Delft University of Technology

Delft Center for Systems and Control

Technical report 13-028

Ant colony routing algorithm for freeway networks*

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

If you want to cite this report, please use the following reference instead:
Z. Cong, B. De Schutter, and R. Babuška, "Ant colony routing algorithm for freeway networks," *Transportation Research Part C*, vol. 37, pp. 1–19, Dec. 2013. doi:10.1016/j.trc.2013.09.008

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://pub.bartdeschutter.org/abs/13_028

Ant Colony Routing Algorithm for Freeway Networks

Zhe Cong[a,*], Bart De Schutter[a], Robert Babuška[a]

[a]*Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands*

Abstract

Dynamic traffic routing refers to the process of (re)directing vehicles at junctions in a traffic network according to the evolving traffic conditions. The traffic management center can determine desired routes for drivers in order to optimize the performance of the traffic network by dynamic traffic routing. However, a traffic network may have thousands of links and nodes, resulting in a large-scale and computationally complex nonlinear, non-convex optimization problem. To solve this problem, Ant Colony Optimization (ACO) is chosen as the optimization method in this paper because of its powerful optimization heuristic for combinatorial optimization problems. ACO is implemented online to determine the control signal — i.e., the splitting rates at each node. However, using standard ACO for traffic routing is characterized by four main disadvantages: 1. traffic flows for different origins and destinations cannot be distinguished; 2. all ants may converge to one route, causing congestion; 3. constraints cannot be taken into account; and 4. neither can dynamic link costs. These problems are addressed by adopting a novel ACO algorithm with stench pheromone and with colored ants, called Ant Colony Routing (ACR). Using the stench pheromone, the ACR algorithm can distribute the vehicles over the traffic network with less or no traffic congestion, as well as reduce the number of vehicles near some sensitive zones, such as hospitals and schools. With colored ants, the traffic flows for multiple origins and destinations can be represented. The proposed approach is also implemented in a simulation-based case study in the Walcheren area, the Netherlands, illustrating the effectiveness of the approach.

Keywords: Ant Colony Optimization, Model Predictive Control, Dynamic Traffic Routing, Stench pheromone.

1. Introduction

Traffic network control involves complex traffic conditions related to traffic flow dynamics, driver behavior, and traffic demand. With the rapid growth of human population and jobs distributed unevenly in different locations, effective and efficient operation of traffic networks is required more than ever. Nowadays, increasing and expanding infrastructures is costly and impractical, and it often only temporarily relieves the burden of traffic networks rather than aiming for a long-term solution. Therefore, development of traffic control strategies is much more desirable. Dynamic traffic routing is one of such promising methods, which is often used to guide

*Corresponding author. Tel:+31-15-27-86524; Fax:+31 1527 86679.

Email addresses: z.cong@tudelft.nl (Zhe Cong), b.deschutter@tudelft.nl (Bart De Schutter), r.babuska@tudelft.nl (Robert Babuška)

drivers in the traffic networks, seeking to achieve some objectives, such as user equilibrium or system optimum (Wardrop, 1952).

A broad literature exists on this topic (Peeta and Ziliaskopoulos, 2001; Ziliaskopoulos, 2000; Carey and Subrahmanian, 2000; Ran et al., 1993; Mahmassani and Peeta, 1993, 1995). Peeta and Ziliaskopoulos (2001) summarize the main dynamic traffic assignment approaches, ranging from mathematical programming (Ziliaskopoulos, 2000; Carey and Subrahmanian, 2000), to optimal control (Ran et al., 1993; Kotsialos et al., 2002), and to simulation-based methods (Mahmassani and Peeta, 1993, 1995). Specifically, in mathematical programming models, the problem is usually formulated in a discrete-time setting. Ziliaskopoulos (2000) uses the so-called cell transmission model to formulate the single destination system optimum dynamic traffic assignment problem as a linear programming problem. Carey and Subrahmanian (2000) also propose a linear system optimum formulation, aiming at deriving insights in the properties of the dynamic traffic assignment model. For optimal control method, Ran et al. (1993) use it to obtain a convex model for the instantaneous user equilibrium dynamic traffic routing problem by defining link inflows and outflows as control variables. Kotsialos et al. (2002) formulated the dynamic traffic routing as a discrete-time optimal control problem for finding the system optimum solution, and used a numerical non-linear optimization algorithm to solve it. The simulation-based approaches focus on traffic simulations instead of analytical evaluation. Mahmassani and Peeta (1993, 1995) use a mesoscopic traffic simulator, DYNASMART, and an iterative algorithm to find the system optimum and user equilibrium solutions. Other work, e.g., (Bottom et al., 1999; Paz and Peeta, 2009b,a), focuses on the drivers' reaction to the guidance information provided in the route choice mechanism. Bottom et al. (1999) developed an analysis framework for generating route guidance instructions, by using real-time measurements and short-term predictions. Paz and Peeta (2009b,a) aim at enhancing the traffic network performance while explicitly accounting for drivers' likely reactions to the real-time information.

Some of the work above uses a numerical non-linear optimization methods, which require extremely high computational burden, while others sacrifice performance for improving the computation speed. In order to find a well-balanced trade-off between the performance and the computation speed, we propose a novel, alternative algorithm for dynamic traffic routing based on artificial ants in this paper. Ants in nature can display surprisingly intelligent collective behavior to find the shortest route between their nest and a source of food, although each individual ant only has a very local searching capability. This results from the fact that ants use a medium called pheromone to communicate with each other so that they can indirectly exchange information. The class of Ant Colony Optimization (ACO) algorithms (Dorigo and Stützle, 2004) originates from this self-organizing behavior of ants, and has been developed to solve a number of combinatorial optimization problems, such as shortest path problems, optimal task assignment problems, best routing scheme problems, and so on. Furthermore, some work has also been done in solving the dynamic traffic routing problem (Tatomir and Rothkrantz, 2006; Alves et al., 2010), through using the standard ACO algorithm. However, most of the algorithms reported in literature have their own limitations. Tatomir and Rothkrantz (2006) use a routing table containing possible splitting rates of traffic flows in every network node. At each node, ants choose one group of splitting rates from the table to apply in the traffic network. This method may result in an exponentially increasing computational burden when more nodes are added in the network, which makes the algorithm less efficient. Moreover, Alves et al. (2010) only investigate single-origin single-destination networks, and they use a static traffic model where the traffic conditions are time-invariant.

In fact, using the standard ACO algorithm for dynamic traffic routing suffers from four main

issues:

1. Ants in ACO have no individually pre-assigned destinations, while each vehicle in a traffic network has its own pre-determined destination;
2. Ants only strive for the user equilibrium, while traffic management has global objectives;
3. Ant networks have no limiting capacities on links, while traffic networks are constrained by link capacity;
4. Link costs in ant networks are fixed and static, while link costs in traffic networks dynamically depend on the time-varying traffic conditions.

Motivated by the four issues above, we introduce an ant-based routing algorithm for traffic networks, called Ant Colony Routing (ACR), in this paper. The first issue has been addressed by Cong et al. (2011) by introducing a variant of the ACO algorithm, called ACO with stench pheromone (ACO-SP), which is developed to solve a network routing problem. The stench pheromone has an opposite function to the regular pheromone in the standard ACO algorithm, and it is used to disperse ants over the network for the sake of a system-wide objective. In order to tackle the second issue, we add a new concept called colored ants, in which each color is assigned to a corresponding destination, and colored ants are only sensitive to their own color. Moreover, the capacity constraints, as well as other limitations on the numbers of vehicles on links, are addressed by properly selecting the stench pheromone function. Finally, time-variant traffic variables, obtained from dynamic traffic models, are taken into account into the link cost for the ant network, and an iterative algorithm is used to capture the dynamic nature.

The ACR algorithm is going to be applied within a Model Predictive Control (MPC) (Maciejowski, 2002; Rawlings and Mayne, 2009; Hegyi et al., 2005) framework. However, before that, we first implement a network pruning step. This is because a traffic network may involve thousands of links and nodes, and some of the links and nodes compose routes that could be relatively long such that they are unlikely to be chosen by drivers. We hence use the network pruning step to remove such "unnecessary" links and nodes to reduce the complexity of the traffic network, and to decrease the computational burden of the ACR algorithm. The network pruning step involves a static optimization problem, which is not solved by ACO but by linear programming, and is only performed once. For on-line dynamic traffic routing, the MPC controller is repeatedly executed by using the ACR algorithm as optimization method. Specifically, at each control step, we first measure the current state of the traffic network, and use a dynamic traffic model to predict the future state for a certain prediction period. Then, both current and predicted states are used to calculate the link cost assigned to each link in the ant network, and accordingly the ACR algorithm is used to optimize the routing problem. As a result, the control signals in the traffic network — splitting rates, are determined by the numbers of ants that have traveled on each link in the ant network. We keep on recycling such a prediction-optimization loop at the same control step until a given convergence criterion is satisfied, and then we will apply the resulting splitting rates to the real traffic network. Next, we shift both control horizon and prediction horizon of MPC one sample time step forward, and repeat the whole process. The structure of the proposed control strategy is illustrated in Figure 1.

The rest of this paper is structured as follows. Section 2 defines the dynamic traffic routing problem. Next, the ACO-SP algorithm is briefly recapitulated in Section 3. Section 4 illustrates the ACR algorithm for solving the dynamic traffic routing problem. In Section 5, a two-step control strategy — network pruning and MPC — is elaborated on. Section 6 illustrates the new dynamic traffic routing control method using a study case involving the freeway network in the

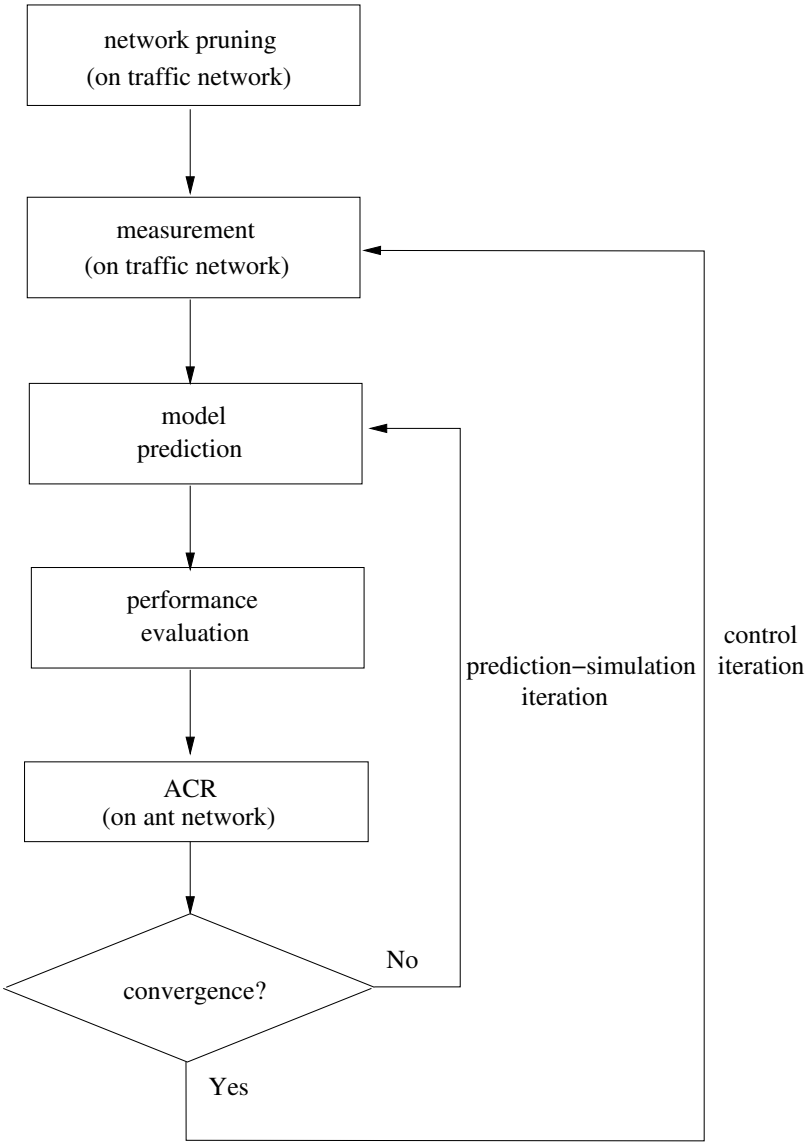

Figure 1: Structure of the proposed dynamic traffic routing control strategy

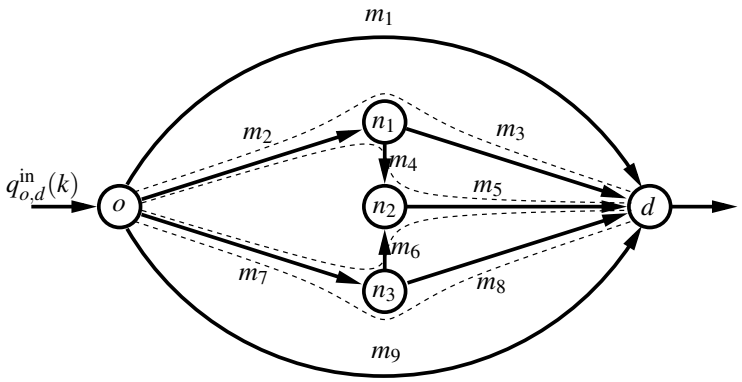

Figure 2: Illustration of inflow $q^{\text{in}}_{o,d}(k)$ of origin-destination pair (o,d) distributed over multiple routes.

Walcheren area, the Netherlands. A short discussion of open issues and topics for future work finally conclude the paper in Section 7.

2. Problem Statement

The objective of this paper is twofold:

1. We aim at finding an optimal routing solution to reduce the travel costs (total time spent (TTS) in this paper[1]) in the traffic network;
2. We aim at reducing the number of vehicles on each link in the traffic network, especially in some sensitive zones, e.g., near hospitals and schools, in order to improve safety and to reduce noise and pollution.

Unlike in most other optimization-based traffic control methods, we do not use an explicit overall objective function in the ACR algorithm, because the two sub-objectives introduced above are respectively evaluated by cost function and stench pheromone function in ACR. More specifically, the first sub-objective is based on the fact that artificial ants always strive for the optimal solution, and the second sub-objective is achieved by the stench pheromone pushing artificial ants away when ants converge on the same link. Through this attracting and pushing mechanism, the two sub-objectives can be implicitly included the ACR algorithm. This paper aims at finding a well-balanced trade-off between sub-objectives 1 and 2. In practice, such a trade-off should depend on traffic policies made by traffic management authorities.

Next, we will introduce some important concepts related to the traffic networks, which will be used in this paper. A traffic network, especially a freeway network, can be modeled as a directed graph with nodes and links as shown in Figure 2. In a traffic network, a stretch of road with uniform characteristics — i.e., without any on-ramp or off-ramp and without any major changes in geometry — is called a *link* (indicated by the index m). For more accurate modeling, each link m, with a length L_m, is divided into N_m segments (indicated by the index i). If major changes occur in the characteristics of a link, then a node is placed there and a new link starts. There are three different types of nodes in the network, origins (indicated by the index o), destinations

[1] Any other cost, e.g., tolling, fuel consumption, emission, can be added to the travel costs in the ACR algorithm.

symbol	unit	meaning
s, t	[-]	vertex in the ant network
n	[-]	node in the traffic network
$\mathscr{O}$	[-]	set of origins in the traffic network
$\mathscr{D}$	[-]	set of destinations in the traffic network
$\mathscr{N}$	[-]	set of intermediate nodes in the traffic network
$\mathscr{M}$	[-]	set of all links in the traffic network
$I(n)$	[-]	set of incoming links for node n
$O(n)$	[-]	set of outgoing links for node n
k	[-]	simulation step counter
k_c	[-]	control step counter
T	[s]	simulation sample time
T_c	[s]	control sample time
$q^{\mathrm{in}}_{o,d}(k)$	[veh/h]	traffic inflow from origin o to destination d at simulation step k
$q_{m,i,d}(k)$	[veh/h]	traffic flow with destination d on segment i of link m at simulation step k
$Q_{n,d}(k)$	[veh/h]	total traffic flow with destination d in node n at simulation step k
$\beta_{n,m,d}(k_\mathrm{c})$	[-]	splitting rate of vehicles with destination d turning from node n to link m at control step k_c
N_p	[-]	prediction horizon
ℓ	[-]	map associating an arc (s,t) in the ant network to a link $m = \ell(s,t)$ in the traffic network
$\varphi_{s,t}(k_\mathrm{c})$	[-]	link cost on arc $(s,t) = \ell^{-1}(m)$ in the ant network at control step k_c

Table 1: Symbols used for networks

(indicated by the index d), and intermediate nodes (indicated by the index n). From an origin o to a destination d, a concatenation of links without loops is defined as a *route* (indicated by the index r). All important symbols used in the paper are listed in Table 1.

We consider a discrete-time set-up for traffic networks in this paper. The simulation sample time T is used to describe the traffic model, and the control sample time is denoted by T_c. Correspondingly, we define k as the simulation step counter, and k_c as the control step counter. For the sake of simplicity, we assume that T is an integer divisor of T_c with a relationship $T_\mathrm{c} = MT$, where M is an integer.

For each origin-destination (OD) pair $(o,d) \in \mathscr{O} \times \mathscr{D}$, with $\mathscr{O}$ the set of origins and $\mathscr{D}$ the set of destinations in the network, $q^{\mathrm{in}}_{o,d}(k)$ is the traffic inflow with destination d entering at origin o at simulation step k. For each segment i of link $m \in \mathscr{M}$, with $\mathscr{M}$ the set of all links, the flow of vehicles at simulation step k that are traveling towards destination d is denoted by [2] $q_{m,i,d}(k)$. Without loss of generality[3], we assume from now on that each origin $o \in \mathscr{O}$ has only one outgoing

[2] All the future traffic states, i.e. flow $q_{m,i}(k)$, density $\rho_{m,i}(k)$, speed variable $v_{m,i}(k)$, and queue length $w_m(k)$ at each simulation step k can be predicted by a dynamic traffic model, provided the O-D demands and the current traffic states are known. More specifically, the O-D demands can be obtained in two ways: via O-D estimation Zhou et al. (2003) or via on-board devices. Usually, the current total flow variable $q_{m,i}(k)$, the density variable $\rho_{m,i}(k)$, or the speed variable $v_{m,i}(k)$ can be directly measured by loop detectors or cameras, and the partial flow variable $q_{m,i,d}(k)$ can be furthermore estimated by using state estimators Stano et al. (2013).

[3] If necessary, we can always introduce a virtual link with zero length and zero travel time connected to a virtual node.

link, and that each destination $d \in \mathscr{D}$ has only one incoming link. For a specific node $n \in \mathscr{N}$, with $\mathscr{N}$ the set of intermediate nodes in the network, the total inflow $Q_{n,d}(k)$ with destination d at simulation step k is given by:

$$Q_{n,d}(k) = \sum_{m' \in I(n)} q_{m',N_{m'},d}(k)$$

with $I(n)$ the set of incoming links of node n, and $q_{m',N_{m'},d}(k)$ the traffic flow of vehicles with destination d on the last segment (with index $N_{m'}$) of incoming link $m' \in I(n)$ at simulation step k. The traffic flow $q_{m,1,d}(k)$ on the initial segment (with index 1) of outgoing link $m \in O(n)$ at simulation step k is then determined by the splitting rate $\beta_{n,m,d}(k_c)$ at the corresponding control step k_c, denoted by $k_c(k)$ with $k_c = \text{floor}(k/M)$, where function floor(x) denotes the largest integer less than or equal to x:

$$q_{m,1,d}(k) = \beta_{n,m,d}(k_c(k))Q_{n,d}(k) \ .$$

As mentioned in Section 1, the optimal traffic routing solution is applied through the splitting rate $\beta_{n,m,d}(k_c(k))$ in each node. In the other words, the goal of the ACR algorithm can also be considered as determining the optimal splitting rate $\beta_{n,m,d}(k_c(k))$ in the traffic network.

3. ACO algorithm with Stench Pheromone

The ACO-SP algorithm (Cong et al., 2011) has been developed to solve a network routing problem by using artificial ants. This algorithm aims at dispersing ants over the network without ending up in a situation where all ants travel on the best[4] route that may cause congestion.

More specifically, ants in ACO-SP always strive for the best route at the beginning of the optimization; if the best route becomes too crowded[5,6] for ants to travel on, then ants will strive for the second best route; if the second best route also gets fully loaded, the third best one will be chosen, etc. The reason that ants can be dispersed rather than converging to the same route is because ACO-SP is characterized by two types of pheromone: the same regular pheromone as in the standard ACO algorithm and the newly introduced stench pheromone. The former is used to attract ants to the best routes in the network so as to guarantee the effectiveness of the algorithm, while the latter is used to decrease the total amount of pheromone levels on the arcs to prevent the regular pheromone from being accumulated too much[5,6] on the same arc. In such a way, a part of the ants are pushed away from their previously selected route, and start to choose an alternative route in the network.

An ant network is a network traveled by ants. It is usually modeled as a weighted graph according to a specific optimization problem (see e.g. Stützle and Dorigo (1999); Salari and Eshghi (2008); Rivero et al. (2012)). Moreover, a particular route r_a in the ant network is a

[4] 'Best' is in the sense of e.g. minimal travel time.

[5] The terms "too crowded" and "too much" may sound vague due to the fact that their definitions could vary in different cases, e.g. a small number of ants could mean "crowded" on the links that represent sensitive zones in the traffic network, while a large number of ants could still mean "not crowded" on the links that represent non-sensitive zones in the traffic network. This indicates an interplay between policy decisions and tuning parameters.

[6] The terms "too crowded" and "too much" are used to explain the second objective, which is related to the occurrence of traffic congestion. Travel costs like the TTS or the other objective (see Footnote 1) should be included in the first objective so that they are properly used in the ACR algorithm.

concatenation of arcs (s,t) chosen by ant a from the origin vertex to the destination vertex, where vertices s and t connected by the arc (s,t). A pheromone variable $\tau_{s,t}$ is associated with each arc (s,t) in the network, and it represents the knowledge acquired by ants about the optimal solution over time. At the beginning of the ACO-SP algorithm, all pheromone variables should be set to some initial value $\tau_0 > 0$.

There are two loops in the ACO-SP algorithm: an inner loop, which is called as the route construction, and an outer loop, which is called as the pheromone updating. We explain these two loops in more detail as follows.

- In the inner loop, all ants repeatedly select arcs to construct a route from an origin vertex to a destination vertex. For each ant a, a route construction starts with an empty route $r_a = \emptyset$, and at each construction step r_a is extended by adding a feasible arc (s,t) if ant a currently stands at vertex s and moves to vertex t. In this loop, ant a chooses the next solution component (s,t) with the probability $p_a(t|s)$ calculated as follows:

$$p_a(t|s) = \frac{\left(\max\{\tau_{\min}, \tau_{s,t}\}\right)^{\alpha}}{\sum_{\tilde{t} \in \mathscr{N}_{s,a}} \left(\max\{\tau_{\min}, \tau_{s,\tilde{t}}\}\right)^{\alpha}}, \tag{1}$$

  with $\tau_{\min} > 0$ a given constant which prevents the denominator of (1) from becoming zero, the parameter $\alpha \geqslant 1$ determining the relative importance of pheromone, i.e. ants are more likely to choose links with high pheromone level when the value of parameter α is higher, and $\mathscr{N}_{s,a}$ the set of vertices that are connected to node s and have not yet been visited by ant a. If $\mathscr{N}_{s,a} = \emptyset$, r_a cannot lead to a valid route and the construction of route r_a is aborted. The construction of the route is completed when a destination vertex is added to r_a.

- In the outer loop, the pheromone value on each arc $\tau_{s,t}$ is updated by the ants that have chosen arc (s,t). After all the ants have finished constructing the solutions, the pheromone value is updated by:

$$\tau_{s,t} \leftarrow (1-\rho_{\text{evap}})\tau_{s,t} + \left(\sum_{r \in \mathscr{R}_{\text{upd}}} \Delta\tau_{s,t}(r)\right) - G_{s,t}(N_{s,t}^{\text{ant}}(k_{\text{c}})), \tag{2}$$

  where $N_{s,t}^{\text{ant}}(k_{\text{c}})$ denotes the number of ants that choose link (s,t) in the current iteration of the outer loop, $\mathscr{R}_{\text{upd}}$ denotes the set of solutions that are used for updating. In fact, there exist various specifications of $\mathscr{R}_{\text{upd}}$, but the one we consider in this paper belongs to the most basic ACO algorithm, called the Ant System (Dorigo et al., 1996). This update rule uses all the valid solutions found in the current trial. The parameter $\rho_{\text{evap}} \in (0,1]$ is called the evaporation rate, which has the purpose of uniformly decreasing the pheromone values to avoid too rapid a convergence towards a sub-optimal solution. The regular pheromone deposited by ant a on arc (s,t) is given by:

$$\Delta\tau_{s,t}(r) = \begin{cases} F(r), & \text{if } (s,t) \in r, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

  where F is a cost function assigning strictly positive values to each solution r, where a higher value of F corresponds to a better solution. The stench pheromone deposited on arc

(s,t) should correspond to the number of ants $N^{\text{ant}}_{s,t}(k_c)$. The more ants choose arc (s,t), the more stench pheromone is required to be deposited. Similar to the regular pheromone, we calculate the stench pheromone on arc (s,t) through a function $G_{s,t}(\cdot)$, that has the following properties:

1. If there are no ants visiting link (s,t), i.e. $N^{\text{ant}}_{s,t}(k_c) = 0$, $G_{s,t}(0) = 0$ because then no stench pheromone has to be deposited;
2. $G_{s,t}(\cdot)$ has a low value for a small number of ants, and the value of $G_{s,t}(N^{\text{ant}}_{s,t}(k_c))$ will be monotonically non-decreasing as $N^{\text{ant}}_{s,t}(k_c)$ increases;
3. $G_{s,t}$ can include one or more intermediate threshold levels, at which the value of $G_{s,t}$ will significantly change (see Section 4 for details);
4. When $N^{\text{ant}}_{s,t}(k_c)$ reaches the storage capacity of link $N^{\text{ant,cap}}_{s,t}$, the value of $G_{s,t}(N^{\text{ant}}_{s,t}(k_c))$ steeply rises.

When a given maximum number L of iterations of the outer loop has then been reached, the entire algorithm stops; the optimal routing solution is then determined based on the assignment of ants in the ant network.

4. Ant Colony Routing (ACR)

The ACR algorithm is proposed to tackle the four differences that are stated in Section 1, between an ant network and a traffic network. The solutions are detailed in the rest of this section. Please recall that we have two sub-objectives in this paper: minimizing TTS and penalizing a large number of vehicles in the sensitive zones. Therefore, the regular pheromone function F is defined based on the travel time, and the stench pheromone function G is defined based on the number of ants/vehicles, with parameters determined according to traffic policy preferences.

4.1. ACO-SP

The ACO-SP algorithm introduced in Section 3 is used to solve the traffic routing problem for reducing the travel cost in a traffic network without creating any congestion, if possible. This is the goal that traffic policy decisions generally aim at. Since ACO-SP is applied for dynamic routing in traffic networks, the ant network should be built up according to the traffic network that we want to optimize. More specifically, the ant network has the same topology as the traffic network[7], i.e., the arcs and the vertices in the ant network respectively correspond to the links and the nodes in the traffic network, and moreover at each control step k_c the cost $\varphi_{s,t}(k_c)$ on arc (s,t) is translated from the traffic information on the corresponding link $m = \ell(s,t)$. Since we choose TTS as the main objective[8] in this paper, the cost $\varphi_{s,t}(k_c)$ is calculated according to the travel time on link $m = \ell(s,t)$. The cost function[9] $F(r,k_c)$ in (3) can be defined as the inverse of travel time on route r:

$$F(r,k_c) = \frac{Q}{\sum_{(s,t)\in r} \varphi_{s,t}(k_c)},$$

[7] In fact, we apply a network pruning step before running ACR to reduce the size of the traffic network. Therefore, strictly speaking, the ant network has the same topology as the pruned traffic network in this paper.

[8] Recall that we can also add other costs (see Footnote 1)

[9] Since we have dynamic link costs in the ACR algorithm, the function F in (3) now should depend on the control step k_c.

with $Q > 0$ is a weight parameter.

Although ants individually strive for a user equilibrium when searching routes, the stench pheromone can push them away from the best route so as to avoid congestion. In this way, in general it yields the assignment of ants in the network results in neither a user equilibrium nor a system optimum, but instead a trade-off between two states. This is exactly the same as the overall goal that we aim at in the traffic network, i.e. to achieve a balance between minimizing the TTS and penalizing a large number of vehicles on the roads.

4.2. Colored ants

Colored ants are used to distinguish vehicles with different destinations, with color $\gamma(d)$ assigned to destination d. The entire ant network consists of (possibly) overlapping subnetworks for each color $\gamma(d)$ (see Section 5.1 for more details). For each subnetwork, the total number of ants $N_d^{\text{ant,total}}$ should depend on the structure of the subnetwork. The more nodes and links the subnetwork has, the more ants are needed to guarantee timely convergence. We assume that the number of ants $N_d^{\text{ant,total}}$ is fixed and time-invariant during the whole ACR run. Furthermore, we define a dynamic factor $\eta_{\gamma(d)}(k_{\text{c}})$ to relate the total number $N_d^{\text{ant,total}}$ of ants with color $\gamma(d)$ to the number of incoming vehicles $N_d^{\text{veh,in}}(k_{\text{c}})$ with destination d during a prediction period $[k_{\text{c}}T_{\text{c}}, (k_{\text{c}}+N_{\text{p}})T_{\text{c}}]$, with N_{p} the prediction horizon. The factor $\eta_{\gamma(d)}(k_{\text{c}})$ is defined as:

$$\eta_{\gamma(d)}(k_{\text{c}}) = \frac{N_d^{\text{ant,total}}}{N_d^{\text{veh,in}}(k_{\text{c}})}, \tag{4}$$

which indicates how many vehicles an ant with color $\gamma(d)$ represents at each control step k_{c}. On each arc in the ant network, the regular pheromone produced by colored ants is colored as well, and it only attracts the ants with the corresponding color, whereas the stench pheromone is not colored because it is used to push all of the ants away from the arc.

4.3. Stench function

Several constraints and thresholds of the number of ants are considered for each arc in the ant network, in order to include the constraints of the number of vehicles in the traffic network. For a link in the traffic network, there is a link capacity that can never be exceeded, a critical density that should preferably not be exceeded since otherwise traffic congestion may occur, and probably one or more threshold values for sensitive zones where a limit number of vehicles are allowed, e.g., hospitals and schools. All these constraints and thresholds are implemented through the stench function $G_{s,t}$ defined on each arc (s,t) in the ant network. For instance, the function $G_{s,t}$ can be specified as a piecewise affine function as (see Figure 3):

$$G_{s,t}\left(N_{s,t}^{\text{ant}}(k_{\text{c}})\right) = \max\left(0, g_1\left(N_{s,t}^{\text{ant}}(k_{\text{c}})\right), g_2\left(N_{s,t}^{\text{ant}}(k_{\text{c}})\right), g_3\left(N_{s,t}^{\text{ant}}(k_{\text{c}})\right)\right), \tag{5}$$

where $g_1(x)$, $g_2(x)$ and $g_3(x)$ are affine functions defined as:

$$\begin{aligned}
g_1(x) &= P_{s,t,1}(x - N_{s,t}^{\text{ant,thresh}}),\\
g_2(x) &= P_{s,t,2}(x - N_{s,t}^{\text{ant,crit}}) + P_{s,t,1}(N_{s,t}^{\text{ant,crit}} - N_{s,t}^{\text{ant,thresh}}),\\
g_3(x) &= P_{s,t,3}(x - N_{s,t}^{\text{ant,cap}}) + P_{s,t,2}(N_{s,t}^{\text{ant,cap}} - N_{s,t}^{\text{ant,crit}}) + P_{s,t,1}(N_{s,t}^{\text{ant,crit}} - N_{s,t}^{\text{ant,thresh}}),
\end{aligned}$$

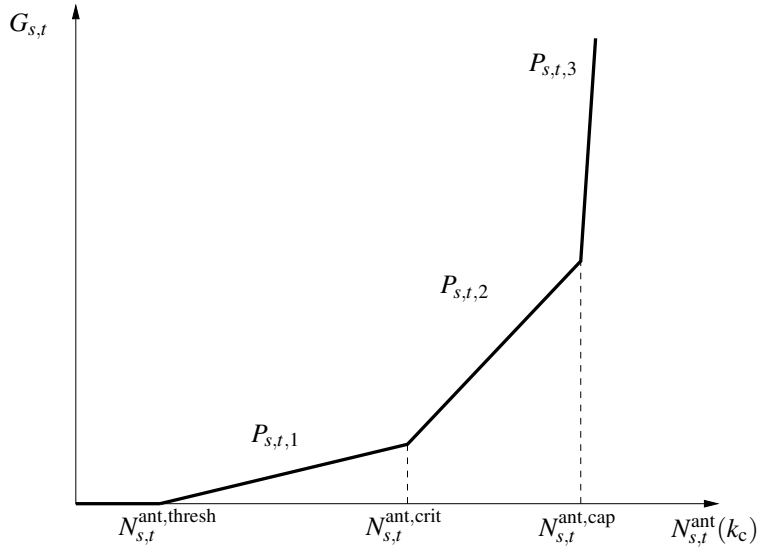

Figure 3: Piecewise affine stench functions $G_{s,t}$

where $N_{s,t}^{\text{ant,thresh}}$ denotes the threshold number[10] of ants corresponding to the threshold value for sensitive zones, $N_{s,t}^{\text{ant,crit}}$ denotes the critical number of ants corresponding to the critical density, $N_{s,t}^{\text{ant,cap}}$ denotes the capacity number of ants corresponding to the link capacity in the traffic network, and $P_{s,t,1}$, $P_{s,t,2}$, and $P_{s,t,3}$ are slopes of the affine functions in each piece of the function $G_{s,t}$, with a relationship $P_{s,t,1} < P_{s,t,2} < P_{s,t,3}$. Recall that the relationship between the number of ants with color $\gamma(d)$ and the number of vehicles with destination d in the period $[k_{\text{c}}T_{\text{c}}, (k_{\text{c}}+N_{\text{p}})T_{\text{c}}]$ is characterized by the factor $\eta_{\gamma(d)}(k_{\text{c}})$ (cf. (4)). In order to be able to add up numbers of ants for different colors, which is required for computing the stench function $G_{s,t}$, we introduce so-called *standard* ants, which have a one-to-one relationship with the vehicles. The number of standard ants on arc (s,t) for period $[k_{\text{c}}T_{\text{c}}, (k_{\text{c}}+N_{\text{p}})T_{\text{c}}]$ is calculated as:

$$N_{s,t}^{\text{ant}}(k_{\text{c}}) = \sum_{d\in\mathscr{D}} \frac{N_{s,t,d}^{\text{ant}}(k_{\text{c}})}{\eta_{\gamma(d)}(k_{\text{c}})}. \tag{6}$$

Moreover, the numbers $N_{s,t}^{\text{ant,thresh}}$, $N_{s,t}^{\text{ant,crit}}$, and $N_{s,t}^{\text{ant,cap}}$ also represent standard ants.

Remark: In practice, the value of the slopes $P_{s,t,1}$ and $P_{s,t,2}$ can be dynamically assigned to better serve and manage the traffic network. During a day, we can increase the value of $P_{s,t,1}$ and $P_{s,t,2}$ to allow fewer vehicles to enter the sensitive zones so as to avoid danger, pollution, and noise, and if really needed, we can decrease the value of $P_{s,t,1}$ and $P_{s,t,2}$ to allow more vehicles to enter the sensitive zones to guarantee the smoothness of traffic flow and to reduce the burden for other zones.

[10] According to the requirements of traffic management center, no threshold value or even two or more threshold values could be considered here.

4.4. Dynamic model

A dynamic traffic model is used to assign a dynamic cost to each arc in the ant network. There exists a wide variety of traffic models (Hoogendoorn and Bovy, 2001; Daganzo, 1997; Papageorgiou, 1990), and in general one has to make a choice among the traffic models according to the application area. For example, a static traffic model requires much less computational effort than a dynamic traffic model, while a dynamic traffic model is much more accurate than a static traffic model in describing the evolution of the traffic system. The desired traffic models in our work should satisfy the following criteria:

- The model should be fast when it is simulated on a computer if the controller is going to operate in real time;
- The model should reproduce the dynamic traffic process with sufficient accuracy.

In this paper, we primarily consider a discrete-time, discrete-space macroscopic traffic flow model. We assume each link m is divided in N_m segments of length L_m (this is e.g. also done in the METANET (Messmer and Papageorgiou, 1990) traffic model, which will be used in the case study of Section 6). Recall from Section 2 that we define k as the simulation step counter and k_c as the control step counter, and that for a given value of k_c corresponding value of the simulation step counter is given by $k = Mk_c$. The traffic model is used to predict the future traffic states $\hat{\mathbf{x}}_{m,i}(k)$ for each segment i of each link m of the traffic network by running the simulation based on the current traffic state $x_{m,i}(k)$ for each segment i of each link m, where $\hat{\mathbf{x}}_{m,i}(k)$ is a vector of traffic states at future simulation steps $k+1, k+2, \cdots, k+MN_p-1$:

$$\hat{\mathbf{x}}_{m,i}(k) = [\hat{x}_{m,i}(k+1|k)\ \hat{x}_{m,i}(k+2|k)\ \cdots\ \hat{x}_{m,i}(k+MN_p-1|k)]^T \tag{7}$$

with $\hat{x}_{m,i}(k+1|k)$ the predicted state for simulation step $k+1$ based on knowledge at simulation step k. The dynamic link cost $\varphi_{s,t}(k_c)$ of link m at control step k_c with $k = Mk_c$ is a function of both the current traffic states $x_{m,i}(k)$ and the future traffic states vector $\hat{\mathbf{x}}_{m,i}(k)$. In Section 5.2.2, we will discuss in more detail how dynamic traffic models can be used to determine the link costs with varying degrees of computational complexity and accuracy.

Remark: Although the models discussed above are discrete-time and discrete-space, the proposed approach can also easily be applied to continuous-time, continuous-space, and microscopic traffic models.

5. ACR Control Strategy

5.1. Network Pruning

The network pruning step aims at removing the "unnecessary" links and nodes in a traffic network, i.e. those links and nodes only belong to routes that are relatively long and that will therefore not be favored by drivers. This is illustrated in Figure 2, where only the routes indicated by dashed lines will be kept, and the remaining routes are removed, because they are too long and thus are "unnecessary". For each OD-pair, we separately use network pruning to obtain the K best routes to form a pruned network, with K an integer that could be different for each OD-pair. Furthermore, the pruned networks with the same destination d are used to build up the ant network with color $\gamma(d)$, and therefore we will obtain N_d subnetworks, with N_d the number of destinations. At last, we combine all the subnetworks to yield an overall ant network. The

colored ants only travel and deposit the pheromone in the subnetwork with the corresponding color, while the stench pheromone is put down in the overall ant network.

We consider three different approaches for network pruning: K-shortest routes, linear programming, and the combination of the first two methods.

The simplest way to find the K best routes is to choose the shortest routes. Several algorithms have been developed for finding the K shortest loopless paths in a network (Yen, 1971; Katoh et al., 1982; Hadjiconstantinou and Christofides, 1999). We can use these algorithms to determine the K shortest routes for each OD-pair $(o,d) \in \mathscr{O} \times \mathscr{D}$ by considering the length of each link, or the average travel time[11] on each link. Afterwards, we remove the links not belonging to any route. In this way, we obtain the pruned network. However, this method cannot guarantee that the capacity of each link on the routes is never exceeded.

One way to address the capacity issue is to use linear programming to find the links with the highest link flows to form the K best routes for each OD-pair. More specifically, we consider a quasi-static approach, where a day is divided into several time slots (e.g., the morning rush hour, the non-busy midday period, and the evening rush hour). For each time slot, we determine the traffic flows on each link m in the traffic network for each destination $d \in \mathscr{D}$ such that the total travel time is minimized. Since the quasi-static case is considered here, all segments of link m have the same flow with destination d as the flow in the first segment $q_{m,1,d}$. For each (non-virtual[12]) link $m \in \mathscr{M}$ the average travel time is defined as $t_m = L_m/v_m$ with L_m the length of link m and v_m the average speed on link m. The linear programming problem to minimize the total travel time is now defined as:

$$\min_{q_{m,1,d}} \sum_{d\in\mathscr{D}} \sum_{m\in\mathscr{M}} T \cdot t_m \cdot q_{m,1,d} \tag{8}$$

subject to

$$q_{m,d} = q^{\text{in}}_{o,d}, \quad \forall (o,d) \in \mathscr{O} \times \mathscr{D}, \forall m \in O(o) \tag{9}$$

$$\sum_{d\in\mathscr{D}} \sum_{m\in I(n)} q_{m,1,d} = \sum_{d\in\mathscr{D}} \sum_{m\in O(n)} q_{m,1,d}, \quad \forall n \in \mathscr{N} \tag{10}$$

$$\sum_{d\in\mathscr{D}} q_{m,1,d} \leqslant q^{\text{cap}}_m, \quad \forall m \in \mathscr{M} \tag{11}$$

$$q_{m,1,d} \geqslant 0, \quad \forall m \in \mathscr{M}, \forall d \in \mathscr{D}, \tag{12}$$

with T the simulation sample time, q^{cap}_m the capacity of link m, $O(o)$ the set of outgoing links of origin o, and $I(n)$ and $O(n)$ respectively the sets of incoming and outgoing links of node n. Note that (8) minimizes the total travel time, because $T \cdot q_{m,1,d}$ expresses the number of vehicles on link m per simulation step, and thus $T \cdot t_m \cdot q_{m,1,d}$ corresponds to the total travel time on link m. Moreover, (10) states that the inflow of node n equals the outflow of node n (conservation of vehicles). It is easy to verify that (8)–(12) is a linear programming problem, which can be solved very efficiently using, e.g., a simplex method or an interior-point algorithm (Nesterov and Nemirovskii, 1994; Wright, 1997). Once the solution is found, we first select only the links for

[11] These average values can be determined based on historical data. Such historical data is usually available to the traffic control centers, where ACR will be actually implemented. Also note that this data is sometimes even publicly available, see e.g. Regiolab-Delft (2013); California Department of Transportation (2013)

[12] The travel time t_m on virtual links (see Footnote 1) is set equal to 0, so that they do not contribute to the cost function.

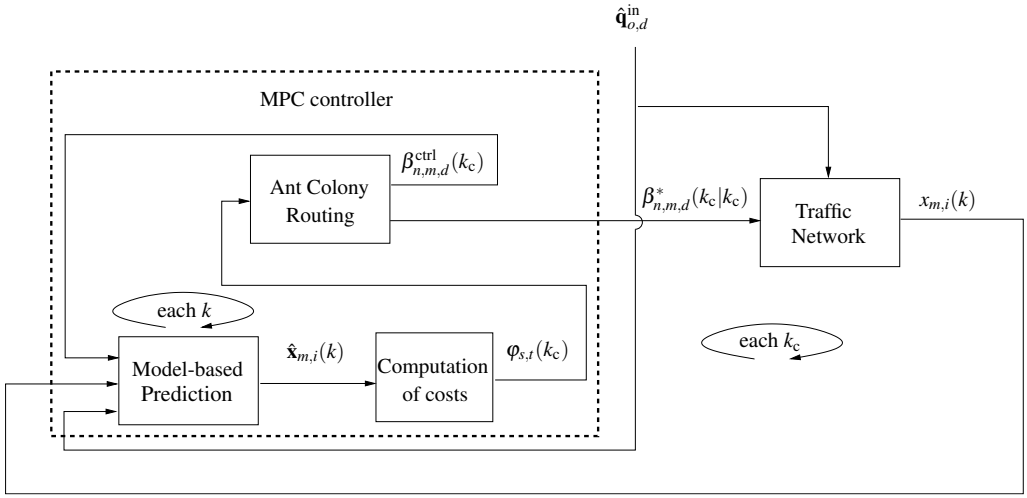

Figure 4: Closed-loop control of a traffic system with the ACR algorithm and the dynamic traffic prediction model.

which $\sum_{d\in\mathscr{D}} q_{m,1,d} \geqslant q_m^{\text{thresh}}$ with q_m^{thresh} a positive threshold value, and remove others. Next, we also remove the links that do not belong to any route from an origin o to destination d, and keep the rest as the pruned network. However, the shortcoming of this method is that possibly no route from o to d is contained in the set of the selected links. If so, then we need to decrease the value q_m^{thresh} to guarantee at least one route.

A third way, which is also the solution to address the capacity problem of the K shortest routes approach and the no-route problem of the linear programming approach is to combine them. We first select a value K, and determine the K shortest routes to construct the reduced network; next, we solve the linear programming problem (8)–(12) for the reduced network. If this linear programming problem is feasible, the capacity problem does not occur; otherwise, we augment the value K and repeat the procedure until we get a feasible linear programming problem. Sometimes, when using a K-shortest-routes algorithm, the problem may occur that there are too many overlapping links in the obtained routes. In that way, the pruned network may be insufficient to build up an ant network. If that happens, besides simply increasing the value of K to find some more candidate routes, we can use a dedicated K-shortest-routes algorithm that avoids having too many links in common between different routes (see e.g. (Zhu et al., 2010)).

5.2. Model Predictive Control

The control scheme of Model Predictive Control (MPC) (Maciejowski, 2002; Rawlings and Mayne, 2009; Hegyi et al., 2005) is shown in Figure 4. The traffic model predicts the evolution of the traffic network at every simulation step k, while the ACR algorithm only runs at each control step k_{c}. The MPC controller consists of three parts, namely model-based prediction, computation of link costs, and the ACR algorithm. Model-based prediction works on the traffic network, the ACR algorithm works on the ant network, and computation of link costs involves mapping the variables from the traffic network to costs in the ant network. Next, the operation of each part is explained.

5.2.1. Model-based prediction

We start the prediction of the traffic system at control step k_c, for which the corresponding simulation step is $k = Mk_c$. The process is described by the selected dynamic traffic model during the prediction period $[k_c T_c, (k_c + N_p)T_c]$. The prediction requires three inputs:

- the current traffic states $x_{m,i}(k)$ for each link m and each segment i at simulation time step $k = Mk_c$;
- a vector of the expected inflows[13] of the network at each simulation step during the prediction period: $\hat{\mathbf{q}}^{\text{in}}_{o,d}(k) = [q^{\text{in}}_{o,d}(k)\ q^{\text{in}}_{o,d}(k+1)\ \cdots\ q^{\text{in}}_{o,d}(k+MN_p-1)]^T$ for all OD-pairs $(o,d) \in \mathscr{O} \times \mathscr{D}$;
- the currently imposed control signal — splitting rates $\beta_{n,m,d}(k_c(k))$ (see Section 5.3 for the way they are computed) for node n, outgoing link m, and destination d, at control step $k_c = \text{floor}(k/M)$.

The output of model-based prediction is the future traffic state vector $\hat{\mathbf{x}}_{m,i}(k)$ during a prediction period $[k_c T_c, (k_c + N_p)T_c]$ defined in (7), which includes the states from simulation step $k+1$ to simulation step $k + MN_p - 1$ based on the knowledge at simulation step k. The state $\hat{\mathbf{x}}_{m,i}(k)$ can be number of vehicles, flow, speed, density, emission, etc, according to different traffic models.

5.2.2. Computation of costs

The cost $\varphi_{s,t}(k_c)$ on each arc (s,t) in the ant network is translated from the traffic state $\hat{\mathbf{x}}_{m,i}(k)$ in the traffic network at each control step k_c. In this paper, only TTS is considered in the cost. We present two different methods to calculate $\varphi_{s,t}(k_c)$, called quasi-static and fully-dynamic, respectively. The main difference between the two cases is whether or not the cost $\varphi_{s,t}(k_c)$ is being updated while we run the ACR algorithm.

Quasi-static case

The quasi-static case first uses the predicted traffic state $\hat{\mathbf{x}}_{m,i}(k)$, in particular the speed $v_{m,i}(k)$ in this paper, to calculate the travel time on segment i of link m, for all simulation steps $k = Mk_c, Mk_c+1, \ldots, M(k_c+N_p)-1$ in the prediction period $[k_c T_c, (k_c+N_p)T_c]$, and then sums up the travel time on all the segments of all the links for all the simulation steps to calculate $\varphi_{s,t}(k_c)$, which is formulated as follows:

$$\varphi_{s,t}(k_c) = \varphi_{\ell^{-1}(m)}(k_c) = \sum_{k=Mk_c}^{M(k_c+N_p)-1} \sum_{i=1}^{N_m} \frac{L_m/N_m}{v_{m,i}(k)} \quad , \tag{13}$$

As a matter of fact, the cost $\varphi_{s,t}(k_c)$ is kept fixed at each control step k_c when we run the ACR algorithm to let ants travel in the ant network. At the end of each iteration of ACR, we update the speed $v_{m,i}(k)$ based on the new splitting rates (cf. (17)), resulting from the ACR iteration, so as to obtain new cost $\varphi_{s,t}(k_c)$, and start a new iteration of the ACR algorithm.

Fully-dynamic case

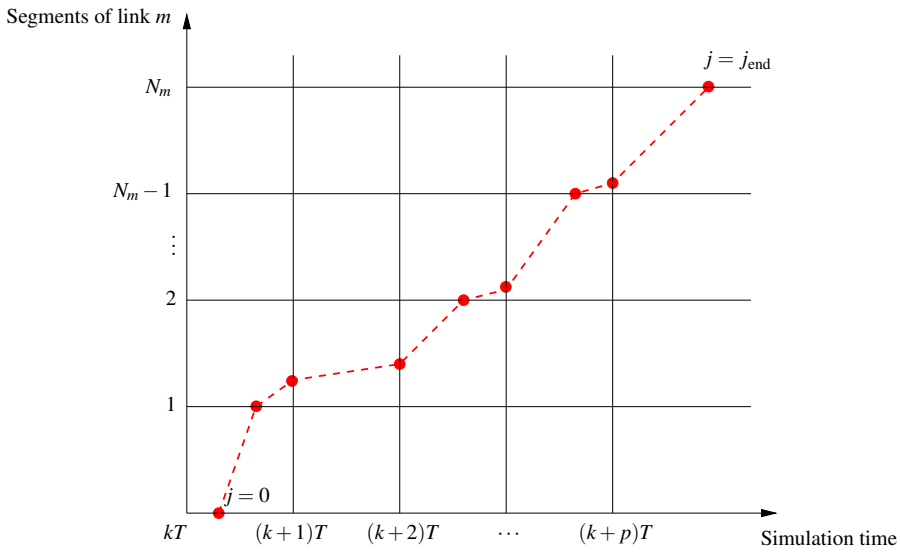

Figure 5: Ant a tracks the travel time on link m

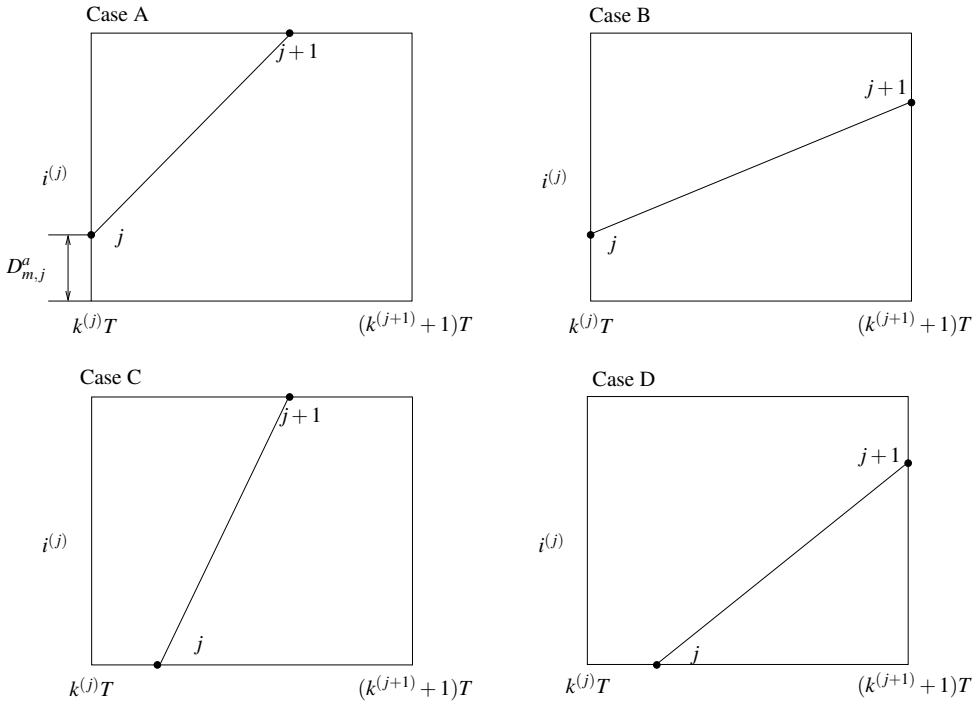

Figure 6: Four possible ways in which ant a on link m can jump from point j to point $j+1$ between the time intervals kT and $(k+1)T$

In the fully-dynamic case, the cost $\varphi_{s,t}(k_c)$ varies during the ACR run. Therefore, for a given arc (s,t), each ant a traveling on it will incur a possibly different cost $\varphi^a_{s,t}(k_c) = \varphi^a_{\ell^{-1}(m)}(k_c)$. When determining the values of the fully dynamic link costs, we will — for the sake of simplicity of the exposition — consider the ants to travel on the links of the traffic network. Figure 5 describes how we can track the travel time predicted by ant a according to the dynamic speed profile $v_{m,i}(k)$ for all link-segment pairs (m,i), and for the simulation time steps k. Since the speed of the traffic model varies in different segments and at different time steps, we call the point where the speed $v_{m,i}(k)$ changes to a new speed $v_{m,i}(k+1)$ or $v_{m,i+1}(k)$ a *jump point* (denoted by j). Obviously, jump points can only placed on the boundary of segments or on the simulation time steps. If a jump point j is placed in the segment i, we denote segment i as $i^{(j)}$ to illustrate the relationship between jump point j and segment i. Similarly, if a jump point j is placed at the simulation step k, we denote the simulation step k as $k^{(j)}$.

At a given jump point j, an ant a predicts the travel time between jump points j and $j+1$, and then jumps to $j+1$ to make a new prediction. Ants will keep this *predict-jump-predict* mode until they reach the end of the link m, which yields the last jump point j_{end}. The time instant when ant a on link m is at jump point j is denoted as $t^a_{m,j}(k_c)$. Figure 6 illustrates the four possible transitions of ant a from jump points j to $j+1$. For each case, the time instant $t^a_{m,j+1}(k_c)$ is calculated as follows:

Case A. $$t^a_{m,j+1}(k_c) = t^a_{m,j}(k_c) + \frac{L_m - D^a_{m,j}(k_c)}{v_{m,i^{(j)}}\left(k^{(j)}\right)}$$

Case B. $$t^a_{m,j+1}(k_c) = \left(k^{(j+1)} + 1\right) T$$

Case C. $$t^a_{m,j+1}(k_c) = t^a_{m,j}(k_c) + \frac{L_m}{v_{m,i^{(j)}}\left(k^{(j)}\right)}$$

Case D. $$t^a_{m,j+1}(k_c) = \left(k^{(j+1)} + 1\right) T$$

where $D^a_{m,j}(k_c)$ is the distance between the beginning of segment $i^{(j)}$ and jump point j as shown in Figure 6. We denote the distance between jump points j and $j+1$ as $d^a_{m,j}$, which is expressed by:

$$d^a_{m,j}(k_c) = v_{m,i^{(j)}}\left(k^{(j)}\right)\left(t^a_{m,j+1}(k_c) - t^a_{m,j}(k_c)\right). \tag{14}$$

Therefore, the total distance between the jump point $j=0$ and current jump point j is the sum $\sum_{z=0}^{j-1} d^a_{m,z}$. Besides, the length of each segment of link m equals L_m. The distance $D^a_{m,j}(k_c)$ can then be obtained by:

$$D^a_{m,j}(k_c) = \operatorname{mod}\left(\sum_{z=0}^{j-1} d^a_{m,z}(k_c), L_m\right), \tag{15}$$

[13] These can be obtained from historical data or from data measured upstream.

where mod denotes the modulo operator. Note that in Case C, we can consider that $D^a_{m,j}(k_c) = 0$, so the equation of Case A also holds for Case C. Hence, a unified formulation of the time instant $t^a_{m,j}(k_c)$ for all four cases is:

$$t^a_{m,j+1}(k_c) = \min\left(t^a_{m,j}(k_c) + \frac{L_m - D^a_{m,j}(k_c)}{v_{m,i^{(j)}}\left(k^{(j)}\right)}, \left(k^{(j+1)}+1\right)T\right). \tag{16}$$

We record the time instant $t^a_{m,0}(k_c)$ when ant a enters link m, and use (16) to calculate the time instant $t^a_{m,j_{\text{end}}}(k_c)$ of the last jump point j_{end}. The cost $\varphi^a_{s,t}(k_c) = \varphi^a_{\ell^{-1}(m)}(k_c)$ on arc (s,t) is then the travel time predicted by ant a:

$$\varphi^a_{s,t}(k_c) = \varphi^a_{\ell^{-1}(m)}(k_c) = t^a_{m,j_{\text{end}}}(k_c) - t^a_{m,0}(k_c).$$

5.3. ACR run

The ACR algorithm is used as the optimization method in the MPC controller, aiming at generating the splitting rates of the traffic flow to each destination $d \in \mathscr{D}$ at each intermediate node $n \in \mathscr{N}$ in the traffic network. The procedure of the ACR algorithm has been introduced in Section 4, and after the ACR algorithm has terminated, the number of the ants $N^{\text{ant}}_{s,t,d}$ of color $\gamma(d)$ that have traveled on each link (s,t) is determined. These values are used to update the splitting rates for each $m \in O(m)$, each $d \in \mathscr{D}$, each $n \in \mathscr{N}$:

$$\beta_{n,m,d}(k_c(k)) = \frac{N^{\text{ant}}_{s,t,d}(k_c)}{\sum_{t' \in \mathscr{S}_{s,d}} N^{\text{ant}}_{s,t',d}(k_c)} \quad , \text{ with } m = \ell(s,t) \tag{17}$$

with $\mathscr{S}_{s,c}$ the set of the nodes in the ant network that are connected to node s for color $\gamma(d)$. Moreover, if a link m in the traffic network does not have a corresponding arc in the ant network for destination d, we set $\beta_{n,m,d}(k_c(k)) = 0$. All of the resulting splitting rates will be applied to the prediction model as mentioned in Section 5.2.1, and correspondingly we will obtain new traffic states for updating the cost $\varphi_{s,t}(k_c)$, as well as new values of $N^{\text{ant}}_{s,t,d}(k_c)$. We repeatedly run the prediction-ACR updating process until one of the criteria below (or their combination) is satisfied:

1. The maximum number of iteration steps N_{fixed} is reached;
2. $|\Delta\beta_{n,m,d}(k_c(k))| < \varepsilon_\beta$, for all n, m, and d, where $\Delta\beta_{n,m,d}(k_c(k))$ is the difference between the new and the previous value of $\beta_{n,m,d}(k_c(k))$, and $\varepsilon_\beta > 0$ is a predefined tolerance.

Then the final splitting rates are the outputs of the MPC controller, and they are used to control the real traffic system via lower level controllers, such as dynamic matrix panels with route information, dynamic tolling, interaction with on-board route guidance devices, and so on. In the future, the approach can also be integrated with autonomous vehicle systems (Al-Hasan and Vachtsevanos, 2002), where a full compliance of the routing instructions can be easily applied. However, the exact implementation of the route guidance mechanism is outside the scope of this paper (see, e.g., (Bishop, 2005) for more information).

Figure 7: Map of Walcheren area (source: google maps)

6. Case study

The ACR algorithm is now tested in a simulation of the Walcheren area in Zeeland, the Netherlands (Huibregtse et al., 2011). First, the set-up of the case study and the routing scenario is described in Section 6.1. In Section 6.2, we show the result of the pruning step, which consists in finding a reduced network for each destination. To analyze the ACR algorithm, we first show the convergence performance of the optimization method in Section 6.3, and then compare a congested traffic situation with and without ACR control in Section 6.4. Finally, a comparison between the ACR algorithm and two other dynamic routing methods is shown in Section 6.5.

6.1. Simulation scenario

A map of Walcheren area is shown in Figure 7. Middelburg at its center is the provincial capital and the biggest municipality on the area. Vlissingen in the south is the main harbor and the second biggest municipality. The third biggest municipality is Veere in the north-east. Moreover, the whole area is connected to the mainland by three main freeways, which are the N57 in the north, the E312 in the middle, and the N254 in the south. The entire freeway network has 142 links and 62 nodes, including the origins and the destinations.

For the purpose of our study, we consider a scenario that drivers outside Walcheren enter the area only through the N57, the E312, or the N254, and they are going to Middelburg, Vlissingen, or Veere by only using the freeway network. The origins of the traffic network are put at the entrances of the N57 (o_1), the E312 (o_2), and the N254 (o_3), indicated by the large white dots

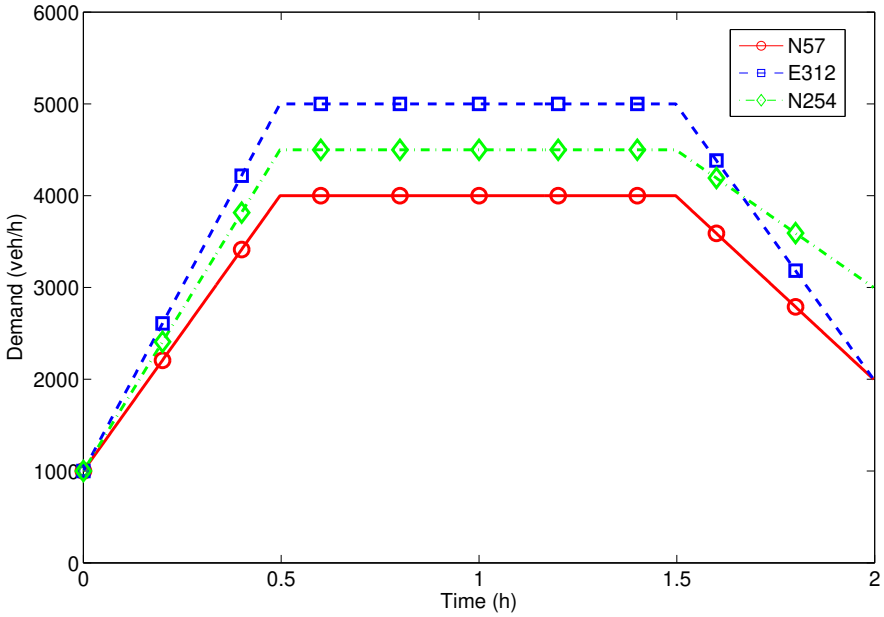

Figure 8: Inflows of the freeway network from the different origins

Table 2: Fractions of traffic flows to different destinations

	d_1	d_2	d_3
o_1	30%	30%	40%
o_2	50%	30%	20%
o_3	20%	40%	40%

in Figure 7. The destination nodes are put at the exits of the freeway network in Middelburg (d_1), Veere (d_2), and Vlissingen (d_3), indicated by the large red, blue and green dots respectively. The simulation period is set to 2 hours, with an empty network as the initial state. The inflow from each origin is shown in Figure 8, and the fractions of the inflows traveling to the different destinations are shown in 2.

For this scenario, the routing instructions are optimized by the ACR approach proposed in this paper. Although we discussed two different ways to compute the link cost in the network, and the fully-dynamic case is much more accurate than the quasi-static case, we only implement the quasi-static case in the case study. This is because the fully dynamic case accumulates the travel time on each link by every jump point for the link cost, which results in a very high computational burden. On contrary, in the quasi-static case we only need to calculate the average travel time on each link, and in that way we can achieve a balanced trade-off between computation speed and accuracy.

The parameter settings of the ACR algorithm are given in Table 3. These settings were determined by using manual tuning and they were found to perform well in the case study.

Table 3: Values of the parameters in the ACR algorithm

Symbol	Explanation	Value
$N_d^{\text{ant,total}}$	Total number of ants for each color	3000
Q	Weight parameter of regular pheromone	70
P_1	Slope of stench function in non-sensitive zones	1
P_2	Slope of stench function in sensitive zones	2
ρ_{evap}	Evaporation rate of pheromone	0.1

6.2. Network pruning

We first use Yen's K-shortest-loopless routes algorithm (Yen, 1971) to find the K shortest routes for each OD pair. To guarantee a suitable-sized reduced network, the initial value of K is set to 3 in the case study. We solve linear programming problem (8)–(12) to check whether the flow on any link of these routes exceed the link capacity. As a result, the link capacity is not exceeded, and thus we do not need to augment the value of K in this case. Because colored ants in the ACR algorithm are only distinguished by destinations, we combine all the routes with the same destination together as one pruned network as the preparation step for the ACR algorithm. As a result, we find three different pruned networks for different destinations as shown in Figure 11. The first pruned network has 26 links and 21 nodes, the second pruned network has 21 links and 9 nodes, and the third pruned network has 30 links and 26 nodes.

6.3. Performance of the optimization method

The performance of the ACR algorithm is evaluated using the convergence speed of the pheromone levels. Figure 10 shows an example of the evolution of three different colored pheromone levels on a representative link, indicated by "RepLink" in Figure 7. The representative link has been chosen because it belongs to all three subnetworks. Moreover, this link has different statuses in different subnetworks:

1. It belongs to the shortest route from origin o_1 to destination d_3 in the green subnetwork, and it is shared by all the routes from o_1 to d_3;
2. It belongs to the shortest route from origin o_1 to destination d_1 in the red subnetwork, but there is an alternative route that does not include the representative link;
3. It does not belong to the shortest route from origin o_1 to destination d_2 in the blue subnetwork.

From Figure 10, we can observe that only the values of green pheromone are positive, while the values of both red and blue pheromone are negative, which means the representative link mainly attracts the green ants.

In every optimization cycle, ants get a maximum of 1000 iteration steps, but all pheromone trajectories level off after iteration step 60, with a slight fluctuation in the following iteration steps. The reason that the pheromone levels still fluctuate slightly around their steady state values is that the ACR algorithm involves a stochastic mechanism, in which a small portion of ants try to explore new routes at each iteration. In our experiment, convergence occurred on the average after about 850 iteration steps, due to reaching the threshold on the error. The maximum number of iterations was only reached in 19 out of 240 runs.

The ACR algorithm is programmed in Matlab by using a desktop computer with an Intel(R) Core(TM) 2 Duo CPU with 3.00 GHz and 4GB RAM. Currently the optimization method is

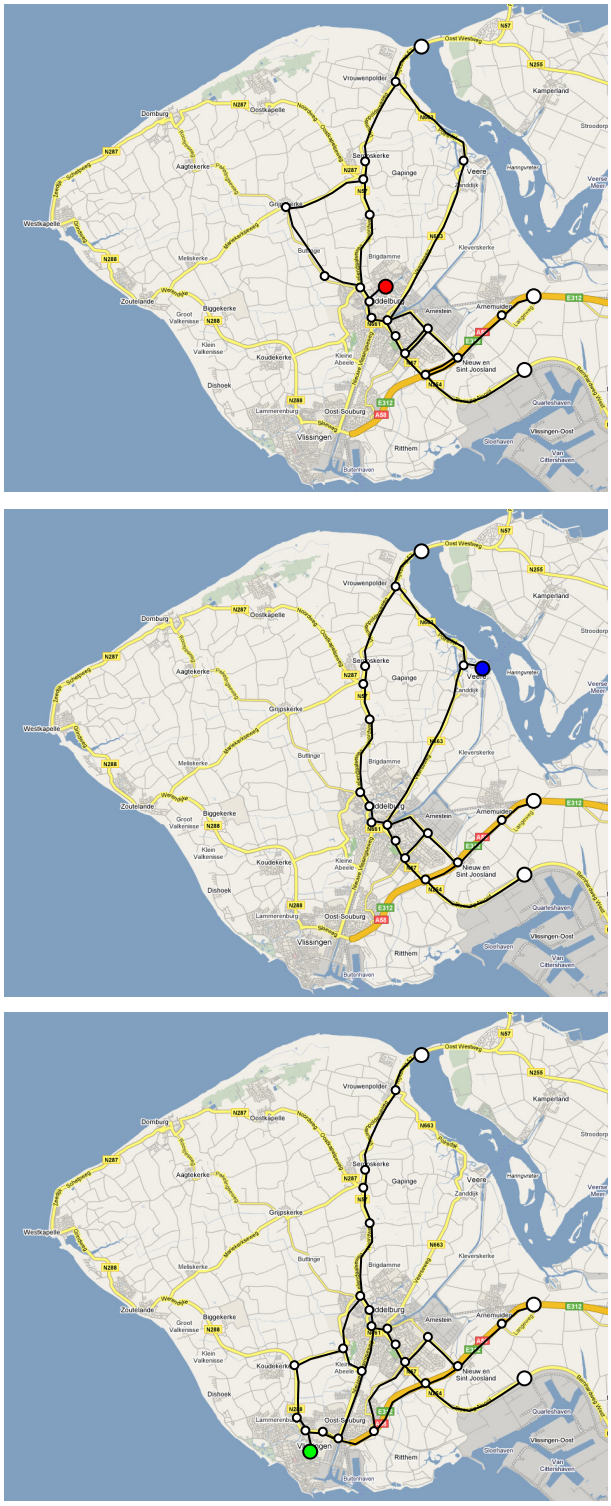

Figure 9: Pruned networks for different destinations

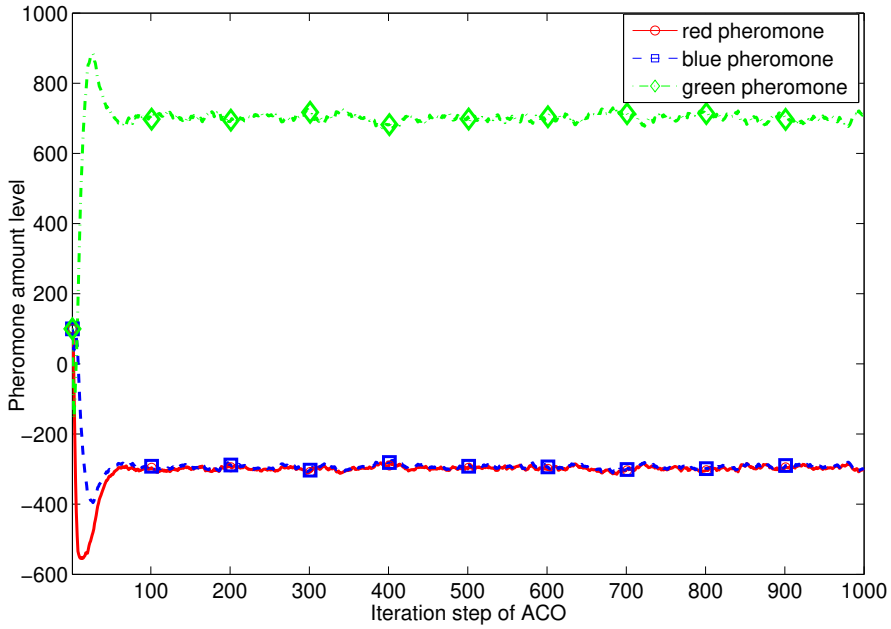

Figure 10: Convergence of pheromone levels on the representative link

programmed serially, so the most computationally intensive part is using ants one after another to randomly search routes.

6.4. Routing results for ACR

In this paper, we assume that the resulting routing instructions will be perfectly followed by the drivers. The reason for this is that the primary objective of this paper is to discuss the functioning of the ACR algorithm. By assuming full compliance, the functioning can be more easily compared with other methods. In practice, in order to convince drivers to follow the suggested route guidance instructions, a financial compensation or penalty can be applied for the system sake. For instance, drivers have to pay if they travel in so-called congestion charge zones, while they need not pay if they travel outside of these areas Small and Gómez-Ibanez (1998), (see e.g. congestion charge schemes in London, Stockholm and Singapore), while in the Netherlands, the government is testing a reward system called Spitsmijden Bliemer and van Amelsfort (2010) that tracks commuters and pays a range of rewards (€3, €5, and €7 per day) to those who avoided traveling during the morning peak (7:30 am to 9:30 am).

As shown in Figure 11, we compare the traffic situations in a representative link, indicated by "RepLink" in Figure 7, with and without the ACR control. When there is no control, we assume that all the drivers always choose the shortest distance route. When they converge to the same links, the maximum of the density on the link is reached, and the speed is quite low. The oscillation of solid lines is caused by traffic congestion. However, by using the ACR algorithm, no traffic congestion will occur.

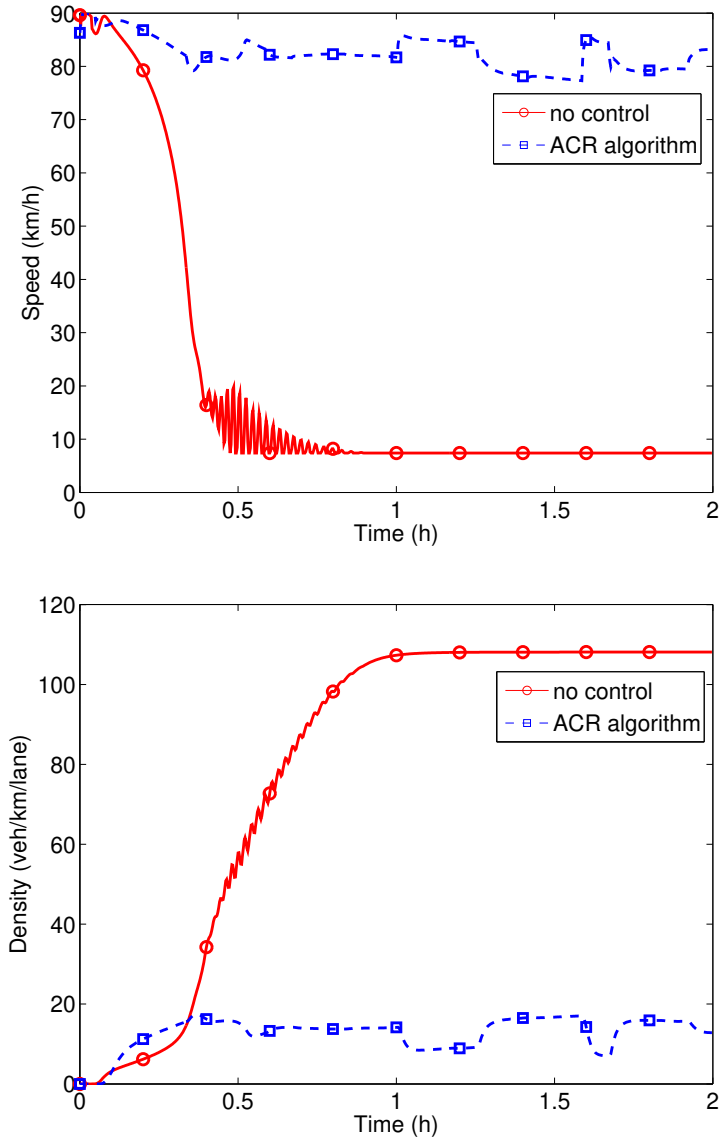

Figure 11: Congested traffic situation with and without ACR control on the representative link

6.5. Results of the comparison with other methods

To evaluate the performance and the computational efficiency of the ACR algorithm, we compare our approach with two other dynamic traffic routing methods, namely non-linear optimal control (Kotsialos et al., 2002) and the time-dependent shortest routes method (Tong and Wong, 2000).

The non-linear optimal control method uses a similar control framework as our approach, as well as the same traffic model. The major difference is that Kotsialos et al. (2002) consider the problem as a discrete-time optimal control problem, and use a numerical non-linear optimization algorithm to solve it. For non-linear optimal control method, the problem is formulated as:

$$\begin{aligned}\min\ J =& J_{\mathrm{TTS}} + J_{\mathrm{pen}} \\ =& T \cdot \sum_{k=1}^{K_{\mathrm{sim}}} \sum_{m \in \mathscr{M}} \sum_{i \in \mathscr{I}_m} \rho_{m,i}(k) \frac{L_m \lambda_m}{N_m} + P \cdot \sum_{k=1}^{K_{\mathrm{sim}}} \sum_{m \in \mathscr{M}} \sum_{i \in \mathscr{I}_m} \left(\rho_{m,i}(k) - \rho_m^{\mathrm{thresh}} \right) \frac{L_m \lambda_m}{N_m} \end{aligned} \tag{18}$$

where T is the length of the simulation time step, P is a penalty factor, $\rho_{m,i}(k)$ is the density on segment i of link m at time step k, ρ_m^{thresh} is the threshold density on link m, L_m is the length of link m, λ_m is the number of lanes of link m, K_{sim} is the simulation horizon, $\mathscr{M}$ is the set of all the links in the network, and $\mathscr{I}_m$ is the set of the segments in the link m . The objective function (18) is not exactly the same as the one that Kotsialos et al. (2002) formulated, because we have a different goal from theirs. They only aim at the minimal the TTS, while we want to find a balance between minimizing TTS and penalizing a large numbers of vehicles on the links. The first term of J is denoted by J_{TTS}, which corresponds to the total time spent by all the vehicles in the network. The second term, denoted by J_{pen}, represents a soft constraint and works similarly as the stench function in the ACR algorithm, for penalizing the number of vehicles exceeding a threshold number on each link. The value of ρ_m^{thresh} is set to either the critical density (in non-sensitive zones) or to a pre-defined value (in sensitive zones). The objective function J is minimized subject to the following constraints:

$$\begin{aligned} & x(k+1) = f(x(k), \beta(k_{\mathrm{c}}(k)), d(k)), \\ & 0 \leqslant \beta(k_{\mathrm{c}}(k)) \leqslant 1, \\ & \text{for } \ k = 1, 2, \ldots, K_{\mathrm{sim}}, \end{aligned}$$

where function f represents the traffic model, $x(k)$ is the traffic state at simulation step k, $d(k)$ is the demand, $\beta(k_{\mathrm{c}}(k))$ is the splitting rate used as the control variable at the control step $k_{\mathrm{c}}(k)$, with $k_{\mathrm{c}}(k)$ the value of the control step counter k_{c} corresponding to simulation step k, and K_{sim} and is the simulation horizon. Moreover, the optimality conditions are expressed in terms of the discrete-time Hamilton function, fulfilled by the Karush-Kuhn-Tucker conditions, and a numerical gradient-based algorithm is used to solve the discrete-time optimal control problem (see (Kotsialos et al., 2002) for more details).

To solve the dynamic user equilibrium problem, the time-dependent shortest routes approach uses an iterative algorithm, in which the total traffic demand is incrementally distributed over more and more routes. Each iteration consists of three steps. In step 1, all links get assigned the free-flow travel time in the first iteration, while in subsequent iterations, a simple traffic model based on the speed-density fundamental diagram is used to determine the travel times on links that carry traffic (other links still get assigned the free-flow travel time). In step 2, Dijkstra's shortest path algorithm is applied to search the shortest route from each origin to each destination,

Table 4: Comparison of the results obtained with different algorithms for one MPC step

Methods	J_{TTS} [veh· s]	J_{pen} [veh]	Computation time [s]
ACR algorithm	3.8316×10^4	0	5.11×10^4
non-linear optimal control	3.8318×10^4	7.903	2.31×10^5
time-dependent shortest paths algorithm	4.7116×10^4	6.303×10^2	2.86×10^2

in which the average travel time is used as link cost instead of the length of each link. Step 3 assigns all traffic to the new shortest route in the first iteration, while in subsequent iterations, traffic from previously selected routes is partly redistributed to the new shortest route. Next, a new iteration starts (see (Tong and Wong, 2000) for more details of the algorithm). In this way, the newly generated route always has a shorter travel time than the previously generated ones in current iteration.

Similarly to our approach, the second approach does not have an explicit objective function like J in (18). In order to compare the three approaches, we define an assessment function J_{eval} in the same manner, $J_{\text{eval}} = J_{\text{TTS}} + J_{\text{pen}}$. The comparison of results is shown in Table 4.

We show the results in one MPC step with 10 iteration steps of optimization. Compared with the non-linear optimal control, we can see in Table 4 that the ACR approach can achieve an even slightly better performance, 3.8316×10^4 veh·s versus 3.8318×10^4 veh·s in J_{TTS}, and 0 veh versus 7.903 veh in J_{pen}, while it requires one order of magnitude less computation time: 5.11×10^4 seconds versus 2.31×10^5 seconds. The fact that the values for J_{TTS} between ACR and non-linear optimal control are almost exactly the same is probably a coincidence, but it shows that ACR can yield almost the same performance as the established optimal control approach of Kotsialos et al. (2002), which requires much more computation time. Moreover, according to the nature of the ACO algorithm, ants can independently search the network at the same time, so we can run the ACR algorithm in parallel if we have enough processors. In that case, the computation time can be reduced even more dramatically. For instance, in our case study, we use a single core computer for simulation, in which 3000 ants search the network, and consequently 5.11×10^4 seconds is spent. If we use 3000 processors with each processor representing one ant, then theoretically we only need about 17 seconds to finish the optimization.

Compared with the time-dependent shortest routes algorithm, although our approach is slower, 5.11×10^4 seconds versus 2.86×10^2 seconds, we have a better performance, 3.8316×10^4 versus 4.7116×10^4 in J_{TTS}, and 0 versus 6.303×10^2 in J_{pen}. Therefore, compared with those two other approaches, the ACR algorithm can achieve a balanced trade-off between accuracy and computational efficiency, which is needed for on-line model-based traffic control. When solving the dynamic routing problem, one should consider a balance between required computation time and performance among these three methods. More specifically, if the computation time is more important than the performance, then the time-dependent shortest paths algorithm is recommended. Otherwise, ACR should be used since it requires less computation time than the non-linear optimal control.

7. Conclusions and Future Work

We have proposed a novel Ant Colony Routing algorithm for solving the dynamic traffic routing problem. The ACR algorithm uses artificial ants to search in the ant network, and the resulting assignment of ants is used to determine the splitting rates in the traffic network. We

apply the ACR algorithm in a two-step control approach: network pruning and Model Predictive Control. Through removing some "unnecessary" links and routes, the network pruning part can reduce the size of the objective network such that ants can more efficiently search in the network. The MPC control part uses the novel ACR algorithm with the stench pheromone and colored ants to efficiently guide the vehicles from multiple origins and to multiple destinations. A simulation-based case study has been tested in the Walcheren area in the Netherlands. The results show that the ACR algorithm is suitable for on-line optimization, and can achieve a well-balanced trade-off between control performance and computational speed.

Future work will include more detailed case studies for an extensive assessment of the performance and efficiency of the ACR algorithm, including different origin-destination flow scenarios. Moreover, further improvement of the control strategy will be considered, e.g., the network pruning algorithm can be executed after each ACR run as a refinement step, instead of just applying it once at the beginning as a preparation step of the ACR algorithm. On the theoretical side, we will analyze the convergence properties and scalability of the ACR algorithm. We will also derive general rules of thumb to select appropriate values for the tuning parameters and guidelines to transform traffic policy preferences into appropriate values for the thresholds of the stench pheromone function.

Acknowledgments

Research supported by the China Scholarship Council (CSC), the European COST Action TU1102, the European 7th Framework Network of Excellence "Highly-complex and networked control systems (HYCON2)", and the Transport Research Center Delft.

References

Al-Hasan, S., Vachtsevanos, G., 2002. Intelligent route planning for fast autonomous vehicles operating in a large natural terrain. Robotics and Autonomous Systems 40, 1–24.

Alves, D., van Ast, J., Cong, Z., De Schutter, B., Babuška, R., 2010. Ant colony optimization for traffic dispersion routing, in: Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC 2010), Madeira Island, Portugal. pp. 683–688.

Bishop, R., 2005. Intelligent Vehicle Technology and Trends. Artech House, Boston, Massachusetts.

Bliemer, M.C.J., van Amelsfort, D.H., 2010. Rewarding instead of charging road users: a model case study investigating effects on traffic conditions. European Transport\Trasporti Europei , 23–40.

Bottom, J., Ben-Akiva, M., Bierlaire, M., Chabini, I., Koutsopoulos, H.N., Yang, Q., 1999. Investigation of route guidance generation issues by simulation with DynaMIT, in: Ceder, A. (Ed.), Transportation and Traffic Theory. Proceedings of the 14th International Symposium on Transportation and Traffic Theory, Pergamon. pp. 577–600.

California Department of Transportation, 2013. Traffic data of California. <http://traffic-counts.dot.ca.gov>.

Carey, M., Subrahmanian, E., 2000. An approach to modeling time-varying flows on congested networks. Transportation Research Part B: Methodological 34, 157–183.

Cong, Z., De Schutter, B., Babuška, R., 2011. 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. pp. 1369–1374.

Daganzo, C.F., 1997. Fundamentals of Transportation and Traffic Operations. Pergamon.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 26, 1–13.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. Bradford Company, Scituate, MA, USA.

Hadjiconstantinou, E., Christofides, N., 1999. An efficient implementation of an algorithm for finding K shortest simple paths. Networks 34, 88–101.

Hegyi, A., De Schutter, B., Hellendoorn, J., 2005. Optimal coordination of variable speed limits to suppress shock waves. IEEE Transactions on Intelligent Transportation Systems 6, 102–112.

Hoogendoorn, S.P., Bovy, P.H.L., 2001. State-of-the-art of vehicular traffic flow modelling. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 215, 283–303.

Huibregtse, O., Hoogendoorn, S., Hegyi, A., Bliemer, M., 2011. A method to optimize evacuation instructions. OR Spectrum 33, 595–627.

Katoh, N., Ibaraki, T., Mine, H., 1982. An efficient algorithm for K shortest simple paths. Networks 12, 411–427.

Kotsialos, A., Papageorgiou, M., Mangeas, M., Haj-Salem, H., 2002. Coordinated and integrated control of motorway networks via non-linear optimal control. Transportation Research Part C: Emerging Technologies 10, 65–84.

Maciejowski, J.M., 2002. Predictive Control with Constraints. Prentice Hall, Harlow, England.

Mahmassani, H., Peeta, S., 1993. Network performance under system optimal and user equilibrium dynamic assignments: Implications for advanced traveler information systems. Transportation Research Record 1408, 83–93.

Mahmassani, H., Peeta, S., 1995. System optimal dynamic assignment for electronic route guidance in a congested traffic network. Urban Traffic Networks: Dynamic Flow Modeling and Control 1408, 2–27.

Messmer, A., Papageorgiou, M., 1990. METANET: A macroscopic simulation program for motorway networks. Traffic Engineering and Control 31, 466–470.

Nesterov, Y., Nemirovskii, A., 1994. Interior-Point Polynomial Algorithms in Convex Programming. SIAM, Philadelphia, Pennsylvania.

Papageorgiou, M., 1990. Dynamic modeling, assignment, and route guidance in traffic networks. Transportation Research Part B 24B, 471–495.

Paz, A., Peeta, S., 2009a. Behavior-consistent real-time traffic routing under information provision. Transportation Research Part C: Emerging Technologies 17, 642–661.

Paz, A., Peeta, S., 2009b. Information-based network control strategies consistent with estimated driver behavior. Transportation Research Part B: Methodological 43, 73–96.

Peeta, S., Ziliaskopoulos, A., 2001. Foundations of dynamic traffic assignment: The past, the present and the future. Networks and Spatial Economics 1, 233–265.

Ran, B., Boyce, D., LeBlanc, L., 1993. A new class of instantaneous dynamic user-optimal traffic assignment models. Transportation Research Record 41, 192–202.

Rawlings, J.B., Mayne, D.Q., 2009. Model Predictive Control: Theory and Design. Nob Hill Publishing, Madison, Wisconsin.

Regiolab-Delft, 2013. Traffic data of South Holland. <http://www.regiolab-delft.nl>.

Rivero, J., Cuadra, D., Calle, J., Isasi, P., 2012. Using the ACO algorithm for path searches in social networks. Applied Intelligence 36, 899–917.

Salari, E., Eshghi, K., 2008. An ACO algorithm for the graph coloring problem. International Journal of Contemporary Mathematical Sciences 3, 293–304.

Small, K.A., Gómez-Ibanez, J.A., 1998. Road pricing for congestion management: The transition from theory to policy, in: Button, K.J., Verhoef, E.T. (Eds.), Road Pricing, Traffic Congestion and the Environment: Issues of Efficiency and Social Feasibility. Edward Elgar, Cheltenham, UK, pp. 213–246.

Stano, P., Lendek, Z., Braaksma, J., Babuska, R., de Keizer, C., den Dekker, A., 2013. Parametric bayesian filters for nonlinear stochastic dynamical systems: A survey. IEEE Transactions on Cybernetics PP, 1–18.

Stützle, T., Dorigo, M., 1999. ACO algorithms for the traveling salesman problem, in: Neittaanmäki, P., Periaux, J., Miettinen, K., Mäkelä, M.M. (Eds.), Evolutionary Algorithms in Engineering and Computer Science. Wiley, Chichester, UK, p. 163–183.

Tatomir, B., Rothkrantz, L., 2006. Hierarchical routing in traffic using swarm-intelligence, in: Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems (ITSC 2006), Toronto, Canada. pp. 230–235.

Tong, C.O., Wong, S.C., 2000. A predictive dynamic traffic assignment model in congested capacity-constrained road networks. Transportation Research Part B: Methodological 34, 625–644.

Wardrop, J.G., 1952. Some theoretical aspects of road traffic research. Proceedings of the Institute of Civil Engineers, Part II 1, 325–378.

Wright, S.J., 1997. Primal-Dual Interior Point Methods. SIAM, Philadelphia, Pennsylvania.

Yen, J.Y., 1971. Finding the K shortest loopless paths in a network. Management Science 17, 712–716.

Zhou, X., Qin, X., Mahmassani, H.S., 2003. Dynamic origin-destination demand estimation with multiday link traffic counts for planning applications. Transportation Research Record: Journal of the Transportation Research Board 1831, 30–38.

Zhu, W., Boriboonsomsin, K., Barth, M., 2010. Defining a freeway mobility index for roadway navigation. Journal of Intelligent Transportation Systems 14, 37–50.

Ziliaskopoulos, A., 2000. A linear programming model for the single destination system optimum dynamic traffic assignment problem. Transportation Science 34, 37–49.