SOFTWARE MODEL OF HIERARCHICAL TRANSPORT CONTROL

Jan Přikryl

Department of Adaptive Systems
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
e-mail: prikryl@utia.cas.cz

Abstract: Real-time adaptive control of signalled intersections is a challenging task. In our paper we will present a software framework built in MATLAB programming environment that makes it possible to conduct clean control experiments with a selected urban traffic network model. Our software image consist of four self-contained modules which make it possible to have several implementations and different urban traffic network descriptions and setups — users specify the particular structure, model, controller and parameters when starting the simulation. This is possible thanks to simple API that offers the possibility to easily extend the software with new model and controller implementations.

Keywords: urban traffic control, traffic model, MATLAB

1. MOTIVATION

Urban surface transport networks are characterised by high density of streets and a large amount of junctions. In many cities these structures cannot easily accommodate the vast volume of traffic, which results in regular traffic congestions. The most common approaches to solution of these transport network deficiencies are attempts to rebuild parts of the network in order to increase their capacity, reroute transit traffic by building large bypasses around affected locations or making drivers pay for entering central zones.

In many cities it is impossible or ineffective to reconstruct the existing street network due to their historical development. Hence, efficient traffic control mechanisms are sought that would provide for higher throughput of the urban transport network without changing its topology. Many commercial systems exist that attempt to solve this task. However, these systems are costly black-boxes that cannot be easily tuned to specific on-site transport problems. We need efficient adaptive traffic control mechanisms that would be transparent and the would allow for easy parameter specification. One way how to build these systems is to use deterministic physically-based state models and tune them using feedback from different traffic detectors (Kratochvílová and Nagy, 2003).

1.1 Software Framework

Our institute hosts two R&D projects targeted at design of such efficient adaptive urban traffic control mechanisms. We make use of large MATLAB-based code libraries (e.g. *Mixtools*) and MATLAB is the programming language of choice for virtually all projects carried out at our

department. MATLAB offers a very good envornment for "rapid prototyping" as it provides a stable environment for mathematical modelling together with a number of libraries (toolboxes) implementing more evolved tasks. It also provides users with easy methods for analysis of their results.

This all means we have a lot of simple "proof of concept" works in the area of urban traffic modelling. These code snippets work quite well, but at the same time are not flexible enough mainly in the area of testing and validation: All the parameters and underlying network structure are fixed and the process of testing the implementation and debugging it is slow and painful. In order to spend less time rewriting code we have been rewriting over oand over the whole last year, and to concentrate more on real problems of traffic modelling and traffic control, we need a *stable software framework for design and testing*.

1.2 Software Framework Requirements

The first step in the design of framework was the process of determining requirements for the final product. As usual, there are two sets of mutually very often exclusive specifications from eng users and programmers.

From the user point of view, we would like to use MATLAB as we have a large code basis and experience with building MATALB-based applications. Our users would like to avoid object-oriented languages until the last possible moment, as they are not programmers and concentrating on aspects of another programming paradigm slows down the R&D process. Finally, from end-user point of view it is necessary to have a clear programming interface allowing for model and controller specification and modification, together with an easy method for specification of an urban network and its parameters.

These last two points are also desirable form the point of view of a software engineer. However, programmers would like to avoid MATLAB as it does not provide a decent programming interface (no strict type definition and control, quite awkward debugging possibilities) They would prefer using object-oriented approach to hide the implementation and internal data of the constructed objects, which is also impossible in Matlab. There is also strong inclination towards standard markup languages, resulting in proposition of using a standard description language (XML, SGML) for network and parameter specification.

Both strategies have been discussed between the member of our group, and we also took use of knowledge of our colleagues who have a long tradition in maintaining complex MATLAB projects. The final agreed set of requirements is as follows:

- We will use MATLAB as the basis of the development for two reasons: (a) it is too troublesome for the team to switch to another programming language at this moment, and (b) we can benefit from the large existing code-base. Also some of our project partners develop parts of their code in MATLAB.
- The stable part of the code will be gradually rewritten in C so that it can be deployed on a wide range of computer systems.
- Even if MATLAB offers basic object-oriented functionality (and a better OOP environment is currently in testing), we will mimic the object-oriented approach by using *function handles* and rely on our users to not touch things they are not allowed to touch.
- For the description of the network we will use structured MATLAB batch files that can

be easily transformed into XML or SGML in the future.

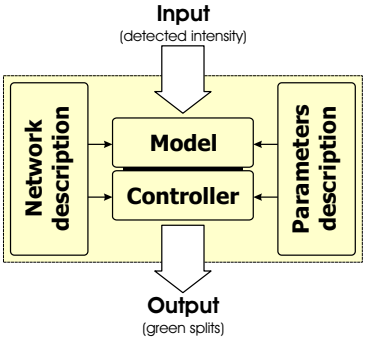

Fig. 1: Building block of the software model.

2. SOFTWARE MODEL

We have identified four basic building blocks of the traffic network controller during the analyis. The software framework follows basic outline of an urban traffic network model described in (Kratochvílová and Nagy, 2004). It is being developed as a composition of the following elements:

Network description block. This block is responsible for reading in all junction and microregion parameters and their mutual interconnection into the hierarchical network of microregions. Junction parameters, as number of arms and lanes, turning rates, and initial signal plans are read in by functions provided by this block.

Parameters description block. This element provides interface for reading the input data specification, and model and controller parameters that influence the computation. In future, this block would be also responsible for specification of parameters for different regression tests.

Model block. This is an implementation of an arbitrary urban traffic network model. In the current version, only the non-linear physically-based state model (Kratochvílová and Nagy, 2004) is used.

Controller block. This part of the framework is tightly coupled with the model block and provides adaptive control of the traffic network based on information provided by *Model block*.

The above mentioned building blocks are designed to be (almost) self-contained (they share only a small number of utility functions). An *application programmer interface* (API) is defined for each of the modules. Hence, the modules can be used also separately in another project, if required. The overall structure of the framework is shown in Fig. 1. We will now briefly review all the modules.

2.1 Network description

The network description module reads in all the data describing the modelled urban traffic network (de Dios Ortúzar and Willumsen, 2001). It reads descriptions of junctions, their arms, and lanes (their physical structure and parameters, placement of input and output traffic detectors),

turning rates from input lanes into output arms (they describe probability a car passing through a junction from given input lane to given output arm), one or more signal plans containing green splits for particular phases and corresponding saturation flows.

We start with a physical structure of a microregion as shown in Fig. 2. Every junction in this microregion is described using a structured MATLAB file. Figure 3 shows an massively simplified example of such a junction description file.

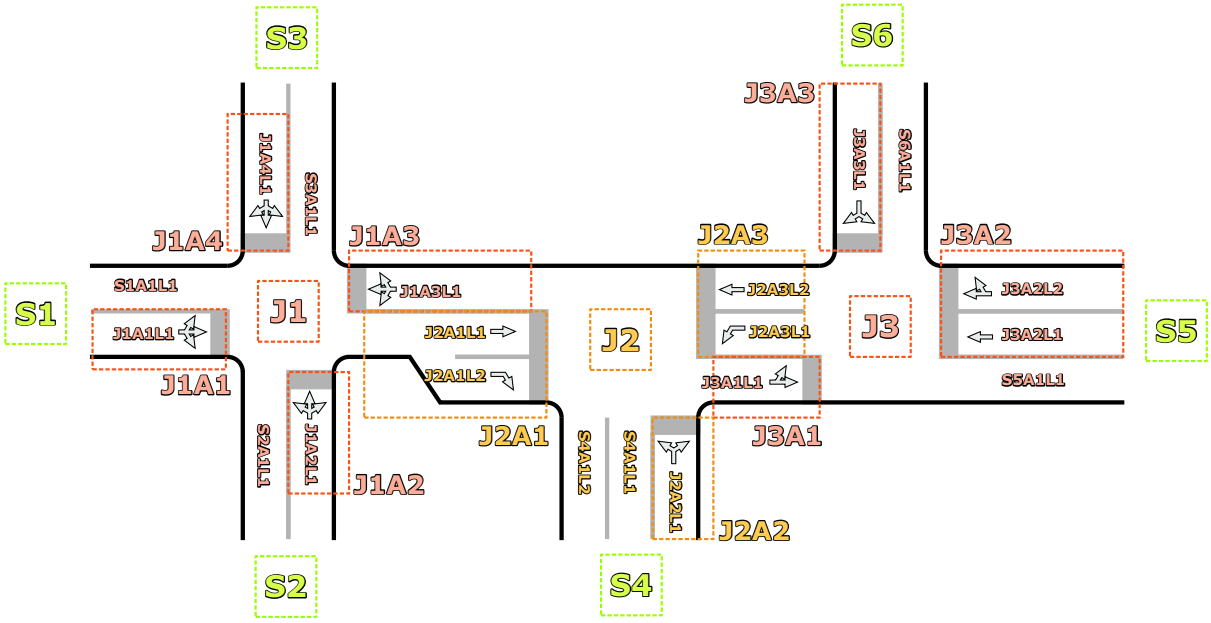

Fig. 2: An example of a structure of an isolated part of an urban traffic network and description of its elements.

This module then builds microregion structure using a user-specified graph of junction interconnection. This is outlined in Fig. 4. The final step is connecting microregions into a complete urban traffic network structure, again using a user-specified graph, as shown in 5.

2.2 Parameter description

Parameter description module allows for specification of different variable conditions that may influence the model and controller. This way we can have multiple sets of model and controller parameters for every model and controller and easily use them for testing.

This block contains mainly specification of traffic conditions via input data parameters (input and output detector intensities and occupancies). It also specifies internal model-dependent parameters (initial covariances for Kalman filter, for example) and internal controller parameters (green split windows, weights for different junctions and arms and so on.

2.3 Model implementation

The model works on global data provided by previous two blocks, which describe physical network structure and its parameters. The framework defines model API and the structure of global data (currenly network structure and parameters are stored as global cell vectors of structures).

```
%% Intersection of Zborovská - V Botanice
BeginJunction('J01');
  %% Zborovská input
  BeginArm('A1');
    CreateLane('L1','FullLane',196,35);
  EndArm();
  ...
  %% Fixed turning rates
  BeginTurningRates ();
    % From Zborovská straight ahead
    TurningRate('A1','A3',0.79);
    ...
  EndTurningRates ();
  %% Fixed signal plan
  BeginSignalPlan();
    BeginPhase('P1');
      GreenTime(20);
      ClearingTime(4);
      % Saturation flow 3800 vehcls/hr.
      GreenForLane('A1','L1',3800.0);
      ...
    EndPhase();
  EndSignalPlan();
EndJunction();
```

Fig. 3: An example junction description file.

The model, when invoked through an API call, is itself responsible for transforming the global data into its own internal representation that is suitable for model computation (this step may include, for example, creation of state-model matrices). The output of the model is a set of quantities used by controller to change inputs of the model (green splits) to maximise throughput of the controlled network. As an example, the model may predict queue lengths using measured junction input and output intensities for given signal settings. The controller then tries to change the signal settings in such a way that the queue lengths in the system are minimised.

2.4 Controller implementation

The controller module works again on global data provided by the parameter description and network description blocks. It makes also use of data provided by model block (these may contain, for example, model parameters and predicted model outputs).

The controller aim is to optimise the input of the model (green splits in signal plan) targeting at maximising throughput of the network. In the current implementation, this criterion is equivalent to minimising queue lengths in the system (Papageorgiou, 1983).

3. SUMMARY

This paper described a software framework implemented in MATLAB for implementation and experimental evaluation of urban traffic control strategies. The presented software model of

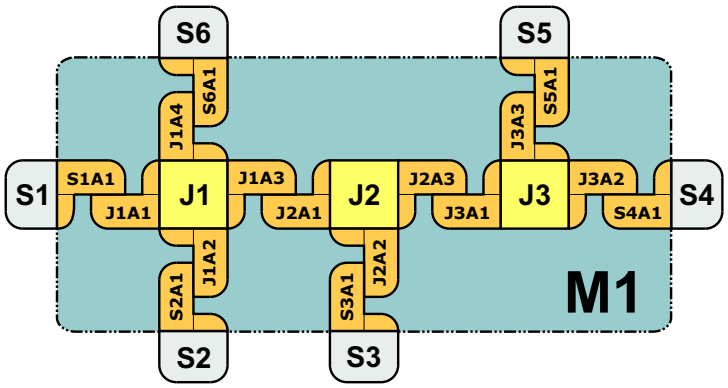

Fig. 4: Microregion from Fig. 2 in a schematic representation as a network of junctions.

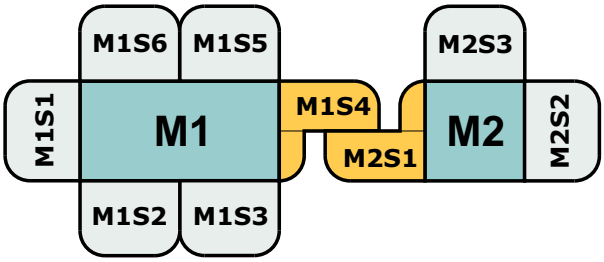

Fig. 5: Second level of hierarchy in a small urban traffic network. Microregion M1 in this scheme is the one depicted in detail in Fig. 2 and 4.

an traffic controller makes if possible to conduct clean control experiments with some urban traffic network model. The framework provides pseudo-class mechanism to group logical parts of code together. This allows for easy replacement of model and controller implementation due to pre-defined API. Also, some degree of separation of model data and parameters from the implementation has been achieved.

Acknowledgements

This work is funded by MŠMT project 1M0572 and by the Czech Ministry of Transportation under project 1F43A/003/120. Support of GA ČR grant no. 102/03/0049 and AV ČR grant no. S1075351 is also acknowledged.

REFERENCES

de Dios Ortúzar, Juan and Luis G. Willumsen (2001). *Modelling Transport*. 3 ed.. John Wiley & Sons.

Kratochvílová, J. and I. Nagy (2003). Bibliographic Search for Optimization Methods of Signal Traffic Control. Technical Report 2081. ÚTIA AV ČR. Praha.

Kratochvílová, J. and I. Nagy (2004). Traffic control of microregion.. In: *CMP'04: MULTIPLE PARTICIPANT DECISION MAKING, Theory, algorithms, software and applications* (J. Andrýsek, M. Kárný and J.. Kracík, Eds.). Advanced Knowledge International. Adelaide. pp. 161 – 171.

Papageorgiou, M. (1983). *Applications of Automatic Control Concepts to Traffic Flow Modeling and Control*. Vol. 50 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag. Berlin.