

Technical report 13-013

Optimal Routing in Freeway Networks via Sequential Linear Programming*

Z. Cong, B. De Schutter, and R. Babuška

To cite this work, please refer to the published version:

Z. Cong, B. De Schutter, and R. Babuška, “Optimal routing in freeway networks via sequential linear programming,” *Proceedings of the 10th IEEE International Conference on Networking, Sensing and Control*, Paris, France, 6 pp., Apr. 2013. Paper FrB01.5.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: <https://www.dcsc.tudelft.nl>

* This report can also be downloaded via <https://dpub.eu/13-013>

Optimal Routing in Freeway Networks via Sequential Linear Programming

Zhe Cong, Bart De Schutter, and Robert Babuška

Abstract—Based on the Ant Colony Optimization (ACO) algorithm, we previously developed an optimization method to solve the dynamic traffic routing problem in freeway networks, called Ant Colony Routing (ACR). This method uses virtual ants to search appropriate routes in a virtual ant network, and accordingly distributes the vehicles over the corresponding traffic network sharing the same topology with the ant network. By using Model Predictive Control (MPC), we can iteratively apply ACR at each control step to generate a control signal — i.e. splitting rates at each node in the traffic network. Motivated by the MPC framework with ACR, we show in this paper that sequential linear programming (SLP) can be used as optimization method for solving the dynamic traffic routing problem in some specific cases, resulting a lower computation time while achieving a similar performance as the ACR algorithm.

I. INTRODUCTION

A freeway network that connects cities usually includes numerous junctions and multiple-lane roads. With unpredictable disturbances, such as incidents, demands, and weather, such freeway networks can become highly non-linear and time-variant systems. Therefore, dynamic traffic control methods are required for managing the traffic flows so that safety on the roads is guaranteed and no or less traffic congestion will occur.

Dynamic traffic routing is one of such important traffic control methods. In particular, it continuously measures the state of the traffic network, and accordingly diverts vehicles at each junction to different outgoing links such that a global or user objective is optimized, e.g., the total travel time or the emission of CO₂. There exists a broad literature on this topic [1], [2], [3], [4], [5]. In [1], Fu has introduced an adaptive routing algorithm by using in-vehicle route guidance systems. Based on real-time information of the current vehicle positions, the system minimizes the expected travel time to the destination in the network, and computes the fastest paths for drivers. The route choice is updated every time when the vehicle enters a new link. In [2], Kim *et al.* also use real-time information technology. They consider a stochastic shortest path problem on a road network composed of links having non-stationary travel times, and the links are assigned to either a congested state or an uncongested state for determining the travel time distribution. In [3], Ericsson *et al.* have developed a navigation system to focus on the optimization of route choice based on the total fuel consumption and the emission of CO₂ instead of

the traditional optimization of the travel time or the travel distance. In [4], Papageorgiou has developed a framework for the dynamic traffic routing problem. Based on a dynamic traffic model, he uses two different methods — optimal control and feedback control — to influence the drivers' behavior w.r.t. the system optimum or to the user equilibrium [6]. Based on that framework, Messmer [5] applied optimal control in closed loop, where the feedback control problem is repeatedly solved by applying a non-linear optimization procedure using gradient-based search over a future time horizon. Additional information about dynamic traffic routing can be found in [7].

Some of the work mentioned above, i.e., [1], [2], [3], does not take into account the availability of future information, and therefore may generate suboptimal solutions, while the other papers, i.e., [4], [5], use numerical non-linear optimization methods, which involve a high computational effort. Motivated by the shortcomings of the existing routing algorithms, we have created an ant-based routing algorithm for freeway networks, called Ant Colony Routing (ACR) [8]. The ACR algorithm aims at minimizing a global objective function (total travel time in our case), and at the same time limits the numbers of vehicles on each link in the traffic network in order to avoid traffic congestion, as well as for environmental reasons. In ACR, the traffic information is translated into link costs in a virtual ant network, and a so-called stench pheromone is used to push ants away when too many ants are crowded on the optimal routes. In that way, some ants start to search alternative routes in the network. When ants finish searching, the number of ants that has traveled on each link is used to determine the control signal — i.e. the splitting rates at each node in the traffic network. Furthermore, through Model Predictive Control (MPC) [9], [10], we can include a prediction of the future traffic states into ACR for optimization, and iteratively apply the resulting splitting rates at each control step¹.

As shown in Figure 1, there are two loops in the entire MPC traffic control system. Inside the MPC controller, there is a prediction-optimization loop, running at each control step. In this loop, a traffic model is used to predict the future traffic states over a prediction horizon, and ACR is then applied as the optimization method to optimize the future traffic flows in the traffic network. At the end, we obtain the splitting rates, and feed them back to the traffic prediction model as the control input to restart the next round

The authors are with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, email: {z.cong,b.deschutter,r.babuska}@tudelft.nl

¹The translation of the splitting rates into routing instructions (via dynamic route information panels or on-board route guidance) is a task delegated to lower level controller [11].

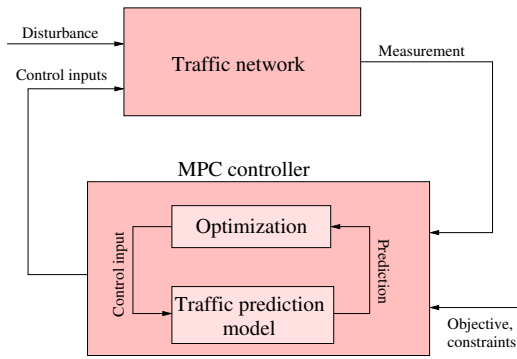


Fig. 1. Schematic representation of MPC

of prediction-optimization. This inner loop keeps recycling until a stopping criterion is satisfied, (e.g., the changes in the optimization variables or in the objective function are below a given threshold or the maximum number of iteration is reached). Outside the MPC controller, there is a feedback control loop. The resulting control signal determined by ACR is implemented in the real traffic network at each control step, and subsequently both the control horizon and the prediction horizon are shifted forward one sample time step. At the next control step, new traffic states are measured for the MPC controller and the whole procedure is repeated.

In this paper, we present a sequential linear programming (SLP) optimization method to solve the dynamic traffic routing problem under the same MPC framework. In particular, we show that ACR can be recast as a linear programming (LP) problem if

- **Condition (a)**²: the link cost in the virtual ant network is constant at each control step;
- **Condition (b)**: the stench pheromone function in ACR is an affine or convex piecewise affine function.

Note that Condition (a) approximately holds if the control time step is not too large. Moreover, Condition (b) can usually be satisfied too as the stench function is constructed by the control designer. Therefore, in the prediction-optimization loop at each control step, the optimization problem will be formulated as a sequence of linear subproblems (see Section III), and is solved iteratively by the LP algorithm in each iteration of the loop. Compared with ACR, SLP can achieve a similar performance, while consuming less computation time. However, for non-linear, non-convex stench functions (e.g., staircase-like stench functions), ACR still applies while SLP cannot be used.

The rest of this paper is structured as follows. Section II recapitulates the ACR algorithm. Next, in Section III, we use SLP to formulate the dynamic traffic routing problem. Then, we compare the simulation results of a case study in the Singapore Expressway Network by using SLP and ACR in Section IV. Section V concludes the paper.

²Condition (a) is also required for the regular implementation of ACR, but for the case where Condition (a) does not apply, one can apply the so-called fully-dynamic extension of ACR [12] (possibly at the cost of increased CPU time).

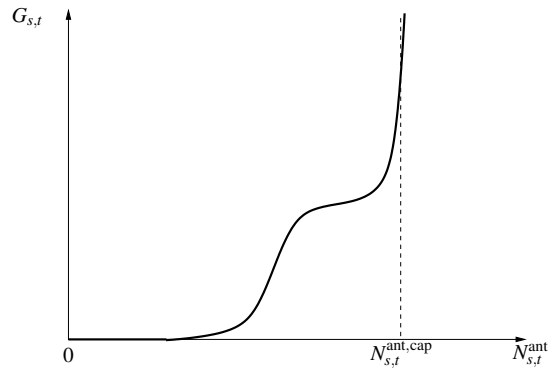


Fig. 2. General description of stench function $G_{s,t}$

II. THE ACR ALGORITHM

The ACR algorithm is an ant-based routing method for determining the optimal flow distribution in a freeway network, originating from the Ant Colony Optimization (ACO) algorithms [13], [14]. ACR has the same function as ACO, which uses virtual ants to find the best route in a network. However, unlike ACO, ACR can push the ants away when the best route is crowded, and force ants to search for alternative routes in the network. In ACR, a virtual ant network consists of a set of nodes and a set of links that connect the nodes. A particular route r is a concatenation of links (s,t) , which are pairs of nodes s and t connected by the given link, from an origin o to a destination d . ACR aims at finding a set of best routes $R_{o,d}^*$ from each origin o to each destination d that satisfy the requirements of the optimization problem, instead of just finding the optimal route $r_{o,d}^*$. There are two loops in ACR: an inner loop in which ants move from one node to an adjacent node in every iteration, and an outer loop in which ants repeat searching routes from their origin to their destination in every iteration.

More specifically, a given ant c moves from its current node s to node t based on a probability $p(t|s)$ at each iteration of the inner loop. The probability $p(t|s)$ depends on the pheromone level on the link (s,t) and all the pheromone levels on the links that are connected to node s :

$$p(t|s) = \frac{\max(\tau_{\min}, \tau_{s,t})^\alpha}{\sum_{\tilde{t} \in \mathcal{N}_s} \max(\tau_{\min}, \tau_{s,\tilde{t}})^\alpha}, \quad \forall t \in \mathcal{N}_s, \quad (1)$$

with $\tau_{s,t}$ the current pheromone level on link (s,t) , τ_{\min} the lower bound for the pheromone levels, and the parameter $\alpha \geq 1$. The feasible neighborhood \mathcal{N}_s of node s is the set of nodes that are connected to node s but have not yet been visited by the given ant c . After selecting the next node t based on (1), ant c adds link (s,t) to its route r_c , and chooses an adjacent node connected to node t in the next iteration. It repeatedly applies (1) to construct a route r_c until it reaches the destination node. All ants construct the routes in parallel, and each route r_c is independently evaluated by a fitness function F after searching is complete. This fitness function F assigns a positive value to each route r_c , which is used to calculate the *regular* pheromone $\Delta\tau_{s,t}(r_c)$ deposited by ant c

on link (s, t) :

$$\Delta\tau_{s,t}(r_c) = \begin{cases} F(r_c) , & \text{if } (s, t) \in r_c \\ 0 , & \text{otherwise.} \end{cases} \quad (2)$$

On the other hand, the *stench* pheromone is calculated by a function $G_{s,t}$ based on the number of ants $N_{s,t}^{\text{ant}}$ that have traveled on the link (s, t) . The general behavior of $G_{s,t}$ is described in Figure 2 with the following properties:

- 1) $G_{s,t}$ has a low value for a small number of ants, while it has a high value for a large number of ants;
- 2) If there are no ants visiting link (s, t) , i.e. $N_{s,t}^{\text{ant}} = 0$, then $G_{s,t}(0) = 0$ because no stench pheromone has to be deposited;
- 3) As $N_{s,t}^{\text{ant}}$ increases, the value of $G_{s,t}(N_{s,t}^{\text{ant}})$ will be monotonically non-decreasing every time $N_{s,t}^{\text{ant}}$ reaches one of the intermediate threshold levels corresponding to e.g. sensitive zones such as schools, hospitals, and residential areas;
- 4) When $N_{s,t}^{\text{ant}}$ reaches the capacity of link $N_{s,t}^{\text{ant, cap}}$, the value of $G_{s,t}(N_{s,t}^{\text{ant}})$ steeply rises.

At the end of the inner loop, the total pheromone level on link (s, t) is updated by both the regular pheromone and the stench pheromone:

$$\tau_{s,t} \leftarrow (1 - \sigma)\tau_{s,t} + \sum_{r_c \in \mathcal{R}_{\text{upd}}} \Delta\tau_{s,t}(r_c) - G_{s,t}(N_{s,t}^{\text{ant}}), \quad (3)$$

with $\sigma \in (0, 1)$ the evaporation rate and \mathcal{R}_{upd} the set of routes that are eligible for the pheromone update.

All the ants repeat searching the routes based on the new pheromone levels, and the pheromone levels are updated by applying (3) at each iteration of the outer loop. When the change in the number of ants on each link from one iteration of the outer loop to the next is below a threshold, or when the maximum number of iterations of the outer loop has been reached, the entire algorithm terminates. The output of the algorithm is the number of ants $N_{s,t}^{\text{ant}}$ in the last iteration of the outer loop, which is subsequently used to calculate the splitting rates β_m in the traffic network,

$$\beta_{\ell(s,t)} = \frac{N_{s,t}^{\text{ant}}}{\sum_{\bar{i} \in \mathcal{N}_s} N_{s,\bar{i}}^{\text{ant}}}, \quad (4)$$

with a relationship ℓ between the link (s, t) in the virtual ant network and the link m in the traffic network defined as $m = \ell(s, t)$.

III. DYNAMIC TRAFFIC ROUTING BY SLP

In this section, we present SLP to solve the same optimization problem as ACR in the dynamic traffic routing problem. Compared to ACR, SLP is a much faster method as it can use fast LP solvers. We first present the basic SLP algorithm without going into the details of the specific implementation of dynamic traffic routing:

for $\kappa = 0, 1, 2, \dots, K$ **do**

Solve the linear sub-problem

$$\min_{x_\kappa} J_\kappa(x_\kappa) = c_\kappa^T x_\kappa$$

subject to $A_\kappa x_\kappa = b_\kappa$,

$$C_\kappa x_\kappa \leq d_\kappa, \quad (5)$$

to obtain $u_{\kappa+1}$;

Update the optimization variables by using a prediction model $x_{\kappa+1} = f(x_\kappa, u_{\kappa+1})$

Stop if converged;

end for

In this algorithm, K denotes the maximum number of the iterations, x_κ denotes the optimization variables³ for the κ th sub-problem, u_κ denotes the control variables, $J_\kappa(x_\kappa)$ denotes the linearized objective function, c_κ , A_κ , and C_κ are a vector and matrices with constant values, and f is the traffic prediction model. The sub-problem in (5) is an LP problem at each sequence, and we will specify how to formulate it next. For illustration purpose, we omit the subscript κ from now on.

In this paper, we assume that the travel time spent (TTS) acts as the primary objective⁴ for the dynamic routing problem. However, in ACR, there is no explicit objective function. Instead, as mentioned in Section II, we interpret the TTS as the link cost, and use a group of ants to evaluate each route in the network, in such a way that a route with lower link cost will attract more ants. Such collective behavior finally leads to a global result. In the SLP approach, we share the same goal to minimize the TTS, and we will repeatedly solve a steady-state static problem, with optimization variables $q_{m,d}$, where $q_{m,d}$ denotes the traffic flow with destination d on link m . The objective function is defined as follows:

$$\min J_{\text{TTS}} = \min \sum_{m \in \mathcal{M}} \sum_{d \in \mathcal{D}} q_{m,d} \cdot T \cdot N_p \cdot \varphi_m \quad (6)$$

where \mathcal{M} denotes the set of all links in the network, T denotes the simulation time interval, N_p denotes the prediction horizon, and φ_m denotes the link cost. In (6), the part $\sum_{d \in \mathcal{D}} q_{m,d} \cdot T$ expresses the number of vehicles on link m in each simulation time interval of length T , and the link cost φ_m expresses the travel time on link m . Therefore, $\sum_{d \in \mathcal{D}} q_{m,d} \cdot T \cdot N_p \cdot \varphi_m$ corresponds to the total travel time on link m over the prediction period. It is easy to verify that if φ_m is constant (Condition (a) in Section I), (6) is a linear formulation. We can easily obtain φ_m by calculating the average travel time during a prediction period $t = kT$ to $t = (k + N_p)T$, with k the current simulation step:

$$\varphi_m = \frac{1}{N_p} \sum_{l=k}^{k+N_p-1} t_m(l)$$

where $t_m(l)$ denotes the travel time on link m at simulation step l . This minimization problem is constrained by:

$$\sum_{m' \in \mathcal{O}(o)} q_{m',d} = d_{o,d}, \quad \forall o \in \mathcal{O}, \quad \forall d \in \mathcal{D} \quad (7)$$

³These variables include the state variables as well as some additional variables (see below for more details).

⁴Note that we can deal with other objective functions as well, as long as they (approximately) satisfy Condition (a).

$$\sum_{m \in I(n)} q_{m,d} = \sum_{\tilde{m} \in O(n)} q_{\tilde{m},d}, \quad \forall n \in \mathcal{N}, \quad \forall d \in \mathcal{D} \quad (8)$$

$$\sum_{d \in \mathcal{D}} q_{m,d} \leq q_m^{\text{cap}}, \quad \forall m \in \mathcal{M} \quad (9)$$

with \mathcal{O} the set of all origin nodes in the network, \mathcal{D} the set of all destination nodes in the network, \mathcal{N} the set of all intermediate nodes in the network, $d_{o,d}$ the inflow with destination d in origin o , q_m^{cap} the capacity of link m , and where I and O respectively denote the sets of incoming and outgoing links. Note that (7)–(9) are linear constraints. Through solving (6)–(9), we can find the most time-efficient links in the network. The optimal flows can next be transformed into splitting rates $\beta_{m,d}$ (using a formula like (4)) at each node $n \in \mathcal{N}$.

Recall that the stench pheromone function in ACR is used to penalize that too many ants converge to the same link, so as to avoid traffic congestion and to limit the number of vehicles in sensitive zones, such as hospitals and schools. Since the stench pheromone function is designable, we can formulate it as a piecewise affine (PWA) function that satisfies the properties required in Section II. In the SLP approach, we consider it as a penalty function J_{pen} , with the following general form:

$$\left\{ \begin{array}{l} P_{m,0} \sum_{d \in \mathcal{D}} q_{m,d}, \\ \quad \text{if } 0 \leq \sum_{d \in \mathcal{D}} q_{m,d} \leq q_{m,1}^{\text{thresh}} \\ \\ P_{m,1} \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,1}^{\text{thresh}} \right) + B_1, \\ \quad \text{if } q_{m,1}^{\text{thresh}} \leq \sum_{d \in \mathcal{D}} q_{m,d} \leq q_{m,2}^{\text{thresh}} \\ \\ \vdots \\ \\ P_{m,j} \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,j}^{\text{thresh}} \right) + B_j, \\ \quad \text{if } q_{m,j}^{\text{thresh}} \leq \sum_{d \in \mathcal{D}} q_{m,d} \leq q_m^{\text{cap}} \end{array} \right. \quad (10)$$

where $q_{m,i}^{\text{thresh}}$, $i = 1, 2, \dots, j$, denote the predefined thresholds corresponding to the sensitive zones for the flow on link m , q_m^{cap} denotes the maximum flow on link m , $P_{m,i}$ denotes the slope of the i th affine sub-function, and B_i is a constant, which is defined as:

$$B_i = \left\{ \begin{array}{l} P_{m,0} q_{m,1}^{\text{thresh}}, \\ \quad \text{for } i = 1 \\ \\ B_{i-1} + P_{m,i-1} (q_{m,i}^{\text{thresh}} - q_{m,i-1}^{\text{thresh}}), \\ \quad \text{for } i = 2, \dots, j \end{array} \right. \quad (11)$$

in order to guarantee continuity of the function J_{pen} . The

penalty function (10) is also an objective to be minimized. In general, a PWA penalty function will result in a so-called mixed-integer linear programming (MILP) problem (see Remark 2). However, if J_{pen} is a *convex* PWA function, then minimizing J_{pen} can be recast by introducing an additional variable g_m :

$$\min \sum_{m \in \mathcal{M}} g_m, \quad (12)$$

subject to

$$g_m \geq P_0 \sum_{d \in \mathcal{D}} q_{m,d}, \quad \forall m \in \mathcal{M} \quad (13)$$

$$g_m \geq P_1 \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,1}^{\text{thresh}} \right) + B_1, \quad \forall m \in \mathcal{M} \quad (14)$$

\vdots

$$g_m \geq P_j \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,j}^{\text{thresh}} \right) + B_j, \quad \forall m \in \mathcal{M} \quad (15)$$

We can easily verify that the LP problem (12)–(15) amounts to minimizing J_{pen} .

If we combine the two objectives, we get

$$\min J_{\text{TTS}} + \zeta J_{\text{pen}} \quad (16)$$

with weight parameter $\zeta > 0$. The minimization of (16) subject to (7)–(9) and (13)–(15) is an LP problem, which can be solved using one of the many available algorithms for linear programming (see e.g. [15, Chapter 1] or [16]). In this way, we can determine the optimal flows $q_{m,d}$ in the network that yield a balanced trade-off between minimizing the total travel time and avoiding congestion.

Remark 1 If the demand at the origins of the traffic network is such that the total capacity of the network is exceeded, then in the real network queues will arise. Such a situation is not yet captured by the LP defined above and in fact the LP will be infeasible if the demand is higher than the total capacity of the network. If this happens we can update the objective and constraint (7) as follows in order to mimic the creation of queues at the origins of the network and to get an LP that is always feasible. For each origin $o \in \mathcal{O}$ we introduce an extra slack variable w_o that reflects the excess demand and we replace (7) by

$$\sum_{m' \in O(o)} \sum_{d \in \mathcal{D}} q_{m',d} = d_o - w_o, \quad \forall o \in \mathcal{O} \quad (17)$$

$$w_o \geq 0, \quad \forall o \in \mathcal{O} \quad (18)$$

and we use

$$\min J_{\text{TTS}} + \zeta J_{\text{pen}} + \eta \sum_{o \in \mathcal{O}} w_o \quad (19)$$

with $\eta > 0$ a weight parameter (with $\eta \ll \zeta$) instead of (16). Note that the updated problem of minimizing (19) subject to (17), (18), (8), (9), (13), and (14) is still an LP problem.

Remark 2 If J_{pen} is a *non-convex* PWA function, the following procedure, which is inspired by [17], shows that

we can solve (16), subject to (7) – (9) by the MILP approach.

First, we rewrite the penalty function (10) as a recursive function:

$$\begin{aligned}
g_{m,0} &= Q_{m,0} \sum_{d \in \mathcal{D}} q_{m,d}, \\
&\quad \text{if } 0 \leq \sum_{d \in \mathcal{D}} q_{m,d} \leq q_{m,1}^{\text{thresh}} \\
g_{m,i} &= g_{m,i-1} + Q_{m,i} \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,i}^{\text{thresh}} \right), \\
&\quad \text{if } q_{m,i}^{\text{thresh}} \leq \sum_{d \in \mathcal{D}} q_{m,d} \leq q_{m,i+1}^{\text{thresh}} \quad (20)
\end{aligned}$$

for $i = 2, \dots, j$, and $q_{m,j+1}^{\text{thresh}} = q_m^{\text{cap}}$. Parameter $Q_{m,i}$ is a new parameter for the slope, which can be obtained according to (10). Then, we associate each condition $\sum_{d \in \mathcal{D}} q_{m,d} \geq q_{m,i}^{\text{thresh}}$ with a binary logical variable $\delta_{m,i} \in \{0, 1\}$ such that

$$[\delta_{m,i} = 1] \Leftrightarrow \left[\sum_{d \in \mathcal{D}} q_{m,d} \geq q_{m,i}^{\text{thresh}} \right], \quad (21)$$

where “ \Leftrightarrow ” means “if and only if”. It is easy to verify that (21) is equivalent to

$$\sum_{d \in \mathcal{D}} q_{m,d} \leq q_{m,i}^{\text{thresh}} - (q_{m,i}^{\text{thresh}} - q_m^{\text{cap}}) \delta_{m,i} \quad (22)$$

$$\sum_{d \in \mathcal{D}} q_{m,d} \geq q_{m,i}^{\text{thresh}} - q_{m,i}^{\text{thresh}} (1 - \delta_{m,i}) \quad (23)$$

Then (10) can be rewritten as:

$$\delta_{m,0} Q_0 \sum_{d \in \mathcal{D}} q_{m,d} + \sum_{n=1}^j \delta_{m,n} \left(Q_{m,n} \left(\sum_{d \in \mathcal{D}} q_{m,d} - q_{m,n}^{\text{thresh}} \right) \right) \quad (24)$$

The term $\delta_{m,i} \sum_{d \in \mathcal{D}} q_{m,d}$ can be replaced by an auxiliary real variable $z_{m,i} = \delta_{m,i} \sum_{d \in \mathcal{D}} q_{m,d}$, which can be expressed as:

$$z_{m,i} \leq q_m^{\text{cap}} \delta_{m,i}, \quad (25)$$

$$z_{m,i} \geq 0, \quad (26)$$

$$z_{m,i} \leq \sum_{d \in \mathcal{D}} q_{m,d}, \quad (27)$$

$$z_{m,i} \geq \sum_{d \in \mathcal{D}} q_{m,d} - q_m^{\text{cap}} (1 - \delta_{m,i}), \quad (28)$$

so then (24) is reduced to the linear expression:

$$Q_0 z_{m,0} + \sum_{n=1}^j Q_{m,n} (z_{m,n} - q_{m,n}^{\text{thresh}} \delta_{m,n}), \quad (29)$$

subject to the linear constraints (22), (23), and (25)–(28). Note that the optimization variables in (29) include continuous variables ($q_{m,d}$ and $z_{m,i}$), and also binary variables ($\delta_{m,i}$). So then the problem becomes a MILP problem, which can be solved efficiently by several existing state-of-the-art commercial and free solvers, such as CPLEX, Xpress-MP, or GLPK [18], [19].

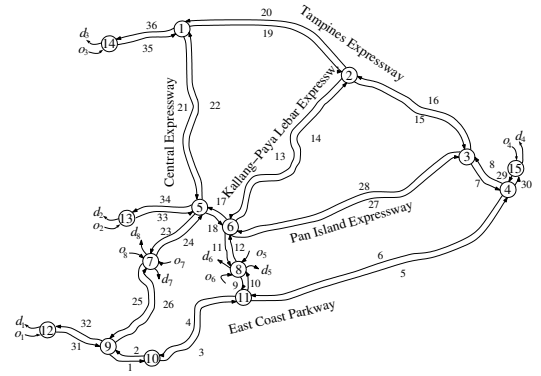


Fig. 3. The central and eastern parts of the Singapore expressway network.

	Link number	Length (km)	Lanes	Nodes	Capacity q_m^{cap} (veh/h)
★	1, 2	3.0	3	9, 10	4500
★	3, 4	4.5	4	10, 11	6000
	5, 6	13.0	4	11, 4	6000
	7, 8	2.0	4	3, 4	6000
★	9, 10	1.0	3	8, 11	4500
★	11, 12	2.0	3	6, 8	4500
	13, 14	8.0	3	2, 6	4500
	15, 16	6.5	3	2, 3	4500
★	17, 18	2.0	4	6, 5	6000
	19, 20	7.0	3	1, 2	4500
	21, 22	7.5	2	1, 5	3000
	23, 24	3.5	2	5, 7	3000
	25, 26	4.5	2	7, 9	3000
	27, 28	11.0	4	6, 3	6000
	29, 30	1.0	4	15, 4	6000
	31, 32	3.0	3	12, 9	4500
	33, 34	3.0	4	13, 5	6000
	35, 36	3.0	3	14, 1	4500

TABLE I
PARAMETERS OF REGULAR LINKS AND SENSITIVE LINKS (WITH ★) IN THE SINGAPORE EXPRESSWAY NETWORK

IV. CASE STUDY

Now we apply both ACR and SLP to the case study considered in [8], which involves the central and eastern parts of the Singapore expressway network (see Figure 3).

A. Simulation Settings

The part of the Singapore expressway network we consider contains 8 origins, 8 destinations, and 36 links. This area includes the central business district, connected to origins and destinations 5, 6, 7, and 8, as well as the connection with the airport through origin and destination 4. The parameters of each link are presented in Table I, where the sensitive links are marked by ★.

We use the METANET model (see [8] for details of the model parameters used) as the traffic prediction model in the MPC controller, and for the simulation of the real traffic

symbol	value	meaning
$N_{\text{ant,total}}$	3000	total number of ants in ACR
τ_0	100	initial pheromone level
σ	0.1	evaporation rate
N^{iter}	1000	maximum number of iterations of the outer loop in ACR
N^{loop}	10	maximum number of prediction-optimization loops in MPC
P_0	0	
P_1	1	slopes of the PWA function
P_2	20	
ζ	0.5	tuning parameter between J_{TTS} and J_{pen}
γ_m	0.5	threshold flow parameter in sensitive zones
	0.7	threshold flow parameter in non-sensitive zones

TABLE II
PARAMETERS OF OPTIMIZATION APPROACH

network as well. For the sake of simplicity, we here just assign one threshold flow for each link in the network, and hence the penalty function is defined as:

$$g_m = \max \left(P_0 q_m, P_1 (q_m - q_m^{\text{thresh}}) + P_0 q_m^{\text{thresh}}, P_2 (q_m - q_m^{\text{cap}}) + P_1 (q_m^{\text{cap}} - q_m^{\text{thresh}}) + P_0 q_m^{\text{thresh}} \right), \quad (30)$$

where $P_0 < P_1 < P_2$ holds to guarantee the convexity of (30) (otherwise the MILP approach should be used). We select the threshold flow q_m^{thresh} as:

$$q_m^{\text{thresh}} = \gamma_m q_m^{\text{cap}}, \quad (31)$$

with $\gamma_m \in (0, 1)$. The parameters used for the optimization are shown in Table II.

B. Simulation Result

We now compare the performance of the two methods. Since there is no explicit objective function in ACR, we define a so-called assessment function for it in a similar way as for SLP (see (16)):

$$J_{\text{ACR}} = \sum_{s,t} N_{s,t}^{\text{ant}} \cdot \varphi_{\ell(s,t)} + \zeta \sum_{s,t} G_{s,t} \quad (32)$$

The total number of simulation-optimization cycles is 10.

The results of the simulations are shown in Figure 4 and 5. We can see that at the second cycle, SLP has achieved the final optimum, while the value of objective function of the ACR is slightly fluctuating. This is because ACR is a stochastic algorithm, so the number of ants $N_{s,t}^{\text{ant}}$ on each link will never be exactly the same at the end of each ACR run, even if the link travel times do no longer change.

By optimizing with ACR and SLP, we will find the resulting optimized flows in the network. From Figure 5, we can see that in the SLP solution only links 1, 3, 5, 7,

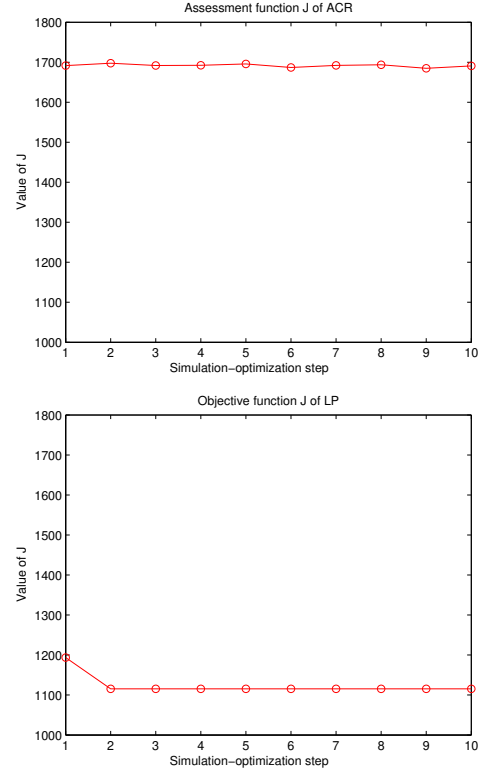


Fig. 4. Assessment/objective function J for ACR and LP

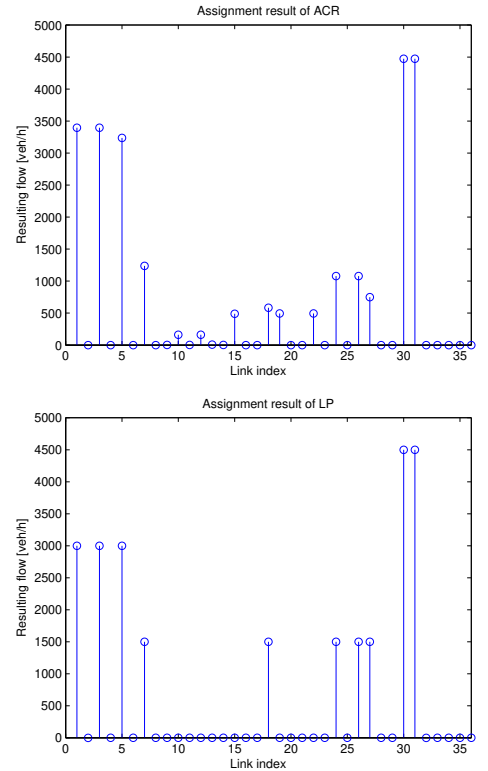


Fig. 5. Final assignments for ACR and LP

18, 24, 26, 27, 30, and 31 are assigned a non-zero flow q_{m,d_4} ; these links actually form two routes in the network: $\mathcal{R}_1 = \{31, 1, 3, 5, 30\}$ and $\mathcal{R}_2 = \{31, 26, 24, 18, 27, 7\}$. To directly compare the assignment results of ACR with SLP, we translate the number of ants on each link $m = \ell(s, t)$ into the flow q_{m,d_4} on the same link through multiplying the number $N_{s,t}^{\text{ant}}$ by the following factor:

$$\mu = \frac{d_{o_1,d_4}}{N_{\text{ant},\text{total}}}$$

Note from Figure 5 that the links found by LP are also chosen by ACR. However, we have even more links found by ACR, that are links 10, 12, 15, 19, and 22, although we just have a low flow on them.

The reason that the result of ACR differs from the result of LP is that optimization mechanism of the two methods is in fact somewhat different. The ACR algorithm does not explicitly apply (16) like SLP, but instead, it uses two opposite types of pheromones to evaluate the routes in the network. Moreover, ACR is a stochastic algorithm, which can also lead to a difference in the final assignments between ACR and SLP.

There is a big advantage of SLP, which is the significantly lower computation time. For the simulation of the Singapore expressway network in MATLAB, the total computation time with SLP is 4.61 s, while the total computation time with ACR is 3.3 h (both on a single processor). Apart from the fact that we have only let ants one by one search the routes in the network, not in parallel, the computational burden of ACR is still much higher than that of SLP. So even if a large number of parallel processors are used, SLP will still outperform ACR from a computational point of view, while for the given case study it yields an assignment that is very close to the ACR assignment. Note however that SLP only works for convex piecewise affine stench functions, while ACR is much more general, with non-linear link costs and non-linear, non-convex stench functions (see Condition (a) and Footnote 2).

V. CONCLUSIONS

We have shown that our previous work on dynamic traffic routing using the Ant Colony Routing (ACR) approach can be also formulated as a Sequential Linear Programming (SLP) problem if Condition (a) and Condition (b) are both satisfied. For a case study we found that the SLP approach can almost achieve the same assignment as ACR, with some slight differences due to the different ways to formulate the objective function in ACR and SLP. Moreover, with SLP we can dramatically improve the efficiency of the optimization method compared to ACR.

Topics for future work include: an in-depth comparison and assessment of the SLP and ACR approaches using more extensive cases studies, tuning of the ACR parameters, and improving the convergence of the simulation-optimization iteration process (e.g., by directly including a relation between flow and travel time into the optimization).

ACKNOWLEDGMENTS

Research supported by the BSIK project “Next Generation Infrastructures (NGI)”, the European COST Actions TU0702 and TU1102, the European 7th Framework Network of Excellence “Highly-complex and networked control systems (HYCON2)”, and the Transport Research Center Delft.

REFERENCES

- [1] L. Fu, “An adaptive routing algorithm for in-vehicle route guidance systems with real-time information,” *Transportation Research Part B: Methodological*, vol. 35, no. 8, pp. 749 – 765, 2001.
- [2] S. Kim, M. Lewis, and C. W. III, “Optimal vehicle routing with real-time traffic information,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178 – 188, June 2005.
- [3] E. Ericsson, H. Larsson, and K. Brundell-Freij, “Optimizing route choice for lowest fuel consumption – potential effects of a new driver support tool,” *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 6, pp. 369 – 383, 2006.
- [4] M. Papageorgiou, “Dynamic modeling, assignment, and route guidance in traffic networks,” *Transportation Research Part B*, vol. 24B, no. 6, pp. 471–495, 1990.
- [5] A. Messmer and M. Papageorgiou, “Route diversion control in motorway networks via nonlinear optimization,” *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, pp. 144–153, Mar. 1995.
- [6] J. G. Wardrop, “Some theoretical aspects of road traffic research,” *Proceedings of the Institute of Civil Engineers, Part II*, vol. 1, pp. 325–378, 1952.
- [7] E. Schmitt and H. Jula, “Vehicle route guidance systems: Classification and comparison,” in *Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems (ITSC 2006)*, Toronto, Canada, 2006, pp. 242 –247.
- [8] Z. Cong, B. De Schutter, and R. Babuška, “A new ant colony routing approach with a trade-off between system and user optimum,” in *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC 2011)*, Washington, DC, U.S., 2011, pp. 1369–1374.
- [9] E. F. Camacho and C. Bordons, *Model Predictive Control in the Process Industry*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.
- [10] J. M. Maciejowski, *Predictive Control with Constraints*. Essex, England: Prentice Hall, 2002.
- [11] R. Bishop, *Intelligent Vehicle Technology and Trends*, ser. Artech House ITS Library. Boston, Massachusetts: Artech House, 2005.
- [12] Z. Cong, B. De Schutter, and R. Babuška, “Ant colony routing algorithm for freeway networks,” 2012, submit to Transportation Research Part C.
- [13] M. Dorigo, V. Maniezzo, and A. Colomi, “The ant system: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 1–13, 1996.
- [14] M. Dorigo and L. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
- [15] P. Pardalos and M. Resende, Eds., *Handbook of Applied Optimization*. Oxford, UK: Oxford University Press, 2002.
- [16] G. Dantzig and M. Thapa, *Linear Programming 1: Introduction*, ser. Springer Series in Operations Research and Financial Engineering. New York, New York: Springer-Verlag, 1997.
- [17] A. Bemporad and M. Morari, “Control of systems integrating logic, dynamics, and constraints,” *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.
- [18] A. Atamtürk and M. Savelsbergh, “Integer-programming software systems,” *Annals of Operations Research*, vol. 140, no. 1, pp. 67–124, Nov. 2005.
- [19] J. T. Linderoth and T. K. Ralphs, “Noncommercial software for mixed-integer linear programming,” in *Integer Programming: Theory and Practice*, J. K. Karlof, Ed. CRC Press, 2005, pp. 253–303.