# An Organic Architecture for Traffic Light Controllers

Fabian Rochner[1], Holger Prothmann[2],
Jürgen Branke[2], Christian Müller-Schloer[1], Hartmut Schmeck[2]

[1]Institute of Systems Engineering
Universität Hannover
Appelstr. 4
30167 Hannover, Germany
{rochner,cms}@sra.uni-hannover.de

[2]Institute of Applied Informatics
and Formal Description Methods
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
{hpr,jbr,hsch}@aifb.uka.de

**Abstract:** Efficient control of traffic networks is a complex but important task. A successful network management vitally depends on the abilities of the traffic light controllers to adapt to changing traffic situations. In this paper a control architecture for traffic nodes is presented that is inspired by the principles of Organic Computing. It allows a node to quickly adapt to changing traffic situations and enables it to autonomously learn new control strategies if necessary.

## 1 Introduction

Control of road traffic networks in urban areas is a challenging task. This is mainly due to the great dynamics of traffic. Therefore, a mere generation of an optimal controller for a certain traffic situation is not sufficient. It is necessary to provide the capability of adjusting quickly to changes in traffic situations and, in particular, to react reasonably in situations that had not been anticipated by the designer of the traffic controller.

In this paper we show how different principles of Organic Computing [Sch05] like self-organization, self-adaptation and the Observer-Controller paradigm can contribute to improve traffic controllers, making them at the same time more flexible and easier to set up, to maintain, and give better results. First ideas that have been incorporated into this architecture have been presented in [RMS04].

From the extensive literature on traffic control, the recent works of Bull et al. [BS+04] and Helbing et al. [HL+05] should be mentioned in the context of this paper. Similar to the ideas presented here, Bull uses a Learning Classifier System (LCS) to control a simplistic traffic node, while Helbing proposes a decentralized control strategy for traffic flows (which not yet considers legal regulations like minimum or maximum green times).

## 2 Motivation

A central requirement for any control architecture for a traffic node is to guarantee the safe functionality of the node at any time, ensuring that conflicting traffic streams are

never allowed to enter the node area simultaneously. Furthermore, it should optimize the performance of the node (e. g. in terms of low waiting times for all road users). To achieve this task, a quick adaption to changing traffic demands is mandatory.

Traffic changes can be observed on different time-scales. Besides small short-time variations around a constant mean arrival rate, a much greater variability of traffic demands occurs on an intra-day basis. A typical workday can be divided into several periods of differing traffic situations including two peak periods with high demands due to commuter traffic. Within these periods the highly demanded traffic streams usually differ, which is illustrated in Figure 1 for the peak periods of a traffic node in the city of Hamburg. The observable traffic patterns, the time of their occurrence, and their durations vary among the network nodes. Furthermore, in the long run traffic patterns are subject to slowly developing changes that can lead to the aging of fixed-time signal programs [BB86].

Different approaches can be used to handle traffic changes. To adapt the phase durations to short-term variations, traffic light controllers are used which utilize traffic information provided by detectors. Due to their limited reactive freedom, these controllers are often not sufficient to handle intra-day changes. In this case, intra-day changes must be handled by a human engineer who designs appropriate signal programs for the different demands and defines conditions for switching between them, but this is a complex and time consuming task that has to be repeated for every single node. Even if the developed signal programs and switching schemes (which often simply depend on the time of day) perform well under normal conditions, the system gets into trouble whenever situations occur that have not been anticipated by the engineer. Such unforeseen situations can be road or lane closures due to road works or incidents in the vicinity of the node that influence the traffic patterns in their surroundings.

## 3   Multi-layer Design

To reduce the necessary effort for a human engineer and to avoid the drawbacks of a completely preplanned solution, we propose a multi-layered organic architecture that can be applied to arbitrary traffic nodes which are equipped with traffic detectors. A first organic layer detects changes in the traffic patterns on-line and switches between appropriate signal programs. Whenever a previously unknown situation occurs, a second organic layer autonomously searches for an appropriate signal program by simulation-based off-line optimization. An architectural overview is shown in Figure 2. In this section we outline the tasks of the different layers which are described in more detail in Section 4, 5 and 6.

**Layer 0: Simple Traffic Light Controller**

On the lowest layer a parameterized traffic light controller (TLC) is used to control the traffic signals. The TLC can implement a fixed time scheme that simply switches phases after predefined amounts of time, or it can be traffic-responsive and therefore vary the phase durations based on detector information. In the first case the phase sequence and
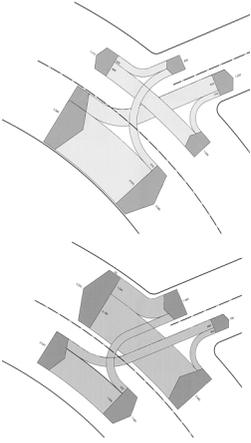
Figure 1: Intra-day peaks (morning, afternoon) shown for one node. Arrow width is proportional to traffic flow. Data from traffic census.
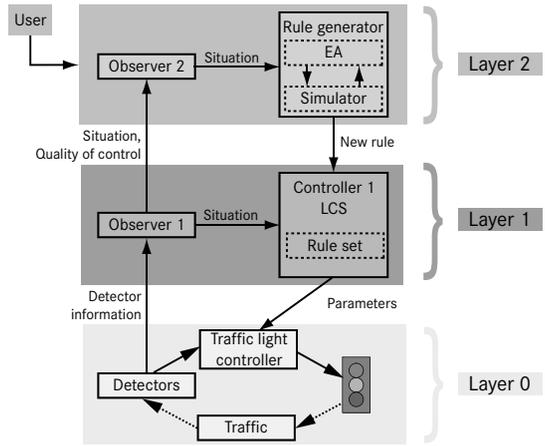
Figure 2: Architecture overview. Layer 0 represents the traffic node, Layers 1 and 2 are organic control layers responsible for the selection and generation of signal programs.

durations are specified as TLC parameters, in the second case parameters specify different aspects of the variation process (e. g. the minimum or maximum duration of a phase). A traffic-responsive TLC can autonomously handle short-term traffic variations, and it can be adapted to different intra-day demands by parameterization. Details on the TLC and its parameters can be found in Section 4.

For the traffic-dependent selection and simulation-based optimization of appropriate TLC parameters, we propose two organic control layers. The TLC on Layer 0 is functional without the organic layers, but it exhibits no learning abilities and it has (if at all) only limited adaptation capabilities.

## Layer 1: On-line Selection of TLC Parameters

The first organic layer (i. e. Layer 1) monitors the traffic situation at the node and switches control parameters of the TLC when necessary. For this purpose an observer component aggregates traffic data collected by the detectors on Layer 0 and determines a flow value for each traffic stream crossing the node. This detected situation serves as input for the controller on Layer 1 which selects appropriate parameters for the TLC. The controller is implemented as a Learning Classifier System (LCS) with reduced functionality, where the rules called classifiers map traffic situations to TLC parameters. The selection of a classifier is based on its value which is adjusted according to the corresponding TLC's performance. The details of the selection and valuation process can be found in Section 5. Since neither observer nor controller on Layer 1 require high computing power, the layer can be implemented decentrally on each node using embedded hardware.

**Layer 2: Off-line Optimization of TLC Parameters**

Whenever a situation is not adequately covered by the classifier population on Layer 1, an optimized classifier covering it is generated by model-based optimization on the second organic layer (i.e. Layer 2). An evolutionary algorithm (EA) generates populations of TLC parameters and evaluates them using a microscopic simulation model of the observed situation. At the end of this process a new classifier is created which maps the observed situation to the best TLC parameters resulting from the optimization. Before the new classifier is inserted into the classifier population on Layer 1, its value is initialized based on the simulated results. The details of the optimization are described in Section 6.

Within the architecture, Layer 2 replaces the genetic operators that would normally be applied by a standard LCS. This replacement is necessary because newly generated classifiers must not be evaluated by applying them in a real traffic network. By using a simulated environment for evaluation, bad TLC parameters can be identified without negative consequences for the traffic system. Furthermore, the employment of a simulation software allows the fast evaluation of many TLC parameters, which should result in an improved learning speed of the overall system.

A drawback of using a simulated environment for classifier evaluation might be potential differences between simulation and reality. Since microscopic simulators are quite accurate after their calibration [XA+05], we expect no major problems here. Residual errors in the valuation of a new classifier can be corrected when the classifier is applied in reality.

The application of Layer 2 requires considerably more computing power that may not be available directly at every traffic node. But since Layer 2 will not be needed continuously at every node, we propose to realize it in a semi-central fashion using several powerful PCs which each perform the generation of new classifiers for several connected nodes.

In the following sections the different layers are examined in some more detail.

# 4   Simple Traffic Light Controller

One of the basic principles of Organic Computing and the Observer/ Controller paradigm in particular is that any system designed with these ideas in mind has to stay operational even if the "organic" parts of the architecture fail. On the lowest level of our architecture a rather simple TLC is placed. This component can have a variable degree of complexity. In the simplest case it behaves just like a conventional fixed time controller: Each phase is switched to green for a fixed amount of time, independently of how many cars are waiting. It is similar to an FSM with time as the only condition for transition of states.

Now, as detectors are available in the traffic network giving information about e.g. how many cars per unit time are passing across or how many cars are waiting for a traffic light to switch to green, these data can be used to improve the behaviour of the TLC. The National Electrical Manufacturers Association (NEMA) has defined a standard for traffic light controllers that can adapt to traffic conditions [NEM03]. The advantage of this standard is that the complexity of corresponding controllers can be increased quite

smoothly. We use TLCs of different functionality based loosely on the NEMA standard. Starting from a fixed time controller, the first step is to use presence detectors: a phase no car is waiting for is skipped. Next, detectors for queue lengths are used to adjust the duration of a phase to allow for all waiting cars to pass the junction. Furthermore, the gaps between cars approaching a currently green traffic light can be measured and, if the gaps get too large, the current phase is terminated.

A TLC gets a set of parameters when initialized and then runs without further interference of the higher layers. The parameters include phases that are available (corresponding to states of an FSM) and the conditions when to switch to which other phase. As phases are predefined and can thus be assumed to be valid and phase transitions always occur according to regulations, there is no way for the system to perform "illegal" actions.

These parameters are only changed if the traffic situation has changed sufficiently. Frequent changes lead to discontinuities in the operation likely to reduce performance. The decision when a change of parameters is appropriate is made by Layer 1. The degree of change in traffic conditions needed for the overall performance to rise despite the inevitable loss during transition depends on the complexity of the TLC: a fixed time controller will have to be adjusted more frequently than a controller that utilizes detector data.

To get optimal performance, a trade-off is necessary between simplicity of the TLC that leads to easy generation of new parameter sets on Layer 2 due to low dimensionality of the search space and flexibility to react on small to medium changes in the traffic situation reducing the disturbances caused by changing the TLC.

## 5   Selecting a Suitable TLC

As outlined in the previous section, the TLC at the lowest layer of the architecture reacts instantly to rather small changes in the environment. Greater deviations from a certain traffic situation a particular TLC (represented by its parameter set) has been optimized for are handled by replacing it with another more suitable one. This is done at Layer 1. Here, the goal is to classify encountered traffic situations into groups and find the most suitable TLC for each group. This is what Learning Classifier Systems (LCS) are supposed to do: classify input, find appropriate action.

### Learning Classifier Systems

LCS in their present form have been proposed by Holland and Wilson [HB+00]. There is no such thing as *the* LCS, many variants have been described. However, the most promising approach appears to be what is called XCS, eXtended Classifier System [Wil95]. XCS strives to find optimal rules for each encountered situation, not just rules that fit situations with high payoffs as is the case with simpler LCS. These rules, called classifiers, connect an input pattern (condition) to an action. The encoding of inputs is done such that different levels of "generality" are possible, hence the range of input values each classifier can match against may vary from a single point to the entire search space. New rules are gen-

erated using genetic operators like crossover and mutation on existing ones, changing both condition and action. Furthermore, every time no classifier matching the current input is available, one or more classifiers with a matching condition and random action are created ("covering").

At initialization an LCS generates classifiers randomly by covering, but later on classifiers that have proven reliable influence the process of generating new rules. The environment gives feedback (also called reward) whenever a classifier's action is applied and the LCS uses this reward to valuate classifiers. The value of a classifier consists of several components, of which the prediction of expected reward and the prediction error are most important. An "optimal" classifier combines high predicted reward with low prediction error. The goal of an XCS is to represent the entire search space with as few classifiers as possible while keeping only those that are accurate in their prediction.

**Reduced Functionality**

The most important task for the LCS within the architecture is to improve the valuation of existing classifiers. This is done using data like traffic flows, average waiting times and queue lengths. These measurements, available either directly or derived from detector data, are combined into a single value by means of an objective function which has to be specified by a human expert. Valuation is done on Layer 1 and Layer 2 using the same objective function, so newly generated classifiers will already be provided with a reliable value derived from simulation. Still, deviations between reality and simulation are inevitable, thus valuation on Layer 1 is necessary. Furthermore, this gives information on the quality of the simulation model.

Apart from this, the functionality of our LCS is reduced. The most significant change is that in our setting the LCS does not generate new actions—it uses only those that have been tested before using Layer 2. So all it can do is find the best action among those available, it does not explore. This is simple if classifiers exist that match the current input: The classifier whose prediction is best is chosen. If no classifier matches, a new matching one has to be generated. The proposed approach is to choose the classifier whose condition is closest to the current input, copy it, widen its condition just enough to match and discount its value slightly. This way a quick response is possible (waiting for Layer 2 to provide a new solution would take too long) and the probability that the chosen action will perform acceptably is greater than if the action was chosen randomly. If this widening exceeds a certain threshold, the current situation is passed simultaneously to Layer 2 so that, when it is encountered again, a specifically optimized classifier is available.

This requires an encoding for the condition that allows for easy adjustment to include some given input. As the input consists of real values the best choice appears to be unordered bound representation [SB03]: each predicate $x_i$ of the condition is represented as lower ($p_i$) and upper ($q_i$) bound of a half-open interval. Order of upper and lower bound does not matter, so there are two representations for each interval $[p_i, q_i)$ if $p_i \neq q_i$ and one if $p_i = q_i$ (in this case the interval is considered closed). This introduces a minimal bias against exact numbers, but this is negligible compared to the bias of e. g. center-spread representation [Wil00] towards the edges of the input space [SB03].

Figure 3 shows a simple example: The input consists of two variables ($x_1$ and $x_2$). The conditions of two classifiers $A$ and $B$ are mapped into this input space (with one of their intervals $[p_i, q_i)$ each). The shown input is not matched by these classifiers, thus covering is necessary. The parameters to be considered when deciding which classifier to start from are: distance to edge ($a$) and center of currently covered region ($b$), number of predicates/ intervals that have to be changed, size of the additional area covered and fitness of the classifiers involved.
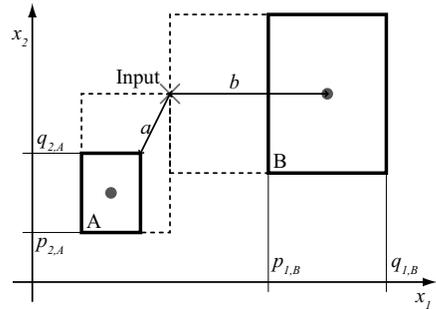


Figure 3: Encoding of conditions and covering.

Investigations on the influence of these parameters are currently under way.

The Classifier System used here has been modified in such a way that it acts deterministically. All exploratory behavior is eliminated. This is due to the application where it is not affordable to just try things out. Therefore, the exploratory parts have been moved to the simulated environment on Layer 2. Still, as the value of the classifiers is adjusted based on their performance in a real environment, reality-based learning takes place on this layer.

# 6 Generating New TLCs

The second organic layer (i. e. Layer 2) is responsible for generating new classifiers. An EA creates TLC parameters and evaluates their applicability for a specified traffic situation using the microscopic traffic simulator AIMSUN [TSS05]. AIMSUN applies the parameters to control a simulated model of the node and provides the data necessary for evaluating the TLC. A single evaluation typically takes a few seconds on a standard PC. The amount of time needed by the EA to optimize TLC parameters ranges from several minutes to a few hours depending on the node architecture, the considered traffic situation, and on the number of TLC parameters.

In contrast to Layer 1, the evolutionary process on Layer 2 is based solely on an environmental model. It uses feedback from this model to generate—within reasonable time—solutions that are "good enough" for being inserted into the classifier population on Layer 1. Since there is no feedback from reality, learning on Layer 2 is purely model-based while Layer 1 uses real detector data to readjust pre-calculated classifier values.

# 7 Conclusion

We presented a new approach to the generation of traffic light controllers that is inspired by Organic Computing. In a multi-layered architecture the tasks of applying, selecting, and generating control rules are assigned to three different layers. The resulting system is capable of adjusting to changes in the traffic situations in a self-organized way, therefore

requiring only limited expert knowledge with respect to traffic control for operation. An LCS is used for the selection process, but—different from standard LCS systems—the generation of new classifiers is moved to a separate evolutionary algorithm that evaluates actions based on simulated results. Otherwise, the system would not be able to guarantee a minimum level of quality for the rules that are employed for controlling real traffic. This modification of LCS should be a promising approach also in other application areas.

# References

[BB86]   M. C. Bell and R. D. Bretherton. Ageing of fixed-time traffic signal plans. In *Proc. of the 2nd IEE Conf. on Road Traffic Control*, 1986.

[BS+04]   L. Bull, J. Sha'Aban, et al. Towards Distributed Adaptive Control for Road Traffic Junction Signals Using Learning Classifier Systems. In L. Bull, editor, *Applications of Learning Classifier Systems*, pages 276–299. Springer, 2004.

[HB+00]   J. H. Holland, L. B. Booker, et al. What is a Learning Classifier System? In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*, pages 3–32. Springer, 2000.

[HL+05]   D. Helbing, S. Lämmer, et al. Self-organized control of irregular or perturbed network traffic. In C. Deissenberg and R. F. Hartl, editors, *Optimal Control and Dynamic Games*, pages 239–274. Springer, 2005.

[NEM03]   Traffic Controller Assemblies with NTCIP Requirements. Technical report, National Electrical Manufacturers Association, Rosslyn, Virginia, USA, 2003.

[RMS04]   F. Rochner and C. Müller-Schloer. Adaptive Decentralized and Collaborative Control of Traffic Lights. In P. Dadam and M. Reichert, editors, *GI Jahrestagung (2)*, volume 51 of *LNI*, pages 595–599. GI, 2004.

[SB03]   C. Stone and L. Bull. For Real! XCS with Continuous-Valued Inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

[Sch05]   H. Schmeck. Organic Computing – A New Vision for Distributed Embedded Systems. In *Proc. of the 8th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing*, pages 201–203, 2005.

[TSS05]   TSS-Transport Simulation Systems. AIMSUN NG Users Manual. Spain, 2005.

[Wil95]   S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[Wil00]   S. W. Wilson. Get Real! XCS with Continuous-Valued Inputs. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems – From Foundations to Applications*, volume 1813 of *LNAI*, pages 209–219. Springer, 2000.

[XA+05]   H. Xiao, R. Ambadipudi, et al. Methodology for Selecting Microscopic Simulators: Comparative Evaluation of AIMSUN and VISSIM. Technical Report CTS 05-05, Department of Civil Engineering, Univ. of Minnesota, 2005.