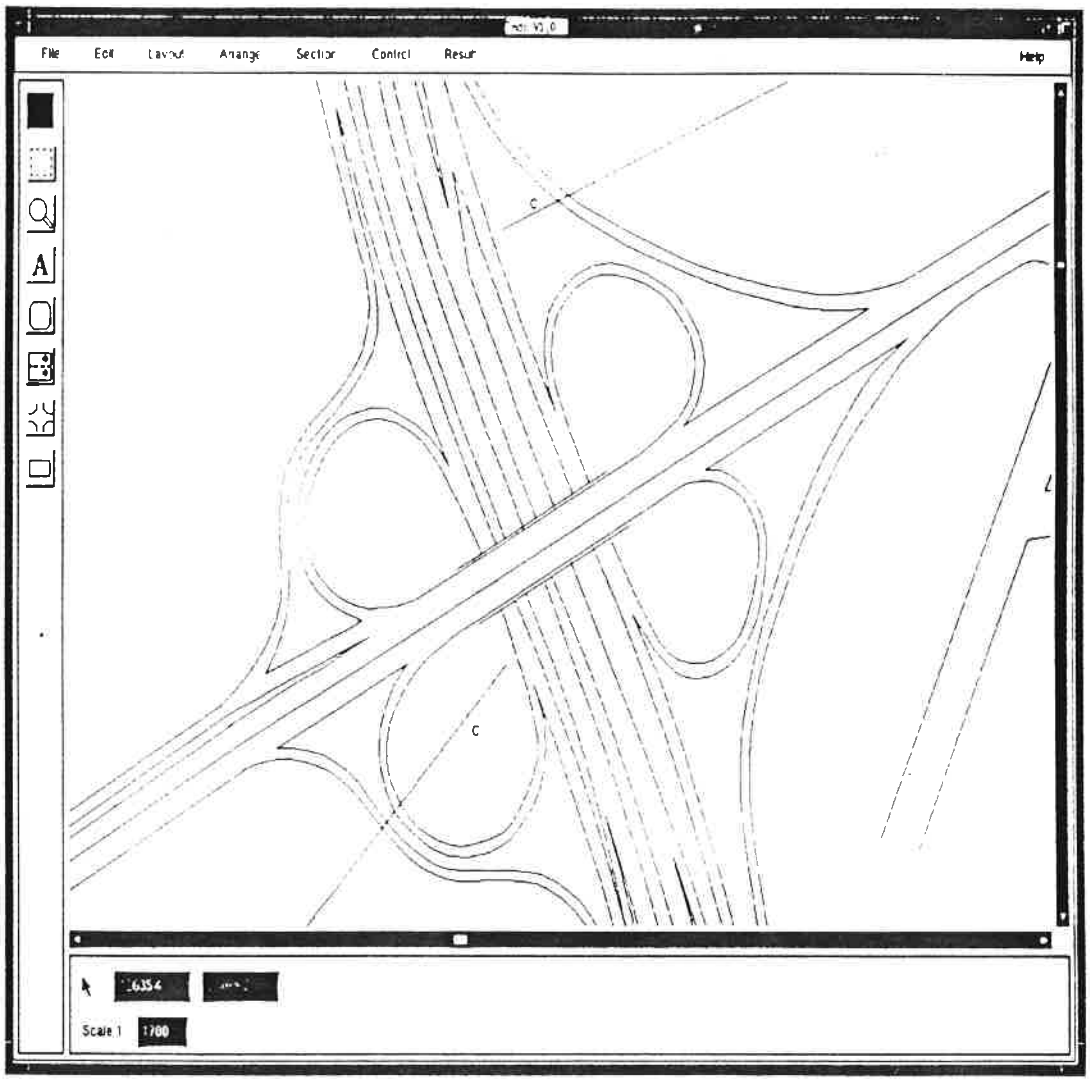
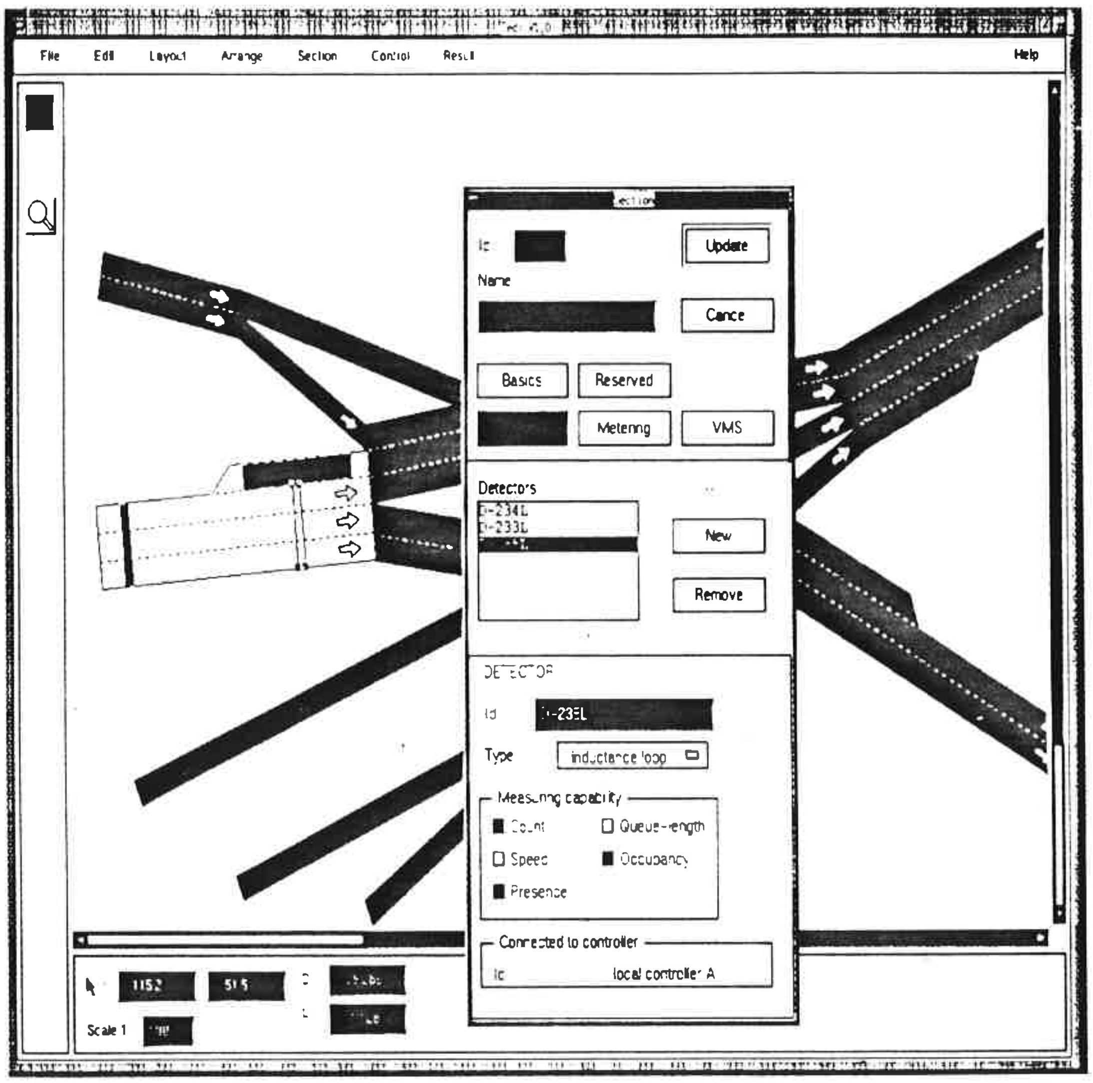


Fig. 10.2 The same area as in Fig. 10.1, with the background that had been used to edit it.



Id

Name

Detectors

D-234L
D-233L

DETECTOR

Id D-23EL

Type inductance loop

Measuring capacity

■ Count ☐ Queue-length

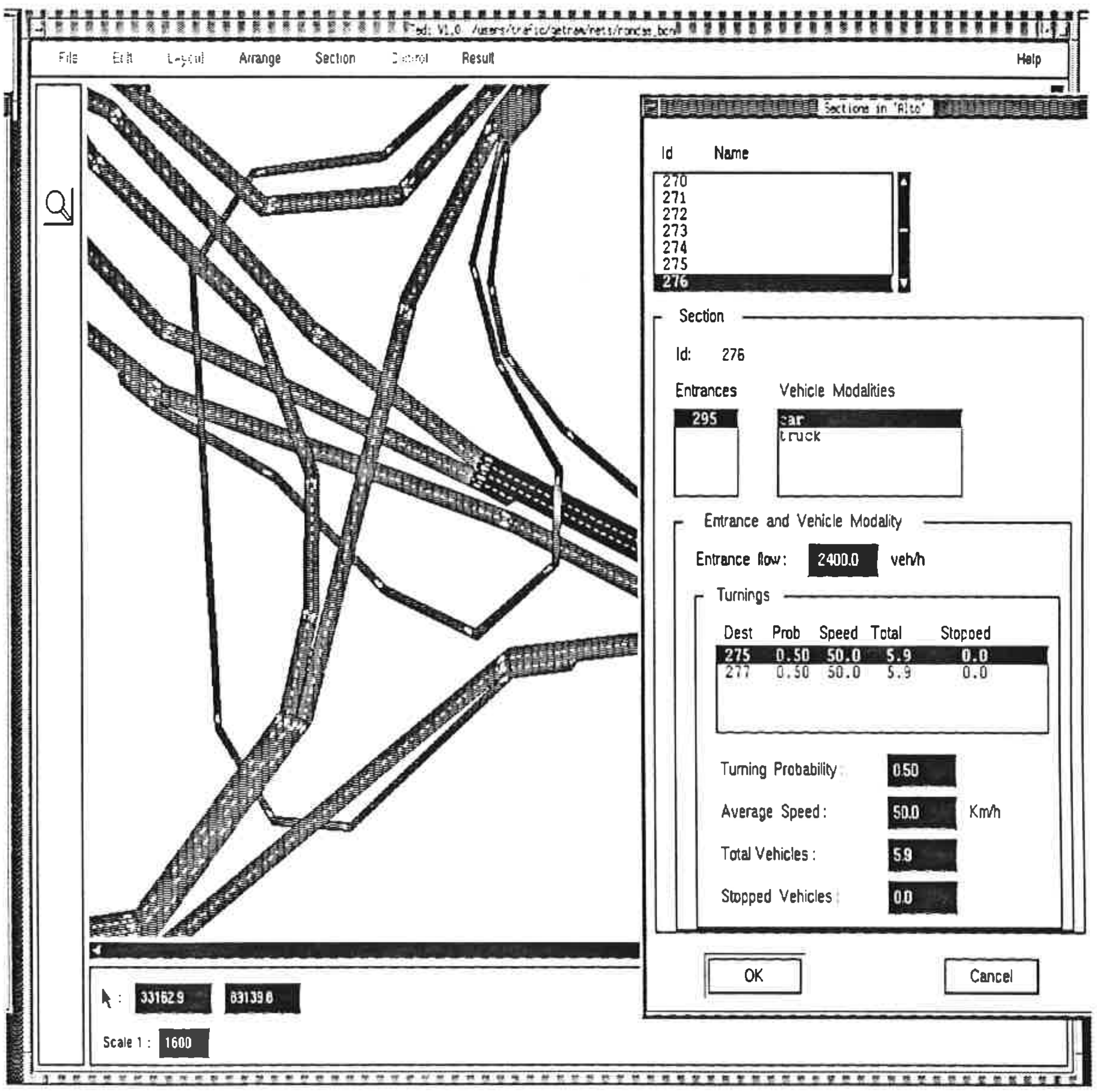
☐ Speed ■ Occupancy

■ Presence

Connected to controller

Id local controller A

Fig. 10.3 Dialog box of a section showing the cascade shape when examining detectors positioned on that section.



Id Name

270
271
272
273
274
275
276

Section

Id: 276

Entrances: 295

Vehicle Modalities: car, truck

Entrance and Vehicle Modality

Entrance flow: 2400.0 veh/h

Turnings

Dest	Prob	Speed	Total	Stopped
275	0.50	50.0	5.9	0.0
277	0.50	50.0	5.9	0.0

Turning Probability: 0.50

Average Speed: 50.0 Km/h

Total Vehicles: 5.9

Stopped Vehicles: 0.0

331162.9 831139.8

Scale 1 : 1600

Fig. 10.4 Display and editing of a traffic state in TEDI.

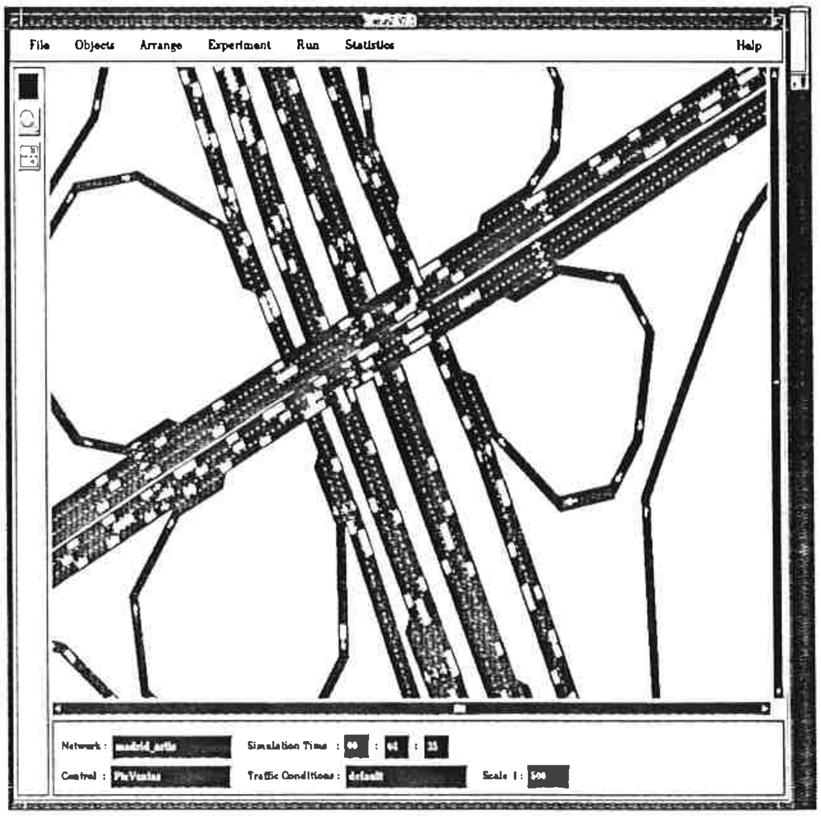
11. Integration of a simulation model

The main advantages that a simulation model can derive from the environment come from the use of the TDFunctions: sub-area analysis, results sharing between other models, DataBase consistency, object-based access to the network, and choice between various storage formats. All them have been previously explained. Additionally, we should include the graphical editor which is available in the environment, and the interface to EMME/2.

Examples of simulators now in GETRAM are CARS V2 and AIMSUN2. Of these, only CARS V2 is included in this Ph.D. thesis, and its integration into GETRAM will be explained in the next part. AIMSUN2 (Ferrer and Barceló 1993), a microscopic traffic simulator, is also integrated into GETRAM:

- It uses the TDFunctions to access the network data and results (network states) produced by other models — assignment or simulation models.
- Equally, view restriction has been made available with no programming cost.

AIMSUN2 offers a graphical interface that, additionally, has been programmed with less cost by using many of the network editor features: dialog boxes from network management, control plans to results containers and network states. Also scrolling, panning and zooming benefit from code sharing. AIMSUN2 interactive output displays the network similarly to the network editor; see Figure 11.1.



Network : madrid_artis   Simulation Time : 00 : 04 : 35
Control : PlzVentas   Traffic Conditions : default   Scale 1 : 500

Fig. 11.1 AIMSUN2 interface showing the same area of Figure 10.1.

12. Experience with the environment

The environment has been used by the author to build and test networks for CARS V2, as will be discussed in the next part. Additionally, the environment has been used within the TRYS project of the GUIDAS team, to which the author belongs. The TRYS project was sponsored by the Dirección General de Tráfico and our task was to model the ring roads of Barcelona, Madrid, and Seville. The environment, notably the graphical editor and AIMSUN2, was applied to build and simulate the ring roads of these three cities. In the case of Barcelona and Madrid, a graphical description of the area — as a DXF formatted file — was provided by the local authorities. Then, the graphical editor TEDI loaded it as a background upon which network building proceeded. TEDI allowed fast network creation: for example, the upper part of Barcelona's ring roads, known as the "ronda de dalt", containing about one hundred nodes, was built in just two man-days. Because at that time the assignment model was not yet available, a set of network states was built using the TDFunctions corresponding to light to congested traffic conditions. From this, AIMSUN2 was able to simulate that area starting from those imaginary network states. AIMSUN2 has an interactive graphical output, so that the correctness of the model — turnings, detectors, etc. — was tested before real network states from the assignment model were available.

Had GETRAM not been available, the building of the "ronda de dalt" model would have required no less than one man-month in AIMSUN1 without the environment. Compare this with the two days required in GETRAM. Also, the use of any other available graphical environment would have limited the model's degree of detail while increasing similarly the required time.

13. Conclusions

In this part, I have presented an environment, GETRAM, that, although initially intended to ease the task of testing CARS, goes beyond this to provide an environment for traffic analysis and modeling, emphasizing integration between assignment and diverse simulation models. Because data input — network building — in such a system is one of the hardest tasks, we propose a network editor that eases it while being capable of providing microscopic detail: the description closely matches the real network, avoiding model-dependent link-node descriptions. In comparison, all other systems available to date fail to deliver such a high degree of integration and ease of use.

GETRAM modules lie on top of a library of functions, named TDFunctions, that provide an access to all the data independently of their storage format. Their object-based design has shown its advantages when programming the network editor, virtually eliminating any data consistency check, so it has been able to focus on graphical capabilities. Integration of CARS V2 and AIMSUN2, although less demanding on these functions, has also been facilitated, this time mostly by the view management that the functions provide. As more storage formats become available, modules will access them without any change to their code.

GETRAM does not aim to provide, just by itself, a complete transportation planning system, but rather to allow a close integration between diverse models. An interface allows the use of EMME/2 to apply all its tools — matrix utilities, assignment, results analysis — to a network in GETRAM. The environment has already been applied to a medium-sized network, and results both in ease of use and in modeling capabilities.

The next part, PART IV, starts with the integration of CARS V2 in the GETRAM environment. Then, new networks are built and tested by using AIMSUN2 into the environment. Results of these tests indicate the need for some improvements in CARS V2, and, finally, a full suite of tests is presented.

14. References

DRIVE V1054 (1992). *System and Scenario Simulation for Testing RTI Systems. (ASTERIX). Final Report, Deliverable 8*, July 1992, Universitat Politècnica de Catalunya.

Doughty, J.R. (1982). "Traffic signs and markings.", *Transportation and Traffic Eng. Handbook*, Prentice-Hall Inc., pp. 712.

Ferrer J.L., and J. Barceló (1994). *AIMSUN2: Advanced Interactive Microscopic Simulator for Urban and Non-urban Networks. System Description*, Universitat Politècnica de Catalunya, Dept. of Statistics and Operational Research, Forthcoming Research Report.

Grau, R. and J. Barceló (1992). *PACKSIM: An experience in using traffic simulation in a demand-responsive traffic control system*, Universitat Politècnica de Catalunya, Dept. of Statistics and Operational Research, Research Report 92/05.

Grau, R. (1993). *GETRAM's Design of the Graphical Traffic Network Editor*, Universitat Politècnica de Catalunya, Dept. of Statistics and Operational Research, Internal Document.

Grau, R. (1994). *GETRAM's Application Programming Interface, version 1.0. Technical Description*, Universitat Politècnica de Catalunya, Dept. of Statistics and Operational Research, Internal Document.

INRO Consultants (1992). *EMME/2 User's Manual. Software Release 6.0.*

MacTrans (1989). *GTRAF User's Guide*, Center for Microcomputers in Transportation, University of Florida.

MVA Systematica (1993). *TRIPS. The Professional's Transportation Planning Package. Description.*

Robertson, D.I. (1974). "Cyclic flow profiles.", *Traffic Eng. Control*, June 1974, 15(14), pp. 640-641.

The Urban Analysis Group (1993). *URBAN-SYS. TRANPLAN 7.2. User's Manual.*

Van Vliet, D. and M. Hall (1991). *SATURN 8.2. A User's Manual*, Institute for Transport Studies, University of Leeds.

PART IV
Additional developments: CARS V2

This is the last part of this Ph.D. thesis; it starts from the demand-responsive traffic control system developed in PART II, CARS V1, and improves it in various respects. First, in order to ease the task of testing the system, it is integrated into the traffic modeling and analysis environment described in PART III. Second, certain parts that directly influence to the effectiveness of the system, such as control timing, adaptive control logics, and communication with the controllers, are revised or totally redone. A suite of tests has been applied to the resulting system, CARS V2, in the four scenarios described in PART II. Finally, to further testing the system and taking advantage of the fact of having real-world data available, it is compared against a vehicle-actuated control at an isolated junction.

1. Integration into the GETRAM environment

In the previous part, PART III, a new environment for traffic analysis and modeling, GETRAM, was described. The initial aim of designing a new environment was to ease the task of testing CARS — network creation, data consistency, sharing the same network objects between CARS and the AIMSUN microscopic simulator, and so on. After reviewing the state of the art of traffic environments, it was clear that new developments in this area were badly needed. The proposed environment improves also in some additional areas, such as support for different levels of detail, results sharing between traffic models, view restriction, a common network editor, and a common object-based Application Programming Interface to access the data.

The first task in this part is to integrate CARS into the GETRAM environment. Because GETRAM was designed to support diverse traffic models — not only CARS —, there are some details that are not supported in CARS or where a compromise was needed. The correspondence between entities and concepts in GETRAM and CARS is as follows (Figure 1.1).

GETRAM	CARS
block	**block** A direct correspondence applies.
section	**section** Neither lateral lanes nor reserved lanes are supported. Would need considerable changes in PACKSIM in order to support them. Detector's measuring capability is restricted to count, speed and occupancy. Detector's length is not considered. There is no support for meterings as CARS is not designed for freeway areas but for urban areas.
node	**junction** Of the two types of nodes, junctions and junctures, only junctions are supported — junctures are used mainly in freeway areas. The junction can be uncontrolled, fixed-controlled, or demand-responsive. Give-ways and any number of stages are supported.

controller	**controller** A controller can be connected to a number of junctions and detectors. Meterings are not supported.
vehicle class	Not supported.
control plan	All junction parameters are supported: control type, critical, offset, stage durations (initial and range). Progression arterials are considered. Meterings are not supported.
results container and network state	A results container provides information about the average vehicle, and the initial turning percentages — CARS opens only the first network state. Information about the various vehicle modalities in the network state — such as length, or mean desired speed — is averaged because CARS deals with an averaged type of vehicle.

Fig 1.1 Correspondence between GETRAM and CARS entities.

As a result of integration into GETRAM, the new CARS — called CARS V2 — uses the GETRAM API to acess most of the data, with the following advantages:

- A 30% reduction in initialization code, because consistency checks are no longer necessary. Also, the GETRAM API provides a large number of functions to help the programmer calculate almost every geometrical detail of the section, and allows access to related objects in a flexible way.
- Object-based access to the network, making the code easier to maintain.
- Potential choice between various storage formats.
- Networks are created and updated by using the TEDI common editor. The same network can then be used both by CARS and by the AIMSUN microscopic simulator. TEDI provides more advanced editing capabilities than CARS V1's CARSedi.

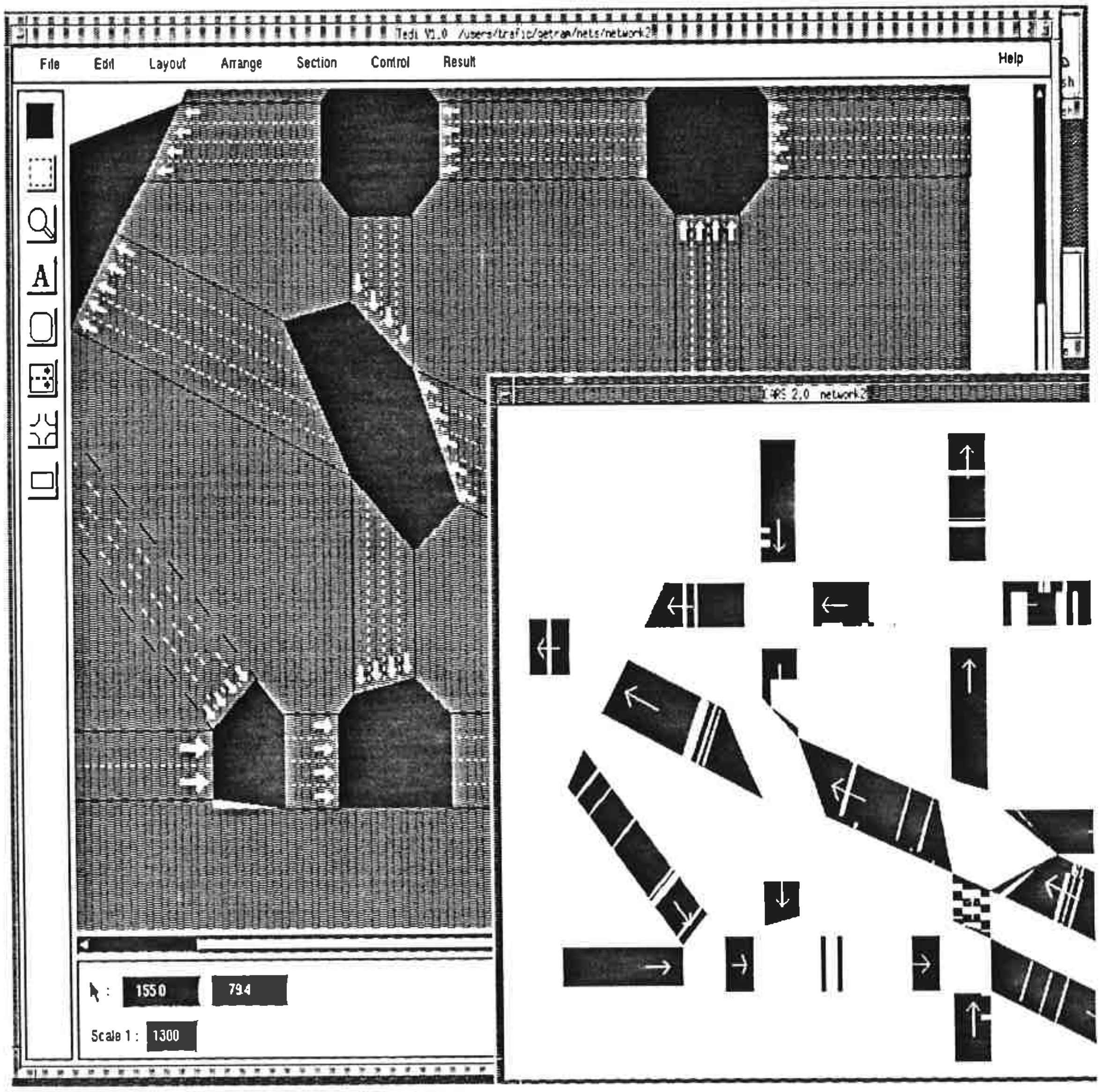


Fig. 1.2 GETRAM's editor (left) and CARS V2 (right) showing the same area.

2. Control timing

As was said in the previous version (PART II.3), time in CARS is discretized in intervals of delta seconds, and the system loops in the System Control Cycle (SCC) of scycle deltas duration in which a series of tasks are performed, mainly, feedback, stepping ahead and planning. This SCC includes advanced features — not found in any reviewed system — such as the stepping ahead task, and the possibility of scycle taking a value greater than one; in fact, tests in PART II have been performed with a scycle value of 5 deltas. CARS V2's control timing improves further on the previous V1 version in that:

- Delta is now a floating point number. Previous tests had been done with integer values of 1 or 2 secs. However, in CARS V2, delta is allowed to take values such as 0.75 or 1.5, adding flexibility to the system.
- The system does not communicate to each junction the signal to end the current phase, but sends complete control plans. As a consequence, the controllers have to be smart enough to update their state in the control plan's cycle, but they are also safer in the event of a systems failure. This simplifies CARS's SCC in that the system only needs to send signals to junctions at the end of the SCC; see Figures 2.1 and 2.2. In CARS V1, previous to the feedback, the system had to sent the phase-endings corresponding to the next (scycle-1) deltas, and, after the planning task, the remaining phase-ending.

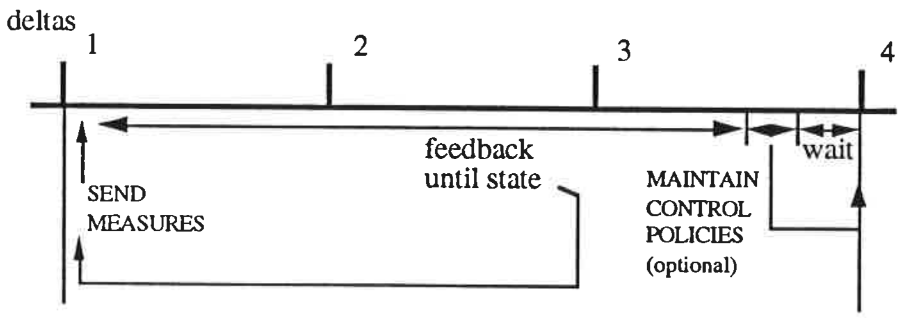


Fig. 2.1 SCC of 3 deltas without planning in CARS V2.

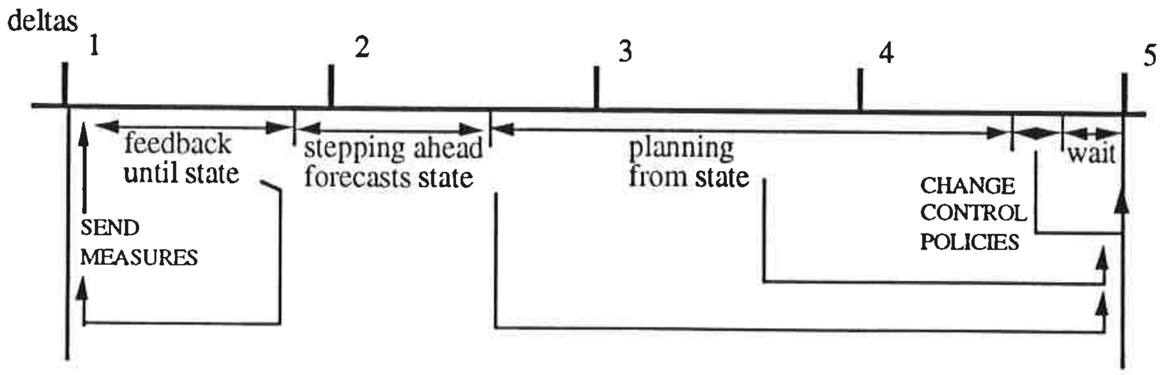


Fig. 2.2 SCC of 4 deltas and planning in CARS V2.

3. Adaptive control logics

A critical area in CARS, as a simulation-based demand-responsive control system, is the algorithm that decides, at each System Control Cycle, which changes in control policy are to be tested. In CARS V1, the algorithm considers incrementing or decrementing in one delta the duration of the current phase, and the tests are done according to the tendency shown in the previous SCC (see PART II.7.1). This reduces the number of required tests to a maximum of three — current control plan, incremented, and decremented current phase — and, by using the tendency, in most cases only two tests are required. The appeal of such an algorithm lies both in the simplicity of coding it and in the very small computing requirements. In contrast, OPAC's (Gartner 1982) method of testing every control plan puts a high load on computing requirements and it is only feasible if the traffic simulation model is very simple — which hinders its use in congested conditions.

CARS V1 showed two shortcomings that, as will be seen hereafter, can be solved by changing — enlarging — the set of changes in control policy that are to be tested. The first shortcoming is the worse performance than the binary-choice approach in an isolated junction operating under light traffic conditions. The second one is the performance degradation when starting from a control plan that is far from optimal given the current traffic conditions. Two different methods are proposed to solve these problems, each one enlarging further the set of changes.

First method: Incrementing or decrementing from 1 to **maxvar** deltas the duration of the current phase — not just one delta, as before. This results in a set of (2 * maxvar + 1) control plans.
Second method: The same as in the first method, but for all the phases in the junction's cycle, not only in the current phase, giving a total set of ((2 * maxvar)^{nbphases} + 1) control plans.

Of course, unfeasible control plans — according to current state in the junction's cycle — are discarded. Provided that the whole set is feasible, note that the second method produces far more control plans than the first method; for example, a maxvar value of 4 deltas gives 9 (first method) and 65 (second method) control plans.

The two methods have been tested in an isolated junction with two phases, as in PART II.10.1, both in light and heavy traffic conditions. In all of them, a delta value of 1 sec. and an SCC of 1 delta were used, and the tests ran a total time of 1,000 secs. Averaged results are shown in Figures 3.1, 3.2 (light conditions), 3.3, and 3.4 (heavy conditions), and each result is given by three objective functions that are accumulated along the run and, at the end, averaged per delta: stopped vehicles, queue length, and average speed, as described in PART II.8. Results applying an optimal fixed-control policy are (6.5, 9.3, 1.8) and (36.1, 50.3, 1.5) for light traffic and heavy traffic conditions, respectively.

As can be seen, maxvar values greater than one give far better results, thus both methods can, potentially, increase CARS V2's performance as compared to CARS V1. When maxvar is greater than 2, the first method's results are similar to those obtained with the second method, although requiring fewer control plan evaluations; particularly, a value of maxvar = 4 deltas in the first method seems a good compromise between effectiveness and number of control plans. Note that, in these tests, CARS V2 demand-responsive control can be as much as 50% more effective than optimal fixed control — if the real world traffic behaves exactly as PACKSIM. As will be seen, this improvement in performance will help to make the system as effective as a binary choice for isolated junctions in light traffic conditions.

—light traffic conditions—

1st method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	3.7	3.8	2.8	2.7	3.0	2.9	3.0	2.9
queue length	5.2	5.2	4.0	3.8	4.3	4.1	4.3	4.2
av. speed	2.0	2.0	2.0	1.9	1.9	1.9	1.8	1.9

2nd method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	2.9	3.2	2.9	2.7	2.6	2.6	2.6	2.6
queue length	4.1	4.4	4.1	3.9	3.7	3.7	3.7	3.7
av. speed	2.0	1.9	2.0	1.9	2.0	2.0	2.0	2.0

Fig. 3.1 Results of applying the two methods to an isolated junction in light traffic conditions. Each column corresponds to a maxvar value.

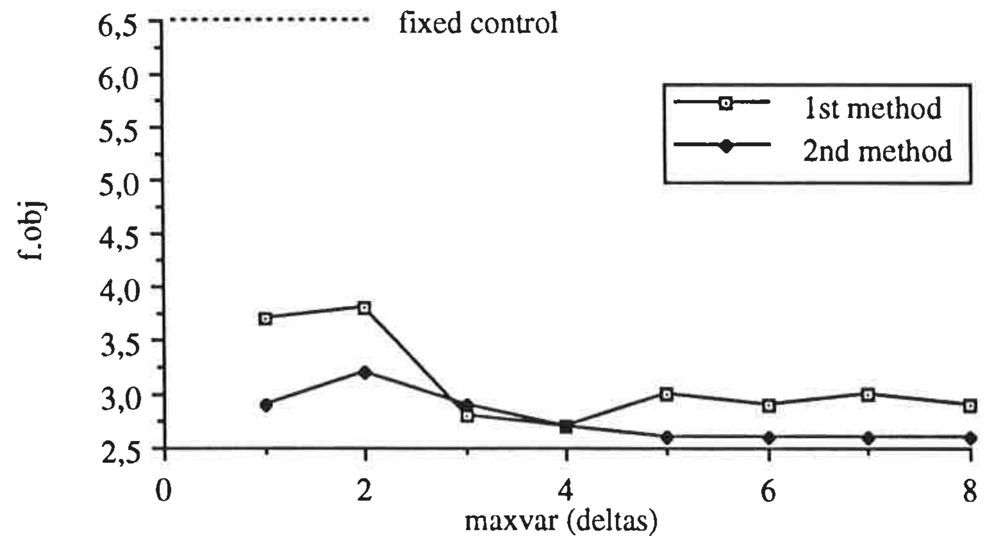


Fig. 3.2 Graphical representation of the data in Figure 3.1. The objective function corresponds to the accumulated number of stopped vehicles, averaged per delta.

—heavy traffic conditions—

1st method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	25.4	25.5	24.5	19.7	22.1	20.5	19.8	21.2
queue length	33.4	33.1	31.9	26.4	30.0	27.6	27.4	29.1
av. speed	1.6	1.6	1.5	1.2	0.9	1.0	1.1	0.9

2nd method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	21.7	22.5	23.4	23.5	22.2	22.6	20.9	21.4
queue length	29.4	29.8	32.1	32.2	30.5	30.8	28.4	29.1
av. speed	1.1	0.9	0.7	0.7	0.9	0.8	1.0	0.9

Fig. 3.3 Results of applying the two methods to an isolated junction in heavy traffic conditions. Each column corresponds to a maxvar value.

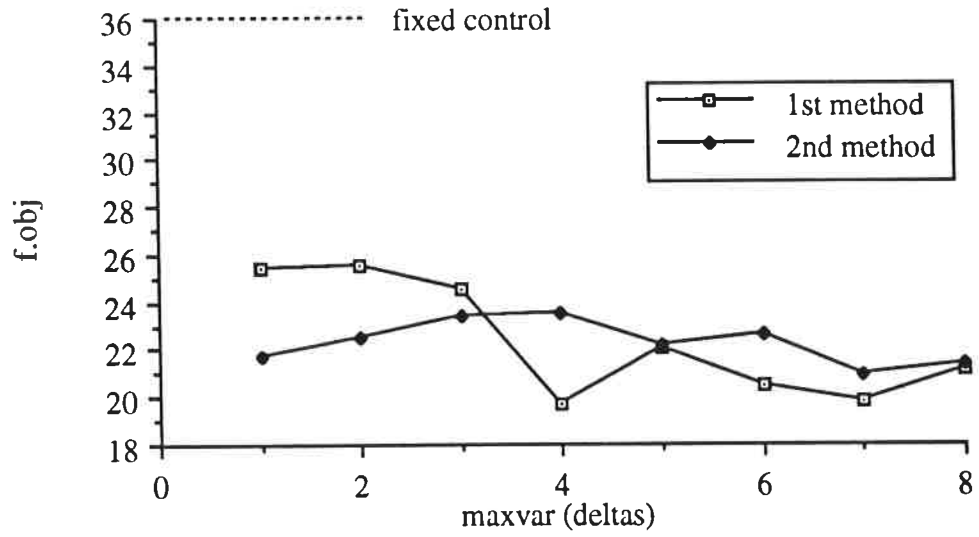


Fig. 3.4 Graphical representation of the data in Figure 3.3. The objective function corresponds to the number of stopped vehicles.

In these tests, CARS started from an optimal control plan. It remains to be seen if the first method can be effective when starting from a bad control plan — the second problem in CARS V1. For example, given the same isolated junction, and vehicle arrivals that will result in the junction becoming congested (500, 1,000, and 500 veh/h *lane in Sections 1, 2, and 3 respectively), results applying an optimal fixed-control policy are (16.7, 25.3, 1.5). When applying demand-responsive control starting from a far from optimal control plan — the control plan that was optimal for light traffic conditions — results can be worse than with fixed control if maxvar is less than 2 deltas, as shown in Figure 3.5, and graphically in Figures 3.6 and 3.7. When maxvar equals 1 or 2 deltas, the system can fall into local optima that prevent it from being effective. This explains why CARS V1 has a problem in this area: the set of control plans tested during each SCC represents too small a variation from the current control plan. A maxvar value of 4 deltas is found, again, to give a good compromise. As expected, effectiveness degrades slowly as scycle — the length of the SCC — augments; nevertheless, performance when scycle equals 1 or 2 is acceptable, leading to a 40% improvement over results obtained with optimal fixed-control plans over 1,000 secs. — results are better in the long run, when the control plan has already been adjusted to the traffic conditions.

scycle = 1

obj \ maxvar	1	2	3	4	5	6	7	8
1st	19.3	13.8	11.2	9.7	9.5	11.1	10.6	10.6
	25.8	19.5	16.8	14.9	14.7	17.0	16.1	16.1
	1.4	1.5	1.5	1.5	1.5	1.4	1.5	1.5
2nd	18.8	18.3	10.5	11.1	10.5	10.3	10.5	10.3
	26.1	26.2	16.1	16.9	15.9	15.9	16.3	15.9
	1.1	1.1	1.5	1.5	1.5	1.5	1.5	1.4

scycle = 2

1st	16.8	14.1	10.6	11.0	11.4	11.8	11.4	12.
	23.4	20.7	16.0	16.8	17.0	17.9	17.0	18.3
	1.4	1.4	1.5	1.5	1.5	1.5	1.5	1.5
2nd	19.2	14.5	10.8	11.5	11.1	10.9	10.8	10.8
	26.7	21.0	16.3	17.4	16.7	16.5	16.3	16.3
	1.1	1.3	1.5	1.4	1.4	1.5	1.5	1.4

scycle = 3

1st	17.8	14.6	11.7	12.6	12.7	13.8	12.9	12.9
	25.5	20.6	17.5	18.8	18.7	20.6	19.2	19.2
	1.3	1.5	1.5	1.5	1.5	1.4	1.5	1.5
2nd	19.2	17.5	12.5	11.9	13.3	13.1	13.8	13.1
	26.8	25.0	18.9	18.0	19.9	19.8	20.7	19.8
	1.1	1.2	1.5	1.4	1.4	1.4	1.4	1.4

scycle = 4

1st	16.9	17.1	12.0	12.5	13.1	12.3	12.3	11.1
	24.0	24.6	17.5	18.6	19.8	18.9	18.9	16.8
	1.3	1.2	1.5	1.5	1.4	1.5	1.5	1.5
2nd	19.8	17.7	13.6	13.6	13.6	13.6	13.6	13.6
	27.6	25.2	20.1	20.1	20.1	20.1	20.1	20.1
	1.1	1.1	1.3	1.3	1.3	1.3	1.3	1.3

scycle = 5

1st	20.2	15.0	13.1	12.8	13.4	14.0	13.8	13.8
	28.1	22.0	19.3	19.1	20.3	21.1	20.6	20.6
	1.2	1.3	1.4	1.4	1.4	1.4	1.4	1.4
2nd	20.3	17.3	14.4	12.8	12.6	12.1	12.7	12.7
	28.2	24.7	21.4	18.9	19.0	18.1	19.2	19.2
	1.2	1.1	1.3	1.4	1.4	1.4	1.4	1.4

Fig. 3.5 Results of applying the two methods to an isolated junction in heavy traffic conditions, starting from an initial light traffic control plan. SCC varies from 1 to 5 deltas. Each column corresponds to a maxvar value.

scycle = 1

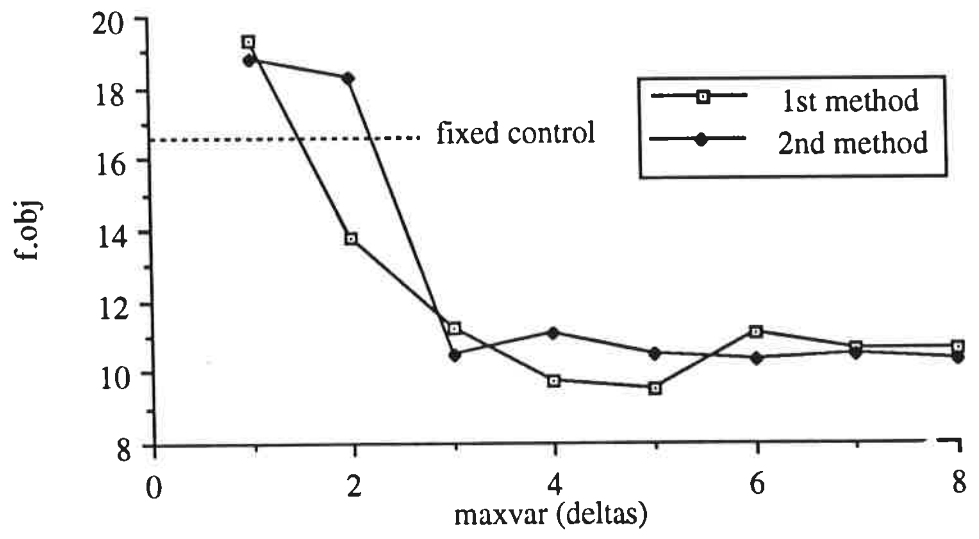
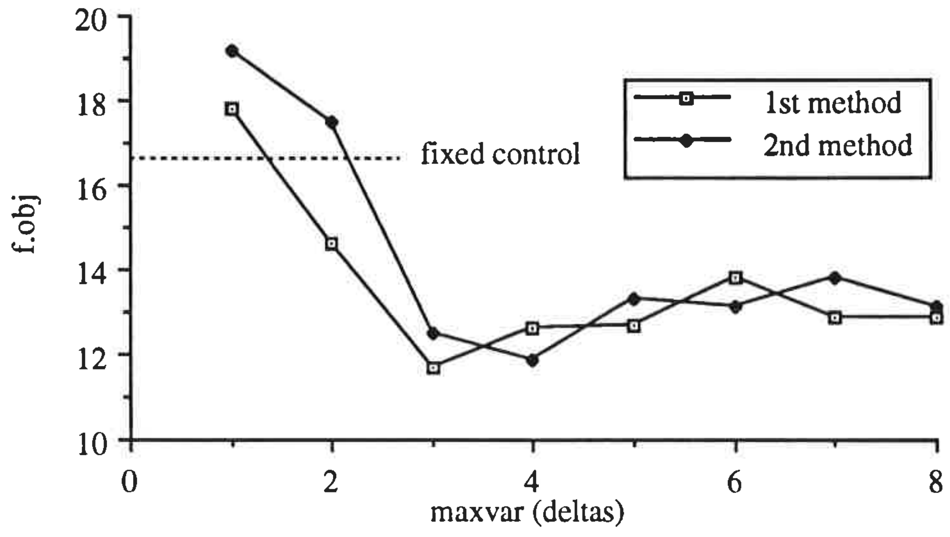


Fig. 3.6 Graphical representation of the data in Figure 3.5, for scycle = 1. The objective function corresponds to the number of stopped vehicles.

scycle = 3



scycle = 5

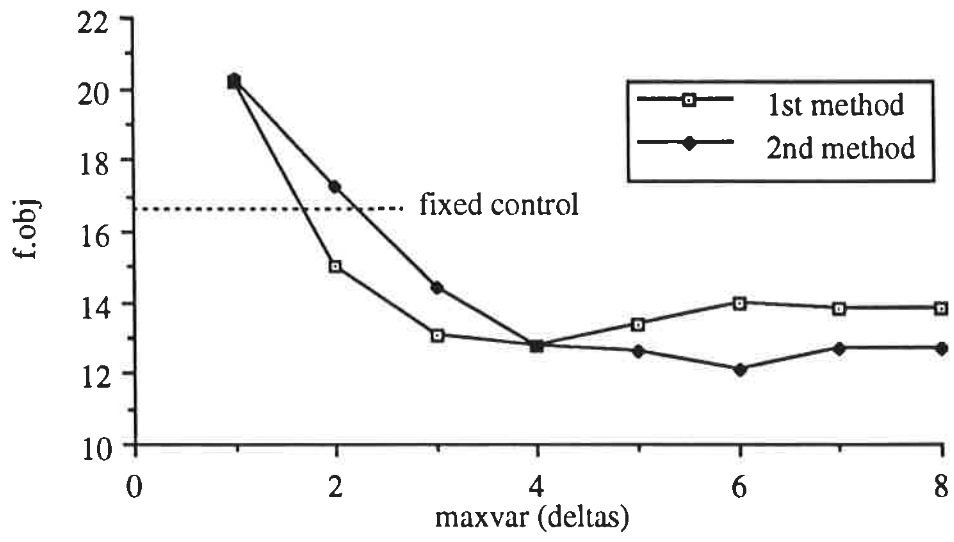


Fig. 3.7 Graphical representation of the data in Figure 3.5, for scycle = 3, and 5. The objective function corresponds to the number of stopped vehicles.

It remains to be seen if the optimal value of maxvar depends on the value of delta. Testing again with the isolated junction, this time with vehicle arrivals of 700 veh/h * lane in each section and starting from a bad control plan (again, the one that was optimal in light traffic conditions), delta = 2 secs. and scycle = 1 delta, the fixed-control plan produces the following results: (28.6, 39.5, 1.0). Again, the first autoadaptive method is the best compromise (see Figures 3.8 and 3.9) between effectiveness and computational needs, particularly when maxvar = 2 deltas, corresponding, in seconds, to the optimal value found in the previous tests; in both cases, maxvar = 4 secs. is the best compromise. Similar results have been found in other traffic conditions and also in networks.

1st method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	32.4	21.6	21.1	21.1	21.1	21.1	21.1	21.1
queue length	43.5	30.0	29.1	29.1	29.1	29.1	29.1	29.1
av. speed	0.9	1.3	1.3	1.3	1.3	1.3	1.3	1.3

2nd method

obj \ maxvar	1	2	3	4	5	6	7	8
stopped veh.	26.8	22.0	20.7	20.7	20.7	20.7	20.7	20.7
queue length	36.3	30.1	29.6	29.6	29.6	29.6	29.6	29.6
av. speed	1.1	1.2	1.3	1.3	1.3	1.3	1.3	1.3

Fig. 3.8 Results of applying the two methods to an isolated junction starting from a control plan that is not suitable for the traffic conditions.

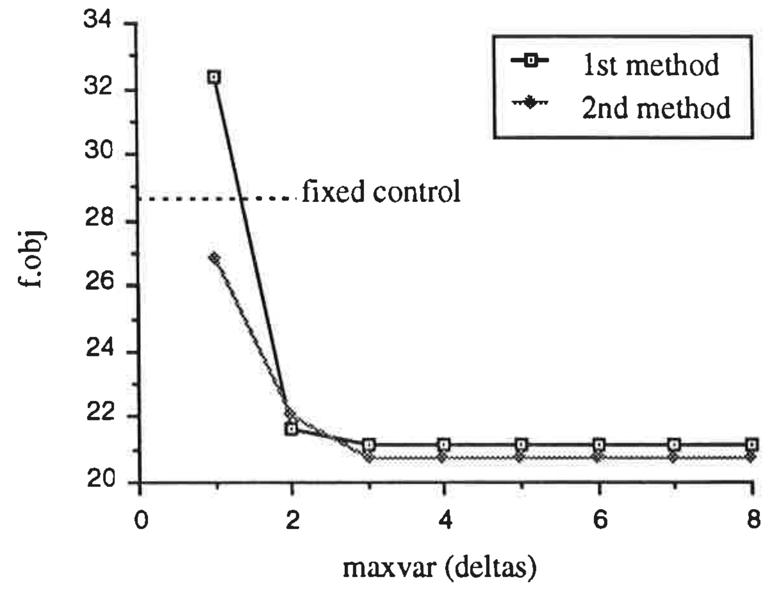


Fig. 3.9 Graphical representation of the data in Figure 3.8.
The objective function corresponds to the number of stopped vehicles.

Another question is whether the impairment of the performance when augmenting scycle (the length of the SCC) given delta = 1 sec. (see Figure 3.5) happens also with other values of delta, and if so, at what pace. Experimenting again with the isolated junction, vehicle entrances of 700 veh/h * lane in every section, starting from a bad control plan, and using the second method (maxvar = 4 secs.), with delta = 1., 1.5, 2., 2.5, and 3. secs. (Figure 3.10), the performance degrades with scycle and there does not seem to be any significant dependence between the pace of degradation and the delta value, as shown in Figure 3.11. Note that results with different delta values cannot be compared — even the fixed control plan produces different results depending on the value of delta — but evolution over the same delta can be compared.

scycle \ delta	1	1.5	2	2.5	3
	25.6	41.8	28.6	47.2	35.5
fixed control	35.1	57.0	39.5	65.1	48.9
	1.0	0.7	1.0	0.6	1.0
	14.2	22.3	21.6	29.8	28.5
1	19.1	30.8	30.0	41.6	38.6
	1.3	1.2	1.3	1.1	1.2
	15.5	22.7	21.2	32.4	29.4
2	20.7	31.4	29.2	44.8	39.7
	1.3	1.2	1.3	1.0	1.2
	17.7	22.2	26.3	33.3	27.9
3	23.9	30.8	35.9	46.2	38.0
	1.3	1.2	1.1	1.0	1.3
	19.1	23.3	23.9	31.5	28.4
4	25.3	32.1	32.8	44.0	38.5
	1.3	1.2	1.2	1.0	1.3
	17.9	23.6	23.3	33.4	29.4
5	23.7	32.6	32.3	46.5	39.8
	1.1	1.2	1.2	1.0	1.2

Fig. 3.10 Comparison of results obtained applying the second method (maxvar = 4 secs.) to an isolated junction, varying the values of scycle and delta.

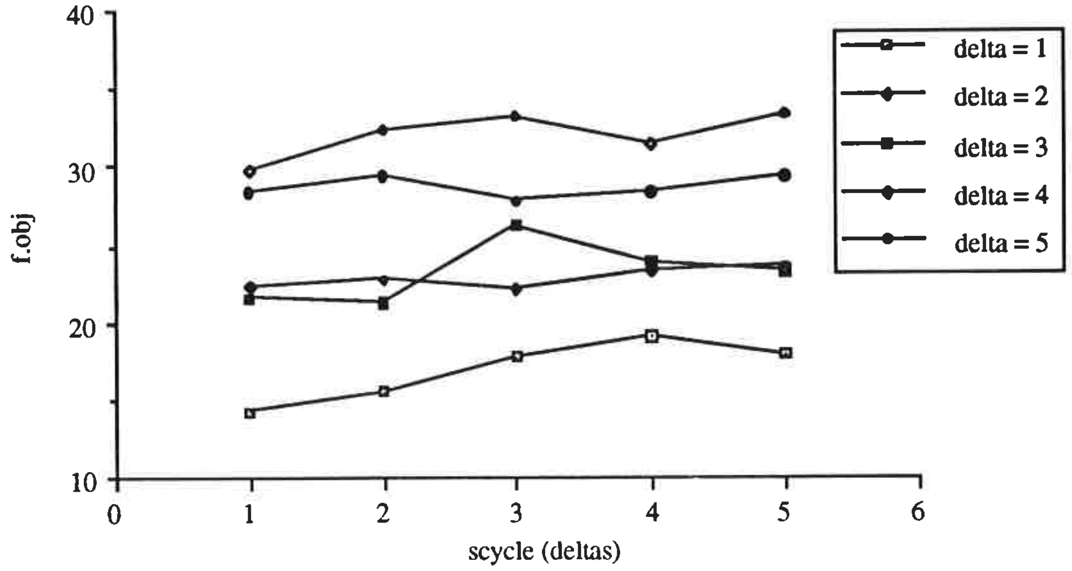


Fig. 3.11 Graphical representation of the data in Figure 3.10.
The objective function corresponds to the number of stopped vehicles.

4. Communication with the controllers

As in CARS V1, controllers in CARS V2 are the logical entities with which CARS interacts with the real world: they send traffic measurements — collected by the detectors — to CARS and receive changes in control policy. A difference lies in the way these changes in control policy are sent: while CARS V1 sends, to each junction, the signal to end the current phase, CARS V2 sends, regularly, complete control plans to each junction. As has been explained in **2**, this simplifies the sequence of tasks in the System Control Cycle. This requires the controllers to be a little smarter — they have to have some memory and update their state in the control plan´s cycle — so that they are responsible for timing the phases according to the last received control plan. Another advantage is that, should CARS fail —and abort — at a given moment, the controller has enough information to continue changing phases until CARS restarts or another control system takes control.

The other difference is that, instead of specifying the interface with a series of functions, CARS V2 communicates with the controllers by using a messaging protocol, whose format is specified in APPENDIX III. The messages have been implemented by using TCP/IP stream sockets, thus they are very portable. The protocol has been designed with flexibility and general-purposeness in mind, so that, by using them, one could employ another control system, another microscopic simulator, or even extend them in order to communicate with real-world controllers.

Eight messages are distinguished: welcome, send timing parameters, request control plan, send control plan, go, request detector measurements, send detector measurements, and end. Two sets of message interchanges are distinguished: at the beginning (Figure 4.1), and periodically during the execution (Figure 4.2).

- Welcome. First message sent by CARS and the controllers to each other.
- Send timing parameters. Used by CARS in order to communicate to all controllers timing parameters that are necessary both for sending the right number of measurements, and to accumulate each measurement during the right number of seconds.
- Request control plan. Used by CARS in order to ask for the state of the control policy in a certain junction — connected to a controller.
- Send control plan. This can be used in two ways:
  - Initially, by the controllers in order to send to CARS the initial state of the control policy in a given junction connected to a controller.
  - By CARS in order to set a junction to a given state in control policy. It is called at the beginning of execution for the uninitialized junctions and thereafter to adapt the control policy in a demand-responsive junction.
- Go. Used to resume processing.
- Request detector measurements. Used by CARS in order to require a detector to send its last scycle measurements, each one of them having been accumulated for delta seconds.
- Send detector measurements. Used by the controllers in order to send to CARS the last scycle measurements — each one has been accumulated for delta seconds — of a detector.

- End. Communicates that CARS will stop.

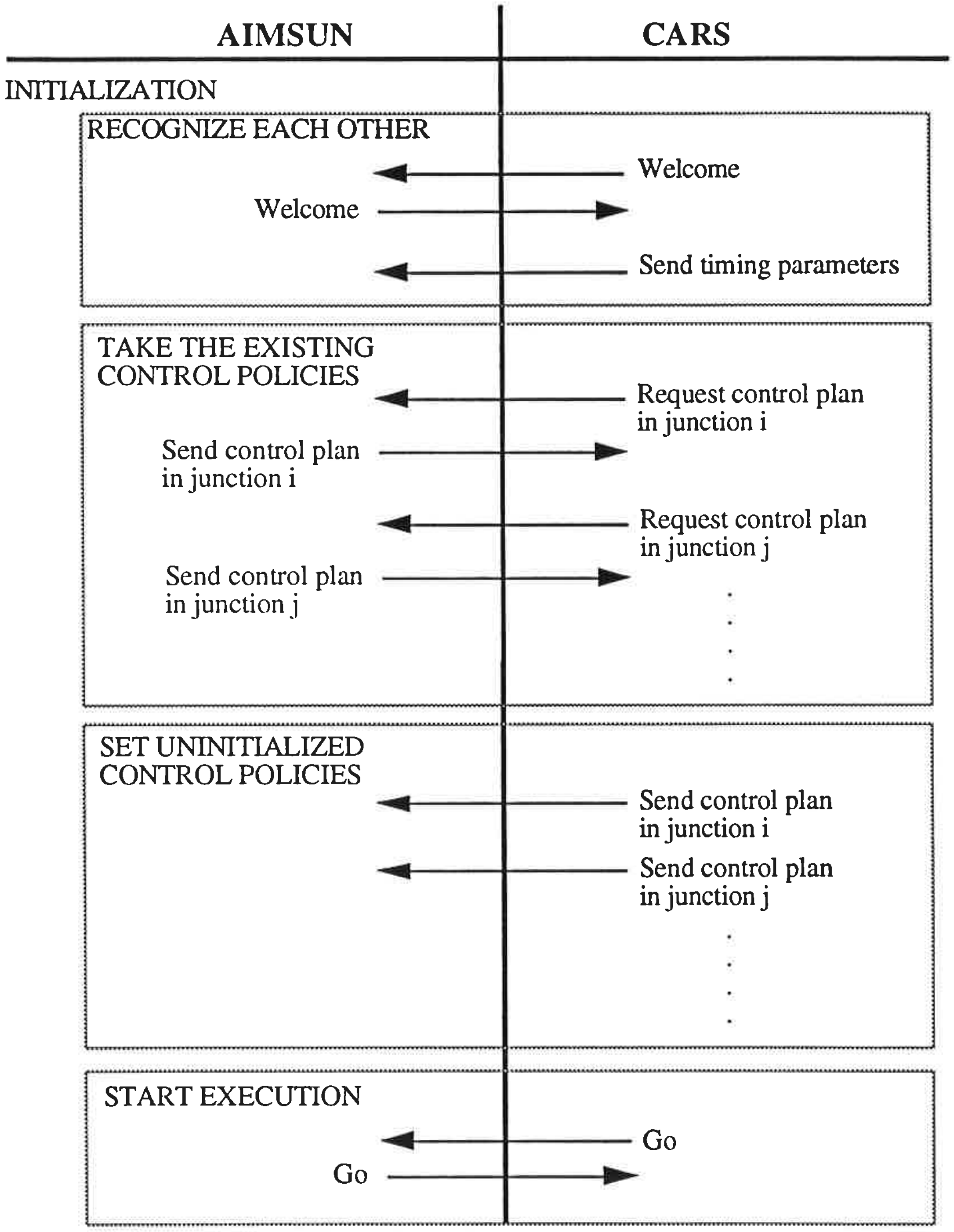

AIMSUN
CARS
INITIALIZATION
RECOGNIZE EACH OTHER
Welcome
Welcome
Send timing parameters
TAKE THE EXISTING CONTROL POLICIES
Request control plan in junction i
Send control plan in junction i
Request control plan in junction j
Send control plan in junction j
SET UNINITIALIZED CONTROL POLICIES
Send control plan in junction i
Send control plan in junction j
START EXECUTION
Go
Go

Fig. 4.1 Flow of messages between CARS V2 and AIMSUN V2 during the initialization process.

The initialization series begins by checking communication ('Welcome'). Then, CARS requests from the controllers the current state of the control policy in each junction ('Request control plan'), and each one of them responds with the current stage durations and position in the junction cycle ('Send control plan'). In the event of the control plan of a signalized junction being uninitialized, CARS sends an initial state ('Send control plan'). The process ends with a message ('Go') to begin the CARS-controlled session.

During the run, in each System Control Cycle (SCC), the controllers send to CARS the traffic measurements accumulated during the last SCC ('Request measurements' and 'Send measurements'), and receive from CARS the new control policies ('Send control plan'). A message leads the way to the next SCC ('Go') or tells the controllers that CARS will disconnect ('End').

Measurements sent by detectors to CARS V2 include not only the accumulated number of vehicles and the averaged speed, but also the occupancy; it is used to revise the network state in CARS only if the detector from which it came is capable of measuring it.

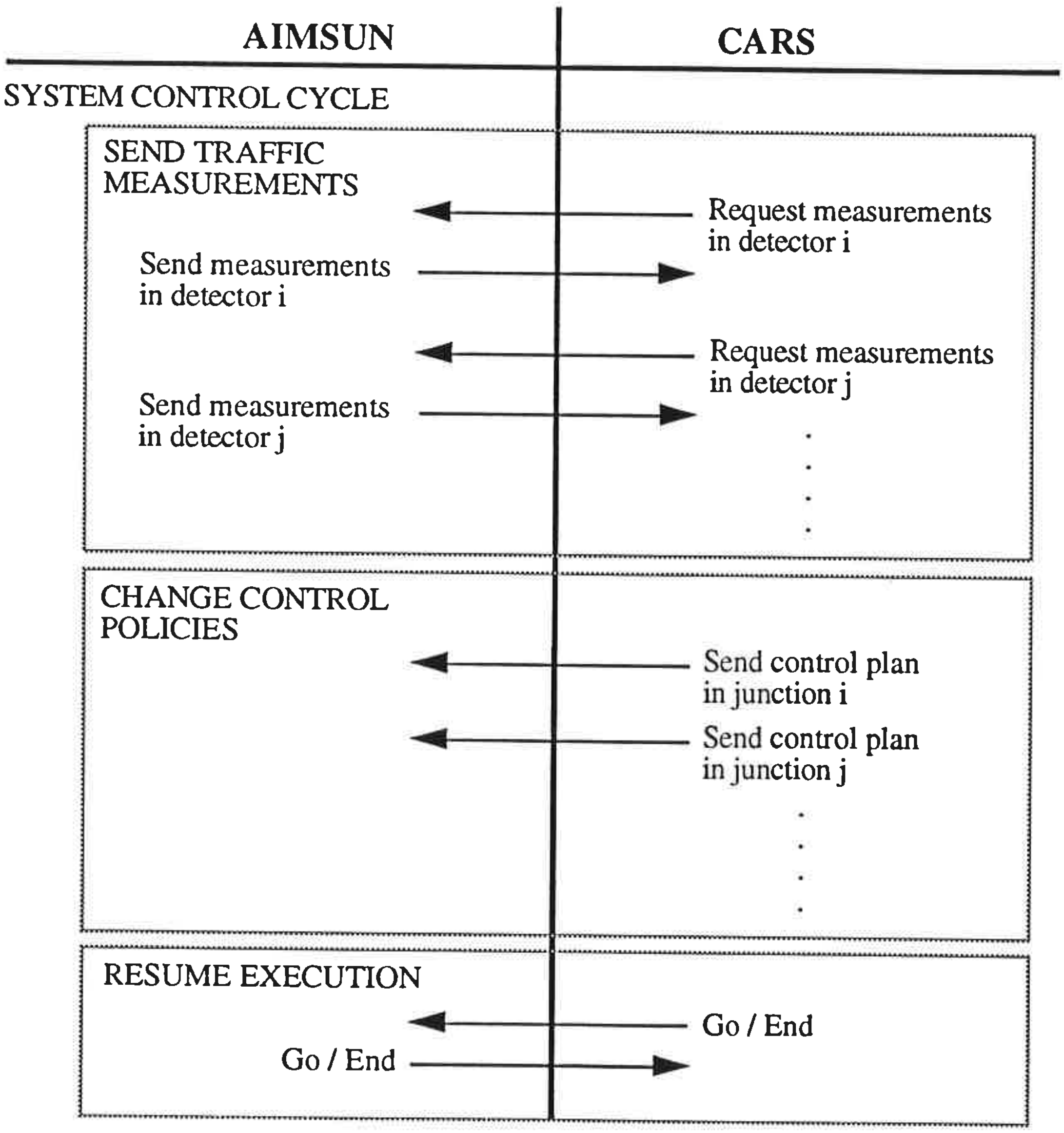

AIMSUN
CARS
SYSTEM CONTROL CYCLE
SEND TRAFFIC MEASUREMENTS
Request measurements in detector i
Send measurements in detector i
Request measurements in detector j
Send measurements in detector j
CHANGE CONTROL POLICIES
Send control plan in junction i
Send control plan in junction j
RESUME EXECUTION
Go / End
Go / End

Fig. 4.2 Flow of messages between CARS V2 and AIMSUN V2 during the execution cycle.

5. Tests

As a result of having CARS V2 integrated into the GETRAM environment, the environment for testing by microscopic simulation has improved dramatically. Networks are easily created in the common editor and then the same specification is shared by both CARS and AIMSUN. Also, the new version of the AIMSUN microscopic simulator, named AIMSUN2 (Ferrer and Barceló 1993), includes a new graphical interface that eases the task. Both AIMSUN2 and CARS V2 run now on the same computer and use the same display (Figure 5.1). AIMSUN sends detector measurements to CARS and receives new control plans.

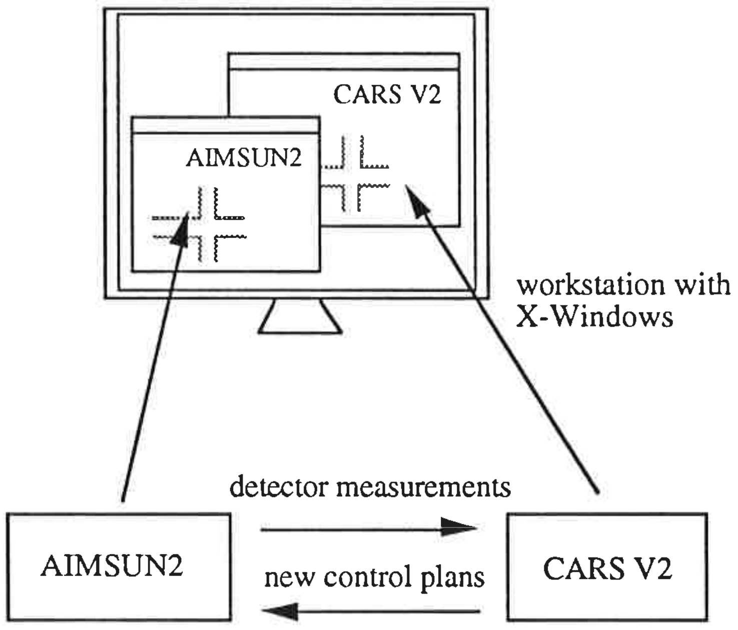


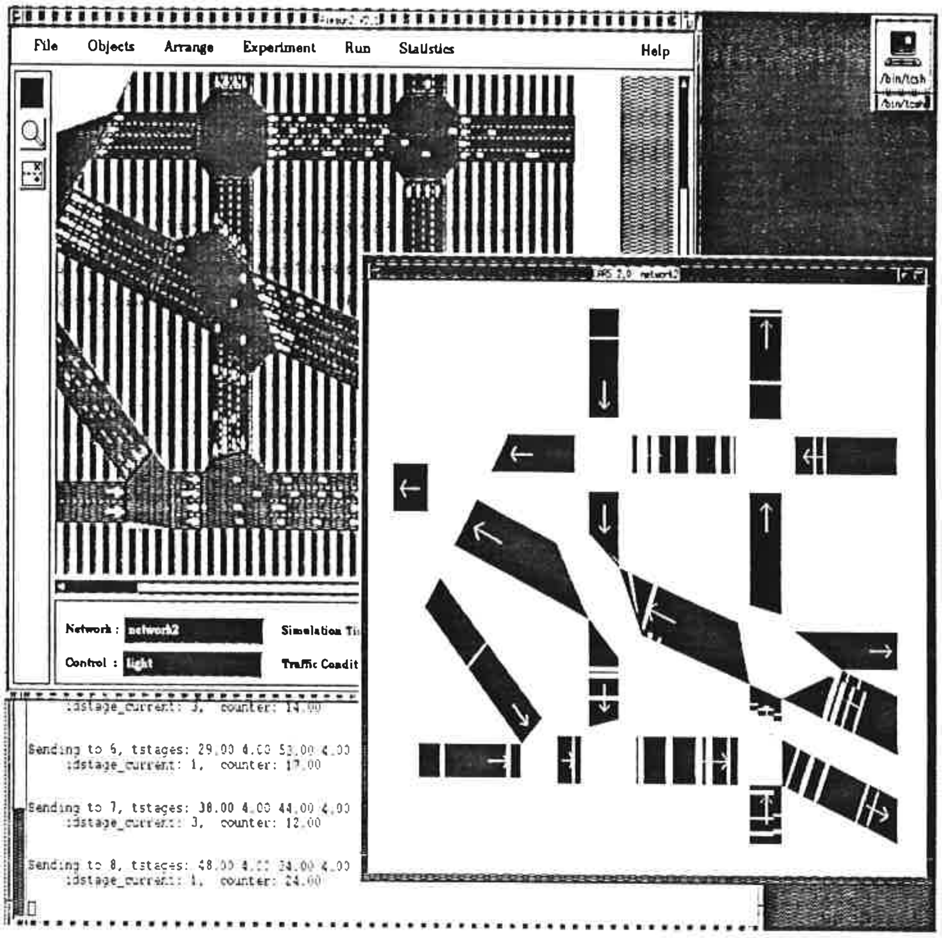
Fig. 5.1 Testing environment for CARS V2.

5.2 Tests in the previous scenarios

We start from the four scenarios that were used by the microscopic-simulation tests in PART II: junction, arterial, network1, and network2. Again, they are tested both in light and in heavy traffic conditions, as was specified in that part. The first step has been to introduce them into the GETRAM environment, using the common editor. This was a straightforward task, given the graphical facilities that the environment provides. Microscopic-simulation tests were then started (Figure 5.2), producing the results shown in Figures 5.3, 5.4 (junction in light traffic conditions), 5.5, 5.6 (junction in heavy traffic conditions), 5.7, 5.8 (arterial in light traffic conditions), 5.9, 5.10 (arterial in heavy traffic conditions), 5.11, 5.12 (network1 in light traffic conditions), 5.13, 5.14 (network1 in heavy traffic conditions), 5.15, 5.16 (network2 in light traffic conditions), 5.17 and 5.18

(network2 in heavy traffic conditions). In all of them, the set of changes is built according to the first method with 4-sec. maxvar, the duration of the SCC is 2 deltas, and delta = 1 sec..

Summarizing these results, CARS's control plans are, on average, 20% more effective than their respective fixed-control plans. An important point is that the advantage is maintained as traffic conditions become congested, in contrast with what happened with the ACTS-based system in PART I. The problem that CARS V1 had, i.e., that starting from a far-from-optimal control plan could yield bad results, has also been solved; in fact, in all these tests, CARS V2 started from a bad control plan. For example, when testing for the junction in heavy traffic conditions, CARS started from the control plan that is optimal for light traffic conditions; had it started from the heavy traffic control plan, the difference in total average delay is of only 2 seconds.



Network : network2
Control : light
Simulation Ti
Traffic Condit

```
      :dstage_current: 3,  counter: 14.00

Sending to 6, tstages: 29.00 4.00 53.00 4.00
      :dstage_current: 1,  counter: 17.00

Sending to 7, tstages: 38.00 4.00 44.00 4.00
      :dstage_current: 3,  counter: 12.00

Sending to 8, tstages: 48.00 4.00 34.00 4.00
      :dstage_current: 1,  counter: 24.00
```

Fig. 5.2 AIMSUN2 and CARS V2 on the same screen when testing network2.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	2 : 34	2 : 27
Delay travel time (secs. / (veh * km))	1 : 40	1 : 29
Av. speed (km / h)	30.55	31.8

Fig. 5.3 Results in junction for light traffic conditions.

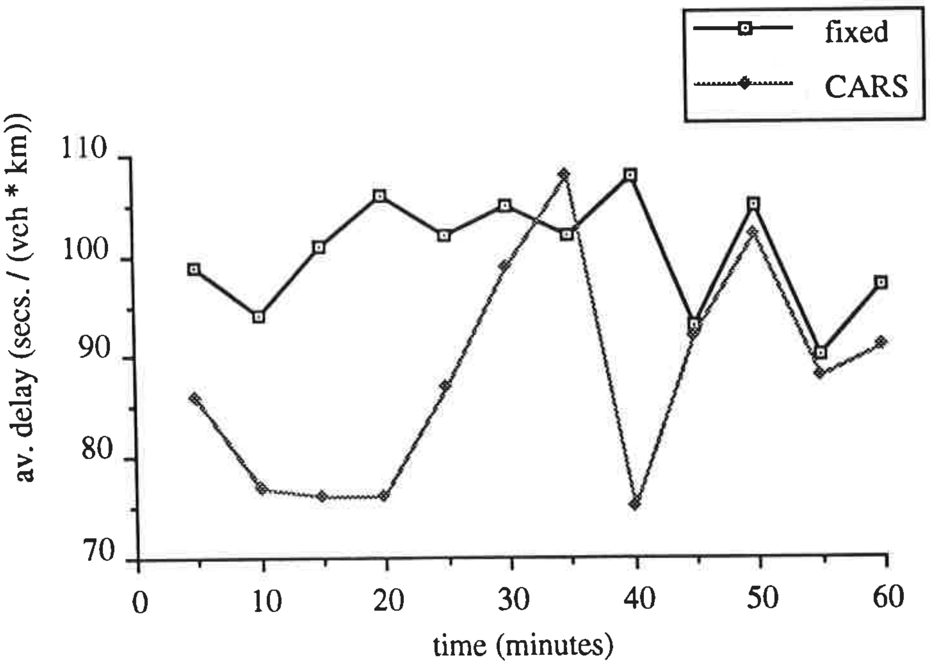


Fig. 5.4 Temporal evolution of delays in junction for light traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	4 : 48	4 : 03
Delay travel time (secs. / (veh * km))	3 : 54	3 : 09
Av. speed (km / h)	22.19	26.4

Fig. 5.5 Results in junction for heavy traffic conditions.

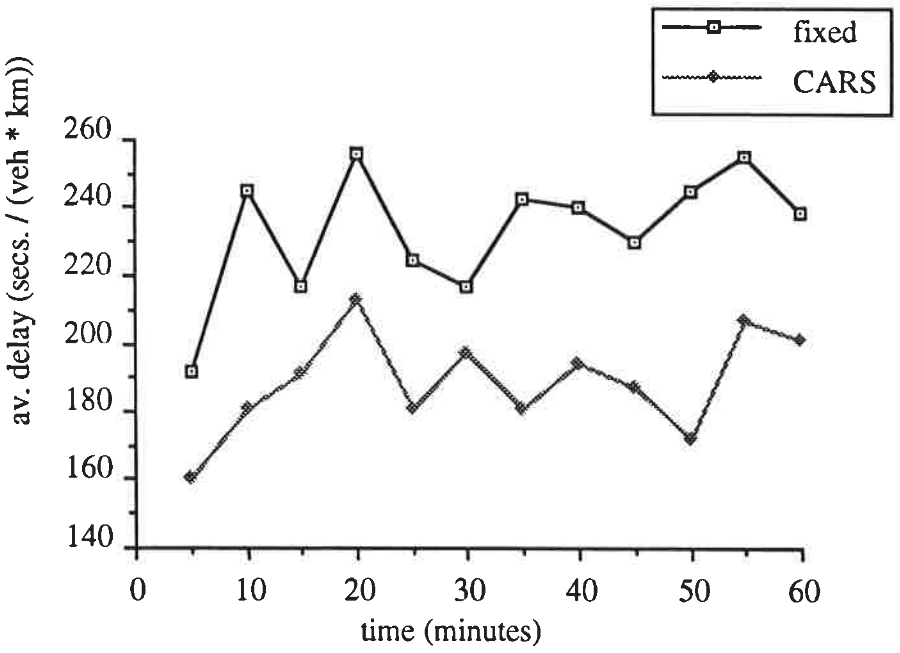


Fig. 5.6 Temporal evolution of delays in junction for heavy traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	5 : 05	4 : 50
Delay travel time (secs. / (veh * km))	4 : 20	3 : 45
Av. speed (km / h)	13.27	14.3

Fig. 5.7 Results in arterial for light traffic conditions.

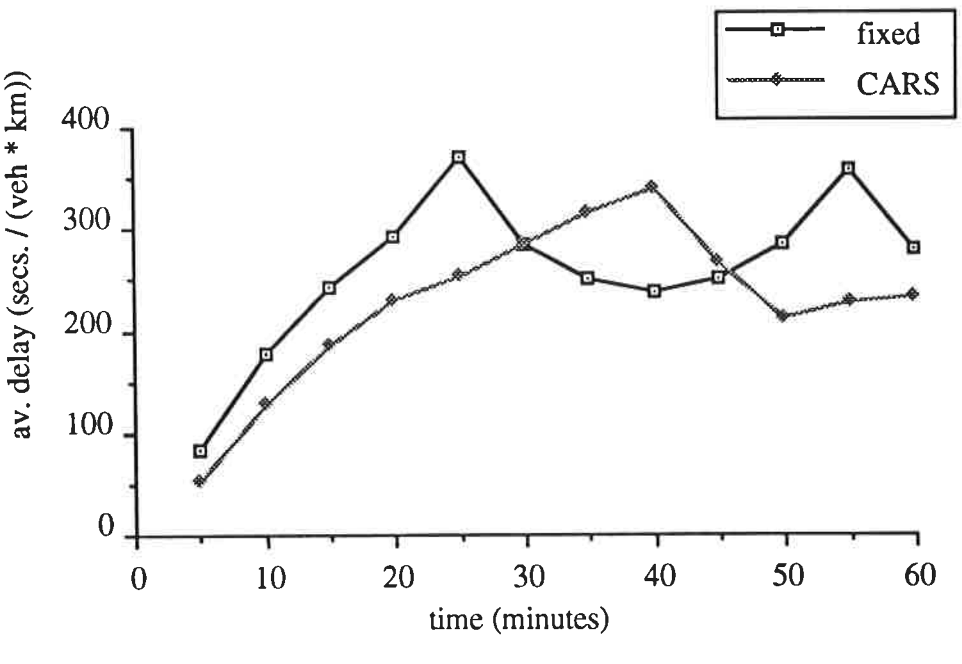


Fig. 5.8 Temporal evolution of delays in arterial for light traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	7 : 02	5 : 55
Delay travel time (secs. / (veh * km))	6 : 30	5 : 01
Av. speed (km / h)	13.18	15.31

Fig. 5.9 Results in arterial for heavy traffic conditions.

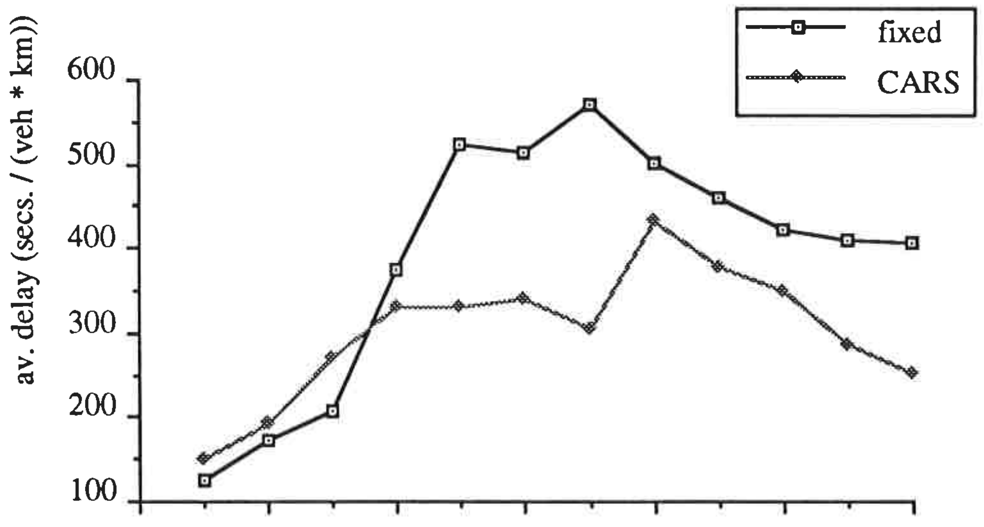


Fig. 5.10 Temporal evolution of delays in arterial for heavy traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	3 : 21	2 : 51
Delay travel time (secs. / (veh * km))	2 : 26	1 : 55
Av. speed (km / h)	20.91	22.9

Fig. 5.11 Results in network1 for light traffic conditions.

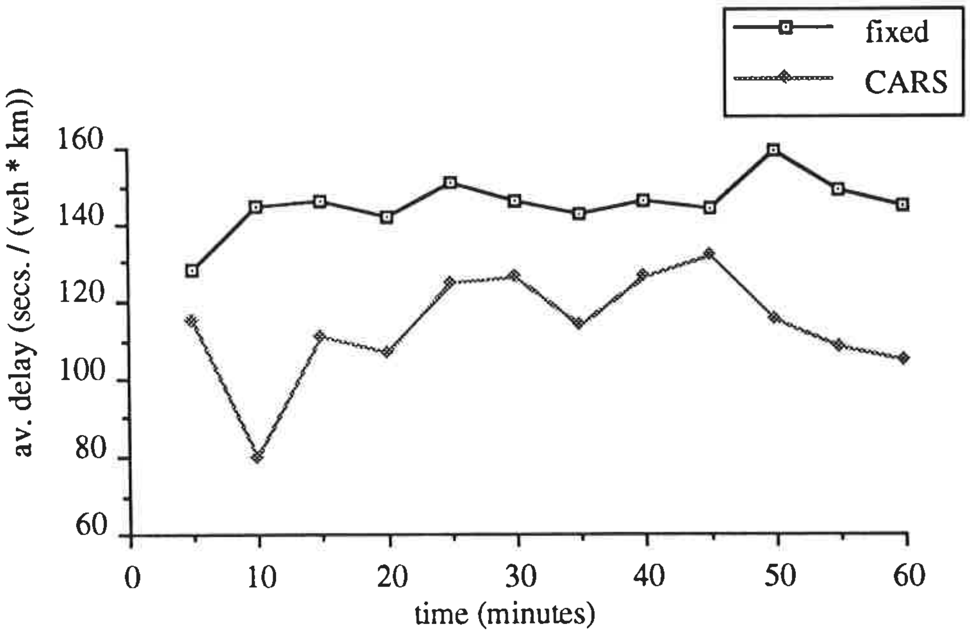


Fig. 5.12 Temporal evolution of delays in network1 for light traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	5 : 13	3 : 18
Delay travel time (secs. / (veh * km))	3 : 31	2 : 32
Av. speed (km / h)	24.6	25.1

Fig. 5.13 Results in network1 for heavy traffic conditions.

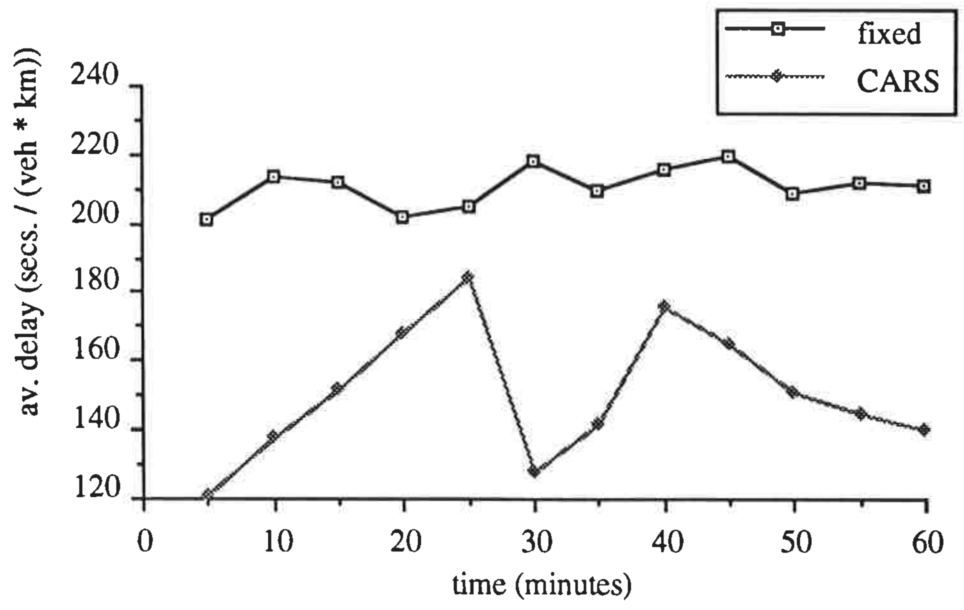


Fig. 5.14 Temporal evolution of delays in network1 for heavy traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	3 : 37	2 : 98
Delay travel time (secs. / (veh * km))	2 : 44	2 : 05
Av. speed (km / h)	19.90	20.8

Fig. 5.15 Results in network2 for light traffic conditions.

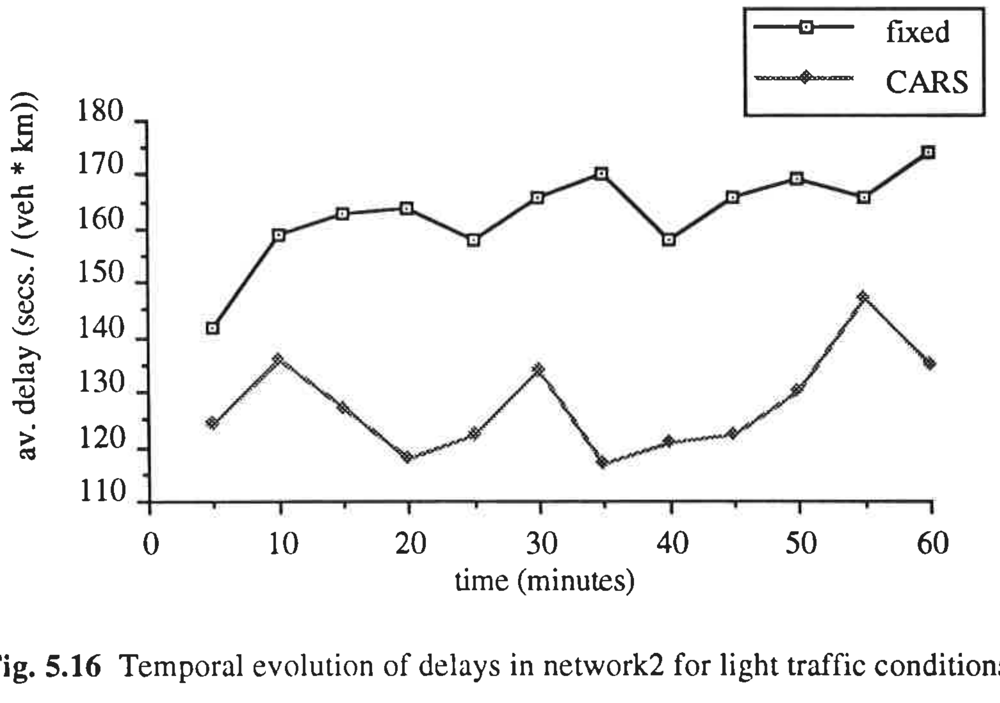


Fig. 5.16 Temporal evolution of delays in network2 for light traffic conditions.

measurements \ control	fixed	CARS
Av. travel time (secs. / (veh * km))	4 : 20	3 : 30
Delay travel time (secs. / (veh * km))	3 : 27	2 : 50
Av. speed (km / h)	21.5	24.5

Fig. 5.17 Results in network2 for heavy traffic conditions.

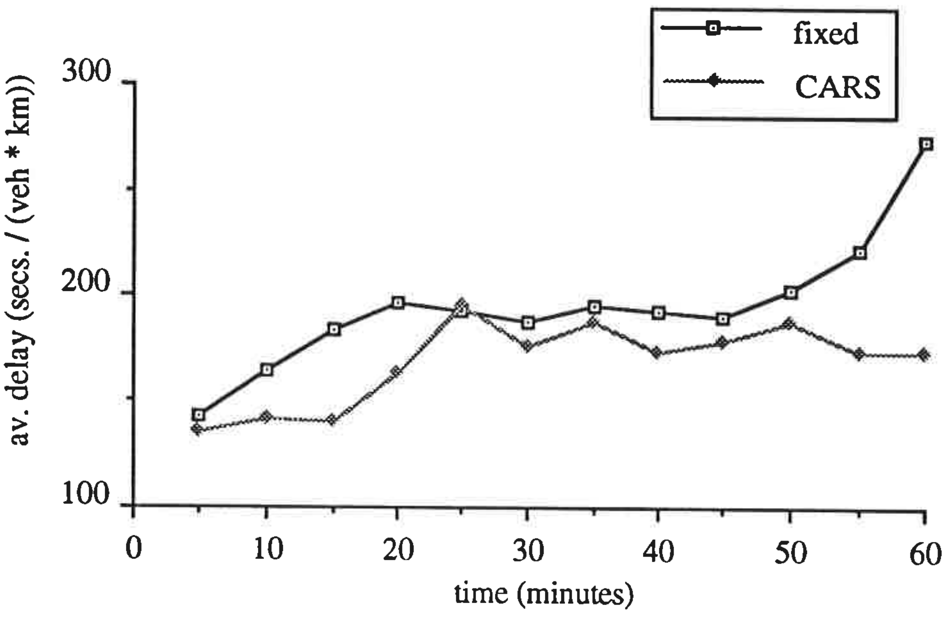


Fig. 5.18 Temporal evolution of delays in network2 for heavy traffic conditions.

5.3 Tests in a fifth scenario

In the last part of this thesis the opportunity has arisen to work with real-world data concerning an isolated junction located in Minneapolis (Minnesota, USA). In this fifth scenario, there is real-world data about geometry, vehicle arrivals, fixed-control plans, and even the parameters of a vehicle-actuated control. Data is partitioned into three times of day, corresponding to an off-peak (7:30p.m. - 9:00p.m.), an a.m.-peak (7:15a.m. - 8:45a.m.), and a p.m.-peak period (3:45p.m. - 5:15p.m.), as it is described below.

This fifth scenario, junction2, is an isolated junction with a total of 8 sections: 4 entrance sections and 4 exit sections (see Figure 5.20). Turning probabilities vary from period to period, as shown in Figures 5.21, 5.22, and 5.23.

In all three periods, the junction has four phases (Figure 5.24); in the second phase, there are conflicting turnings where the left-turning vehicles give way. Clearance (interphase) periods take the values 5.4, 6., 4.8, and 4.8, from the first to the fourth phase, respectively — all them contain 3.6 secs. amber. For the third phase, the clearances pertain to the left-northbound arrow, the southbound arrow remains green. For the fourth phase, the clearances pertain to the left southbound arrow, the southbound through indication remaining green.Two sets of fixed-control plans have been supplied, the first one by the city of Minneapolis (Figure 5.25), the second by the Dept. of Transportation of the University of Minneapolis (Figure 5.26); in the following results they will usually be referred to as 'fixed' and 'fixed2'.

Three sets of vehicle arrivals, corresponding to the three periods, are shown in Figures 5.27, 5.28, and 5.29. The maximum speed in all sections is 55 km/h.

Additionally, this junction can be operated, in the real world, by a vehicle-actuated control. To this aim, left-turning bays have area detectors, and all the other approaching lanes have a 'calling detector' located close to the stop line, and a point detector located at approximately 100 m from the stop line. This detector placement allows a full-actuated operation where all the stages are vehicle-actuated, specifically with density-controller features — variable initial interval and gap reduction —, at least in the first, second and fourth phases; the third phase is activated through presence. The parameters, taken from a source at the University of Minneapolis, are:

- In the first and second phases: 10 secs. initial interval, 4.5 secs. vehicle interval, 35 secs. maximum interval, and 4 secs. per actuation. Gap reduction has been activated, with 20 secs. of time before reduction, 20 secs. time for reduction, and 0.75 secs. minimum gap.
- In the third phase: 5 secs. initial interval, 2 secs. vehicle interval, and 10 secs. maximum interval.
- In the fourth phase: 5 secs. initial interval, 2 secs. vehicle interval, 10 secs. maximum interval, and 4 secs. per actuation. Gap reduction parameters are the same as in the first and second phases.

In order to test these features, a vehicle-actuated control system has been programmed. The system communicates with AIMSUN2 following the same messaging protocol as CARS V2.

The last control mode to be tested is the demand-responsive control, represented by CARS V2. As in the previous tests, the set of changes is built according to the first method with 4 secs. maxvar, the duration of the SCC is 2 deltas, and delta = 1 secs..

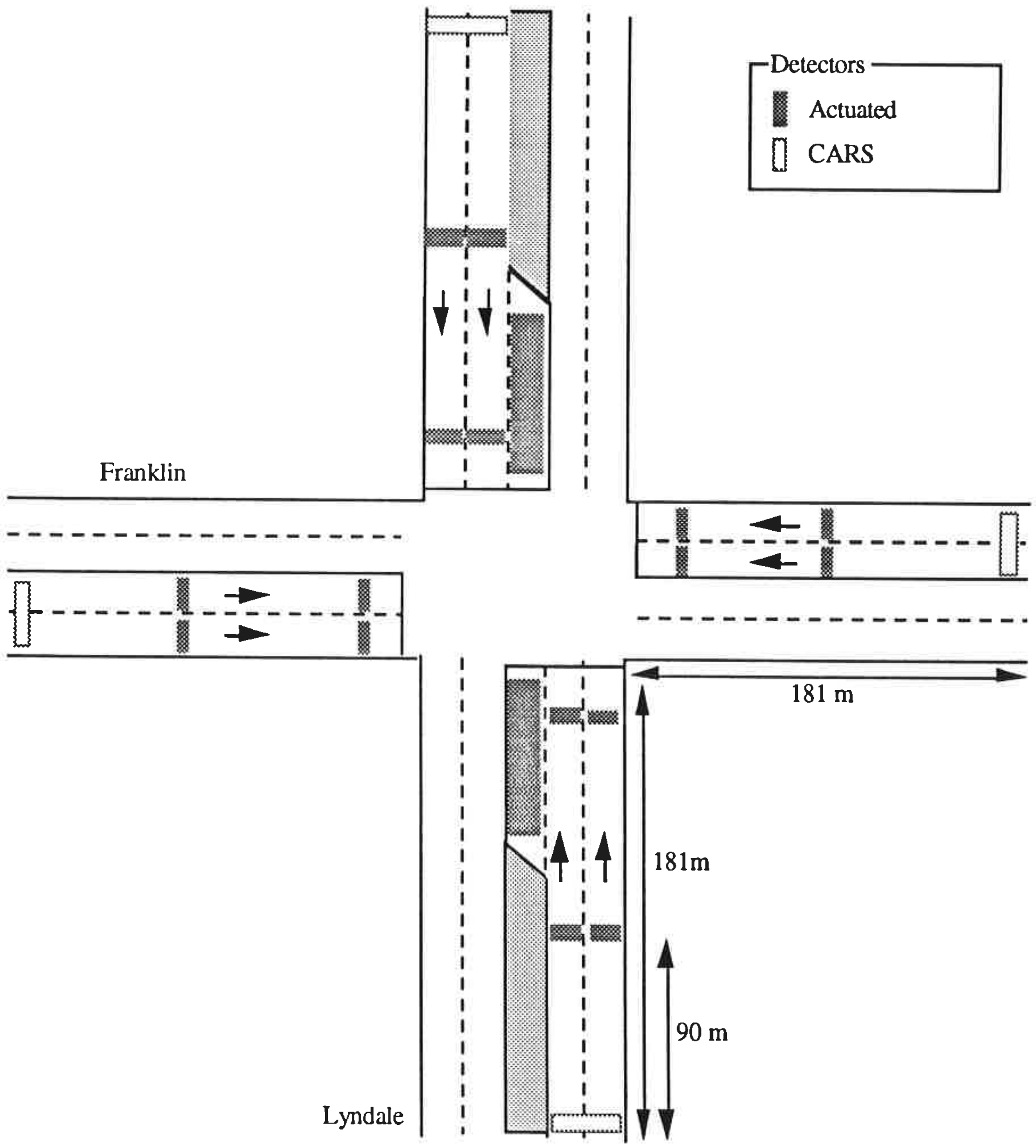


Fig. 5.20 Fifth scenario, an isolated junction used to test CARS V2.

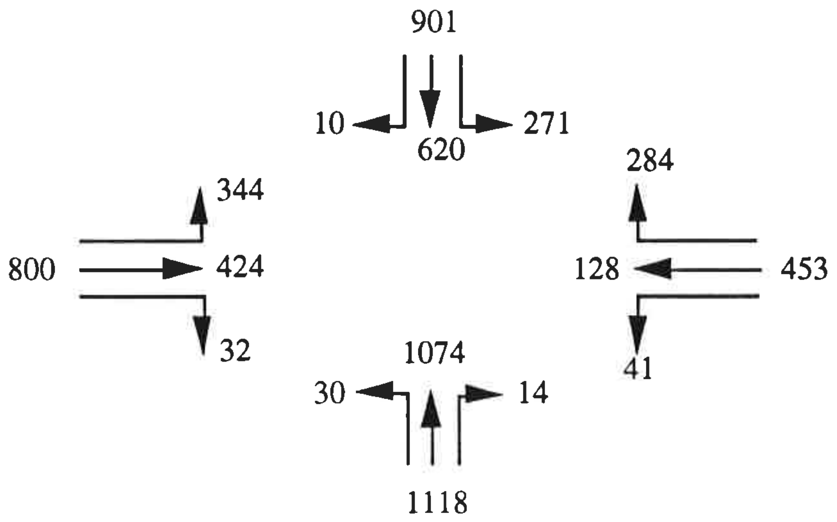


Fig. 5.21 Turning percentages corresponding to a.m.-peak hour demand at the isolated junction in Figure 6.20.

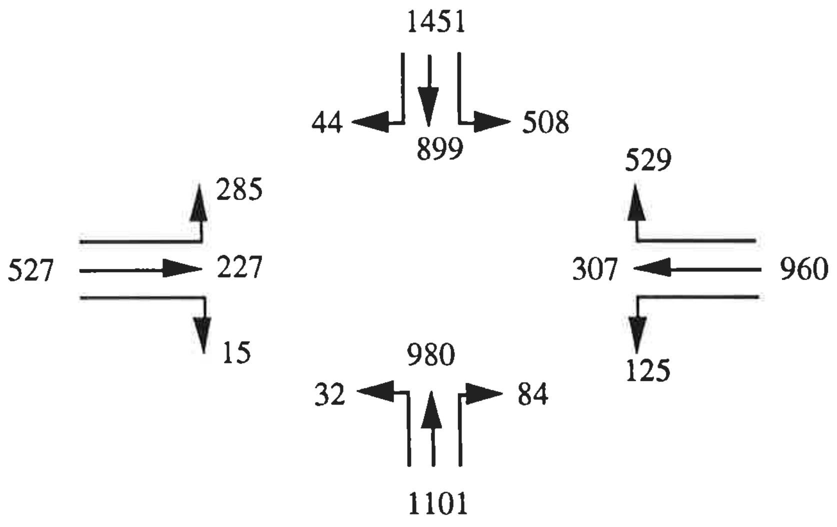


Fig. 5.22 Turning percentages corresponding to p.m.-peak hour demand at the isolated junction in Figure 6.20.

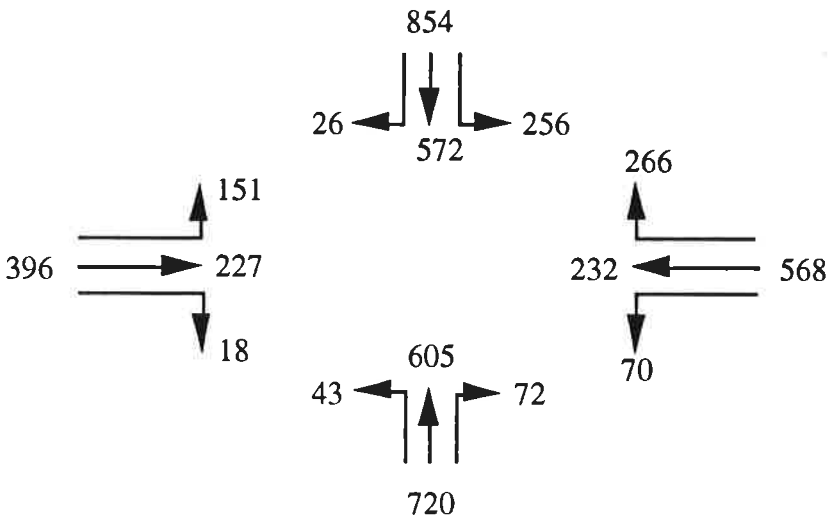


Fig. 5.23 Turning percentages corresponding to off-peak hour demand at the isolated junction in Figure 6.20.

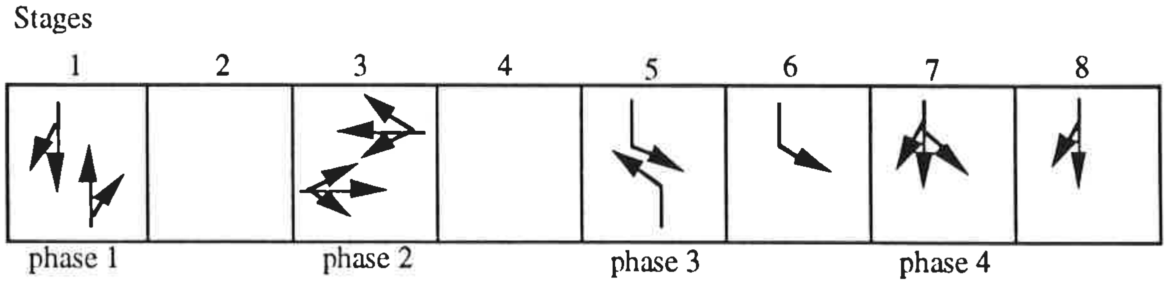


Fig. 5.24 Stage sequence with rights of way at the isolated junction in Figure 6.20.

	PHASE 1	PHASE 2	PHASE 3	PHASE 4
off-peak	34	33	14	9
a.m.-peak	34	33	14	9
p.m.-peak	35	32	14	9

Fig. 5.25 Current pretimed phase durations, in seconds, at the isolated junction in Figure 6.20.

	PHASE 1	PHASE 2	PHASE 3	PHASE 4
off-peak	32	31	13	9
a.m.-peak	38	32	11	9
p.m.-peak	37	34	10	9

Fig. 5.26 Alternative pretimed phase durations, in seconds, at the isolated junction in Figure 6.20.

TIME	NB	EB	SB	WB
19:30	864	360	696	552
19:35	576	360	720	744
19:40	768	288	600	840
19:45	696	168	1176	528
19:50	840	216	912	552
19:55	792	408	888	504
20:00	792	288	912	600
20:05	720	504	912	408
20:10	888	264	864	624
20:15	768	312	888	480
20:20	1056	408	1080	768
20:25	768	648	960	864
20:30	600	336	960	720
20:35	648	384	696	792
20:40	576	312	864	936
20:45	600	456	888	504
20:50	816	312	840	744
20:55	648	288	816	504
21:00	720	336	1008	552

Fig. 5.27 Vehicle arrivals during the off-peak period.

TIME	NB	EB	SB	WB
7:15	1008	552	1272	504
7:20	1224	672	1248	552
7:25	1056	960	1128	408
7:30	1296	696	936	528
7:35	1176	1152	1224	432
7:40	1080	1224	1224	456
7:45	1296	888	1200	528
7:50	1344	768	936	576
7:55	1176	960	1080	432
8:00	1128	624	1200	360

8:05	1248	768	1152	504
8:10	1032	792	1200	576
8:15	1128	864	1488	384
8:20	1416	696	1176	528
8:25	984	600	1152	576
8:30	1032	528	1440	504
8:35	1248	576	1176	576
8:40	1272	696	1032	456
8:45	984	576	1152	456

Fig. 5.28 Vehicle arrivals during the a.m.-peak period.

TIME	NB	EB	SB	WB
3:45	1176	648	1392	864
3:50	1200	672	1416	1032
3:55	1008	1032	1800	576
4:00	1056	840	1488	912
4:05	1008	768	1152	1176
4:10	1320	672	1656	1008
4:15	1128	984	1512	1224
4:20	1032	1008	1728	984
4:25	1392	744	1512	936
4:30	1080	744	1632	720
4:35	1104	696	1704	984
4:40	1272	768	1344	816
4:45	1032	744	1680	888
4:50	1128	648	1752	1080
4:55	912	672	1848	744
5:00	984	600	1512	888
5:05	1128	456	1632	1056
5:10	1200	384	1104	864
5:15	1320	504	1320	912

Fig. 5.29 Vehicle arrivals during the p.m.-peak period.

Before applying CARS to this junction, a problem has to be solved in order to model left-turning bays. Previous networks did not contain left-turning bays and therefore this problem did not arise. The problem is that PACKSIM — the internal simulation model in CARS — cannot deal directly with them. To work around this problem, the section has to be decomposed in CARS, adding a dummy node as shown in Figure 5.30.

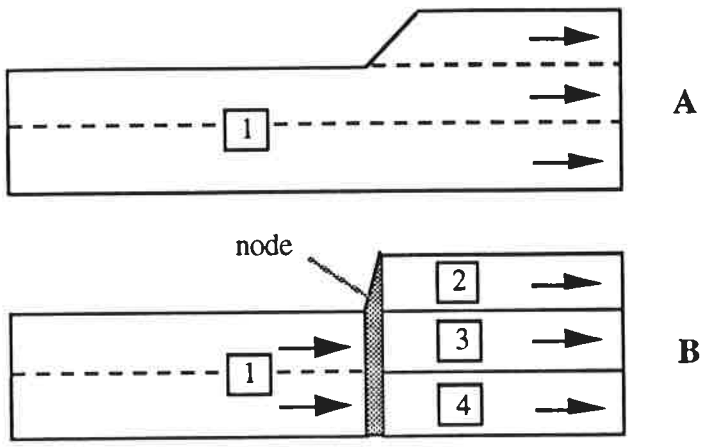


Fig. 5.30 A section with a left-turning bay (A) has to be decomposed into several sections and one dummy node.

We begin the tests by comparing the performance of the various control methods in the three periods. The results (Figures 5.31, 5.32, 5.33) show that CARS is more effective than the best fixed-control plan in all cases, its advantage (see Figure 5.34) augmenting from light traffic conditions — 12% in the off-peak period — to congestion — 45% in the a.m.-peak and 26% in the p.m.-peak period. Similarly, CARS is more effective than vehicle-actuated control, showing a 33%, 46%, and 27% advantage in the off-peak, a.m.-peak, and p.m.-peak periods, respectively. Therefore, even in light traffic conditions, CARS shows a better performance than the other control methods. Examining the temporal evolution of the delay in each period (Figures 3.35, 3.36, and 3.37), the advantage of CARS is fairly constant. In these tests, CARS started from a random initial control plan; in fact, contrary to what happened in the first version of CARS, further tests indicate that results do not seem to depend on the initial control plan, thanks to using the second method with maxvar = 4 secs.

measurements \ control	fixed	fixed2	vehact	CARS
Av. travel time (secs. / (veh * km))	2 : 27	2 : 13	2 : 36	2 : 04
Delay travel time (secs. / (veh * km))	1 : 32	1 : 17	1 : 41	1 : 08
Av. speed (km / h)	32.05	34.40	32.70	35.79

Fig. 5.31 Results in junction2 during the off-peak period.

measurements \ control	fixed	fixed2	vehact	CARS
Av. travel time (secs. / (veh * km))	4 : 29	4 : 10	4 : 13	2 : 43
Delay travel time (secs. / (veh * km))	3 : 33	3 : 14	3 : 17	1 : 48
Av. speed (km / h)	23.29	26.42	26.88	31.68

Fig. 5.32 Results in junction2 during the a.m.-peak period.

measurements \ control	fixed	fixed2	vehact	CARS
Av. travel time (secs. / (veh * km))	6 : 03	5 : 48	5 : 52	4 : 32
Delay travel time (secs. / (veh * km))	5 : 08	4 : 52	4 : 57	3 : 37
Av. speed (km / h)	18.43	19.83	25.30	23.08

Fig. 5.33 Results in junction2 during the p.m.-peak period.

period \ advantage of CARS	fixed	fixed2	vehact
off-peak	28%	12%	33%
a.m.-peak	50%	45%	46%
p.m.-peak	30%	26%	27%

Fig. 5.34 Advantage of CARS against the other control methods, measured in difference between delays. Results extracted from Figures 5.28, 5.29, and 5.30.

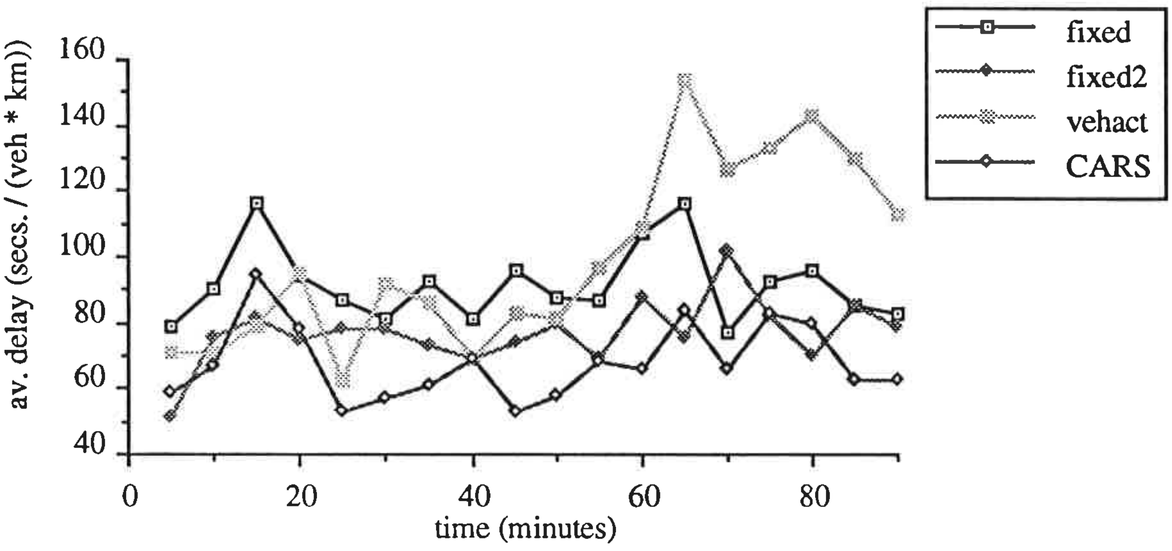


Fig. 5.35 Temporal evolution of delays at junction2 during the off-peak period.

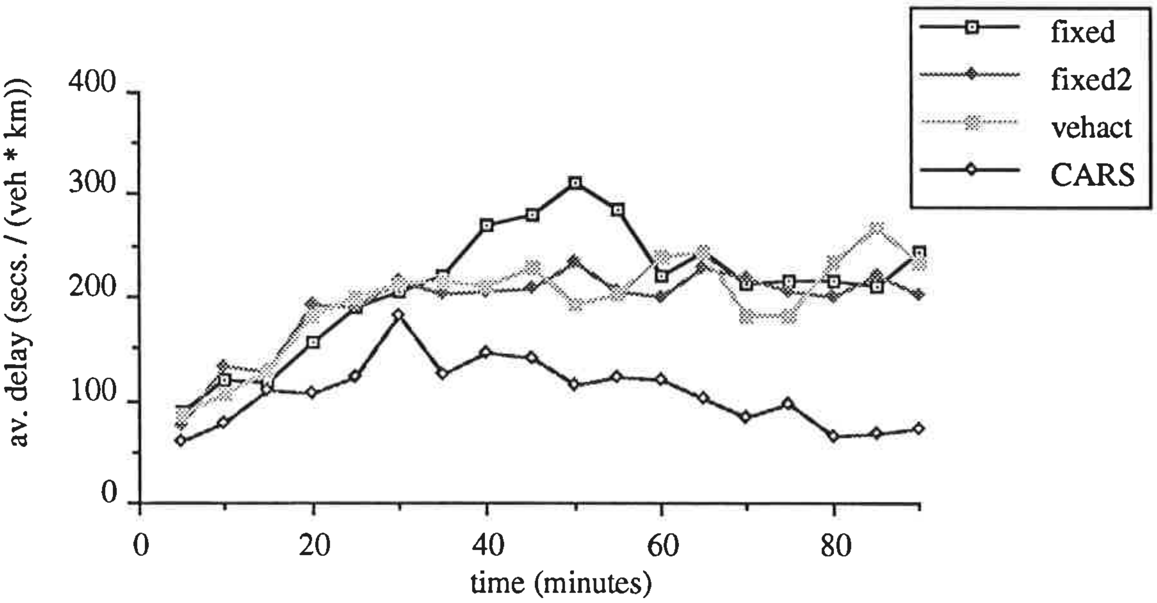


Fig. 5.36 Temporal evolution of delays at junction2 during the a.m.-peak period.

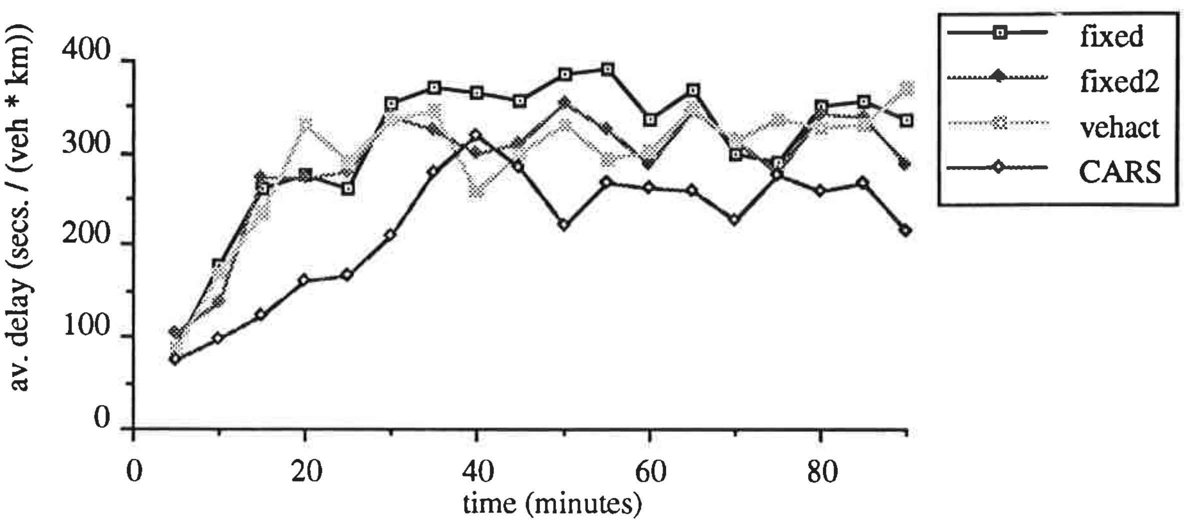


Fig. 5.37 Temporal evolution of delays at junction2 during the p.m.-peak period.

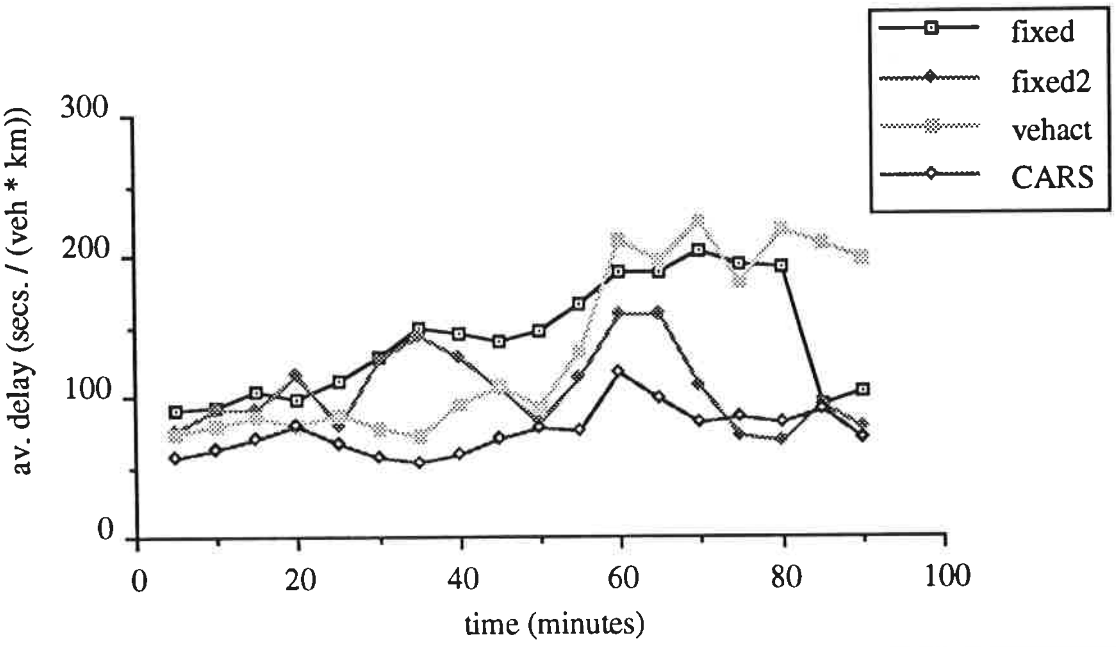
Another test consists of, given certain traffic conditions and fixed-control plans, augmenting the flow of vehicles in one direction and comparing the performance of these fixed-control plans against both vehicle-actuated and CARS controls. Particularly, it is interesting to know whether, in such an apparently well-suited situation for vehicle-actuated control, CARS will maintain its performance advantage. With this aim, we start from the off-peak conditions and augment the flow of vehicles entering the north bound approach in increments of 200 veh/h on top of the data from Figure 5.27. Results are shown in Figure 5.38, where the first two columns correspond to the fixed-control plan that was supplied for the original off-peak conditions. As expected, maintaining the same fixed-control plan produces increasingly bad results but, curiously, vehicle-actuated control does not ameliorate the results either. In contrast, CARS's performance is always better than the others, and it deteriorates smoothly.

traffic conditions / control	fixed	fixed2	vehact	CARS
off-peak	92	77	101	68
+ 200 veh/h	97	79	85	74
+ 400 veh/h	140	107	139	76
+ 600 veh/h	193	172	175	116
+ 800 veh/h	194	183	227	112
+ 1000 veh/h	198	183	195	164

Fig. 5.38 Average delays, in sec / (veh * Km), produced in the junction2 when augmenting the traffic in 200 veh/h increments from off-peak conditions.

An explanation is needed regarding this data (Figure 5.38), as average delays grow very slowly from a certain value, about 170 secs.. This is due to the fact that AIMSUN2 does not consider the delay in the vehicles that cannot enter the junction; therefore, once a section is totally full of stopped vehicles, the average delay in that section does not grow.

On examining the evolution of the delay in the junction for various flow increments (Figure 5.39), one sees that CARS consistently reduces the delay, maintaining its advantage throughout almost all the time intervals. If delay is measured only in the north-bound approach (Figure 5.40), CARS shows an even bigger advantage, producing far less delays than the other control strategies.



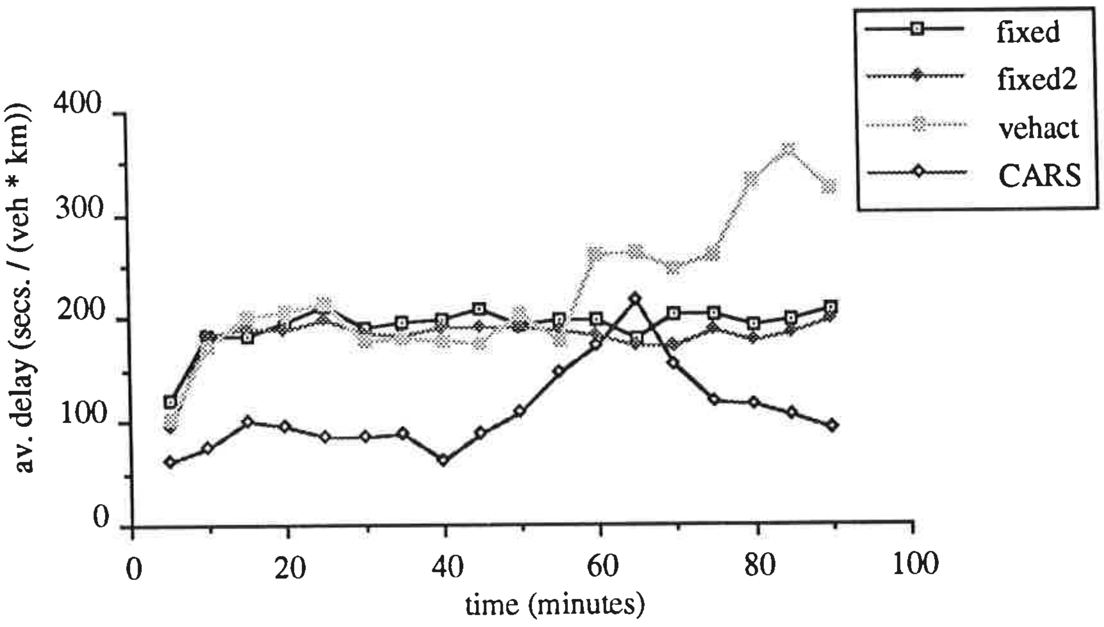
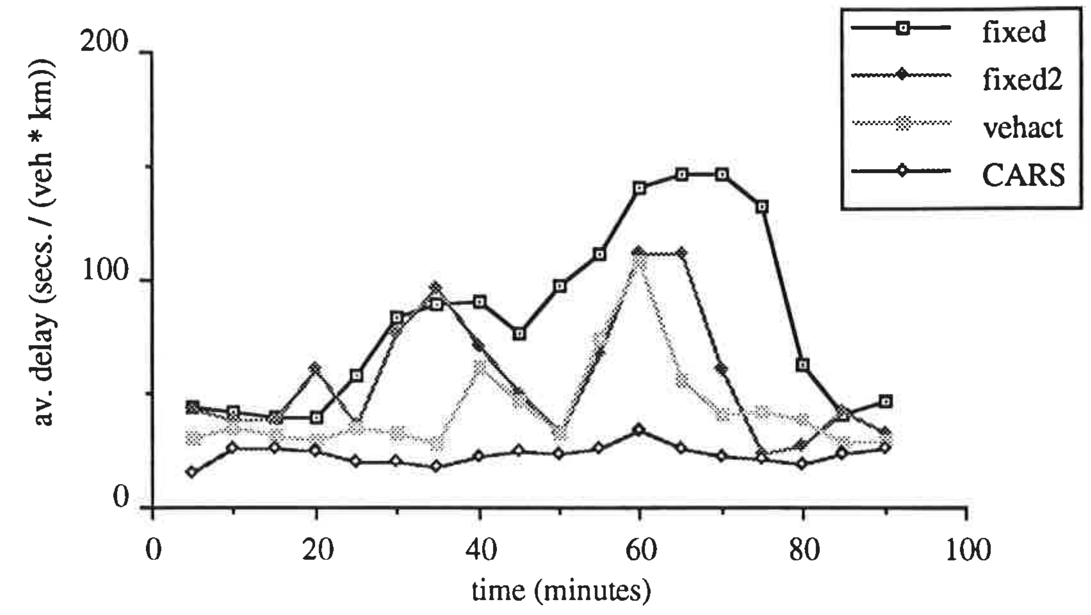


Fig. 5.39 Temporal evolution of delays at junction2, starting from off-peak conditions and augmenting the flow of vehicles entering the north bound approach in increments of 400 veh/h (top) and 800 veh/h (bottom).



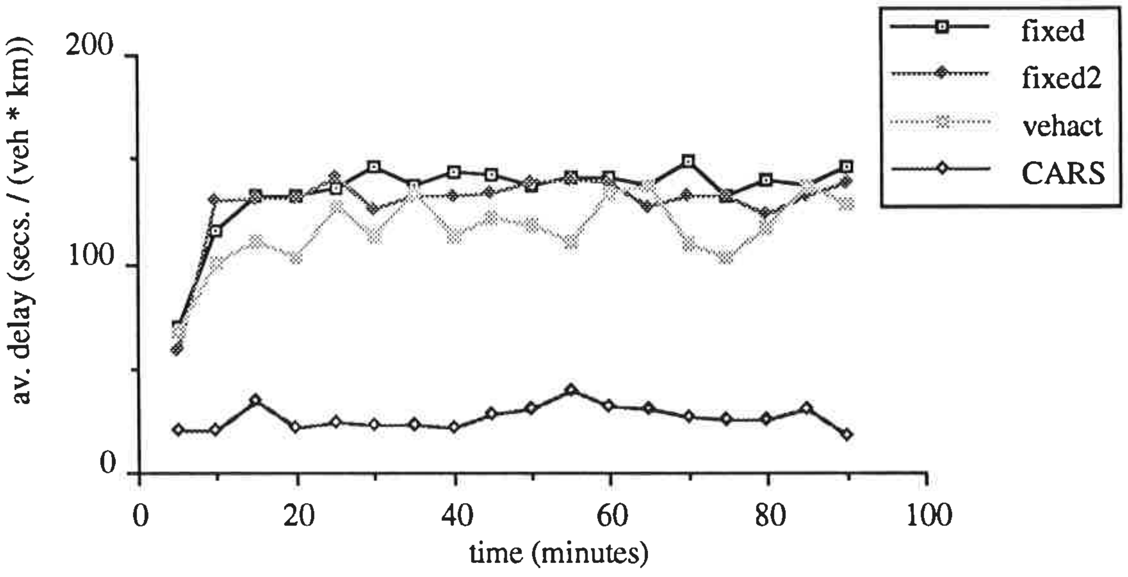


Fig. 5.40 Temporal evolution of delays only in the north bound approach at junction2 as opposed to total delay at the junction (Figure 5.37), corresponding to increments of 400 veh/h (top) and 800 veh/h (bottom).

6. Conclusions and future research

This part proposes some ameliorations to the demand-responsive system developed in PART II, CARS V1, and integrates it into the traffic modeling and analysis environment described in PART III, GETRAM. The resulting system, named CARS V2, solves the previous problems in that it improves effectiveness in light traffic conditions, and it is capable of starting from a far from optimal control plan. In order to solve these issues, new, more computing-intensive algorithms have had to be introduced that can nevertheless still be applied to small networks — up to nine junctions in a SUN SparcStation 10 with one processor. Moreover, given the pace at which advances in computing technologies and parallelism are introduced, application to bigger networks can be envisaged in the near future.

Microscopic-simulation testing of the system show that CARS V2 demand-responsive control has significant advantages over optimal fixed-control plans in a wide range of traffic conditions — from light to heavy traffic conditions — with an average expected 20% improvement. However, it is in unexpected conditions that the initially optimal fixed-control plan turns out to be inadequate, and that CARS V2 shows dramatic advantages. Also, tests at an isolated junction show the system to be more effective that vehicle-actuated control, even in light traffic conditions.

This research had two objectives. The first was to explore, starting from a review of the state of the art in demand-responsive systems, new models and methods that could improve effectiveness in a wide range of traffic conditions and on various types of networks and geometries — from isolated junctions to highly signalized networks. This has produced the CARS V1 and CARS V2 demand-responsive systems. The second objective was to provide an easy-to-use but powerful testing environment; because of the lack of tools in this area, this entailed providing a generic environment for traffic analysis and modeling, GETRAM, that surpasses the needs of merely testing demand-responsive strategies, being capable of hosting different traffic models and interchanging data between them.

Contributions to the field can be classified according to these two objectives. As for the first one, the demand-responsive control, it incorporates the following new elements:

- An internal mesoscopic simulation model, PACKSIM, that models the traffic as packets of vehicles moving according to an adhoc 'packet-following' model and stopping in horizontal queues — at a given moment, there can exist multiple groups of stopped vehicles in a link. Forbidden zones in the intersections as well as mergings and give-ways are carefully modeled. A packet can join other packets or can subdivide and thus contains a variable, floating number of vehicles. Special emphasis has been put on queue modeling in order to deal with congested situations, accurately evaluate free space, and achieve a progressive signalization in arterials.
- A very flexible system control cycle, i.e., the series of tasks that the system continuously repeats, that allows wide real-time requirements. This is mainly due to the introduction of the 'stepping ahead' task between feedback and planning. As a result, the system does not need to plan in each time interval, so that small time intervals —

such as 1 sec., or even less — can be used without requiring a powerful computing platform.

- It accepts an arbitrary number and positioning of detectors. Because this affects the performance of the system, certain rules are recommended in order to assure the effectiveness of the demand-responsive control.
- It uses prediction models based on real-time measured traffic information. Predictors cover three areas: turning percentages, vehicle arrivals, and free space. In all three, predictors try to take advantage of the knowledge of the upstream traffic in the network — vehicles arriving from upstream sections and the signal plan in upstream junctions and they are automatically updated when the control plan changes.
- Although producing acyclic control plans, the system can successfully coordinate adjacent junctions without explicitly considering offsets. The resulting control can be influenced by specifying critical junctions and progressive arterials, so that the traffic can be given priority in certain directions.
- The autoadaptive algorithm tests small changes in the current control plan. The set of changes is different from that of previous systems, and has been reduced to strike a compromise between effectiveness and computational requirements. Moreover, the system can start from a control plan that is not suitable for the initial traffic conditions. All considered, although the computational needs of PACKSIM preclude searching in larger sets of changes, the strategy pays off in that efficiency is maintained in a wide range of traffic conditions.
- The effectiveness of control plan changes is evaluated on variable-sized subnetworks and along a dinamically-calculated time horizon. The size of the subnetworks depends on the traffic conditions, and it is efficiently reduced by the prediction method. A time horizon equal to the current duration of the junction's cycle produces the best results.
- Queue volume has been chosen as the best objective function in all traffic conditions, becoming preferable as conditions become more congested.

For its part, the GETRAM environment has the following advantages when compared to other existing environments:

- It distinguishes a series of basic network objects, such as sections, nodes, and controllers that serve as building blocks from which complex urban and interurban geometries are built. Each one of these objects is extensible throught subobjects such as laterals (in a section) or stages (in a node). For comparison, most other environments distinguish numerous types of sections (with or without an entrance lateral lane, with both entrance and exit laterals, on both sides, etc.), nodes, and so on so that the user is obliged to remember all these types, and there is no easy way to switch from one type to another type of object.
- Data is stored in a common DataBase, and the various modules and traffic models access it through a library of high-level, object-based Application Programming Interface (API) functions that greatly reduce the effort of integrating a new module into the environment and, in general, accessing any data.
- It provides a common graphical editor to specify the network geometry and traffic parameters. The editor is designed to be very easy and fast to use, the resulting

specification having the detail needed by a microscopic simulator without requiring more effort than if macroscopic detail only were needed. The editor accepts a background image that serves as a reference and avoids the need for a previous digitization.

- A traffic model can start from a network state produced by the same or another model. Network states are stored with a high level of detail so that, if available, they include for each section, previous turning, vehicle modality and turning: the total number of vehicles, the number of stopped vehicles, average speed, and turning percentage. Entrance flows are specified for each section, previous turning, and vehicle modality.
- A view hierarchy can be specified so that a traffic model can focus on a subview — of another view or the whole network — and can start from a network state produced by another traffic model on a higher-level view. The GETRAM API manages all the complexities associated with view filtering — objects, turnings, entrance and exit sections, network states, control plans, etc. — so that a model's code deals with it transparently as if it were the whole network. This is made easier by using a methodology where traffic models ranging from macroscopic to mesoscopic and microscopic are applied as the area of analysis reduces and more detail is required.

Future research

Although the demand-responsive control system has been shown to be effective in microscopic simulation tests, these tests were carried out in densely signalized urban areas, where section length is not greater than 400 m. In fact, the following points would have to be revised in order to apply the system to other areas:

1) To deal with longer sections, a platoon dispersion model has to be included. Also, the method of changing the link's width — capacity — according to the queues and turning percentages in the section would have to be revised.
2) Currently PACKSIM cannot elegantly model left-turning bays: it has to decompose the section into several smaller ones. This slows the system down and can be a source of inaccuracies during the feedback.
3) The model does not apply a special treatment for trams, nor bus lines. However, support for them should be straightforward by extending the objective function.
4) The autoadaptive logic has no provision for phase-skipping, nor changes in the stage sequence. Support for these features may change the optimal time horizon, and affect the way the predictors work.
5) To extend CARS to interurban areas, support for ramp metering control has to be included.
6) Even focusing on signalized areas, support for reversible lanes is also needed as more cities introduce this feature in the vicinity of signalized junctions.

A solution avoiding all these extensions and revisions is to apply CARS, as it is, only to the sub-areas for which it was intended. Specialized control systems could apply to specific sub-areas and, if needed, a global control system could be employed to coordinate them and take measures — give priority to a direction, restrict access to an area, etc.— to prevent congestion and bottlenecks.

7. References

Ferrer, J.L. and J. Barceló (1993). *AIMSUN2: Advanced Interactive Microscopic Simulator for Urban and Non-urban Networks. System Description*, Universitat Politècnica de Catalunya, Dept. of Statistics and Operational Research, Forthcoming Research Report.

Gartner, N.H. (1982). *Demand-responsive decentralized urban traffic control: Part I: Single-intersection policies*, Office of University Research, US Department of Transportation, Rept. DOT-RSPA-DPB-50-81-24.

Morgan, J.T. and J.D.C. Little (1964). "Synchronizing traffic signals for maximal bandwidth.", *Operational Research*, Vol. 12, No. 6, pp. 896-912.

Webster, F.V. and B.M. Cobbe (1966). "Traffic signals.", *Road Research Tech. Paper*, No. 56, London: Her Majesty's Stationery Office, pp. 55-60.

APPENDIX I
CARS V1: Controller API

This appendix includes the call format and description of each function in the Application Programming Interface between CARS V1 and the real-world controllers. They were first introduced in PART II, Section 9. In order to test CARS, these functions have to be implemented as a higher level above the controllers — in the real world or in a simulator.

CONNECT CONTROLLER

Connects a controller to the CARS system, providing information on time intervals and initial state.

- **Format**

retcode = **ConnectController** (idcontroller, delta, scycle, current_state);

```
ushort idcontroller, delta;
struct controller_state *current_state;

ushort retcode;

where:

struct controller_state {
            /* 0 if phase, 1 if interphase period */
    ushort type;
    union {
                /* phase descriptor */
            struct phase_state {
                        /* identifier of the current phase */
                    ushort idphase;
                        /* number of delta intervals to finish this phase */
                    ushort n_delta;
                        /* identifier of the next phase */
                    ushort idphase_next;
            } p;
                /* interphase descriptor */
            struct interphase_state {
                        /* identifier of the origin phase */
                    ushort idphase_orig;
                        /* identifier of the destination phase */
                    ushort idphase_dest;
                        /* delta interval number from the beginning of the period */
                    ushort n_delta;
            } i;

    } u;
};
```

- **Returns**

Value	Description
0	success
1	error

- **Arguments**

  idcontroller
  The identifier of the controller.
  delta
  The duration, in seconds, of the time interval.
  scycle
  The duration, in seconds, of the system control cycle.
  current_state
  Initial state in the stage sequence of the junction. The controller is required to put the traffic lights in line with this state. A NULL is passed if unsignalized junction.

- **Description**

CONNECT CONTROLLER is used previous to executing CARS, to check that the controller exist, providing the controller with certain necessary values: the duration of delta, scycle, and the initial state in the control of the junction. The function has to return immediately, without waiting for the traffic lights to be in the proper state. The controller's clock doesn't start yet.

It is assumed that the controller already 'knows' the actions to be performed to change the traffic lights according to each phase and interphase period.

SET CONTROLLERS

Checks that all controllers have reached the required initial state.

• **Format**

retcode = **SetControllers** ();

ushort retcode;

• **Returns**

Value	Description
0	success
1	unable to reach the required state in a controller

• **Arguments**

none

• **Description**

SET CONTROLLERS checks to see if all the controllers have already reached the required initial state. Returns when this condition has been reached, or if an error has been produced in a controller. The controller's clock does not start until it receives the SYNCHRONIZE CONTROLLERS call.

SYNCHRONIZE CONTROLLERS

Resets the clock in all the controllers, thus synchronizing them.

- **Format**

retcode = **SynchronizeControllers** ();

ushort retcode;

- **Returns**

Value	Description
0	error
> 0	the number of delta intervals until the next synchronization will be required.

- **Arguments**

none

- **Description**

SYNCHRONIZE CONTROLLERS resets the clock in all the controllers. After the call, the clock in each controller is started from the value 0, and increments each delta seconds.

The function has to return immediately, without waiting. If an error is detected it will be reported in the next CHANGE JUNCTION or RESET DETECTORS call.

CHANGE JUNCTION

Indicates to the controller when the associated junction has to finish the current phase.

• **Format**

retcode = **ChangeJunction** (idcontroller, n_delta, idphase_next);

ushort idcontroller, n_delta, idphase_next;

ushort retcode;

• **Returns**

Value	Description
0	success
1	error

• **Arguments**

idcontroller
The identifier of the controller with which the junction is associated.
n_delta
The delta number on the controller's clock when the phase has to finish.
idphase_next
The identifier of the next phase towards which the junction has to move after finishing the current phase. The junction may enter an interphase period before the next phase.

• **Description**

CHANGE JUNCTION indicates to a junction when it has to finish the current phase, and enter an interphase period until the next phase. The identifier of the next phase, although not necessary, is given for checking purposes only. The function has to return immediately, without waiting.

It is assumed that the controller already 'knows' the actions to be performed to change the traffic lights according to each phase and interphase period.

CONNECT DETECTOR

Connects a detector to the CARS system, providing information about the data communication.

- **Format**

```
retcode = ConnectDetector (idcontroller, iddetector, measurements);

ushort idcontroller, iddetector;
struct measurement *measurements;

ushort retcode;

where:

struct measurement {
            /* number of vehicles, accumulated over a delta seconds period */
    float nbvehicles;
            /* vehicle speed, averaged over a delta seconds period */
    float speed;
};
```

- **Returns**

Value	Description
0	success
1	error

- **Arguments**

idcontroller
The identifier of the controller to which the detector is connected.

iddetector
The identifier of the detector in the controller. Also known as connection number.

measurements
The address of the array of scycle structures of measurement, where the controller has to deposit the scycle last measured when required by the RESET DETECTORS function.

- **Description**

CONNECT DETECTOR is used previous to executing CARS, to check that the detector exists, providing the address where the controller has to deposit, when required by the RESET DETECTORS function, the measurements received from the detector. The function has to return immediately.

RESET DETECTORS

Indicates to the controllers that the last scycle measurements from the detectors have to be sent to CARS.

• **Format**

retcode = **ResetDetectors** ();

ushort retcode;

• **Returns**

Value	Description
0	success
1	error

• **Arguments**

none

• **Description**

RESET DETECTORS communicates to all the controllers that they have to send the sets of last scycle measurements received from their connected detectors. The address of where to copy the measurements to has previously been communicated by the CONNECT DETECTOR function.

APPENDIX II
Notation used in CARS V1

This appendix describes the notation that has been used in CARS V1.

Timing notation

Δt:	time interval (s).
SCC:	system control cycle.
scycle:	number of Δt in a SCC.

PACKSIM notation

Ap:	packet acceleration, a constant (m/s^2).
Dpes:	distance from a packet to the end of the section (m).
Dpp:	distance between two packets (m).
Ds:	average distance between stopped vehicles (m).
Lb:	bridge length (m).
Ll:	link length (m).
Lp:	packet length (m).
Lvl:	average length of a vehicle in the link (m).
Lvs:	average length of a vehicle in the section (m).
Nlanes:	number of lanes in a section.
nVehDisl:	number of vehicles discharged in the gradual start of a packet in a link.
nVehp:	number of vehicles in a packet.
Pt:	turning ratio.
SP:	free space for vehicles to enter inside a section (number of vehicles).
Ti:	influence time (s).
μ:	fluidity coefficient.
Vp:	packet speed (m/s).
Vsmax:	maximum speed (speed limit) in the section (m/s).
Vtmax:	maximum turning speed (m/s).
Wl:	link width (number of lanes).
Wldes:	desired link width (number of lanes).
Xdes:	desired position of the packet in the link (m).
Xold:	old position of the packet in the link (m).

APPENDIX III
CARS V2: Controller messages

This appendix describes the format of the messaging protocol that is used in the communication between CARS V2 and AIMSUN V2. The functionality is described in PART IV. These messages have been designed with flexibility and general-purposeness in mind, so that, by using them, one could employ another control system, another microscopic simulator, or even extend them in order to communicate with real-world controllers. Moreover, as they are implemented on TCP/IP sockets, they are highly portable.

Format: General rules.

All messages start with the character '#', followed by a message identifier — two characters —, and end with the character '!'. The character '@' following the message identifier indicates that some parameters follow, and these are then included between this character and the '!', with a blank character at the end of each parameter.

idmessage[@parameter1 parameter2 ...]!

Eight messages are distinguished: welcome, send timing parameters, request control plan, send control plan, go, request detector measurements, send detector measurements, end.

- **Welcome**

  #01!

After the two sockets have been connected, AIMSUN sends a 'Welcome' message to CARS. After receiving it, CARS sends a 'Welcome' message to AIMSUN.

- **Send timing parameters**

  #02@scycle delta !

  scycle (format xx): The number of measurements that will be requested from each detector.
  delta (format xxx.xx): The interval of time along which measurements are to be accumulated, in seconds.

Used by CARS in order to communicate to all controllers certain timing parameters that are necessary for both sending the right number of measurements, and to accumulate each measure for the right number of seconds.

When testing with AIMSUN, the execution does not run in real time, so that these timing parameters are also used by AIMSUN in order to send detector measurements each scycle * delta seconds.

- **Request control plan**

  #03@idcontroller idconnection idnode !

idcontroller (format 20x): The identifier of the controller to which the junction is connected.
idconnection (format xxxxx): The connection identifier of the junction with the controller.
idnode (format xxxxx): The identifier of the junction.

Used by CARS in order to ask for the state of the control policy in a certain junction — connected to a controller.

- **Send control plan**

    #04@idcontroller idconnection idnode initialized nbstages tstage$_1$ ... tstage$_{nbstages}$

    idstage_current counter !

idcontroller (format 20x): The identifier of the controller to which the junction is connected.
idconnection (format xxxxx): The connection identifier of the junction with the controller.
idnode (format xxxxx): The identifier of the junction.
initialized (format x): Indicates whether the junction's control timings have been initialized.
nbstages (format xx): The number of stages in this junction.
tstage (format xxx.xx): The duration, in seconds, of a stage.
idstage_current (format xx): The identifier, from 1 to nbstages, of the current stage.
counter (format xxx.xx): The number of seconds that the junction has already been in the current stage, from 0 to tstage.

It can be used in two ways:
- Initially, by AIMSUN in order to send to CARS the initial state of the control policy in a given junction connected to a controller.
- By CARS in order to set, in AIMSUN, a junction to a given state in control policy. It is called at the beginning of execution for the uninitialized junctions and thereafter to adapt the control policy in a demand-responsive junction.

- **Go**

    #05!

Communicates, to all controllers, that they can resume processing: counting detector measurements and grouping them in scycle measurements of delta seconds each one.

- **Request detector measurements**

  #06@idcontroller idconnection iddetector !

  idcontroller (format 20x): The identifier of the controller to which the detector is connected.
  idconnection (format xxxxx): The connection identifier of the detector with the controller.
  iddetector (format 20x): The identifier of the detector.

Used by CARS in order to require a detector to send its last scycle measurements, each one of them having been accumulated for delta seconds.

- **Send detector measures**

  #07@idcontroller idconnection iddetector nbmeasurements
  (nbvehicles, speed, occupancy)$_1$ ... (nbvehicles, speed, occupancy)$_{nbstages}$!

  idcontroller (format 20x): The identifier of the controller to which the detector is connected.
  idconnection (format xxxxx): The connection identifier of the detector with the controller.
  iddetector (format 20x): The identifier of the detector.
  nbmeasurements (format xx): The number of measurements that follow. It must be equal to the scycle value.
  nbvehicles, speed, occupancy (all of them, format xxxxxx.xx): A detector measurement containing the accumulated number of vehicles, averaged speed (in km/h, weighted by number of vehicles) and occupancy for delta seconds.

Used by AIMSUN in order to send to CARS the last scycle measurements — each one having been accumulated for delta seconds — made by a detector. The measurements are sent so that the first one corresponds to the older measurement, and the last one to the newer. In each measurement, the number of vehicles is accumulated, the speed is averaged according to the number of vehicles in each count, and the occupancy takes values from 0. to 1.

- **End**

  #08!

Communicates that CARS or AIMSUN will stop.