Available online at www.sciencedirect.com

ScienceDirect

The International Symposium on Transportation and Traffic Theory 00 (2013) 1–

Procedia
Social and Behavioral Sciences

www.elsevier.com/locate/procedia

Linear-Quadratic Model Predictive Control for Urban Traffic Networks

Tung Le[a], Hai L. Vu[a]
Yoni Nazarathy[b], Bao Vo[a] and Serge Hoogendoorn[c].
[a]*Swinburne University of Technology, Melbourne Australia.*
[b]*School of Mathematics and Physics, The University of Queensland, Brisbane, Australia.*
[c]*Delft University of Technology, Delft, The Netherlands..*

Abstract

Advancements in the efficiency, quality and manufacturability of sensing and communication systems are driving the field of intelligent transport systems (ITS) into the twenty first century. One key aspect of ITS is the need for efficient and robust integrated network management of urban traffic networks. This paper presents a general model predictive control framework for both centralized traffic signal and route guidance systems aiming to minimize network congestion. Our novel model explicitly captures both non-zero travel time and spill-back constraints while remaining linear and thus generally tractable with quadratic costs. The end result is a central control scheme that may be realized for large urban networks containing thousands of sensors and actuators.

We demonstrate the essences of our model and controller through a detailed mathematical description coupled with simulation results of specific scenarios. We show that using a central scheme such as ours may reduce the congestion inside the network by up to half while still achieving better throughput compared to that of other conventional control schemes.

Keywords: Model Predictive Control, Intelligent Transport System, Congestion Control

1. Introduction

Increasing population and economic activities in modern societies have led to a significant rise in the demand for mobility and transportation. Consequently, urban road networks are becoming frequently congested which creates severe economical, social, and ecological challenges. Research into traffic management systems aims to make a better use of existing network infrastructure to improve traffic conditions. Nowadays, traffic control centers receive data from remote sensors and apply control policies that respond to the prevailing traffic conditions. While real-time signal control systems responding to traffic conditions can help in alleviating congestions, optimal network-wide control strategies remain a challenge due to the combinatorial nature of the related optimization problems (see e.g., Papageorgiou, 2003).

Historically, the SCOOT (Hunt, 1982) and SCATS (Lowrie, 1982) systems were among the earliest efforts to develop adaptive traffic control systems in the 1970s. These well-known and widely-used traffic-

responsive control systems are based on heuristic optimization algorithms. In the 1980s, new optimization methods for traffic control were introduced based on rolling horizon optimization using dynamic programming – the prominent examples include OPAC (Gartner, 1983), PRODYN (Farges *et al.*, 1983) and RHODES (Mirchandani & Head, 1998) or backtracking search on complex decision tree implemented in ALLONS-D (Porche *et al.*, 1996). The more recent works on traffic control systems have adopted results of modern control theory. In particular, the TUC system (Diakaki *et al.*, 1999; Diakaki, 2002) applies a multivariable feedback regulator approach to calculate in real time the signal control plans (splits) as a linear-quadratic (LQ) optimal control problem. This approach is based on the store-and-forward model of traffic network which was first proposed by Gazis and Potts (Gazis & Potts, 1963). While its simplicity enables efficient algorithms for split optimization, cycle time and offsets must be delivered by other control algorithms (Diakaki, 2003). H. M. Abdul Aziz (2012) recently presents an optimal control framework which is based on the cell-transmission-model. The model assumes discrete time, discrete space and linear relationship between density and flow. Another linear programming approach is the work of Waller & Ziliaskopoulos (2006). Their model provides robust solutions while explicitly considers constrains, however, it only focuses on single destination networks.

Early works on traffic control systems based on the theory of Model Predictive Control (MPC) focused on the problems of ramp metering and variable speed limit control in freeway traffic management (Bellemans *et al.*, 2002; Hegyi *et al.*, 2003) where a free way link is geometrically divided into segments. Each segment is characterized by traffic density, mean speed and flow. Although the prediction model provides a comprehensive expression of fundamental traffic diagram on each link, its non-linearity feature is impractical in large urban networks. Aboudolas *et al.* (2009, 2010) and many others such as Tettamanti *et al.* (2008, 2010); Tettamanti & Varga (2010) investigate an MPC-based approach to urban traffic control. In their approach, the store-and-forward traffic modeling is employed to represent the states of the links in an urban road traffic network. The objective of the control system is to reduce the risk of congestions by balancing the number of vehicles between links. However, their approach focuses on long term converged states of links rather than the rapid evolution of traffic phenomena inside link e.g. the wave of congestion or vehicles speed; and the travel time in the roads between intersections is ignored. Another MPC-based approach introduced by Lin *et al.* (2011) considers vehicles speed by estimating the time of travel through links by a non-linear non-convex prediction model. In order to overcome computational complexity this model assumes constant delay and thus leads to the model inaccuracy in light traffic.

Our paper proposes an urban traffic control strategy to coordinate the green time split and turning fractions at intersections aiming to minimize the number of vehicles in the controlled area. The model is designed to predict the queuing dynamics through links while retaining the linearity of state evolution equations. Retaining linearity is important since the main idea behind the MPC approach is to repeatedly solve optimization problems on-line to find a near-optimal control strategy for a large network. To this end, we smartly describe and emulate real traffic behavior in the proposed MPC framework using only linear dynamics and constraints. In our framework, the roads are modeled as series of queues where the nonuniform characteristics of links are captured by different queue parameters. Travel time on the link is effectively taken into consideration by modeling the vehicles' movements from queue to queue. The turning fractions of vehicles between queues are treated as control variables, assuming full compliance from drivers with our centralized controller. The route guidance without concerning about drivers' preferences is applicable in the emergency scenarios such as disaster evacuation or in traffic scenarios with heavy congestion where exiting the controlled area is drivers' first priority. In contrast to store-and-forward models, our model can essentially adapt to changes of link conditions due to accidents and other disturbances. In short, our approach is superior than the above approaches in three aspects: we explicitly consider travel time in the road between intersections; our model unifies traffic light control and route guidance; the state prediction equation in our proposed MPC framework is linear so the model is capable to handle large network in real time control.

The main contribution of this paper is thus a novel framework for real time control of urban traffic networks. It allows for optimization of network wide conditions by jointly considering traffic control and route guidance. The numerical comparison of our model with other control schemes shows that in general our model is superior in reducing congestion.

The remainder of the paper is organized as follows. Section 2 describes the our discrete time MPC

framework. Section 3 discusses the simulation and results. We conclude in Section 4.

2. Discrete Time MPC Framework

MPC is a multi-variable control method that is usually used to optimally control complex systems while explicitly considering constraints. The system dynamics are represented by a discrete time predictive model where the next state is a function of the current state, current demands (disturbance) and the current control vector with constraints. In each time instance, the optimal control problem is solved online based on the measured (estimated) current state (at time $n = n_0$) and the predictive demands over a N step finite horizon (at time $n = n_0, \ldots, n_0 + N - 1$). The result of the optimization is a sequence of control vectors over time $n = n_0, \ldots, n_0 + N - 1$ but only the first control vector (at time $n = n_0$) is applied to the system. In the next state (at time $n = n_0 + 1$), the optimal control problem is solved again for time horizon $n = n_0 + 1, \ldots, n_0 + N$ then only the control vector at time $n = n_0 + 1$ is applied and so forth. Section 2.1 below describes our predictive model while Section 2.2 presents our optimal control problem in detail.

2.1. A General Controlled Network Model

Below is a technical definition of our network model to be used as part of the MPC controller. The main attribute of the model is that it evolves in time synchronous with traffic light cycles (similarity to a cellular automaton model), yet the dynamics have been kept linear with an assumption of continuous vehicle counts. This ensures general computational tractability yet presents a model that is accurate enough to capture phenomena appearing in real urban traffic networks. Our novelty is in using linear dynamics and constraints wisely to describe and emulate real traffic behavior.

2.1.1. Definition Of Network Elements And Control Vectors

Our representation of urban traffic networks is motivated by multi-class queueing networks (van Leeuwaarden *et al.*, 2010), where classes relate to different types of network elements. We assume that time evolves in discrete steps $n = 0, 1, 2, \ldots$ corresponding to traffic light cycles. The network state vector is denoted by $X(n)$ and the control vector is denoted by $U(n)$. The network state maintains counts of the quantity (continuous approximation) of vehicles at different abstractions of locations based on the following 4 types of vehicle classes: *delay* (D), *route* (R), *queue* (Q) and *sink* (S). Vehicles originate exogenously, pass through delay, route, and queue classes, then end up in sink classes upon reaching destination. On the way, vehicles are subject to routing control and traffic lights at intersections.

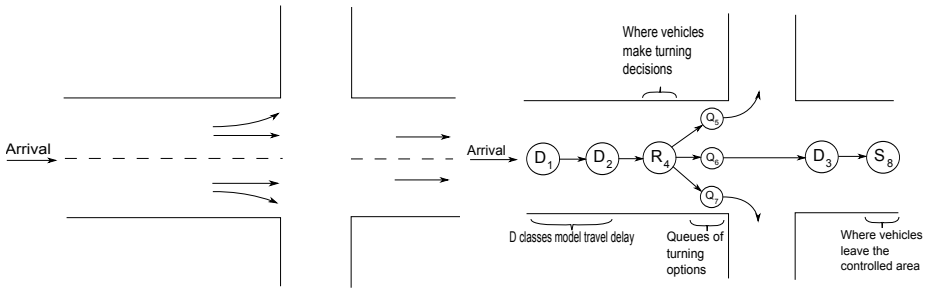
Let K_D, K_R, K_Q, K_S denote the number of classes of each type respectively. Classes are indexed by $k = 1, \ldots, K$ with $K = K_D + K_R + K_Q + K_S$. We denote the classes of type j by $\mathcal{K}_j$ for $j \in \{D, R, Q, S\}$ and we number the classes as follows: $\mathcal{K}_D = \{1, \ldots, K_D\}$, $\mathcal{K}_R = \{K_D + 1, \ldots, K_D + K_R\}$, etc. For convenience, we use the notation $\mathcal{K}_{\{j,j'\}}$ to indicate the union of $\mathcal{K}_j$ and $\mathcal{K}_{j'}$, for $j, j' \in \{D, R, Q, S\}$, sometimes extending the notation to more indices. For example, $\mathcal{K}_{\{D,R,Q\}}$ are all classes except the sink classes. Also for readability, let $K_{\{j,j'\}} = |\mathcal{K}_{\{j,j'\}}|$ and the same with more indexes.

The total number of vehicles "in transit" at any time n is distributed among the $K_{\{D,R,Q\}}$ classes of type D, R and Q. We now describe the classes in more detail:

Queue classes: A queue class represents a turning direction before an intersection. Each queue has capacity, $c_k, k \in \mathcal{K}_Q$, which is the maximum number of vehicles that can be in queue k, typically based on physical limitation.

Route classes: A route class represents a portion of a multiple-lane street before queue classes where, under the assumption of full drivers' compliance, the turning rates to downstream queues can possibly be controlled. Similar to queue classes, each route class has its own capacity, $c_k, k \in \mathcal{K}_R$.

Delay classes: A delay class represents a portion of a multiple-lane street where vehicles just have one turning option to move to, its downstream D or R class. Delay classes are used to model free flow traffic. Each delay class also has its capacity, $c_k, k \in \mathcal{K}_D$.



(a) A real network segment (b) A model network segment

Fig. 1: An illustration of classes

Sink classes: A sink class maintains cumulative counts of arrivals to a destination. Keeping track of this quantity in the state allows our model to emulate the destination preferences of drivers in a macroscopic manner.

Vehicles arrive exogenously according to pre-estimated time-varying quantities $a_k(n)$ to class k, $k \in \mathcal{K}_{D,R,Q}$. $a_k(n)$ usually has positive value in some classes k in the edge of the network and zero value elsewhere. This means that at each time step, n, $a_k(n)$ vehicles arrive to class k from outside of the network. Similarly, at each time step, vehicles potentially leave queue, route, and delay classes and move downstream. A class $k \in \mathcal{K}_{D,R,Q}$ with exogenous arrival $a_k(n) > 0$ has $c_k = \infty$ to avoid the situation that the arrival traffic exceeds the class' capacities.

A typical urban traffic road that connects two intersections has a number of delay classes at the beginning. This number is proportional to the length and flow characteristic of that of street. In practice the number of delay classes may be adjusted based on time of day and overall congestion level, yet for simplicity we keep it constant in this paper. The delay classes are followed by a route class and then by queue classes at a junction. The number of queue classes is the number of turning directions at the end of the street. See Figure 1 for an illustrative example.

As time evolves, vehicle mobility is modeled by shifting the continuous approximation of vehicles from classes to other classes. This is represented by means of the network state.

Network states: The network state for class k, $x_k(n)$, is a instantaneous continuous count of vehicles in class $k \in \mathcal{K}_{D,Q,R,S}$ at time n. We define the vectors of network states as follows: $X_D = [x_1 \dots x_{K_D}]'$, $X_R = [x_{K_D+1} \dots x_{K_D+K_R}]'$, etc. For convenience, we also use the notation $X_{\{i,j\}}$ to indicate $[X_i' \, X_j']'$, for $i, j \in \{D, R, Q, S\}$, sometimes extending the notation to more indices.

Links: Vehicles flowing out of a class move to downstream classes through predefined links. Only route classes may be the origin of more than one link, meaning that vehicles have to make turning decisions to move to different downstream classes at the next step. Obviously, there are no link start at sink classes. A link, j, is a directional connection between its source class, s_j, $s_j \in \mathcal{K}_{D,R,Q}$, and its destination class, d_j, $d_j \in \mathcal{K}_{D,R,Q,S}$. We number the links $j = 1, \dots, L$. With each link we associate a flow rate, f_j, which is the maximum number vehicles that can go from s_j to d_j in an unit of time. These flow rates are to be adjusted in the model based on traffic measurements. In the current paper we assume they are known.

Within a time step we assume dynamics are occurring at a rate faster than that of our model and controller. We thus assume that at each time instant a link may be active, inactive or combination of both. For the instants during which a link is active, vehicles are assumed to move through that link at rate f_j. A link is inactive means there is no vehicles move through the link.

Time fractions: A time fraction, $u_j(n)$, is a variable that defines the fraction of a time unit during which link j is active. So the number of vehicles that leave class s_j for class d_j is $f_j u_j(n)$. The number of vehicles that leave class k is $\sum_{\{j:s_j=k\}} f_j u_j(n)$. Similarly, the number of vehicles join class k is $\sum_{\{j:d_j=k\}} f_j u_j(n)$. We denote a set of all time fractions $\mathcal{U}$ and number them $j = 1, \dots, L$. Note that the time fractions determine both the flow of vehicles in the network and the effective turning rates at routing classes.

In addition to time fractions, our framework also features a second type of control variable: green time fraction at intersections. As described below these are upper limits of the corresponding time fractions.

Traffic intersections: We denote the set of all intersection by $\mathcal{M}$ and number them $i = 1, \ldots, M$. For simplicity in this paper we assume all intersections are of the "West-East"/"North-South" type. Each intersection has two traffic light control variables $u_{L+1+(i-1)*2}(n)$, the green duration for West-East direction, and $u_{L+1+(i-1)*2+1}(n)$, the green duration for North-South direction. This seemingly complicated indexing is simply to allow setting all control variables in an ordered vector. We denote the set of all traffic light control variables by $\mathcal{T}$, and number them $j = L + 1, L + 2, \ldots, J$ where $J = L + 2 * M$.

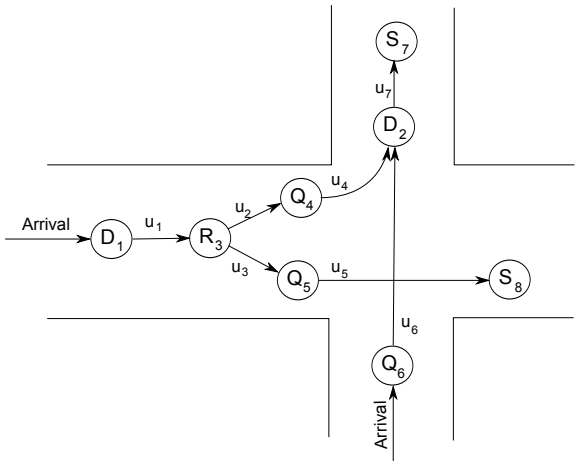


Fig. 2: Arrival rate: $a_1(0) = 20$, $a_6(0) = 10$, other arrival rates have value 0. Flow rate: $f = 20$ for all classes. Initial state: $x(0)$=[30 0 20 10 10 20 0 0]'. Time fraction: $u(0)$=[1 0.2 0.3 0.5 0.5 0.5 0]'. Based on the definitions, the next state is: $x(1)$=[30 20 30 4 6 20 0 10]'

Figure 2 contains a simple example to demonstrate how arrival rates, $a_k(n)$, and time fractions, $u_j(n)$, of current state affect the next state. Note that the network state at $n = 1$ may become undesirable with certain variable $u(0)$ e.g. negative values. Thus, we introduce a number of constraints to keep a realistic network evolution in section 2.1.2.

2.1.2. Constraints

This section describes a list of constraints that restrict the feasibility region of variables defined in the previous section.

Non-negative control constraints: For $1 \le j \le L$, the number of vehicles that move through link j at time n is a non-negative number and less than f_j. If $L + 1 \le j \le J$, the green duration for a direction at an intersection is between 0 and 1, a traffic light cycle. Thus, we have constraints:

$$0 \le u_j(n) \le 1, \quad 1 \le j \le J. \tag{1}$$

Traffic light cycle constraints: Sum of the green duration for West-East direction and the green duration for North-South direction at any intersection have to be less than or equal to 1, the traffic light cycle.

$$u_{L+1+(i-1)*2}(n) + u_{L+1+(i-1)*2+1}(n) \le 1, \quad i \in \mathcal{M}. \tag{2}$$

Green duration constraints: The active time of a link across an intersection must be less than or equal to the corresponding green time. We define $\mathcal{WE}(i), i \in \mathcal{M}$, as the set of links which are active during *green duration* at West-East direction. Similarly, $\mathcal{NS}(i), i \in \mathcal{M}$, as the set of links which are active during *green*

duration at North-South direction. Note that we ensure that $\mathcal{WE}(i)$ and $\mathcal{NS}(i)$ are disjoint sets. For $i \in \mathcal{M}$:

$$\begin{aligned} u_j(n) &\leq u_{L+1+(i-1)*2}(n), \quad j \in \mathcal{WE}(i), \\ u_j(n) &\leq u_{L+1+(i-1)*2+1}(n), \quad j \in \mathcal{NS}(i). \end{aligned} \tag{3}$$

Flow conflict constraints: At intersection, two links may collide, e.g. j_3 and j_5 or j_2 and j_4 in Figure 2. In that case, at most one link may be active at the same time. We define $C^p_{\mathcal{WE}}(i)$ to be subset of $\mathcal{WE}(i), i \in \mathcal{M}$, such that if $j, j' \in C^p_{\mathcal{WE}}(i)$, they cross each other. Hence, typically $|C^p_{\mathcal{WE}}(i)| = 2$. We number the path collisions occurring at intersection i when the *green traffic light* is active at West-East direction by $p = 1, 2, \ldots, C_{WE}(i)$. Similar definitions hold for $C^p_{\mathcal{NS}}(i)$ and $C_{NS}(i)$. These variables take part in the following constraints. For $i \in \mathcal{M}$:

$$\begin{aligned} \sum_{j \in C^p_{\mathcal{WE}}(i)} u_j(n) &\leq u_{L+1+(i-1)*2}(n), \quad p = 1, \ldots, C_{WE}(i), \\ \sum_{j \in C^p_{\mathcal{NS}}(i)} u_j(n) &\leq u_{L+1+(i-1)*2+1}(n), \quad p = 1, \ldots, C_{NS}(i). \end{aligned} \tag{4}$$

Note that this constraint does not take into account the link priority such as for example when turning vehicles have to give way to go-straight vehicles. Such behavior can be achieved by setting a slightly bigger flow rate for higher priority links. Further details are in section 3.

Non-negative queue constraints: A class can only be drained if there are vehicles in it. We thus have the constraint:

$$\sum_{\{j:s_j=k\}} u_j(n) f_j \leq x_k(n), \quad k \in \mathcal{K}_{D,R,Q}. \tag{5}$$

Capacity constraints: Additionally, we also impose the constraints in which the number of leaving vehicles of any class will be limited by the states and capacities of its downstream classes. For $k \in \mathcal{K}_{D,R,Q}$:

$$\begin{aligned} x_k(n+1) &\leq c_k, \\ x_k(n) + a_k(n) + \sum_{\{j:d_j=k\}} u_j(n) f_j - \sum_{\{j:s_j=k\}} u_j(n) f_j &\leq c_k. \end{aligned} \tag{6}$$

The number of vehicles in a class $k \in \mathcal{K}_{D,R,Q}$ in the next state is the total of the number of vehicles in this class in the current state plus the number of vehicles arriving exogenously, plus the number of vehicles arriving from other classes, and minus the number of vehicles leaving this class. Constraint (6) states that the number of vehicles in a class k in the next state can never exceed the class capacity c_k. This "key constraint" allows us to model spill-back – *yet with a linear model!* Given that the arrival rate is greater than departure rate of a road, e.g. red traffic light, and vehicles tend to move forward, this constraint causes congestion build up from downstream queues, right before intersection, up to upstream classes.

For further illustration of constraints (2), (3) and (4) consider the example intersection in Figure 3 where the traffic light cycle constraint (2) is $u_7 + u_8 \leq 1$; the green duration constraints (3) are $u_j \leq u_7$, $j = 1, \ldots, 7$; and flow collision constraints (4) are $u_2 + u_4 \leq u_7, u_5 + u + 3 \leq u_7$. The latter (i.e. flow collision constraints) ensure that the controller will not guide vehicles through conflicting flows more than the amount they can handle, thus prevent unnecessary congestion.

2.1.3. State Dynamics

The network state, $x_k(n)$, $k \in \mathcal{K}_{\{D,R,Q,S\}}$ is a (continuous) count of the number of vehicles in class k at time n. Hence the network state is $K_{\{D,R,Q,S\}}$ dimensional. Observe also that the state of sink classes is non-decreasing. The control U is J dimensional.

The state evolves as follows:

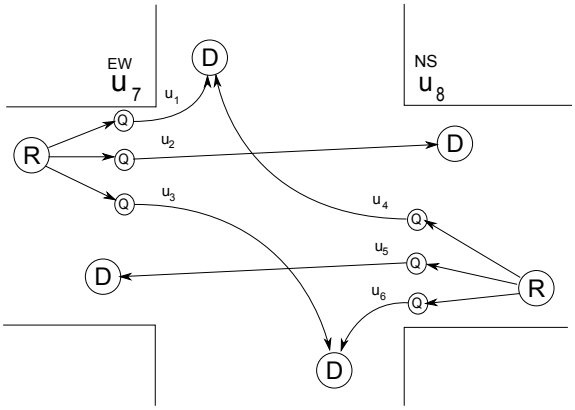


Fig. 3: Each Q represents a turning direction and the arrows indicate traffic flows. $u_1 \dots u_6$ are time fraction control variable. u_7 controls the green duration in West-East direction and u_8 is for the North-South direction. $M = 1$, $L = 6$. $\mathcal{WE}(1) = \{1, 2, 3, 4, 5, 6\}$. $C_{WE}(1) = 2$. $C^1_{\mathcal{WE}}(1) = \{2, 4\}$. $C^2_{\mathcal{WE}}(1) = \{5, 3\}$. Traffic light cycle constraints: $u_7 + u_8 \leq 1$. Green duration constraints: $u_j \leq u_7, j = 1, \dots, 7$. Flow collision constraints: $u_2 + u_4 \leq u_7, u_5 + u + 3 \leq u_7$

$$x_k(n+1) = x_k(n) + a_k(n) + \sum_{\{j:d_j=k\}} u_j(n) f_j - \sum_{\{j:s_j=k\}} u_j(n) f_j, \quad k \in \mathcal{K}_{D,R,Q,S}. \tag{7}$$

Subject to the constraints appearing in (1-6) for all times n.

2.1.4. Matrix Representation

Efficient computation and optimization using our model requires a matrix-based formulation. We present this briefly below. Our system is a constrained time-varying affine system based on matrices B, F and vectors $d(n)$ and $g(n)$ as defined below. We now represent the state evolution by

$$X(n+1) = X(n) + B\,U(n) + d(n). \tag{8}$$

Where $X(n) = [x_1(n) \dots x_K(n)]'$, $U(n) = [u_1(n) \dots u_J(n)]'$. We also partition $U(n)$ into $U_l(n)$, of dimension L, and $U_t(n)$, of dimension 2*M, such that $U(n) = [U_l'(n)\, U_t'(n)]'$.

The detailed expression of state dynamic equation which contains the matrix B and the vector $d(n)$ are as follows:

$$\begin{bmatrix} X_{D,R,Q}(n+1) \\ X_S(n+1) \end{bmatrix} = \begin{bmatrix} X_{D,R,Q}(n) \\ X_S(n) \end{bmatrix} + \begin{bmatrix} (D_d - D_s)diag(f) & 0 \\ D_{dS}\,diag(f) & 0 \end{bmatrix} \begin{bmatrix} U_l(n) \\ U_t(n) \end{bmatrix} + \begin{bmatrix} a(n) \\ 0 \end{bmatrix}. \tag{9}$$

We assume that the control $U(n)$ is a state feedback control with full observations (it is a function of $X(n)$). At every time n, given $X(n)$, the control needs to satisfy linear constraints of the form

$$F \begin{bmatrix} X(n) \\ U(n) \end{bmatrix} \leq g(n).$$

The detailed expression of linear constraints which contains the matrix F and vector $g(n)$ are as follows:

$$\begin{bmatrix} 0 & -I & 0 \\ 0 & 0 & -I \\ 0 & I & 0 \\ 0 & 0 & H \\ 0 & S_{WEM} & -S_{WEm}S_{WEt} \\ 0 & S_{NSM} & -S_{NSm}S_{NSt} \\ 0 & S_{WEC} & -S_{WEc}S_{WEt} \\ 0 & S_{NSC} & -S_{NSc}S_{NSt} \\ -I & D_s diag(f) & 0 \\ I & (D_d - D_s)diag(f) & 0 \end{bmatrix} \begin{bmatrix} X_{D,R,Q}(n) \\ U_l(n) \\ U_t(n) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ c - a(n) \end{bmatrix}. \tag{10}$$

Here the first three rows correspond to constraints (1), the forth row corresponds to constraints (2), the fifth and sixth rows correspond to constraints (3), the seventh and eighth rows correspond to constraints (4), the ninth row corresponds to constraints (5) and the last row corresponds to constraints (6).

Where:

- f is the vector of flow rates where element j is equal to f_j. Note that f_j is defined in section (2.1.1)
- For a vector, f, $diag(f)$ denotes a diagonal matrix with diagonal elements f.
- D_s is a $K_{D,R,Q} \times L$ matrix with element (k, j) set to 1 if $s_j = k$ and 0 otherwise.
- D_d is a $K_{D,R,Q} \times L$ matrix with element (k, j) set to 1 if $d_j = k$ and 0 otherwise.
- D_{dS} is a $K_S \times L$ matrix with element (k, j) set to 1 if $d_j = k$ and 0 otherwise.

The definitions of other matrices of (10) are given in the Appendix B.

2.2. The MPC Controller

This section describes the MPC approach in detail. The controller is parameterized by a discrete *time horizon* $N > 0$, a positive semi-definite matrix $Q = \mathbf{1}'\mathbf{1}$, of dimensions $K_{\{D,R,Q\}}$ and a vector $R = -\epsilon f'$, $\epsilon \ll 1$, of dimension K_L. At time n, the controller uses the optimal solution of a quadratic programming (QP) problem in which the decision variables are the controls over the time horizon: $n, n+1, \ldots, n+N-1$. Given the current state $X(n)$ and the controls $U(i)$, $i = n, \ldots, n+N-1$, the prediction of the state over the time horizon, denoted by $\hat{X}$, is generated (details below) and appears in the objective and constraints of the QP.

We partition F to $[F_x\, F_u]$ where F_x is a $K_{D,R,Q}$ column matrix and F_u is a J column matrix. The following is the QP formulation:

$$\begin{aligned} &\min \textstyle\sum_{i=n}^{n+N-1} \hat{X}_{D,R,Q}(i+1)' Q \hat{X}_{D,R,Q}(i+1) + RU(i) \\ &\text{s.t.} \\ &F_x \hat{X}_{D,R,Q}(i) + F_u U(i) \leq g(i), \quad i = n, \ldots, n+N-1. \end{aligned} \tag{11}$$

The objective function in (11) quadratically penalizes the number of vehicles in the network and linearly penalizes the control decision. Generally, the MPC model is able to serve different control objectives depending on how the positive semi-definite matrix Q is set-up. One possibility is minimizing the sum of weighted queue lengths (Aboudolas *et al.*, 2010) by setting Q as a diagonal matrix where the diagonal elements correspond to the queue capacities. However, we would like to maximize network throughput which is equivalent to minimize the total number of vehicles in the entire network. That is why Q is all 1 matrix. Consequently, keeping vehicles in any particular class nearly has the same penalty as moving those vehicles to other classes. The only small cost difference is in term RU as is addressed later in this section. It may be argued that our objective, minimizing the sum of all queue lengths, increases the risk of spill back because the controller tries to guide vehicles to the shortest route, but we believe that the benefit of overall system

performance out-weighs the risk of spill back in some network segments. Additionally, the simulations show that our MPC controller archives good congestion control in all the scenarios studied.

Note that the above formulation does not take the sink classes into account. These may be incorporated in order to accommodate for driver routing preferences on a macroscopic scale. This is the subject of our future research.

An attribute of our formulation is that for large queues, the linear cost is negligible in comparison to the quadratic cost, so the primary objective is to reduce the quadratic cost through the matrix Q. Only when the quadratic cost can no longer be reduced, a secondary objective achieved through the optimization is to minimize RU. Hence, we take advantage of the small linear cost to model the non-linearity dynamics of traffic flows without compromising the main objective of our QP solver. Particularly, the linear cost addresses a so-called holding back problem in the optimal traffic assignment. Holding back occurs when vehicles are held back even when there is available space in the downstream queue. The linear cost gives small incentive for vehicles to move forward, thus, it prevents the holding back issue.

The output of the QP optimization is $U = [U_l' \, U_t']'$. However, the only control variables are U_t, the optimal green splits, and some variables $u_j \in U_l$, $s_j \in \mathcal{K}_R$, the optimal turning rates. The other variables only serve the purpose of predicting the number of vehicles move to downstream classes. This is a further novelty of our model – as it allows to incorporate "cellular automaton" type behaviour in a model that is linear and thus tractable. For illustration, consider a road segment far away from intersections which is modeled by a sequence of D classes. The number of vehicles traveling to the downstream class in next step is determined by the biggest number of the number of vehicles currently in the upstream class, available space in the downstream class, flow rate etc... Thus vehicles tend to move forward if it is possible due to the small negative cost RU. This allows the MPC controller to effectivly predict future states.

Finally, the constraints in (11) are just repetitions of (10) over the horizon.

Detailed Description of the QP

In order to facilitate implementation, we now give the detail description of our QP. For readability, we assume in the description below that $n = 0$, generalization to arbitrary time (due to time varying arrivals), is straightforward and we also exclude S classes from all of the following equations. Define

$$\underline{\hat{\mathrm{X}}} = \underline{\mathrm{A}} X_0 + \underline{\mathrm{B}}\,\underline{\mathrm{U}} + \underline{\mathrm{A}}_1\,\underline{\mathrm{d}},$$

where

$$\underline{\mathrm{A}} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}, \quad \underline{\mathrm{B}} = \begin{bmatrix} B & 0 & \cdots & 0 \\ B & B & & \vdots \\ \vdots & & \ddots & \\ B & & \cdots & B \end{bmatrix}, \quad \underline{\mathrm{d}} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N-1) \end{bmatrix}, \quad \underline{\mathrm{A}}_1 = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & & \vdots \\ \vdots & & \ddots & \\ I & I & \cdots & I \end{bmatrix},$$

$$\underline{\hat{\mathrm{X}}} = \begin{bmatrix} \hat{X}(1) \\ \hat{X}(2) \\ \vdots \\ \hat{X}(N) \end{bmatrix}, \quad \underline{\mathrm{U}} = \begin{bmatrix} U(0) \\ U(1) \\ \vdots \\ U(N-1) \end{bmatrix}.$$

We denote by $\underline{\mathrm{Q}}$ block diagonal matrix of Q with dimensions $N \cdot K_{\{D,R,Q\}}$, and by $\underline{\mathrm{R}}$ block vector that stack R N times. Multiplying $\underline{\mathrm{S}}_U^i$, $i = 0, \ldots, N-1$, by the control vector over the whole time horizon, $\underline{\mathrm{U}}$, results in the control vector at the i time step. Similarly, multiplying $\underline{\mathrm{S}}_X^i$, $i = 1, \ldots, N$, by the state vector over the whole time horizon, $\underline{\hat{\mathrm{X}}}$, results in the control vector at the i time step. Therefore, $\underline{\mathrm{S}}_U^i$ is a $J \cdot NJ$ block matrix where I is at $(i+1)$th block. Similarly, $\underline{\mathrm{S}}_X^i$ is a $K_{D,R,Q} \cdot (N-1)\, K_{D,R,Q}$ block matrix where I is at (i)th block

$$\underline{\mathrm{S}}_U^i = \begin{bmatrix} 0 \cdots I \cdots 0 \end{bmatrix}, \quad \underline{\mathrm{S}}_X^i = \begin{bmatrix} 0 \cdots I \cdots 0 \end{bmatrix}$$

We substitute the above into (11) and obtain:

$$\begin{aligned}
&\min_{\underline{\mathrm{U}}} \quad \underline{\mathrm{U}}'\underline{\mathrm{B}}'\,\underline{\mathrm{Q}}\,\underline{\mathrm{B}}\underline{\mathrm{U}} + \left(2(X_0{}'\,\underline{\mathrm{A}}' + \underline{\mathrm{d}}'\,\underline{\mathrm{A}}_1')\underline{\mathrm{Q}}\,\underline{\mathrm{B}} + \underline{\mathrm{R}}\right)\underline{\mathrm{U}} + (X_0'\underline{\mathrm{A}}' + \underline{\mathrm{d}}'\,\underline{\mathrm{A}}_1')\,\underline{\mathrm{Q}}\,(\underline{\mathrm{A}}X_0 + \underline{\mathrm{A}}_1\underline{\mathrm{d}}) \\
&\text{s.t.} \\
&\begin{bmatrix} F_u\underline{\mathrm{S}}_u^0 \\ F_u\underline{\mathrm{S}}_u^1 + F_x\underline{\mathrm{S}}_x^1\underline{\mathrm{B}} \\ \vdots \\ F_u\underline{\mathrm{S}}_u^{N-1} + F_x\underline{\mathrm{S}}_x^{N-1}\underline{\mathrm{B}} \end{bmatrix}\underline{\mathrm{U}} \leq \begin{bmatrix} g(0) - F_xX_0 \\ g(1) - F_x\underline{\mathrm{S}}_x^1(\underline{\mathrm{A}}X_0 + \underline{\mathrm{A}}_1\,\underline{\mathrm{d}}) \\ \vdots \\ g(N-1) - F_x\underline{\mathrm{S}}_x^{N-1}(\underline{\mathrm{A}}X_0 + \underline{\mathrm{A}}_1\,\underline{\mathrm{d}}) \end{bmatrix}
\end{aligned} \tag{12}$$

As we show in Appendix A, the QP (12) is in general convex but not strictly convex. The convexity implies it is amenable to an efficient numerical solution (i.e. feasible). The fact it may be non-strict implies that there may be multiple optimal solutions. This allows to incorporate a further secondary optimization for a refined solution if needed – catering to additional secondary objectives.

3. Illustrative Examples and Discussion of Results

In this section we demonstrate the performance of the proposed MPC controller in improving network throughput and reducing congestion under heavy traffic through two simple traffic scenarios. The first scenario focuses on controlling traffic light timings to reduce congestion at a bottleneck intersection, while the second scenario deals with routing traffic through a network consisting of bottleneck links. We conduct several simulations to obtain the performance metrics in terms of the queue lengths at each intersection and the total number of vehicles that successfully leave the network. Note that in our proposed framework both the signalization at the traffic intersections and the turning rate (i.e. routing) are represented by the same control variable (i.e. a time fraction during which vehicles leave a particular class) and thus it is sufficient to show the performance improvements through two separate scenarios focusing on either signalization or routing.

There are few input parameters to the simulation that need to be measured or estimated based on historical data or relevant traffic models. Firstly, the average free flow speed is needed to convert the real network to the model. Our model assumes that in non-congested conditions, it takes one step (a traffic light cycle) for vehicles to move from a queue to the consecutive queue. Thus in our model, the queue is a road segment with length equal to the distance traversed by a vehicle at free low speed during one traffic light cycle. Secondly, the queues' capacity is estimated by the maximum number of average-length vehicles that can be packed into the road segment with minimum distance in between. Thirdly, the flow rates also need to be measured. Generally, flow rate is a function of link density. However, for simplicity and retaining linearity, we assume the flow rate of a particular link is a known constant.

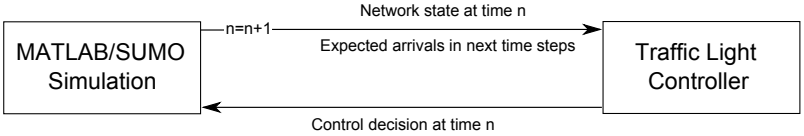


Fig. 4: Simulation design

Two simulations are conducted for each scenario: A MATLAB simulation and a SUMO simulation. Figure 4 demonstrates the general simulation design. At the beginning of each traffic light cycle, the MATLAB or SUMO simulator reports the current state and mean exogenous arrival rate to the controller. The controller will in turn solves the QP problem in (11), returns the current control decision and waits for the state update of the next cycle. The *mean* arrival rates, $a_k(n)$, are deterministic and known but are perturbed by a zero-mean Gaussian noise with a relatively large variance. Thus at each time step a random noise is added to $a_k(n)$. The MATLAB simulation updates the next state based on current state, current control decision and current arrival as described in (7). The MATLAB simulation represents the ideal case where the

controller has a perfect prediction of the future states except for the disturbance of arrivals. Despite the fact of reusing the linear prediction model in (7), the simulation still produces the essential congestion phenomena and basic driver behaviors. On the other hand, the SUMO simulation (SUMO)(website, 2013a) updates the next state through a microscopic traffic simulation model. The SUMO simulation is more realistic than the MATLAB simulation in modeling traffic network evolution. In SUMO simulation, the controller does not have such a good prediction of future state. The accuracy of state prediction is partly dependent on the parameter estimation (i.e. queue capacity, link capacity) which is beyond the scope of this paper.

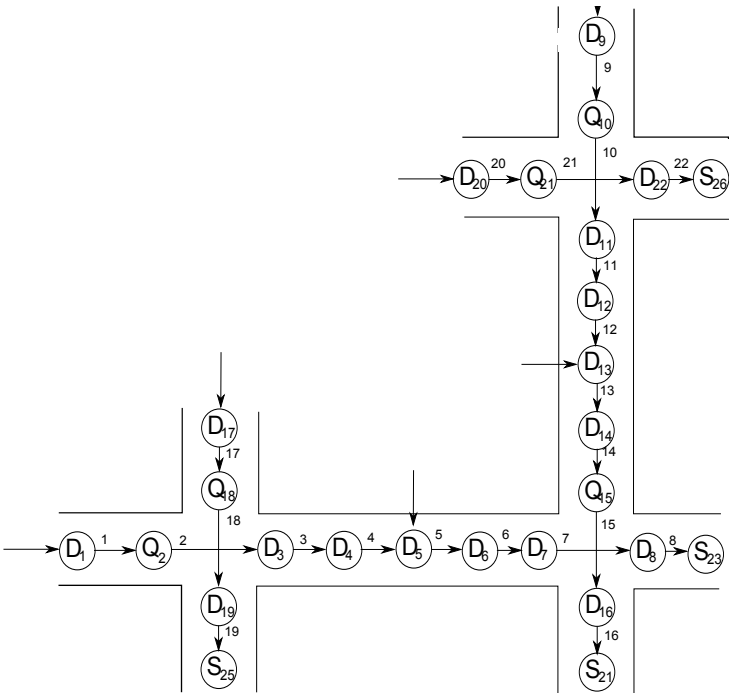


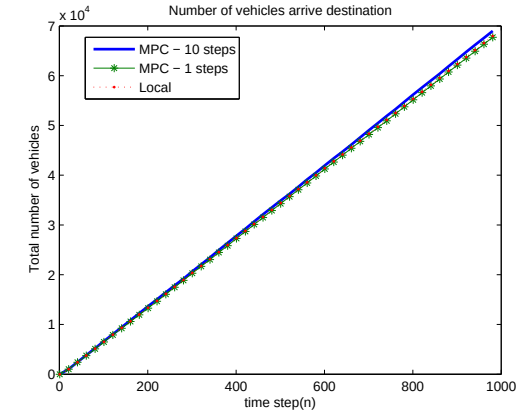
Fig. 5: Symmetric network example. Capacity: $c_1 \dots c_{16} = 270$; $c_{17} \dots c_{22} = 135$. Flow rate: $f_1 \dots f_{16} = 45$; $f_{17} \dots f_{22} = 20$. Arrival rare to D_1, D_9, D_5, D_{13}, D_{17}, D_{20}: are normal random variable with (mean, deviation): (17,8), (17,8), (7,3), (7,3), (13.2,6), (13.2,6) respectively. All negative arrival rates are set to zero.

To ensure fair and comparable results, the same noise is applied for all control schemes and the controllers only have knowledge about the average (deterministic) arrival rates. The QP formulation defined in (11) is solved using CPLEX from MATLAB/TOMLAB tool box (TOMLAB)(website, 2013b) in each step of the simulation to find the optimal value and apply them for the decision variables such as green time split or turning fraction at intersections.

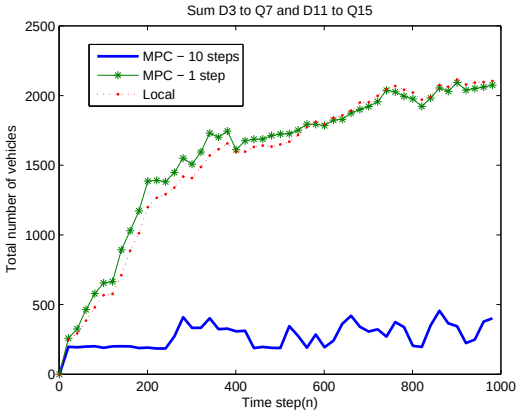
Our main objective in this section is to compare the performance of the proposed MPC approach with other popular approaches under the same conditions rather than evaluate the correctness of the prediction model itself. Towards that end, in each scenario, the performance of the proposed MPC scheme is compared with that of the MPC without look ahead horizon (denoted as MPC-1 step) and a so-called local signal plan. In the MPC-1 step scheme, N is set to 1 in (11) and the controller is unable to see the effect of its current decision to the number of vehicles in the next step. In other words, the MPC-1 step scheme will push traffic forward as much as possible which represents the widely implemented algorithm in the Sydney coordinated adaptive traffic system (SCATS)(website, 2012). On the other hand, the local signal scheme allocates the green time proportionally to the queues at intersections regardless of the downstream queue

conditions. Note that for this local control, in case of spill back (heavy congestion that goes beyond one link), green time may still be allocated for traffic even if the corresponding downstream link is congested, and thus is wasted as no vehicle can move further downstream.

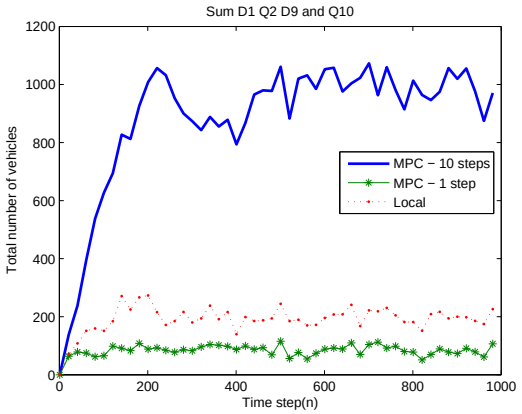
We now describe the detailed traffic scenarios and simulation results obtained for each of the above control schemes.



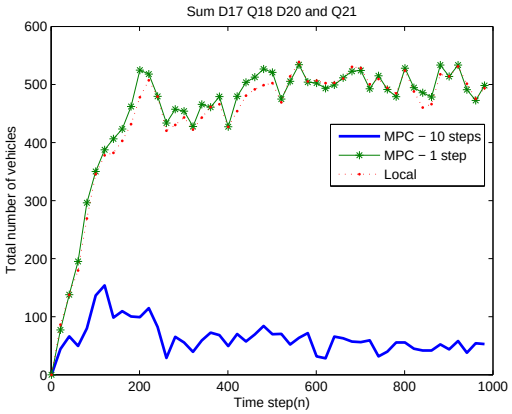
(a) Total vehicles in the network



(b) Queue length at bottleneck intersection



(c) Queue length of main traffic

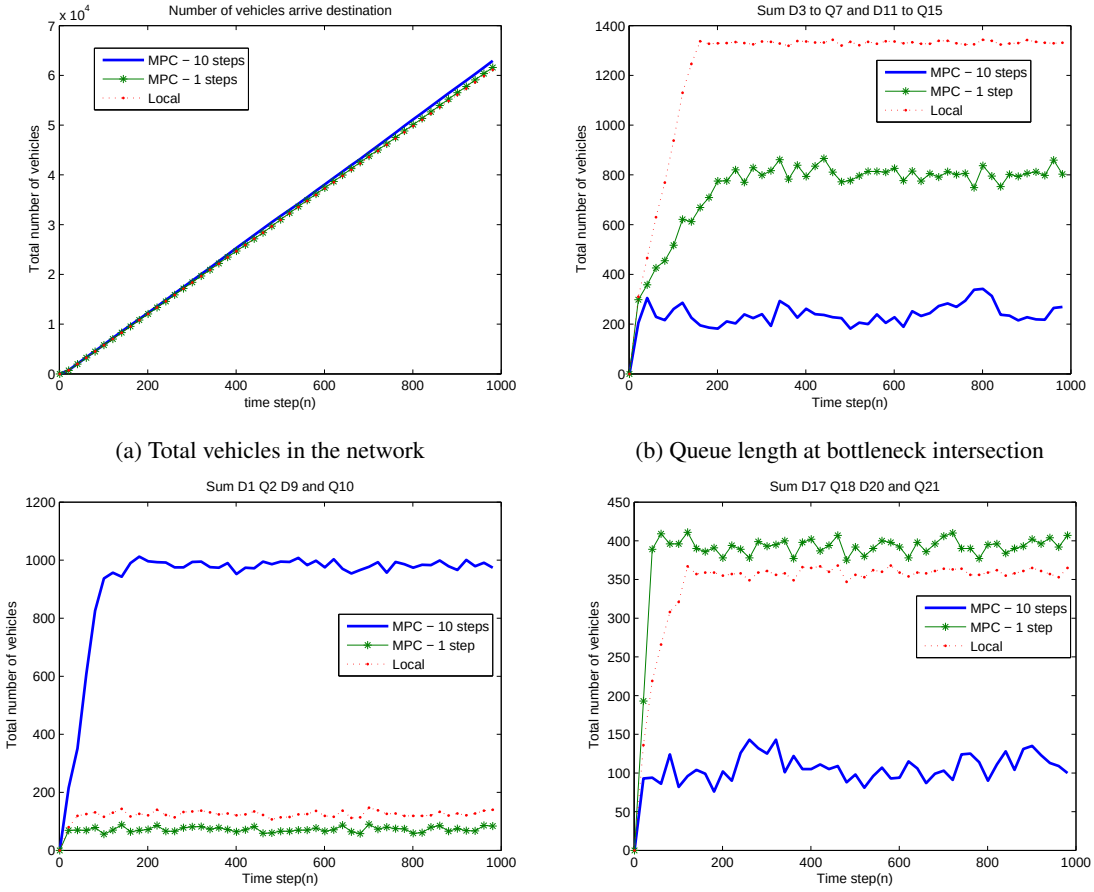


(d) Queue length of side traffic

Fig. 6: Queue lengths for scenario 1 - MATLAB simulation

3.1. Scenario 1: Symmetric Gating Example

This scenario shows the ability of the proposed MPC controller to maximize network throughput and reduce the congestion level within the network. Consider a simple scenario (Figure 5) with fixed routing and three intersections: top intersection, left intersection and the bottom right intersection. The main roads in the West-East and North-South directions consist of $D_1 \dots D_8$ and $D_9 \dots D_{18}$ delays, respectively, accommodate main traffic with larger arrival rates that may cause congestions at the bottom right intersection. The free flow speed is capped at 60 km/h and each queue models vehicles in a 1 km length road segment. Hence, it will take about 1 minute for vehicles to move from one queue to the next consecutive queue. The queue capacity is 135 cars per lane where the average car's length is 5m and the minimum distance between cars in a congested road is 2.5m. The flow rates are measured by a long running SUMO simulation under heavy traffic load with an arbitrary traffic signalization. At the top and left intersections where smaller side traffic flows intersect, the optimal controller is expected to reduce the green time of the incoming traffic



(a) Total vehicles in the network

(b) Queue length at bottleneck intersection

(c) Queue length of main traffic

(d) Queue length of side traffic

Fig. 7: Queue lengths for scenario 1 - SUMO simulation

to the downstream link on the main roads in order to prevent congestion at the bottleneck bottom right intersection, and at the same time reserve more green time for the side traffic on link 18 and 21, respectively.

In this scenario, both links 7 and 15 have maximum outgoing flow rate of 45 cars/cycle at the bottom right intersection. Thus at most 45 vehicles can pass through this bottleneck intersection in a single simulation step (i.e. one traffic light cycle) regardless of the control algorithms. As a result, an optimal control is to allocate green time for link 2 and 10 (Figure 5) so that at most 45 cars may go to bottom right intersection at every step to avoid the building up of congestion inside the network on those links of the main roads between intersections. The rest of the green time allocation is then given to links 18 and 21 enabling side traffic to move. The average number of cars arriving in each cycle for the main traffic flows is 17 cars/cycle at the ingress queues and 7 cars/cycle at the links between intersections. The added random noise has the standard deviation of 8 and 3 respectively in this scenario. These arrival rates are chosen just above the saturation point to increase the chance of congestion and spill back inside the network. The arrival rate into the side traffic flows has the mean of 13.2 cars/cycle with noise having the standard deviation of 6. In the proposed MPC with horizon we expect to see the synchronization between the top and left intersections to cope with the variations in demands and to avoid congestion at the bottom right intersection.

Illustrative sample trajectories of different queue lengths using different control schemes are shown in Figure 6 and Figure 7. Observe that the proposed MPC with time horizon of 10 steps (denoted by MPC-10 step scheme) has slightly better throughput (Figure 6.a and Figure 7.a) while significantly reducing

congestion inside the network (Figure 6.b and 7.b) for both the MATLAB and SUMO simulations. In particular, the total queue length from D_3 to Q_7 and D_{11} to Q_{15} is kept well below the total capacity and almost at a constant level (Figure 6.b and 7.b). It indicates a synchronization between the top and left intersections where the controller only allows a total of about 31 cars passing through the top and left intersections heading to the bottleneck bottom right intersection in any single time step (i.e. traffic cycle). Note that there are about 14 cars arriving to D_5 and D_{13} every cycle to make up the total of 45 cars/cycle which is the capacity of bottom right intersection. The rest of the traffic light cycle are allocated to the side traffic flows. At a result, it fully utilizes links across intersections in the network and minimizes congestion inside network.

On the other hand, the local controller allocates green time based on the ingress queue lengths, Q_2 and Q_{10}, even if the downstream queues, D_3 and D_{11}, are unable to accommodate more traffic. In such a case, by wasting green time for congested flow, it actually creates more congestion at Q_{18} and Q_{21} (Figure 6.d and Figure 7.d). In contrast, the MPC-10 step schemes are able to keep the queues $D17$, $Q18$, $D20$ and $Q21$ stable. At the same time, the MPC-10 step scheme is also able to keep the congestion inside the network low. As a result, there is no spill back in the case of MPC-10 step control scheme.

Furthermore, pushing traffic into D_3 and D_{11} more than what the bottom right intersection can handle, as in case of the local controller and MPC-1, causes more congestion in the inner links that spills back to the top and left intersections. In overall, the unreasonable allocation of green time leads to reduce network throughput and cause unnecessary congestion.

Similar to the local controller, the MPC without horizon (MPC-1 step) also allocates more green time to the main roads (link 2 and 10) resulting in queues building up in the inner links $D3 - Q7$ and $D11 - Q15$. However, in this case, when the spill back reaches D_3 and D_{11} the the MPC-1 step would only allocate green time for side traffic (link 18 and 21). Because there is no wasting resource at any intersection, the MPC-1 step scheme has better performance in term of throughput compared to that of the local controller. However, it still has a high congestion level inside the network (Figure 6.b and 7.b).

This scenario clearly demonstrates the concept of *gating* where the proposed MPC controller *automatically* allows an optimal amount of traffic downstream to avoid congestion inside the network but at the same times also achieves the best possible throughput.

3.2. Scenario 2: Routing Enabled Network

This scenario demonstrates the efficiency of the proposed MPC control scheme for routing purposes. Consider a grid network where vehicles arrive at the top left edge and exit at the bottom right edge (Figure 8). In order to reach the destination, vehicles have to pass through one of the bottleneck links in between. With a non-optimal controller, it is likely that some bottleneck links will experience heavy congestion while others have light traffic. In this scenario, we expect that our MPC controller will find a policy (i.e. turning fractions at intersection) that reasonably balances the traffic load among all the bottleneck links, thus, reduces the overall congestions inside the network. Because the green split times at intersections do not have any impact on the network state, we set those decision variables representing the green times to some fixed values through this scenario.

Vehicles are assumed to completely follow the turning advices from our centralized controllers at top left and middle left intersections. The bottleneck links originate from D_4, D_{17} and D_{23} delays and finish in D_6, D_{19}, D_{25}, respectively. Similar to the previous scenario, the free flow speed is capped at 60 km/h in normal links and 7.2 km/h in bottleneck links. Each queue models vehicles in a 1 km road segment. Hence, it will take about 1 minute for vehicles moving from any queue to the consecutive queue. Cars' length is again set to 5 meter long and the minimum safety distance is 2.5 meters. As a result the queue capacity is about 135 cars for all queues. We only use single lane road in this scenario. The outgoing flow rate of the bottleneck links is 7 cars/cycle which is lower (for example they have lower speed limit) than other links in the network which have the flow rate of 30 cars/cycle. The average arrival rate at the top left edge is chosen to be a less than the aggregate flow rates of all the bottleneck links, i.e. average arrival of 17 cars/cycle with added noise following the normal distribution with zero mean and standard deviation of 10.

We show the sample trace results for different control schemes in Figure 9 and Figure 10. Our proposed MPC controller with a time horizon of 12 steps achieves better throughput and congestion control compared

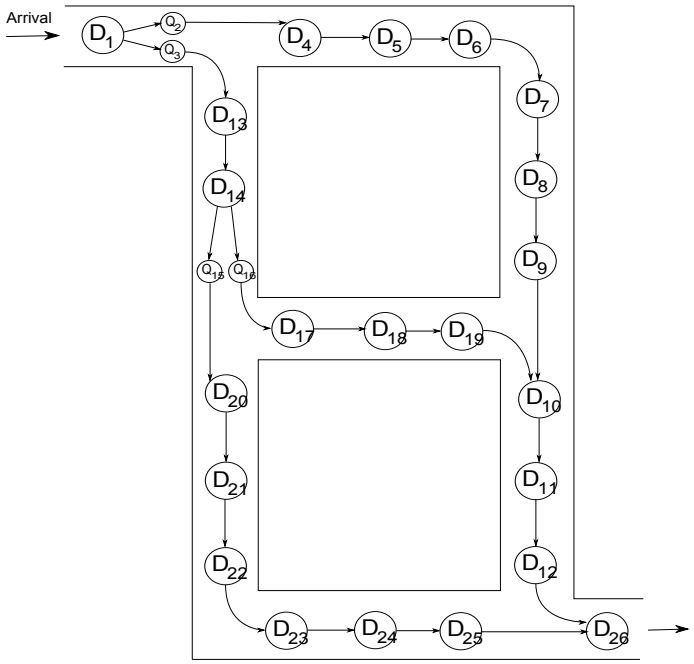


Fig. 8: Routing enable network example. Capacity: All capacities are 135. Flow rates between D_4 and D_5, D_{17} and D_{18}, D_{23} and D_{24} are 7; All other flow rates are 30. Arrival rare is normal random variable mean 17 deviation 10.

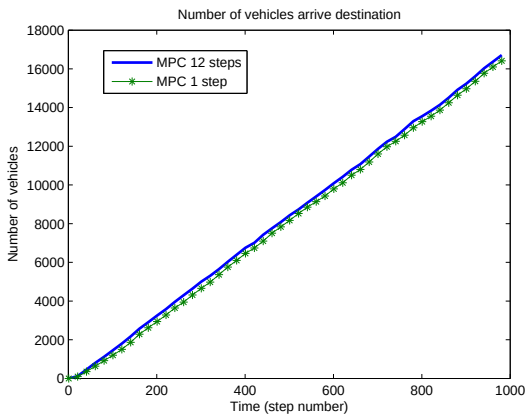
to that of other schemes. Observe that in both simulations, our proposed MPC scheme maintains around 14 cars at the top and middle bottleneck links where the maximum queue is 270 cars. The rest goes to the bottom bottleneck link. The proposed control scheme significantly reduces congestions thus preventing spillback inside the network.

The MPC-1 step scheme on the other hand does not realize the congestion at bottleneck links until it spills back to the signalized intersections. Therefore, drivers are advised to turn at random under these controllers resulting in some bottleneck receiving more traffic than the others. This causes congestion to build up. In the MATLAB simulation, the effect of spillback is ignored for simplicity so vehicles are still able to move at normal rate in heavy congestion at intersections. Hence, the throughput of MPC-1 is very close to the throughput of MPC-12 which does not have any spillback (Figure 9.a). However, it is not the case in the SUMO simulation. When spillback reaches Q2 and Q3, vehicles are hardly able to move to D13 resulting in a light traffic at the middle and bottom bottleneck links. Consequently, the throughput of MPC-1 is significantly lower than MPC-12 (Figure 10.a)

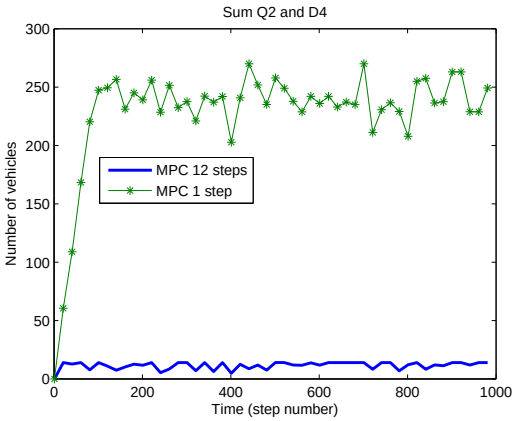
4. Conclusion

We have developed in this paper a general model predictive control framework for centralized traffic signal and route guidance systems aiming to minimize network congestion. The proposed controller mechanism is based on a finite rolling-horizon (model predictive) control scheme where both the non-zero travel time on the link and possible spill-back due to congestion are taken into account. We have provided the detailed mathematical description of the framework together with the derivation of the linear model with state dependent constraints. As a result the linearity and tractability are some of the key features of our proposed coordinated central control of large area urban networks.

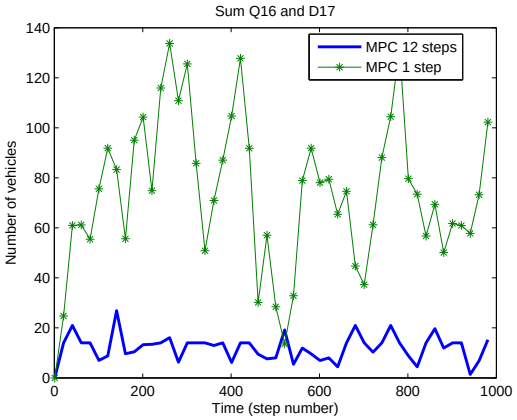
We compared the control algorithm to a simple local control signal plan and similar model predictive control scheme without look ahead. The latter is resembles current control mechanism (SCATS system) used



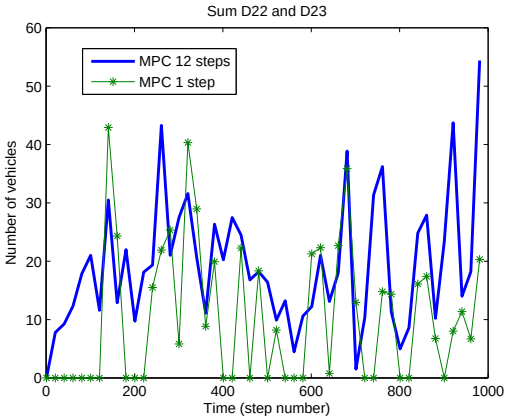
(a) Number of vehicles that arrive destination



(b) Queue at the top bottleneck link



(c) Queue at the middle bottleneck link



(d) Queue at the bottom bottleneck link

Fig. 9: Queue lengths for scenario 2 - MATLAB simulation

to coordinate traffic in various urban cities around the world. We have showed via numerical experiments that using the proposed control scheme may reduce the congestion inside the network significantly while still achieving better throughput compared to that of other conventional control schemes that we studied.

In future research we intend to incorporate state estimation into the model as well as carry out large-scale experiments on networks with hundreds of nodes. This is achievable due to the linear-quadratic structure of our framework.

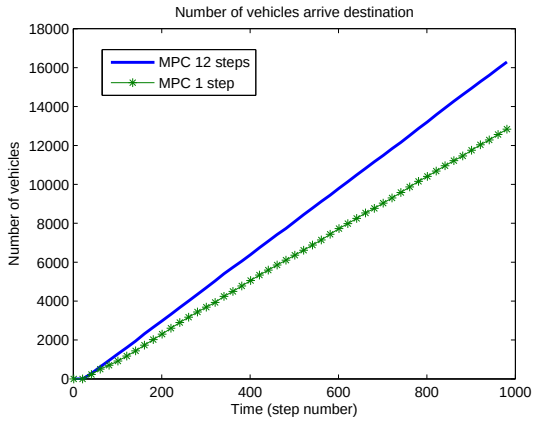
Acknowledgements

This work was supported by the Australian Research Council (ARC) Future Fellowships grant FT120100723.

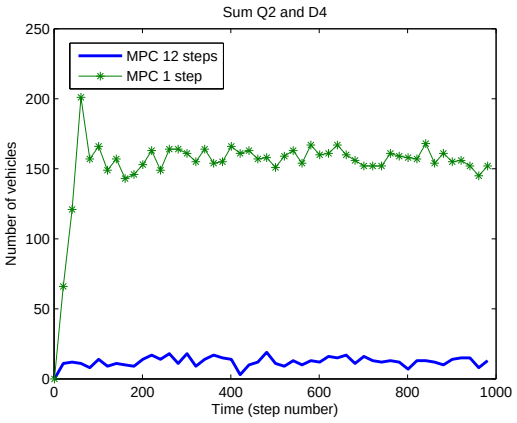
Appendix A. Convexity

Given that Q is all 1 matrix, we now prove that the minimized objective function $f(\underline{\mathrm{U}}) = g(\underline{\mathrm{U}}) + h(\underline{\mathrm{U}})$ in (12) where $g(\underline{\mathrm{U}}) = \underline{\mathrm{U}}^{'}\underline{\mathrm{B}}'\,\underline{\mathrm{Q}}\,\underline{\mathrm{B}}\underline{\mathrm{U}}$ and $h(\underline{\mathrm{U}}) = \left(2(X_0{}'\,\underline{\mathrm{A}}' + \underline{\mathrm{d}}'\,\underline{\mathrm{A}}_1')\underline{\mathrm{Q}}\,\underline{\mathrm{B}} + \underline{\mathrm{R}}\right)\underline{\mathrm{U}} + (X_0'\underline{\mathrm{A}}' + \underline{\mathrm{d}}'\,\underline{\mathrm{A}}_1')\,\underline{\mathrm{Q}}\,(\underline{\mathrm{A}}X_0 + \underline{\mathrm{A}}_1\underline{\mathrm{d}})$ is convex but not strictly convex. Therefore, we are likely to have more than one optimal solution.

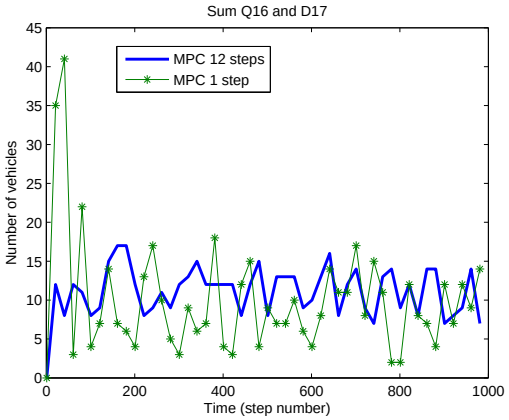
Let $\underline{\mathrm{Z}}$ is a vector of dimension $N \times K_{D,R,Q}$ (for readability, we denote $K_{D,R,Q}$ by K until the end of this section) where $\underline{\mathrm{Z}} = \underline{\mathrm{B}}\,\underline{\mathrm{U}}$ or $z_i = \sum_{j=1}^{N\times J} b_{i,j}u_j$. Note that z, b are elements of $\underline{\mathrm{Z}}$, $\underline{\mathrm{B}}$ respectively and $\underline{\mathrm{B}}$ is a lower



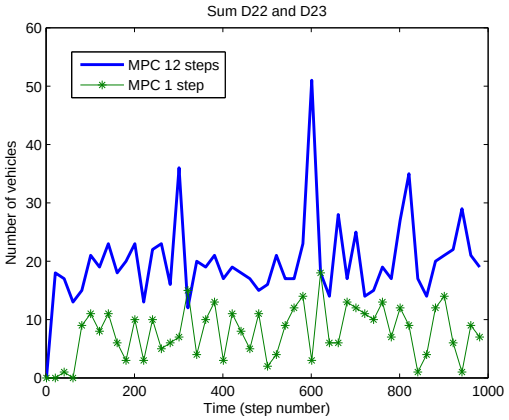
(a) Number of vehicles that arrive destination



(b) Queue at the top bottleneck link



(c) Queue at the middle bottleneck link



(d) Queue at the bottom bottleneck link

Fig. 10: Queue lengths for scenario 2 - SUMO simulation

block triangular matrix of dimension $N \cdot K \times N \cdot J$ which is defined above. Hence:

$$\begin{aligned} g(\underline{U}) = \underline{Z}'\,\underline{Q}\,\underline{Z} &= \sum_{k=1}^{N}\Big(\sum_{i=(k-1)*K+1}^{(k-1)*K+K} z_i\Big)^2 \\ &= \sum_{k=1}^{N}\Big(\sum_{i=(k-1)*K+1}^{(k-1)*K+K}\big(\sum_{j=1}^{N\times J} b_{i,j}u_j\big)\Big)^2 \\ &= \sum_{k=1}^{N}\Big(\sum_{j=1}^{N\times J}\Big(\sum_{i=(k-1)*K+1}^{(k-1)*K+K} b_{i,j}\Big)u_j\Big)^2 \end{aligned} \tag{A.1}$$

Equations (A.1) shows that $g(\underline{U})$ is non negative for arbitrary $\underline{U}$ and $\underline{B}$. Hence, $\underline{B}'\underline{Q}\,\underline{B}$ is positive semi-definite for arbitrary $\underline{B}$. Thus, $g(\underline{U})$ is convex. The second term, $h(\underline{U})$ is also convex. So $f(\underline{U}) = g(\underline{U}) + h(\underline{U})$ is convex.

To investigate the strictly convex property, let us consider the linear mapping $y = \tilde{B}\underline{U}$ where $\tilde{B}$ has dimension $(N+1) \times N \cdot J$. If $1 \le i \le N$,

$$\tilde{b}_{i,j} = \Big(\sum_{i=(k-1)*K+1}^{(k-1)*K+K} b_{i,j}\Big).$$

If $i = N + 1$,

$$\tilde{b}_{i,j} = [2({X_0}' \underline{\mathrm{A}}' + \underline{\mathrm{d}}' \underline{\mathrm{A}}_1')\underline{\mathrm{Q}}\,\underline{\mathrm{B}} + \underline{\mathrm{R}}]_j.$$

Because $(N + 1) << N \cdot J$, there are vectors $\tilde{U}$ such that $y = \tilde{B}\underline{\mathrm{U}} = \tilde{B}\tilde{U}$. It is straightforward to verify that if $\underline{\mathrm{U}}$ is a solution of the QP program, so does $\tilde{U}$. Hence, $f(\underline{\mathrm{U}})$ is not strictly convex.

Appendix B. Further Detailed Notation

The followings are definitions of matrices, vectors in (10)

- H is a $M \times 2M$ matrix with element (i, j) set to 1 if $j = 1 + (i - 1) * 2$ or $j = 1 + (i - 1) * 2 + 1$ and 0 otherwise.
- S_{WEt} is a $M \times 2M$ matrix with element (i, j) set to 1 if $j = 1 + (i - 1) * 2$ and 0 otherwise.
- S_{NSt} is a $M \times 2M$ matrix with element (i, j) set to 1 if $j = 1 + (i - 1) * 2 + 1$ and 0 otherwise.
- S_{WEm} is a $(s \times M)$, $s = \sum_{i=1}^{M} |\mathcal{WE}(i)|$, block diagonal matrix where the diagonal elements are $\mathbf{1}$ vectors of dimensions $(|\mathcal{WE}(i)|)$, $i = 1, \ldots, M$
- S_{NSm} is a $(s \times M)$, $s = \sum_{i=1}^{M} |\mathcal{NS}(i)|$, block diagonal matrix where the diagonal elements are $\mathbf{1}$ vectors of dimensions $(|\mathcal{NS}(i)|)$, $i = 1, \ldots, M$
- We order $\{j_i \in \mathcal{WE}(i)\}$ such that $j_1 < j_2 < \ldots < j_{|\mathcal{WE}(i)|}$
  S^i_{WEM} is a $(|\mathcal{WE}(i)| \times L)$ matrix with element (m, n) set to 1 if $n = j_m$ and 0 otherwise.
  $S_{WEM} = [S^1_{WEM} \ldots S^M_{WEM}]'$.
- We order $\{j_i \in \mathcal{NS}(i)\}$ such that $j_1 < j_2 < \ldots < j_{|\mathcal{NS}(i)|}$
  S^i_{NSM} is a $(|\mathcal{NS}(i)| \times L)$ matrix with element (m, n) set to 1 if $n = j_m$ and 0 otherwise.
  $S_{NSM} = [S^1_{NSM} \ldots S^M_{NSM}]'$.
- S_{WEc} is a $(s \times M)$, $s = \sum_{i=1}^{M} C_{WE}(i)$, block diagonal matrix where the diagonal elements are $\mathbf{1}$ vectors of dimensions $C_{WE}(i)$, $i = 1, \ldots, M$
- S_{NSc} is a $(s \times M)$, $s = \sum_{i=1}^{M} C_{NS}(i)$, block diagonal matrix where the diagonal elements are $\mathbf{1}$ vectors of dimensions $C_{NS}(i)$, $i = 1, \ldots, M$
- S^i_{WEC} is a $(C_{WE}(i) \times L)$ matrix with element (i, j) set to 1 if $j \in C^i_{\mathcal{WE}}$ and 0 otherwise.
  $S_{WEC} = [S^1_{WEC} \ldots S^M_{WEC}]'$.
- S^i_{NSC} is a $(C_{NS}(i) \times L)$ matrix with element (i, j) set to 1 if $j \in C^i_{\mathcal{NS}}$ and 0 otherwise.
  $S_{NSC} = [S^1_{NSC} \ldots S^M_{NSC}]'$.

It is straightforward to verify that (10) is matrix version of conditions (1)-(6)

References

Aboudolas, K., Papageorgiou, M., & Kosmatopoulos, E. 2009. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, **17**(2), 163 – 174.

Aboudolas, K., Papageorgiou, M., Kouvelas, A., & Kosmatopoulos, E. 2010. A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, **18**(5), 680 – 694.

Bellemans, T., De Schutter, B., & De Moor, B. 2002. Model predictive control with repeated model fitting for ramp metering. *In: Proceedings of the Fifth IEEE Intelligent Transportation Systems Conference*.

Diakaki, C., Papageorgiou, M., & McLean, T. 1999. Application and evaluation of the integrated traffic-responsive urban corridor control strategy IN-TUC in Glasgow. *In: Preprint CD-ROM of the 78th Annual Meeting of the Transportation Research Board*. paper no. 990310.

Diakaki, C., Dinopoulou V. Aboudolas K. Papageorgiou M. Ben-Shabat E. Seider E. Leibov A. 2003. Extensions and New Applications of the Traffic-Responsive Urban Control Strategy: Coordinated Signal Control for Urban Networks. *Transportation Research Record*, 202–211.

Diakaki, C., Papageorgiou M. Aboudolas K. 2002. A multivariable regulator approach to traffic-responsive network- Wide signal control. *Control Engineering Practice*, **10**(2), 183–195.

Farges, J.L., Henry, J.J., & Tufal, J. 1983. The PRODYN real-time traffic algorithm. *Pages 307–312 of: Proceedings of the 4th IFAC Symposium on Transportation Systems*.

Gartner, N.H. 1983. OPAC: a demand-responsive strategy for traffic signal control. *Transportation Research Record*, **906**, 75 – 84.

Gazis, D.C., & Potts, R.B. 1963. The oversaturated intersection. *Pages 221–237 of: Proceedings of the Second International Symposium on Traffic Theory*.

H. M. Abdul Aziz, Satish V. Ukkusuri. 2012. Unified Framework for Dynamic Traffic Assignment and Signal Control with Cell Transmission Model. *Transportation Research Record: Journal of the Transportation Research Board*, 73 –84.

Hegyi, A., De Schutter, B., & Hellendoorn, J. 2003. MPC-based optimal coordination of variable speed limits to suppress shock waves in freeway traffic. *In: Proceedings of the American Control Conference*.

Hunt, P.B., Robertson D.I. Bretherton R.D. 1982. The SCOOT on-line traffic signal optimisation technique (Glasgow). *Traffic Engineering & Control*, **23**(4), 190 – 192.

Lin, Shu, De Schutter, B., Xi, Yugeng, & Hellendoorn, H. 2011. Fast Model Predictive Control for Urban Road Networks via MILP. *Intelligent Transportation Systems, IEEE Transactions on*, **12**(3), 846 –856.

Lowrie, P.R. 1982. SCATS: the Sydney co-ordinated adaptive traffic system Principles, methodology, algorithms. *Pages 67 – 70 of: Proceedings of the IEE International Conference on Road Traffic Signalling*.

Mirchandani, P., & Head, L. 1998. RHODES a realtime traffic signal control system: architecture, algorithms, and analysis. *Pages 307–312 of: TRISTAN III (Triennial Symposium on Transportation Analysis), vol. 2*.

Papageorgiou, M., Diakaki C. Dinopoulou V. Kotsialos A. Wang-Y. 2003. Review of road traffic control strategies. *Proceedings of the IEEE*, 20432067.

Porche, I., Sampath, M., Sengupta, R., Chen, Y.-L., & Lafortune, S. 1996 (sep). A decentralized scheme for real-time optimization of traffic signals. *Pages 582 – 589 of: Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*.

Tettamanti, T., Varga, I., Kulcsar, B., & Bokor, J. 2008 (june). Model predictive control in urban traffic network management. *Pages 1538 –1543 of: Control and Automation, 2008 16th Mediterranean Conference on*.

Tettamanti, T., Varga, I., & Péni, T. 2010. MPC in urban traffic management. *In:* Zheng, Tao (ed), *Model Predictive Control*. SciYo.

Tettamanti, Tamás, & Varga, István. 2010. Distributed traffic control system based on model predictive control. *Periodica Polytechnica ser. Civil. Eng.*, **54**(1), 3–9.

van Leeuwaarden, Johan S. H., Lefeber, Erjen, Nazarathy, Yoni, & Rooda, Jacobus E. 2010. Model predictive control for the acquisition queue and related queueing networks. *Pages 193–200 of: Proceedings of the 5th International Conference on Queueing Theory and Network Applications*. QTNA '10. New York, NY, USA: ACM.

Waller, S. Travis, & Ziliaskopoulos, Athanasios K. 2006. A chance-constrained based stochastic dynamic traffic assignment model: Analysis, formulation and solution algorithms. *Transportation Research Part C: Emerging Technologies*, **14**(6), 418 – 427.

website, SCATS. 2012. *http://www.scats.com.au/*, aug.

website, SUMO. 2013a. *http://sumo.sourceforge.net/*, feb.

website, TOMLAB. 2013b. *http://tomopt.com/tomlab/*, feb.