2015 IEEE 18th International Conference on Intelligent Transportation Systems

On Distributed Traffic Signal Control

Jan Přikryl
Department of Applied Mathematics
Faculty of Transportation Sciences
Czech Technical University in Prague
Prague, Czech Republic
Email: prikryl@fd.cvut.cz

Jakub Novotný
Faculty of Nuclear Sciences
and Physical Engineering
Czech Technical University in Prague
Prague, Czech Republic

Václav Šmídl
Department of Adaptive Systems
ÚTIA AVČR, v.v.i
Prague, Czech Republic
Email: smidl@utia.cas.cz

***Abstract*—Distributed control mechanisms have been studied in past decades in different application areas. Currently, multi-agent systems are a popular topic also in the area of intelligent transportation systems, where numerous approaches to distribution of system intelligence are being tested. One of the main problems in distributed control, however, remains the guarantee of reaching the global optimum – while most of the applications perform sufficiently well, there is no way to tell that they cannot perform even better, or that they may fail under certain operating conditions. Luckily, for certain control paradigms such guarantees may be given. In this paper we study two approaches to distributed control, namely distributed linear-quadratic control and distributed non-linear control using COBYLA algorithm, and apply them to urban traffic control scenario using two traffic models of different complexity. We show that the convergence conditions are met and that the results achieved with distributed control converge to those of centralised control mechanisms.**

I. Introduction

Large systems, as are for example urban traffic networks or energy transmission networks, can be subdivided into smaller sub-systems that interact with each other. These smaller units can be controlled in decentralised manner in the hope that breaking the large system into smaller units will decrease the dimensionality of the original system and will result in better scalability of the control.

Large systems can be found in different areas, as, for example, chemical plant control, power grid control, or urban traffic control. These systems are typically composed of many smaller and similar parts, and their centralised control is difficult due to the number of components and limited communication infrastructure. In some case it is even impossible to obtain all measurements in real time and decentralised control is therefore an imperative. These are the main reasons why in the past more than four decades, decentralised control has been intensively studied and different forms of decentralised systems have been proposed [1].

This paper is further organised as follows: Section II gives an introduction to distributed model-predictive control and outlines control approaches employed in this paper. Two urban traffic models that will serve as a testbed for our experiment are introduced in Section III. In Section V we present and evaluate the results of our experiments. Section VI summarises and concludes the paper.

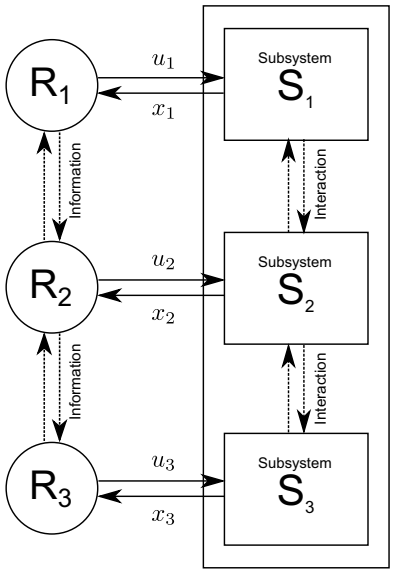

Fig. 1. Distributed control: Regulators R_i control their corresponding subsystems S_i with inputs u_i and measured quantities x_i. Regulators communicate between neighbors [1].

II. Distributed Model-Predictive Control

In control theory, *Model-Predictive Control* (MPC) denotes a control paradigm, where at certain time-step k a mathematical model of a real-world process is used to predict the future behaviour of that process at time-steps $k+1, k+2, \ldots, k+h$, given a candidate vector $\mathbf{u}[k]$ of its control inputs [2]. The control inputs $\mathbf{u}[k]$ are chosen in a way that minimises some pre-defined objective function $J(\mathbf{s}[k])$, where $\mathbf{s}[k]$ denotes a vector of system state variables at time-step k. As the minimisation of $J(\mathbf{s}[k])$ over just a single time-step (i.e. optimising control just for $k+1$) could lead to sub-optimal results in the following steps, the minimisation is typically performed over a longer horizon of h steps using an extended state vector $\mathbf{s}[h|k]$.

The distributed management model assumes communication between subsystem controllers (in this paradigm they are usually called *agents*). The local information available to these controllers includes the subsystem state variables and information on the behavior of neighbouring subsystems. The transmitted information typically contains the subsystem status or control variables and their predictions [3]. This way an agent is able to predict the effects of interaction. Diagram of

978-1-4673-6596-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ITSC.2015.150

a distributed system is shown in Fig. 1.

The principle of predictive control is to minimize the system state criterion at time-step k on time horizon of length h using $J(\mathbf{s}[h|k])$. Vector $\mathbf{s}[h|k]$ is in this case composed of all the variables of the large system $\mathcal{S}$. The principle of decentralization is to divide $\mathcal{S}$ into disjoint sub-systems $\mathcal{S}_i$, $\mathcal{S} = \cup\mathcal{S}_i$, $\cap\mathcal{S}_i = \emptyset$.

In the case of decentralized model-predictive control, each controller R_i controls only its subsystem based on the minimization criteria $J_i(\mathbf{s}_i[h|k])$. This task can be summarized as a series of optimizations in the form

$$\min J_i(\mathbf{s}_i[h|k]) \tag{1}$$

where vector $\mathbf{s}_i[h|k]$ consists only of state variables that belong to the subsystem $\mathcal{S}_i$. A "naïve" controller would then attempt to achieve the best local state, which may of course considerably differ from the global optimum.

This is the reason why in the case of distributed control another approach has to be taken. For large systems, each subsystem usually does not interact with all other subsystems, but only with those in its immediate surroundings — otherwise the decentralised control would not be feasible at all. In this situation, every agent R_i treats the system as it would consist of three disjoint subsets $\mathcal{S} = \mathcal{S}_i \cup \mathcal{S}_i^{\text{nei}} \cup \mathcal{S}_i^{\text{rem}}$ [4]. The set $\mathcal{S}_i$ contains local variables measured and controlled by agent R_i. The $\mathcal{S}_i^{\text{nei}}$ are variables of neighbouring subsystems directly interacting with $\mathcal{S}_i$ and the set $\mathcal{S}_i^{\text{rem}}$ contains the rest. For agent R_1 in Fig. 1 therefore $\mathcal{S}_1^{\text{nei}} = \mathcal{S}_2$ and $\mathcal{S}_1^{\text{rem}} = \mathcal{S}_3$. If communication between agents includes all direct neighbours, the local optimization may be based on the vector $\mathbf{s}_{i,\text{nei}}[h|k]$ which is composed of local and neighbouring variables $\mathcal{S}_i$ and $\mathcal{S}_i^{\text{nei}}$. The question remains, whether such a distributed control scheme is able to achieve results of the same quality as a centralised control scheme.

The answer is partly given in a theorem due to Camponogara [5], which states 9 conditions that have to be fulfilled in order for local MPC optimalisations to convert to the optimum obtained using a centralised approach, *provided that the local controllers operate sequentially*. Reaching the global optima in a distributed environment is thus possible given a suitable system decomposition and agent coverage. Experiments have even shown that we do not need to limit ourselves to linear and convex systems [4], [5].

A. Heuristics for Asynchronous Runs

In a real world scenario, the individual agents run usually in parallel on different computers and exchange information through some type of communication network. The original requirement for sequential run is therefore difficult to fulfill. In practice it appears that results approaching globally optimal solution can be achieved also in asynchronous runs using local predictions based on information received in the previous time-step [6].

One possibility how to achieve this in our scenario is to transmit an agent's network model of subsystem $\mathcal{S}_i$ to its neighbouring agents $\mathcal{S}_i^{\text{nei}}$, that can then use a mathematical model to estimate the behavior of surrounding agents and does not need to wait for current measurements. Although the model is built on (older) information from the previous time step, it allows all agents to run in parallel.

III. Discrete Traffic Models

Let us now briefly overview two urban traffic models that we will use for our experiment. Both are discrete-time models meaning that they operate on, and produce, data sequences that are sampled at discrete time steps, given usually by the signal plan cycle length T_c or the data measurement period T.

A. R-model

The R-model [7] has been developed to study the effect of the active green phase length on the corresponding queue length. It considers a constant cycle length T_c. For the sake of simplicity we initially assume $T = T_\text{c}$, although operation with $T \neq T_\text{c}$ is also possible.

The R-model has a form of an autoregressive formula for queue length $Q_j[k]$ at some lane j at time step k. This model is based on traffic flow conservation law which leads to an equation describing the transition from the queue length $Q_j[k]$ to the new length at time-step $k+1$ as

$$Q_j[k+1] = Q_j[k] + I_j[k] - O_j[k] + e_{j,Q}[k],$$

that is, in every time step the queue decreases with the outflow $O_j[k]$ and increases with the inflow $I_j[k]$. The equation takes into account also the measurement noise $e_{j,Q}[k]$, which is related to the given lane, queue length and time step. The outflow $O_j[k]$ depends on the queue length and on the inflow, and its maximum is given by the capacity of the lane,

$$K_j[k] = s_j(G_j[k] + b_j).$$

Here, s_j denotes the saturation flow, $G_j[k]$ is the active green length for the lane and b_j is the corresponding clearing time. The outflow computation has therefore two branches for unsaturated and saturated conditions, namely

$$O_j[k] = \begin{cases} Q_j[k] + I_j[k]g_j[k] & \text{if } Q_j[k] + I_j[k]g_j[k] \leq K_j[k], \\ K_j[k] & \text{if } Q_j[k] + I_j[k] > K_j[k]. \end{cases} \tag{2}$$

Here, $g_j[k] = G_j[k]/T_\text{c}$ denotes the relative green length.

As the typical measurement device used to collect vehicle count is the inductive loop and the loops are usually located quite close to the stop-bar, the model attempts to compensate the situations when the queue tail reaches behind the loop location by incorporating an additional regressive model of loop occupancy $\omega_j[k]$

$$\omega_j[k+1] = \beta_0 + \beta_1\omega_j[k] + \beta_2 Q_j[k] + e_{j,\omega}[k],$$

where β_0, β_1, and β_2 are regressors. This model can be used to estimate $Q_j[k]$ in cases when the vehicle count measurements are not reliable due to detector saturation. The relationship between ω and Q is in fact more complex [8], but the original authors claim that regressive models with higher order approximations do not bring any significant improvement and that the linear regression model is sufficient [7].

B. S-model

The S-model [9] also works with queue length, in this case, however, the queue length is not only bound to the given lane, but also to the direction of the next turn. For every lane i we have therefore several queue lengths $Q_{i,j}[k]$ standing for the queues of vehicles that will leave lane i and enter lane j. Similarly as above, the S-model also initially assumes that the data measurement period is equal to the cycle length, $T = T_c$.

The queue length is expressed as

$$Q_{i,j}[k+1] = Q_{i,j}[k] + T_c(\Delta q_{i,j}[k] - o_{i,j}[k]), \tag{3}$$

where $\Delta q_{i,j}[k]$ is the vehicle flow entering the tail of queue $Q_{i,j}[k]$, and $o_{i,j}[k]$ is the outflow, which depends on green length $G_{i,j}[k]$, destination approach capacity C_j, queue length and queue inflow, turning rate $\alpha_{i,j}$ and vehicle count $N_j[k]$ as

$$o_{i,j}[k] = \min\left\{ \frac{s_{i,j}G_{i,j}[k]}{T_c}, \frac{Q_{i,j}[k]}{T_c} + \Delta q_{i,j}[k], \frac{\alpha_{i,j}(C_j - N_j[k])}{T_c} \right\}. \tag{4}$$

The vehicle count in approach j is computed iteratively as

$$N_j[k+1] = N_j[k] + T_c(i_j[k] - o_j[k]), \tag{5}$$

where the inflow $i_j[k]$ into the approach j is computed as a sum of all outflows that discharge into j, that is

$$i_j[k] = \sum_{i \in \Lambda_j} o_{i,j}[k].$$

The vehicle flow $\Delta q_{i,j}[k]$ of vehicles entering the tail of queue is given by

$$\Delta q_i[k] = \frac{T_c - \gamma_i[k]}{T_c} i_i[k - \delta_i[k]] + \frac{\gamma_i[k]}{T_c} i_i[k - \delta_i[k] - 1]. \tag{6}$$

Here, $\tau_i[k]$ denotes the delay with which the vehicles arrive to the tail of the current queue after leaving the previous intersection, while $\delta_i[k] = \lfloor \tau_i[k]/T_c \rfloor$ and $\gamma_i[k] = \tau_i[k] \bmod T_c$ hold the number of signal plan cycles and the offset, respectively, of this delay. The delay $\tau_i[k]$ can be expressed as

$$\tau_i[k] = \frac{(C_i - Q_i[k])l}{n_i v_{\text{ff}}}$$

where l is the length of the approach, n_i denotes the number of its lanes, and v_{ff} is the average free-flow speed of a vehicle.

C. Update for $T \neq T_c$

The requirement of both R-model and S-model for data measurement with period $T = T_c$ is difficult to fulfill in real world conditions. While the typical cycle length varies from cca 60 to cca 120 s, traffic flow measurements are provided in different periods as depicted in Fig. 2. We have to generalize both models to cope with situations when $T \neq T_c$.

1) R-model: Under assumption of constant T_c the update of R-model for different data collection periods is quite straightforward and can be accomplished by relating all T_c-related quantities to the data collection period T. The only equation affected by this change is the output model (2), where the relative green $g_j[k]$ becomes $g_j[k] = G_j[k]/T$.

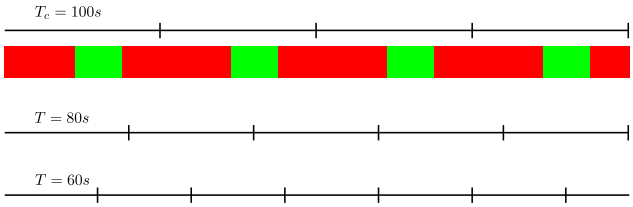

Fig. 2. Signal plan cycle length vs. different data collection periods.

2) S-model: At a first glance, the adaptation of S-model to different sampling period does not seem to be too complicated. The queue development model for time-step $k+1$ will be almost identical to Eq. (3), the only change will be using T instead of T_c,

$$Q_{i,j}[k+1] = Q_{i,j}[k] + T(\Delta q_{i,j}[k] - o_{i,j}[k]).$$

Similarly, we will update the Eq. (5) for computing the vehicle count in an approach,

$$N_j[k+1] = N_j[k] + T(i_j[k] - o_j[k])$$

and the flow of vehicles entering the queue in Eq. (6),

$$\Delta q_{i,j}[k] = \frac{T - \gamma_i[k]}{T} \sum_{j \in \Lambda_i} o_j[k - \delta_i[k]] + \frac{\gamma_i[k]}{T} \sum_{j \in \Lambda_i} o_j[k - \delta_i[k] - 1].$$

The situation gets more complicated when one starts to compute the outflow $o_{i,j}[k]$. The effective green length changes in every computational step even in cases where the effective green is kept constant — in every modelling step the relative offset of the green phase changes as the offset is relative to the signal plan length rather than to the data measurement period (see Fig. 2).

In such a case it seems beneficial to compute with the relative ratio of green length in approach i to the data collection period, $g_i[k] \in [0; 1]$. We will also need the time offset θ of the beginning of the signal plan at the intersection, $\theta \in [0; T_c - 1]$, and the delay of the green phase for the given lane from the beginning of the cycle, $\Delta G_i[k]$. Should the T_c change, we will need to keep the beginning of the cycle in memory; in cases of constant T_c, $g_i[k]$ can be eliminated out of the equation. Let us denote ℓ the count of complete signal plan cycles that completed since the end of the k-th data collection period as

$$\ell = \left\lfloor \frac{kT - \theta}{T_c} \right\rfloor. \tag{7}$$

In the k-th simulation step, the green phase starts at $\theta + \ell \cdot T_c + \Delta G_i[k]$, but this time may not exceed the end of the k-th period. The starting time will therefore be

$$g_{i,\text{start}}[k] = \min\{\theta + \ell \cdot T_c + \Delta G_i[k], (k+1)T\} \tag{8}$$

and the end time

$$g_{i,\text{end}}[k] = \min\{\theta + \ell \cdot T_c + \Delta G_i[k] + G_i[k], (k+1)T\}. \tag{9}$$

Using Eqns. (8) and (9) the effective relative green $g_i[k]$ in data measurement period T can be written as

$$g_i[k] = \frac{g_{i,\text{end}}[k] - g_{i,\text{start}}[k]}{T}. \tag{10}$$

Substituting (10) into (4) we get

$$o_{i,j}[k+1] = \min\left\{ s_{i,j} g_{i,j}[k],\ \frac{Q_{i,j}[k]}{T} + \Delta q_{i,j}[k],\ \frac{\alpha_{i,j}(C_j - N_j[k])}{T} \right\}. \tag{11}$$

IV. Linear-quadratic and non-linear control

The most popular class of system models used in control theory are linear models. As the name suggests, these models assume linear relationship between system variables and are therefore often expressed as a set of linear equations. As a result, rigorous mathematical proofs of system behaviour are often possible.

Linear-quadratic (LQ) controllers [10] are used to control dynamic linear systems by minimising a quadratic objective (cost) function $J(\mathbf{s}[h|k])$ — see Eq. (1). Under the assumption of linearity, this minimisation can be computed using a closed-form formula and is therefore relatively fast. In our case, the objective function is given by the weighted sum of squared predicted queue lengths,

$$J(\mathbf{s}[h|k]) = \sum_{i=0}^{h} \sum_{j=1}^{N} w_j \left(Q_j[k+h]\right)^2. \tag{12}$$

Unfortunately, both R-model and S-model are not strictly linear — rather they are composed of piecewise-linear components (see the Eq. (2) for R-model and Eqns. (8), (9), and (11) for S-model). The LQ control assumes that the next system state can be computed as $\mathbf{s}[k+1] = \mathbf{A}\mathbf{s}[k] + \mathbf{B}\mathbf{u}[k]$, though. In our case it means that at every step k the model is fixed to certain linear working regime and that this setpoint remains fixed even if the optimalised green lengths would cause the model to switch to another subset of equations.

Both our models can be completely described by non-linear system function f in the form $\mathbf{s}[k+1] = f(\mathbf{s}[k], \mathbf{u}[k])$. However, finding a minimal cost for such a generic system is not an easy task and advanced non-linear optimalization methods are required. One of these methods is COBYLA (Constrained optimization by linear approximation) [11], which iteratively approximates the actual non-linear optimization problem with linear programming problems and is therefore very computationally demanding.

V. Experiments

In order to demonstrate the behaviour of both models in centralised and distributed control scenarios, a series of experiments has been carried out in microscopic traffic simulator Aimsun [12]. All traffic models and control algorithms have been implemented using Aimsun API (AAPI) extensions in Python programming language. The SciPy and NumPy libraries [13] were used for linear algebra operations. An own implementation of linear-quadratic (LQ) optimizer and a SciPy package implementing the non-linear COBYLA optimization algorithm have been used for optimization.

The experimental network used for experiments described in this section is depicted in Fig. 3. Several simulation scenarios were tested, but due to space constraints only one will be presented here. The vehicle count for the presented scenario is given in Fig. 4. The remaining scenarios may be found in [14].

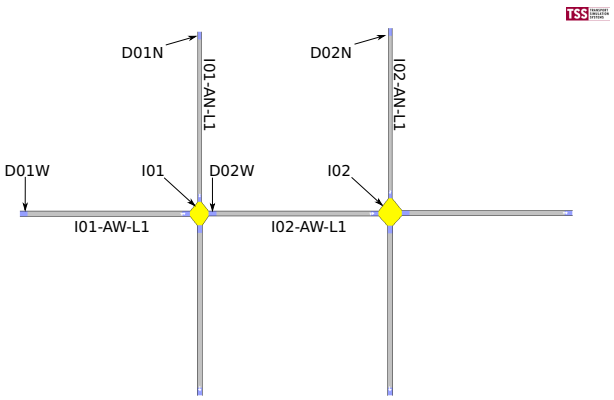

Fig. 3. Experimental network. All lanes are 100 m long and one-way, traffic flows from west to east and from north to south. Turning rates are set to 50 %.

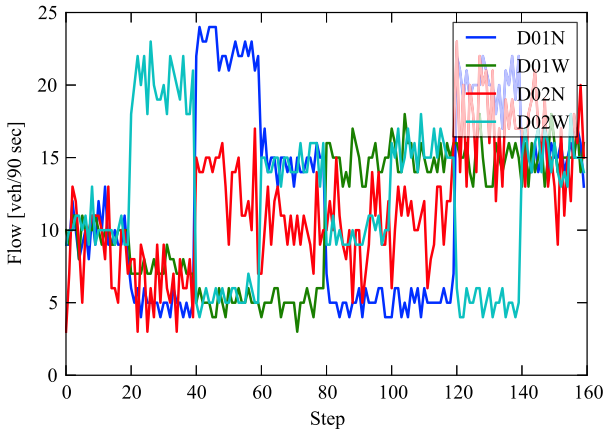

Fig. 4. Traffic demand for the presented experiment. Vehicle count per $T = 90$ s (40 samples correspond to 1 hour of data), detector labels refer to detectors in Fig. 3.

A. Effectivity of centralized LQ control on R-model and S-model

In the first experiment we studied the effectivity of LQ control by comparing the queue lengths collected by Aimsun micro-simulator before and after an application of centralized LQ control. First, the reference measurements were obtained using a fixed signal plan with phase lengths computed from the average demand given in Fig. 4. Then, both S-model and R-model were tested using LQ control, 30-minute averages were computed and the result compared with the reference values.

Even during the 30-minute interval where the traffic demand shall be quite monotonous, the queue lengths show quite a high variance. In order to verify the validity of our results, we conducted the two-sample t-test of mean values on significance level $\alpha = 0.05$. The results are shown in Tab. I. A complete analysis of experimental results from [14] shows that using the LQ control leads to mild deterioration in less than 2% cases and that in more than 80% cases the average queue length is significantly shorter.

TABLE I. DIFFERENCES IN AVERAGE QUEUE LENGTHS AND THE p-VALUES OF t-TEST FOR DATA IN FIG. 4.

		R-model		S-model	
$k_{\min}$	$k_{\max}$	$\overline{Q_{\text{ref}}} - \overline{Q_{\text{mdl}}}$	p	$\overline{Q_{\text{ref}}} - \overline{Q_{\text{mdl}}}$	p
1	20	0.75	0.390	−0.50	0.583
21	40	4.20	0.000	4.50	0.000
41	60	12.05	0.000	10.85	0.000
61	80	5.50	0.000	3.55	0.000
81	100	0.50	0.551	0.00	1.000
101	120	1.50	0.012	1.05	0.043
121	140	6.50	0.000	5.80	0.000
141	160	0.15	0.891	−0.25	0.795

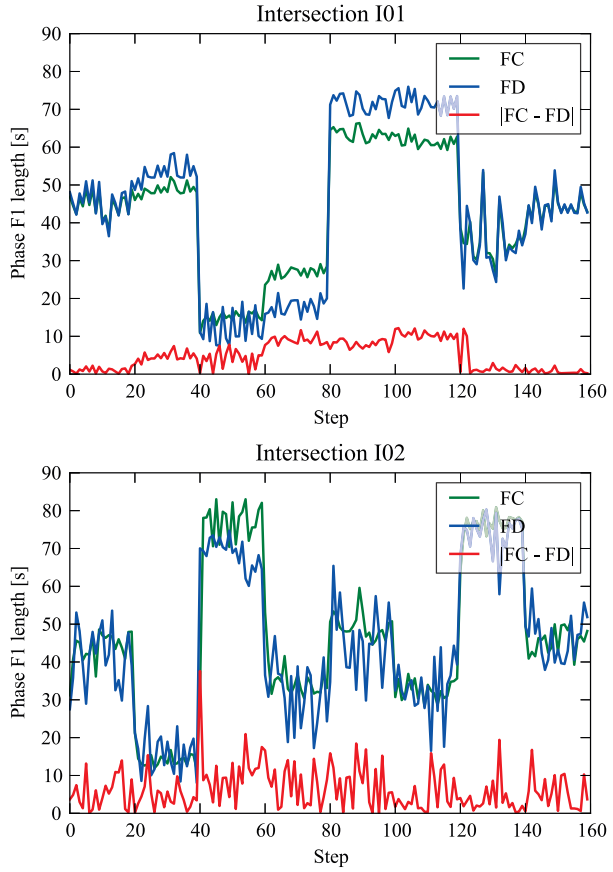

Fig. 5. Comparison of centralized and distributed LQ control of S-model on phase length F1 for intersections I01 and I02 from Fig. 3. FC: centralized LQ, FD: distributed LQ. Ideally, FD should converge to FC.

B. Convergence of distributed LQ control

In the previous subsection we have shown that a centralized management of traffic signals using the LQ optimization is a quite effective way to improve the state of the traffic situation. However, especially when using complex algorithms, the demand for computing power grows fast with the increasing number of transport network nodes. Therefore in our second experiment we studied the performance of distributed LQ control, optimizing separately each intersection in the transport network.

In Fig. 5 we can see a demonstration of distributed LQ control on two intersections from Fig. 3. Inasmuch we would like the distributed control to converge to its centralized counterpart as the theory suggests, we can see that in some cases the distributed control converges only very slowly to the centralized solution. We believe that the reason for this behaviour are the sudden changes in the traffic demand and the relatively short time (20 computational steps) that is given to the local controllers to react on the change, given their limited knowledge of the whole system.

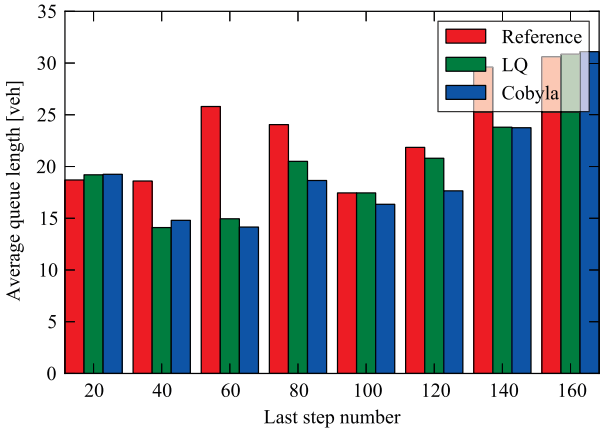

Fig. 6. Comparison of queue lengths using S-model with centralized LQ optimalisation and non-loinear COBYLA optmizer. The reference is given by optimal fixed signal plan.

C. Non-linear control using COBYLA optimizer

The piecewise-linear form of both R-model and S-model, discussed in Sec. IV suggests that better results could be achieved using non-linear control. This led us to the third set of experiments where we investigated the tradeoff between using a non-linear optimalisation method (which, in theory, should achieve better results) and the original LQ control.

1) Centralised LQ vs. COBYLA: As expected, the centralized non-linear control using COBYLA optimizer achieves results which are slightly better than those of LQ control. As we have seen already in Tab. I, there are situations where the LQ control delivers inferior results and the same can be observed also for COBYLA — see Fig. 6. This phenomenon is probably caused by the traffic model, which does not permit accurate predictions of the system state which are crucial for MPC control.

2) Distributed LQ vs. COBYLA: In the final experiment we wanted to examine whether the nonlinear distributed control using COBYLA is able to achieve similar results to its centralized version. Note that while we know that the distributed LQ control shall converge to its centralized version, there are no convergence guarantees in case of non-linear control. Again, we will limit the presentation to results for S-model, which is shown in Fig. 7. The remaining results can be again found in [14].

3) Time complexity of LQ vs. COBYLA: The main advantage of using the COBYLA optimizer instead of linear LQ control is that almost any kind of model can be used with COBYLA without any need for linearisation, model

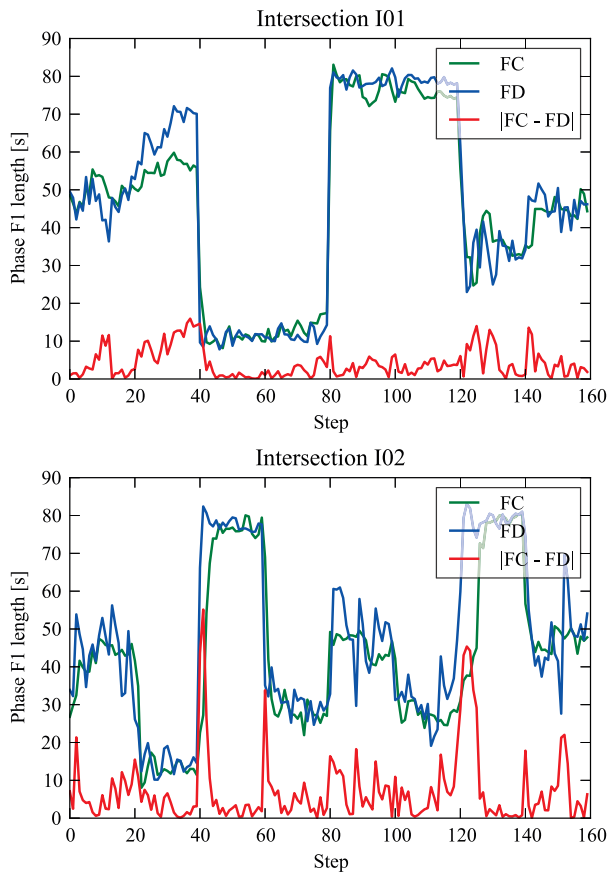

Fig. 7. Comparison of centralized and distributed non-linear control of S-model on phase lengths for intersection I01 and I02. F1: centralized COBYLA, F2: distributed COBYLA. Ideally, F2 should converge to F1.

TABLE II. TIME REQUIREMENTS FOR CONTROL SYNTHESIS FOR INTERSECTIONS WITH TWO LANES AND TWO SIGNAL PLAN PHASES. MEAN VALUE OF 10 RUNS.

Intersection count	LQ time [s]	COBYLA time [s]
1	0.006	0.184
2	0.008	0.563
3	0.012	1.121
4	0.018	1.997

simplification, and other changes. However, as we mentioned earlier, the tradeoff of this generality lies in much higher computational demands of COBYLA.

The time complexity of both methods is demonstrated in Table II. We can see that while the mean computational time for four signal plan phases (two simple intersections with two-phase signal plan) of centralized LQ scheme is still below 10 ms, the non-linear optimalization needs more than 500 ms for the same task. Moreover, the computational time grows rapidly as the number of intersections increases. This problem, though, can be at least partially eliminated by distributing the computation over all intersections – our experiments indicate that the time needed for optimizing a single intersection are in the order of 200 ms.

TABLE III. DIFFERENCES IN QUEUE LENGTHS FROM REFERENCE VALUES FOR STUDIED MODELS AND CENTRALISED AND DISTRIBUTED CONTROL METHODS.

$k_{\max}$	R+LQ	S+LQ	R+CO	S+CO	S+DLQ	S+DCO
20	$-1\vert_{-4}^{2}$	$0\vert_{-3}^{3}$	$0\vert_{-3}^{3}$	$1\vert_{-2}^{3}$	$-0\vert_{-2}^{2}$	$0\vert_{-3}^{2}$
40	$-4\vert_{-9}^{0}$	$-4\vert_{-8}^{-2}$	$-4\vert_{-10}^{0}$	$-2\vert_{-7}^{1}$	$-4\vert_{-9}^{-1}$	$-4\vert_{-8}^{2}$
60	$-12\vert_{-17}^{1}$	$-11\vert_{-15}^{1}$	$-12\vert_{-17}^{0}$	$-11\vert_{-18}^{8}$	$-12\vert_{-18}^{1}$	$-11\vert_{-16}^{-1}$
80	$-6\vert_{-12}^{1}$	$-4\vert_{-9}^{6}$	$-7\vert_{-11}^{0}$	$-7\vert_{-12}^{6}$	$-7\vert_{-12}^{-3}$	$-5\vert_{-10}^{5}$
100	$-0\vert_{-7}^{4}$	$0\vert_{-5}^{5}$	$-3\vert_{-9}^{7}$	$-3\vert_{-7}^{9}$	$-1\vert_{-6}^{9}$	$-2\vert_{-6}^{7}$
120	$-2\vert_{-6}^{3}$	$-1\vert_{-4}^{2}$	$-5\vert_{-13}^{2}$	$-5\vert_{-10}^{1}$	$-3\vert_{-7}^{2}$	$-5\vert_{-9}^{-1}$
140	$-6\vert_{-12}^{2}$	$-6\vert_{-12}^{2}$	$-4\vert_{-11}^{7}$	$-3\vert_{-10}^{6}$	$-6\vert_{-11}^{5}$	$-4\vert_{-9}^{4}$
160	$-0\vert_{-7}^{12}$	$0\vert_{-5}^{10}$	$-1\vert_{-12}^{12}$	$2\vert_{-4}^{18}$	$-0\vert_{-7}^{12}$	$3\vert_{-5}^{17}$

Entry format: average difference$\vert_{\text{minimal difference}}^{\text{maximal difference}}$. *Abbreviations:* R, S – models, LQ – LQ control, CO – non-linear optimalisation using COBYLA, DLQ, DCO – distributed variants of LQ and CO.

VI. CONCLUSION

In this paper we have studied two approaches to distributed control of urban traffic network — distributed LQ control and distributed non-linear control using COBYLA algorithm. The results are summarized in Tab. III. Our experiments using two different discrete models of queue length development have demonstrated that both LQ and COBYLA are suitable to control our experimental network and that their distributed variants produce results that are comparable to those of centralised control mechanisms.

REFERENCES

[1] R. Scattolini, "Architectures for distributed and hierarchical Model Predictive Control – A review," *Journal of Process Control*, vol. 19, pp. 723–731, 2009.

[2] M. Morari, C. Garcia, J. Lee, and D. Prett, *Model predictive control*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[3] R. Negenborn, B. D. Schutter, and J. Hellendoorn, "Multi-agent model predictive control for transportation networks: serial versus parallel schemes," *Engineering Applications of Artificial Intelligence*, 2008.

[4] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 44–52.

[5] E. Camponogara, "Controlling networks with collaborative nets," Ph.D. dissertation, Carnegie Mellon University, 2000.

[6] D. Jia and B. H. Krogh, "Distributed model predictive control," in *Proceedings of the American Control Conference*, 2001, pp. 2767–2772.

[7] J. Homolová, "Traffic flow control management," Ph.D. dissertation, Czech Technical University in Prague, Faculty of Transportation Engineering, Prague, 2007.

[8] J. Přikryl and J. Kocijan, "Modelling occupancy-queue relation using gaussian process," *Neural Network World*, vol. 25, no. 1, pp. 35–52, 2015.

[9] S. Lin, "Efficient model predictive control for large-scale urban traffic networks," Ph.D. dissertation, Delft University of Technology, 2011.

[10] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[11] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in optimization and numerical analysis*, S. Gomez and J.-P. Hennart, Eds. Dordrecht: Kluwer Academic, 1994, pp. 51–67.

[12] *AIMSUN Getram v4.2 getting started - User's manual*, 2003.

[13] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2015-05-01]. [Online]. Available: http://www.scipy.org/

[14] J. Novotný, "Distributed traffic signal control," Master thesis, Czech Technical University in Prage, Faculty of Nuclear Sciences and Physical Engineering, 2014, in Czech.