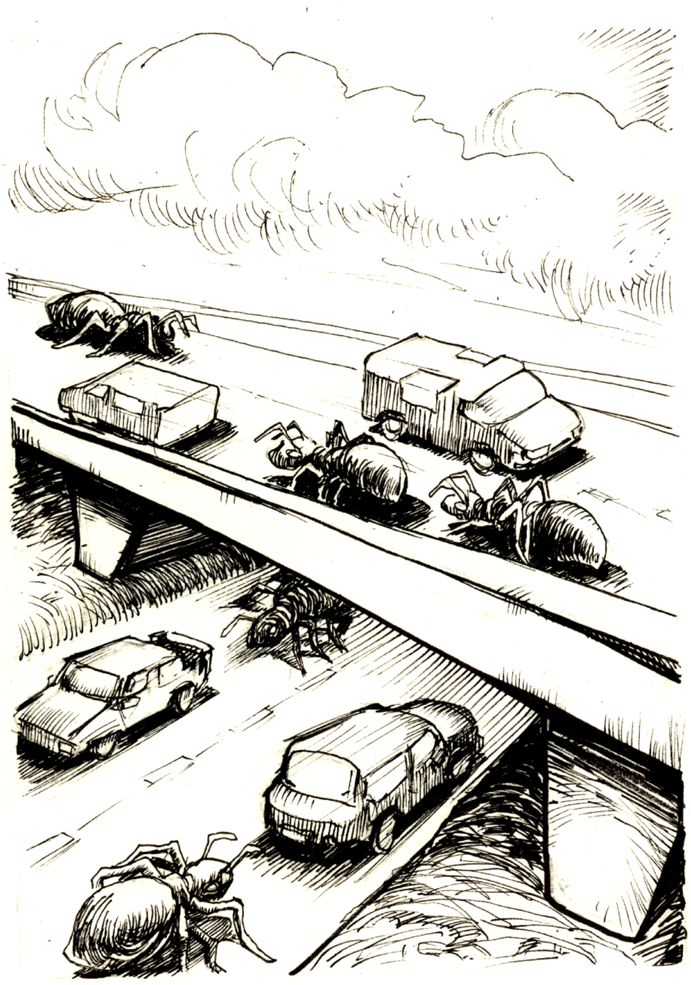
M.Sc. Thesis

Ant Dispersion Routing for Traffic Optimization

Diogo Alves

June 25th 2009



Delft Center for Systems and Control

Delft University of Technology

Ant Dispersion Routing for Traffic Optimization

Master of Science Thesis

For obtaining the degree of Master of Science in Mechanical Engineering
at Delft University of Technology

Diogo Garcia Almeida Alves

June 25, 2009

Faculty of Mechanical Engineering · Delft University of Technology

Preface

The motivations and objectives of this thesis stem from the findings of my literature survey (Alves, 2008), where I had the objective of finding new possible applications of Swarm Intelligence (SI) theory in traffic control related problems. A problem that is partially addressed in (Bedi et al., 2007), and in which I was particularly interested is the Network Equilibrium problem (Chudak and Eleuterio, 2006; Dong and Wu, 2003). As a student of systems and control engineering, the stability of a system is a major point of interest, and in the network equilibrium problem that is our main concern. Not only that, we also are looking into improving the characteristics of the system. Since we wanted to use SI to solve this problem, the focus of this thesis is the Ant Colony Optimization (ACO) class of algorithms, which is the basis of the framework to the new algorithm to be designed: the Ant Dispersion Routing (ADR). When used for routing, ACO algorithms can find the shortest or more generally the least costly routes with low computational effort. However, like all other routing algorithms, they are optimizing the solution of a driver (agent) in a traffic network (global system), with many other drivers. Assuming that all drivers have access to a routing solution, this can be used to increase the efficiency of the network, instead of finding the shortest for each driver in a "selfish" way. The ADR algorithm output suggests a cooperative behaviour among drivers, which if followed, decreases average density on all roads, and improves traffic flows, benefiting all drivers in the network. The appendices at the end of this paper contain extra information about the network used for the case study, and also the results of trials used to tune the parameters present in the ADR algorithm.

Diogo Alves
June 2009

Acknowledgment

I would like to use this opportunity to thank my supervisors Robert Babuška and Jelmer van Ast for their continuous support. Thanks to the academic freedom and through the frequent discussions we had, I got the opportunity of pinpointing the problem I wanted to solve, and address it the way I wanted. I would also like to thank Bart De Schutter and Andreas Hegyi for their expert input during important parts of my thesis.

Cover picture by Gilles van Kralingen

List of symbols

ADR notation

Symbol	Meaning
J^*_{UE}	cost of user equilibrium state
J^*_{SO}	cost of system optimum state
N_{t}	number of iterations of algorithm
p_{r_1,r_2}	probability of going from road r_1 to road r_2
τ_r	pheromone level on road r
ρ_{ev}	pheromone evaporation rate
n_{a}	number of ants a
n_{c}	total number of cars in network
$\mathcal{N}_r$	set of roads connected to road r
$n_{a,r}$	number of ants using road r
$n_{a,i}$	number of ants using route i
$n_{c,i}$	number of cars using route i
$\rho_{\mathrm{adr},r}$	predicted density on road r by algorithm
$\rho_{\mathrm{meta},r}$	worst real density on road r seen in the METANET simulator
γ_r	travel time cost of road r
φ_i	travel time cost of route i
Ω	network cost
$\mathcal{R}_i$	set of roads on route i
$\mathcal{I}$	set of roads on all routes $\mathcal{R}_i$
W_1, W_2	weightings of pheromone update
M, ϵ	parameters of travel time cost function

ACO notation

Symbol	Meaning
V	set of nodes
E	set of edges
n_{n}	number of nodes n
n_{a}	number of ants a
L^a	distance travelled by ant a
τ_{ij}	pheromone level one edge connecting nodes i and j
α	pheromone importance constant
p_{ij}	probability of an ant choosing to go to node j when in node i
$\mathcal{N}_i$	set of node choices at node i
ρ_{ev}	pheromone evaporation rate
$\Delta\tau^a_{ij}$	pheromone deposited by ant a on edge connecting nodes i and j
S^a	sequence of edges visited by ant a

Network and METANET simulator notation

Symbol	Meaning
L_r	length of road r
l	fixed length of each segment s
T	time step of the simulator
K	number of iterations of the algorithm
λ_r	number of lanes of road r
N_r	number of segments of road r
s	segment $s \in \{1, 2, ..., N_r\}$
n_r	number of roads r
r	road $r \in \{1, 2, ..., n_\mathrm{r}\}$
n_o	number of origins o
o	origin $o \in \{1, 2, ..., n_\mathrm{o}\}$
n_d	number of destinations d
d	destination $d \in \{1, 2, ..., n_\mathrm{d}\}$
n_i	number of routes i
i	route $i \in \{1, 2, ..., n_\mathrm{i}\}$
$n_{\mathrm{c},r}$	number of cars on road r
$\rho_{r,s,r'}$	density on road r, segment s, heading for road r'
$V_{r,s}$	theoretical speed on road r, segment s
$v_{r,s}$	real speed on road r, segment s
$q_{r,s,r'}$	flow on road r, segment s, heading for road r'
$a_m, v_\mathrm{free}, \rho_\mathrm{crit}$	parameters of exponential model
κ, η, ν	parameters of real speed equation
w_o	number of vehicles waiting at node o
D_o	demand from node o
$q_{\mathrm{in},o}$	flow from node o
Q_o	maximum flow allowed from node o
$q_{\mathrm{max},r}$	maximum flow allowed in road r

Ant Dispersion Routing for Traffic Optimization

Diogo Alves, Jelmer van Ast, and Robert Babuška

Abstract

Ant Colony Optimization (ACO) algorithms are inspired by the social interaction between ants that work together to find the best possible path between their nest and food. In the past decade they have been successfully used as routing algorithms, with their main advantage being their computational efficiency. This paper introduces a new type of ACO algorithm, that will be used as a network optimizer as opposed to a single route optimizer. Contrary to traditional routing algorithms, the Ant Dispersion Routing (ADR) algorithm has the objective of creating routes for every driver in the network, to increase network efficiency, by limiting density on roads through the dispersion of flows. The framework for the new ADR algorithm is presented, as is the design of a new cost function that translates the motivations and objectives of the algorithm. Parameters of the algorithm are tuned in several different traffic networks to ensure the robustness of the results. The ADR algorithm is also used in a case study of the Singapore Expressway network, to control traffic going from the airport towards the central business district, area prone to heavy density of vehicles and traffic jams.

Keywords: Ant Colony Optimization, network equilibrium, traffic routing

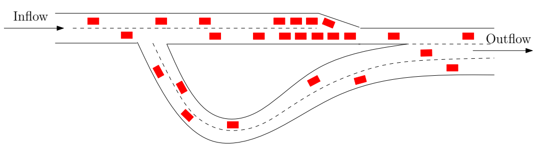
1 Introduction

The concept of Traffic Network Equilibrium was introduced by economist Frank Knight in his dissertation about social costs (Knight, 1924) as a response to Arthur Pigou's work, in which he created a distinction between private and overall costs (Pigou, 1920). Knight stated that, from an economics' points of view, individual freedom of choice results in bad distributions of investments.

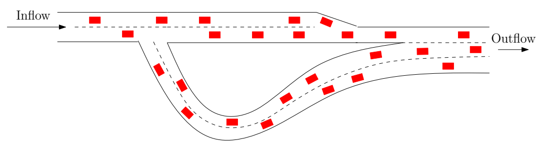
Wardrop (1952) formalized Knight's statements in a traffic context, which are today known as Wardrop's first and second principle of equilibrium. Wardrop's first principle is the equilibrium we see every day on the traffic network where each driver chooses the route that at that moment benefits him most, based on the information he has. This is defined in the literature as the User Equilibrium (UE) state. Wardrop's second principle assumes that users of the road network demonstrate a cooperative behaviour possibly via a centralized controller, and their routes are chosen for them, with the objective of maximizing the traffic network efficiency. This state of the road network is defined in the literature as the System Optimum (SO) state. A summary of the traffic network equilibrium problem can be found in (Chudak and Eleuterio, 2006).

During the last decades, this subject has gained a lot of attention, and has been studied more extensively in the areas of traffic modelling and traffic control. Several traffic models of these two states were created, in order to try and simulate these behaviours for traffic optimization (Dong and Wu, 2003; D'Acierno et al., 2006; Zhang et al., 2007). Also traffic control laws and optimization algorithms have been created in order to reach these two states (Hong et al., 2007; Rodriguez-Perez et al., 2008; Xu et al., 2008).

In this paper we acknowledge these two states and define that in a traffic network an optimum can be defined by combining and balancing these two states, which benefits both the network and its users. For this purpose, and in order to find this optimum a new routing algorithm will be introduced, derived from the existing class of Ant Colony Optimization (ACO) algorithms.



(a) In this first scenario all drivers head for the upper road, which while shorter, has a bottleneck. A traffic jam will originate, resulting in the rest of the drivers to use the other road (being informed by radio or variable message signs).



(b) This is our objective scenario: traffic is distributed so that both roads have short travel times, and more importantly, no traffic jam occurs, resulting in near optimal flow conditions for both roads.

Figure 1: Difference between a normal case scenario, which is seen everyday in traffic networks, and the main goal scenario of the paper. In both cases the inflow is equal.

Being a combinatorial optimization algorithm, ACO has more widespread applications for traffic, such as traffic simulation (Hoar et al., 2002), routing and jam avoidance algorithms (Tatomir and Rothkrantz, 2006; Bedi et al., 2007), or traffic assignment, where several types of costs are considered, such as fuel, travel time, and allocation of resources (Xu et al., 2008).

However, routing algorithms found throughout the literature pursue the UE, as they are “selfish” and do not consider the impact their actions will have in the traffic network (see Figure 1). This limitation is the main motivation for the development of the algorithm presented in this paper, the Ant Dispersion Routing (ADR). Algorithms such as the one presented in (Hong et al., 2007) already include in their cost function predicted states of the network, to avoid this “selfish” behaviour and ensure a more efficient routing policy. However this algorithm disregards the fact that drivers want to optimize their own route, and a safeguard has to be implemented to avoid penalizing some drivers for the benefit of others, even if this improves network efficiency. Also it 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 a macroscopic traffic model to analyse the impact of its decisions in 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.

Section 2 presents the problem statement, defines the objectives, and states the benefits the algorithm can bring to a real traffic network. Section 3 introduces traffic models and simulators, namely the fundamental diagram of traffic which will be the traffic model used in the ADR algorithm. In Section 4 the ACO class of algorithms is presented, followed by the presentation of the modifications done to the basic Ant System (AS) algorithm in order to design the framework of the ADR algorithm. Section 5 shows the potential of the ADR algorithm by applying it to a real traffic network in a case study of the Singapore Expressway network. A short discussion on possible ways to implement the ADR algorithm into a real traffic network is done in Section 6. Conclusions, as well as suggestions for future work are presented in Section 7.

2 Problem Statement

In an age where traffic congestions are frequent, and the capacities of road networks are saturated, the optimization of these networks is imperative for progress. In this section, the traffic states explained in Section 1 are formally defined. Also the motivations behind this paper and the different objectives that will be solved are stated in this section.

The UE state was defined by Wardrop as the state where all drivers choose, at each moment, the route that minimizes his own travel time cost. Let φ_i be the travel time cost of route i between a fixed origin and destination, and each driver's decision process is represented as:

$$J^*_{\text{UE}} = \min_i \varphi_i \,. \tag{1}$$

Using the definition of Wardrop, the SO state can be reached if the average travel time cost of the drivers using the network is minimized. Let $n_{\text{c},i}$ be the number of cars using route i, and the SO is defined as:

$$J^*_{\text{SO}} = \min_{n_{\text{c},1},\cdots,n_{\text{c},n_{\text{i}}}} \frac{\sum_{i=1}^{n_i} \varphi_i n_{\text{c},i}}{\sum_{i=1}^{n_i} n_{\text{c},i}} \,. \tag{2}$$

Now that we have defined the travel time cost, UE and SO states, we can also define the problem to be solved during the course of this paper, and the motivations behind it.

Throughout the last decades, urban and population growth have exceeded the road capacity growth. This unbalance has originated a lack of infrastructures to accomodate the heavy traffic big cities are subject to, originating bottlenecks in the traffic network. These bottlenecks are the points in the network where traffic jams occur during rush hour. If from an origin to a destination only two possible routes exists, and if all drivers select the best route available to them they will provoke congestion, in which case, only then will they pursue the other available route, until both routes have a similar travel time cost. If a good model of these bottlenecks is available, this opens up the possibility of pro actively acting against traffic jams and through routing define a distribution of flows that generates such equilibrium before a congestion occurs.

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 improved, 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, so we have to establish 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:

A - Avoid congestion by keeping the flows below the known bottleneck capacities;

B - The difference in travel times between the fastest and the slowest routes must be below a certain threshold;

C - The fastest route must have as many cars as possible under the constraints imposed by Conditions A and 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 fairly safe to assume that most of the drivers whose suggested routes are slower, then those of their UE, will not accept the routing choices defined for them, jeopardizing the potential benefits of the outcome of ADR. This is due to the fact that if a driver knows a certain zone, he will disregard the route suggested to him if he has knowledge of a faster one. Though the conditions of the algorithm already take this into consideration (by minimizing the time differences as stated in Condition B), this can be further aided by using existing technology and laws. If in the future GPS devices become standard in cars, and even mandatory by law, it is possible to first improve the results of the routing algorithms, and secondly observe users that are not following the suggested route, and adapt the control measures to the new situation. This scenario, while being futuristic, is also a realistic prediction, especially in a country as modern and small as The Netherlands, or the country used in the case study of this paper: Singapore. Also, this will enable to keep historical data on which drivers use which suggestion, so that in the medium run everyone benefits from the system. An example and more detailed discussion on a possible implementation of this system is done in Section 6. The benefits can even go outside the scope of this paper, since the better distribution of flows improves driving conditions, reducing the stress drivers experience, as well as accidents.

3 METANET Simulator

3.1 Introduction

When studying traffic related problems, it is of the utmost importance to use an adequate simulator. Traffic simulators can either be based on macroscopic models which deal with averages of traffic states; or they can be based on microscopic models which model every single car in the traffic network. In this paper a macroscopic model is used for two important reasons. There is no need to distinguish between cars, since our algorithm will disperse flows and not individual vehicles, and thus, the microscopic model would give us an excess of information that is not required. Secondly, the computational times of both types of simulators are very different. A microscopic simulator is on average about 100 times slower than a macroscopic model, and this is not helpful for investigation purposes. In this section the fundamental diagram of traffic is introduced, and the METANET model, used as the simulator, explained.

3.2 Fundamental Diagram of Traffic

The fundamental diagram of traffic flow is the basis of macroscopic traffic study. It gives us the relations between the 3 macroscopic traffic variables: density (ρ), speed (V), and flow ($q = \rho \cdot V$). These relationships are extremely useful to calculate maximum capacities of roads, mean speeds, and other variables in each segment of our macroscopic traffic model. Several diagrams have been proposed throughout the years, varying in complexity, and accuracy. Others consider multi regimes, and even discontinuities.

The fundamental diagram of traffic is usually divided in 3 functions that define the relationships between the variables:

- speed-density: $V = f_{V\rho}(\rho)$;
- flow-density: $q = f_{q\rho}(\rho)$;
- speed-flow: $V = f_{Vq}(q)$.

The simplest fundamental diagrams have usually two regions, a stable region where there exists free flow, and a region where traffic is unstable, which means that under these conditions, if there is a disturbance, such as a car braking suddenly, the system will collapse and a jam will occur. This can be seen in Figure 2.

These models are usually derived either through curve fitting, with data extracted from real traffic conditions, or by using microscopic models to obtain data. Also the parameters of these

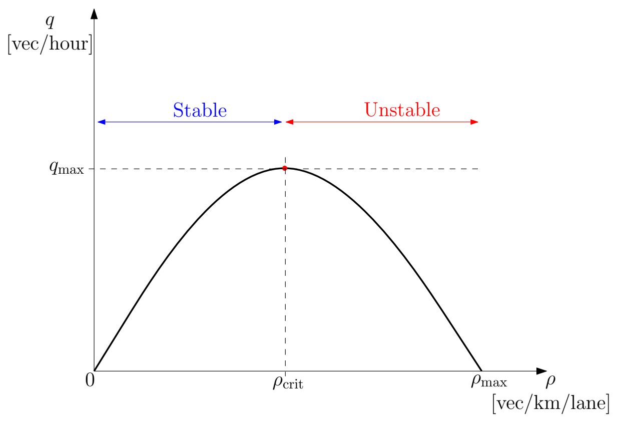


Figure 2: Simple Fundamental Diagram of Traffic of $q = f_{q\rho}(\rho)$ showing stable and unstable regions

models are varying under different road conditions, weather conditions, type of driving style of drivers in that road, and under different control methods, such as speed limits and variable message signs. In this paper an exponential model will be used, which is commonly associated with the METANET simulator. Besides being used in the METANET simulator, this will also be our model for traffic conditions in the ADR algorithm. Let a_m, v_{free} and ρ_{crit} be constant parameters that reflect the particular characteristics of a chosen network and the theoretical speed V is given as:

$$V = v_{\text{free}} \cdot e^{\left(-\frac{1}{\text{am}}\left(\frac{\rho}{\rho_{\text{crit}}}\right)^{\text{am}}\right)} . \tag{3}$$

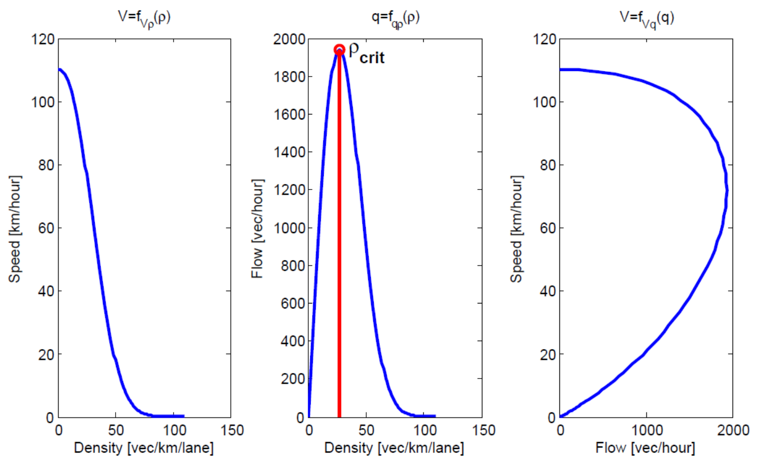


Figure 3: Exponential Model.

3.3 Simulator

For the highway traffic model, the METANET model (Messmer and Papageorgiou, 1990) was chosen. This was done due to its proven computational efficiency and accurate simulation results. The basis of the model is to divide each highway into several segments of length l (usually 500 m in size), as can be seen in Figure 4. At every time step $k = 1, ..., K$ (with $t = kT$ being the real time in seconds and usually $T = 10$) the state of each segment of the highway is calculated.

Let $r = 1, ..., n_\mathrm{r}$ be the index of the highway, $s = 1, ..., N_r$ be the index of the segment, and λ_r be the number of lanes on highway r. Then the flow at segment s of highway r at time step k is calculated through the fundamental relationship:

$$q_{r,s}(k) = \rho_{r,s}(k) v_{r,s}(k) \lambda_r \,. \tag{4}$$

The flow is the product of the density of cars per segment and lane by the speed of that segment.

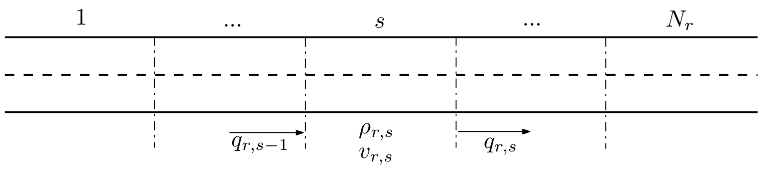


Figure 4: Partition of the highway into segments.

Since both density and speed are given states, the only missing initial condition was the flow. The boundary conditions such as the number of vehicles entering the network are also known. The METANET model recalculates at each time step these conditions, and the first one to be calculated is the density, which is done by using the conservation of vehicles law (derived from the conservation of mass law in fluid/thermo dynamics). Let T be the size of the time step in seconds, and the conservation of vehicles law gives us the new densities:

$$\rho_{r,s}(k+1) = \rho_{r,s}(k) + \frac{T}{l\lambda_r}(q_{r,s-1}(k) - q_{r,s}(k)) \,. \tag{5}$$

The density at the new time step is the density at the previous time step, plus the difference between cars entering the segment and leaving the segment during that time step.

The speed is a function of the density, since the the speed of vehicles is limited by the vehicles around them, and by speed limits. Let $v_{r,s}$ be the real speed in road r, segment s, $V = f_{V\rho}(\rho)$ be the velocity as a function of the density taken from the fundamental diagram of traffic (3), and let τ, η, κ be model parameters, and the velocity update function is given as:

$$v_{r,s}(k+1) = v_{r,s} + \frac{T}{\nu}(V_{r,s}(k) - v_{r,s}(k)) + \frac{T}{l} v_{r,s}(k)(v_{r,s-1}(k) - v_{r,s}(k)) - \frac{\eta T}{\nu l} \frac{\rho_{r,s+1}(k) - \rho_{r,s}(k)}{\rho_{r,s}(k) + \kappa} \,. \tag{6}$$

The above equation is derived from the Payne model (Payne, 1971), and is a more complex calculation since it takes into consideration aspects of driving and shock wave behaviours, and thus is divided in three parts. The first term is a smoothing function taken from the comparison of the theoretical speed in the fundamental diagram, and the real speed; the second term is a concave term, and the third term is an anticipation term, to model the behaviour that drivers adjust their speed according to what they see in front of them, that is, the density state of the next segment.

Traffic is fed into the highway, or network of highways, using origin nodes, that use a queue model. They are defined by their flow capacity, and number of vehicles in their buffer (queue), and they react to the demand on the first segment adjacent to it. This is defined by the user depending on what simulation study is required. Assume node o is only initializing traffic in road r. Let w_o be the number of vehicles waiting at node o, D_o the demand on road r from node o, and $q_{\mathrm{in},o}$ the flow from node o to road r. The number of cars waiting at node o at each time step is:

$$w_o(k+1) = w_o(k) + T\Big(D_o(k) - q_{\text{in},o}(k)\Big)\,. \tag{7}$$

The number of vehicles waiting at a queue is calculated similarly to the density, where the number of vehicles waiting at the entrance of a highway is the difference between the flow entering segment 1 and exiting the origin node, plus the number of vehicles already at the queue. The demand D_o is defined by the user according to the simulations to be run. The demand is usually extracted from historical data of the real traffic network to be simulated. Let Q_o be the maximum flow possible exiting node o and $q_{\max,r}$ be the maximum flow entering highway r, where the outflow is given by:

$$q_o(k) = \min\left[D_o(k) + \frac{w_o(k)}{T}, q_{\max,r}(k)\right], \tag{8}$$

with:

$$q_{\max,r}(k) = \left\{ \begin{array}{rl} Q_o & \text{if } \rho_{r,1} < \rho_{\text{crit},r} \\ Q_o \frac{\rho_{\max} - \rho_{r,1}}{\rho_{\max} - \rho_{\text{crit},r}} & \text{else} \end{array} \right. . \tag{9}$$

The outflow of node o is then the demand on road r, plus the number of vehicles that are in the buffer and can enter the highway at that time step.

For the problem at hand, a network of highways has to be built, and for that reason the density update (5) function becomes destination oriented. Let r' be the destination of the next highway the fraction of traffic is heading for, with the destination being decided at each node through the flow splitting matrix p. The density in the destination oriented model is calculated as follows:

$$\rho_{r,s,r'}(k+1) = \rho_{r,s,r'}(k) + \frac{T}{l\lambda_r}(q_{r,s-1,r'}(k) - q_{r,s,r'}(k))\,. \tag{10}$$

The METANET algorithm is presented in Algorithm 1.

Algorithm 1 METANET Simulator

```
1: Initialize $K$, $T$, $\kappa$, $\eta$, $\nu$
2: Load Network Information
3: for $k = 1, ..., K$ do
4:     Calculate inflows from origin nodes with (7, 8, and 9)
5:     Update flows in all segments with (4)
6:     Use conversation of vehicles law to calculate density (10)
7:     Calculate theoretical speed in each segment (3)
8:     Calculate real speed in each segment (6)
9: end for
```

4 Ant Dispersion Routing

Ant Dispersion Routing (ADR) is a new algorithm that belongs to the broader class of Ant Colony Optimization (ACO) algorithms, and was created with the objective of solving the traffic network equilibrium problem stated in Section 2.

ACO algorithms are used in combinatorial problems to find an optimal solution by mimicking the social process through which ants find the best paths between their nest and food sources. In ACO algorithms each ant informs the other ants about the quality of the path they just travelled by depositing a proportional amount of pheromone on that path. The main difference in ADR is that the pheromone deposited on the paths is a function of how well the distribution of the population of ants through the available paths benefits the colony, by not having an excess of ants on a single path.

This draws a parallel with traffic networks, in which traffic jams may occur when all vehicles use the best route available to them, resulting in reduced network efficiency.

In Section 4.1 an introduction of ACO algorithms is presented and the benefits it provides stated. In Section 4.2 the most basic ACO algorithm is explained. Section 4.3 introduces the new cost function designed for the ADR algorithm and Section 4.4 introduces the ADR algorithm. Section 4.5 shows the results of the algorithm for a simple one way traffic network.

4.1 Introduction

ACO is based on the ants' ability to find the best path leading to food sources and transmit this information to the rest of the colony. The objective of an ACO algorithm is to find the least costly path between an origin and destination nodes in a graph consisting of a set of edges. Edges have associated cost functions and are connected by nodes, which is similar to a routing problem. There are a few reasons why ACO was chosen above other routing algorithms that exist:

- Excellent traffic assignment and routing results, with faster than average processing time (Bullnheimer et al., 1999; Matos and Oliveira, 2004; Tatomir and Rothkrantz, 2006);
- Framework gives several different possible options to tackle the problem;
- Can be made into a distributed problem, which is a vital component, since if centralized approach were to be used, routing choices in place A would affect traffic in place B 500 km away, making computation requirements impossible to meet. This approach will have also to take into consideration where to draw the boundaries of each region;
- Does not need to be reinitialized since between optimization runs it saves the pheromone states. Unless conditions in the real network have changed drastically, the algorithm will converge faster, due to already being near the optimum.

Figure 5 shows the control diagram of the problem. Let ρ_r be the density in road r, $n_{\mathrm{c},r}$ the number of cars heading for road r, $p_{r,r'}$ be the distribution of flows at each intersection, $\rho_{\mathrm{aco},r}$ be the densities resultant from the distribution of flows according to the traffic model (3) used by ADR, and $q_{\mathrm{in},o}$ be the flow of cars entering through origin of the traffic network o.

In the next section the most simple ACO algorithm will be presented: the Ant System (AS).

4.2 Ant System

Let us state the general problem that the AS algorithm will solve. The problem consists in finding the shortest path between two nodes, of which an example is shown in Figure 6 where we want to go from node 1 to 4.

We then have the problem of minimizing the cost of paths between pairs of nodes along $G = (V, E)$, where V is the set of vertices and E is the matrix of the edges between nodes. The graph has $n_{\mathrm{n}} = |V|$ nodes. Let path length L^a be defined as the distance length travelled by ant $a = 1, ..., n_{\mathrm{a}}$ from the origin until the destination. Each edge (i, j) will have a pheromone concentration value τ_{ij}.

The algorithm starts with the initialization of n_{a} ants at the origin node where they start their journey to the destination. At each iteration t, each ant will have to decide their next action through the following probability function

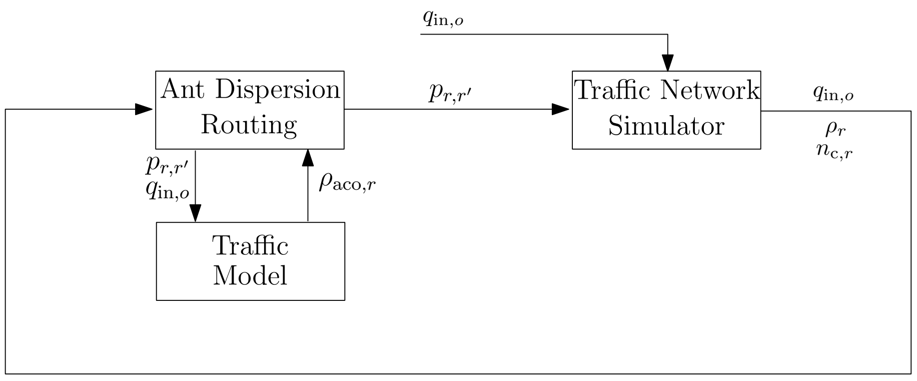


Figure 5: Diagram of the control loop consisting of the traffic network simulator, the ADR algorithm, the ADR traffic model (3), and the information exchanged between them.

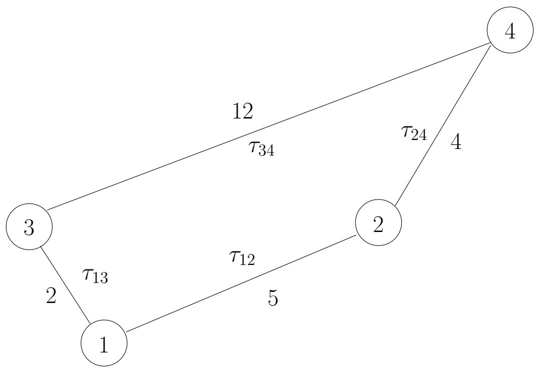


Figure 6: Network where 1 is the origin node and 4 is the destination node, and each edge has an associated pheromone value and distance cost.

$$p_{ij}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)}{\sum_{j' \in \mathcal{N}_i} \tau_{ij'}^{\alpha}(t)} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} , \tag{11}$$

where $\mathcal{N}_i$ is the set of possible node choice for an ant at node i. The parameter α is used to determine pheromone impact: set α to zero and all possible nodes have the same probability of being chosen. An increase in the parameter α gives a higher bias to high pheromone edges.

After an ant reaches its destination, its path is analysed for possible loops by checking if the same node was visited twice. If that is the case, the sequence of edges between the repeated nodes is removed from the ant's memory. The ant then retraces its steps depositing pheromone on the edges chosen. Obviously the pheromones deposited should be proportional to the inverse of the travelled length of the path, which is our cost function to be minimized. Let $\Delta\tau_{ij}^{a}(t)$ be the pheromone deposit by ant a at time step t, and the pheromone deposit equation is:

$$\Delta\tau_{ij}^{a}(t) = \frac{1}{L^{a}(t)} . \tag{12}$$

As we can see in Figure 6, we can expect to see the higher values of pheromone to be τ_{12} and τ_{24}, which is the shortest path. Ants that used a certain edge deposit the pheromone calculated in (12) in that edge. The new level of pheromone is then calculated:

$$\tau_{ij}(t+1) = (1-\rho_{\text{ev}})\tau_{ij}(t) + \rho_{\text{ev}} \sum_{a=1}^{n_{\text{a}}} \Delta\tau_{ij}^{a}(t). \tag{13}$$

where $\rho_{\text{ev}} \in (0,1]$, is the evaporation rate. The evaporation rate component is introduced so that the ants do not converge too rapidly, and spend some time exploring the graph. Only the edges that receive new pheromone deposits evaporate, so that the algorithm does not converge to sub-optimal results. If ρ_{ev} is near 0 almost no evaporation occurs and the ants cannot influence each other, since also no deposit occurs. If ρ_{ev} is near 1 at each iteration almost all pheromone evaporates turning the choosing of the next node based solely on the last iteration step.

The AS algorithm (used as a shortest-path finder) can be summarized as follows, where $S^a(t)$ is the sequence of edges used by ant a.

Algorithm 2 AS Algorithm

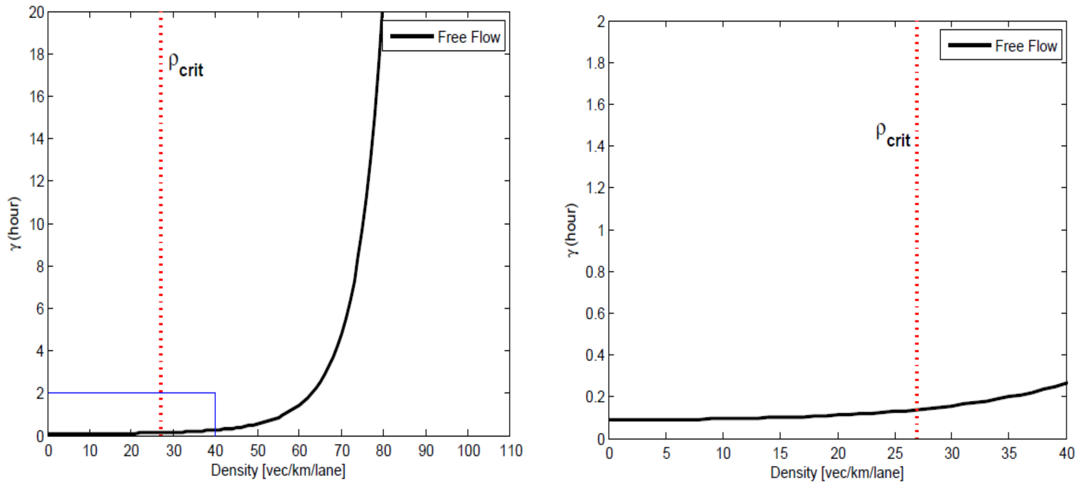
```
1: Given graph G
2: Initialize $\tau_{ij}$ and $t$
3: Place $n_{\text{a}}$ ants on the starting node
4: repeat
5:     for each ant $a = 1, ..., n_{\text{a}}$ do
6:         $S^a(t) = \emptyset$
7:         Construct a path $S^a(t)$
8:         repeat
9:             Select next node based on the probability defined in equation (11)
10:            Add edge $(i, j)$ to path $S^a(t)$;
11:        until destination node has been reached;
12:        Remove all loops from $S^a(t)$;
13:        Calculate the path length $S^a(t)$;
14:        Update pheromone values $\tau_{ij}$, using equation (13)
15:    end for
16:    t=t+1;
17: until stopping condition is true
18: Return the path $S^a(t)$ with smallest $f(S^a(t))$ as the solution;
```

The Ant Dispersion Routing (ADR) algorithm will be based on the routing components of the above algorithm, however, the social components of the AS algorithm will be taken a step further. New features are added to the basic AS, since the problem to be solved is quite different from the ones where AS has been applied to in the past. The main modification to AS originates from the fact that it finds an optimal path, and that means a single path. In our case we want to distribute ants (traffic) into several near optimal paths. A mechanism will have to be created in order to avoid all ants converging to a single path, by creating a cost function that successfully represents the goals of this paper.

4.3 Travel Time Cost

First the travel time cost function will be discussed, since it formalizes Condition A specified in Section 2. Since the main objective of the algorithm is to minimize the origination of traffic jams, it should not allow densities to pass the critical level ρ_{crit} of the fundamental diagram of traffic. The travel time cost γ_r is the time it costs to travel a stretch of highway, and this is calculated by integrating the highway's length over its speed ($t = x/v$). In our case, since we are using the same density for each segment in the ADR algorithm, the travel time cost becomes the total length of the highway over the theoretical speed as a function of the density:

$$\gamma_r = \frac{L_r}{V_r(\rho_r)}. \tag{14}$$



(a) Personal Cost Function γ_r (b) Zoom in on ρ_{crit}

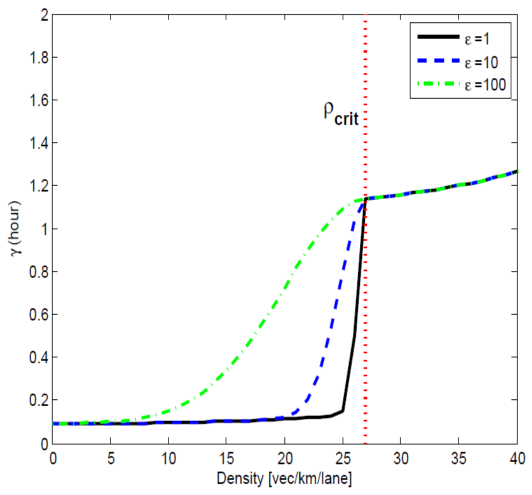
Figure 7: Function γ_r, calculated using (14), as a function of the density in a highway of length 10 km.

Figure 7 shows the travel time cost function, and it is noticeable that passing the critical point has little impact in terms of cost. While (14) does accurately measure the cost impact of different densities, it should be made so that passing the critical level of density ρ_{crit} is severely penalized. For that reason a new component must be added to the γ_r function that does just that, where for levels of density below ρ_{crit} is a bell function with its maximum peak M being achieved at critical density, and for levels of density above ρ_{crit} is the constant M:

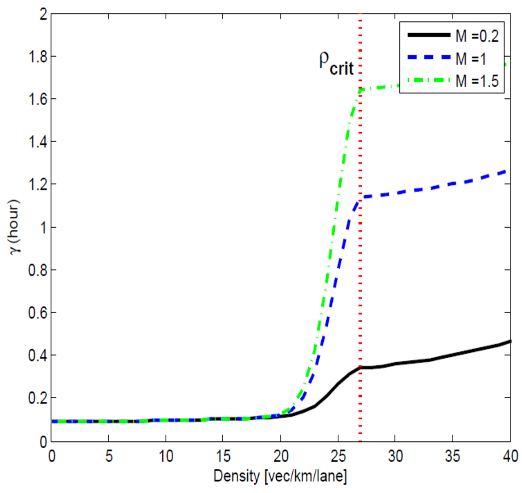
$$\gamma_r = f_\gamma(\rho_r) = \begin{cases} \frac{L_r}{V_r(\rho_r)} + Me^{\left(-\frac{(\rho_r - \rho_{\text{crit}})^2}{\epsilon}\right)} & \text{if } \rho_r \leq \rho_{\text{crit}} \\ \frac{L_r}{V_r(\rho,r)} + M & \text{if } \rho_r > \rho_{\text{crit}} \end{cases}, \tag{15}$$

where the parameter ϵ is the steepness of the bell function, and lower values of ϵ result in higher steepness of the function. The effects of the parameters ϵ and M can be seen in Figure 8, and the differences they cause in γ_r shows that careful tuning will be needed, specially for high vehicle density problems. Based on experimental results of simple networks (Appendix C), the parameters M and ϵ were tuned.

The above travel time cost function will enable Condition A to be met, 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 ADR algorithm results in traffic jams. Looking at Figure 8 an analogy can be made, where the function's steepness is a hill for an ant to climb, and the ant would rather circle around it and find a best route. Conditions B, and C will be discussed in Section 4.4.2.



(a) γ_r for varying steepness ϵ $(M = 1)$



(b) γ_r for varying magnitude M $(\epsilon = 10)$

Figure 8: γ_r, calculated using (15), as a function of the density in a highway of length 10 km.

4.4 Ant Dispersion Routing

The Ant System is the basis of a new algorithm in the class of the Ant Colony Optimization algorithms: the Ant Dispersion Routing. In the literature, ACO algorithms, which have proved to provide fast results, are used extensively in distributed optimization problems, and are particularly practical for network problems, since the optimization is done in a graph framework.

AS is used as the starting point for ADR, due to other more complex ACO algorithm having certain features that are not helpful or even prejudicial for the problem we want to solve. For example in (Dorigo et al., 1996) the algorithm uses an heuristic component used to help ants converge faster by having them biased to certain paths. Since we do not want our algorithm to converge to specific routes, it is a feature that will not yield good results. However, under a different framework, the heuristic component can also be benefitial to the ADR convergence time if it can bias the algorithm towards better distribution of flows. This is however not the focus of the current research.

Another example is the Ant Colony System (Dorigo and Gambardella, 1997) which introduced the elitist ants feature, where only the ants that found better solutions were allowed to deposit pheromone. This again is not a good feature for our problem, since we are evaluating a network solution, and the solutions of *each* ant are required. Therefore the AS is used as the basis for the ADR algorithm, since it includes the fundamentals of the ACO, while being relatively simple, and a good platform to build upon.

AS is designed so that it converges to the best solution, which in a routing problem, corresponds to the least costly route. The algorithm converges to a certain solution due to the fact that as more ants use it, the value of the pheromone on that solution increases until every ant uses it.

Ideally, in our case the algorithm should converge to a certain value of probability for each solution (route) smaller than 1, so that network efficiency is optimized (though in low density cases routing all vehicles to the same route is the network optimum). In this section we present the modifications that lead to the ADR algorithm, and the optimization of a network.

ADR is composed of two separate algorithms: a normal ant-based routing algorithm that finds the multiple shortest paths based on present traffic conditions, and the optimization algorithm which finds the correct distribution of flows on these paths that optimizes the network conditions.

4.4.1 Initialization

The AS has been successfully applied to online traffic routing in the literature (Tatomir and Rothkrantz, 2006), and thus the task at hand is to adapt it into our ADR framework. The routing algorithm is initialized like a normal AS algorithm, with the network being loaded, and the pheromone levels in all roads being set to the same small value. The probability function changes slightly from (11), to account for the fact that in ADR a different notation is used, with road indexes instead as opposed to node indexes, for simplification purposes. The probability of choosing each road is:

$$p_{r,r'} = \begin{cases} \frac{\tau_{r'}}{\sum_{r'_j \in \mathcal{N}_r} \tau_{r'_j}} & \text{if } r' \in \mathcal{N}_r \\ 0 & \text{otherwise} \end{cases} . \tag{16}$$

All ants pick their routes using the probability function (16), and then all routes are evaluated to decide on the fastest routes that will be used in the optimization process. This is done by calculating the travel time cost of each route φ_i based on the highest densities of each road r given by the METANET simulator $\rho_{\text{meta},r}$. In order to do this the travel time cost γ_r of each road is computed using the cost function derived in Section 4.3. In this case the density value used is $\rho_{\text{meta},r}$ since we are evaluating the current state of the traffic network, which in this case is provided by the METANET simulator:

$$\gamma_r = f_\gamma(\rho_{\text{meta},r}) \, . \tag{17}$$

The travel time cost φ_i of the whole route i is then just the addition of the individual travel time costs of the roads that compose that route:

$$\varphi_i = \sum_{r \in \mathcal{R}_i} \gamma_r \, . \tag{18}$$

Finally the routing algorithm adds pheromone to the routes identified as the fastest $i = 1, ..., n_\text{i}$ (proportionally just as in the ACO algorithm), and also removes all pheromone from roads that are not part of any route. This is done in order to reduce both the network size, and also immediately remove all bad solutions that can hinder the algorithm performance and severely increase the computational time. If pheromones on routes are set to zero then those roads become invisible to ants, thus reducing the network size, 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:

$$\Delta\tau_i = \frac{1}{\varphi_i} \, . \tag{19}$$

Let ρ_ev be the evaporation rate, and let $\mathcal{I}$ be the set of roads that compose the routes $\mathcal{R}_i$, and the new pheromone levels are:

$$\tau_r \leftarrow (1 - \rho_\text{ev})\tau_r + \sum_{\substack{i \in \mathcal{I}; \\ r \in \mathcal{R}_i}} \rho_\text{ev} \Delta\tau_i \, , \forall r \, ; \, \exists i \in \mathcal{I}; r \in \mathcal{R}_i \, . \tag{20}$$

The new pheromone level is the previous pheromone level discounted by the evaporation rate plus the new pheromone deposit by ants using that route.

This routing algorithm while simple, has the advantage, that it can find the best routes. It also transforms the traffic network of the traffic simulator into a reduced network, composed only of the routes of interest that will be used in the optimization part of the ADR algorithm. The pseudo-code of the initialization component of the ADR can be seen in Algorithm 3.

Algorithm 3 ADR Initialization

```
1: Given traffic network
2: Initialize $\tau_r$
3: for each destination $d = 1, ..., n_\mathrm{d}$ do
4:     Place $n_\mathrm{a}$ ants on the origin node
5:     for each ant $a = 1, ..., n_\mathrm{a}$ do
6:         $\mathcal{R}_i^a = \emptyset$
7:         Construct a route $\mathcal{R}_i^a$
8:         repeat
9:             Select next road based on the probability defined in equation (16)
10:            Add road $r$ to route $\mathcal{R}_i^a$
11:        until destination node has been reached;
12:        Remove all loops from $\mathcal{R}_i^a$
13:        Calculate road costs $\gamma_r$ of roads in $\mathcal{R}_i^a$ using equation (17)
14:        Calculate the route cost $\varphi_i$ of $\mathcal{R}_i^a$
15:        Update pheromone values $\tau_r$, using equation (20)
16:    end for
17: end for
18: Return the $n_\mathrm{i}$ routes in $\mathcal{R}_i$ with smallest cost $\varphi_i$;
```

4.4.2 Flow Optimization

Now that the fastest routes have been identified, ADR can proceed to optimize the distribution of traffic flows in this reduced network. As was stated earlier a clear departure must be made from the AS algorithm when it comes to the optimization since the AS always converges to only one solution. We want to optimize the distribution of flows that leads a better usage of the network as opposed to finding the optimal route which will benefit the first drivers to use it, but in time and for many drivers, may cause severe congestion. This happens due to the fact that the more ants use a route, the more attractive that route becomes, due to more ants depositing pheromone on it. Thus the pheromone deposit function in ADR cannot be based on the number of ants using it, and instead the pheromone deposits are based on the aggregated solution of all ants. Also the function should be a function of the costs that represent the conditions stated in Section 2, the cost of a route φ_i, and the network cost Ω.

Similarly to the initialization presented above, the ants have the objective of finding the best solution according to the probability function defined in (16). The number of ants must then be converted into number of cars so that densities can be correctly calculated according to the traffic model used by ADR. Let $n_{c,i}$ be the number of cars using route i, and $n_{a,i}$ be the number of ants using route i, and the number of cars using each road is:

$$n_{c,i} = n_{a,i} \frac{n_\mathrm{c}}{n_\mathrm{a}}. \tag{21}$$

Now that we have this information, the traffic model of the ADR, which in this case is simply the fundamental diagram of traffic (3), can calculate the densities on each road of the network. Using the inflow of the entrance we want to control, the theoretical density $\rho_{\mathrm{adr},r}$ as predicted by the ADR algorithm is:

$$\rho_{\mathrm{adr},r} = \frac{\frac{n_{a,r}}{n_\mathrm{a}} q_{\mathrm{in},o}}{\lambda_r V(\rho_\mathrm{crit})}. \tag{22}$$

This relation stems from the fact that we know that in macroscopic traffic theory the relationship between flow, density, and speed is $\rho = q/V$, so we know that the flow on a certain road is the fraction of ants using it, times the inflow. The flow is then divided by the number of lanes since we define density as a function of the number of lanes. The density is directly used in the travel time cost function for each road that was derived in Section 4.3:

$$\gamma_r = f_\gamma(\rho_{\mathrm{adr},r})\,. \tag{23}$$

Since the ADR algorithm deals with routes instead of roads, the travel time of each route φ_i must be computed by summing the individual road costs of each route, as was already derived in (18). The final cost component can now be calculated, and that is the network cost Ω, which is the average cost of each driver. Let $n_{c,i}$ be the number of cars using each route i and the network cost is computed as:

$$\Omega = \frac{\sum_{i=1}^{n_{\mathrm{i}}} \varphi_i n_{c,i}}{\sum_{i=1}^{n_{\mathrm{i}}} n_{c,i}}\,. \tag{24}$$

The network cost is computed as the average of the cost of every driver using the ADR algorithm, that is, every driver using one of the n_{i} routes optimized by the ADR. While Condition A was addressed in the cost function γ_r, Conditions B and 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 (12), it will represent a minimization of differences between costs (network and per route). Intuitively we want to minimize the cost of each route φ_i and the network cost Ω, and the pheromone deposit should be a weighted sum of the minimization of these components. Let W_1 and W_2 be these weights and the new pheromone deposit is defined as following:

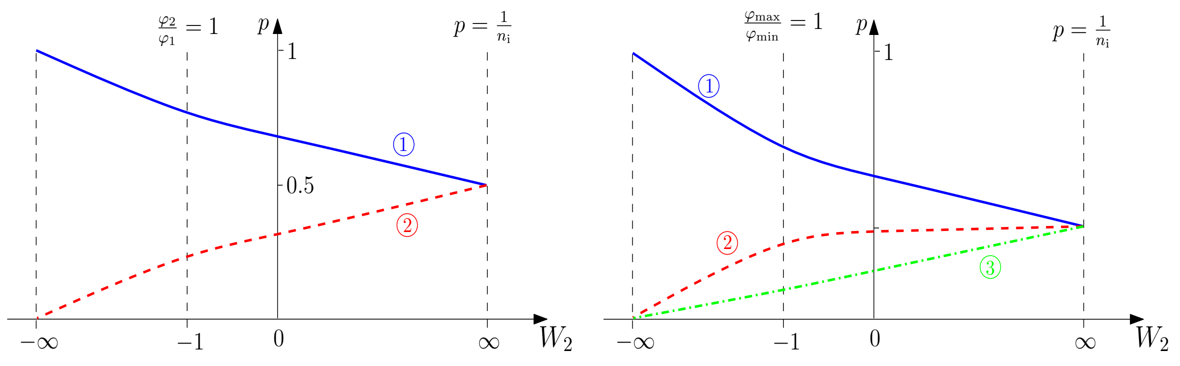
$$\Delta\tau_i = \frac{W_1}{\varphi_i} + \frac{W_2}{\Omega}\,. \tag{25}$$

This equation still makes the algorithm converge to only one route. That happens because Ω at each iteration of the ADR will always be equal on all routes, and thus the algorithm only minimizes φ_i. That is why it was stated that we want to minimize differences between costs, and use to our advantage the fact that Ω is constant for all routes. When distributing flows through optimal and near-optimal routes, it can be assumed that there is one point where the cost of all routes can be equal. That equilibrium occurs when $\varphi_1 = \varphi_2 = \cdots = \varphi_{n_{\mathrm{i}}} = \Omega$. This can be defined in the pheromone deposit equation (25) by defining $W_1 = 1$ and $W_2 = -1$, that is, by minimizing the difference between the constant network cost Ω and the cost of each route:

$$\min_{c_1,\ldots,c_{n_{\mathrm{i}}}} |\varphi_i - \Omega|\,. \tag{26}$$

It was stated we do not want the cost of all routes to be the same, but rather a bias towards the shortest route if possible. That is done by making $W_2 \in (-1, 0)$, giving more importance to the cost of a route.

However a thorough study of how the algorithm reacts for different values of $W_2 \in (-\infty, \infty)$ can be seen in Figure 9 ($W_1 = 1$). Increasing W_2 will force all routes' probabilities to be equal since it removes the personal route cost from the equation and adds the same pheromone to all routes $W_2 >> W_1 \Rightarrow \Delta\tau_i = \frac{W_2}{\Omega}$. If we let W_2 go to -1 the algorithm will try to minimize the difference between the inverse of the costs φ_i of all routes. Also the term $\frac{W_1}{\varphi_i}$ fulfills Condition C



(a) Two road network

(b) Three road network

Figure 9: Variation of the probabilities for each road as a function of the weight W_2 with $W_1 = 1$. The first figure shows the probabilities for a two route distribution of flows, and the other figure for a three route distribution of flows.

if $W_2 > -1$, since it gives higher pheromone deposits to less costly edges ($\frac{W_2}{\Omega}$ is the same for all edges at each iteration) making more ants converge to them.

The tuning of the weight W_2 depends only on the constraints we want to set between the difference of the least and most costly routes. For a small difference of around $5-10\%$, the weight W_2 should be close to -1, for larger differences of around 20% the weight should tend to 0. Results of the tuning of weight W_2 can be seen in Appendix B.

The last step of the ADR algorithm is again the pheromone update equation (20). The optimization part of the ADR algorithm can be seen in Algorithm 4, and together with the initialization algorithm represents the full ADR algorithm. In the next section the working of the algorithm is illustrated by showing a problem of a simple one way network with only two roads for the ants to choose from. In the next chapter the algorithm will be tested in a full network with multiple origins, and multiple destinations.

4.5 Results

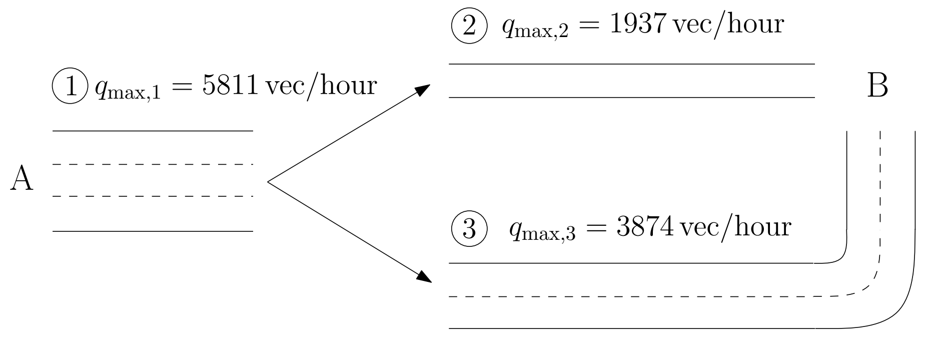
To test the algorithm, and tune the new parameters we will use a simple one way network with two routes to choose from. An extra highway will serve to feed traffic into the network, and to create a more realistic approach to the intersection of interest where the flow distribution will be optimized. Figure 10 represents the network, where all traffic starts at point A and heads toward point B. The first highway has three lanes, the shortest and theoretically fastest highway has one lane, and the longer highway has two lanes. The maximum flow $q_{\max}$ of each highway is the maximum point taken from the fundamental diagram of traffic in the $q(\rho)$ curve multiplied by the number of lanes, which means that these are the parameters that the ADR will extract from its traffic model.

The parameters of the network and of the METANET simulator are as follows:

- Highway 1: $L_1 = 5\,\text{km}$, $\lambda_1 = 3\,\text{lanes}$, $q_{\max,1} = 5811\,\text{vec/hour}$;
- Highway 2: $L_2 = 10\,\text{km}$, $\lambda_2 = 1\,\text{lanes}$, $q_{\max,2} = 1937\,\text{vec/hour}$;
- Highway 3: $L_3 = 13\,\text{km}$, $\lambda_3 = 2\,\text{lanes}$, $q_{\max,3} = 3874\,\text{vec/hour}$;
- Segment length: $l = 0.5\,\text{km}$;
- Time: $T = 10\,\text{sec}$;

Algorithm 4 ADR Flow Optimization

```
1: Given $n_\mathrm{i}$ best routes, $\rho_\mathrm{meta}$, and $\tau_r$
2: Remove roads not present in the $n_\mathrm{i}$ best routes from the network
3: for $t = 1, ..., N_\mathrm{t}$ do
4:     Set $A_i = 0 \quad \forall \quad i = 1, ..., n_\mathrm{i}$
5:     Place $n_\mathrm{a}$ ants on the origin node
6:     for each ant $a = 1, ..., n_\mathrm{a}$ do
7:         $\mathcal{R}_i^a = \emptyset$
8:         Construct a route $\mathcal{R}_i^a$
9:         repeat
10:            Select next road based on the probability defined in equation (16)
11:            Add road $r$ to route $\mathcal{R}_i^a$;
12:        until destination node has been reached;
13:        Remove all loops from $\mathcal{R}_i^a$;
14:        Register which route $i$ was used by ant $a$
15:    end for
16:    Normalize number of ants into number of cars using equation (21)
17:    Calculate predicted densities $\rho_\mathrm{aco}$, cost of roads $\gamma_r$, cost of routes $\varphi_i$ and social cost $\Omega$
18:    Update pheromone values $\tau_r$, using equation (20)
19: end for
20: Return node splits $p$
```



① $q_{\mathrm{max},1} = 5811$ vec/hour
② $q_{\mathrm{max},2} = 1937$ vec/hour
③ $q_{\mathrm{max},3} = 3874$ vec/hour
A
B

Figure 10: Simple traffic network with two possible routes to choose from.

- Exponential model: $\rho_\mathrm{crit} = 27$ vec/km/lane, $v_\mathrm{free} = 110$ vec/hour, $a_m = 2.34$;
- Speed equation parameters: $\eta = 30$ km^2/hour, $\tau = 10$ sec, $\kappa = 20$ vec/km.

Table 1 shows how the algorithm behaves with different inputs (flows entering highway 1). Condition B is represented by the four last columns with the respective costs for each route and network cost Ω, and relative costs between them. Condition C is analysed via the second and third columns which show the percentage of cars using each highway. It should also be noted that route 1 is the route composed of roads 1 and 2, and route 2 of roads 1 and 3.

For big inflows of 5000 and 5500 Condition B ($\frac{\varphi_\mathrm{max}}{\varphi_\mathrm{min}}$) loses importance since the algorithm is trying to avoid jamming both roads. For small inflows the difference between both highway costs also increases. This happens due to the fact that under low density, the algorithm cannot minimize the difference between costs of highways 2 and 3 due to not having enough flow of cars to impact costs. The algorithm can only accommodate Condition B under certain traffic conditions, since if the inflow into the network is small, it cannot increase density on the shorter

Table 1: Several Optimization runs using different inflows.

q_{in} **(vec/hour)**	$n_{c,2}$ (%)	$n_{c,3}$ (%)	φ_1 **(sec)**	φ_2 **(sec)**	Ω **(sec)**	φ_2/φ_1
1000	100	0	504	-	504	-
2000	72.6	27.4	520	592	550	1.139
3000	51.5	48.5	541	602	573	1.113
4000	40.6	59.4	555	623	601	1.123
4500	35.9	64.1	561	653	623	1.164
5000	34.1	65.9	583	690	657	1.184
5500	34.8	65.2	665	739	714	1.111

highway to minimize cost differences, and if the inflow into the network is high, the algorithm will have to prevent traffic jams and disregard average costs. ADR should not be used under small inflow conditions, since it has no added benefit under those conditions. However, under medium inflow conditions and high inflow conditions it has several advantages, by first optimizing the network usage, and secondly avoiding traffic jams.

Some results of the ADR algorithm working on a simulator are now presented. In Figure 11 a constant inflow near critical levels ($q_{\text{in}} = 5500$ vec/hour) is used throughout the test trial. The algorithm converges directly in its routing policy, and keeps the inflow for highway 2 below the critical level. Since the algorithm is dealing with a near critical inflow, it is pro actively avoiding traffic jams, and it has to disregard Conditions B and C in order to do that. If the algorithm routed more cars to the fastest road as stated in Condition C a traffic jam would immediately originate in highway 2.

In Figure 12 other properties of the algorithm are explored. The inflow at highway 1 is of 5500 vec/hour for the first 15 minutes, 5800 vec/hour (critical) for the next 10 minutes, and then it is lowered to 2500 vec/hour for the rest of the trial.

The algorithm has a slight problem handling the critical inflow due to the fact that it is now fighting to keep both the densities of highways 2 and 3 below critical level, and the exploratory randomness introduced by the ants makes it occasionally slightly tip over the critical level for one or another. However, this is a problem that is not expected to be seen in a real network scenario, since the previous node would stop routing that amount of cars into highway 1 to begin with. Still, as soon as the inflow in highway 1 lessens, the algorithm recovers quickly and here the main objective of the algorithm as stated in Section 2 is observed, where the algorithm constrains the difference between the cost of using each highway, and routes the bigger percentage of cars into the faster highway.

A comparison was also made to measure the benefits that the ADR algorithm has. The same simulation will be run with all drivers using the ADR algorithm, all drivers using a normal routing algorithm (routing all the drivers to the shortest route, unless its jammed), and a network where most drivers use the shorter road, and the rest use the longest one (normal case scenario). The simulation will last 30 minutes and during the first 15 minutes the inflow at the first highway is $q_{\text{in}} = 1500$ vec/hour, which is below the critical level of road 2 (where most cars will head for). During the last 15 minutes the inflow at the first highway is of $q_{\text{in}} = 2300$ vec/hour. Recall that route 1 is composed of roads 1 and 2, and route 2 of roads 1 and 3. During the first 15 minutes routing every car for the shortest highway is perfectly acceptable since the inflow at highway 1 is well below the critical level of $q = 1937$ vec/hour. The normal routing algorithm, and also the ADR algorithm do just this, since even if all cars are routed to the shortest algorithm, the increase in density will not make this highway slower than the longer one. Normal driver behaviour without using routing keeps the density in highway 2 lower, but drivers using highway

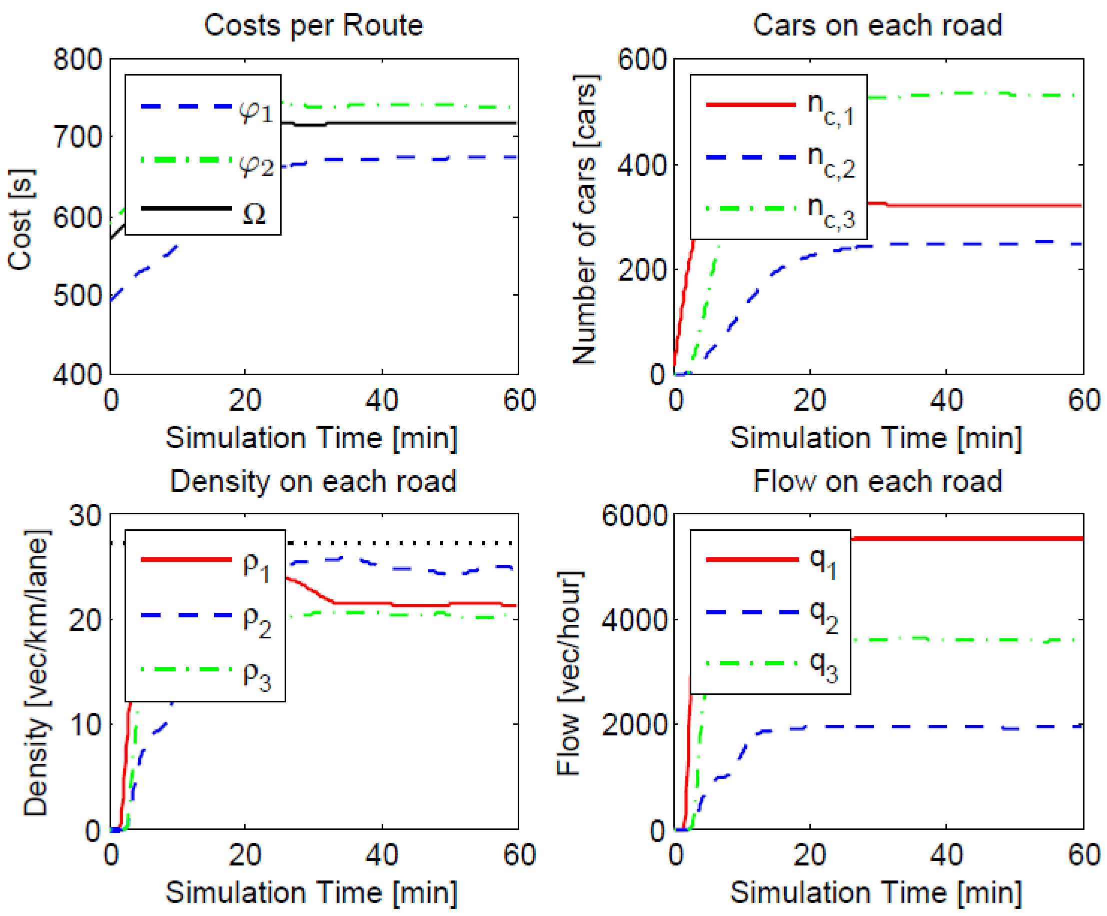


Figure 11: Test trial with the ADR algorithm running every 5 minutes, and an inflow at highway 1 of 5500 vec/hour.

3 have a route that is significantly slower than the other drivers.

When the inflow changes to a value above the critical level, the benefits of the ADR algorithm become immediately clear. First since it knows the inflow at highway 1 has changed, it proceeds to quickly route a certain percentage of cars to the wider highway, and thus the network cost Ω increases slightly in the first $2-3$ minutes of the new inflow. After that the pro-active routing policy is proved to work correctly since the flow and density values stabilize and good performance is achieved. The normal routing algorithm continues sending more and more cars to the shortest highway since it is still not jammed, and the density levels while higher, and affecting the flow of the highway, are still not near congestion levels. Also Condition B is always met, since when the inflow is $q_{\text{in}} = 1500$ vec/hour no vehicles are using route 2 making Condition B redundant. When the inflow is $q_{\text{in}} = 2300$ vec/hour the ADR algorithm distributes the flows accordingly and the costs of routes 1 and 2 become approximately 550 and 590 seconds ($\frac{\varphi_2}{\varphi_1} = 1.07$).

Also as seen in Figure 11 the ADR algorithm can keep the flows below critical levels until an inflow of around 5500 vec/hour, while the normal routing algorithm is already prejudicial to the traffic network with inflows of 2000 vec/hour.

This shows that the objectives of the algorithm are justified, can be accomplished, and are incorporated into its framework. The question now is, how does ADR scale to a full road network, with traffic that it does not control. In the next section these questions will be answered, by having the ADR algorithm negotiate a zone with several highway intersections, and different

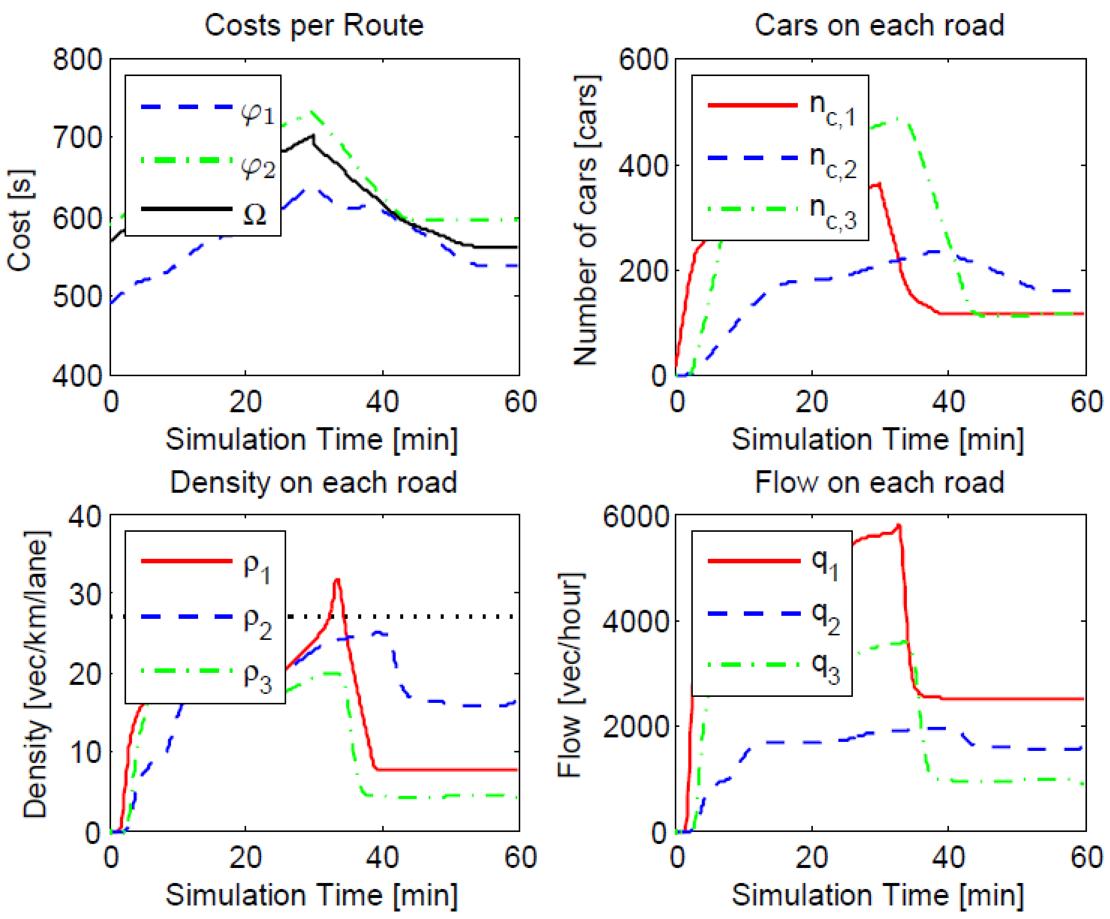


Figure 12: Test trial with the ADR algorithm running every 5 minutes, and a varying inflow at highway 1.

possibilities of routes.

5 Case Study: Singapore Expressway Network

5.1 Introduction

Singapore is a small island city-state located at the south of the Malay Peninsula. Since its independence this city-state has had a huge economic, population and overall quality of life growth, and is now one of the main business centres of Asia. It is also one of the wealthiest countries in the world, and most multi-national companies have their Asian business based in Singapore. While small (710 km^2), Singapore has a population of around 4.9 Million people, and a big percentage of this population either lives or commutes during the day to the central-south area of Singapore where the financial and commercial districts are located, near the Singapore River.

While Singapore is serviced by one of the best public transportation systems in the world, a high density of traffic is ever present in these central areas, and space is severely limited to create extra infrastructures. The Singapore government actively finances research in traffic control areas, and in 2005 was the first country to implement a traffic prediction system in its Expressway Network (highways in Singapore are referred to as expressways). The need to apply

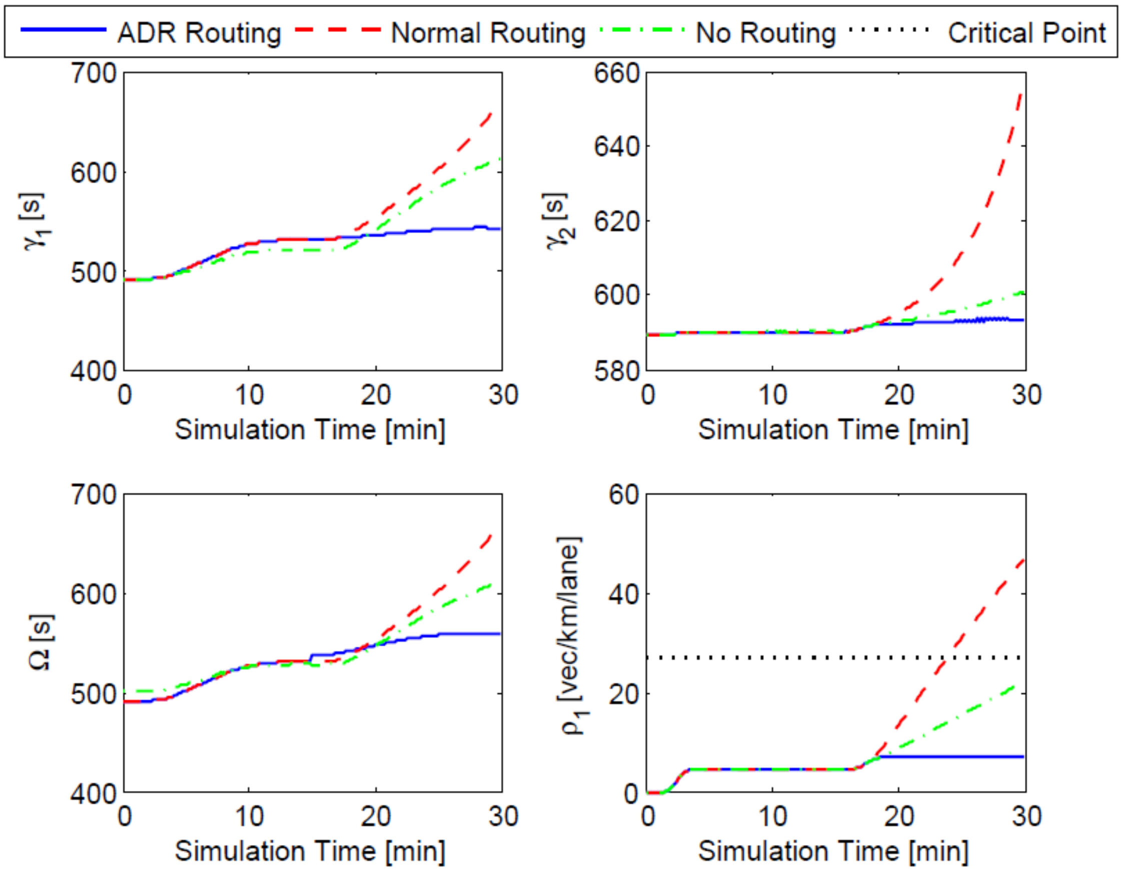


Figure 13: Comparison between different types of driver behaviour according to routing algorithms.

a traffic prediction system came from the government wanting to introduce a variable real time road pricing to fight congestion. Developed together with IBM the system used historical traffic data of certain places in the expressway and real time traffic states extracted from several sources such as CCTV cameras. It used that information to predict levels of congestion up to one hour in advance, and that way, controlling the costs of electronic road pricing (Figure 14) to influence traffic to use certain expressways. Accuracies of the predictions ranged between 85-90%.

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. While again the assumption is that all traffic states are available through floating car data, control of only one zone is available, and all drivers coming from other areas do not use the ADR algorithm.

5.2 Singapore Expressway Network

The Singapore expressway network was chosen due to having a 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 transition between highways is assumed to be limited to just removal or 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 (Figure 15) will be used since those are enough to test the ADR algorithm

(a) Singapore Expressway (Prabhas, 2007). (b) Electronic Road Pricing (ERP) in a Singapore expressway (Wiki, 2005).

Figure 14: Pictures of Singapore expressways and technology available on them.

and contain the points of the island that are usually subject to traffic jams.

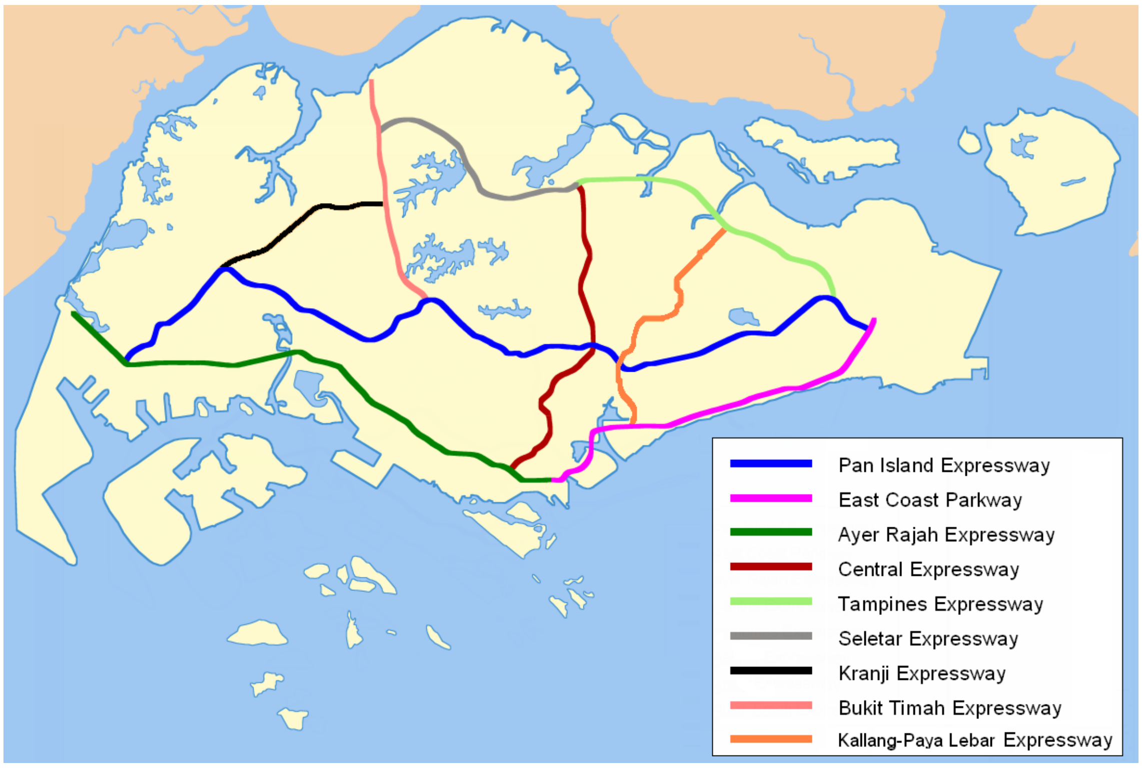


Figure 15: Expressway Network of Singapore (Yong, 2007).

The network is composed of 18 stretches of highway (36 if we consider both directions), 8 origins and 8 destinations, as seen in Figure 16 and Appendix A. The critical area of the network is in the business district, the existence of several origins and destinations in that area, and the fact that the Central Expressway has only two lanes which severely limits capacity. Not only that, the Central Expressway is the only one that connects the northern residential areas of the island (including traffic coming from Malaysia) to the centre. In total, the business district area can be accessed through 4 different destinations, increasing the freedom of the ADR algorithm.

Traffic from and to the airport is exchanged with the network through origin and destination 4. The Kallang-Paya Lebar Expressway is also prone to high density traffic since it is a good

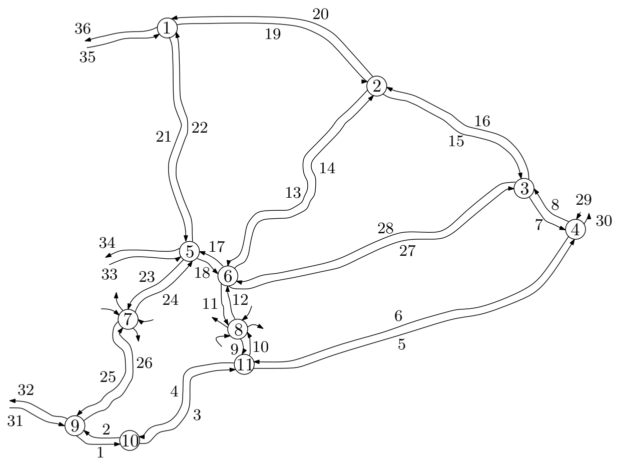


Figure 16: Network used by simulator and ADR.

alternative to entering the business district, though it has an extra lane, and deals only with a small fraction of the traffic coming from the northern part of the island.

5.3 Results

In this case study, we will first only open one origin in one of the edges of the network, and will control all traffic with the ADR algorithm. After that, the real traffic scenario case will be presented where all origins are open, and traffic enters through every possible origin of the network. However, only one of these origins is controlled by the ADR algorithm. The algorithm must be capable of routing cars through several different routes, while also accounting for the traffic it is not controlling.

5.3.1 Single Origin

In this first example the ADR algorithm will be applied at origin 4 (traffic coming from the airport and east region of the island), and all traffic is assumed to have as destination the business district (destinations $5, 6, 7, 8$). The ADR has then to first compute the ideal routes to disperse traffic to, and then optimize flows through them.

Intuitively, and looking at Figure 16, several possible routes exist, but two of them are considerably shorter than all others. These routes are $\mathcal{R}_1 = \{29, 6, 10\}$ and $\mathcal{R}_2 = \{29, 8, 28, 11\}$, with respectively 15 and 16 kilometres in length. The shorter route's biggest bottleneck occurs from road 6 to road 10, due to the decrease of one lane.

According to our fundamental diagram of traffic, which is slightly different due to the fact that Singapore Expressways have a 90 km/hour limit, the maximum flow per lane below the critical level of density is of 1585 vec/hour. If the inflow coming from the airport is larger than 4755 vec/hour, a dispersion of flows will be required and ADR will be used. The inflow will also never be above 6340 vec/hour, since that is the maximum capacity of road 29.

Recalling the Conditions that were stated throughout this paper, the ADR algorithm should keep densities on all roads below the critical level ($\rho_{\text{crit}} = 27\,\text{vec/km/lane}$), it should keep the travel time cost of all routes similar ($5\% - 10\%$), and the fastest route should have more cars using it.

If we set the algorithm to use the two best routes ($n_{\text{i}} = 2$), the first part of the ADR algorithm (Section 4.4.1) will return the routes mentioned above: $\mathcal{R}_1 = \{29, 6, 10\}$ and $\mathcal{R}_2 = \{29, 8, 28, 11\}$. This was to be expected since we are starting the traffic network under stable conditions with

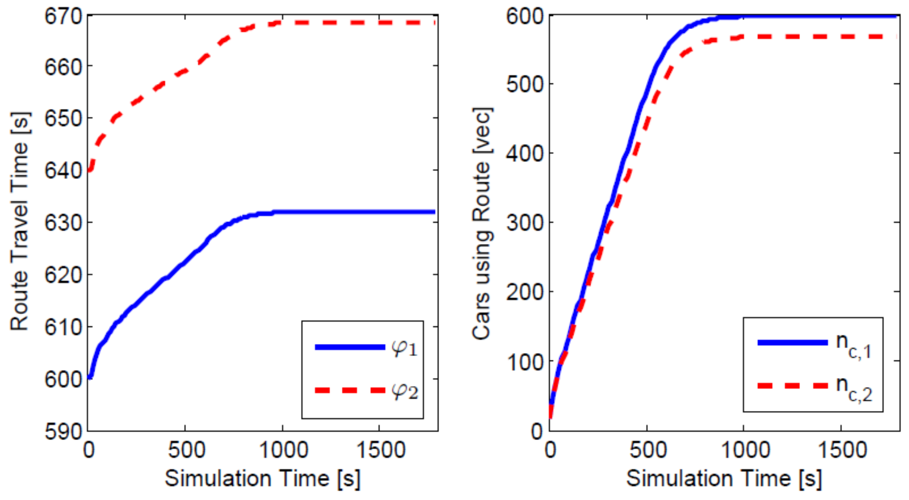


Figure 17: Performance of the algorithm with the option of two different routes $\mathcal{R}_1 = \{29, 6, 10\}$, and $\mathcal{R}_2 = \{29, 8, 28, 11\}$, for $q_{\text{in}} = 6000$ vec/hour.

all roads under free flow conditions, and the next best route has a length of 19.5 km. During the trial a constant flow of 6000 vec/hour is entering the network from the airport, and every car wants to arrive at the business district. The algorithm will then disperse flows between the two available routes as is shown in Figure 17, where we can see clearly that Conditions B and C have been satisfied by the algorithm. More cars are using route 1 ($n_{c,1} = 596$) than route 2 ($n_{c,2} = 572$) as seen in Figure 18. It can also be seen that the difference between the travel time cost of using each route is small, more precisely it is 5.86%.

In Figure 18 it is also noticeable that the final condition, of keeping the density of all roads below critical level, is met on all roads resulting in free flow conditions across the network. All relevant states of the network are also present in Figure 18 including the separate travel time cost of each road, flow, and density. The lines represent the sequence of each route and the traffic conditions that drivers experience on each road.

Table 2: ADR Performance with three routes.

	Route 1	Route 2	Route 3
φ [s]	631.4	666.8	805.3
Route Usage [%]	52.33	35.19	12.48

If we select the total number of different routes to be used as three ($n_{\text{i}} = 3$), then the algorithm will proceed to also use $\mathcal{R}_3 = \{28, 8, 28, 17, 23\}$. Table 2 shows how the algorithm performs when distributing flows between these three routes, and the results in it prove clearly that the optimal number of routes for this specific case between the airport and the business district is 2 and not 3. That is due to the fact that the third route not only is relatively much longer than the two others, it is also dependent of route 2. This means that most of route 3 is in fact route 2, that is, route 2 is part of route 3, and as such when the algorithm tries to route more cars towards the two shortest routes, it is in fact also routing cars to the longer route, making the minimization of the difference between travel time costs of each route impossible. The only road in route 2 that does not belong also to route 3 is road 11. However this road is both shorter and has more lanes than the alternative presented in route 3 so it makes no sense, and the algorithm has little incentive to use route 3.

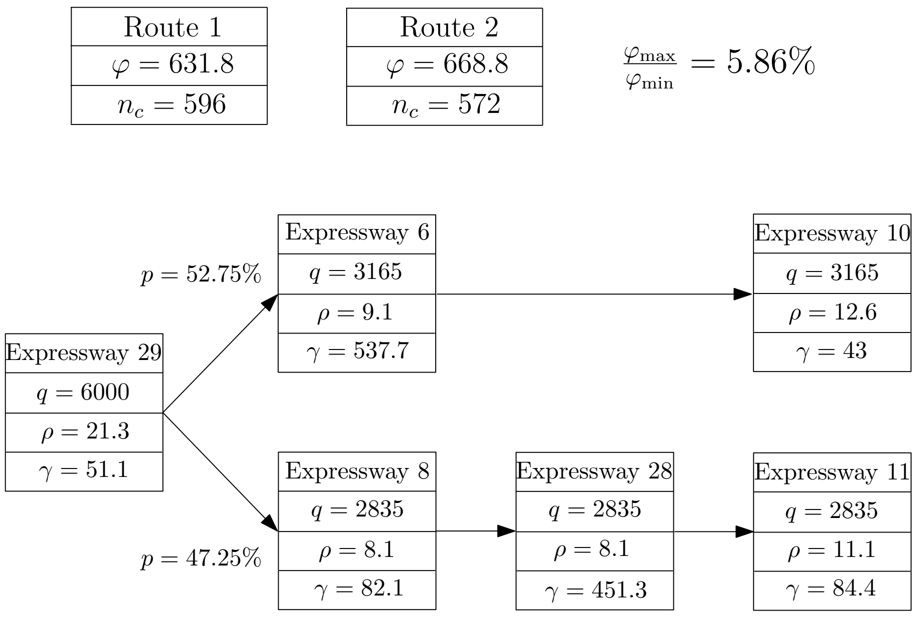


Figure 18: States of the roads that belong to the two different routes, and corresponding split of flows.

This is a limitation of the network, due to the fact that we are not using urban traffic. It is possible that route 3 would be useful if we were using an urban network of roads, and the ADR had to route to a specific point, instead of an area.

Since the algorithm was told to use three routes it will still route some traffic to the third route, and this results in Condition B not being satisfied since route 3 is over 27% slower than route 1. In this second example the traffic flow is split in two points of the network: first at the end of road 29, with 52.33% of traffic going for route 1 and the rest for routes 2 and 3. The second split occurs at the end of road 28 with 73.82% of the traffic using route 2 and the rest continuing for route 3. The only scenario where using the third route would become acceptable would be if a crash would occur in road 11, making the use of route 3 extremely helpful. However, this would become again a two route problem since then the ADR would only have to balance the usage of two routes.

In the next section a more realistic trial will be run, where the network has traffic that is not controlled by the ADR algorithm. It does however, have knowledge of the traffic network states, since all cars in the network transmit floating car data to a central station.

5.3.2 Multi Origin

For the mixed traffic scenario we will use again the ADR algorithm to control the traffic going from the airport (origin 4) towards the business district (destinations $5, 6, 7, 8$). For simplicity's sake, traffic is not flowing from the origins in the business district since this will be traffic in the opposite direction as the routes being controlled. In this scenario the ADR will check if there are roads already past the critical level of density, and then find the ideal routes from the airport to the business district. In Table 3 the state of each origin is defined, with 4 origins feeding traffic into the network. These drivers are always assumed to be using the shortest routes, which is realistic since these are low levels of inflows and will not create traffic jams.

Intuitively the algorithm will select the two same routes used in Section 5.3.1 ($\mathcal{R}_1 = \{29, 6, 10\}$ and $\mathcal{R}_2 = \{29, 8, 28, 11\}$), since all roads in those routes are still with low levels of uncontrolled

Table 3: Normal Traffic Conditions.

Origin	Destination	Flow[vec/hour]	Route
1	5	2000	$\{31, 1, 3, 10\}$
1	7	1000	$\{31, 26\}$
2	6	1000	$\{33, 18, 11\}$
2	8	1000	$\{33, 23\}$
3	1	1000	$\{35, 21, 23, 25, 32\}$
3	2	2000	$\{35, 21, 34\}$
3	4	1000	$\{35, 19, 15, 7, 30\}$
4	$5, 6, 7, 8$	4000	ADR

traffic. Also we select the number of routes to be used as two, since we have already proved that in this case, this is the optimal number for the dispersion algorithm.

We should recall that the algorithm has floating car data available and as such can calculate what the estimated density will be relative to the uncontrolled traffic on every road. Notice that there are already 2000 vec/hour using road 10 and 1000 vec/hour using road 11. This means that the one kilometre less that route 1 has, is compensated by route 2 having less cars on it at the start of the trial, making the difference in travel time cost between the two routes close to zero.

This is an important situation since now Condition C will be impossible to be met due to two reasons: now that the travel time cost of each route is the same, if the algorithm routes more cars to one road that road will become more costly than the other, and vice-versa. Also as we can see in Table 4 the algorithm will route more cars into route 2. That is due to the fact that originally without the ADR, traffic in road 10 (route 1) has higher density then in road 11 (route 2), and thus the ADR cannot route more cars to route 1 due to the penalty introduced in (15).

Table 4: ADR Performance for Mixed Traffic.

Route 1				Route 2			
$\varphi = 622.9$ s		$p = 47.08\%$		$\varphi = 665.5$ s		$p = 52.92\%$	
Road	γ [s]	ρ_{ctr}	ρ_{unctr}	ρ [vec/km/lane]	q_{ctr}	q_{unctr}	q [vec/hour]
29	45.1	15.65	0	15.65	5000	0	5000
6	528.4	6.65	0	6.65	2354	0	2354
10	49.4	8.59	11.35	19.94	2354	2000	4354
8	81.7	7.51	0	7.51	2646	0	2646
28	449.5	7.51	0	7.51	2646	0	2646
11	89.2	5.68	9.37	15.05	2646	1000	3646

Table 4 shows that Conditions A and 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 in road 10 with density $\rho = 19.94$ vec/km/lane. Also the difference between travel time costs of both routes is kept inside the usual interval of $5 - 10\%$ with $\varphi_2/\varphi_1 = 1.068$. It was clear that the algorithm in this scenario was limited by the uncontrolled traffic, though it did not even have any origins or destinations close by to influence the densities around the controlled origin. As such it is important to modify the ADR algorithm so that it can control the whole network and create a better global efficiency of the network, which will be discussed in the following section.

6 Discussion

Though outside of the scope of this paper, it is interesting to discuss possible ways of how this algorithm could actually be implemented in a real world traffic network. As it is, the algorithm will always improve the efficiency of the network, and only a smaller fraction of the drivers use routes that are more costly. The questions now are: how can ADR benefit all drivers, and how does ADR distribute the control outcome to the drivers. These problems can be solved at the same time, by logging the routing suggestions of the ADR for each driver.

Let us imagine a scenario with two routes where under ADR route 1 takes 20 minutes, and is used by 80% drivers, and route 2 takes 35 minutes and is used by the remaining 20% of the drivers. Without ADR, all drivers will use route 1, causing higher density and making the cost of travelling there 30 minutes. The selling point of ADR then would be that in 1 week each driver will be using route 2 once, and route 1 four times, making the travel time of all drivers per week the same. Secondly the cost per week of using the ADR will be lower than without ADR:

- With ADR: $20 \times 4 + 35 = 115$ seconds;
- Without ADR: $30 \times 5 = 150$ seconds.

The logging system can even be expanded further. For example similarly to electronic road pricing, where usually better and faster highways are paid, it is also possible to envision situations where drivers can pay to always use faster routes, and vice-versa. The logging system can also enable drivers to have preference of which day they may use the faster routes, such as in days with morning meetings, or business travels, where arriving fast to their destination is imperative.

Another possible addition to this logging system would be a reward system. Traffic infractions usually have monetary, and crime penalties associated with them. ADR can then be used to apply additional penalties which would incentivate good behaviour. A speeding infraction, besides the traditional penalties, can carry a penalty of 1 year where ADR would suggest slower routes to that driver. This way, the driver would have speed restricted (by using the slower route) which is the action that gave him an infraction in the first place.

This ADR can also helpful for emergency situations, since ambulances, police, and other public services will always have a fast route available to them, improving overall safety.

7 Conclusions & Future Research

7.1 Conclusions

In this paper, we have introduced a novel online traffic routing algorithm that optimizes the distribution of traffic flows. The Ant Dispersion Routing algorithm was designed to solve the traffic network equilibrium problem. This is a complex optimization problem since it has two conflicting objectives: reduce travel times of drivers, while improving network efficiency. This algorithm is model-based, scalable and computationally efficient. In ADR ants negotiate several optimal solutions to decide in the set of solutions that is best for the colony.

During the course of this paper, the results have benefited from the fact that we are using a simulator with a well known and similar traffic model as the one ADR is using also. However, the algorithm is more widely applicable and it should work with any simulator, and with real traffic, as long as the traffic model used in the ADR algorithm can calculate accurately speeds as a function of density for that particular situation. Theoretically it is even possible to use a simple traffic model as the one used in this ADR algorithm to control real traffic, if some robustness constraints are added. These constraints would be needed to account for different driving styles, and also infrastructures in the traffic network such as toll collection booths.

The algorithm proved to work accurately for different networks in size and shape, accomplishing the goals defined, and significantly reducing densities on roads while still being acceptable by the drivers. In the Singapore case study the algorithm was used successfully in a mixed traffic scenario, where most of the traffic is not controllable by the algorithm. It was also proved that it can work for any number of routes, if the roads that compose these routes are exclusive to each route. However an algorithm should be designed that finds the optimal number of routes the ADR should use.

Throughout this paper, it was stated that the ADR algorithm as it is can only control optimally routes going from the same origin (or origins if they are close together) to the same destinations. The ADR can control traffic from all origins to all destinations, however, this has to be done sequentially from each single origin to each single destination, which will frequently result in suboptimal solutions. This can be overcome by optimizing for each possible sequence of origin/destination pairs, but then the computational times would become infeasible for a real world application. In the next section future research on multiple origins and destinations, as well as other relevant topics, will be discussed.

7.2 Future Research

After the design and testing of the ADR algorithm, three important research questions were born.

The first one is how to change the algorithm so that multiple origins and destinations can be optimized at the same time without getting suboptimal results. A solution that does not guarantee a optimal solution, but can improve results would be to have the origin/destination pair with more demand be optimized first, and do this sequentially until arriving at the origin/destination pair with less demand in the network. This would ensure higher stability, since the bigger inflow that can originate traffic jams, would be dealt with first.

The second question, and the most obvious one, is how ADR can be changed to cope with urban traffic. In urban traffic there are several dynamics that are disregarded by the algorithm such as roundabouts, intersections, traffic lights, and smaller and more varied routes. As far as intersections and roundabouts go, these can always be translated into a travel time cost, and have even more extreme penalties in case of a jam on those areas, since they will jam multiple routes. Another problem with urban traffic is that due to the short roads, the number of possible routes in a small network such as the one in Singapore increases dramatically, making it computationally infeasible if it is treated as a centralized problem.

Finally, the last question is if ADR can be used in future search for other types of problems. Theoretically the ADR algorithm can be applied in different application domains, where an optimum in a multi-agent system needs to be found with respect to system costs and individual costs, such as game theory.

The most important step however in future research must be to apply and test ADR with well tuned parameters, on more accurate and realistic microscopic simulators, and add support for highway features, such as on ramps, off ramps, interchanges. After that step is taken, the algorithm will be a valuable tool for application in traffic control processes.

References

Alves, D. (2008). Swarm intelligence for traffic management. *Literature Survey, DCSC-TU Delft.*

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

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328.

Chudak, F. and Eleuterio, V. D. S. (2006). The traffic equilibrium problem. *IFOR Miteilungen.*

D'Acierno, L., Montella, B., and Lucia, F. D. (2006). A stochastic traffic assignment algorithm based on ant colony optimization. In *Lecture Notes in Computer Science*, pages 25–36.

Dong, J. and Wu, J. (2003). Urban traffic networks equilibrium status recognition with neural network. *Proceedings Intelligent Transportation Systems, 2003 IEEE*, 2:1049–1053 vol.2.

Dorigo, M. and Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, pages 29–41.

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

Hong, W., Tian, Y., and Xu, Y. (2007). The research of dynamic path planning for centralized vehicle navigation. *Automation and Logistics, 2007 IEEE International Conference on*, pages 1198–1202.

Knight, F. H. (1924). Some fallacies in the interpretation of social cost. *The Quarterly Journal of Economics*, 38:582–606.

Matos, A. C. and Oliveira, R. C. (2004). An experimental study of the ant colony system for the period vehicle routing problem. *ANTS 2004*, pages 286–293.

Messmer, A. and Papageorgiou, M. (1990). Metanet: A macroscopic simulation program for motorway networks. *Traffic Engng. and Control*, 31:466–470.

Payne, H. (1971). Models of freeway traffic and control. *Simulation Counceling Proceedings*, 1:55–61.

Pigou, A. C. (1920). *Economics of Welfare*. Macmillan and Co.

Prabhas (2007). Singapore highway. `http://lh4.ggpht.com/_qcJJV2d-XNY/R7_U9Tg_6YI/AAAAAAAAAhY/UwCxrFBpB8U/Singapore+Highway.jpg`.

Rodriguez-Perez, M., Herreria-Alonso, S., Fernandez-Veiga, M., Suarez-Gonzalez, A., and Lopez-Garcia, C. (2008). Achieving fair network equilibria with delay-based congestion control algorithms. *Communications Letters, IEEE*, 12(7):535–537.

Tatomir, B. and Rothkrantz, L. (2006). Hierarchical routing in traffic using swarm-intelligence. *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 230–235.

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

Wiki (2005). Erp gantry at north bridge road, next to parco bugis junction. Picture Wikipedia mailerdiablo. `http://upload.wikimedia.org/wikipedia/commons/7/7a/ERPBugis.JPG`.

Xu, Z., Sun, H., Li, X., Chen, D., and Yu, S. (2008). Ant colony optimization arithmetic of capacity restraint traffic assignment. *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 972–976.

Yong, A. H. Y. (2007). Map of the expressways and semi-expressways of singapore. Picture Wikipedia. `http://upload.wikimedia.org/wikipedia/commons/2/22/Expressways_and_semi-expressways_of_Singapore.png`.

Zhang, J., Xu, B., Cai, L., and Zheng, X. (2007). Traffic network equilibrium simulation based on multi-agent. *Automation and Logistics, 2007 IEEE International Conference on*, pages 2437–2440.

A Singapore Expressway Network

For the Singapore case study 6 different expressways in the east area of the island will be used, divided in stretches at each interconnection. In Table 5 the parameters of each stretch of expressway are presented including the respective name, length and number of lanes. Each expressway has two IDs to account for all expressways having traffic both ways.

Table 5: Parameters of each expressway in Singapore as modelled in the simulator.

n∘	**Length** (km)	**Lanes**	**Start Node**	**End Node**	**Name**
1, 2	3	3	9	10	AYE - Ayer Rajah Expressway S1
3, 4	4.5	4	10	11	ECP - East Coast Parkway S1
5, 6	13	4	11	4	ECP - East Coast Parkway S2
7, 8	2	4	3	4	PIE - Pan Island Expressway S4
9, 10	1	3	8	11	KPE - Kallang-Paya Lebar Expressway S1
11, 12	2	3	6	8	KPE - Kallang-Paya Lebar Expressway S2
13, 14	8	3	2	6	KPE - Kallang-Paya Lebar Expressway S3
15, 16	6.5	3	2	3	TPE - Tampines Expressway S1
17, 18	2	4	6	5	PIE - Pan Island Expressway S2
19, 20	7	3	1	2	TPE - Tampines Expressway S2
21, 22	7.5	2	1	5	CTE - Central Expressway S3
23, 24	3.5	2	5	7	CTE - Central Expressway S2
25, 26	4.5	2	7	9	CTE - Central Expressway S1
27, 28	11	4	6	3	PIE - Pan Island Expressway S3
29, 30	1	4	o_4	4	ECP - East Coast Parkway S3
31, 32	3	3	o_1	9	AYE - Ayer Rajah Expressway
33, 34	3	4	o_2	5	PIE - Pan Island Expressway
35, 36	3	3	o_3	1	SLE - Seletar Expressway

In Figure 19 the complete network for reference is shown, including road IDs, origins and destinations. Note that this is a simplified network, and that means that off ramps and on ramps have been disregarded due to the ADR being applicable to only highways.

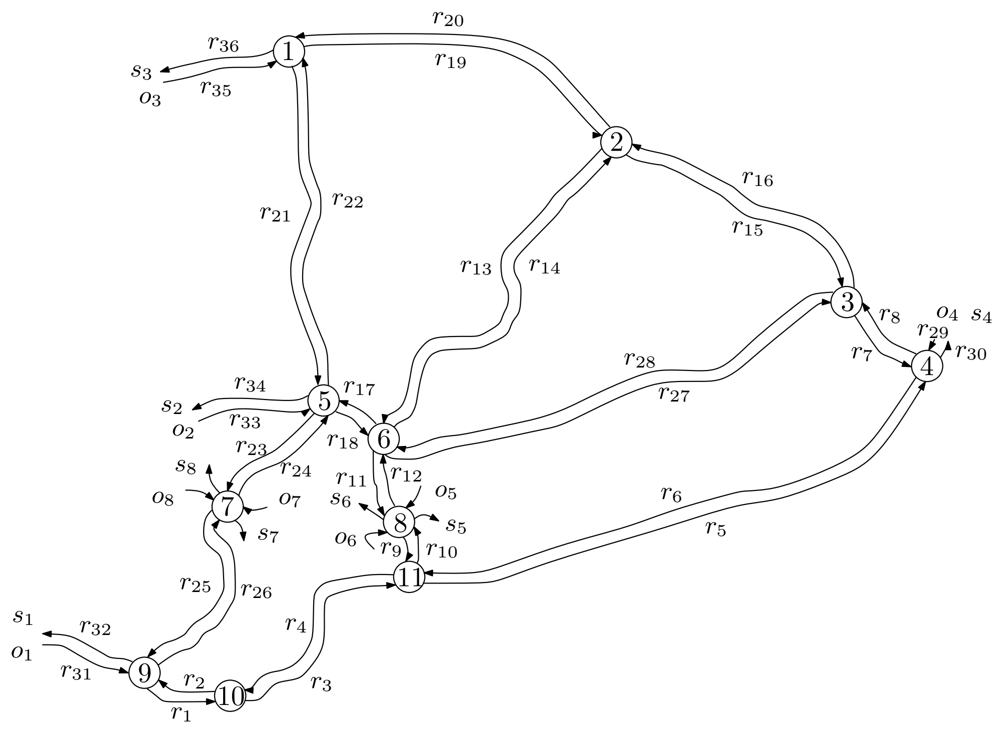


Figure 19: Network used by simulator and ADR.

B Weight Tuning

Table 6 shows the results of the algorithm for two different works. One is similar to the one presented in Section 4.5, with two different roads to choose from with different lengths. The second network is similar to the first one, but it has an extra third optional road. Both networks are tested with different number of cars requesting routes. W_1 is always fixed at 1. The column C represents Condition C, and in case it is checked it means the condition has been accomplished.

The $\frac{\gamma_{\max}}{\gamma_{\min}}(\%)$ column represents Condition B. Smaller values mean that the travel time cost of the most costly and least costly routes used are similar.

Table 6: Networks 1 and 2 results for varying weights W_2 and number of cars n_c.

Network 1				
	$n_c = 200$		$n_c = 1000$	
W_2	$\frac{\gamma_{\max}}{\gamma_{\min}}(\%)$	C	$\frac{\gamma_{\max}}{\gamma_{\min}}(\%)$	C
-0.7	8.1 ± 0.08	$\surd$	0.6 ± 0.03	$\surd$
-0.8	5.95 ± 0.01	$\surd$	0.37 ± 0.02	$\surd$
-0.9	3.31 ± 0.01	$\surd$	0.29 ± 0.09	$\surd$
Network 2				
	$n_c = 200$		$n_c = 1000$	
W_2	$\frac{\gamma_{\max}}{\gamma_{\min}}(\%)$	C	$\frac{\gamma_{\max}}{\gamma_{\min}}(\%)$	C
-0.7	15.31 ± 0.09	$\surd$	4.64 ± 0.12	$\surd$
-0.8	10.88 ± 0.01	$\surd$	3.21 ± 0.05	$\surd$
-0.9	5.87 ± 0.02	$\surd$	1.97 ± 0.37	$\surd$

C Travel Time Cost Parameter Tuning

The parameters of the travel time cost equation M and ϵ where tuned and tested using the network presented in Section 4.5. These parameters can be tuned depending on the objective of the user wants to accomplish with the ADR. They can be tuned to make traffic jam avoidance the most important objective by having a hard constraint. They can also be tuned to have bigger accuracy at the risk that on more varying inflow conditions the algorithm might pass the critical level of density in its routing policy. The parameters used in this paper are the ones that ensure bigger stability, since in Section 2 it was stated that the most important objective it to avoid creating traffic jams.

- Increasing M provides more accuracy, but also introduces more risk of the system becoming unstable, that is, the algorithm has more abrupt changes, and though rare, can unstabilize the system, since it is always close to the critical level. A good range of values for M is $[2, 10]$: higher than that the changes at each optimization step are too abrupt, lower than that the effects of the bell function start to be unnoticeable. However the higher the inflow, the higher M needs to be, so for example, even if $M = 2$ is a safer value, it might not work for higher inflows.

- Again, increasing ϵ introduces more risk of the system unstabilizing, but is also more accurate. That is because a low ϵ represents a steeper ascent near the critical level, which acts as a barrier, while a high ϵ gives a more gradual increase of the cost. A good range of values for ϵ is $[0.1, 0.5]$.

Table 7 shows the statistics of several test runs with different parameters that reflect the behaviour that was just described. It does not include parameters where the results made highway 2 enter an unstable regime. In these trials the ADR algorithm is used every 5 minutes, so that if the critical density is approached the algorithm can counteract after 5 minutes. Let Ψ be the percentage of times the optimization algorithm gives an inflow of vehicles per hour that is higher than the maximum capacity of that road, that is, with $q_{\text{in}} = 5500$ vec/hr, when the split for highway 2 is higher than $\frac{Q_{\text{cap}}}{q_{\text{in}}} = \frac{1937}{5500} = 35.22\%$.

Table 7: Algorithm results for $q_{\text{in}} = 5500$ vec/hr.

M	ϵ	$\max \rho_2$ [vec/km/lane]	Ψ (%)
10	0.5	25.69	39.6
10	0.1	25.50	16.7
5	0.5	26.07	37.5
5	0.1	25.53	14.6
2	0.5	26.06	38.3
2	0.1	25.89	25.0