Delft University of Technology

Delft Center for Systems and Control

Technical report 10-043

Ant colony optimization for traffic dispersion routing*

D. Alves, J. van Ast, Z. Cong, B. De Schutter, and R. Babuška

If you want to cite this report, please use the following reference instead:
D. Alves, J. van Ast, Z. Cong, B. De Schutter, and R. Babuška, "Ant colony optimization for traffic dispersion routing," *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC 2010)*, Madeira Island, Portugal, pp. 683–688, Sept. 2010.

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.deschutter.info/abs/10_043.html`

Ant Colony Optimization for Traffic Dispersion Routing

Diogo Alves, Jelmer van Ast, Zhe Cong, Bart De Schutter, and Robert Babuška

Abstract— **Ant Colony Optimization (ACO) has proven to be a very powerful optimization heuristic for combinatorial optimization problems. This paper introduces a new type of ACO algorithm that will be used for routing along multiple routes in a network as opposed to optimizing a single route. Contrary to traditional routing algorithms, the Ant Dispersion Routing (ADR) algorithm has the objective of determining recommended routes for every driver in the network, in order to increase network efficiency. We present the framework for the new ADR algorithm, as well as the design of a new cost function that translates the motivations and objectives of the algorithm. The proposed approach is illustrated with a small simulation-based case study for the Singapore Expressway Network.**

I. Introduction

The objective of traffic dispersion routing algorithm is to dynamically control the traffic network equilibrium such that traffic flow in the network is optimized. The concept of traffic network equilibrium was introduced by Knight in 1924 [1] and was formalized in a traffic context by Wardrop in 1952 [2], resulting in what is today known as Wardrop's first and second principle of equilibrium. Wardrop points out in his first principle that each driver in the traffic network non-cooperatively seeks routes to benefit himself most, which is usually the case in reality. At equilibrium no driver has any incentive to change his current route, and thus it is defined as the User Equilibrium (UE) state. Wardrop's second principle assumes that there is a central decision maker that assigns routes to drivers. When the goal is achieved, all drivers collectively optimize the utilization of the network and the average travel time is minimum. This state is defined in the literature as the System Optimum (SO) state.

The problem of traffic modeling and traffic control with respect to the UE and the SO states has been studied extensively in the last years. Modeling and simulating these states in a traffic network has been studied in [3], [4], [5], while optimization and control for this purpose has been published in [6], [7], [8]. In general, reaching the SO state and reaching the UE state are conflicting objectives, where in the SO state the users will not all maximize their personal objectives, and where for the UE state the collective objective will not be maximized. In our work, we therefore aim at optimizing traffic routing by balancing both states. For this purpose, we introduce a novel routing algorithm, derived from the existing class of Ant Colony Optimization (ACO) algorithms. ACO has widespread applications in traffic, such as traffic simulation [9], routing and jam avoidance algorithms [10], [11], and traffic assignment [8]. However, most routing algorithms found throughout the literature pursue the UE and do not consider the impact the actions of the users will have on the traffic network. This limitation is the main motivation for the development of the algorithm presented in this paper, the Ant Dispersion Routing (ADR) algorithm. Algorithms such as the dynamic path planning algorithm presented in [6] already include in their cost function predicted states of the network, in order to avoid the users making "selfish" decisions and to ensure a more efficient routing policy. However, that particular algorithm is based on unrealistic assumptions, disregarding the fact that drivers want to optimize their own routes. It also does not use a traffic model, but solely a constant capacity constraint on each road, disregarding the location of the drivers on that road. The novelty in our algorithm is the use of an ant-based optimization method in combination with a traffic prediction model to analyze the impact of routing decisions on the future states of the network, and the use of a more complete cost function, which includes the states described above so that it can pro-actively act against traffic jams and even under low density traffic conditions optimize the distribution of flows to improve network efficiency and overall travel time.

The rest of this paper is structured as follows. In Section II the basic ACO algorithm is briefly reviewed. Section III contains the problem statement and defines the objectives. In Section IV the novel ACO-based algorithm for traffic routing is presented. Section V illustrates the approach using a small simulation-based case study for the Singapore Expressway Network. We conclude with a short discussion of open issues and topics for future work in Section VI.

II. Ant Colony Optimization

ACO algorithms have been developed to solve hard combinatorial optimization problems [12]. A combinatorial optimization problem can be represented as a tuple $P = \langle \mathscr{S}, F \rangle$, where $\mathscr{S}$ is the solution space with $s \in \mathscr{S}$ a specific candidate solution and where $F : \mathscr{S} \to \mathbb{R}_+$ is a fitness function assigning strictly positive values to candidate solutions, where higher values correspond to better solutions. The purpose of the algorithm is to find a solution $s^* \in \mathscr{S}$, or a set of solutions $\mathscr{S}^* \subseteq \mathscr{S}$ that maximize the fitness function. The solution s^* is then called an optimal solution and $\mathscr{S}^*$ is called the set of optimal solutions.

In ACO, the combinatorial optimization problem is represented by a graph consisting of a set of vertices and a set of arcs connecting the vertices. A particular solution s is a concatenation of solution components (i, j), which are pairs of vertices i and j connected by the arc ij. The concatenation of solution components forms a path from

The authors are with the Delft Center for Systems and Control, Delft, The Netherlands, email: diogo.a.alves@gmail.com, {j.m.vanast,z.cong,b.deschutter,r.babuska}@tudelft.nl

the initial vertex to the terminal vertex. Two values are associated with the arcs of the graph: a pheromone trail variable τ_{ij} and a heuristic variable η_{ij}. The pheromone trail represents the acquired knowledge about the optimal solutions over time and the heuristic variable provides a priori information about the quality of the solution component, i.e., the quality of moving from vertex i to vertex j. In general, a heuristic variable represents a short-term quality measure of the solution component, while the task is to acquire a concatenation of solution components that overall form an optimal solution. A pheromone variable, on the other hand, encodes the measure of the long-term quality of concatenating the respective solution components.

The most basic ACO algorithm is called the Ant System (AS) [13] and works as follows. A set of M ants is randomly distributed over the vertices. The heuristic variables η_{ij} are set to encode the prior knowledge by favoring the choice of some vertices over others. For each ant c, the partial solution $s_{\mathrm{p},c}$ is initially empty and all pheromone variables are set to some initial value $\tau_0 > 0$ (note that this ensures that later on the pheromone values will keep on being strictly positive). In each step, each ant decides based on some probability distribution, which solution component (i,j) to add to its partial solution $s_{\mathrm{p},c}$. The probability $p_c\{j|i\}$ for ant c on vertex i to move to a vertex j within its feasible neighborhood $\mathscr{N}_{i,c}$ is defined as:

$$p_c\{j|i\} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in\mathscr{N}_{i,c}}\tau_{il}^{\alpha}\eta_{il}^{\beta}}, \forall j \in \mathscr{N}_{i,c}, \tag{1}$$

with $\alpha \geq 1$ and $\beta \geq 1$ determining the relative importance of η_{ij} and τ_{ij} respectively. The feasible neighborhood $\mathscr{N}_{i,c}$ of an ant c on a vertex i is the set of not yet visited vertices that are connected to i. By moving from vertex i to vertex j, ant c adds the associated solution component (i,j) to its partial solution $s_{\mathrm{p},c}$ until it reaches its terminal vertex and completes its candidate solution.

The candidate solutions of all ants are evaluated using the fitness function $F(s)$ and the resulting value is used to update the pheromone levels by:

$$\tau_{ij} \leftarrow (1-\alpha_{\mathrm{ev}})\tau_{ij} + \sum_{s\in\mathscr{S}_{\mathrm{upd}}}\Delta\tau_{ij}(s), \tag{2}$$

with $\alpha_{\mathrm{ev}} \in (0,1)$ the evaporation rate and $\mathscr{S}_{\mathrm{upd}}$ the set of solutions that are eligible to be used for the pheromone update, which will be explained further on in this section. This update is called the *global* pheromone update. The pheromone deposit $\Delta\tau_{ij}(s)$ is computed as:

$$\Delta\tau_{ij}(s) = \begin{cases} F(s) & , \text{ if } (i,j)\in s \\ 0 & , \text{ otherwise.} \end{cases}$$

In an ACO algorithm, there is an inner and an outer loop. Within the inner loop, each ant repeatedly applies (1) to constructs a solution. In the outer loop, all solutions are evaluated over the fitness function and the pheromone levels are updated using (2). We define an iteration as one cycle of the outer loop.

The pheromone levels are a measure of how desirable it is to add the associated solution component to the partial solution. In order to incorporate forgetting, the pheromone levels decrease by some factor in each iteration. This is called pheromone evaporation in correspondence to the physical evaporation of the chemical pheromones for real ant colonies. By evaporation, it can be avoided that the algorithm prematurely converges to suboptimal solutions. Note that in (2) the pheromone level on all vertices is evaporated and only those vertices that are associated with the solutions in the update set receive a pheromone deposit.

In the following iteration, each ant repeats the previous steps, but now the pheromone levels have been updated and can be used to make better decisions about which vertex to move to. After some stopping criterion has been reached (e.g., when the difference between pheromone levels across iterations is less than some threshold, or after a pre-specified number of iterations), the values of τ and η on the graph encode the solution for all (i,j)-pairs. This final solution can be extracted from the graph by selecting the pairs (i,j) such that $j = \arg\max_l(\tau_{il}^{\alpha}\eta_{il}^{\beta})$. Note that all ants are still likely to follow suboptimal trajectories through the graph, thereby exploring constantly the solution space and keeping the ability to adapt the pheromone levels to changes in the problem structure.

There exist various rules to construct $\mathscr{S}_{\mathrm{upd}}$, of which the most standard one is to use all the candidate solutions found in the trial. This update rule is typical for the Ant System. However, other update rules have shown to outperform the AS update rule in various combinatorial optimization problems. Rather than using the complete set of candidate solutions from the last trial, either the best solution from the last trial, or the best solution since the initialization of the algorithm can be used. The former update rule is called *Iteration Best* in the literature, and the latter is called *Best-So-far* or *Global Best* in the literature [12]. These methods result in a strong bias of the pheromone trail reinforcement towards solutions that have been proven to perform well and additionally reduce the computational complexity of the algorithm. As the risk exists that the algorithm prematurely converges to suboptimal solutions, these methods are only superior to AS if extra measures are taken to prevent this [14], [15]. In this paper we will use an AS-like update rule because of its easier implementation. Note, however, that we could also use one of the other update rules instead.

III. Traffic Network Equilibrium

A. Problem Statement

The aim of this paper is to develop an ACO-based routing algorithm. In order to determine the "optimal" routes (in the UE or SO sense), a cost has to be assigned to a given route. There are several ways to express travel costs for a route, such as the travel time, the length of the route, the traffic density on the route, etc. or a weighted combination of these. In this paper we focus on the travel time as the

main component of the travel cost. However, the approach can easily be extended to other cost measures.

To improve the traffic network efficiency, flows have to be redistributed, as opposed to the UE state where all drivers go for the best available road. However, odds are that the SO state will consist of a distribution of flows that severely increases the travel times of some drivers to benefit the entire system, which is undesirable as well, since the routing suggestions are likely to not be accepted by the drivers. The objective of our algorithm is to, given a certain origin and destination pair, redistribute flows such that the traffic network efficiency is optimized, and in extreme cases of density in the network, traffic jams are prevented at all cost. This is done while considering the independent wishes of each driver that at all times want to use the best individual solution available to them. These are conflicting objectives, which requires establishing priorities. The objective is translated into a dynamic balance between the UE and SO states that will be quantified by a cost function. This balance must satisfy the following conditions:

Condition A: Avoid congestion by keeping the flows below the known bottleneck capacities.

Condition B: The difference in travel time between the fastest and the slowest routes must be below a certain threshold.

Condition C: The fastest route must have as many vehicles as possible under the constraints imposed by Condition A and Condition B.

There will be cases where all three conditions cannot be met, and in those cases Condition A takes precedence over Condition B, and Condition B takes precedence over Condition C.

It is important to note that for the sake of brevity we will consider a static prediction model based on the equilibrium relation between speed and density of the fundamental diagram. In fact, for the static prediction model, both steps of the ADR algorithm of Section IV can also be solved using standard graph algorithms. However, the ADR algorithm presented below is modular and also allows to include a dynamic traffic prediction model. Moreover, the ACO approach naturally lends itself to massive parallel execution, which is a big advantage for large-scale traffic networks.

B. Travel Time Cost

The travel time cost γ_r is the time it takes to travel a stretch of road (indicated by "road" for short in the sequel) r and — in the static case — is calculated by dividing the length of the road by the average speed of the vehicles on it, i.e.:

$$\gamma_r = \frac{L_r}{V_r(\rho_r)}, \tag{3}$$

where L_r is the total length of road r and $V_r(\rho_r)$ is the relation of the fundamental diagram that gives the equilibrium speed V_r corresponding to the given density ρ_r.

In order to severely penalize passing the critical level of density ρ_{crit}, beyond which the traffic system is unstable and easily results in a jam, an updated travel time cost function should be introduced based on (3):

$$\gamma_r = f_r(\rho_r) = \begin{cases} \frac{L_r}{V_r(\rho_r)} + M \exp\left(-\frac{(\rho_r - \rho_{\text{crit},r})^2}{\varepsilon}\right) & \text{if } \rho_r \leq \rho_{\text{crit},r}, \\ \frac{L_r}{V_r(\rho_r)} + M & \text{otherwise.} \end{cases} \tag{4}$$

where M is a constant, $\rho_{\text{crit},r}$ is the critical density for road r, and the parameter ε is the steepness of the bell function (a lower value of ε results in a larger steepness of the function). The above travel time cost function will satisfy Condition A, since the costs on roads that exceed the critical value of density ρ_{crit} will be unattractive for ants, thus preventing that the new distribution of flows to be computed by the algorithm results in traffic jams.

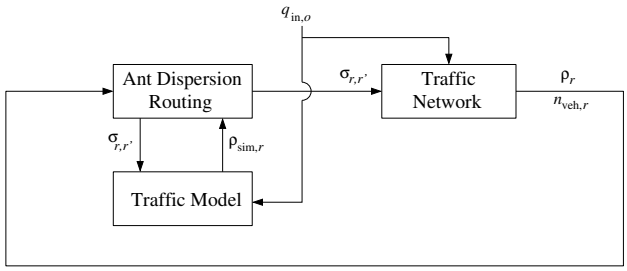

Fig. 1. Schematic representation of the closed-loop control traffic system with the ADR algorithm and the traffic prediction model.

C. Equilibrium States

Now we formally characterize the User Equilibrium (UE) and System Optimum (SO) traffic states that were already mentioned in Section I.

The UE state was defined by Wardrop [2] as the state where all drivers choose, at each moment, the route that minimizes their own travel time cost. Let φ_i be the travel time cost of route i between a fixed origin and destination, which is just the sum of the individual travel time costs of the roads that compose that route:

$$\varphi_i = \sum_{r \in R_i} \gamma_r \ , \tag{5}$$

where R_i is the set of roads in route i. Therefore the decision process of each driver is represented as the optimization of the following cost function:

$$J^*_{\text{UE}} = \min_i \varphi_i \ . \tag{6}$$

The SO state can be reached if the average travel time cost of the drivers using the network is minimized. Let $n_{\text{veh},i}$ be the number of vehicles on route i and let n_{r} be the number of routes. Then the SO state is defined as the optimization of the following cost function:

$$J^*_{\text{SO}} = \min_{n_{\text{veh},1},\ldots,n_{\text{veh},n_{\text{r}}}} \frac{\sum_{i=1}^{n_{\text{r}}} \varphi_i n_{\text{veh},i}}{\sum_{i=1}^{n_{\text{r}}} n_{\text{veh},i}} \ . \tag{7}$$

IV. Ant Dispersion Routing Algorithm

A. Main Algorithm

A schematic representation of the closed-loop operation of the ADR algorithm is given in Fig. 1. As indicated before we consider a static traffic prediction model in this paper to predict the behavior of the traffic network (in the form of simulated densities $\rho_{\text{sim},r}$ for each road r). However, we can also use a dynamic traffic prediction model instead.

The ADR algorithm is composed of two separate main steps, viz. network pruning and flow optimization. The network pruning step consists of a normal ant-based routing algorithm that finds multiple paths with less travel time costs than other paths, based on the present traffic conditions. Next, the flow optimization steps consists in determining the correct distribution of flows on these paths (or equivalently, the desired splitting rates $\sigma_{r,r'}$ from road r to r' in each node of the network) such that the overall network conditions (expressed in function of the travel time costs) are optimized. We assume the presence of another control layer in the traffic network that translates these splitting rates into route guidance commands as well as settings for other traffic control measures that could have an impact on route choice (such as traffic signals, speed limits, ramp metering, etc.).

B. Network Pruning

For the network pruning part of ADR, the algorithm largely follows the structure of the AS. Initially, the pheromone levels on all roads are set to a small initial value. The probability function from (1) is taken without the heuristic variables in order to prevent biasing the ants towards certain paths. The probability of choosing road r' from road r at an intersection is:

$$p_{r,r'} = \begin{cases} \dfrac{\tau_{r'}}{\sum_{j \in \mathscr{N}_r} \tau_j} & \text{if } r' \in \mathscr{N}_r \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

with $\mathscr{N}_r$ the set of roads connected to road r at the intersection. All ants iteratively make these decisions in order to find routes through the network. Then all routes are evaluated to determine the pheromone deposit. This is done by calculating the travel time cost φ_i of each route i based on the currently measured traffic state ρ_r for each road r. Next, the travel time cost γ_r of each road and the travel time cost φ_i of each route i are computed using the expression given in Section III-B. Finally the routing algorithm adds pheromone deposits to the routes identified as the fastest n_{r} routes, and also sets all pheromone levels to zero for roads that are not part of any of those fastest n_{r} routes. Note that if pheromone levels on routes are set to zero then those roads become invisible to ants, thus pruning the network, while also reducing the unnecessary exploration by ants of routes that are of no interest. The pheromone deposit on each route is the inverse of the cost of that route:

$$\Delta\tau_i = \frac{1}{\varphi_i}. \tag{9}$$

Let α_{ev} be the pheromone evaporation rate and let $\mathscr{I}$ be the set of routes found by the ants. The global pheromone update is then defined as:

$$\tau_r \leftarrow (1-\alpha_{\text{ev}})\tau_r + \alpha_{\text{ev}} \sum_{\substack{i \in \mathscr{I}; \\ r \in \mathscr{R}_i}} \Delta\tau_i \,, \forall r : \exists i \in \mathscr{I} : r \in \mathscr{R}_i. \tag{10}$$

This simple routing algorithm transforms the traffic network into a reduced network, composed only of the routes of interest that will be used in the flow optimization part of the ADR algorithm.

C. Flow Optimization

Now the ADR algorithm can proceed to optimize the distribution of traffic flows in this reduced network. A clear departure must be made from the AS when it comes to the optimization of flows since, when the AS converges, it always converges to only one optimal route. This happens due to the fact that the more ants use a route, the more attractive that route becomes, because of the pheromone deposits. However, we want to optimize the distribution of flows that leads to a better usage of the network as opposed to finding one single optimal route. The pheromone deposit function in ADR thus cannot be based on the number of ants using it. Instead, the pheromone deposits should be based on the aggregated solutions of all ants. Likewise, the function should incorporate the conditions stated in Section III, the cost of a route φ_i, and the network cost Ω.

Similarly to the network pruning part presented above, the ants have the objective of finding the best solution according to the probability function defined in (8). The number of ants must then be converted into the number of vehicles so that densities can be correctly calculated according to the traffic model used by ADR. Let $n_{\text{veh},i}$ be the number of vehicles using route i, and $n_{\text{ants},i}$ be the number of ants using route i. Similarly, n_{veh} is the total number of vehicles and n_{ants} the total number of ants. The number of vehicles using each route i is:

$$n_{\text{veh},i} = n_{\text{ants},i} \frac{n_{\text{veh}}}{n_{\text{ants}}}. \tag{11}$$

Now the ADR traffic model can calculate the density $\rho_{\text{sim},r}$ on each road r of the network, from which we can again derive the value of the travel time cost function φ_i for each route i. The network cost Ω can now be computed as the average cost over all drivers:

$$\Omega = \frac{\sum_{i=1}^{n_{\text{r}}} \varphi_i n_{\text{veh},i}}{\sum_{i=1}^{n_{\text{r}}} n_{\text{veh},i}}. \tag{12}$$

While Condition A was addressed in the cost function γ_r, Condition B and Condition C have yet to be represented in this framework. This will be done by creating a new pheromone deposit equation, which is clearly different from the ones traditionally used in ACO. Instead of the pheromone deposit equation representing a minimization of costs, it will represent a minimization of differences between costs. We want to minimize the cost of each route φ_i and the network cost Ω, so the pheromone deposit should be a weighted sum

of the inverse of these components. The new pheromone deposit is defined as follows:

$$\Delta\tau_i = \frac{1}{\varphi_i} + \frac{W}{\Omega}, \tag{13}$$

with W a weighting factor. Minimizing the difference between the constant network cost Ω and the cost of each route can be formalized as follows:

$$\min_{c_1,\ldots,c_{n_i}} |\varphi_i - \Omega|, \tag{14}$$

which reaches its minimum when $\varphi_1 = \varphi_2 = \cdots = \varphi_{n_i} = \Omega$, i.e., when the costs of all routes are the same. In order to achieve this minimum, in (13) we must choose $W = -1$. If we do not want the cost of all routes to be exactly the same, but have a certain bias towards the shortest route, we must choose $W \in (-1, 0)$. The tuning of W depends on the constraints we want to set between the difference of the least and most costly routes. For a small difference of around 5–10 %, W should be only slightly larger (less negative) than -1. For larger differences of around 20%, the weight should tend more towards 0.

The last step of the ADR algorithm is again the pheromone update equation (10).

V. Case Study

We now illustrate the ADR algorithm on a case study involving the Singapore Expressway Network. The Singapore Expressway Network was chosen due to its high density of highways, with several intersections, meaning that there are several possible routes from each origin to each destination without requiring the driver to use urban roads. A simplification of the network will be done since we are interested in the routing results of the algorithm rather than the precision of the network. As such, speed limits in tunnels are removed, and the transition between highways is assumed to be limited to just the removal or the addition of lanes, instead of the more complex interchanges, on-ramps, and off-ramps. Only the central and eastern parts of the full Singapore Expressway Network will be used (see Fig. 2) since those are enough to test the ADR algorithm and contain the points of the island that are usually subject to traffic jams.

The network is composed of 18 stretches of highway (36 if we consider both directions), 8 origins and 8 destinations, as seen in Fig. 2. The critical area of the network is in the business district (connected to origins o_5, o_6, o_7, o_8 and destinations d_5, d_6, d_7, d_8). The existence of several origins and destinations in that area and the fact that the Central Expressway has only two lanes severely limit capacity.

In our case study, the ADR algorithm will be used to lead drivers optimally from a certain area of Singapore towards the central business district. We assume that all traffic states are available through floating car data. However, control of only one zone (the airport, corresponding to origin o_4) is assumed to be available, and all drivers coming from other areas do not use the ADR algorithm. Note that in total the business district area can be accessed through 4 different

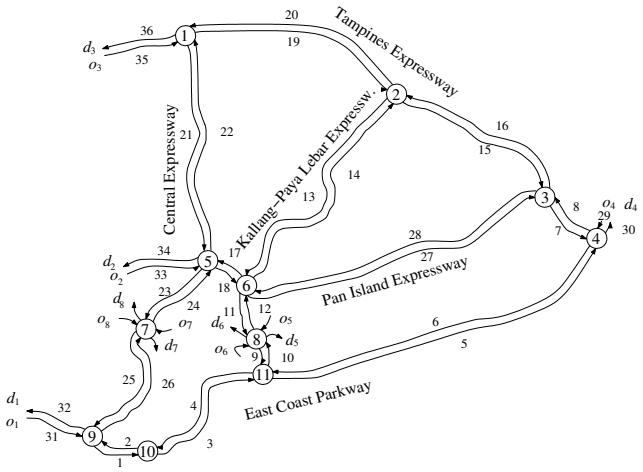

Fig. 2. Part of the Singapore Expressway Network used for the case study.

TABLE I
Traffic conditions for the case study.

Origin	Destination	Flow [veh/h]	Route
o_1	d_5	2000	$\{31,1,3,10\}$
o_1	d_7	1000	$\{31,26\}$
o_2	d_6	1000	$\{33,18,11\}$
o_2	d_8	1000	$\{33,23\}$
o_3	d_1	1000	$\{35,21,23,25,32\}$
o_3	d_2	2000	$\{35,21,34\}$
o_3	d_4	1000	$\{35,19,15,7,30\}$
o_4	d_5, d_6, d_7, d_8	4000	ADR

destinations (d_5, d_6, d_7, d_8), increasing the freedom of the ADR algorithm.

As simulation model we use the METANET model [16], [17] and as prediction model we use the static model discussed in Section III.

In this case study, only one of the origins is controlled by the ADR algorithm. So the algorithm must be capable of routing vehicles through several different routes, while also accounting for the traffic it is not controlling. We will use the ADR algorithm to control the traffic going from the airport (origin o_4) towards the business district (destinations d_5, d_6, d_7, and d_8). In Table I the demand at each origin is defined, with 4 origins feeding traffic into the network. In the given scenario, ADR will check whether there are roads on which the density is already larger than the critical level, and then find the ideal routes from the airport to the business district.

First the ADR determines the optimal routes from o_4 to the business district, which turn out to be $\mathscr{R}_1 = \{29,6,10\}$ and $\mathscr{R}_2 = \{29,8,28,11\}$, with respectively 15 and 16 kilometers in length. This result can be easily verified as follows. Intuitively, looking at Fig. 2, several possible routes exist, but two of them are considerably shorter than all others: $\mathscr{R}_1$ and $\mathscr{R}_2$.

Note that at the start of the trial there are already 2000 veh/h using road 10 and 1000 veh/h using road 11. This means that the fact that there are fewer vehicles on route 2 compensates for the shorter length of route 1. This makes the difference in travel time cost between the two routes close

TABLE II

ADR PERFORMANCE ON ROUTES $\mathscr{R}_1$ AND $\mathscr{R}_2$.

	Route $\mathscr{R}_1$	Route $\mathscr{R}_2$
φ [s]	622.9	665.5
Route usage [%]	47.08	52.92

TABLE III

ADR PERFORMANCE ON ALL ROADS. THE SUBSCRIPT "CONTROLLED" REFERS TO THE VEHICLES CONTROLLED BY THE ADR ALGORITHM; "UNCONTROLLED" REFERS TO THE OTHER VEHICLES, AND "TOT" REFERS TO ALL VEHICLES TOGETHER.

	Road 29	Road 6	Road 10	Road 8	Road 28	Road 11
γ [s]	45.1	528.4	49.4	81.7	449.5	89.2
$\rho_{\text{controlled}}$	15.65	6.65	8.59	7.51	7.51	5.68
$\rho_{\text{uncontrolled}}$	0	0	11.35	0	0	9.37
ρ_{tot} [veh/km/lane]	15.65	6.65	19.94	7.51	7.51	15.05
$q_{\text{controlled}}$	5000	2354	2354	2646	2646	2646
$q_{\text{uncontrolled}}$	0	0	2000	0	0	1000
q_{tot} [veh/h]	5000	2354	4354	2646	2646	3646

to zero. This is an important situation since now Condition C will be impossible to be met for two reasons: now that the travel time cost of each route is the same, if the algorithm routes more vehicles to one road that road will become more costly than the other, and vice versa. Also, as we can see in Table II, the algorithm will route more vehicles towards route 2. That is due to the fact that originally without ADR, traffic on road 10 (route 1) has a higher density than on road 11 (route 2), and thus ADR cannot route more vehicles to route 1 due to the penalty introduced in (4).

Tables II and III show that Condition A and Condition B (the most important ones) are accomplished by the algorithm. No traffic jams occur in the network with the maximum density on roads used by the algorithm is the one on road 10 with $\rho = 19.94\,\text{veh}/\text{km}/\text{lane}$. Also the difference between travel time costs of both routes is kept inside the usual interval of 5–10% with $\varphi_2/\varphi_1 = 1.068$, which means that the difference is 6.8%.

VI. CONCLUSIONS AND FUTURE WORK

We have introduced a novel on-line ant-based traffic routing algorithm that optimizes the distribution of traffic flows. This model-based ADR algorithm was designed to solve the traffic network equilibrium problem, which is a complex optimization problem since it has two conflicting objectives: reducing travel times of individual drivers, while improving network efficiency. The algorithm proved to work accurately for a case study network involving the Singapore Expressway Network.

Several topics for future work can be identified. Note that in this paper we used a static traffic prediction model for the sake of brevity. If a dynamic traffic prediction model is used, it is in general necessary to iterate the two main steps of the ADR algorithm, viz. network pruning and flow optimization, several times in order to obtain an equilibrium situation. The convergence speed and convergence properties of this process should be investigated. In addition, the scalability, the performance, and the computational requirements of the ADR algorithm using a dynamic traffic prediction model and more complex travel cost function should be assessed for a wide range of case studies (involving both freeway and urban networks) and compared to that of other methods from the literature for static or dynamic traffic routing. Moreover, the robustness of the algorithm against model mismatches and disturbances should be investigated.

ACKNOWLEDGEMENTS

Research supported by the China Scholarship Council, the Transport Research Center Delft, the European COST Action TU0702, and the European 7th Framework Network of Excellence "Highly-complex and networked control systems (HYCON2)".

REFERENCES

[1] F. H. Knight, "Some fallacies in the interpretation of social cost," *The Quarterly Journal of Economics*, vol. 38, pp. 582–606, 1924.

[2] J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings, Institute of Civil Engineers, Part II*, vol. 1, pp. 325–378, 1952.

[3] J. Dong and J. Wu, "Urban traffic networks equilibrium status recognition with neural network," *Proceedings Intelligent Transportation Systems, 2003 IEEE*, vol. 2, pp. 1049–1053 vol.2, Oct. 2003.

[4] L. D'Acierno, B. Montella, and F. D. Lucia, "A stochastic traffic assignment algorithm based on ant colony optimization," in *Lecture Notes in Computer Science*, 2006, pp. 25–36.

[5] F. Zhang and N. E. Leonard, "Coordinated patterns of unit speed particles on a closed curve," *Systems and Control Letters*, p. 92, 2007.

[6] W. Hong, Y. Tian, and Y. Xu, "The research of dynamic path planning for centralized vehicle navigation," *Automation and Logistics, 2007 IEEE International Conference on*, pp. 1198–1202, Aug. 2007.

[7] M. Rodriguez-Perez, S. Herreria-Alonso, M. Fernandez-Veiga, A. Suarez-Gonzalez, and C. Lopez-Garcia, "Achieving fair network equilibria with delay-based congestion control algorithms," *Communications Letters, IEEE*, vol. 12, no. 7, pp. 535–537, July 2008.

[8] Z. Xu, H. Sun, X. Li, D. Chen, and S. Yu, "Ant colony optimization arithmetic of capacity restraint traffic assignment," *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pp. 972–976, Sept. 2008.

[9] R. Hoar, J. Penner, and C. Jacob, "Evolutionary swarm traffic: if ant roads had traffic lights," *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, pp. 1910–1915, 2002.

[10] B. Tatomir and L. Rothkrantz, "Hierarchical routing in traffic using swarm-intelligence," *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pp. 230–235, 2006.

[11] P. Bedi, N. Mediratta, S. Dhand, R. Sharma, and A. Singhal, "Avoiding traffic jam using ant colony optimization - a novel approach," *Computational Intelligence and Multimedia Applications, International Conference on*, vol. 1, pp. 61–67, 2007.

[12] M. Dorigo and C. Blum, "Ant colony optimization theory: a survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, November 2005.

[13] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 1, pp. 29–41, 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, 1997.

[15] T. Stützle and U. Hoos, "MAX MIN Ant System," *Journal of Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.

[16] A. Messmer and M. Papageorgiou, "METANET: A macroscopic simulation program for motorway networks," *Traffic Engineering and Control*, vol. 31, no. 8/9, pp. 466–470, Aug./Sept. 1990.

[17] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham, "Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 282–292, Dec. 2002.