



CEUR-WS.org Vol-  
1678  
urn:nbn:de:0074-  
1678-2

Copyright © 2016 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

# ATT 2016

## Ninth International Workshop on Agents in Traffic and Transportation

Proceedings of the Ninth International Workshop on Agents in Traffic and Transportation (ATT 2016)  
co-located with the 25th International Joint Conference On Artificial Intelligence (IJCAI 2016)

New York, USA, July 10, 2016.

Edited by

**Ana Lúcia C. Bazzan** \*

**Franziska Klügl** \*\*

**Sascha Ossowski** \*\*\*

**Giuseppe Vizzari** \*\*\*\*

\* Institute of Informatics, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil

\*\* School of Science and Technology, Örebro University, Örebro, Sweden

\*\*\* Centre for Intelligent Information Technologies (CETINIA), University Rey Juan Carlos, Madrid, Spain

\*\*\*\* Complex Systems and Artificial Intelligence research center, Università degli Studi di Milano-Bicocca, Milano, Italy

# Table of Contents

## Session 1: Tolls

- **Delta-Tolling: Adaptive Tolling for Optimizing Traffic Throughput**  
*Guni Sharon, Josiah Hanna, Tarun Rambha, Michael Albert, Peter Stone, Stephen D. Boyles*
- **When are Marginal Congestion Tolls Optimal?**  
*Reshef Meir, David C. Parkes*

## Session 2: Wayfinding and route choices for pedestrians and vehicles

- **Conflicting Tendencies in Pedestrian Wayfinding Decisions: a Multi-Agent Model Encompassing Proxemics and Imitation**  
*Luca Crociani, Giuseppe Vizzari, Stefania Bandini*
- **Pedestrians' Route Choice Model for Shopping Behavior**  
*Bruno Rocha Werberich, Carlos Oliva Pretto, Helena Beatriz Bettella Cybis*
- **Using Topological Statistics to Bias and Accelerate Route Choice: Preliminary Findings in Synthetic and Real-World Road Networks**  
*Fernando Stefanello, Bruno C. da Silva, Ana L. C. Bazzan*
- **Combining Car-to-Infrastructure Communication and Multi-Agent Reinforcement Learning in Route Choice**  
*Ricardo Grunitzki, Ana L. C. Bazzan*
- **On Estimating Action Regret and Learning From It in Route Choice**  
*Gabriel de O. Ramos, Ana L. C. Bazzan*

## Session 3: Innovative mobility systems

- **An Empirical Investigation of Adaptive Traffic Control Parameters**  
*Jeffery Raphael, Elizabeth I. Sklar, Simon Maskell*
- **An Architecture for Safe Evacuation Route Recommendation in Smart Spaces**  
*Marin Lujak, Stefano Giordani, Sascha Ossowski*
- **Urban Traffic Control Assisted by AI Planning and Relational Learning**  
*Alberto Pozanco, Susana Fernández, Daniel Borrajo*
- **A Novel Approach of Cognitive Base Station with Dynamic Spectrum Management for High-Speed Rail**  
*Qingting Wu, Yiming Wang, Zhijie Yin, Hongyu Deng, Cheng Wu*
- **Iterative Committee Elections for Collective Decision-Making in a Ride-Sharing Application**  
*Sophie L. Dennisen, Jörg P. Müller*
- **Model-Driven Engineering of Simulations for Smart Roads**  
*Alberto Fernández-Isabel, Rubén Fuentes-Fernández*

# Delta-Tolling: Adaptive Tolling for Optimizing Traffic Throughput

Guni Sharon<sup>1</sup>, Josiah Hanna<sup>1</sup>, Tarun Rambha<sup>2</sup>, Michael Albert<sup>1</sup>, Peter Stone<sup>1</sup>, Stephen D. Boyles<sup>2</sup>

<sup>1</sup>Department of Computer Science, <sup>2</sup>Department of Civil Engineering  
The University of Texas at Austin,  
Austin, TX 78712 USA

## Abstract

In recent years, the automotive industry has been rapidly advancing toward connected vehicles with higher degrees of autonomous capabilities. This trend opens up many new possibilities for AI-based efficient traffic management. This paper investigates traffic optimization through the setting and broadcasting of dynamic and adaptive tolls under the assumption that the cars will be able to continually reoptimize their paths as tolls change.

Previous work has studied tolling policies that result in optimal traffic flow and several traffic models were developed to compute such tolls. Unfortunately, applying these models in practice is infeasible due to the dynamically changing nature of typical traffic networks. Moreover, this paper shows that previously developed tolling models that were proven to yield optimal flow in theory may not be optimal in real-life simulation. Next, this paper introduces an efficient tolling scheme, denoted  $\Delta$ -tolling, for setting dynamic and adaptive tolls. We evaluate the performance of  $\Delta$ -tolling using a traffic micro-simulator.  $\Delta$ -tolling is shown to reduce average travel time by up to 35% over using no tolls and by up to 17% when compared to the current state-of-the-art tolling scheme.

## 1 Introduction

In recent years, communication and computation capabilities have become increasingly common onboard vehicles. Such capabilities present opportunities for developing safer, cleaner and more efficient road networks. This paper combines knowledge from mechanism design, game theory, network flow optimization, and multi-agent simulations for investigating responsive pricing, as a scheme for managing and optimizing traffic flow.

It has been known for nearly a century that drivers seeking to minimize their private travel times need not minimize the total level of congestion. In other words, self-interested drivers may reach an equilibrium that is not optimal from a system perspective. On the other hand, charging each agent with an amount equivalent to the damage it inflicts on all

other agents (also known as the *marginal cost*) results in optimal flow [Pigou, 1920; Beckmann *et al.*, 1956; Braess, 1969]. The damage inflicted by a given agent is evaluated through the marginal slowdown caused by it and is commonly evaluated using stylized traffic models. Such stylized models take a “macroscopic” view of traffic, where delay can be expressed as a smooth function of travel demand. We hereafter refer to such models as macro-models. The marginal slowdown, evaluated by such models, is then used to infer appropriate tolls. However these macro-models make many approximations and assumptions that don’t hold in practice.

Modern simulation tools and computational power allow for much more fine-grained simulation of traffic networks, referred to as micro-simulation models. Using such a realistic traffic simulator we demonstrate the potential of using tolls for reducing average travel time and increasing average utility. In this paper we show (empirically) that computing tolls using a macro-model does not lead to optimal performance in a realistic simulator. We explain this effect by noting that macro-models assume deterministic conditions, and have a number of unrealistic features. In recent years, researchers have relaxed the assumptions of the first macroscopic tolling models to incorporate responsiveness to roadway disruptions such as accidents [de Palma and Lindsey, 1998; Yang, 1999a,b; Lindsey, 2009; Boyles *et al.*, 2010] and to the total level of travel demand [Nagae and Akamatsu, 2006; Chen and Subprasom, 2007; Gardner *et al.*, 2008, 2011]. However, the effectiveness of all of these models is still restricted by the use of simplifying assumptions such as constant and known demand and capacity for each link.

In response to the suboptimal performance of existing macro-models, this paper introduces a novel tolling scheme denoted  $\Delta$ -tolling.  $\Delta$ -tolling approximates the marginal cost of each link using only two variables (current travel time and free flow travel time) and one parameter. Due to its simplicity,  $\Delta$ -tolling is fast to compute, adaptive to current traffic, and accurate. We prove that, under some assumptions,  $\Delta$ -tolling results in tolls that are equivalent to the marginal cost and demonstrate that it can lead to near-optimal performance in practice.

## 2 Motivation

This section defines the notion of *user equilibrium* (UE) and *system optimum* (SO). Applying tolls is then introduced as a

mechanism that allows UE and SO to coincide. The *marginal cost toll* (MCT) policy is then presented followed by some macroscopic traffic models that approximate it. We discuss some of the drawbacks of such macro-models, which provides the motivation for the current study.

## 2.1 Computing User Equilibrium

Consider a directed network  $G = (V, E)$ , where  $V$  and  $E$  are the set of nodes and links respectively. Suppose that the demand (flow rates) between every pair of nodes is known. In this paper we assume that the travel time on a link  $e \in E$  is a function of its flow ( $x_e$ ) and is represented using a non-decreasing function  $t_e(x_e)$  (also called volume delay or link-performance functions). In practice, the Bureau of Public Roads (BPR) function  $t_e(x_e) = T_e(1 + \alpha(\frac{x_e}{C_e})^\beta)$  is commonly used as the delay function, where  $T_e$  is the free flow travel time and  $C_e$  is the capacity of link  $e$ .  $\alpha$  and  $\beta$  are parameters whose default values are 0.15 and 4 respectively.

When agents choose routes selfishly, a state of equilibrium, called user equilibrium (UE) [Wardrop, 1952], is reached in which all used routes between an origin-destination (OD) pair have equal and minimal travel time. The link flow rates corresponding to this state can be obtained by solving a non-linear convex program that minimizes the Beckmann potential function ( $\sum_{e \in E} \int_0^{x_e} t_e(x_e) dx$ ) Beckmann *et al.* [1956]. This objective ensures that the KKT (Karush-Kuhn-Tucker) conditions [Kuhn and Tucker, 1951; Karush, 1939] of the convex program correspond to Wardrop’s UE principle [Wardrop, 1952]. The constraints of the optimization problem include non-negativity and flow conservation constraints. This model, also known as the traffic assignment problem (see Patriksson [1994] for a thorough overview), has been widely studied because of the mathematically appealing properties associated with convex programming.

## 2.2 Computing System Optimum

The system optimal (SO) problem can be formulated using a set of constraints similar to those used for computing UE but replacing the objective function with  $\sum_{e \in E} x_e t_e(x_e)$ . As mentioned before, all agents do not experience equal and minimal travel times at the SO state which incentivizes agents to switch routes. Instead, if an optimal tolling policy is applied, the flows resulting from a UE assignment in which agents minimize the generalized cost (time + toll) coincides with the SO solution. MCT is proven to be such a policy (UE=SO) [Pigou, 1920; Beckmann *et al.*, 1956; Braess, 1969]. In MCT each agent is charged a toll that is equal to the increase in travel time it inflicts on all other agents. Unfortunately, knowing in advance the marginal impact of an agent on traffic is infeasible in practice.

## 2.3 Approximating Marginal-Cost Tolls

The focus of this paper is methods that approximate the marginal cost. Most of these methods assume that demand on each link is constant. In such cases MCT can be formally defined as follows: given a link ( $e$ ) and flow ( $x_e$ ) the toll applied to  $e$  equals the change in travel time caused by an infinitesimal flow ( $\frac{dt_e(x_e)}{dx_e}$ ) multiplied by the number of agents currently on this link ( $x_e$ ).

A number of researchers have attempted to develop macro-models that approximate MCT for a given system [Yang *et al.*, 2004; Han and Yang, 2009]. However, a major drawback of such macro-models is that they are static and do not capture the time-varying nature of traffic. They also assume that the delay on each link is a function of its flow and hence neglect effects of intersections and traffic shocks. Although there has been some research on congestion pricing using finer traffic flow models, most of the existing models either assume complete knowledge of demand distribution over time [Wie and Tobin, 1998; Joksimovic *et al.*, 2005] or are restricted to finding tolls on freeways in which travelers choose only between parallel tolled and free general-purpose lanes [Gardner *et al.*, 2013, 2015; Yin and Lou, 2009]. This limitation motivates us to employ a simulation framework to replicate traffic in a more realistic manner, evaluate the performance of existing macro-models, and develop new methods to determine optimal tolls while adapting to unknown and changing demand.

## 3 Simulation

In order to evaluate the effectiveness of different tolling models on traffic flow optimization, we used a modified version of the *Autonomous Intersection Manager* (AIM) micro-simulator [Dresner and Stone, 2008]. On the one hand, AIM is very realistic in the sense that it allows simulating accelerations of individual vehicles in response to traffic conditions. On the other hand, due to computational limitations, AIM cannot scale to large road networks (only up to  $3 \times 3$  grid network). For our experiments AIM was chosen since, unlike other simulators, it allows non deterministic traffic behavior, provides (direct) measurements on vehicle following distances, lane changes, gap acceptance, etc.

### 3.1 Autonomous Intersection Manager Simulator

AIM provides a multiagent framework for simulating autonomous vehicles on a road network grid; it presents a realistic traffic flow model that allows experimenting with adaptive tolling. The AIM simulator uses two types of agents: intersection managers, one per intersection, and driver agents, one per vehicle. Intersection managers are responsible for directing the vehicles through the intersections, while the driver agents are responsible for controlling the vehicles to which they are assigned. To improve the throughput and efficiency of the system, the driver agents “call ahead” to the intersection manager and request a path reservation (space-time sequence) within the intersection. The intersection manager then determines whether or not this request can be met. If the intersection manager approves a driver agent’s request, the driver agent must follow the assigned path through the intersection. On the other hand, if the intersection manager rejects a driver agent’s request, the driver agent may not pass through the intersection but may attempt to request a new reservation.

At every intersection, the driver agent navigator runs an  $A^*$  search [Hart *et al.*, 1968] to determine the shortest path leading to the destination of the vehicle associated with it. The navigator then directs the driver agent to drive via the shortest route. This behavior ensures that each vehicle acts greedily with respect to minimizing travel time. Next, we describe the



required enhancements to the standard AIM simulator [Dresner and Stone, 2008] necessary to simulate realistic tolling experiments.

### 3.2 Enhancements to the AIM Simulator

In order to evaluate adaptive-tolling using AIM the following modifications were required:

- **Link toll:** each link ( $e$ ) in the road network is associated with a toll,  $toll_e$ , which can adapt in real-time according to traffic conditions.
- **Link travel time:** each link stores: (1) an estimated travel time,  $t_e$ , that is based on real-time observed flow speed, and (2) an estimated free flow travel time  $T_e$ , that is based on the link's length divided by its speed limit.
- **Route selection:** when a car has several routes leading to its destination, the driver agent chooses the route ( $r = e_1, e_2, \dots, e_3$ ) that minimizes  $\sum_{e \in r} t_e \times VOT + toll_e$ , where  $VOT$  is the monetary *value Of time*.
- **Value Of Time:** each driver agent is associated with a randomly generated  $VOT$  that is drawn from a normal distribution. We assume monetary units are chosen such that the mean value is 1¢ per second, and assume a standard deviation of 0.2.  $VOT$  represents the value (in cents) of one second for the driver. A driver with  $VOT = x$  is willing to pay up to  $x$ ¢ in order to reduce travel time by 1 second.

### 3.3 Macroscopic Model

This paper uses a macroscopic model to approximate MCT. This model is used to solve the convex program described in Section 2 using Algorithm B [Dial, 2006]. Algorithm B is a bush-based/origin-based algorithm which exploits the fact that at equilibrium, all used routes carrying demand from a particular origin must belong to an acyclic subgraph in which each destination can be reached from the origin (such a subgraph is also called a bush). At each iteration, the algorithm maintains a collection of bushes (one for each origin), shifts agents within a bush to minimize their generalized costs, and adds/removes links in a bush until equilibrium is reached. Closeness to equilibrium is measured using *average excess cost*, which represents the average of the difference between each agent's generalized cost and the least cost path at the current flow solution. In the experiments presented in this paper, the algorithm was terminated when the average excess cost of the flow solution dropped below  $1E-13$ .

## 4 Empirical Evaluation: Macroscopic Model

One of the main contributions of this paper is an empirical demonstration that setting tolls based on macro-models can lead to suboptimal results when evaluated in a more realistic micro-simulator. This section presents these empirical results, which motivate our new tolling scheme as presented in the next section.

### 4.1 Exemplar Road-Network

Figure 1 illustrates an exemplar road network that demonstrate the impact of tolls that adapt to traffic demand. The

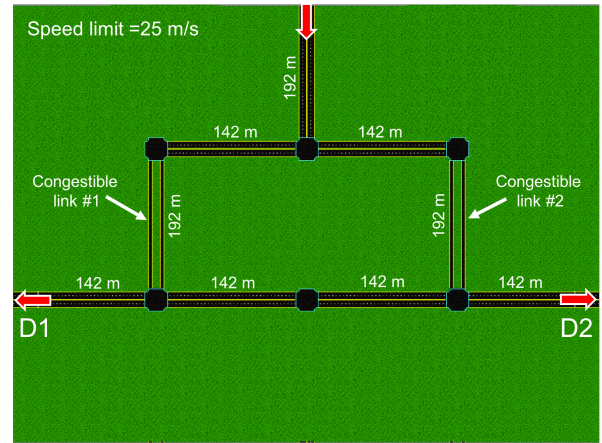


Figure 1: Exemplar road network within the AIM simulator.

speed limit across all roads is 25 meters per second. Each horizontal road is 142 meters long, and each vertical road is 192 meters long. We examined a scenario in which agents enter the network from a single source, the top road (incoming arrow), and leave the network from one of two destinations (outgoing arrows) D1 or D2. All roads are composed of two lanes per direction and assumed to have infinite capacity<sup>1</sup> except the two vertical roads in the middle of the network (Congestible link #1 and #2), which possess only one lane (capacity = 1,908 agents per hour). An agent entering the system and heading towards D1 (or symmetrically D2) has two possible routes to choose from: a short route (668 m) or a long route (964 m). Each agent chooses one of the two routes according to the distance, traffic conditions, and tolls associated with it. This road network represents a special case where if most agents are heading to D1 (or symmetrically D2) then link #1 (#2) should be tolled while link #2 (#1) should not. We define  $z$  (or symmetrically  $1 - z$ ) to be the proportion of agents heading to D1 (D2). The incoming traffic rate was set to 2,160 agents per lane per hour.

### 4.2 Computing the Optimal Tolls

First, we computed, in a brute-force manner, the toll values that optimize average travel time for each  $z \in \{0.0, 0.1, 0.2, \dots, 1\}$ . We considered tolling only congestible link #2. Tolling uncongestible links is unnecessary as there is no congestion externality associated with travel on these links. Moreover, there is no reason to toll both congestible links simultaneously (#1 and #2) since any of the two possible routes (leading from source to  $D_i$ ) includes exactly one of these links. A negative toll value for link #2 is symmetrical to a positive toll on link #1. We distinguish between the optimal *adaptive toll* and the optimal *static toll*. The optimal adaptive toll is the toll value that minimizes travel time for a given  $z$  value. The optimal static toll is the toll value that minimizes travel time over all  $z$  values (assuming equal weighting of the  $z$  values, i.e., all  $z$  values have the same probability), found to

<sup>1</sup>The capacity on roads with two lanes is higher than the rate in which agents are spawned. Hence, we consider such roads as having infinite capacity.

$z$	Toll Values		AVG Travel Time (seconds)				
	Optimal	Macro Model	No Tolls	Static Tolls	Optimal Tolls	Macro Tolls	$\Delta$ -Tolls
0.0	15	14.8	46.0	47.6	40.9*	<b>40.3*</b>	40.5*
0.1	10	14.8	43.2	45.1	39.1*	39.3*	<b>39.0*</b>
0.2	10	14.8	38.4	40.8	<b>35.8*</b>	38.4	36.9*
0.3	10	14.8	34.3	35.1	33.8	37.7	<b>33.1*</b>
0.4	0	14.8	31.7	32.4	31.7	36.8	<b>31.4</b>
0.5	5	-5.3	<b>30.8</b>	31.2	<b>30.8</b>	30.9	30.9
0.6	5	-14.8	<b>31.1</b>	31.5	<b>31.1</b>	34.7	31.6
0.7	-5	-14.8	<b>32.2</b>	32.2	<b>32.2</b>	35.2	32.8
0.8	-10	-14.8	37.0	<b>34.1*</b>	<b>34.1*</b>	36.2	35.8*
0.9	-10	-14.8	40.7	<b>36.2*</b>	<b>36.2*</b>	36.8*	36.5*
1.0	-15	-14.8	43.1	39.0*	38.5*	<b>38.1*</b>	38.7*

Table 1: The left side of the table shows the empirical optimal and macro-model predicted toll values (imposed on link #2) for different  $z$  values. The right side shows average travel times when no tolls, static tolls, optimal tolls, macro-model tolls and  $\Delta$ -tolls are applied as calculated by the AIM simulator. \* indicates statistical significance over no tolls (using unpaired  $t$ -test with  $p_{value} = 0.05$ ).

be  $-10$  in this example. While it might seem like the optimal static toll should be zero, asymmetries in the model arising from differences between left and right turns affect junction delays and skew the optimal static toll to one side.

Optimal adaptive tolls for each  $z$  value are presented in Table 1. Notice that as the  $z$  value increases, the optimal toll steadily decreases. Intuitively, when all agents go to one destination ( $z = 0$  or  $z = 1$ ) we need more of them to choose the longer route to achieve the optimal system flow, thus requiring a more extreme toll. When  $z \approx 0.5$ , a zero toll is optimal since agents that choose their longer route will only make congestion worse for agents going to the other destination. As a result, enforcing tolls for  $0.2 < z < 0.8$  did not result in a significant improvement over enforcing no tolls. The reason that Table 1 present values different than zero for that range stems from noise and asymmetries in the model.

### 4.3 Evaluating Optimal Tolls Using a Macro-Model

We compared the empirically optimal tolls against the toll values predicted by the macro-model. Toll values calculated by the macro-model are also presented in Table 1. Table 1 also presents average travel time under different tolling policies (for now ignore the  $\Delta$ -tolls column). Though the macro-model obtains near optimal results for the extreme  $z$  values and  $z = 0.5$ , it is sub-optimal for intermediate values. One explanation for this phenomenon is that the stylized congestion models assume that delays on a link are a function solely of flow on that link, ignoring interactions between links at intersections. For the extreme  $z$  values this assumption is more reasonable because almost all agents on congestible links are heading in the same direction. However for the intermediate values (excluding 0.5) the agents on the congestible links encounter traffic on the bottom horizontal link (by cars taking the longer route) causing changes in the capacity of the congestible links that cannot be captured by a stylized model. These results lead us to the following conclusions:

1. Toll can reduce average travel time by up to 11% compared to applying no tolls (see  $z = 0$ ).

2. Static tolls might have a negative effect in some cases (see  $z < 0.6$ ).
3. The macro-model fails to achieve system optimal in some cases reaching up to 10% suboptimality (see  $z = 0.3$ ).

Both static and adaptive macro-model based tolls (a) result in suboptimal performance and (b) require that the demand over all OD pairs is known and fixed. As a result, neither is applicable to real-world traffic. There is thus, a need for a new tolling scheme that is dynamic, adaptive, and results in near-optimal traffic flow.

## 5 Delta-tolling

This section introduces the main technical contribution of the paper, a new tolling scheme denoted  $\Delta$ -tolling. Unlike macroscopic models,  $\Delta$ -tolling is adaptive to unknown and changing link capacities and demands. We first define  $\Delta$ -tolling and then prove, under mild assumptions, that it is equivalent to MCT.

$\Delta$ -tolling is defined over a directed network  $G = (V, E)$  (a road network for example) with a set of current flow values (traffic volume for example). The output of  $\Delta$ -tolling is a set of toll values, one toll value per link. We use  $t_e$  to denote the current flow time on link  $e \in E$ . Recall that  $T_e$  denotes the free flow travel time and  $toll_e$  to denote the toll value assigned to link  $e$ . For each link ( $e$ ),  $\Delta$ -tolling assigns a toll equivalent to the difference between the current flow time ( $t_e$ ) and the free flow time ( $T_e$ ) multiplied by a parameter ( $\beta$ ). More formally:  $\Delta-toll_e = \beta(t_e - T_e)$ . As a rule of thumb,  $\beta$  should be correlated to the mean VOT. High  $\beta$  values result in high toll values which are needed to influence agents with high VOT. Calculating  $\Delta-toll_e$  requires a constant amount of time. As a result, the complexity of computing tolls for an entire network is  $\Theta(E)$ .

Next we prove that  $\Delta$ -tolling is equivalent to MCT under some conditions. This is a desirable property, since MCT results in system optimal (see Section 2). First, we list the assumptions under which the above statement holds:

	Macro-model	$\Delta$ -tolling
Parameters	Required	
$\alpha$	yes	<b>no</b>
$\beta$	yes	yes
Variables	Required	
Demands	yes	<b>no</b>
$C_e$	yes	<b>no</b>
$T_e$	yes	yes
$t_e$	<b>no</b>	yes
Properties	Satisfied	
Adaptive	no	<b>yes</b>
MCT	<b>yes</b>	<b>yes</b>

Table 2: The different parameters, variables and properties of  $\Delta$ -tolling and macro-model tolling. MCT refers to approximating the marginal cost.

1. The delay on each link is expressed by the BPR volume delay function,  $t_e(x_e) = T_e(1 + \alpha(\frac{x_e}{C_e})^\beta)$ .
2. Changes in traffic flow are negligible between the time an agent plans its route and the time it traverses the network.

**Lemma 1** *Under the above assumptions, the tolls computed by  $\Delta$ -tolling are equivalent to the MCT.*

**Proof:** We express the BPR volume delay function as:

(1)  $t_e(x_e) = T_e + ax_e^\beta$  where  $a = T_e \frac{\alpha}{C_e^\beta}$ . MCT is defined as the derivative of the delay function ( $\frac{dt_e(x_e)}{dx_e}$ ) multiplied by the flow ( $x_e$ ). Calculating MCT requires knowing the future flow but under Assumption 2 we can use current flow instead. So we get:

$$(2) MCT_e = x_e \frac{dt_e(x_e)}{dx_e} = x_e(\beta ax_e^{\beta-1}) = \beta ax_e^\beta = \beta(T_e + ax_e^\beta - T_e).$$

Combining (1) and (2) we get:

$$MCT_e = \beta(t_e - T_e) = \Delta\text{-toll}_e. \square$$

The main theoretical differences between  $\Delta$ -tolling and macroscopic models are summarized in Table 2. In the next section we study the differences in performance using the adapted AIM simulator.

Although the assumptions made in this section might not hold in all possible traffic networks, we provide experimental results showing that in realistic simulations,  $\Delta$ -tolling improves traffic flow and may achieve near optimal flow.

## 6 Empirical Evaluation: Delta-Tolling

This section analyzes the performance of  $\Delta$ -tolling on a representative road network. We then generalize our findings and show they also hold for randomly generated networks. We begin by comparing the system performance when using  $\Delta$ -tolling on the exemplar road network (presented in Figure 1). Table 1 also presents the average travel time for  $\Delta$ -tolling. Unlike the macro-model,  $\Delta$ -tolling achieves performance that is similar to the optimal. We do not report the toll values for  $\Delta$ -tolling as they are dynamically changing across the simulation.

Next, we present results for larger networks. In such networks finding the optimal tolls in a brute force manner is

infeasible.<sup>2</sup> For the following experiments we used grid networks of size  $3 \times 3$  that include 9 intersection (see Figure 3 for an example). Agents enter at the same rate of 300 agents per hour from any incoming lane (a road with three lanes, for example, spawns 900 agents per hour). Each agent entering the system is assigned one of two possible exit roads with equal probability (0.5). Each agent is also assigned two alternative exits. Exiting via an alternative exit imposes a predefined, randomly generated, delay.<sup>3</sup> We justify allowing alternative exits as follows, in many real-life scenarios, several routes, usually of different length, may lead an agent to its destination. For example, a driver exiting *Manhattan* and heading to *Queens* will prefer to exit via *Queens Midtown Tunnel*, it can suffer some delay and exit from *Ed Koch Queensboro Bridge* or suffer a severe delay while exiting via *Williamsburg Bridge*. Following this logic, we view the simulated network as part of a larger road network in which agents may use paths outside of the network to reach their final destination.

Some roads in the simulated network are more congestible than others i.e., the number of lanes varies. The number of lanes for each road was randomly picked from  $[1, 4]$ . We ran the simulator for 5000 seconds for each reported setting.<sup>4</sup> In the following experiments we used an upper bound on toll values equal to  $25\text{¢}$ .<sup>5</sup> The upper bound is set for two reasons: (1) avoiding overcharging in links with temporary heavy congestion (2) avoiding oscillation in congestion caused by overpricing: heavy congestion may cause a steep increase to the toll value which later leads to the link being vacated which leads the toll value to reduce to zero. Zero toll value results, again, in heavy congestion. Applying no cap on toll values resulted in up to 5% reduced utility. We report three different measurements:

- **Time** - the average travel time.
- **Utility** - the average utility (in cents). Where utility is defined for each agent as its travel time multiplied by its VOT plus the summation of tolls paid by it.
- **Standardized Utility (SU)** - toll revenue may be redistributed back to the drivers in the form of road improvements or tax reductions. We define *refund* as the sum of collected tolls divided by the number of agents that exited the system. SU is defined as average utility minus refund.

### 6.1 Representative Road Network

The purpose of our first experiment is to determine how different  $\beta$  values affect system performance. For this experiment we used a single randomly generated instance of a  $3 \times 3$

<sup>2</sup>Examining different combinations of toll assignment to all links in the system leads to an exponential blowup.

<sup>3</sup>When each agent is assigned only one possible exit, distributing traffic becomes impossible in many cases. For such scenarios, imposing tolls did not have a positive effect in our experiments.

<sup>4</sup>When running the simulator, in order to allow the system to balance, we exclude data from the first 500 seconds.

<sup>5</sup>The output from the macro-model contained no toll greater than  $25\text{¢}$ . Hence we deduced that greater tolls won't have a positive affect and we set the cap accordingly.

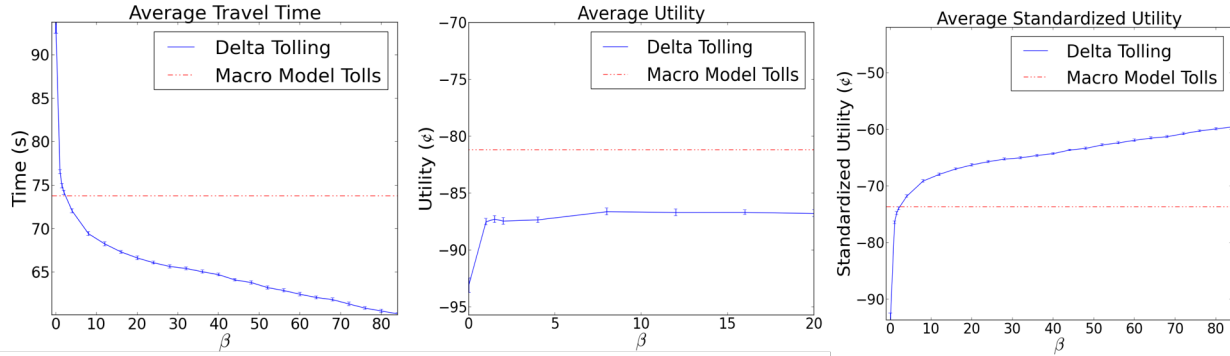


Figure 2: Average travel time, utility and standardized utility as a function of  $\beta$  for the representative road network.

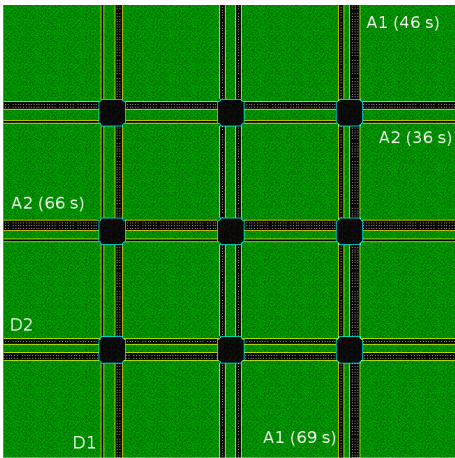


Figure 3: A representative road network. Each agent is assigned one of to destinations (**D1**, **D2**). **A1** and **A2** denote alternative destinations for **D1** and **D2** respectively. The time penalty associated with each alternative destination is given in parenthesis.

road network - depicted in Figure 3. Average travel time, Utility and SU for different  $\beta$  values are presented in Figure 2. Notice that  $\beta = 0$  represent the case where no tolls are used.

Setting  $\beta = 80$  gives an improvement of 35% in average travel time over no tolls.  $\beta = 80$  also gives an improvement of 35.01% for SU over no tolls.  $\beta$  values greater than 80 result in average travel times that are not significantly worse or better. Increasing  $\beta$  (up to 80) results in higher toll values which better distribute congestion. However, higher tolls also negatively impact utility as drivers are forced to pay more. Utility is maximized with  $\beta = 8$  which gives a 6.96% improvement over no tolls. We also report performance when tolls as computed by the macro-model are used, given as a dashed (red) line across the result graphs.  $\Delta$ -tolling outperforms macro-model tolling for  $\beta \geq 4$  by up to 18% in both average travel time and SU. On the other hand, macro-model tolling exceeds by 6.25% when utility is considered. The main reason for the macro-model’s advantage w.r.t utility is that  $\Delta$ -tolling imposes higher toll values.  $\Delta$ -tolling (with  $\beta = 8$ ) collected a total of \$1,921 while macro-model tolling col-

$\beta$	Time	Utility	SU
<b>0.0</b>	<b>69.9</b>	<b>-70.0</b>	<b>-70.0</b>
8.0	51.4*	-63.5	-51.1*
20.0	50.3*	-67.0	-49.8*
80.0	49.5*	-76.6	-48.8*

Table 3: Average travel time, utility, and SU for  $\beta$  values 8, 20, 80. These  $\beta$  values represent a trade-off between the three metrics. \*Denotes a statistically significant improvement over no tolling (using a paired  $t$ -test with  $p_{value} = 0.05$ ).

lected only \$825. Unfortunately, we observed that higher tolls are required to better distribute congestion and optimize system performance. On the other hand, we believe that standard utility is a more relevant measurement for performance comparison between the models. In real road networks tolls are most often used to fund road maintenance, effectively redistributing the money collected back to the public. When SU is considered, delta tolling significantly outperforms the macro-model in all but very low  $\beta$ . Moreover, macro-model tolling relies on static traffic conditions and so, unlike  $\Delta$ -tolling it is not applicable to real-life, dynamic road networks.

## 6.2 General Case

In order to validate that the results obtained from a single randomized instance are representative, we reran the same experiment using 50 different randomized road networks. Each of these networks is a  $3 \times 3$  grid, similar to the representative road network, but the exit roads, alternative exits, alternative exits’ delay, and number of lanes per road are randomized. Table 3 shows results for three representative  $\beta$  values (8, 20, 80) compared to no tolling.  $\beta = 8$  and  $\beta = 80$  are chosen since they maximized performance with respect to utility and travel time/SU.  $\beta = 20$  represents a good balance between utility and travel time.

We observe that the advantage of  $\Delta$ -tolling is robust to changes in network topology. For the general case,  $\Delta$ -tolling achieves an improvement over no tolling of 29.2%, 9.31% and 30.28% in Time, Utility and SU respectively.

## 7 Conclusions

This paper considers applying tolls to road networks in order to direct the route choice of self-interested agents towards a system optimal. The notion of such a tolling scheme is becoming more practical as cars are becoming increasingly autonomous and the computational effort required to evaluate several alternative routes is becoming more feasible.

This paper makes two main contributions. First, using a traffic micro-simulator (AIM), we provide empirical evidence suggesting that stylized macroscopic traffic models are unable to approximate optimal tolls accurately. Given this finding and the fact that such models require full knowledge of demand and supply and assume that these remain fixed, we conclude that using such models to set tolls in real-life road networks is impractical. This conclusion leads us to the second contribution, the presentation and evaluation of a new tolling scheme, denoted  $\Delta$ -tolling. We provide theoretical and empirical evidence that  $\Delta$ -tolling results in near-optimal system performance while being adaptive to traffic conditions and computationally feasible.

Our ongoing research agenda includes evaluating the performance of  $\Delta$ -tolling in dynamic environments, in which traffic demand and supply is time varying.

## 8 Acknowledgments

A portion of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), and AFOSR (FA9550-14-1-0087). Peter Stone serves on the Board of Directors of, Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research. The authors would also like to acknowledge the support of the Data-Supported Transportation Operations & Planning Center and the National Science Foundation under Grant No. 1254921.

## References

- Manfred J. Beckmann, C. B. McGuire, and C. B. Winston. *Studies in the Economics of Transportation*. Yale University Press, New Haven, CT, 1956.
- Stephen D. Boyles, Kara M. Kockelman, and S. Travis Waller. Congestion pricing under operational, supply-side uncertainty. *Transportation Research Part C*, 18(4):519–535, 2010.
- Dietrich Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1969.
- Anthony Chen and Kittu Subprasom. Analysis of regulation and policy of private toll roads in a build-operate-transfer scheme under demand uncertainty. *Transportation Research Part A*, 41(6):537–558, 2007.
- Andr'e de Palma and Robin Lindsey. Information and usage of congestible facilities under different pricing regimes. *Canadian Journal of Economics*, 31(3):666–692, 1998.
- Robert B. Dial. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B*, 40(10):917–936, 2006.
- Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, pages 591–656, 2008.
- Lauren Gardner, Jennifer Duthie, Avinash Unnikrishnan, and S. Travis Waller. Robust pricing for networks with demand uncertainty, 2008. Presented at the 87th Annual Meeting of the Transportation Research Board, Washington, DC.
- Lauren Gardner, Stephen D. Boyles, and S. Travis Waller. Quantifying the benefit of responsive pricing and travel information in the stochastic congestion pricing problem. *Transportation Research Part A*, 45:204–218, 2011.
- Lauren Gardner, Hillel Bar-Gera, and Stephen D. Boyles. Development and comparison of choice models and tolling schemes for high-occupancy/toll (HOT) facilities. *Transportation Research Part B*, 55:142–153, 2013.
- Lauren Gardner, Stephen D. Boyles, Hillel Bar-Gera, and Kelly Tang. Robust tolling schemes for high-occupancy/toll (HOT) facilities under variable demand. *Transportation Research Record*, 2450:152–162, 2015.
- Deren Han and Hai Yang. Congestion pricing in the absence of demand functions. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):159 – 171, 2009.
- Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.
- Dusica Joksimovic, Michiel C.J. Bliemer, and Piet H.L. Bovy. Optimal toll design problem in dynamic traffic networks with joint route and departure time choice. *Transportation Research Record: Journal of the Transportation Research Board*, 1923(1):61–72, 2005.
- William Karush. *Minima of functions of several variables with inequalities as side constraints*. PhD thesis, Masters thesis, Dept. of Mathematics, Univ. of Chicago, 1939.
- H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.
- Robin Lindsey. Cost recovery from congestion tolls with stochastic capacity and demand. *Journal of Urban Economics*, 66(1):16–24, 2009.
- T. Nagae and T. Akamatsu. Dynamic revenue management of a toll road project under transportation demand uncertainty. *Networks and Spatial Economics*, 6:345–357, 2006.
- M. Patriksson. *The Traffic Assignment Problem — Models and Methods*. VSP, Utrecht, Netherlands, 1994.
- Arthur C. Pigou. *The Economics of Welfare*. Macmillan and Co., London, 1920.
- John G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II*, 1:352–362, 1952.
- Byung-Wook Wie and Roger L. Tobin. Dynamic congestion pricing models for general traffic networks. *Transportation Research Part B: Methodological*, 32(5):313 – 327, 1998.

- Hai Yang, Qiang Meng, and Der-Horng Lee. Trial-and-error implementation of marginal-cost pricing on networks in the absence of demand functions. *Transportation Research Part B: Methodological*, 38(6):477 – 493, 2004.
- Hai Yang. Evaluating the benefits of a combined route guidance and road pricing system in a traffic network with recurrent congestion. *Transportation*, 26:299–322, 1999.
- Hai Yang. System optimum, stochastic user equilibrium, and optimal link tolls. *Transportation Science*, 33(4):354–360, 1999.
- Y. Yin and Y. Lou. Dynamic tolling strategies for managed lanes. *Journal of Transportation Engineering*, 135(2):45–52, 2009.

# When are Marginal Congestion Tolls Optimal?

Reshef Meir, Technion  
reshefm@ie.technion.ac.il

David C. Parkes, Harvard University  
parkes@eecs.harvard.edu

## Abstract

Marginal tolls are known to provide the existence of an optimal equilibrium in atomic congestion games, but unlike nonatomic games, there might be additional equilibria even with linear cost functions on resources. In this paper, we show that in games with a large number of players, all equilibria are near-optimal.

## 1 Introduction

It is well known that selfish routing results in suboptimal social behavior and in increased latency [Pigou, 1920]. The modern literature formalizes selfish routing scenarios as *congestion games*, where the inefficiency due to strategic behavior is quantified as the *Price of Anarchy* (PoA)—the ratio between the optimal total latency and the maximal total latency in equilibrium [Roughgarden and Tardos, 2007].

The game theoretic literature on selfish routing can be classified into models of atomic (unsplittable) flow and nonatomic flow, where in the latter, each agent accounts for an infinitesimally small fraction of the total congestion. While in both models a pure equilibrium is guaranteed to exist, and can be found via a simple local best-response dynamics, atomic congestion games are considered more challenging to analyze. Atomic games may have multiple equilibria of different costs, and the price of anarchy can be much higher than in nonatomic games.

The PoA is well understood in congestion games, both atomic and nonatomic, and almost independent from the topology of the network [Roughgarden, 2009]. That is, the inefficiency depends mostly on the edge latency functions, and a simple network of two parallel edges (or roads) is sufficient to create instances with the highest possible PoA.

Still, it is interesting to look to change the behavior of agents by charging them for using a resource. It has been known since [Beckmann *et al.*, 1956] that to enforce optimal behavior in nonatomic games (i.e. such that all equilibria have minimum total latency), it is sufficient to impose *marginal congestion tolls*, i.e., charge each agent based on the latency he *currently* adds to the other agents.<sup>1</sup> Note that we assume tolls are *dynamic* that depend on monitoring the actual congestion on one hand, but can be easily computed. This is in contrast to *static tolls* that typically depend on the

<sup>1</sup>It is typically assumed that the tolls themselves are not calculated as part of the total cost, e.g. because they return to the society indirectly, or because the central authority only cares about the latency. Non-refundable tolls are also studied [Cole *et al.*, 2006] but not in this paper.

*optimal congestion*, and often require extensive computation (see e.g. [Bonifaci *et al.*, 2011]).

For atomic games, it is known that marginal tolls guarantee the existence of at least one optimal equilibrium [Sandholm, 2007], however there may be other inefficient equilibria, even in games with linear latencies [Caragiannis *et al.*, 2010a]. The problem becomes even more involved if we take into account more general notions of equilibrium such as mixed and correlated equilibrium. For a specific classes of atomic routing games, marginal tolls guarantee optimal behavior in any pure equilibrium. This is the case for example for symmetric networks with parallel links (also known as *resource selection games*) since in such networks the equilibrium is unique. The class of networks for which marginal tolls are optimal was extended first in an unpublished (and unfinished) work by Singh [Singh, 2008]. However Singh’s result was very recently refuted by Igal Milchtaich (personal communications) who provided the correct characterization.

Several other papers studied more complicated taxation schemes and how low they can affect the PoA [Fotakis and Spirakis, 2008; Caragiannis *et al.*, 2010a].

**Our contribution** We show that for any fixed network, if the number of players is sufficiently large, then any equilibrium under marginal tolls is near-optimal. Further, this result extend to mixed, correlated, and coarse correlated equilibria.

We use the smoothness framework [Roughgarden, 2009], which enables the PoA bounds to be established with relatively short and simple proofs.

We also consider agents with variable sensitivity to monetary tolls [Cole *et al.*, 2006; Karakostas and Kolliopoulos, 2004; Fotakis *et al.*, 2010], reflecting how agents trade-off money for time. As discussed in [Yang and Zhang, 2008; Meir and Parkes, 2015b], the parameter may be unobservable, and thus unknown to the central authority setting the tolls. Thus, following [Meir and Parkes, 2015b] and in contrast to most of the mechanism design literature, we assume that a marginal toll is applied, and analyze the equilibrium for a population as the sensitivity parameter varies.

Along the way, we state formally some known results on marginal tolls that seem to have been overlooked in the recent study of atomic congestion and routing games.

## 2 Preliminaries

For an integer  $m$ ,  $[m] = \{1, 2, \dots, m\}$ . We use bold letters to denote vectors, e.g.,  $\mathbf{a} = (a_1, \dots, a_m)$ .

Following the definitions in [Roughgarden, 2007], a routing game is a tuple  $G = \langle V, E, N, \mathbf{c}, \mathbf{u}, \mathbf{v} \rangle$ , where

- $(V, E)$  are vertices and edges of a directed graph;



- $N$  is a finite set of agents of size  $n$ ;
- $\mathbf{c} = (c_e)_{e \in E}$ , where  $c_e(x) \geq 0$  is a non-decreasing function indicating the cost incurred when  $x$  agents use edge  $e$  ( $c_e$  are called *latency functions*);<sup>2</sup>
- $\mathbf{u}, \mathbf{v}$  are vectors of  $n$  vertices each, where  $(u_i, v_i)$  are the source and target nodes of agent  $i$ ;

We denote by  $A_i \subseteq 2^E$  the set of all directed paths between the pair of nodes  $(u_i, v_i)$  in the graph. Thus  $A_i$  is the set of *actions* available to agent  $i$ . We denote by  $A = \cup_i A_i$  the set of all directed source-target paths. A routing game is *symmetric* (also called *single-source-single-target*) if all agents have the same set of actions, i.e.,  $A_i = A$  for all  $i$ .

An *action profile*  $\mathbf{a} = (a_i)_{i \in N}$  specifies the path  $a_i \in A_i$  of each agent  $i$ , and  $\mathcal{A} = \times_{i \in N} A_i$  is the set of all action profiles. We denote by  $\mathbf{s}_e(\mathbf{a}) \in \mathbb{N}$  the congestion on edge  $e \in E$  in profile  $\mathbf{a}$ , i.e.,  $\mathbf{s}_e = \mathbf{s}_e(\mathbf{a}) = |\{i \in N : e \in a_i\}|$  ( $\mathbf{a}$  is omitted when clear from context).

The cost for agent  $i$  in profile  $\mathbf{a}$  is summed over all edges,  $C_i(\mathbf{a}) = \sum_{e \in a_i} c_e(s_e)$ . The *social cost* in a profile  $\mathbf{a}$  in game  $G$  is attained by summing over all agents:

$$SC(G, \mathbf{a}) = \sum_{i=1}^n C_i(\mathbf{a}) = \sum_{i=1}^n \sum_{e \in a_i} c_e(s_e) = \sum_{e \in E} s_e c_e(s_e). \quad (1)$$

We denote by  $\mathbf{a}^* = \mathbf{a}^*(G) = \operatorname{argmin}_{\mathbf{a} \in \mathcal{A}} SC(G, \mathbf{a})$  the profile that minimizes the social cost (optimal profile).

A profile  $\mathbf{a}$  is a *pure Nash equilibrium* if no agent can gain by changing her strategy, i.e. if for all  $i \in N, a'_i \in A_i, C_i(\mathbf{a}) \leq C_i(\mathbf{a}_{-i}, a'_i)$ , where  $\mathbf{a}_{-i} = (a_j)_{j \neq i}$ . The definition of equilibrium extends to mixed and correlated strategies. We omit the formal details. Denote by  $PNE(G) \subseteq \mathcal{A}$  the sets of pure Nash equilibria of  $G$ .

The *price of anarchy* (PoA) of  $G$  is the ratio between the social cost of worst equilibrium and the optimal profile, i.e.  $\text{PoA}(G) = \frac{\max\{SC(G, \mathbf{a}) : \mathbf{a} \in PNE(G)\}}{SC(G, \mathbf{a}^*)}$  (the definition of mixed- and correlated-POA is similar). It is well known that the PoA can be upper bounded using only the class of latency functions in  $G$ , regardless of the structure of  $(V, E)$ . For example, if all of  $c_e$  are affine functions ( $c_e(x) = a_e x + b_e$  for  $a_e, b_e \geq 0$ ) then  $\text{PoA}(G) \leq \frac{5}{2}$ , and this is true for mixed and correlated-PoA as well [Roughgarden, 2009].

The *price of stability* (PoS) of  $G$  is similarly defined as the ratio between the *best* equilibrium and the optimal profile [Christodoulou and Koutsoupias, 2005], i.e.  $\text{PoS}(G) = \frac{\min\{SC(G, \mathbf{a}) : \mathbf{a} \in PNE(G)\}}{SC(G, \mathbf{a}^*)}$ .

**Biased games** We are interested in a *biased game*, in our case because of the use of tolls.<sup>3</sup> A biased game is a pair  $(G, \hat{G})$  such that  $G, \hat{G}$  are identical except in their latency functions. Informally, we assume that players behave according to the “biased costs”  $(\hat{c}_e)_{e \in E}$  (e.g. play an equilibrium of  $\hat{G}$ ), but social cost is measured w.r.t. the “real costs”  $(c_e)_{e \in E}$ .

<sup>2</sup>Some authors prefer the term “arc” for directed edges. We stick with the common term in computer science.

<sup>3</sup>Biased games are also used to model cognitive and behavioral traits such as risk aversion [Ordóñez and Stier-Moses, 2010] or altruism [Caragiannis et al., 2010b].

The *biased price of anarchy/stability* (BPoA/BPoS) compares the equilibria of  $\hat{G}$  to the optimum of  $G$ , using the real social cost of both. Formally,  $\text{BPoA}(G, \hat{G}) = \frac{\max\{SC(G, \mathbf{a}) : \mathbf{a} \in PNE(\hat{G})\}}{SC(G, \mathbf{a}^*)}$ , and similarly for BPoS.

The primary bias we will consider in this paper is tolls, and in particular *marginal tolls*. That is, we define  $\tau_e(x) = (x-1)[c_e(x) - c_e(x-1)]$ , and set  $\hat{c}_e^M(x) = c_e(x) + \tau_e(x)$ . Toll  $\tau_e(x)$  is exactly the marginal cost inflicted upon the remaining  $x-1$  agents who use  $e$  due to an additional agent. Other toll schemes  $T$  can be similarly defined, replacing  $\tau_e(x)$  with any other non-negative function  $T_e(x)$ .

A toll scheme  $T$  *strongly enforces* optimal flow in a game  $G$  if all equilibria of  $\hat{G}^T$  (i.e., the game with biased costs  $\hat{c}^T$ ) are optimal in  $G$  (equivalently, if  $\text{BPoA}(G, \hat{G}^T) = 1$ ) [Fotakis and Spirakis, 2008]. Similarly, a toll scheme *weakly enforces* optimal flows if  $\text{BPoS}(G, \hat{G}^T) = 1$ .

Marginal tolls in the nonatomic Pigouvian model were suggested by Beckmann [Beckmann et al., 1956], who showed they strongly enforce optimal flows in that model. Our goal is to understand the power of marginal tolls in atomic routing games.

### 3 Marginal tolls are weakly optimal

The marginal toll scheme for atomic games coincides with the taxes proposed by Sandholm [Sandholm, 2007], albeit Sandholm defined taxes at the strategy level, rather than tolls on particular edges. The observation that marginal tolls weakly enforce optimal flows was also made in an unpublished report by Singh [Singh, 2008].<sup>4</sup> We state the result for the standard routing games framework.

**Theorem 1** ([Sandholm, 2007; Singh, 2008]). *For any atomic congestion game  $G$ , there is a pure Nash equilibrium in  $\hat{G}^M$  that is optimal in  $G$ . Equivalently,  $\text{BPoS}(G, \hat{G}^M) = 1$ .*

The theorem follows from a simple observation:  $\hat{G}^M$  is a potential game [Rosenthal, 1973], whose potential function  $\phi(\hat{G}^M, \mathbf{a})$  coincides with the social welfare of  $G$ . Thus the optimum of  $SC(G, \mathbf{a})$  must be a local minimum of  $\phi(\hat{G}^M, \mathbf{a})$ , i.e. a pure Nash equilibrium. Quite strikingly, the theorem was extended to a much more general framework where agents have idiosyncratic preferences over strategies, and congestion may depend on agents weight or other features [Sandholm, 2007; Singh, 2008].

Unfortunately, in atomic games there may be additional suboptimal equilibria.

**Example 1.** *Consider a game with 3 parallel links,  $E = \{a, b, c\}$  and 3 agents  $N = \{1, 2, 3\}$ .  $A_1 = \{a, b\}$ ,  $A_2 = \{b, c\}$ , and  $A_3 = \{c\}$ . Latency functions are  $c_b(x) = c_c(x) = x$ ,  $c_a \equiv 2$  (see Fig. 1). The modified cost functions under any edge-independent nonnegative tolls can be written as  $\hat{c}_b(x) = \hat{c}_c(x) = (1, 2 + T(x), 3 + T'(x))$ . The unique optimum is  $\mathbf{a}^* = (a, b, c)$  with cost  $SC(\mathbf{a}^*) = 2 + 1 + 1 = 4$ , which is also a PNE. However, there is another PNE  $\mathbf{a}' =$*

<sup>4</sup>Recent works on tolls in routing games seem to be unaware of this observation [Fotakis and Spirakis, 2008; Fotakis et al., 2010; Swamy, 2012].



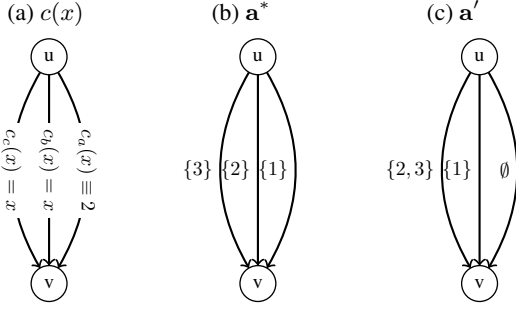


Figure 1: Figure (1a) shows the base game  $G$ . The other figures show the optimal state  $\mathbf{a}^*$  and the state  $\mathbf{a}'$  which is an additional equilibrium of both  $G$  and  $\hat{G}^T$ .

(b, c, c) with cost  $SC(\mathbf{a}') = 1 + 2 + 2 = 5$ . This remains a PNE of  $\hat{G}^T$  as long as  $\hat{c}_b^T(x) = \hat{c}_c^T(x)$ : agent 2 is not allowed to use edge  $a$ , and agent 1 does not want to use it since  $\hat{c}_a(1) = 2 > 1 = \hat{c}_b(1)$ .

This means that marginal tolls in atomic games do not, in the general case, strongly enforce optimal flows.

## 4 Strongly Enforcing Optimal Flows

The prominent technique for proving PoA bounds is *smoothness analysis*. In short, a game  $G$  is  $(\lambda, \mu)$ -smooth if for all  $\mathbf{a} \in \mathcal{A}$  there is  $\mathbf{a}' \in \mathcal{A}$  such that  $\sum_{i \in N} C_i(\mathbf{a}_{-i}, a'_i) \leq \lambda SC(G, \text{OPT}(G)) + \mu SC(G, \mathbf{a})$ . If a game  $G$  (not just a routing game) is  $(\lambda, \mu)$ -smooth, then  $\text{PoA}(G) \leq \frac{\lambda}{1-\mu}$  [Roughgarden, 2009]. Further, this holds for the mixed, correlated, and coarse-correlated PoA as well. For routing games, it is also shown that restricting the class of latency functions results in smooth games. For example, if all cost functions are affine, then  $G$  is  $(\frac{5}{3}, \frac{1}{3})$ -smooth (thereby showing  $\text{PoA}(G) \leq \frac{5}{2}$ ).

Given a biased game  $(G, \hat{G})$ , we can similarly define the property of *biased smoothness*.

**Definition 1.**  $(G, \hat{G})$  is  $(\hat{\lambda}, \hat{\mu})$ -biased smooth (BS), if there is  $\mathbf{a}'$  s.t. for any profile  $\mathbf{a}$ ,

$$\sum_{j \in N} (C_j(\mathbf{a}) + \hat{C}_j(\mathbf{a}_{-j}, a'_j) - \hat{C}_j(\mathbf{a})) \leq \hat{\lambda} SC(G, \text{OPT}(G)) + \hat{\mu} SC(G, \mathbf{a}). \quad (2)$$

It is easy to see that if  $G$  is  $(\lambda, \mu)$ -smooth, then  $(G, G)$  is  $(\lambda, \mu)$ -BS: we set  $\mathbf{a}' = \text{OPT}(G)$ , and note that  $\sum_{j \in N} (C_j(\mathbf{a}) + C_j(\mathbf{a}_{-j}, a'_j) - C_j(\mathbf{a})) = \sum_{j \in N} C_j(\mathbf{a}_{-j}, a'_j)$ .

**Theorem 2.** Suppose that  $(G, \hat{G})$  is  $(\hat{\lambda}, \hat{\mu})$ -BS. Let  $\sigma$  be any equilibrium (pure, mixed, correlated, or coarse-correlated) of the game  $\hat{G}$ . Then  $SC(G, \sigma) \leq \frac{\hat{\lambda}}{1-\hat{\mu}} SC(G, \text{OPT}(G))$ .

The original proof of Roughgarden [Roughgarden, 2009] for the PoA (and coarse-correlated PoA) naturally extends to biased smoothness.<sup>5</sup> For completeness, we provide the proof

<sup>5</sup>A similar definition of smoothness was applied, for example, for finite congestion games with altruism: when  $\hat{C}(\mathbf{a})$  is a combination of  $C(\mathbf{a})$  and  $SC(\mathbf{a})$ , then the BPoA coincides with the ‘‘robust PoA’’ of Chen et al. [Chen et al., 2011].

(almost identical to the ones in [Roughgarden, 2009; Chen et al., 2011]) in the appendix.

In particular,  $(1, 0)$ -BS means that  $\text{BPoA}(G, \hat{G}) = 1$ , i.e. that any PNE of  $\hat{G}$  is optimal in  $G$ .

We are interested in showing that  $(G, \hat{G}^M)$  is BS for some reasonable parameters  $\hat{\lambda}, \hat{\mu}$ .

### 4.1 Smoothness in the large

When an atomic game becomes *large*, i.e. when we fix the network and increase the number of players, there is evidence that the game behaves more similarly to a nonatomic game [Feldman et al., 2015]. We show how to extend biased-smoothness analysis (and in particular marginal tolls) to large atomic games. While we can not apply the results of Feldman et al. directly, our techniques are inspired by theirs.

**Lemma 3.** Let  $\mathbf{a}, \mathbf{a}'$  be any two profiles in  $G$  with  $n$  agents, and let  $\epsilon = \epsilon(G) = \max_{e \in E, x \in \mathbb{N}} (c_e(x+1) - c_e(x))$ . Then  $\sum_{j \in N} C_j(\mathbf{a}_{-j}, a'_j) - C_j(\mathbf{a}) \leq \sum_{e \in E} (s'_e - s_e) c_e(s_e) + O(n\epsilon)$ .

*Proof.*

$$\begin{aligned} & \sum_{j \in N} (C_j(\mathbf{a}_{-j}, a'_j) - C_j(\mathbf{a})) \\ &= \sum_{j \in N} \left( \left( \sum_{e \in a'_j \setminus a_j} c_e(s_e + 1) + \sum_{e \in a_j \cap a'_j} c_e(s_e) \right) - \sum_{e \in a_j} c_e(s_e) \right). \end{aligned}$$

By definition of  $\epsilon$ , we continue:

$$\begin{aligned} & \leq \sum_{j \in N} \left( \left( \sum_{e \in a'_j \setminus a_j} (c_e(s_e) + \epsilon) + \sum_{e \in a_j \cap a'_j} c_e(s_e) \right) - \sum_{e \in a_j} c_e(s_e) \right) \\ & \leq \sum_{j \in N} \left( \sum_{e \in a'_j} c_e(s_e) - \sum_{e \in a_j} c_e(s_e) \right) + \sum_{j \in N} \sum_{e \in E} \epsilon \\ & = \sum_{j \in N} (s'_e c_e(s_e) - s_e c_e(s_e)) + n|E|\epsilon. \end{aligned}$$

□

That is, we can write the sum of deviations as a function of the aggregate congestion (approximately).

Next, we think of a sequence of atomic games with increasing  $n$ : We fix a network  $(V, E)$  and continuous quasi-convex cost functions  $\mathbf{c} = (c_e)_{e \in E}$ , where  $c_e : [0, 1] \rightarrow \mathbb{R}^+$ . For ease of presentation, we consider symmetric games (i.e. where there is just one source-target pair  $u, v \in V$ ), although similar arguments extend to asymmetric games. This already induces a symmetric nonatomic game  $\tilde{G} = (V, E, u, v, \mathbf{c})$ . For  $n \in \mathbb{N}$ , we define  $G_n$  by setting  $G_n = (V, E, N, u, v, \mathbf{c}^n)$ , where  $c^n(x) = c(x/n)$ . Thus  $\tilde{G}$  is the limit of  $(G_n)_{n=1,2,\dots}$  (we call it the *limit game*).

Our continuous cost functions can also be subject to biases. Let  $\hat{c}_e$  be the biased continuous cost of  $\tilde{c}_e$ , and  $\hat{c}_e^n(x) = \hat{c}_e(x/n)$ . Biased-smoothness for continuous cost functions was defined and explored in [Meir and Parkes, 2015b]: we say that  $c$  is  $(\hat{\lambda}, \hat{\mu})$ -biased smooth w.r.t.  $\hat{c}$  if for all  $t, t' \in \mathbb{R}_+$ ,

$$c(t)t + \hat{c}(t)(t' - t) \leq \hat{\lambda} c(t')t' + \hat{\mu} c(t)t.$$

Clearly, if  $\tilde{c}$  is  $(\hat{\lambda}, \hat{\mu})$ -biased smooth w.r.t.  $\hat{c}$ , then  $c^n$  is  $(\hat{\lambda}, \hat{\mu})$ -biased smooth w.r.t.  $\hat{c}^n$  for any  $n$ .

**Theorem 4.** Consider a limit game  $\tilde{G}$ , where  $\tilde{c}_e$  are quasi-convex and  $(\hat{\lambda}, \hat{\mu})$ -biased smooth w.r.t. the bias  $\hat{c}$ . Then for any  $\delta > 0$  there are  $\epsilon > 0, n(\epsilon)$  s.t. for all  $n > n(\epsilon)$ , the atomic game  $(G^n, \hat{G}^n)$  is  $((1 + \delta)\hat{\lambda}, \hat{\mu})$ -BS. In particular,

$$BPoA(G^n, \hat{G}^n) \leq (1 + \delta) \frac{\hat{\lambda}}{1 - \hat{\mu}},$$

and this extends to any coarse-correlated equilibrium.

*Proof.* Let  $\mathbf{a}' = \text{OPT}(G^n)$ ,  $Z^n = SC(G^n, \mathbf{a}')$ . Since  $SC(G^n, \mathbf{a}') = \Omega(n)$  (the cost for each agent is at least some constant), we write  $Z^n > \rho n$  for some  $\rho > 0$  and  $n > n(\rho)$ .

Since  $\tilde{c}_e$  is bounded and continuous for all  $e \in E$ ,

$$\begin{aligned} \max_{x \in [n]} \{c_e^n(x+1) - c_e^n(x)\} &= \max_{x \in [n]} \left\{ \tilde{c}_e\left(\frac{x}{n} + \frac{1}{n}\right) - \tilde{c}_e\left(\frac{x}{n}\right) \right\} \\ &\leq \sup_{t \in [0,1]} \left\{ \tilde{c}_e\left(t + \frac{1}{n}\right) - \tilde{c}_e(t) \right\} \xrightarrow{n \rightarrow \infty} 0, \end{aligned}$$

and thus for all  $\epsilon > 0$  there is some  $n(\epsilon)$  s.t. for all  $n > n(\epsilon)$ , we have  $c_e^n(x+1) - c_e^n(x) < \epsilon$ . By Lemma 3

$$\begin{aligned} SC(G^n, \mathbf{a}) + \sum_{j \in N} \hat{C}_j^n(\mathbf{a}_{-j}, a_j) - \hat{C}_j^n(\mathbf{a}) \\ &\leq SC(G^n, \mathbf{a}) + \sum_{e \in E} (s'_e - s_e) \hat{c}_e^n(s_e) + O(n\epsilon) \\ &= \sum_{e \in E} (s_e c_e^n(s_e) + (s'_e - s_e) \hat{c}_e^n(s_e)) + n\epsilon' \\ &\leq \sum_{e \in E} (\hat{\lambda} c_e^n(s'_e) s'_e + \hat{\mu} c_e^n(s_e) s_e) + n\epsilon' \quad (\text{smoothness}) \\ &= \hat{\lambda} Z^n + \hat{\mu} SC(G^n, \mathbf{a}) + n\epsilon' \\ &< \hat{\lambda} SC(G^n, \mathbf{a}') + \hat{\mu} SC(G^n, \mathbf{a}) + \frac{1}{\rho} Z^n \epsilon' \quad (Z^n > \rho n) \\ &= \left(\hat{\lambda} + \frac{\epsilon'}{\rho}\right) Z^n + \hat{\mu} SC(G^n, \mathbf{a}) \\ &\leq \left(1 + \frac{\epsilon'}{\rho}\right) \hat{\lambda} Z^n + \hat{\mu} SC(G^n, \mathbf{a}). \quad (\hat{\lambda} \geq 1) \end{aligned}$$

Selecting  $\epsilon' < \delta \rho$  (and thus sufficiently small  $\epsilon > 0$ , and  $n > \max\{n(\rho), n(\epsilon)\}$ ), completes the proof. The BPoA bound then follows directly from Theorem 2.  $\square$

Since biased smoothness hold for various pairs of cost functions, Theorem 4 is quite useful. Mainly, we get that marginal tolls strongly enforce near-optimal flow if there are enough players.

**Corollary 5.** Consider any limit game  $\tilde{G}$ , where  $\tilde{c}_e$  are quasi-convex. Then for any  $\delta > 0$  there is some  $n(\delta)$  s.t. for all  $n > n(\delta)$ ,  $BPoA(G^n, \hat{G}^n) \leq 1 + \delta$ .

*Proof.* Consider the continuous version of marginal tolls  $\hat{c}(t) = \tilde{c}(t) + t \cdot \frac{\partial c(t)}{\partial t}$  [Beckmann *et al.*, 1956].<sup>6</sup> The proof follows directly from Theorem 4 and from the fact that any quasi-convex function  $\tilde{c}$  is  $(1, 0)$ -biased smooth w.r.t.  $\hat{c}$  [Meir and Parkes, 2015b].  $\square$

<sup>6</sup>Due to rounding,  $\hat{c}^n(x)$  is very close, but not identical to the discrete  $\hat{c}^M(x)$  we previously defined.

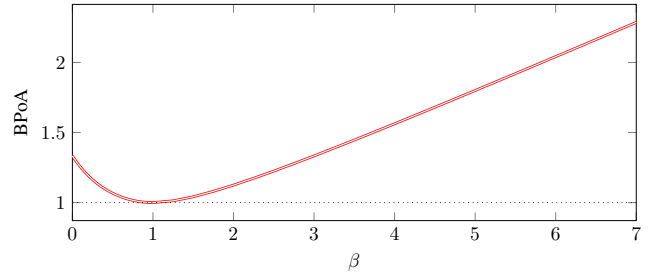


Figure 2: The X-axis shows the tax-sensitivity of agents, where  $\beta = 0$  means they ignore the tax.

The double red lines are the tight bounds on the BPoA for large games with affine costs stated in Corollary 6.

## 5 Tax-sensitivity

We next consider agents with variable sensitivity to monetary tolls, as in [Cole *et al.*, 2006]. Formally, the marginal toll  $\tau_e(x)$  is imposed on edge  $e$ , but the cost experienced by the agents is  $\hat{c}_e^\beta(x) = c(x) + \beta \cdot \tau_e(x)$ , where  $\beta$  is a parameter reflecting how agents trade-off money for time. Denote by  $\hat{G}_\beta$  the biased game obtained from  $G$  by replacing every cost function  $c_e(x)$  with  $\hat{c}_e^\beta(x)$ . We analyze the equilibrium for a population with parameter  $\beta$  (where  $\beta = 1$  means that  $\hat{c}_e^\beta(x) = \hat{c}_e^M(x)$ ).

In [Meir and Parkes, 2015b], various BPoA bounds are derived for nonatomic games with various classes of cost functions (general/convex/polynomial/linear). We show how these bounds extend to large games.

For large atomic games, all the biased smoothness bounds from [Meir and Parkes, 2015b] for tax-sensitivity and other biases immediately apply. These bounds are also known to be tight.

For example, it was shown that affine cost functions (of the form  $\tilde{c}(t) = at + b$  for  $a, b \geq 0$ ) are  $(1, \frac{(1+\beta)^2}{4} - \beta)$ -biased smooth w.r.t.  $\hat{c}(t)$  as defined above for all  $\beta \leq 1$  and  $(\frac{(1+\beta)^2}{4\beta}, 0)$ -biased smooth for  $\beta \geq 1$ . We get the following corollary due to Theorem 4:

**Corollary 6.** Consider any limit game  $\tilde{G}$ , where  $\tilde{c}_e$  are affine. Then for any  $\delta > 0$  there is some  $n(\delta)$  s.t. for all  $n > n(\delta)$ ,  $BPoA(G^n, \hat{G}^n) \leq \frac{1}{(\beta+1) - \frac{(1+\beta)^2}{4}}$  if  $\beta \leq 1$ , and  $BPoA(G^n, \hat{G}^n) \leq \frac{(1+\beta)^2}{4\beta}$  if  $\beta \geq 1$ .

Another benefit of smoothness-in-the-large is that the parameters  $\hat{\lambda}, \hat{\mu}$  are typically much smaller for classes of continuous functions than for the corresponding class of discrete costs. Indeed, [Feldman *et al.*, 2015] show that the PoA of large games is significantly smaller due to this: for linear costs the PoA drops from  $\frac{5}{2}$  to  $\frac{4}{3}$ , and for polynomials of degree  $d$ , the PoA drops from  $\Omega(2^d)$  to  $O(\frac{d}{\ln d})$ . Our result shows that this still holds for large games with biases. For brevity we do not re-state all the results from [Meir and Parkes, 2015b] for large atomic games, however Fig. 2 shows the bounds for affine costs.

## 6 Discussion

We have studied the problem of strongly enforcing optimal flows in atomic congestion games through marginal congestion tolls. Such tolls always weakly enforce optimal flows, and strongly enforce optimal tolls in large games. Further, our analysis extends to games where agents' tax-sensitivity is not aligned with that of the designer. This is particularly important in the context of mechanism design where we seek to shape drivers' incentives and lead the system to a good equilibrium [Tumer and Agogino, 2006], and when drivers are subject to cognitive and behavioral biases such as risk-aversion [Ordóñez and Stier-Moses, 2010; Nikolova and Stier-Moses, 2015]. One important challenge is to extend the BPoA bounds to games where agents differ in their levels of risk aversion or tax sensitivity. This has been done to some extent in nonatomic games [Meir and Parkes, 2015a; 2015b].

More broadly, this work provides more evidence for the usefulness of "biased-smoothness" analysis, in the line of [Chen *et al.*, 2011; Meir and Parkes, 2015b], and we hope it can lead to a better understanding of routing games where agents are subject to either external influences (like tolls) or behavioral biases.

## Acknowledgments

We thank Itai Arieli for pointing us to Sandholm's work.

## References

- [Beckmann *et al.*, 1956] M. Beckmann, B. McGuire, and C. Winsten. *Studies in the Economics of Transportation*. Yale University Press, New Haven, 1956.
- [Bonifaci *et al.*, 2011] Vincenzo Bonifaci, Mahyar Salek, and Guido Schäfer. Efficiency of restricted tolls in non-atomic network routing games. In *SAGT'11*, pages 302–313, 2011.
- [Caragiannis *et al.*, 2010a] Ioannis Caragiannis, Christos Kaklamani, and Panagiotis Kanellopoulos. Taxes for linear atomic congestion games. *ACM Transactions on Algorithms*, 7(1):13, 2010.
- [Caragiannis *et al.*, 2010b] Ioannis Caragiannis, Christos Kaklamani, Panagiotis Kanellopoulos, Maria Kyropoulou, and Evi Papaioannou. The impact of altruism on the efficiency of atomic congestion games. In *TGC*, pages 172–188. Springer, 2010.
- [Chen *et al.*, 2011] Po-An Chen, Bart De Keijzer, David Kempe, and Guido Schäfer. The robust price of anarchy of altruistic games. In *WINE'11*, pages 383–390. Springer, 2011.
- [Christodoulou and Koutsoupias, 2005] George Christodoulou and Elias Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. In *Algorithms-ESA 2005*, pages 59–70. Springer, 2005.
- [Cole *et al.*, 2006] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. How much can taxes help selfish routing? *Journal of Computer and System Sciences*, 72(3):444–467, 2006.
- [Feldman *et al.*, 2015] Michal Feldman, Nicole Immorlica, Brendan Lucier, Tim Roughgarden, and Vasilis Syrgkanis. The price of anarchy in large games. *arXiv preprint arXiv:1503.04755*, 2015.
- [Fotakis and Spirakis, 2008] Dimitris Fotakis and Paul G Spirakis. Cost-balancing tolls for atomic network congestion games. *Internet Mathematics*, 5(4):343–363, 2008.
- [Fotakis *et al.*, 2010] Dimitris Fotakis, George Karakostas, and Stavros G Kolliopoulos. On the existence of optimal taxes for network congestion games with heterogeneous users. In *SAGT'10*, pages 162–173. Springer, 2010.
- [Karakostas and Kolliopoulos, 2004] George Karakostas and Stavros G Kolliopoulos. Edge pricing of multicommodity networks for heterogeneous selfish users. In *FOCS'04*, pages 268–276, 2004.
- [Meir and Parkes, 2015a] Reshef Meir and David Parkes. Congestion games with distance-based strict uncertainty. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.
- [Meir and Parkes, 2015b] Reshef Meir and David Parkes. Playing the wrong game: Smoothness bounds for congestion games with behavioral biases. In *NETECON'15*, pages 67–70, 2015.
- [Nikolova and Stier-Moses, 2015] Evdokia Nikolova and Nicolas E Stier-Moses. The burden of risk aversion in mean-risk selfish routing. In *ACM-EC'15*, pages 489–506. ACM, 2015.
- [Ordóñez and Stier-Moses, 2010] Fernando Ordóñez and Nicolás E Stier-Moses. Wardrop equilibria with risk-averse users. *Transportation Science*, 44(1):63–86, 2010.
- [Pigou, 1920] Arthur Cecil Pigou. *The economics of welfare*. Palgrave Macmillan, 1920.
- [Rosenthal, 1973] Robert W Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.
- [Roughgarden and Tardos, 2007] Tim Roughgarden and Eva Tardos. Introduction to the inefficiency of equilibria. In Noam Nisan *et al.*, editor, *Algorithmic Game Theory*, pages 443–459. Cambridge University Press, 2007.
- [Roughgarden, 2007] Tim Roughgarden. Routing games. In Noam Nisan *et al.*, editor, *Algorithmic game theory*, pages 459–484. Cambridge University Press, 2007.
- [Roughgarden, 2009] Tim Roughgarden. Intrinsic robustness of the price of anarchy. In *STOC'09*, pages 513–522. ACM, 2009.
- [Sandholm, 2007] William H Sandholm. Pigouvian pricing and stochastic evolutionary implementation. *Journal of Economic Theory*, 132(1):367–382, 2007.
- [Singh, 2008] Chandramani Singh. Marginal cost pricing for atomic network congestion games. Technical report, Department of Electrical Communication Engineering, Indian Institute of Science, 2008.
- [Swamy, 2012] Chaitanya Swamy. The effectiveness of stackelberg strategies and tolls for network congestion games. *ACM Transactions on Algorithms*, 8(4):36, 2012.
- [Tumer and Agogino, 2006] Kagan Tumer and Adrian Agogino. Agent reward shaping for alleviating traffic congestion. In *Workshop on Agents in Traffic and Transportation*. Citeseer, 2006.
- [Yang and Zhang, 2008] Hai Yang and Xiaoning Zhang. Existence of anonymous link tolls for system optimum on networks with mixed equilibrium behaviors. *Transportation Research Part B: Methodological*, 42(2):99–112, 2008.

## A Omitted proofs

**Theorem 2.** *Suppose that  $(G, \hat{G})$  is  $(\hat{\lambda}, \hat{\mu})$ -BS. Let  $\sigma$  be any equilibrium (pure, mixed, correlated, or coarse-correlated) of the game  $\hat{G}$ . Then  $SC(G, \sigma) \leq \frac{\hat{\lambda}}{1-\hat{\mu}} SC(G, OPT(G))$ .*

*Proof.* For a correlated profile  $\sigma$  we denote  $SC(G, \sigma) = E_{\mathbf{a} \sim \sigma}[SC(G, \mathbf{a})]$ .

By Def. 1, there is a profile  $\mathbf{a}'$  s.t. Eq. (2) holds for every profile  $\mathbf{a}$ .

It is sufficient to prove for a CCE  $\sigma$ . By definition of CCE, for any  $i \in N, b_i \in A_i$ ,

$$E_{\mathbf{a} \sim \sigma}[\hat{C}_i(\mathbf{a})] \leq E_{\mathbf{a} \sim \sigma}[\hat{C}_i(\mathbf{a}_{-i}, b_i)]. \quad (3)$$

$$SC(G, \sigma) = E_{\mathbf{a} \sim \sigma}[SC(G, \mathbf{a})] \leq E_{\mathbf{a} \sim \sigma}[SC(G, \mathbf{a}')] \quad (4)$$

$$\begin{aligned} &+ \left( \sum_{i=1}^n E_{\mathbf{a} \sim \sigma}[\hat{C}_i(\mathbf{a}_{-i}, a'_i)] - E_{\mathbf{a} \sim \sigma}[\hat{C}_i(\mathbf{a}')] \right) \\ &= E_{\mathbf{a} \sim \sigma} \left[ SC(G, \mathbf{a}') + \sum_{i=1}^n \hat{C}_i(\mathbf{a}_{-i}, a'_i) - \hat{C}_i(\mathbf{a}') \right] \quad (5) \end{aligned}$$

$$\begin{aligned} &= E_{\mathbf{a} \sim \sigma} \left[ \sum_{i=1}^n \left( C_i(\mathbf{a}') + \hat{C}_i(\mathbf{a}_{-i}, a'_i) - \hat{C}_i(\mathbf{a}') \right) \right] \\ &\leq E_{\mathbf{a} \sim \sigma} \left[ \hat{\lambda} SC(G, OPT(G)) + \hat{\mu} SC(G, \mathbf{a}') \right] \quad (6) \end{aligned}$$

$$= \hat{\lambda} SC(G, OPT(G)) + \hat{\mu} SC(G, \sigma), \quad (7)$$

where Inequality (4) follows from Eq. (3) with  $b_i = a'_i$ , (5)+(7) from linearity of expectation, and (6) from Eq. (2) applied for each  $\mathbf{a}$ . By rearranging terms, we get the bound in the theorem.  $\square$

# Conflicting Tendencies in Pedestrian Wayfinding Decisions: a Multi-Agent Model Encompassing Proxemics and Imitation

Luca Crociani, Giuseppe Vizzari and Stefania Bandini

Complex Systems and Artificial Intelligence Research Centre,  
University of Milano-Bicocca, Viale Sarca 336/14, 20126, Milan, Italy

## Abstract

Computer-based simulation of pedestrian dynamics is a consolidated application of agent-based models but it still presents open challenges. The wayfinding of pedestrians is a fundamental aspect to allow the application of such models on complex environments. Several novel approaches have recently been proposed in the literature, yet the lack of empirical knowledge still limits the reliability of the heuristics used in the models. In this paper, a novel model for the simulation of pedestrian wayfinding is discussed and the aim is to provide general mechanisms that can be calibrated for the reproduction of empirical evidences. The model is, in fact, inspired by the behaviors observed in a experiment performed with human volunteers in November 2015, which were put into a trade off scenario, since different paths were available but the shortest one was quickly congested. We observed that several pedestrians choose longer trajectories to preserve high walking speed, and often do so following a first *emerging leader*. The proposed model encompasses both a proxemic tendency to avoid congestion, as well as an *imitation* mechanism: these conflicting tendencies can be calibrated according to empirical evidences. A demonstration of the simulated dynamics on a larger scenario will be illustrated in the paper.

## 1 Introduction

The simulation of the movement of pedestrians and crowds in spatial structures is a consolidated research and application context that still presents challenges for researchers in different fields and disciplines: both the automated analysis and the synthesis of pedestrian and crowd behaviour, as well as attempts to integrate these complementary and activities [Vizzari and Bandini, 2013], present open issues and potential developments in a smart environment perspective [Sassi *et al.*, 2015]. Although the currently available commercial tools are used on a day-to-day basis by designers and plan-

ners<sup>1</sup>, according to a report commissioned by the Cabinet Office [Challenger *et al.*, 2009] there is still room for innovations in models, to improve their effectiveness in modeling pedestrians and crowd phenomena, their expressiveness (i.e. simplifying the modeling activity or introducing the possibility of representing phenomena that were still not considered by existing approaches) and efficiency.

Even if we only consider choices and actions related to walking, modeling human decision making activities and actions is a complicated task: different types of decisions are taken at different levels of abstraction, from path planning to the regulation of distance from other pedestrians and obstacles present in the environment. Moreover, the measure of success and validity of a model is definitely not the *optimality* with respect to some cost function, as (for instance) in robotics, but the *plausibility*, the adherence of the simulation results to data that can be acquired by means of observations or experiments.

The present research effort is aimed at producing insights on this aspect: an experiment involving pedestrians has been set up to investigate to which extent pedestrians facing a relatively simple choice (i.e. choose one of two available gateways leading to the same target area) in which, however, they can face a trade-off situation between length of the trajectory to be covered and estimated travel time. The closest gateway, in fact, is initially selected by most pedestrians but it is too narrow to allow a smooth passage of so many pedestrians, and it quickly becomes congested. The other choice can therefore become much more reasonable, allowing a higher average walking speed and comparable (if not even lower) travel time. We observed that several pedestrians choose longer paths to preserve high walking speed, and often do so following a first *emerging leader*. Modeling this kind of choices with current approaches can be problematic.

The present work represents a step in the direction of producing a general model fitting this kind of evidences. The proposed model encompasses both a proxemic tendency to avoid congestion, as well as an *imitation* mechanism: these conflicting tendencies can be calibrated according to empirical evidences. After a discussion of relevant related works, an analysis of different alternatives for modeling and simulating

---

<sup>1</sup>See <http://www.evacmod.net/?q=node/5> for a large list of pedestrian simulation tools).

this kind of scenario will be illustrated in Section 3. Results of the application of the proposed model in a real world scenario, initially described in [Wagoum *et al.*, 2012], will then be described, with reference to their plausibility. Conclusions and future works will end the paper.

## 2 Related Works

The inclusion in simulation models of decisions related to trade off scenarios, such as the one between overall trajectory length and presumed travel time (considering congestion in perceived alternative gateways), represent an issue in current modeling approaches.

Commercial instruments, for instance, mostly provide basic tools to the modelers, that are enabled and required to specify how the population of pedestrians will behave: this implies that the operator constructing the simulation model needs to specify how the pedestrians will generally choose their route (generally selecting among different alternatives defined by means of annotation of the actual spatial structure of the simulated environment through landmarks representing intermediate or final destinations [Kretz *et al.*, 2014]), as well the conditions generating exceptions to the so called “least effort principle”, suggesting that pedestrians generally try to follow the (spatially) shortest path toward their destination. Space, in fact, represents just one of the relevant aspects in this kind of choice: since most pedestrians will generally try to follow these “best paths” congestion can arise and pedestrians can be pushed to make choices that would be sub-optimal, from the perspective of traveled distance.

Recent works in the area of pedestrian and crowd simulation started to investigate this aspect. In particular, [Guo and Huang, 2011] proposed the modification of the floor-field Cellular Automata [Burstedde *et al.*, 2001] approach for considering pedestrian choices not based on the shortest distance criterion but considering the impact of congestion on travel time. [Wagoum *et al.*, 2012] explored the implications of four different strategies for the management of route choice operations, through the combination of applying the shortest or quickest path, with a local (i.e., minimize time to vacate the room) or global (i.e., minimize overall travel time) strategies.

Iterative approaches, borrowing models and even tools from vehicular transportation simulation, propose to adopt a more coarse grained representation of the environment, i.e. a graph in which nodes are associated to intersections among road sections, but the process can be also adopted in buildings [Kretz *et al.*, 2014]. In this kind of scenario, pedestrians can start by adopting shortest paths on a first round of simulation: as suggested before, the fact that all pedestrians take the best path generally leads to congestion and sub-optimal travel times. Some selected pedestrians, especially those whose actual travel time differs significantly from the planned one, will change their planned path and a new simulation round will take place. The iteration of this process will lead to an equilibrium or even to system optimum, according to the adopted travel cost function [Lämmel *et al.*, 2009]. This iterative scheme has also been employed in multi-scale modeling approaches [Lämmel *et al.*, 2014; Crociani *et al.*, 2016].

The above approach naturally leads to consider that this kind of problem has been paid considerable attention in the field of Artificial Intelligence, in particular by the planning community. Hierarchical planning [Sacerdoti, 1974] approaches, in particular, provide an elegant and effective framework in which high level abstract tasks can be decomposed into low level activities. Despite the fact that the formulation of the approach date to the seventies, it is still widely considered and employed in the close area of computer graphics [Kapadia *et al.*, 2013], in which actions of virtual pedestrians are planned with the aim of being visually plausible and decided within real-time constraints. Within this framework, also issues related to the reconsideration of choices and plans were analyzed, mostly within the robotics area [Levihn *et al.*, 2013]. In the pedestrian simulation context, one could consider that microscopic decisions on the steps to be taken can follow a high-level definition of a sequence of intermediate destinations to be reached by the pedestrian. This kind of approach, which we experimentally investigated in [Crociani *et al.*, 2015], also allows exploiting already existing models dealing with low level aspects of pedestrian actions and perceptions.

The main issues in transferring AI planning results within this context of application, and more generally producing generally applicable contributions to the field, are partly due to the above suggested fundamental difference between the measures of success between *simulation* and *control* applications. Whereas the latter are targeted at *optimal* solutions, the former have to deal with the notions of *plausibility* and *validity*. Moreover, we are specifically dealing with a *complex system*, in which different and conflicting mechanisms are active at the same time (e.g. proxemics [Hall, 1966] and imitative behaviors [Helbing *et al.*, 1997]). Finally, whereas recent extensive observations and analyses (see, e.g., [Boltes and Seyfried, 2013]) produced extensive data that can be used to validate simulations within relatively simple scenarios (in which decisions are limited to basic choices on the regulation of mutual distances among other pedestrian while following largely common and predefined paths like corridors with unidirectional or bidirectional flows, corners, bottlenecks), we still lack comprehensive data on way-finding decisions.

## 3 A Model To Encompass the Pedestrian Movement and Route Choice






This Section will propose a multi-agent model designed for the simulation of pedestrian movement and route choice behavior. The model of agent is composed of two elements, respectively dedicated to the low level reproduction of the movement towards a target (i.e. the operational level, considering a three level model described in [Michon, 1985]) and to the decision making activities related to the next destination to be pursued (i.e. the route choice at the tactical level). The component dedicated to the operational level behavior of the agent is not extensively described since, for this purpose, the model described in [Bandini *et al.*, in press] has been applied. For a proper understanding of the approaches and mechanisms that will be defined at the tactical level, on the other hand, a brief description on the representation of the environ-

ment, with different levels of abstractions, is firstly provided in this Section. More attention will then be dedicated to the introduction and discussion of the model for the management of the route choice, which represents the main contribution of this paper.

### 3.1 The Representation of the Environment and the Knowledge of Agents

The adopted agent environment [Weyns *et al.*, 2007] is discrete and modeled with a rectangular grid of 40 cm sided square cells. The size is chosen considering the average area occupied by a pedestrian [Weidmann, 1993], and also respecting the maximum densities usually observed in real scenarios. The cells have a state that informs the agents about the possibilities for movement: each one can be vacant or occupied by obstacles or pedestrians (at most two, so as to be able to manage locally high density situations).

To allow the configuration of a pedestrian simulation scenario, several *markers* are defined with different purposes. This set of objects has been introduced to allow the movement at the operational level and the reasoning at the tactical level, identifying intermediate and final targets:

- *start areas* , places where pedestrians are generated: they contain information for pedestrian generation both related to the type of pedestrians (e.g. the distribution of their destinations), and to the frequency of generation;
- *openings* , sets of cells that divide, together with the obstacles, the environment into regions. These objects constitute the decision elements, intermediate destinations, for the route choice activities;
- *regions* , markers that describe the type of the region where they are located: with them it is possible to design particular classes of regions (e.g. stairs, ramps) and other areas that imply a particular behavior of pedestrians;
- *final destinations* , the ultimate targets of pedestrians;
- *obstacles* , non-walkable cells defining obstacles and non-accessible areas.

An example of environment annotated with this set of markers is proposed in Fig. 1(b). This model uses the *floor fields* approach [Burstedde *et al.*, 2001], using the agents' environment as a container of information for the management of the interactions between entities. In this particular model, discrete potentials are spread from cells of obstacles and destinations, informing about distances to these objects. The two types of floor fields are denoted as *path field*, spread from openings and final destinations (one per destination object), and *obstacle field*, a unique field spread from all the cells marked as obstacle. In addition, a *dynamic* floor field that has been denoted as *proxemic field* is used to reproduce a proxemic behavior [Hall, 1966] in a repulsive sense, letting the agents to maintain distances with other agents. This approach generates a plausible navigation of the environment as well as an anthropologically founded means of regulating interpersonal distances among pedestrians.

This framework, on one hand, enables the agents to have a position in the discrete environment and to perform movement towards a user configured final destination. On the other hand, the presence of intermediate targets allows choices at the tactical level of the agent, with the computation of a graph-like representation of the walkable space, based on the concept of *cognitive map* [Tolman, 1948]. The method for the computation of this environment abstraction has been defined in [Crociani *et al.*, 2014] and it uses the information of the scenario configuration, together with the floor fields associated to openings and final destinations. In this way a data structure for a complete knowledge of the environment is pre-computed. Recent approaches explore also the modeling of partial knowledge of the environment by agents (e.g. [Andresen *et al.*, in press]), but this aspect goes beyond the scope of the current work. The cognitive map identifies *regions* (e.g. a room) as nodes of the labeled graph and *openings* as edges. An example of the data structure associated to the sample scenario is illustrated in Fig. 1(c). Overall the cognitive map allows the agents to identify their position in the environment and it constitutes a basis for the generation of an additional knowledge base, which will enable the reasoning for the route calculation.

This additional data structure has been called *Paths Tree* and it contains the information about *plausible* paths towards a final destination, starting from each region of the environment. The concept of plausibility of a path is encoded in the algorithm for the computation of the tree, which is discussed in [Crociani *et al.*, 2015] and only briefly described here. The procedure starts by defining the destination as the root of the tree and it recursively adds child nodes, each of them mapped to an intermediate destination reachable in the region. Nodes are added if the constraints describing the plausibility of a path are satisfied: in this way, paths that imply cycles or a not reasonable usage of the space (e.g. passing inside a room to reach the exit of a corridor, as illustrated in Fig. 1(a)) are simply avoided.

The results of the computation is a tree whose nodes are mapped to targets in the environment and each edge refers to a particular path between two targets. The root of the tree is mapped to a final destination, while the underlying nodes are only mapped to openings. Hence, each branch from the root to an arbitrary node describes a *minimal* (i.e. plausible) path towards the final destination associated to the tree. To complete the information, each node  $n$  is labeled with the free flow travel time<sup>2</sup> associated to the path starting from the center of the opening associated to  $n$  and passing through the center of all openings mapped by the parent nodes of  $n$ , until the final destination. In this way, the agents know the possible paths through the environment and their respective estimated traveling times.

For the choice of their path, agents access the information of a Paths Tree generated from a final destination  $End$  with the function  $Paths(R, End)$ . Given the region  $R$  of the agent, the function returns a set of couples  $\{(P_i, tt_i)\}$ .  $P_i = \{\Omega_k, \dots, End\}$  is the ordered set describing paths

<sup>2</sup>The travel time that the agent can employ without encountering any congestion in the path, thus moving at its free flow speed.



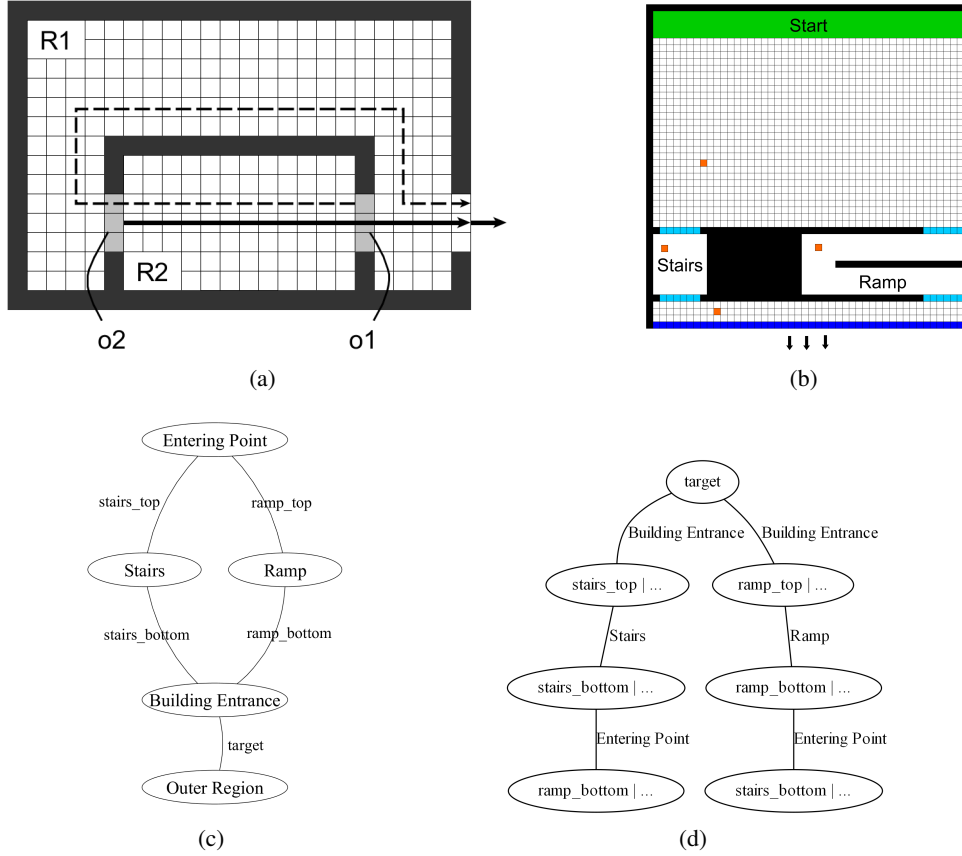


Figure 1: (a) An example of plausible (continuous line) and implausible (dashed) paths in a simple environment. (b) A simulation scenario with the considered annotation tools and its respective cognitive map (c) and the shortest path tree (d).

which start from  $\Omega_k$ , belonging to  $Openings(R)$ , and lead to  $End$ .  $tt_i$  is the associated free flow travel time.

### 3.2 The Route Choice Model of Agents

This aspect of the model is inspired by the behaviors observed in a experiment performed with human volunteers in November 2015 at the University of Tokyo, aiming at identifying basic behavior at the wayfinding level. The participants were put into a trade off scenario, since different paths were available but the shortest one was quickly congested. Empirical analysis related to this experiment are not presented in this paper for lack of space. Qualitatively, it has been observed that several persons preferred to employ a longer trajectories for achieving higher walking speed, but this kind of choice seemed to be taken more frequently and easily after a first *emerging leader* had performed it.

By considering these aspects, the objective is to propose an approach that would enable agents to choose their path considering distances as well as the evolution of the dynamics. At the same time, the model must provide a sufficient variability of the results (i.e. of the paths choices) and a calibration over possible empirical data.

The discussion of the model must starts with an overview of the agent life-cycle, in order to understand which activity

is performed and in which order. The workflow of the agent, encompassing the activities at operational and tactical level of behavior at each time-step, is illustrated in Figure 2.

First of all, the agent performs a perception of his situation considering his knowledge of the environment, aimed at understanding its position in the environment and the markers perceivable from its region (e.g. intermediate targets). At the very beginning of its life, the agent does not have any information about its location, thus the first assignment to execute is the *localization*. This task analyses the values of floor fields in its physical position and infers the location in the Cognitive Map. Once the agent knows the region where it is situated, it loads the Paths Tree and evaluates the possible paths towards its final destination.

The evaluation has been designed with the concept of *path utility*, assigned to each path to successively compute a probability to be chosen by the agent. The probabilistic choice of the path outputs a new intermediate target of the agent, used to update the reference to the floor field followed at the operational layer with the local movement.

The utility-based approach fits well with the needs to easily calibrate the model and to achieve a sufficient variability of the results.

The core functions of the wayfinding model are *Evaluate*



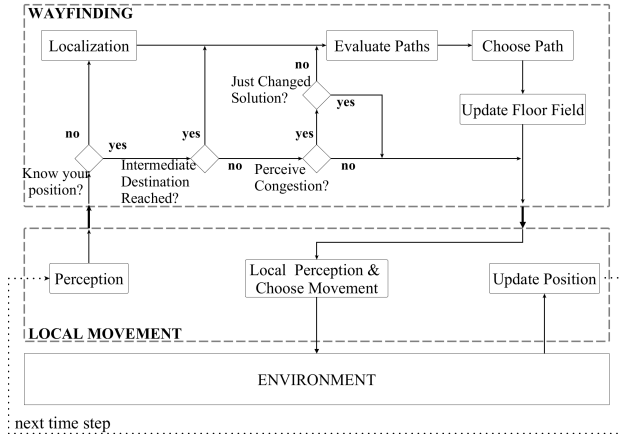


Figure 2: The life-cycle of the agent, emphasizing the two components of the model.

*Paths* and *Choose Paths*, which will be now discussed.

### The Utility and Choice of Paths

The function that computes the probability of choosing a path is exponential with respect to the utility value associated to it. This is completely analogous to the choice of movement at the operational layer:

$$Prob(P) = N \cdot e^{U(P)} \quad (1)$$

The usage of the exponential function for the computation of the probability of choosing a path  $P$  is a good solution to emphasize the differences in the perceived utility values of paths, limiting the choice of relatively bad solutions (that in this case would lead the agent to employ relatively long paths).  $U(P)$  comprises the three observed components influencing the route choice decision, which are aggregated with a weighted sum:

$$U(P) = \kappa_{tt} Eval_{tt}(P) - \kappa_q Eval_q(P) + \kappa_f Eval_f(P) \quad (2)$$

where the first element evaluates the expected travel times; the second considers the *queuing* (crowding) conditions through the considered path and the last one introduces a positive influence of perceived choices of nearby agents to pursue the associated path  $P$  (i.e. imitation of emerging leaders). All the three functions provide values normalized within the range  $[0, 1]$ , thus the value of  $U(P)$  is included in the range  $[-\kappa_q, \kappa_{tt} + \kappa_f]$ .

In theory, there is no best way to define these three components: the usage of very simple functions as well as complicated ones might provide the same quality to the model. The only way to evaluate the reliability of this model, in fact, is with a validation procedure over some empirical knowledge. Hence, these three mechanisms have been designed with the main objective to allow the calibration over empirical datasets, preferring the usage of simple functions where possible.

### The Evaluation of Traveling Times

The evaluation of traveling times is a crucial element of the model. First of all, the information about the travel time  $tt_i$  of a path  $P_i$  is derived from the Paths Tree with  $Paths(R, End)$  (where  $End$  is the agent's final destination, used to select the appropriate Paths Tree, and  $R$  is the region in which the agent is situated and it is used to select the relevant path  $P_i$  in the Paths Tree structure) and it is integrated with the free flow travel time to reach the first opening  $\Omega_k$  described by each path:

$$TravelTime(P_i) = tt_i + \frac{PF_{\Omega_k}(x, y)}{Speed_d} \quad (3)$$

where  $PF_{\Omega_k}(x, y)$  is the value of the path field associated to  $\Omega_k$  in the position  $(x, y)$  of the agent and  $Speed_d$  is the *desired velocity* of the agent, that can be an arbitrary value (see [Bandini *et al.*, in press] for more details of this aspect of the model). The value of the traveling time is then evaluated by means of the following function:

$$Eval_{tt}(P) = N_{tt} \cdot \frac{\min_{P_i \in Paths(r)} (TravelTime(P_i))}{TravelTime(P)} \quad (4)$$

where  $N_{tt}$  is the normalization factor, i.e., 1 over the sum of  $TravelTime(P)$  for all paths. By using the minimum value of the list of possible paths leading the agent towards its own destination from the current region, the range of the function is set to  $(0, 1]$ , being 1 for the path with minimum travel time and decreasing as the difference with the other paths increases. This modeling choice, makes this function describe the *utility* of the route in terms of travel times, instead of its *cost*.

This design is motivated by the stability of its values with the consideration of relatively long path, which might be represented in the simulation scenario. By using a cost function, in fact, the presence of very high values of  $TravelTime(P)$  in the list would flatten the differences among cost values of other choices after the normalization: in particular, in situations in which most relevant paths have relatively similar costs, excluding a few outliers (even just one), the normalized cost function would provide very similar values for most sensible paths, and it would not have a sufficient discriminating power among them.

### The Evaluation of Congestion

The behavior modeled in the agent in this model considers congestion as a negative element for the evaluation of the path. This does not completely reflect the reality, since there could be people who could be attracted by congested paths as well, showing a mere *following* behavior. On the other hand, by acting on the calibration of the parameter  $\kappa_q$  it is possible to define different classes of agents with customized behaviors, also considering attraction to congested paths with the configuration of a negative value.

For the evaluation of this component of the route decision making activity associated to a path  $P$ , a function is first introduced for denoting agents  $a'$  that precede the evaluating agent  $a$  in the route towards the opening  $\Omega$  of a path  $P$ :

$$Forward(\Omega, a) = |\{a' \in Ag \setminus \{a\} : Dest(a') = \Omega \wedge PF_{\Omega}(Pos(a')) < PF_{\Omega}(Pos(a))\}| \quad (5)$$

where  $Pos$  and  $Dest$  indicates respectively the position and current destination of the agent; the fact that  $PF_{\Omega}(Pos(a')) < PF_{\Omega}(Pos(a))$  assures that  $a'$  is closer to  $\Omega$  than  $a$ , due to the nature of floor fields. Each agent is therefore able to perceive the main direction of the others (its current destination). This kind of perception is plausible considering that only preceding agents are counted, but we want to restrict its application when agents are sufficiently close to the next passage (i.e. they perceive as important the choice of continuing to pursue that path or change it). To introduce a way to calibrate this perception, the following function and an additional parameter  $\gamma$  is introduced:

$$PerceiveForward(\Omega, a) = \begin{cases} Forward(\Omega, a), & \text{if } PF_{\Omega}(Pos(a)) < \gamma \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

The function  $Eval_q$  is finally defined with the normalization of  $PerceiveForward$  values for all the openings connecting the region of the agent:

$$Eval_q(P) = \frac{PerceiveForward(FirstEl(P), myself)}{N \cdot width(FirstEl(P))} \quad (7)$$

where  $FirstEl$  returns the first opening of a path,  $myself$  denotes the evaluating agent and  $width$  scales the evaluation over the width of the door (larger doors sustain higher flows).

### Propagation of Choices - Following Behavior

This component of the decision making model aims at representing the effect of an additional stimulus perceived by the agents associated to sudden decision changes of other persons that might have an influence. An additional grid has been introduced to model this kind of event, whose functioning is similar to the one of a dynamic floor field. The grid, called *ChoiceField*, is used to spread a gradient from the positions of agents that, at a given time-step, change their plan due to the perception of congestion.

The functioning of this field is described by two parameters  $\rho_c$  and  $\tau_c$ , which defines the diffusion radius and the time needed by the values to *decay*. The diffusion of values from an agent  $a$ , choosing a new target  $\Omega'$ , is performed in the cells  $c$  of the grid with  $Dist(Pos(a), c) \leq \rho_c$  with the following function:

$$Diffuse(c, a) = \begin{cases} 1/Dist(Pos(a), c) & \text{if } Pos(a) \neq c \\ 1 & \text{otherwise} \end{cases} \quad (8)$$

The diffused values persist in the *ChoiceField* grid for  $\tau_c$  simulation steps, then they are simply discarded. The index of the target  $\Omega'$  is stored together with the diffusion values, thus the grid contains in each cell a vector of couples  $\{(\Omega_m, diff_{\Omega_m}), \dots, (\Omega_n, diff_{\Omega_n})\}$ , describing the values of

influence associated to each opening of the region where the cell is situated. While multiple neighbor agents changes their choices towards the opening  $\Omega'$ , the values of the diffusion are summed up in the respective  $diff_{\Omega'}$ . In addition, after having changed its decision, an agent spreads the gradient in the grid for a configurable amount of time steps represented by an additional parameter  $\tau_a$ . In this way it influences the choices of its neighbors for a certain amount of time.

The existence of values  $diff_{\Omega_k} > 0$  for some opening  $\Omega_k$  implies that the agent is influenced in the evaluation phase by one of these openings, but the probability for which this influence is effective is, after all, regulated by the utility weight  $\kappa_f$ . In case of having multiple  $diff_{\Omega_k} > 0$  in the same cell, an individual influence is chosen with a simple probability function based on the normalized weights  $diff$  associated to the cell. Hence, for an evaluation performed by an agent  $a$  at time-step  $t$ , the utility component  $Eval_f$  can be equal to 1 only for one path  $\bar{P}$ , between the paths having  $diff_{\Omega_k} > 0$  in the position of  $a$ .

## 4 Evaluation of the Model

The evaluation of the model is here discussed with a simulation of a large scenario, with the aim of verifying the behavior of the model in a real-world environment and to perform a qualitative comparison of the results with another wayfinding model from the literature.

All the presented results have been achieved with the calibration weights of the utility function configured as  $\Omega_{tt} = 100, \Omega_q = 27; \Omega_f = 5$ , while the parameters related to the *ChoiceField* are set to  $\rho_c = 1.2m, \tau_c = 2$  time-steps =  $0.44s$  and  $\tau_d = 4$  time-steps =  $1s$ . The desired speed of agents have been configured with a normal distribution centered in  $1.4$  m/s and with standard deviation of  $0.2$  m/s, in accordance with the pedestrians speeds usually observed in the real world (e.g. [Willis *et al.*, 2004]). The distribution is discretized in classes of  $0.1$  m/s, and cut by configuring a minimum velocity of  $1.0$  m/s and a maximum one of  $1.8$  m/s (see the blue boxes in Fig. 3(c)). To allow a maximum speed of  $1.8$  m/s —considered plausible in this *outflow* scenario—the time-step duration is assumed to  $\bar{\tau} = 0.22s$ .

The simulation scenario describes the outflow from a portion of the Düsseldorf Arena, as described in [Wagoum *et al.*, 2012]. The annotated environment used for the simulation with the discussed model is illustrated in Fig. 3(a): 4 starting areas models the bleachers of the stadium and generates the agents in the simulation, whose aim is to reach the outside area indicated with the blue object. Cyan objects are the intermediate targets describing the wayfinding decisions of agents. 250 agents are generated at the beginning of the simulation from each start area, producing a total of 1000 pedestrians.

The heat map shown in Figure 3(b) provides information about the usage of the space during the simulation, by describing the average local densities perceived by the agents (so-called *cumulative mean density*). The major congested areas are located in front of the exit doors, given their relatively small width of  $1.2$  m. An interesting point that comes out from this analysis (also visible in the screen-shot in Fig. 3(a)) is that the present configuration of the environment implies

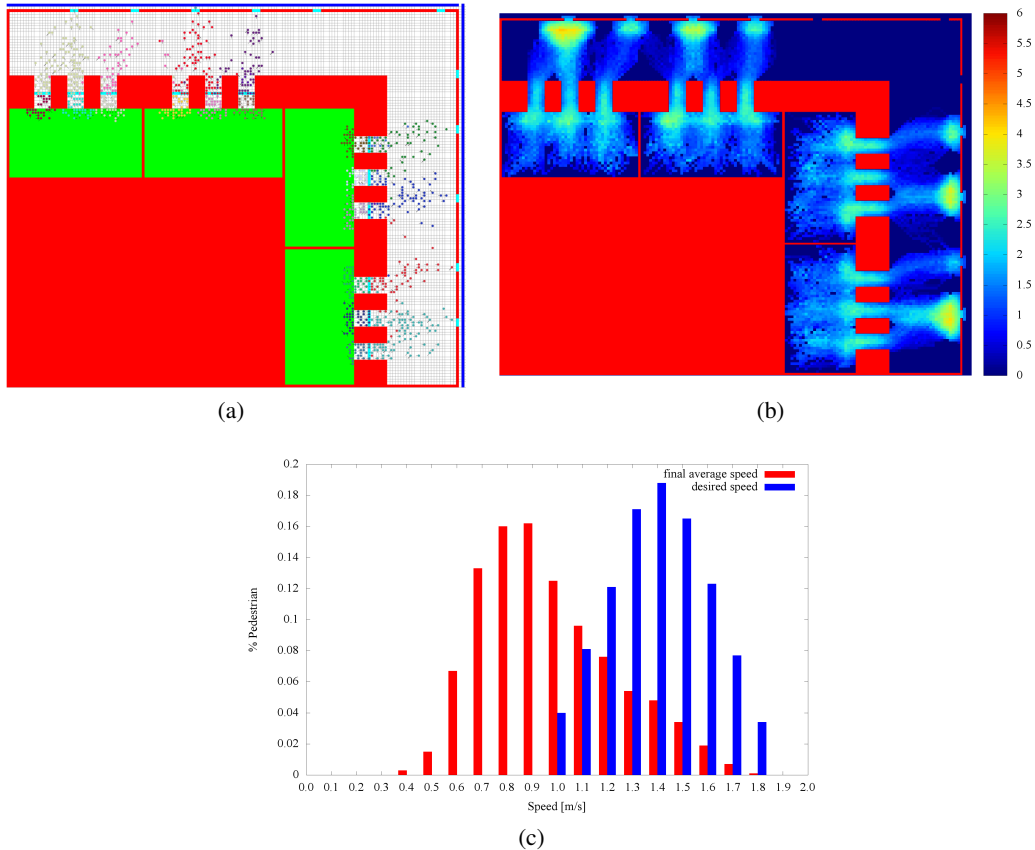


Figure 3: (a) A screenshot of the simulation of the Düsseldorf Arena. Spatial markers are also displayed and the colors of the agents identifies their current target. (b) Cumulative mean density map and (c) average speed distributions configured (blue) and achieved (red).

that several exits receive an incoming flow from more sources (i.e. corridors), while there are 3 exits in the upper right corner of the environment which are not employed at all by the agents during the simulation. In addition, the usage of the exits is unbalanced, causing the level of density to be higher in some of them. The evaluation of this evidence would require empirical data that could be used either to support the modeling choices or to confute these results and lead to a different calibration (e.g. adopting a lower weight for the consideration of travel time, that would lead to an increased usage of the far exits).

The corridors connecting each bleacher to the atrium are affected as well by high densities (around  $2.5\text{--}3$  persons/ $\text{m}^2$ ) but their widths guarantee a sensibly higher flow, causing smoother congestion—and so higher speeds—inside the starting regions.

The red boxes of Fig. 3(c) shows the distribution of desired walking speeds compared to the achieved average walking speeds of agents during the simulation. The congestion arisen in the exit doors of the atrium sensibly affected the travel time of the agents. This caused that a small portion of the simulated population succeeded in maintaining its desired speed (the agents generated in positions closer to the exit), while

most of them experienced a significant delay during their way.

## 5 Conclusions

The present paper has introduced a general model for decision making activities related to pedestrian route choices. The model encompasses three aspects influencing these choices, as observed in an experimental observation: expected travel time, perceived level of congestion on the chosen path, and decisions of other preceding pedestrian to pursue a different path. Achieved results are both plausible and encouraging, though a proper validation of the model would require additional results but also the acquisition of empirical evidences on human wayfinding decisions in congested situations.

## References

- [Andresen *et al.*, in press] Erik Andresen, David Haensel, Mohcine Chraïbi, and Armin Seyfried. Wayfinding and cognitive maps for pedestrian models. In *Proceedings of Traffic and Granular Flow 2015 (TGF2015)*. Springer, (in press).
- [Bandini *et al.*, in press] Stefania Bandini, Luca Crociani, and Giuseppe Vizzari. An approach for managing het-

- erogeneous speed profiles in cellular automata pedestrian models. *Journal of Cellular Automata*, (in press).
- [Boltes and Seyfried, 2013] Maik Boltes and Armin Seyfried. Collecting pedestrian trajectories. *Neurocomputing*, 100:127–133, jan 2013.
- [Burstedde *et al.*, 2001] C Burstedde, K Klauck, A Schadschneider, and J Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3–4):507–525, 2001.
- [Challenger *et al.*, 2009] Rose Challenger, Chris W Clegg, and Mark A Robinson. Understanding Crowd Behaviours: Supporting Evidence. Technical report, University of Leeds, 2009.
- [Crociani *et al.*, 2014] Luca Crociani, Alberto Invernizzi, and Giuseppe Vizzari. A hybrid agent architecture for enabling tactical level decisions in floor field approaches. *Transportation Research Procedia*, 2:618–623, 2014.
- [Crociani *et al.*, 2015] Luca Crociani, Andrea Piazzoni, Giuseppe Vizzari, and Stefania Bandini. When reactive agents are not enough: Tactical level decisions in pedestrian simulation. *Intelligenza Artificiale*, 9(2):163–177, 2015.
- [Crociani *et al.*, 2016] Luca Crociani, Gregor Lämmel, and Giuseppe Vizzari. Multi-scale simulation for crowd management: a case study in an urban scenario. In *Proceedings of the 1st Workshop on Agent Based Modelling of Urban Systems (ABMUS 2016)*, 2016.
- [Guo and Huang, 2011] Ren-Yong Guo and Hai-Jun Huang. Route choice in pedestrian evacuation: formulated using a potential field. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(04):P04018, 2011.
- [Hall, 1966] Edward T. Hall. *The hidden dimension*. Doubleday New York Ed., 1966.
- [Helbing *et al.*, 1997] Dirk Helbing, Frank Schweitzer, Joachim Keltsch, and Péter Molnár. Active Walker Model for the Formation of Human and Animal Trail Systems. *Physical Review E*, 56(3):2527–2539, 1997.
- [Kapadia *et al.*, 2013] Mubbasir Kapadia, Alejandro Beacco, Francisco M. Garcia, Vivek Reddy, Nuria Pelechano, and Norman I. Badler. Multi-domain real-time planning in dynamic environments. In *The ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA '13, Anaheim, CA, USA, July 19-21, 2013*, pages 115–124. ACM, 2013.
- [Kretz *et al.*, 2014] Tobias Kretz, Karsten Lehmann, Ingmar Hofsaß, and Axel Leonhardt. Dynamic Assignment in Microsimulations of Pedestrians. *Annual Meeting of the Transportation Research Board, 14-0941 (2014)*, pages 14–0941, jan 2014.
- [Lämmel *et al.*, 2009] Gregor Lämmel, Hubert Klüpfel, and Kai Nagel. The MATSim network flow model for traffic simulation adapted to large-scale emergency egress and an application to the evacuation of the Indonesian City of Padang in case of a tsunami warning, pedestrian behavior. *Pedestrian behavior: Models, Data Collection and Applications*, pages 245—265, 2009.
- [Lämmel *et al.*, 2014] Gregor Lämmel, Armin Seyfried, and Bernhard Steffen. Large-scale and microscopic: a fast simulation approach for urban areas. In *Transportation Research Board 93rd Annual Meeting*, number 14-3890, 2014.
- [Leviñh *et al.*, 2013] Martin Leviñh, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Mike Stilman. Foresight and reconsideration in hierarchical planning and execution. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 224–231. IEEE, 2013.
- [Michon, 1985] John A. Michon. A Critical View of Driver Behavior Models: What Do We Know, What Should We Do? In Leonard and Evans and Richard C. Schwing, editors, *Human Behavior and Traffic Safety*, pages 485–524. Springer US, 1985.
- [Sacerdoti, 1974] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artif. Intell.*, 5(2):115–135, 1974.
- [Sassi *et al.*, 2015] Andrea Sassi, Claudio Borean, Roberta Giannantonio, Marco Mamei, Dario Mana, and Franco Zambonelli. Crowd steering in public spaces: Approaches and strategies. In 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, Liverpool, United Kingdom, October 26-28, 2015, pages 2098–2105. IEEE, 2015.
- [Tolman, 1948] Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189–208, 1948.
- [Vizzari and Bandini, 2013] Giuseppe Vizzari and Stefania Bandini. Studying pedestrian and crowd dynamics through integrated analysis and synthesis. *IEEE Intelligent Systems*, 28(5):56–60, 2013.
- [Wagoum *et al.*, 2012] A. U. Kemloh Wagoum, A. Seyfried, and S. Holl. Modelling dynamic route choice of pedestrians to assess the criticality of building evacuation. *Advances in Complex Systems*, 15(07):15, 2012.
- [Weidmann, 1993] Ulrich Weidmann. Transporttechnik der Fussgänger - Transporttechnische Eigenschaftendes Fussgängerverkehrs (Literaturstudie). Literature Research 90, Institut fuer Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau IVT an der ETH Zürich, 1993.
- [Weyns *et al.*, 2007] Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents Multi-Agent Systems*, 14(1):5–30, 2007.
- [Willis *et al.*, 2004] Alexandra Willis, Nathalia Gjersoe, Catriona Havard, Jon Kerridge, and Robert Kukla. Human movement behaviour in urban spaces: implications for the design and modelling of effective pedestrian environments. *Environment and Planning B: Planning and Design*, 31(6):805–828, 2004.

# Pedestrians' Route Choice Model for Shopping Behavior

Bruno Rocha Werberich Carlos Oliva Pretto Helena Beatriz Bettella Cybis

Department of Production Engineer

Federal University of Rio Grande do Sul

Porto Alegre, Brazil

{bruno.rwe;cpretto}@gmail.com , helenabc@producao.ufrgs.br

## Abstract

This paper presents an agent-based model to address the pedestrian route choice problem in shopping malls. Route choice in shopping malls may be defined by a number of causal factors. Shoppers may follow a pre-defined schedule, they may be influenced by other people walking, or may want to get a glimpse of a familiar shopping. The route choice process assumes that the cost of each route can be calculated as a function of three factors: route length, impedance generated by other pedestrians and attraction for areas of interest on the environment. The impedance generated by the friction between pedestrians is assumed to exist even before physical contact, due to the psychological tendency to avoid passing close to individuals with high relative velocity. Pedestrians seek minimal route length and minimal friction with other pedestrians. In order to represent shopping areas environments, a new factor is being considered in the calculation of the route cost: the attraction for areas of interest on the environment. Simulation results were compared to real data collected by video recording in a shopping mall.

## 1 Introduction

Modelling of pedestrian's behavior is a complex task and has been studied by different research areas. In order to represent motion of pedestrians more realistically, models are required to simulate several processes, including sense and avoidance of obstacles, interaction with other pedestrians and route choice. Agent-based abstraction has been widely used for pedestrian modeling, mainly due to its capacity to provide insights about system's reactions from changes on entities properties, capturing information over space and time at a detailed level [Klühl and Bazzan 2012; Macal et al. 2006; Rossetti R. et. al. 2002]. Agent-based models represent agents' decision-making ability based on their profile and perception over the environment.

Agent-based pedestrians models require the aggregation of different levels of abstraction, that are modeled on different layers. The majority of pedestrian models present a multi-layer simulation approach [Gaud et al. 2008; Hoogendoorn

et al. 2002] composed by, at least, two layers: a tactical and an operational layer.

The tactical layer chooses a path regarding an origin-destination pair and a route choice criteria such as minimum distance and/or travel times. The tactical model determines the desired pedestrian directions, which are used in the operational model [Pretto et al. 2011].

The operational model determines the low level microscopic movements of pedestrians. It is ruled by principles of pedestrians' sense and avoidance of obstacles. Most models reported in literature can be regarded as using force-based approaches [Helbing et al. 1991; Helbing et al. 1995]. In force-based models, agents evaluate forces exerted by infrastructure and by other agents. Helbing and Molnar (1995) presented a relevant work on force-based models in which they use Newtonian mechanics and a continuous space representation to model a long-range interaction. The concept behind this approach suggests that the motion of a pedestrian can be described by combination of several forces (including the repulsive forces from walls and other pedestrians). The social force model reproduces various emergent phenomena observed on pedestrian dynamics.

The tactical model is responsible for route choice. Realistic route choice is a complex process because most route selection strategies are based on subconscious decisions. Most models presented in the literature are concerned only with the quickest or shortest route, like Kirik et. al. (2009), Dressler et. al. (2010) and Lämmel et. al. (2014). However, other factors play an important role in route choice behavior, such as: peoples' habits, number of crossings, pollution and noise levels, safety, shelter from poor weather conditions and other environment stimulations [Papadimitriou E., 2012]. Most relevant route choice models are concerned with pedestrians' evacuation. In Kretz et. al. (2011), for instance, pedestrians routes are chosen based on the minimal remaining travel time to destination. Kretz et. al. (2014) introduce a generic method for dynamic assignment used with microsimulation of pedestrian dynamics. In the paper, the routes mark the most relevant routing alternatives in any given walking geometry, reducing the infinitely many trajectories by which a

pedestrian can move from origin to destination to a small set of routes. Crociani and Lämmel (2016) present a work with two major topics. In the first topic, a novel cellular automaton (CA) model is proposed, which describes the pedestrian movement by a set of simple rules, and the second topic describes how the CA can be integrated into an iterative learning cycle where the individual pedestrian can adapt travel plans based on experiences from previous iterations. Patil et. al. (2010) propose an interactive algorithm to direct and control crowd simulation. The model presented by Treuille et. al., (2006) unifies route planning and local collision avoidance by using a set of dynamic potential and velocity. Teknomo (2008) and Teknomo et al., (2008) described a self-organization route choice approach to model the dynamics of agents, such as pedestrians and cars on a simple network graph. The agents decide, when reaching a vertex, which edge to enter next. This decision is based on a set of rules regarding the agent's observation of the local environment. In order to represent complex networks, such as shopping areas and urban scenarios, agents need to represent more complex characteristics and capabilities.

The literature presents several agent-based applications to simulate different pedestrians' behaviors and environments. The pedestrians' simulation in a commercial environment, such as shopping malls, is particularly complex since pedestrians are exposed to different stimulus and attractions [Wang, W. et. al. 2014]. Agent-based simulation is particularly valuable for these cases because environment stimulus exert distinct influences depending on the person profile. Dijkstra et al., (2013) provide a model for pedestrian activity simulations in shopping environments. This framework provides an activity agenda for pedestrian agents, guiding their shopping behavior in terms of destination and time spent in shopping areas. Pedestrian agents need to successively visit a set of stores and move over the network. The authors assumed that pedestrian agents' behavior is driven by a series of decision heuristics. Agents need to decide which stores to choose, in what order and which route to take, subject to time and environment constraints.

Route choice in shopping malls may be defined by a number of causal factors. Shoppers may follow a pre-defined schedule, they may be influenced by other people walking, or may want to get a glimpse of a familiar shopping.

Shopping agents, as described in the literature [Borgers, A., and Timmermans, H., 1986; Ali, W. and Moulin, B., 2006] usually decide (i) in which stores to stop, (ii) in what order and (iii) which route to take. In practice, however, shopping mall users' behaviour is a combination of planned and unplanned decisions. Planned decisions can be defined by a set of origin-destination pairs. Unplanned decisions may be resultant from eventual impulses or the attraction exerted by shopping windows.

This paper presents an agent-based route choice model to represent pedestrians' in a shopping mall environment. The pedestrian model allows the representation of shopping users capable to perform either planned and unplanned behaviour, depending on the agent's profile. Simulation results were compared to real data collected by video recording in a shopping mall.

## 2 The Model

An agent-based model is proposed to address pedestrian route choice problem. Agent-based models represent agents' decision-making ability based on agents' characteristics profile and perception over the environment. In the proposed model, pedestrians are agents able to choose and recalculate routes. Pedestrians are not assigned to predetermined routes.

In this model, a route is a set of coordinates followed by a pedestrian from origin to destination. Route choice process comprises three factors for calculation: (i) distance, (ii) interaction with other pedestrians (avoiding jams) and (iii) attraction for areas of interest on the environment (in this specific case: shop windows).

The framework adopted to describe pedestrian behavior in this model (Figure 1) presents a three-layer structure, each layer representing:

- (i) Demand for travel - set of origin and destination. Each origin-destination pair is associated to a number of trips and a pedestrian generation rate. Origins and destinations are associated with nodes on the environment layer.
- (ii) Simulation environment structure - The environment is described as a continuous space and is composed by geometric entities, such as rooms, doors, and other obstacles. The environment entities are linked by a graph-based structure providing a route to all entities. In this model, nodes are defined by a set of coordinates (x, y). Nodes also contain properties defining local features of the environment.
- (iii) Pedestrians movement, sense and avoidance of obstacles: set of equations and agents behavior rules. The social force model (1) describes pedestrian walking behavior regarding agents' low-level motion, collision avoidance and velocity adaptation. Pedestrians freely walk on the modeling environment seeking the next graph node of the designated route. Pedestrians' movements are ruled by the sense and avoidance model and are not restricted to a strict set of links.

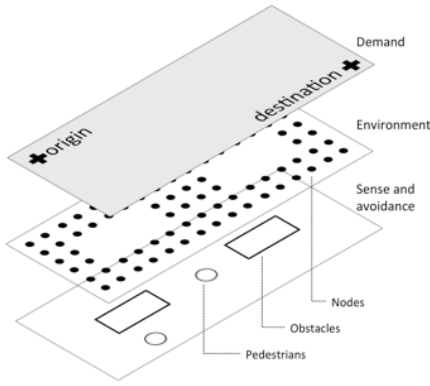


Figure 1 - Layers

## 2.1 The Route Choice Process

The presented route choice process is derived from a model established by Werberich et al. (2014). Werberich et al. propose that the cost of each route can be calculated as a function of two factors: route length and the impedance generated by other pedestrians. The impedance generated by the friction between pedestrians is assumed to exist even before physical contact, due to the psychological tendency to avoid passing close to individuals with high relative velocity [Helbing D. et al., 2000]. Pedestrians seek minimal route length and minimal friction with other pedestrians. In this model, a new factor is being considered in route cost calculation: attraction for areas of interest on the environment.

The total route cost is the sum of all link costs. Dijkstra algorithm [Dijkstra E., 1959] is adopted to generate valid routes for any origin/destination pair. Figure 2 describes the cost calculation for a link.

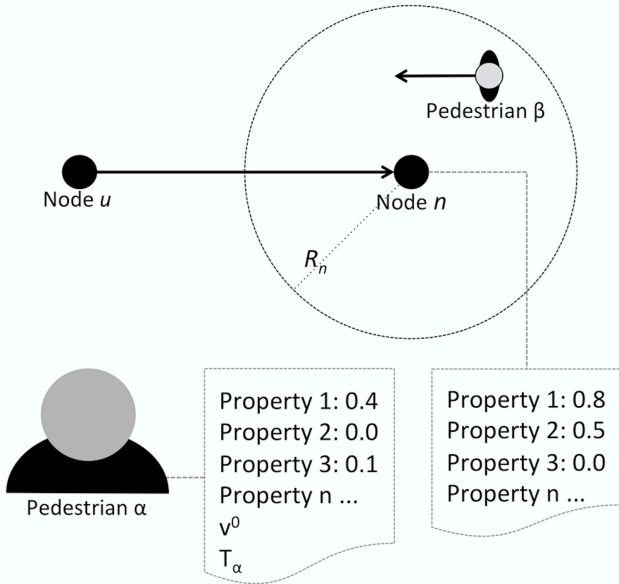


Figure 2 – Pedestrian's profile and node attraction

Figure 2 presents the elements involved in the route choice process. The cost estimation for a Pedestrian  $\alpha$  to walk from node  $u$  to  $n$  involves three factors: (i) the distance between nodes ( $\|\vec{r}_n - \vec{r}_u\|$ ), (ii) the impedance perceived by the pedestrian  $\alpha$  exerted by other pedestrians ( $I_\alpha$ ) and (iii) the environment attraction perceived by pedestrian  $\alpha$  for the node  $n$  ( $A_\alpha^n$ ).

Impedance exerted by the pedestrians in the simulation is calculated by simple vector operations. Subtracting the desired velocity of pedestrian  $\alpha$  from the velocity of pedestrians closer to node  $n$  (pedestrians  $\beta$ ) it is possible to estimate  $I_\alpha$  (equation 1).

$$I_\alpha = \sum_{\beta} \left| \vec{v}_\beta - \left( \frac{\vec{r}_n - \vec{r}_u}{\|\vec{r}_n - \vec{r}_u\|} \right) * v_\alpha^0 \right| \quad (1)$$

where:

$\vec{v}_\beta$  = Pedestrian's  $\beta$  current velocity;

$\vec{r}_n$  = Node's  $n$  vector position;

$\vec{r}_u$  = Node's  $u$  vector position ;

$v_\alpha^0$  = Pedestrian's  $\alpha$  desired speed.

The calculation of  $I_\alpha$  considers a neighborhood area around the node  $n$ , defined by the radius  $R_n$ . All pedestrians inside the neighborhood area, at the instant of the route choice, are nominated pedestrians  $\beta$ .  $I_\alpha$  is the sum of the friction forces exerted by each pedestrian  $\beta$  over the desired velocity of the pedestrian  $\alpha$ .

As mentioned above, the graph nodes contain properties that classify local features of the environment. Node properties define the environment characteristics. For example, properties can be defined as female clothes store, male clothes store, electronics store, shoe store, etc. Nodes are defined by a set of values for all simulated properties. Higher properties values mean the node is closer of the related feature. Properties can assume values in the range  $[0 - 1]$ .

The attraction exerted by these nodes properties on pedestrians vary depending on pedestrians profiles. Pedestrians' profiles also present a set of values for all simulated environment properties, that represent their attraction for these features. For example, male pedestrians probably have higher values for a property relating to a male clothes store. These properties also assume values in the range  $[0 - 1]$ .

The attraction of node  $n$ , perceived by pedestrian  $\alpha$  ( $A_\alpha^n$ ), is calculated as a weighted average (Equation 2):

$$A_\alpha^n = \frac{\sum_{i=0}^p P_i^\alpha * N_i^n}{\sum_{i=0}^p N_i^n} \quad (2)$$

where:



$p$  = total number of properties;  
 $P_i^\alpha$  = pedestrian  $\alpha$  property  $i$  value;  
 $N_i^n$  = node  $n$  property  $i$  value.

The total estimated cost for pedestrian  $\alpha$  to walk from node  $u$  to  $n$  ( $W_\alpha^{u,n}$ ), is a balance between distance, impedance and attractiveness, as described in Equation 3:

$$W_\alpha^{u,n} = \|\vec{r}_n - \vec{r}_u\|. (1 + I_\alpha / I_{\max} + (1 - A_\alpha^n)) \quad (3)$$

where:

$I_{\max}$  = settable parameter that adjusts the balance between distance and impedance. Further description of this parameter can be obtained in Werberich et al. (2014).

Elected routes minimize the total cost  $W_\alpha$ . Equation 3 ensures pedestrians are attracted to areas of interest considering their profile. Pedestrians also avoid congested areas and passing close to other pedestrians with high relative velocity.

## 2.2 Pedestrian Stopping Behavior

It is expected that pedestrians walking on shopping environment, when attracted by an environmental stimulus, may stop for a while. For example, pedestrians attracted by a shop window frequently stop walking when they get closer to this interest point. This model simulates pedestrians route choice process subjected to attraction by interest areas, typical of shopping environments.

To simulate pedestrians' stopping behavior the model introduces the concept of hotspots. Hotspots are defined by a location on the environment ( $x$  and  $y$  coordinates) and a neighborhood area (radius  $R$ ). Hotspots have the same environment properties as graph nodes. When a pedestrian reaches the neighborhood area of a hotspot, he decides whether to stop or not. This decision process considers the pedestrian profile and the hotspot properties. Pedestrian profile includes a value denoting the tendency to stop on a hotspot ( $T_\alpha$ ). Higher values of  $T_\alpha$  means the pedestrian have higher tendency to stop on hotspots.  $T_\alpha$  values also respect the range  $[0-1]$ . Equation 4 defines the probability of a pedestrian  $\alpha$  stopping on a hotspot  $q$  ( $S_\alpha^q$ ).

$$S_\alpha^q = \frac{\sum_{i=0}^p (P_i^\alpha * H_i^q)}{\sum_{i=0}^p H_i^q} * T_\alpha \quad (4)$$

where:

$p$  = total number of properties;  
 $P_i^\alpha$  = pedestrian  $\alpha$  property  $i$  value;  
 $H_i^n$  = hotspot  $q$  property  $i$  value;  
 $T_\alpha$  = pedestrian  $\alpha$  tendency to stop on a hotspot.

If a pedestrian decides to stop on a hotspot neighborhood, the hotspot coordinates become his new destination for the stopping period. The balance between the pedestrian desired

speed vector ( $v_\alpha^0$ ) and the forces exerted by the hotspot walls, keep the pedestrian standing in the neighborhood area. During this period, the interaction between pedestrians is maintained, allowing a realistic representation of pedestrians behavior at window shops. When a pedestrian stopping time has expired, a new route is recalculated to the final the destination.

The time a pedestrian stops at a hotspot may has variable assumptions. In this formulation, pedestrians stopping time is assumed to be fixed, equal to 20 seconds. Assumptions about stopping times can be discussed in more detail. An important work regarding time spent at store windows was developed by Dijkstra J. et. al. (2014). In this paper, authors describe the time spent in a store based on pedestrians profile and store segment.

Figure 3 presents a flowchart of the agent's internal process.

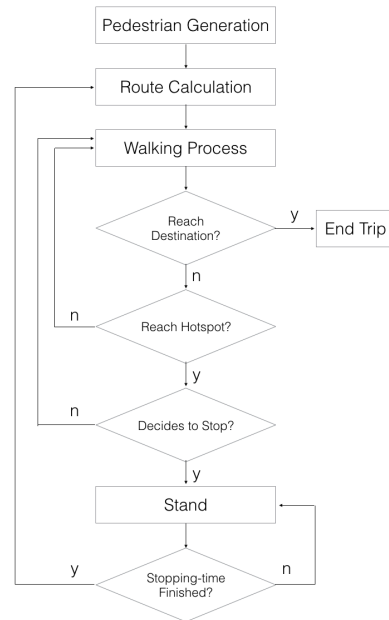


Figure 3 - Agent's internal process

As presented in this flowchart, a pedestrian only performs a route recalculation after stopping at a hotspot. A Social Force-based route choice process considers the interaction with other pedestrians, which provides a dynamic behavior. However, if necessary, when simulating complex scenarios, the model structure allows the introduction of route recalculation areas. When simulating small scenarios, where the decision at the beginning of the trip was based on a good assessment of the way forward for all simulation timeframe, route recalculation may not be necessary.



### 3 Collected Data

Video data were collected in a shopping mall of Porto Alegre, Brazil. The camera collected images from a hall that connects the two main corridors of the first floor. Figure 4 presents an image of the studied area and the collected pedestrian routes.

The software *Tracker* was used to collect pedestrians' data in a semi-automatic process. The collected data is composed by a set of coordinates ( $x$  and  $y$ ) over 1 minute of video for each pedestrian.

In order to simplify the data analysis, the environment was segmented in cells. A color map representing the cumulative occupation of each cell is shown at figure 5, segmented by gender.



Figure 4 – The Mall

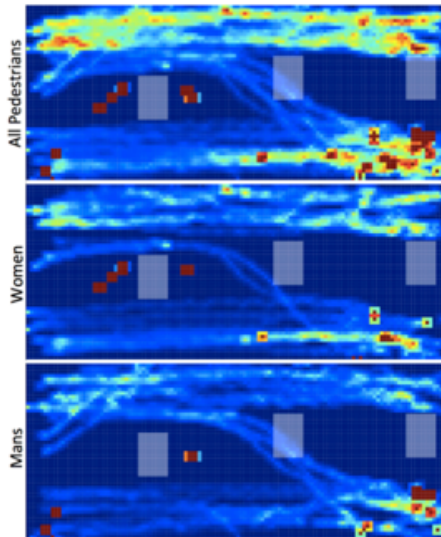


Figure 5 –Collected data

Data analysis allows the identification of three stores with higher pedestrian attraction. Table 1 shows the number of pedestrians, men (M) and women (W), that were attracted and stopped closer to these areas.

Table 1 – Stopped pedestrians

Store	M	W
jewelry	1	5
toy store	3	2
shoes store	2	3

### 4 Simulation

The proposed model has the potential to represent several properties regarding agents' profile and environment characteristics. In order to simplify the simulation, only two properties were considered in this experiment: Male Store Attraction ( $MSA_s$ ) and Female Store Attraction ( $FSA_s$ ). These two properties were applied to:

- i. Scenario elements: hotspots and graph nodes ( $MSA_s$  and  $FSA_s$ );
- ii. Agents ( $MSA_a$  and  $FSA_a$ ).

The experiment was developed to identify the influence of  $MSA_a$  and  $FSA_a$  in the number of pedestrians that are attracted to hotspots. The  $MSA_a$  and  $FSA_a$  were calibrated based on collected data.

The model was implemented using *c#* programming language (simulation engine) and Windows Presentation Foundation for the graphical interface.

#### 4.1 Simulation Scenario

Figure 6 shows the simulation scenario built to represent the observed environment. Green areas ( $h1$ ,  $h2$ ,  $h3$ ) are the hotspots. The hotspots correspond to stores where mall users used to stop on the real site. Dots are the graph nodes. Rectangles represent mall kiosks.

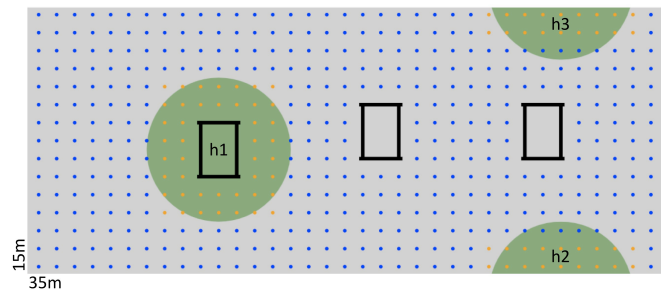


Figure 6 – Simulation scenario

Table 2 shows the values for  $MSA_s$  and  $FSA_s$  considered for the hotspots and its surrounding yellow graph nodes. Blue

graph nodes (Figure 6) exert no attraction over the agent, the value for both  $MSA_s$  and  $FSA_s$  are zero. The  $MSA_s$  and  $FSA_s$  values were assumed to be constants. The  $MSA_s$  and  $FSA_s$  definition can be enhanced by considering effects of various design and management attributes. An example of the evaluation of consumers attraction can be found in Oppewal, H., and Timmermans, H. (1999). The authors estimated a stated preference model from responses to descriptions of an hypothetical shopping centers considering attributes such as: area for pedestrians, window displays, street layout, and street activities.

Table 2 – Hotspots configuration

hotspot	role	$MSA_s$	$FSA_s$
h1	jewelry	0.2	0.5
h2	toy store	0.8	0.6
h3	shoes store	0.8	0.6

#### 4.2 Calibration

The calibration process aimed to calibrate the agents' profile ( $MSA_a$  and  $FSA_a$ ) in order to reproduce the number of stopped pedestrians at each hotspot. For this purpose, four groups of simulations were run (s1, s2, s3, s4). For each simulation group, 50 simulations were performed. Two agents classes were implemented: male agents (MA) and female agents (FA). By definition, male agents have  $FSA_a = 0$  and female agents have  $MSA_a = 0$ . Table 3 shows the configuration profiles defined for each simulation group.

Table 3 – Agents profile configuration

simulation group	MA	FA
s1	$MSA_a = 0.1$	$FSA_a = 0.1$
s2	$MSA_a = 0.5$	$FSA_a = 0.5$
s3	$MSA_a = 0.7$	$FSA_a = 0.7$
s4	$MSA_a = 0.9$	$FSA_a = 0.9$

The only variables in simulations were  $MSA_a$  and  $FSA_a$ . The scenario configuration was kept constant. Agents' tendency to stop ( $T_\alpha$ ) was set to 0.7. According to observed data, each simulation run comprised 80 agents, 40% MA and 60% FA. Pedestrians are generated with a fixed rate over time, with 40% of change to be male and 60% of change to be female. Figure 7 shows a simulation screenshot, MA are green circles and FA are red circles. A simulation video is available at: <https://youtu.be/100UgNMaoNA>.

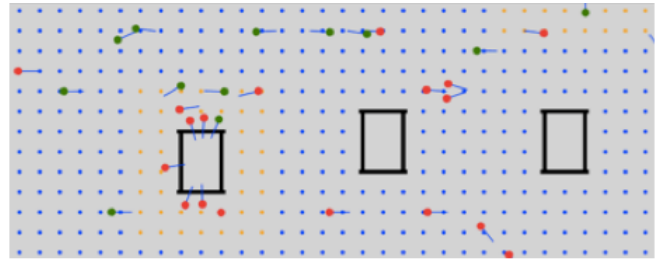


Figure 7 – Simulation screenshot

Figure 8 shows a color map of the results for all simulation groups (s1, s2, s3, s4), and the average number of agents stopping at each hotspot (h1, h2, h3) over 50 simulation runs.

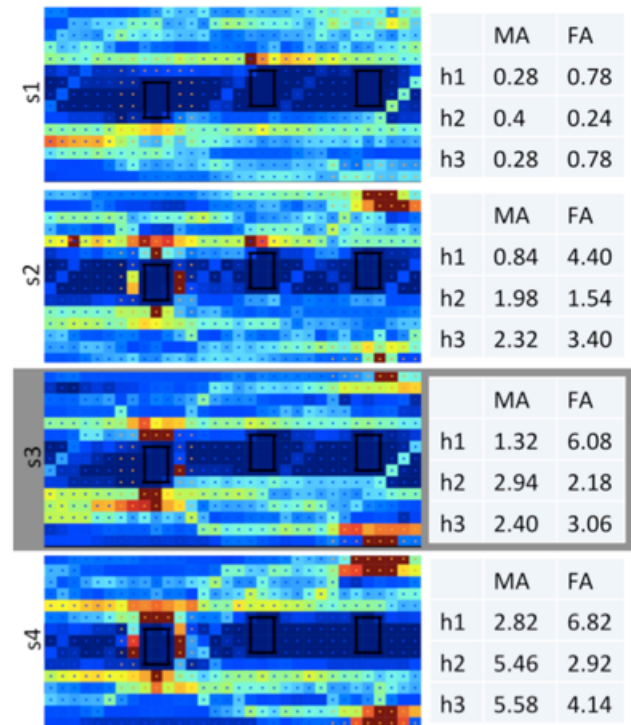


Figure 8 – Simulations results

#### 4.2 Simulation Analysis

Simulation group s3 presented the best adjustment to the observed data. Higher values of  $MSA$  and  $FSA$  lead to higher attraction to hotspots. However, it is important to highlight that even though a pedestrian chooses a route to get closer to a shop window, he needs to reach a hotspot to stop. If the hotspot area is too crowded, he may not reach the hotspot, due to the social force effect, and do not stop. Thus, the attraction effect has a tendency to be balanced. Figure 9 show the s3 color map and the color map generated from real data. The s3 color map is one of 50 simulations. It is possible to observe differences in color patterns between simulation and real data. This difference is due the noise of

pedestrians' tracking process and camera perspective. It is important to highlight stopping pattern at hotspots is similar.

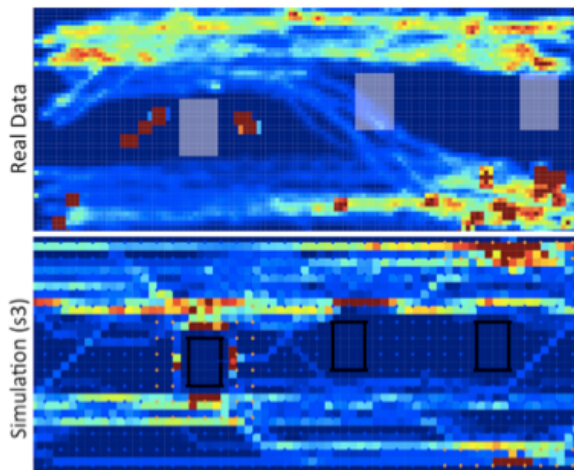


Figure 9 – Real data versus simulation data

## 5 Conclusions

The modeling approach presented in this paper provides a sound representation of pedestrian route choice dynamics considering the attraction to shop windows. Route choice is based on a combination of distance, impedance generated by other pedestrians and shop window attraction. The model differs from other pedestrians' route choice approaches because it seamlessly incorporates pedestrians social force into the route choice decision process.

In this model, we have created an association between the pedestrian's profile and store segment. When a pedestrian defines a route, due to its attraction to a store, he draws his chance to stop at a hotspot. The formulation of stopping chances can be enhanced through a more complex agent abstraction. However, it is well known that increasing model complexity usually leads to an increase in the calibration process effort.

The analysis from simulations indicates that the agents' emerging behavior provides a promising approach for real case applications. This model formulation is capable of supporting more complex agents' profiles and applications to different environments, such as variable shopping premises, expositions sites and passengers terminals.

## Acknowledgments

The authors want to thank the financial support by the National Council for the Improvement of Higher Education (CAPES) and National Council for Scientific and Technological Development (CNPq), both of Brazil.

## References

- [Klühl, F., & Bazzan, A. L., 2012] Agent-based modeling and simulation. *AI Magazine*, 33(3), 29.
- [Macal, C. M., & North, M. J., 2006] Introduction to agent-based modeling and simulation. In *MCS LANS Informal Seminar*.
- [Rossetti, R. J., Bordini, R. H., Bazzan, A. L., Bampi, S., Liu, R., & Van Vliet, D., 2002] Using BDI agents to improve driver modelling in a commuter scenario. *Transportation Research Part C: Emerging Technologies*, 10(5), 373-398.
- [Gaud N, Galland S, Gechter F, et al., 2008] Holonic multilevel simulation of complex systems: application to real-time pedestrians simulation in virtual urban environment. *Simul Model Pract Theory* 2008; 16: 1659-1676.
- [Hoogendoorn S, Bovy P and Daamen W., 2002] Microscopic pedestrian wayfinding and dynamics modelling. In: Schreckenberg M and Sharma S (eds) *Pedestrian and evacuation dynamics*. Berlin: Springer, 2002, pp.123-155.
- [Pretto CO, Jacobsen AC and Cybis H B B. 2011] A multi-layer simulation model for vehicle and pedestrian. In: *Proceedings of the 90th Annual Meeting of the Transportation Research Board*, Vol. 1, Washington, 2011, pp.1-15.
- [Helbing D., 1991] A mathematical model for the behavior of pedestrians. *Behav Sci*. 1991; 36: 298-310.
- [Helbing D and Molnar P., 1995] Social force model for pedestrian dynamics. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdisciplin Top* 1995; 51: 42-82.
- [Kirik ES, Yurgel'yan TB and Krouglov DV., 2009] The shortest time and/or the shortest path strategies in a pedestrian dynamics model. *J Siberian Fed Univ Math Phys* 2009; 2: 271-278.
- [Dressler D, Groß M, Kappmeier J, et al. 2010] On the use of network flow techniques for assigning evacuees to exits. *Procedia Eng* 2010; 3: 205-215.
- [Lämmel G and Plaue M., 2014] Getting out of the way: collision avoiding pedestrian models compared to the real world. In: Weidmann U, Kirsch U and Schreckenberg M (eds) *Pedestrian and evacuation dynamics 2012*. Berlin: Springer, 2014.
- [Papadimitriou E., 2012] Theory and models of pedestrian crossing behaviour along urban trips. *Transp Res Part F Traffic Psychol Behav* 2012; 15: 75-94.
- [Kretz T, Große A, Hengst S, et al., 2011] Quickest paths in simulations of pedestrians. *Adv Complex Syst* 2011; 14: 733-759.
- [Kretz, T., Lehmann, K., Hofsaß, I., & Leonhardt, A., 2014] Dynamic assignment in microsimulations of pedestrians. *arXiv preprint arXiv:1401.1308*

- [Crociani, L., & Lämmel, G., 2016] Multidestination Pedestrian Flows in Equilibrium: A Cellular Automaton-Based Approach. *Computer Aided Civil and Infrastructure Engineering*, 31(6), 432-448.
- [Patil S, Van Den Berg J, Curtis S, Lin MC and Manocha D., 2010] Directing crowd simulations using navigation fields, *IEEE Trans Visualization Comput Graphics* 2010; 17(2): 244–254.
- [Treuille A, Cooper S and Popovic Z., 2006] Continuum crowds. *ACM Trans Graphics* 2006; 25(3):1160–1168.
- [Teknomo K. 2008., 2008] Modeling mobile traffic agents on network simulation. In: *Proceedings of the 16th annual conference of the Transportation Science Society of the Philippines (TSSP)*, Manila, 2008.
- [Teknomo K, Bauer D and Matyus T., 2008] Pedestrian route choice self-organization. In: *Proceedings of the 3rd international symposium of simulation in transportation*, Gold Coast, Australia, 2008.
- [Wang, W. L., Lo, S. M., Liu, S. B., & Kuang, H. 2014] Microscopic modeling of pedestrian movement behavior: Interacting with visual attractors in the environment. *Transportation Research Part C: Emerging Technologies*, 44, 21-33.
- [Dijkstra, J., Timmermans, H. J., & de Vries, B., 2013]. Activation of shopping pedestrian agents—Empirical estimation results. *Applied Spatial Analysis and Policy*, 6(4), 255-266.
- [Borgers, A., & Timmermans, H., 1986]. A Model of Pedestrian Route Choice and Demand for Retail Facilities within InnerCity Shopping Areas. *Geographical analysis*, 18(2), 115-128.
- [Ali, W., & Moulin, B., 2006]. How artificial intelligent agents do shopping in a virtual mall: A ‘believable’ and ‘usable’ multiagent-based simulation of customers’ shopping behavior in a mall. In *Advances in Artificial Intelligence*(pp. 73-85). Springer Berlin Heidelberg.
- [Werberich, B. R., Pretto, C. O., & Cybis, H. B. B., 2014] Pedestrian route choice model based on friction forces. *Simulation*, 0037549714547295.
- [Helbing D, Farkas I and Vicsek T., 2000] Simulating dynamical features of escape panic. *Nature* 2000; 407: 487–490.
- [Dijkstra E., 1959] A note on two problems in connection with graphs. *Numer Math* 1959; 1: 269–271.
- [Dijkstra, J., Timmermans, H., Jessurun, J., & de Vries, B., 2014]. Modeling Time Duration of Planned and Unplanned Store Visits in a Multi-Agent Simulation of Pedestrian Activity in City Centers. In *Pedestrian and Evacuation Dynamics 2012* (pp. 815-823). Springer International Publishing.
- [Oppewal, H., & Timmermans, H., 1999] Modeling consumer perception of public space in shopping centers. *Environment and Behavior*, 31(1), 45-65.



# Using Topological Statistics to Bias and Accelerate Route Choice: preliminary findings in synthetic and real-world road networks

Fernando Stefanello, Bruno C. da Silva, Ana L. C. Bazzan

Instituto de Informática – Universidade Federal do Rio Grande do Sul – Porto Alegre, RS, Brazil  
{fstefanello, bsilva, bazzan}@inf.ufrgs.br.

## Abstract

This paper discusses the first steps towards the definition of novel statistics and metrics that characterize networks in terms of the complexity they pose to the traffic assignment problem. Here, we follow an approach in which the assignment emerges from routes selected by learning agents. Specifically, we deal with issues related to how routes are *coupled*. We first define and quantify route coupling, i.e., how much a given route is coupled with other routes that can be used by learning agents. The investigation of route coupling is important in multi-agent reinforcement learning settings since it measures how a change in action selection by one agent interferes with the actions taken by other agents. Our preliminary empirical results indicate that using route coupling to bias the learning process of agents results in faster convergence in the traffic assignment problem.

## 1 Introduction

Traffic networks can be represented as directed graphs  $G = (V, A)$ , where  $V$  represents the set of nodes (street or road intersections, points of interest, or districts), and  $A$  the set of arcs (street or road segments). The topology of these graphs, as well as other characteristics such as how demand is distributed and how latency (cost) functions are defined, have a great influence on the *traffic assignment problem* (TAP), whose goal is to assign the traffic demand (number of trips, vehicles) onto the arcs of a network  $G$ .

Our objective is to characterize road networks by designing qualitative metrics and statistics. In particular, given a graph, an origin-destination matrix, and latency functions, we wish to design metrics that numerically represent the degree of difficulty posed to the assignment task. We study the TAP not under the traditional, centralized approach, but via an agent-based variant. Here, each agent (in this context, a driver or vehicle) learns to select a route taking it from its origin to its destination. Given that in such a process a decision made by an agent affects the outcome of other agents, this is a typical multiagent reinforcement learning problem (MARL).

A closely related issue to the one studied in this paper is that of how to characterize a traffic network in terms of how

high the price of anarchy (*PoA*); [Koutsoupias and Papadimitriou, 1999; Roughgarden and Tardos, 2002]) is. The *PoA* measures the loss in performance caused by a situation in which each driver seeks to minimize its travel time independently. Some works refer to this as *selfish routing*. One of our goals is to investigate how the magnitude of the *PoA* affects the learning task. Knowing this, one could anticipate how complex the assignment will be. As the Braess paradox—a case in which the *PoA* can be very high—shows, the addition of arcs to a network may end up causing more congestion [Braess, 1968].

This paper discusses the first steps towards the definition of novel statistics and metrics to characterize networks in terms of the complexity they pose to the TAP. In particular, we propose a metric based on topological properties of the network, whose goal is to measure how *coupled* routes are. *Coupling* refers, intuitively, to how many points of interaction a route has with other routes, and how likely it is that agents might decide to switch routes. This serves as a way of estimating how strongly an agent on one route might be affected by other agents who might change their behavior. Measuring this effect is important since strong dependencies make learning in MARL settings harder. We describe a way of using coupling statistics to bias the learning process of agents in a way that empirically counteracts the negative effects of non-stationary in the learning process, and that is conducive to faster convergence to an equilibrium. We evaluate our methods in several road networks with different topologies and demands—both real-world networks and synthetic ones, such as those affected by the Braess paradox. In order to construct challenging networks that are affected by this paradox, we modify an existing method to extend Braess networks to arbitrary sizes.

This paper is organized as follows: Section 2 introduces the classical, optimization-based approach to the TAP, and an alternative MARL-based approach to compute a user equilibrium. Section 3 introduces our main methods and presents a few networks that we use to illustrate them. We present preliminary results in Section 4, related work in Section 5, and present concluding remarks in Section 6.

## 2 The Traffic Assignment Problem

This section introduces a mathematical formulation for the traffic assignment problem and presents the notation that will

be used throughout the paper. As previously mentioned, a transportation network can be represented as a directed graph  $G = (V, A)$ . Each arc  $a \in A$  has a latency function which is a function of the traffic in that arc—it quantifies the effects of network usage, such as traffic congestion. This function depends on parameters of the arc such as the time  $\tau$  to traverse it without congestion (this is also known as free flow time), and the nominal capacity  $\rho$  of the arc (e.g., in terms of number of vehicles).

In this work we denote the set of incoming arcs to node  $v \in V$  by  $IN(v)$ , and the set of outgoing arcs from node  $v \in V$  by  $OUT(v)$ . In addition, let  $\mathcal{C} = \{(o(1), d(1)), \dots, (o(|\mathcal{C}|), d(|\mathcal{C}|))\} \subseteq V \times V$  denote the set of commodities, i.e., a set of origin-destination (OD) pairs. Here,  $o(\sigma)$  and  $d(\sigma)$  represent, respectively, the origin and destination nodes for  $\sigma = 1, \dots, |\mathcal{C}|$ . Each commodity  $\sigma$  has an associated demand  $r_\sigma = r_{o(\sigma), d(\sigma)}$ ; i.e., each OD pair  $(o(\sigma), d(\sigma))$  has an associated demand  $r_\sigma$  that emanates from node  $o(\sigma)$  and terminates in node  $d(\sigma)$ .

Furthermore, each arc has a latency function that expresses how travel time depends on the traffic flow on that arc. If drivers were to selfishly select routes that minimize their individual travel times, they could simply select the shortest path that satisfies their desired origin and destination nodes. This strategy, however, makes several underlying assumptions which are often not met, or are unrealistic: for example, that the time taken to traverse an arc is constant and independent of other drivers. This is clearly not the case in real traffic networks, where the maximum flow allowed in an arc depends on which routes other drivers take and on how many drivers occupy an arc at a given time.

When simulating traffic conditions on a given network, a designer needs to select a latency function that approximates the real-life costs of navigating in that network. One of the best-known and widely used latency function for real-world networks, often referred to as the BPR function, was introduced by the U. S. Bureau of Public Roads [Bureau of Public Roads, 1964]. This is a non-linear, convex, and strictly increasing function. Linear functions are also frequently used (e.g., in the case of networks affected by the Braess paradox) to represent the latency on each arc. In this work, whenever we refer to networks affected by this paradox, we assume that the latency is represented by  $l_a(f_a) = m_a f_a + n_a$ , where  $m_a \in \mathbb{R}^+$  and  $n_a \in \mathbb{R}$  are parameters and  $f_a$  is the flow on arc  $a$ . We also assume that  $l_a(f_a) \geq 0$ .

In the next section we introduce a mathematical model for assignment problem—this is a classical, optimization-based method to solve the TAP. We then describe an alternative way of solving a version of this problem, namely by searching for a user equilibrium via MARL techniques.

## 2.1 A Model of Traffic Assignment

In this subsection we present mathematical models describing the two main principles that characterize the traffic assignment: the *system optimum* (SO) and *user equilibrium* (UE) [Wardrop 1952]. The latter principle states that “under equilibrium conditions traffic arranges itself in congested networks such that all used routes have equal and minimum costs, while all those routes that were not used have greater

or equal costs”. The former principle refers to the system as a whole and states that the average trip time is minimum.

Beckmann *et al.* [1956] were the first to propose and solve a mathematical model to compute both the SO and UE solutions. In what follows we present an arc-based mathematical model for SO and for the UE model. A path-based mathematical model may also be used to represent the respective assignment problems. Let  $x_a^\sigma$  be variables indicating the flow on arc  $a$  for the commodity  $\sigma$ ; let  $f_a$  be the total flow on arc  $a$  and  $\Phi_a$  be the associated cost for the arc  $a$ . The **SO model** for a multi-commodity network can be written as:

$$\min \Phi = \sum_{a \in A} \Phi_a \quad (1)$$

subject to:

$$m_a f_a^2 + n_a f_a \leq \Phi_a \quad (2)$$

$$f_a = \sum_{\sigma \in \mathcal{C}} x_a^\sigma \quad \forall a \in A \quad (3)$$

$$\sum_{a \in IN(v)} x_a^\sigma - \sum_{a \in OUT(v)} x_a^\sigma = \begin{cases} d_\sigma, & \text{if } v = d(\sigma) \\ -d_\sigma, & \text{if } v = o(\sigma) \\ 0, & \text{otherwise} \end{cases} \quad \forall v \in V, \sigma \in \mathcal{C} \quad (4)$$

$$x_a^\sigma \geq 0, \forall a \in A, \forall \sigma \in \mathcal{C} \quad (5)$$

$$f_a \geq 0, \Phi_a \geq 0 \quad \forall a \in A. \quad (6)$$

Objective function (1) aims at finding a flow assignment for each arc that minimizes the total cost for the system—resulting in an assignment respecting the SO principle. Constraints (2) associate the cost of each arc  $a$  to the variable  $\Phi_a$ ; constraints (3) associate the total flow in arc  $a$  to variables  $f_a$ ; constraints (4) ensure flow conservation, and constraints (5) and (6) define the domain of variables. Note that this model has quadratic constraints, since it contains a product between flow variables in the constraints (2)—in particular, the product of latency costs and arc flows (i.e.,  $(m_a f_a + n_a) f_a$ ). The SO model can be extended to networks that consider the BPR latency function by changing constraints (2). This formulation uses a set of variables  $\Phi_a$  and constraints (2) to define the latency cost on each arc; this is especially useful for the case where the latency function is a composition of linear functions (see Case 2, Section 3.1, for more details).

We now consider the **UE model**, whose objective is to minimize the function

$$\Phi = \sum_{a \in A} \int_{\bar{x}_a}^{f_a} l_a(x) dx \quad (7)$$

Since we assume that the latency function is  $l_a(x) = m_a x + n_a$ , we can simplify this expression. For the case where  $n_a \geq 0$ ,  $\bar{x}_a = 0$ . For the case where  $n_a < 0$ ,  $\bar{x}_a$  should be the max  $x$  such that  $l_a(x) = 0$ ; i.e.,  $\bar{x}_a = -n_a/m_a$ . Since we also consider  $l_a(x) \geq 0$ , the latency function becomes a composition of two line segments, thereby defining a piecewise linear function which is convex and strictly increasing. To completely define the UE model, we now only need to replace the constraints (2) with the following set of constraints:

$$\frac{1}{2}m_a f_a^2 + n_a f_a - \left( \frac{1}{2}m_a \bar{x}_a^2 + n_a \bar{x}_a \right) \leq \Phi_a. \quad (8)$$

If  $n_a \geq 0$ , then  $\bar{x}_a = 0$  and constraints (8) are reduced to

$$\frac{1}{2}m_a f_a^2 + n_a f_a \leq \Phi_a \quad (9)$$

Solving the above-mentioned models involves assigning a traffic flow to each arc in order to obtain a global assignment that is consistent either with the SO or UE hypotheses. The models can be solved via mathematical programming using general-purpose solvers such as CPLEX and MOSEK.

## 2.2 Multiagent Learning for Route Choice

Mathematical programming-based methods like the ones previously mentioned may have difficulties if non-linear latency functions are used. Furthermore, these methods can typically only solve static assignments. Unlike optimization approaches that use mathematical models suitable only for static assignment with linear or convex latency functions, MARL can be used to compute traffic assignment solutions by considering each individual driver as an autonomous agent, in a microscopic fashion. This strategy can be used to tackle a wide range of problems, such as those involving static or dynamic assignment, and also ones that require the simulation of complex systems.

In this paper we assume that when using MARL to compute traffic assignment solutions, each agent learns to make decisions (i.e., to select routes) by using reinforcement learning. We use the Q-learning algorithm to update the value of each state-action pair of the agent; this value represents the expected long-term utility that the agent hopes to achieve by selecting a given action in a state, and following the current action-selection strategy thereafter. This update is performed based on an experience tuple  $(s, a, s', rew)$  according to Equation 10, where  $\alpha$  is the learning rate and  $\gamma$  is a discount rate applied for future rewards. Details of the use of Q-learning for the TAP are given in Section 4.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left( rew + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \quad (10)$$

## 3 Methods

We are interested in characterizing traffic networks of different types—for instance, synthetic (or pictorial) networks such as those used to illustrate the Braess paradox; networks whose demand distributions are closer to real-world cases (versus symmetric-demand distributions such as those in Braess paradox); single versus multicommodity cases; networks with linear versus non-linear latency functions; etc. Hence, prior to discussing the statistics we employ to measure the coupling between routes, we introduce and discuss the nature of a few selected networks: arbitrarily large Braess-paradox networks (in Section 3.1), the OW network (Section 3.2), and the Sioux Falls network (Section 3.3). We then introduce a metric for characterizing some properties of these networks—the *coupling statistic* (Section 3.4).

## 3.1 Arbitrarily Large Braess-Paradox Networks

The Braess paradox occurs whenever adding resources to a transportation network deteriorates the quality of a UE. Using Beckmann’s model, Braess [1968] described situations in which adding a road to a congested traffic network could have a counter-intuitive outcome—namely, the overall travel time could increase. This phenomenon can be interpreted as follows: suppose we close a road or increase its free travel time by decreasing the maximum allowed speed; if the cost (e.g., the total travel time at UE) decreases, then we observe the Braess paradox.

Roughgarden [2001] discusses the problem of designing networks so that the Braess paradox does *not* occur—more specifically, which edges should be removed from a network to obtain the best possible flow at Nash equilibrium. This author also discusses how to create arbitrarily large Braess graphs, whose sizes depend on a factor  $p$ ; a few examples are shown in Figure 1. Although single-commodity, these networks are interesting since they are associated with a high PoA. Roughgarden’s method allows the investigation of the PoA in large graphs, rather than in simple ones like the network in Figure 1a. Being able to produce large networks with this property is useful in the context of our work because we aim at defining novel statistics and metrics to characterize networks (such as Braess networks of different sizes) in terms of the complexity they pose to the TAP—in particular in the context of using MARL algorithms to solve it.

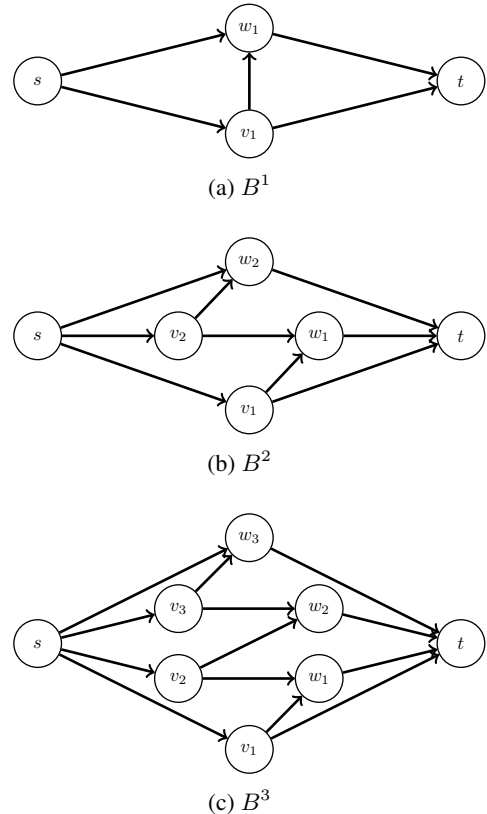


Figure 1: Sample graphs that result in the Braess paradox.

Braess graphs of arbitrary size can be generated as follows: given a size parameter  $p$ , the  $p$ -th Braess graph  $B^p$  is constructed with a set  $V^p = \{s, v_1, \dots, v_p, w_1, \dots, w_p, t\}$  of  $2p + 2$  vertices; and a set of arcs  $A^p$  defined by  $\{(s, v_i), (v_i, w_i), (w_i, t) : 1 \leq i \leq p\} \cup \{(v_i, w_{i-1}) : 2 \leq i \leq p\} \cup \{(v_1, t)\} \cup \{(s, w_p)\}$ . Next, we describe how to associate latency functions with each of these arcs.

### Latency Functions

A key decision when designing arbitrarily large Braess networks is how to associate latency functions to the arcs. We extend the method described by Roughgarden [2001] in three ways: (i) we allow networks with arbitrary constant arc costs  $c$ , instead of unitary costs; (ii) we allow arbitrary demand values<sup>1</sup>  $r$  instead of only fractions of a unitary demand; and (iii) we introduce simpler piecewise latency functions which, although resulting in lower PoA values, allow for solutions to be more easily obtained via standard commercial optimization packages.

In what follows, the demand for a given commodity (or origin–destination pair)  $\sigma \in \mathcal{C}$  is indicated as  $r_\sigma$ . Let  $c > 0$  be the cost associated with constant-cost arcs, and  $r \in \mathbb{R}^*+$  be the total demand of the network. Roughgarden [2001] uses  $c = 1$  and  $r = p$ ; this constrains the networks that can be generated since  $p$  is typically much smaller than the demand. We modify this formulation so that  $r$  can be arbitrarily large. By default, we consider  $c = 10$  and  $r = 4200$ .

We now define the latency function  $l_a(x)$  associated with each arc  $a$ . The value of  $i$  of each arc is the same as described in Section 3.1. We omit the subscript  $p$  in the latency function to simplify notation. The latency functions are defined as:

- $l_a(x) = 0$  for arcs of form  $a = (v_i, w_i) \forall i \in \{1, 2, \dots, p\}$ ;
- $l_a(x) = c$  for arcs of form  $a = (v_i, w_{i-1}) \forall i \in \{2, \dots, p\}, (s, w_p)$  or  $(v_1, t)$ .

For the remaining arcs of the form  $a = (w_i, t)$  or  $(s, v_{p-i+1}) \forall i \in \{1, 2, \dots, p\}$ , the latency function is defined as a function of the flow in the arc. We use two strategies to define the latency of these arcs. The first strategy is to use a linear function with  $n_a = 0$ , and the second one is to use a piecewise function based on the latency function described by Roughgarden [2001]; these are henceforth referred to as *Case 1* and *Case 2* respectively.

#### Case 1 - Linear function with $n_a = 0$

In this case, the latency function  $l_a(x) = m_a x + n_a$  is a simple line segment satisfying  $l_a(0) = 0$  and  $l_a^p(\frac{r}{p}) = ic$ , i.e.:

$$m_a = \frac{icp}{r} \quad (11)$$

$$n_a = 0. \quad (12)$$

#### Case 2 - Piecewise Linear function with $n_a < 0$

In this case, the latency function  $l_a(x) = m_a x + n_a$  is a composition of two line segments satisfying  $l_a(0) = 0$ ,

$$l_a(\frac{r}{p+1}) = 0, \text{ and } l_a^p(\frac{r}{p}) = ic, \text{ i.e.,}$$

$$m_a = \frac{icp^2 + cip}{r} \quad (13)$$

$$n_a = -cip. \quad (14)$$

Figure 2 depicts fixed-cost arcs in blue, while arcs with cost equal to zero appear in red. The remaining arcs (in black) are the ones with latency functions that depend on the arc's flow.

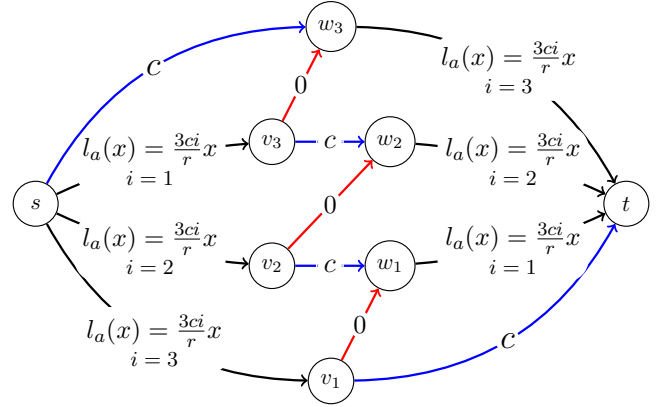


Figure 2: Braess Graph  $B^3$  with latency functions.

Figure 3 shows a few sample latency functions:  $f_0$  is the latency function for the family functions proposed by Roughgarden [2001];  $f_1$  represents the latency function for Case 1, and  $f_2$  represents the latency function for Case 2.

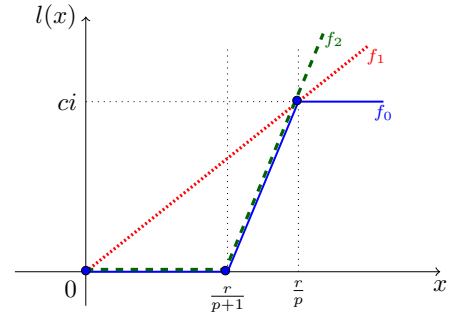


Figure 3: Sample latency functions.

### 3.2 The OW Network

Besides Braess-paradox networks, another network of interest in this work is the OW network (due to Ortúzar and Willumsen [2001]), depicted in Figure 4 and henceforth referred to simply as OW. Although this not a full reproduction of a real-world network, it contains interesting real-world elements. This network represents two residential areas (nodes A and B in the figure) and two major shopping areas (nodes L and M). The numbers associated with arcs,  $\tau_a$ , denote travel times in those arcs under free flow (in both ways). The proposed demand for this network corresponds to a total of  $r = 1700$  trips, distributed among four commodities: AL,

<sup>1</sup>In this paper we use the term *demand*, rather than *traffic rate*, but keep the symbol  $r$  used by Roughgarden [2001].





decide to switch to any given alternative route. If two routes share many arcs, for instance, the effective flow on both of them will be more strongly affected by agents deciding to travel on those routes or deciding to abandon them in favor of other options; these routes are, therefore, highly coupled.

We define the coupling  $\Psi(R_i)$  of a route  $R_i$  as the expected value of the *interaction*  $I(R_i, R_j)$  of that route with other routes  $R_j$  in the system. This expectation is defined with respect to a probability distribution  $P$  over possible routes: routes that are more likely to be selected by agents (based on their individual preferences for reaching particular places in the network) have higher probability. Specifically, we define:

$$\begin{aligned}\Psi(R_i) &= E_P[I(R_i, \cdot)] \\ &= \sum_{j=1}^{|R|} P(R_j)I(R_i, R_j)\end{aligned}\quad (16)$$

where  $I(R_i, R_j)$  is the normalized number of shared arcs between routes  $R_i$  and  $R_j$ :

$$I(R_i, R_j) = \frac{1}{|R|} \frac{|R_i \cap R_j|}{|R_i|}\quad (17)$$

Intuitively,  $I$  measures how much of the underlying structure and resources of the network are shared by two routes, and  $P$  reflects the agents’ demands for different routes at some point in time—defined according to their preferences for reaching different regions of the network<sup>2</sup>.

In our experiments, we refer to  $\Psi(R_i)$  as the *mean coupling* of a route  $R_i$  whenever  $P$  is assumed to be a uniform distribution. This corresponds to the case where we have no prior information about agents’ preferences for reaching particular nodes of the network. When we do have that information, we can encode it in  $P$ , which then represents the relative preferences of agents for choosing different routes.

To illustrate the use of the proposed coupling statistics, we start with a simple example—the Braess graph  $B^1$ , depicted in Figure 1(a). This network has a single commodity: the entire demand of  $r = 4200$  drivers travels from  $s$  to  $t$ . There are 3 possible routes they can select from: st1, st2, and st3, as shown in Table 1. These form the set  $R$  of routes, which was generated by using a  $k$ -shortest paths algorithm Yen [1971]. This algorithm returns  $k$  shortest paths (when analyzed under free flow) associated with a given commodity. The st1 path is the shortest one to satisfy the single demand in  $B^1$ ; st2 and st3 are the second and third shortest paths, respectively.

Table 2 shows the normalized number of shared arcs between any two routes, which is the second term in Equation 17. This table should be read row-wise: e.g., st1 shares 33% of its elements with st2. The coupling  $\Psi$  of each route (Equation 16) is shown in Table 3 for the case of uniform  $P$ . Under this distribution, it is equivalent to the average normalized number of shared arcs with other routes:  $\Psi(st2)$ , for instance,

<sup>2</sup>Route coupling does not take into account interactions of a route with itself. We abuse notation in Equations 16 and 17. In reality, these are defined over the set  $R - \{R_i\}$ , not  $R$ .

is  $\frac{1}{2}(50.00 + 0.00)$ . Note that st1 has the highest coupling—but traditional MARL approaches ignore this information.

We propose to use the coupling statistic to bias the learning process of each agent. We report results that relate to a very simple type of biasing; namely, the Q-table of each agent is initialized with the negative of the coupling for each route. Continuing with our example (the  $B^1$  network), the Q-values associated with action st1 were initialized with  $-33.33$ , and the Q-values associated with actions st2 and st3 were initialized with  $-25.00$ . This means we are using information about route coupling to bias agents’ preferences in a way that leads more agents to prefer routes st2 or st3, rather than st1. Our hope is that this will successfully bias route selections, guiding agents in their exploration of which are the best actions to perform and accelerating convergence to an equilibrium.

Route Name	Arcs
st1	$sv_1 \rightarrow v_1w_1 \rightarrow w_1t$
st2	$sw_1 \rightarrow w_1t$
st3	$sv_1 \rightarrow v_1t$

Table 1:  $k = 3$  shortest routes for network  $B^1$

	st1	st2	st3
st1	100.00	33.33	33.33
st2	50.00	100.00	0.00
st3	50.00	0.00	100.00

Table 2: Normalized number of shared arcs ( $B^1$ )

$\Psi$		
st1	st2	st3
33.33	25.00	25.00

Table 3: Route Coupling  $\Psi$  under uniform  $P$  ( $B^1$  network)

## 4 Simulations and Results

In order to evaluate the use of the coupling  $\Psi$  as a biasing method in MARL we will use the OW network, since solving the traffic assignment problem in it has been shown to be a complex task. In our experiments, agents use Q-Learning, a standard reinforcement learning algorithm, to learn to select routes and reach a UE. Note that at this point we disregard the fact that the UE may be socially bad; see discussion in Section 5. In a traditional reinforcement learning setting, each agent keeps its own Q-table. Q-values are associated with each action—in this case, one of the  $k$  shortest routes available for an agent to travel from its origin to its destination. Note that this formulation resembles a repeated game, where there is just one state Claus and Boutilier [1998]. This means that Equation 10 can be simplified and does not require a discount rate. While this simplifies the learning problem, the large number of concurrently-learning agents in MARL makes the problem inherently more complex. Each agent selects actions according to an  $\epsilon$ -greedy strategy: with probability  $1 - \epsilon$ , the action with highest Q-value is selected; with

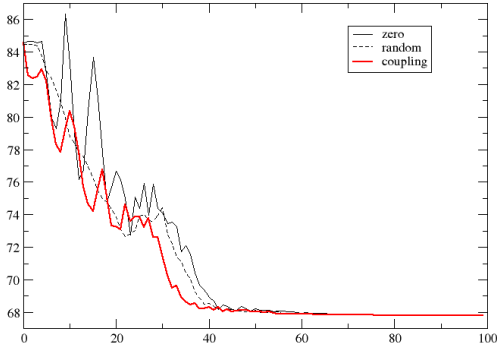


Figure 6: Mean latency as a function of number of episodes.

probability  $\epsilon$ , a random action is selected. We initialize  $\epsilon$  with a high value and decay it by  $(1 - \delta)\%$  at the end of each episode. This allows for high exploration at the beginning of the learning process. In our experiments, the reward is the negative of the individual travel time of an agent,  $\alpha = 0.3$ ,  $\epsilon = 1.0$  at the first episode, and  $\delta = 0.9$ . These values were selected after extensive tests with different ranges of values.

The OW network, used in the following experiments, has  $|\mathcal{C}| = 4$  commodities, and we associate with it a total demand of  $r = 1700$ . We computed  $k = 5$  shortest paths per commodity. Table 4 shows the coupling for each of the  $|\mathcal{C}| \times k = 20$  shortest routes, computed via Equation 16.

In our experiments, we measure the average latency of the  $r$  agents in the system and plot it as a function of time (Figure 6). We do so under three distinct situations: (i) the Q-table of agents is initialized with zeros; (ii) the Q-table is initialized with the negative of the coupling statistic  $\Psi(R_i)$  associated with route  $R_i$ ; (iii) the Q-table is initialized with a random value between 0 and the (negative) maximum value of  $\Psi$ . Note that an approximation of the UE for the OW network (which can be computed, e.g., via CPLEX) is of approximately 67 minutes of average latency. In Figure 6 we show that biasing the Q-values of agents with  $\Psi$  leads to a faster convergence to the UE.

In the future we plan to perform further experiments; first, one that starts with a lower value of  $\epsilon$  so that agents can exploit the bias provided at the beginning of the learning process for longer periods of time. Due to lack of space, we omit a table that shows that, at the end of the learning process, the number of agents using each of the  $k$  paths is roughly correlated with  $\Psi$  of the corresponding path. This suggests that the coupling statistic does serve, indeed, as a proxy to how desirable different routes are—one that considers how agents selecting between them are (as a consequence of the concurrent learning aspect of MARL) negatively impacted by non-stationarity.

## 5 Related Work

A number of techniques from transportation planning, economics, operations research, and computer science deal with the TAP. Due to lack of space, we omit classical approaches. The reader is referred to Ortúzar and Willumsen [2001]. For

Name	Arcs	Mean Coupling
AL1	AC → CG → GJ → JI → IL	36.84
AL2	AC → CG → GJ → JL	39.47
AL3	AC → CF → FI → IL	23.68
AL4	AC → CD → DG → GJ → JI → IL	30.70
AL5	AC → CD → DG → GJ → JL	31.58
AM1	AC → CD → DH → HK → KM	31.58
AM2	AC → CG → GJ → JK → KM	37.89
AM3	AC → CG → GH → HK → KM	32.63
AM4	AD → DH → HK → KM	22.37
AM5	AC → CG → GJ → JM	35.53
BL1	BD → DG → GJ → JI → IL	27.37
BL2	BD → DG → GJ → JL	27.63
BL3	BA → AC → CG → GJ → JI → IL	32.46
BL4	BA → AC → CG → GJ → JL	33.68
BL5	BA → AC → CF → FI → IL	21.05
BM1	BE → EH → HK → KM	21.05
BM2	BD → DH → HK → KM	27.63
BM3	BD → DE → EH → HK → KM	20.00
BM4	BE → ED → DH → HK → KM	18.95
BM5	BD → DG → GJ → JK → KM	28.42

Table 4: Meaning Coupling statistic of the  $k = 5$  routes associated with each of the four OD Pairs in the OW network.

approaches that seek to balance the UE and the SO, we refer the reader to Bazzan and Chira [2015]. We focus on those that seek to approximate the UE by means of reinforcement learning or other agent-based approaches.

To the best of our knowledge, no attempts have been made to bias the learning of the UE by using similar statistics and metrics as the one we propose. Similar metrics such as the Path Size Logit model [Ben-Akiva and Bierlaire, 1999] and the C-Logit model [Cascetta *et al.*, 1996] are used to select routes in a network in route choice models. However, these strategies differ in the formulation metrics since they are based on the length of arcs.

A natural way to tackle the problem of route choice is via agent-based simulation techniques. Examples are MAT-Sim Balmer *et al.* [2004], Klügl and Bazzan [2004] and Dia and Panwai [2014]. A learning-based approach to route choice was proposed by Tumer and Agogino [2006], where agents learn to select from pre-computed routes in a single-commodity network.

Finally, another topic related to our objective is the study of performance degradation caused by the selfish behavior of individual road users—this remains an important research topic, as shown by Koutsoupias and Papadimitriou [1999] regarding the PoA problem.

## 6 Conclusions and Future Work

Traffic assignment and route choice are difficult learning problems because the routes available for agents may be highly coupled. Furthermore, the price of anarchy might be strongly affected by issues such as the topology of the network, the demand distribution, and the nature of the latency functions, among other factors. In this paper we presented the

first steps towards the definition of statistics and metrics that quantify how difficult the traffic assignment is, in general—and in particular how difficult the computation of an UE is.

In this work we assumed that agents trying to reach an UE learn to select routes independently. As a consequence, their learning processes can be negatively affected by the non-stationarity intrinsic to MARL settings. To counteract this effect, it might be beneficial to bias agents' decisions in order to accelerate convergence towards less coupled routes. We have shown how to define and compute such a coupling metric and experimented with using the coupling statistics as initial Q-values; our objective was to give agents initial incentives to select routes that are less coupled. Preliminary results show that this strategy can lead to faster convergence.

Ongoing work is being developed to improve the biasing strategy. We are also working towards evaluating the proposed approach in more complex networks, such as the Sioux Falls network and the Braess graphs with higher  $p$  values. Since the Sioux Falls network has 528 commodities, it may be time consuming to compute couplings for, say, all  $k = 4$  routes for each of the commodities. We plan to use previous results from Chudak *et al.* [2007] to pre-select routes that contain the most congested arcs, and concentrate our analyzes on them. Finally, given that we do have prior information about agents' preferences for reaching particular nodes of the network (i.e., we do know  $r_\sigma$  for each commodity), we can also define a variant of the coupling statistic  $\Psi$  which is weighted by these preferences. We also plan to extend the learning process to a state-based one, such as that described in Bazzan and Grunitzki [2016], where agents learn to select an *arc* (action) at each *node* (state) of the network that is visited. This contrasts with our current model, where agents learn to select a complete pre-defined route among  $k$  options.

## Acknowledgments

Fernando Stefanello was supported by CAPES/PNPD.  
Ana Bazzan is partially supported by CNPq.

## References

- Michael Balmer, Nurhan Cetin, Kai Nagel, and Bryan Raney. Towards truly agent-based traffic and mobility simulations. In N.R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi Agent Systems, AAMAS*, volume 1, pages 60–67, New York, USA, July 2004. New York, IEEE Computer Society.
- Ana L. C. Bazzan and Camelia Chira. Hybrid evolutionary and reinforcement learning approach to accelerate traffic assignment (extended abstract). In R. Bordini, E. Elkind, G. Weiss, and P. Yolum, editors, *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 1723–1724. IFAA-MAS, May 2015.
- Ana L. C. Bazzan and Ricardo Grunitzki. A multiagent reinforcement learning approach to en-route trip building. In *To appear in 2016 International Joint Conference on Neural Networks (IJCNN)*, 2016.
- Martin Beckmann, C. B. McGuire, and Christopher B. Winsten. *Studies in the economics of transportation*. Technical report, Yale University Press, 1956.
- Moshe Ben-Akiva and Michel Bierlaire. Discrete choice methods and their applications to short term travel decisions. In *Proceedings of the International Series in Operations Research & Management Science*, pages 5–33. Springer, 1999.
- D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258, 1968.
- Bureau of Public Roads. *Traffic Assignment Manual*. US Department of Commerce, Urban Planning Division, Washington D.C., 1964.
- Ennio Cascetta, Agostino Nuzzolo, Francesco Russo, and Antonino Vitetta. A modified logit route choice model overcoming path overlapping problems. specification and some calibration results for interurban networks. In *Proceedings of the 13th International Symposium on Transportation and Traffic Theory*, pages 697–711, 1996.
- Fabian A. Chudak, Vania Dos Santos Eleuterio, and Yurii Nesterov. Static traffic assignment problem. a comparison between Beckmann (1956) and Nesterov & de Palma (1998) models. In *Proceedings of the 7th Swiss Transport Research Conference*, pages 1–22, sep 2007.
- Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998.
- H. Dia and S. Panwai. *Intelligent Transport Systems: Neural Agent (Neugent) Models of Driver Behaviour*. LAP Lambert Academic Publishing, 2014.
- F. Klügl and Ana L. C. Bazzan. Route decision behaviour in a commuting scenario. *Journal of Artificial Societies and Social Simulation*, 7(1), 2004.
- Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science (STACS)*, pages 404–413, Berlin, Heidelberg, 1999. Springer-Verlag.
- Juan de Dios Ortúzar and Luis G. Willumsen. *Modelling Transport*. John Wiley & Sons, 3rd edition, 2001.
- Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.
- Tim Roughgarden. Designing networks for selfish users is hard. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 472–481. IEEE, 2001.
- K. Tumer and A. Agogino. Agent reward shaping for alleviating traffic congestion. In *Workshop on Agents in Traffic and Transportation*, Hakodate, Japan, 2006.
- John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institution of Civil Engineers*, volume 1, pages 325–362, 1952.
- Jin Y. Yen. Finding the  $k$  shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.



# Combining Car-to-Infrastructure Communication and Multi-Agent Reinforcement Learning in Route Choice

Ricardo Grunitzki and Ana L. C. Bazzan

Instituto de Informática

Universidade Federal do Rio Grande do Sul

Porto Alegre, RS, Brazil

{rgrunitzki, bazzan}@inf.ufrgs.br

## Abstract

Route choice is an important stage in transport planning and modeling. Most of the existing approaches do not consider that road users can nowadays consult new technologies to plan their routes. In this paper, we combine multi-agent reinforcement learning (MARL) and car-to-infrastructure communication (C2I) to deal with route choice. The agents (road users) and the infrastructure interact with each other to exchange traffic information about the road network. The agents send the travel cost of the edges they crossed to the infrastructure. The infrastructure uses these costs to compute shortest paths, which are transmitted to the agents when requested. The agents use such received shortest path to update their knowledge base. The obtained results are compared against a classical MARL approach that does not use C2I communication. Experimental results show that our approach overcomes the compared method in terms of average travel cost.

## 1 Introduction

Route choice is an important stage in the classical transport planning and modeling [Ortúzar and Willumsen, 2011]. Route choice methods select routes and assign them to road users, aiming to connect their individual origins with their destinations. The output of these methods describes the state of the transportation system, which is a relevant input for testing the consequences of changes in the physical infrastructure of the network. Most of the methods found in the literature assume the existence of a central authority that computes and allocates routes for the road users. In real scenarios, such assumption is not valid because a system manager cannot directly control the behavior of the road users in terms of route choice.

The rapid diffusion of intelligent transportation systems (ITS) enables road users to take into account the traffic information available on ITS to help them in their route choice process. Such information can be acquired from several sources such as inductive loops, video vehicle detection, GPS devices, etc. From this information, it is possible to compute estimated shortest routes, which are then recommended to

road users. If many road users decide to follow the recommended routes, they may overload those routes causing jams and increased travel times. This problem gets even worse when there are several ITS (e.g., route guidance systems, car-to-infrastructure-based systems, etc.) recommending routes to road users. Such systems have no control over the total flow that will be redirected to the suggested routes because the real-world road users have their own beliefs about which route they should follow. Therefore, the correct use of available traffic information is still an open problem.

The present work combines multi-agent reinforcement learning (MARL) and car-to-infrastructure (C2I) communication to model the behavior of modern road users (agents), which may use traffic information provided by an ITS to plan their routes. The ITS assumes the existence of communication devices installed over the network. The communication devices and agents can exchange traffic information with each other. Traffic information represents the cost of traveling some path over the road network. The agents are implemented as independent learners and behave competitively in the system, i.e., each agent attempts to minimize his own travel cost, regardless of the consequences his actions on other agents. The agents have full autonomy to decide which route to follow. However, they can count on traffic information provided by the infrastructure to support their decision-making process. The infrastructure uses the travel costs observed by the agents during their trip to estimate the shortest paths that can be transmitted to the agents. We compared our approach to a MARL one that does not assume the exchanging of traffic information provided by a C2I model. Experimental results showed that present approach overcomes other method in terms of average travel cost.

This paper is organized as follows. The route choice problem is defined in Section 2. The related works is presented in Section 3. In Section 4, we present the infrastructure modeling (Section 4.1) and agent modeling (Section 4.2). The experimental results are discussed and analysed in Section 5. Final remarks and future directions are presented in Section 6.

## 2 Route Choice in Transportation Systems

A transportation system is composed of two parts: demand and supply. The demand represents the users of the infrastructure (referred to road users, trips or vehicles). The demand can be represented by an origin-destination matrix

(OD-matrix). An OD-matrix  $T$  contains  $I$  lines (origin zones) and  $J$  columns (destination zones). Each element  $T_{ij}$  represents the amount of trips from vertex  $i$  to  $j$  in a given time interval. It is said that  $i \in I$  and  $j \in J$  is an OD-pair.

The second part of the transportation system, the supply, represents the road network and can be modeled as a directed graph  $G = (V, E)$ , where  $V$  is a collection of nodes, and  $E$  is a collection of directed edges. An edge  $e \in E$  is represented as a two-element subset of  $V$ :  $e = \{u, v\}$  for some  $u, v \in V$ , where  $u$  is the *origin* and  $v$  is the *destination* node of  $e$ . The set of incoming edges of node  $v \in V$  is defined by the set of edges  $E^-(v) : \{e \in E | e = \{u, v\} \wedge u \in V\}$ . Each edge  $e$  has a travel cost  $c_e$  associated to its crossing—for instance, the cost can be travel time, fuel spent, travel distance, and so on. As route choice is usually done in a macroscopic way due to the simplicity of implementation, the cost of crossing an edge is abstracted by a function. Volume-delay functions (VDF) are well-known abstractions for this purpose. An example of a VDF is the one suggested by Bureau of Public Roads (BPR) [Bureau, 1964] in Equation 1, where  $c_e$  represents the travel time, in minutes, for traveling edge  $e$ ;  $c_e^f$  is the travel time per unit of time under free-flow conditions (free-flow travel time);  $f_e$  is the volume of vehicles (in vehicles per unit of time) using the edge  $e$ ;  $C_e$  is the edge capacity; and  $a$  and  $b$  are parameters specifically defined for each edge. A path (or route)  $p = \{v_1, v_2, v_3\}$  is defined by a set of connected edges. The cost of  $p$  is the sum of the costs of all edges of  $p$ .

$$c_e = c_e^f \left[ 1 + a \left( \frac{f_e}{C_e} \right)^b \right] \quad (1)$$

Route choice (or, alternatively, route/traffic assignment) methods connect supply and demand, respecting the restrictions of origin and destinations present on OD-matrices [Ortúzar and Willumsen, 2011]. In studies of route choice, network equilibrium models are commonly used for the estimation of traffic patterns on scenarios that are subject to congestion. Wardrop’s first principle [Wardrop, 1952] is one of the most accepted principles of equilibrium, and states that: “no road user can unilaterally reduce his/her travel costs by shifting to another route”. This is also known as user equilibrium (UE). In this paper, the UE is used to assess the quality of the solutions obtained by our approach.

### 3 Related Work

Route choice is an extensively studied field of research. The Frank-Wolfe algorithm [Frank and Wolfe, 1956] is a classical algorithm still often used to deal with optimization problems where the objective function is convex and the constraints of the problem are linear. The adaptation of the Frank-Wolfe algorithm for the calculation of UE in route choice problems is originally presented in [LeBlanc *et al.*, 1975]. This algorithm focuses on the computing of UE in large-scale scenarios. They consider the existence of a central authority responsible for computing and assigning routes for the road users. This approach does not assume the road drivers can change their route along the trip. The present paper focuses on mod-

eling the individual decision-making of the road users under the presence of traffic information.

Multi-agent systems are often used in decentralized approaches for route choice [Ramos and Grunitzki, 2015; Dia and Panwai, 2007; Klügl and Bazzan, 2004]. In these approaches, road users have the autonomy to decide which route to take. In [Tumer *et al.*, 2008], a MARL approach that stimulates the cooperation between agents is presented. The agent’s task is to learn the best route from a set of pre-computed routes. In the present work, the agents learn their route during the trip (en-route mechanism). This makes the learning task harder because the search space is significantly increased. A MARL approach that stimulates cooperation between road users, but in an en-route perspective is presented in [Grunitzki *et al.*, 2014]. In each of these MARL approaches mentioned here, the exchanging of traffic information is not considered. The agents learn their routes according to the knowledge they acquire during the episodes. The present paper uses a C2I-based system to simulate the behavior of real road users that use traffic information to support their decision-making process.

Existing route guidance systems provide only route guidance after congestions happen. Some approaches that propagate the traffic flow in the route guidance system according to route intentions of the agents are presented [Claes *et al.*, 2011; Wang *et al.*, 2014]. In [Wang *et al.*, 2014], the authors propose a C2I system in an en-route perspective for shortest routes. In [Cao *et al.*, 2016], there are agents situated over the network junctions collecting the road users’ intentions, in order to update the route guidance system. These approaches assume that agents follow the requested route, which is used to propagate the flows of road users on the network. However, in practice, this cannot be assumed because each road user has his own motivations to make his choices. The present paper focuses on modeling the behavior of modern road users, which makes use of available traffic information only to support their decisions.

## 4 Approach

MARL C2I-based approach is composed of two kinds of entities: agents and communication devices, as illustrated in Figure 1. The agents represent cars, whilst the communication devices represent the infrastructure. During the execution of the method, agents and infrastructure can interact with each other in order to exchange traffic information.

The learning is organized in episodes and time steps. A time step represents the time needed by the agent to execute an action, in this case, traveling a given edge. An episode represents one trial, in which all agents start their learning process in their initial state and make successive interactions with the environment until reaching their final state (destination). An episode ends when all agents have reached their final state. These two concepts, episodes and time steps, are important to understanding the moment in which the entities can interact with each other. In the following sections, the modeling of agents and infrastructure is presented in detail.

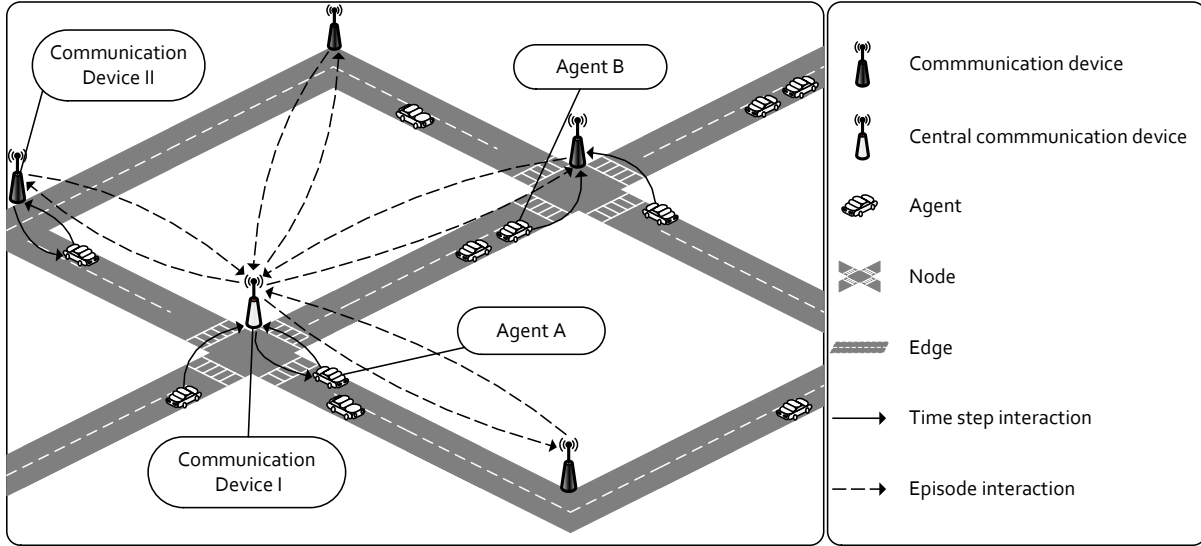


Figure 1: Interaction scheme between agent and infrastructure.

#### 4.1 Infrastructure Modeling

The infrastructure is composed of a set of communication devices,  $D$ , distributed over the network. Each node  $v \in V$  is associated with one communication device  $d_v \in D$ . As shown in Figure 1, every node (junction) has a communication device physically installed. One of the communication devices is called central communication device (communication device I, in Figure 1) because it has the extra responsibility of: i) concentrating traffic information for all edges; ii) computing all shortest paths; and iii) transmitting the computed shortest paths back to other devices.

In present work, the word *agent* is not used to refer to the communication devices. This avoids possible confusion between: learning agents (road users) and nonlearning agents (communication devices). For this reason, whenever the word *agent* is used, it will be referring to a road user.

The communication between agents and communication devices is modeled as a two-way dedicated short-range communication system (DSRCS). Note that there are two kinds of interactions, represented by dashed and full lines in Figure 1. These lines represent the sending of a message from a sender to a receiver entity. Full lines represent the communication between agents and communication devices. This kind of interaction can occur once at each time step of an episode, as illustrated in Figure 1. In a single message, an agent can send traffic information and also request a shortest path. On the other hand, the dashed lines represent the interactions between the communication devices and the central communication device. These interactions always occur at the beginning of each episode, before agents start their trips.

The communication between agents and infrastructure is only possible when they (agents and infrastructure) are topologically close. Agents cannot communicate with each other. This makes the system simple because it dispenses the need for a channel between an agent and a central authority that

represents an infrastructure.

In short, a communication device  $d$  plays the following roles: i) storing local information about the travel cost of all incoming edges of the node in which  $d$  is situated; ii) computing the estimated shortest paths from its node to all other node of the network; iii) exchanging travel information to agents. In the following we detail these roles.

#### Information Storing

Each communication device  $d$  is responsible for storing traffic information about the incoming edges of a node  $v_d$ . The traffic information is communicated to the agents that cross the edges with destination node  $v_d$ . When an agent has crossed an edge  $e = \{u, v\}$ , he perceives the travel cost  $c_e$  on  $e$ . This cost is communicated to the communication device  $d_v$  when the agent arrives at node  $v$ . The communication device  $d$  updates its knowledge base with this traffic information about  $e$ . During the execution, many vehicles cross the incoming edges of a given node. So, the communication device of this node needs to update its knowledge very often throughout the episode's steps.

The C2I communication enables the communication between agent and infrastructure when they are nearby. Besides that, the agent can measure the travel cost of the edges he crossed. The traffic information is measured by the agents instead of the one provided by sensors in order to make the system simpler. The use of local sensors, such as inductive loops or cameras, has the disadvantage to need physical mechanisms distributed along the edges. Besides that, they require a specific communication channel to transmit the observed traffic information to the communication device.

#### Computing estimated shortest paths

Each communication device can send to the agents the estimated shortest path from its current node to the destination node of the agents, as illustrated by the communication device



II, in Figure 1. The weights of the edges are estimated based on travel cost the communication devices have in their knowledge base. At the end of each episode—when all agents finish their trip—, all communication devices transmit their traffic information to the central communication device. The central communication device (communication device I, in Figure 1) uses the Floyd-Warshall algorithm [Warshall, 1962] for finding the shortest paths of the network. In a single execution, the algorithm finds the costs of the shortest paths between all pairs of nodes. The output of this algorithm is used to recursively compute the set of edges that represents each shortest path. After that, the central communication device sends to all other communication devices the estimated shortest paths. The term *estimated* is used because the real travel cost, in a next episode, may change due to the actual actions performed by the agents.

### Exchanging traffic information

When an agent is close to a communication device, he can request traffic information. In Figure 1, the agent A requests a shortest path to his destination node  $v_{s_+} \in V$ . The communication device finds a route  $r = \{e_0, \dots, e_n\}$  in its knowledge base, where  $e_0$ 's origin node is  $v_d \in V$ ; and  $e_n$ 's destination node is the agent's destination  $v_{s_+}$ . This route is then transmitted to the agent. Every time such information is requested, the communication device sends it to the agent. How the information is used by the agent is explained in the next section. Here, the interest is in showing how the iterations between agent and communication device work.

## 4.2 Agents Modeling

The learning agents (vehicles/road users) are implemented as independent learners, through multiple independent learners technique [Buşoniu *et al.*, 2008]. Consequently, the agent's decision-making process ignores the existence of other agents. This is needed because transportation systems may have thousands or millions of agents interacting. In such condition, the use of join-action learners is infeasible, as remarked by [Tuyls and Weiss, 2012].

The learning task of each agent is to build a route that connects its origin to its destination and minimizes its travel costs. The routes are built dynamically along the trip (en-route learning). Compared to approaches that use pre-established sets of precomputed routes that connect agent's origin to destination (route-based learning) [Tumer and Agogino, 2006], the current formulation is harder to be handled by the MARL. In route-based approaches, the search space is restricted by the number of routes presents in the precomputed set of routes. In en-route approaches, as in the current paper, the search space is restricted by the set of valid routes between one OD-pair, which grows according to the size and topology of the network. However, compared to route-based learning approaches, the en-route approach has the following advantages: i) it does not require the input of the initial subset of routes; and ii) it does not restrict the agent search space, enabling them to explore any feasible route (not only those pre-given).

The decision-making process of agents is modeled as a finite Markov decision process (MDP), which is composed of

a set of states  $S$  and a set of actions  $A$ . For each pair state-action  $Q(s, a)$  there is a  $Q$ -value associated to it. The  $Q$ -values represent how good the expected future reward is following a given state-action transition. The goal in an MDP is to find the sequence of transitions (policy) that maximizes the reward of the agent over its lifetime. In our approach, an agent's state  $s \in S$  represents the node  $v \in V$ , in which he is situated. The set of actions  $A$  represents the edges  $e \in E$ . The set of actions in a state  $s$ ,  $A(s)$ , is represented by the set of outgoing edges  $E^+(v_s)$ . The reward function is defined by  $R(s, a) = -c_{e_a}$ , which represents the travel cost of edge  $e$ . The reinforcement learning algorithm used to update the  $Q$ -values  $Q(s, a)$  is  $Q$ -Learning [Watkins and Dayan, 1992], given in Equation 2, where  $\alpha$  is the learning rate;  $\gamma$  is the discount factor; and  $s'$  is the resulting state of being in state  $s$  and taking the action  $a$ .

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left( r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \quad (2)$$

At each time step, agents can interact with the infrastructure aiming at: i) send traffic information about their crossed edges to a communication device; and ii) request the shortest path from their current node to their destination. When an agent crosses an edge, he observes its travel cost and automatically communicates it to the infrastructure (item i), as illustrated by the agent B, in Figure 1. The shortest path request (item ii), can be realized at each time step, with a probability  $0 \leq \tau \leq 1$ , as illustrated by the agent A. A high value of communication rate,  $\tau \rightarrow 1$ , makes the agents update their MDP very often along the episode, while low values,  $\tau \rightarrow 0$ , makes the agents not update their MDP with the traffic information provided by the infrastructure.

When an agent requests a route, he transmits his destination node to a given communication device. The communication device returns to the agent a shortest path  $p$ . This shortest path connects the node in which the communication device is installed to the destination of the agent. When the agent receives a shortest path, he needs to update his MDP according to the travel cost of  $p$ . This cost must be comparable with the other  $Q$ -values of the MDP, which represent the expected discounted reward that the agent may receive following a given pair state-action. In the present route choice approach,  $Q$ -values represent the expected discounted travel cost from a given node to a destination node. The travel cost is discounted by  $\gamma$ , according to  $Q$ -learning update rule. Thus, we use the Bellman equation [Bellman, 1957], presented in Equation 3, to evaluate the  $Q$ -value of a given route  $p = \{v_0, v_1, \dots, v_n\}$ , where  $s$  represents the vertex  $v_0$ ;  $a$  is the action that represents the edge  $e = \{v_0, v_1\}$ ;  $s'$  is the state that represents vertex  $v_1$ ; and  $a'$  the action that represents the edge  $e' = \{v_1, v_{1+1}\}$ . This equation expresses a relationship between the value of the edges that connect the nodes of a given route.

$$Q^p(s, a) = r(s, a) + \gamma Q^p(s', a') \quad (3)$$

In this en-route mechanism, even if an agent receives a shortest path, he only will follow it if the action selection

strategy select it. The action selection is given by the  $\epsilon$ -decreasing strategy given by Equation 4, where the exploration probability is initialized by  $\epsilon_0$  and exponentially decreases along the episodes  $\lambda \in \Lambda$ , by a factor  $D$ . In this manner, agents choose actions randomly (exploration) with probability  $\epsilon$ , and greedily (exploitation) with probability  $1 - \epsilon$ . The selection of random actions is used to stimulate agents to explore the travel time of other possible routes. Once such actions detect attractive edges, this information could be propagated to the other agents through the infrastructure.

$$\epsilon_\lambda = \epsilon_0 D^\lambda \quad (4)$$

## 5 Experiments

### 5.1 Scenario

We evaluate our approach in a well-known transportation problem presented in the literature, called scenario Sioux Falls (SF). Although it is inspired by the city of Sioux Falls, USA, it is not considered a realistic scenario. All data sets containing network, demand, and cost function are available at <https://github.com/bstabler/TransportationNetworks>. The demand is comprised by 360600 trips distributed among 528 OD-pairs. The road network, presented in Figure 2, has 24 vertices and 76 edges. The numbers in the edges represent their travel time under free-flow condition, in both directions. The cost function of this scenario is defined by the VDF proposed by the Bureau of Public Road[Bureau, 1964], shown in Equation 1. The parameters  $a$  and  $b$  are defined in 0.15 and 4, respectively, as suggested by [LeBlanc *et al.*, 1975].

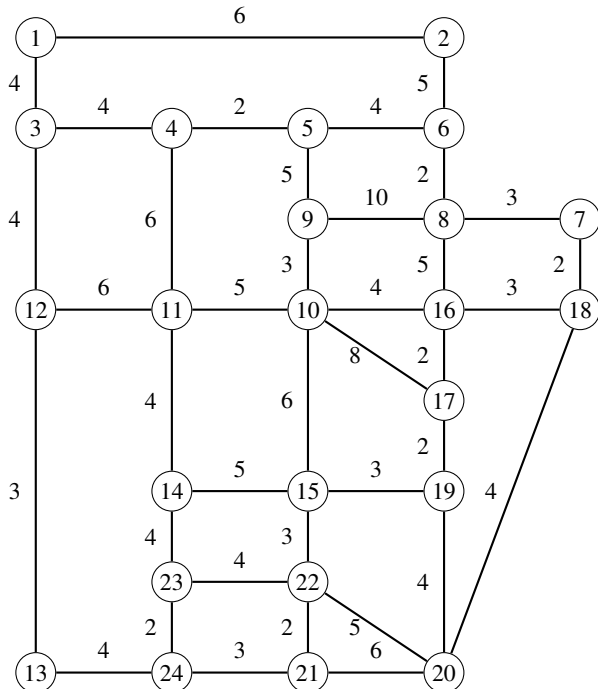


Figure 2: Road network topology of scenario SF.

Relevant aspects of the SF scenario are summarized in Table 1. The presented average travel time (ATT) under user equilibrium were obtained using the Frank-Wolfe algorithm. It is important to remark that the algorithm produces an approximation to the UE. This is used in this paper to assess how close to the UE the proposed approach can get.

Table 1: Relevant aspects of scenario SF.

Feature	Scenario SF
trips	360600
OD-pairs	528
vertices	24
edges	76
cost function	VDF-BPR
ATT under UE	$\approx 20.76$

### 5.2 Numerical Results

The  $Q$ -Learning algorithm has some parameters to be set: the learning rate ( $\alpha$ ), the discount factor ( $\gamma$ ), and the exploration rate ( $\epsilon$ ).

The learning rate and the discount factor used in all experiments of present work were empirically found:  $\alpha = 0.9$  and  $\gamma = 0.99$ . The discount factor plays a major role than the learning rate in route choice. This can be explained by the fact that action selection (outgoing edges) are very important in this problem since learning aims at minimizing the travel cost in the whole route. For this reason, a high discount factor must be used. The learning rate is also high due to the stochastic characteristics of the environment. This makes the agent override the old information with a greater proportion of the most recent information.

The exploration strategy used in this work starts with a high probability of exploration which is reduced along the episodes in order to enable agents to exploit more and more. The exploration rate starts at  $\epsilon_0$  and decreases exponentially by a factor  $D$  at each learning episode. The multiplicative factor must be set to fit the simulation horizon. In this paper, we used 1000 episodes,  $\epsilon_0 = 1.0$ , and  $\lambda = 0.99$ . As will be shown, not all combinations of parameters need to be run for 1000 episodes because the convergence for some route choice pattern is reached much earlier. However, for uniformity, the same value for episodes (1000), initial exploration (1.0) and multiplicative factor (0.99) are used in all cases.

Our approach has an extra parameter to be defined, the communication rate ( $\tau$ ). This parameter represents the probability of an agent to request an information from the infrastructure during his decision-making process. As mentioned before, the  $Q$ -Learning performance is directly related to the balance between exploration and exploitation defined for the action selection rule. In our approach, the key parameter that must be set is the communication rate ( $\tau$ ). We tested some combination of values for  $\tau$  in order to find the best one. The space of possible values is discretized in  $\tau = \{0, 0.25, 0.5, 0.75, 1\}$ . Thus, we can evaluate the effects of zero (0%), low (25%), medium (50%), high (75%), and full (100%) communication during the action selection. All

result presented in this paper represent the average of 30 repetitions.

The results for the scenario SF are presented in Table 2. The baseline used for the sake of comparison is the  $\tau = 0$  configuration. This is equivalent to the application of  $Q$ -Learning for the route choice problem, without C2I communication. The obtained results for  $\tau = 0$  show that the baseline cannot converge to the approximate UE ( $\approx 20.76$  minutes). The baseline solution is 1.14 minutes worse than the UE condition. In the presence of no communication with the infrastructure, agents have no way to identify whether an action is good nor not, except through experimenting it. As the environment is highly dynamic due to the large number of agents who take actions simultaneously and generating noise in the MDP of the other agents, this is difficult to the MARL algorithm to converge to most appropriated policies.

Table 2: Average travel time (ATT) and standard deviation (SD) for different values of  $\tau$ .

$\tau$	0	0.25	0.5	0.75	1
ATT	21.9	21.265	21.337	21.358	33.941
SD	0.131	0.053	0.075	0.066	2.759

The results for  $\tau > 0$  represent the obtained solutions of the present approach. Our approach yields better results when  $0.25 \leq \tau \leq 0.75$ . For  $\tau = 1$ , the MARL converges to inadequate solutions. High values of  $\tau$  make agents update their knowledge base quite often. As consequence, every agent has the information about the most attractive route known by the infrastructure. Even if the route's cost are based on historical information, each agent will have a high probability to choose the route that is known by the other agents as the most attractive one. In congested scenarios like the SF one, such behavior makes agents compete for the edges of most attractive paths. Consequently, they overload some routes, whereas others are being underutilized. Low values of  $\tau$  reduce the competition by the most attractive edges and enables agents to better utilize the knowledge they acquire by experiencing the environment. It reduces the noisy knowledge present in their MDP and allows them to better balance their experienced knowledge with the knowledge acquired from infrastructure.

A t-test with 95% of confidence interval was conducted for all distributions present in Table 2. The conclusion is that the proposed approach is better than the baseline for values of  $\tau = \{0.25, 0.5, 0.75\}$ . The best ATT yield by our approach is obtained with  $\tau = 0.25$ , which is 0.63 minutes better than the baseline. Note that even for  $\tau = 0.75$ , the proposed approach yields an ATT better than the baseline. In the baseline, agents receive only the feedback from the environment (reward) and it is related to the action they choose. It is hard for them to make good decisions when there are too many agents generating noise in their MDP and due to the high probability they have to select random actions in the early episodes. Such noise can make the agent understand that a given action is bad, when actually it is convenient for reaching his objective. In the proposed approach, the traffic information provided by the infrastructure is able to be fixed in the upcoming episodes,

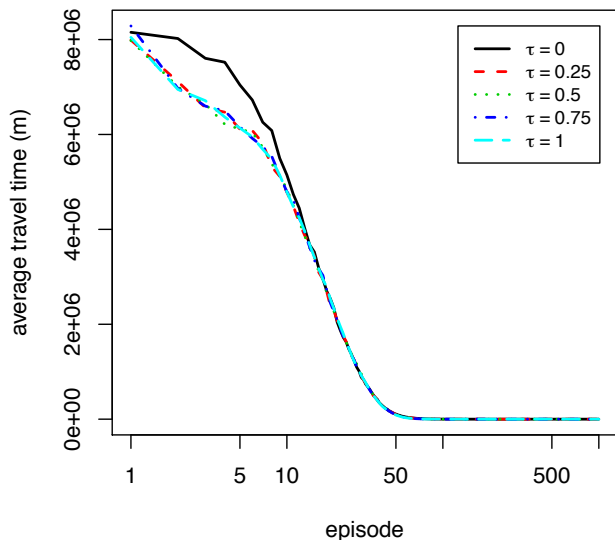


Figure 3: Performance vs. time on scenario SF

making the agent reconsider such action while building his behavior. On the other hand, the excess of traffic information provided by  $\tau = 1.0$  can result in poor performance for the system.

In the next experiment, we demonstrate the convergence speed of the proposed approach compared to the baseline. Figure 3 shows the performance (in terms of ATT) along the episode for each value of  $\tau$  evaluated. Note that for  $\tau \geq 0.25$  all curves have a similar shape. In initial episodes, our approach presents learning curves steeper than the baseline. This is explained by the traffic information that the agents receive, which is capable of guiding them to their destination faster. In the baseline, the agents may drive in a looping manner due to the bad actions they take.

The ATT in the initial episodes is quite high due to the characteristics of the cost function. Since the travel time grows exponentially according to the flow (see Equation 1), when the flow exceeds the edge capacity, the travel time of the edge grows rapidly. This condition, associated with the large demand taking suboptimal actions in the early episodes, makes the ATT be high in early episodes of all cases.

## 6 Conclusions and Future Work

This paper combines MARL and car-to-infrastructure (C2I) communication in an approach for route choice. Road users (agents) and infrastructure can interact with each other in order to exchange traffic information about the road network. The traffic information is provided by a C2I intelligent transportation system, in which agents can request traffic information whenever they want.

We evaluated our approach on a classic scenario present in the literature. The obtained results were compared against a MARL approach for route choice, without C2I communication. The obtained results show the proposed approach can

overcome the compared method when the frequency of use of traffic information is properly set. In the experiments, the agents that use the traffic information very often may impair their travel time due to the large flow allocated in the most attractive routes. Reducing the frequency of use of traffic information allows the agents better exploit the knowledge gained on previous episodes, regardless of whether it has been acquired via C2I communication or experiencing the environment.

The present work focused on the combination of MARL and C2I communication. However, for its implementation to be feasible in the real world, limitations as the following must be addressed. The demand used in this paper is homogeneous in terms of individual preferences, i.e., all road users goal is to minimize their travel cost. However, in the real world, they also have personal preferences/restrictions associated with the trip, such as the avoidance of large roads, tolls or even the exposure of their trip information. Besides this, the road users exchange traffic information from a single source. However, in the real world, they may use multiple sources. In this kind of system, the traffic information may differ from one system to other according to the mechanisms they use to get and manipulate it. The effects of multiple traffic information systems interacting with the agents must be investigated. The evaluation of different strategies to balance exploration and exploitation, such as the ones that weight the random actions according to its quality, must be conducted in order to speed up the convergence. Finally, a comparison against communication-based approaches available on literature must be conducted.

## Acknowledgements

R. Grunitzki is supported by CAPES. A. L. C. Bazzan is partially supported by CNPq (grant 305062).

## References

- [Bellman, 1957] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.
- [Bureau, 1964] of Public Roads Bureau. *Bureau of Public Roads: Traffic Assignment Manual*, 1964.
- [Buşoniu *et al.*, 2008] L. Buşoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.
- [Cao *et al.*, 2016] Zhiguang Cao, Hongliang Guo, Jie Zhang, and Ulrich Fastenrath. Multiagent-based route guidance for increasing the chance of arrival on time. In *30th AAAI Conference on Artificial Intelligence (AAAI)*, Phoenix, Arizona, USA, 2016.
- [Claes *et al.*, 2011] Rutger Claes, Tom Holvoet, and Danny Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. *IEEE Transactions on Int. Transp. System*, (99), March 2011.
- [Dia and Panwai, 2007] H. Dia and S. Panwai. Modelling drivers' compliance and route choice behaviour in response to travel information. *Special issue on Modelling and Control of Intelligent Transportation Systems, Journal of Nonlinear Dynamics*, 49(4):493–509, 2007.
- [Frank and Wolfe, 1956] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.
- [Grunitzki *et al.*, 2014] Ricardo Grunitzki, Gabriel de O. Ramos, and Ana L. C. Bazzan. Individual versus difference rewards on reinforcement learning for route choice. In *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, pages 253–258, Oct 2014.
- [Klügl and Bazzan, 2004] F. Klügl and Ana L. C. Bazzan. Route decision behaviour in a commuting scenario. *Journal of Artificial Societies and Social Simulation*, 7(1), 2004.
- [LeBlanc *et al.*, 1975] Larry J LeBlanc, Edward K Morlok, and William P Pierskalla. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research*, 9(5):309–318, 1975.
- [Ortúzar and Willumsen, 2011] Juan de Dios Ortúzar and Luis G. Willumsen. *Modelling transport*. John Wiley & Sons, Chichester, UK, 4 edition, 2011.
- [Ramos and Grunitzki, 2015] Gabriel de Oliveira Ramos and Ricardo Grunitzki. An improved learning automata approach for the route choice problem. In Fernando Koch, Felipe Meneguzzi, and Kiran Lakkaraju, editors, *Agent Technology for Intelligent Mobile Services and Smart Societies*, volume 498 of *Communications in Computer and Information Science*, pages 56–67. Springer Berlin Heidelberg, 2015.
- [Tumer and Agogino, 2006] K. Tumer and A. Agogino. Agent reward shaping for alleviating traffic congestion. In *Workshop on Agents in Traffic and Transportation*, Hakodate, Japan, 2006.
- [Tumer *et al.*, 2008] Kagan Tumer, Zachary T. Welch, and Adrian Agogino. Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the 7th Int. Conference on Autonomous Agents and Multiagent Systems*, pages 655–662, Estoril, May 2008. IFAAMAS.
- [Tuyls and Weiss, 2012] K. Tuyls and G. Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–52, 2012.
- [Wang *et al.*, 2014] Shen Wang, Soufiene Djahel, and Jennifer McManis. A multi-agent based vehicles re-routing system for unexpected traffic congestion avoidance. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2541–2548. IEEE, oct 2014.
- [Wardrop, 1952] John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institution of Civil Engineers*, volume 1, pages 325–362, 1952.
- [Warshall, 1962] Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.
- [Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

# On Estimating Action Regret and Learning From It in Route Choice

Gabriel de O. Ramos and Ana L. C. Bazzan

Instituto de Informática  
Universidade Federal do Rio Grande do Sul  
{goramos,bazzan}@inf.ufrgs.br

## Abstract

The notion of regret has been extensively employed to measure the performance of reinforcement learning agents. The regret of an agent measures how much worse it performs following its current policy in comparison to following the best possible policy. As such, measuring regret requires complete knowledge of the environment. However, such an assumption is not realistic in most multiagent scenarios. In this paper, we address the route choice problem, in which each driver must choose the best route between its origin and its destination. The expected outcome corresponds to an equilibrium point in the space of policies where no driver benefits from deviating from its policy, a concept known as User Equilibrium (UE). Considering the limited observability of such a scenario, we investigate how the agents can estimate their regret based exclusively on their experience. To this regard, we introduce the concept of estimated action regret, through which an agent can estimate how much worse it performs by taking a given action rather than the best in hindsight. Additionally, we show how such estimations can be used as a reinforcement signal to improve their performance. We empirically evaluate our approach in different route choice scenarios, showing that the agents produce reasonable estimations of their regret. Furthermore, we show that using such estimations as the reinforcement signal provides good approximations to the UE.

## 1 Introduction

Reinforcement learning (RL) in multiagent domains is a challenging task. In RL, an agent must learn by trial-and-error how to behave within the environment in order to maximise its utility. When multiple agents share a common environment, they must adapt their behaviour to those of others. The problem becomes even harder when the agents are selfish and compete for a common resource. An example is the route choice problem, which concerns how rational drivers<sup>1</sup> behave when choosing routes between their origins and desti-

nations to minimise their travel costs. Learning is a fundamental aspect of route choice because the agents must adapt their choices to account for the changing traffic conditions. In other words, the agents must adapt to each others' decisions.

An interesting class of multiagent RL techniques comprises the regret minimisation approaches. In this context, regret has been typically employed to measure the performance of reinforcement learning agents. Specifically, regret measures how much worse an agent performs following its current policy in comparison to following the best possible policy one in hindsight. In this sense, regret minimisation can be seen as an inherent definition on how rational agents behave over time. Along these lines, the regret measure naturally fits as a guide of the learning process.

In recent works [Zinkevich *et al.*, 2008; Bowling and Zinkevich, 2012; Waugh *et al.*, 2015], regret has been used to improve the learning process. However, calculating regret requires complete knowledge of the environment (i.e., the utility associated with every possible policy). In fact, one may assume that an online service broadcasts the required information through mobile devices. Nevertheless, investigating methods to accomplish such a task in the absence of any global information is more challenging and is also relevant, especially in highly competitive scenarios like traffic [Bazzan and Klügl, 2013; Stone and Veloso, 2000].

In this paper, we address the route choice problem by minimising regret. Specifically, we investigate how the agents can estimate their regret locally (i.e., based exclusively on their experience) and how such estimations can be employed to guide the RL process. To this regard, each agent keeps an internal history of experienced rewards, which is used for estimating the regret associated with each of its actions. We refer to such measure as the *estimated action regret* and employ it for updating the agents' policies. The expected outcome corresponds to an equilibrium point in the space of policies in which no driver benefits from deviating from its policy. This is the so-called User Equilibrium (UE) [Wardrop, 1952]. To the best of our knowledge, this is the first attempt to improve the learning process by using regret estimations as the reinforcement signal.

Through experiments, we show that our approach provides fairly precise estimations of the agents' regret relying only on agents' experience. Moreover, we present good evidence that using such regret estimates as the reinforcement signal is

---

<sup>1</sup>Henceforth, we use the terms agent and driver alternately.

beneficial for the learning process. Consequently, in all tested cases, the results are reasonably close to the UE.

We remark that this work represents our very first step towards developing rational agents able to analyse their learning performance and to improve their expected outcome. In the medium-term, we aim at investigating formal aspects of the learning process to guarantee the efficiency of RL under multiagent domains.

This paper is organised as follows. The background on route choice, RL and regret algorithms is presented in Section 2. In Sections 3 and 4, we describe how the agents can estimate their regret locally and how they can learn using such estimations, respectively. The experimental evaluation is discussed in Section 5. Finally, Section 6 presents the concluding remarks and future work directions.

## 2 Background

### 2.1 Route Choice

The route choice problem concerns how rational drivers behave when choosing routes between their origins and destinations. In this section, we introduce the basic concepts related to route choice. For a more comprehensive overview, we refer the reader to [Ortúzar and Willumsen, 2011].

A road network can be represented as a directed graph  $G = (N, L)$ , where the set of nodes  $N$  represent the intersections and the set of links  $L$  represent the roads between intersections. The demand for trips generates a flow of vehicles on the links, with  $f_l$  the flow on link  $l$ . To this regard, each link  $l \in L$  has a cost  $c_l : f_l \rightarrow \mathbb{R}^+$  associated with crossing it. Let  $T_{ij}$  be the demand for trips between origin  $i \in N$  and destination  $j \in N$  (we refer to an origin-destination pair as simply OD pair). The set of all such demands is represented by an OD matrix  $T = \{T_{ij} \mid \forall i, j \in N, i \neq j, T_{ij} \geq 0\}$ . The total demand is denoted  $d = \sum_{T_{ij} \in T} T_{ij}$ . A trip is made by means of a route  $R \subseteq L$ , which is a sequence of links connecting an origin to a destination. The cost of a route  $R$  is given by  $C_R = \sum_{l \in R} c_l$ .

The cost of a link is typically modelled using the volume-delay function (VDF) abstraction. A VDF takes as input the flow of vehicles within a link and, based on its attributes (such as length and capacity), returns the cost (travel time) of such link. A simple VDF is presented in Equation (1), with  $t_l$  denoting the free flow travel time (i.e., minimum travel time, when the link is not congested). In this particular VDF, the travel time on link  $l$  is increased by 0.02 for each vehicle/hour of flow.

$$c_l(f_l) = t_l + 0.02 \times f_l \quad (1)$$

In the route choice process, drivers decide which route to take every day to reach their destinations. Usually, this process is modelled as a commuting scenario, where drivers' daily trips occur under approximately the same conditions. In this sense, each driver  $i \in D$ , with  $|D| = d$ , is modelled as an agent, which repeatedly deals with the problem of choosing the route that takes the least time to its destination. The utility  $u_i : R \rightarrow \mathbb{R}$  received by driver  $i$  after taking route  $R$  is inversely associated with the route's cost, as in Equation (2). The expected outcome of this problem can be described by

the Wardrop's first principle: "the cost on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route" [Wardrop, 1952]. Such a solution concept is known as User Equilibrium (UE). However, observe that the UE does not describe the system at its best operation. Indeed, such a state is only achieved when the average travel cost is minimum, as described by the Wardrop's second principle [Wardrop, 1952]. To this regard, such solution concept is commonly referred as System Optimum (SO).

$$u_i(R) = -C_R \quad (2)$$

### 2.2 Reinforcement Learning

Reinforcement learning (RL) concerns with how an agent learns a behaviour by reward and punishment from interactions with its environment. We can formulate the RL problem as a Markov decision process (MDP). An MDP is a tuple  $\langle S, A, T, r \rangle$ , where:  $S$  is the set of environment states;  $A$  is the set of actions;  $T : S \times A \times S \rightarrow [0, 1]$  is the state transition function, with  $T(s, a, s') = P(s' \mid s, a)$  representing the probability of reaching state  $s'$  after taking action  $a$  in state  $s$ ; and  $r : S \times A \rightarrow \mathbb{R}$  is the reward function, with  $r(s, a)$  denoting the reward received after taking action  $a$  in state  $s$  [Sutton and Barto, 1998].

In the context of the route choice problem, the actions of an agent represent the choice of routes between his origin and destination. Such a set of actions is known a priori by the agents. In this sense, the set of states and, consequently, the transition functions can be ignored. Moreover, we can define the reward received after taking action  $a$  as  $r(a) = u(R)$ , with  $a = R$  (we will refer to reward and utility, rather than cost, hereinafter). On this basis, we can model the route choice problem as a stateless<sup>2</sup> MDP.

Solving a stateless MDP involves finding a policy  $\pi$  (i.e., which route to take) that maximises the accumulated reward. When the model of the environment dynamics (i.e., the reward function  $r$ ) are known by the agent, finding such an optimal policy is trivial. However, this is rarely the case, especially in multiagent settings. To this regard, the agent must repeatedly interact with the environment to learn a model of its dynamics. A particularly suitable class of RL algorithms here comprises the so-called temporal-difference algorithms, through which an agent can learn without an explicit model of the environment.

The Q-learning algorithm is a good representative of temporal-difference methods [Watkins and Dayan, 1992]. In the case of a stateless MDP, a Q-learning agent learns the expected return  $Q(a)$  for selecting each action  $a$  by exploring the environment. The exploration of the environment must balance exploration (gain of knowledge) and exploitation (use of knowledge). A typical strategy here is  $\epsilon$ -greedy, in which the agent chooses a random action with probability  $\epsilon$  (exploration) or choosing the best action with probability  $1 - \epsilon$  (ex-

<sup>2</sup>Observe that a stateless MDP is equivalent to having an initial state with actions corresponding to the routes available to the agent, and an ending state with no actions. When an agent chooses action  $a$  on the initial state, it performs the desired action and reaches the ending state with probability 1, receiving reward  $r(a)$ .

ploitation). Usually,  $\epsilon$  starts with 1.0 and, at the end of each learning episode, it is multiplied by a decay rate  $\lambda$  in order to increase exploitation as the agent converges to its best action. After taking action  $a$  and receiving reward  $r(a)$ , the stateless Q-learning algorithm updates  $Q(a)$  using Equation (3), where the learning rate  $\alpha \in [0, 1]$  weights how much of the previous estimate should be retained. The Q-learning algorithm is guaranteed to converge to an optimal policy in the limit under certain assumptions.

$$Q(a) = (1 - \alpha)Q(a) + \alpha r(a) \quad (3)$$

### 2.3 Regret

The regret concept was introduced in the context of evaluating the performance of learning rules [Hannan, 1957]. Regret measures how much worse an agent  $i$  performs, on average, by following its current policy  $\pi_i \in \Pi$  as compared to following the best possible policy in hindsight. Precisely, the regret  $\mathcal{R}_i^T$  of agent  $i$  at time  $T$  is given by Equation (4), where  $r^t(a)$  represents the reward of action  $a$  at time  $t$  and, slightly abusing notation,  $\pi^t$  represents the action taken at time  $t$  under policy  $\pi$ . Therefore, regret can be seen as the average amount lost for following a policy other than the best one.

$$\mathcal{R}_i^T = \max_{\pi \in \Pi} \frac{1}{T} \sum_{t=1}^T r^t(\pi^t) - \frac{1}{T} \sum_{t=1}^T r^t(\pi_i^t) \quad (4)$$

In the context of reinforcement learning (RL), regret has been typically used as a measure of convergence [Shoham *et al.*, 2007; Buşoniu *et al.*, 2008; Zinkevich, 2003; Bowling and Veloso, 2002; Powers and Shoham, 2005; Banerjee and Peng, 2005]. An RL algorithm is said no-regret if it learns a policy  $\pi$  for which  $\mathcal{R}^T \rightarrow 0$  as  $T \rightarrow \infty$  [Hannan, 1957]. Along these lines, regret minimisation can be seen as a natural definition on how rational agents behave over time. In this paper, we use the regret measure to guide the learning process.

We remark that, by definition, computing regret assumes complete knowledge of the environment. Specifically, an agent cannot compute its regret without knowing the reward of every other action along time. Consequently, agents cannot (i) calculate their regret and (ii) learn using their regret, except if very strong assumptions are made (e.g., assuming that every agent knows the reward of all actions along time). Therefore, using the regret of Equation (4) to guide the learning process is not realistic in multiagent scenarios.

Zinkevich *et al.* introduced the concept of counterfactual regret and proposed an algorithm for minimising it [Zinkevich *et al.*, 2008]. The counterfactual regret is used to estimate the regret when the information about states is incomplete (useful in extensive form games with imperfect information). This is one of the first works to include the regret in the learning process. Subsequently, Waugh *et al.* employed a regression algorithm to provide enhanced estimates on the counterfactual regret [Waugh *et al.*, 2015]. However, these works assume that the regret is known by the agents, which is unrealistic for the problem we are considering.

In [Bowling and Zinkevich, 2012], the authors propose a graph representation to express the relation between actions and the associated regret. Such a representation was employed to mimic the structure of local search methods, thus

allowing no-regret algorithms to minimise a broader class of optimisation problems. Nevertheless, their work also assumes that the utility function is available to the agents.

Powers and Shoham proposed a set of performance criteria regarding multiagent learning and proposed an algorithm that meets such criteria [Powers and Shoham, 2005]. Their algorithm, however, makes some strong assumptions regarding the environment observability (e.g., an agent can observe its opponents' rewards). Banerjee and Peng extended Powers and Shoham's approach to a class of no-regret algorithms and dropped some of the observability assumptions [Banerjee and Peng, 2005]. Notwithstanding, these approaches do not employ the regret to guide the learning process.

Along these lines, in this paper we investigate how agents can *estimate* their regret based on their experience and propose the use of such estimations to guide the agents' learning process. Moreover, we disaggregate the regret formulation by measuring the regret of actions rather than of policies. We show that performing such estimates is realistic and improves the learning process. To the best of our knowledge, this is the first effort towards addressing regret estimation and learning through such regret.

### 3 Estimating Regret Locally

In this section, we discuss how agents can estimate their regret. As discussed in Section 2.3, the agents cannot compute their real regret (using Equation (4)) due to the lack of information regarding the routes rewards. The point is that regret is measured considering (i) the agent's average reward under their current policy and (ii) the average reward under the best policy in hindsight. Calculating the latter requires knowing the rewards of all routes along time. However, after each trip, an agent can observe the reward of the route taken, but cannot observe the reward of the other routes. Such a full observability property would only be possible under strong assumptions (e.g., a central authority broadcasting such information), which can be unrealistic in traffic domains. Furthermore, investigating methods to accomplish such a task in the absence of any supporting service is more challenging and is also relevant, especially in the highly competitive settings considered here [Stone and Veloso, 2000].

In this paper, we investigate how agents can estimate their regret based exclusively on local information (i.e., the rewards actually experienced by them). To this regard, we propose an alternative definition of regret that describes the estimated regret of each action.

Let  $A_i \subseteq A$  denote<sup>3</sup> the set of routes of agent  $i$ . At time  $t$ , agent  $i$  performs a given action  $a_i^t \in A_i$  and receives a reward of  $r^t(a_i^t)$ . We represent the history of experiences of agent  $i$  as  $H_i = \{r_{i,a}^t \mid a \in A_i, t \in [1, T]\}$ , with  $r_{i,a}^t$  the reward experience of driver  $i$  for taking action  $a$  at time  $t$ . However, recall that an agent cannot observe the reward of action  $a$  on time  $t$  except if it has taken such action at that time, i.e., if  $a = a_i^t$ . To this regard, the agents can assume the reward of non taken actions to be the same as the most up to date experience on that route. Precisely, let  $\tilde{r}_{i,a}^t$  represent the newest reward

<sup>3</sup>We slightly abuse notation here to account for drivers with different OD pairs, whose route sets are obviously different.



experience of agent  $i$  for taking action  $a$  on time  $t$  (either the current reward or the last<sup>4</sup> actually experienced one), as given by Equation (5). The history of experiences of agent  $i$  can then be rewritten as  $H_i = \{\tilde{r}_{i,a}^t \mid a \in A_i, t \in [1, T]\}$ .

$$\tilde{r}_{i,a}^t = \begin{cases} r^t(a_i^t) & \text{if } a = a_i^t \\ \tilde{r}_{i,a}^{t-1} & \text{otherwise} \end{cases} \quad (5)$$

Given the above definitions, we can now formulate the average estimated regret of agent  $i$  for taking action  $a$  at time  $t$  as in Equation (6). In general terms, we will refer to this formulation as *estimated action regret*. The estimated action regret  $\tilde{\mathcal{R}}_{i,a}^t$  can be seen as the estimated amount lost by agent  $i$  for taking action  $a$  at time  $t$  instead of the best action regarding its experience. Additionally, we can reformulate Equation (4) to obtain the *estimated agent regret*, as in Equation (7). The estimated agent regret  $\tilde{\mathcal{R}}_i^t$  expresses how much worse agent  $i$  performed, on average, up to time  $t$  for not taking only the best action regarding its experience. The main advantage of this formulation over the real regret (Equation (4)) is that it can be computed locally by the agents, eliminating the need for a central authority. Moreover, as the required information is already available to the agents, they can use such regret to guide their learning process.

$$\tilde{\mathcal{R}}_{i,a}^t = \max_{b \in A_i} \frac{1}{t} \sum_{s=1}^t \tilde{r}_{i,b}^s - \frac{1}{t} \sum_{s=1}^t \tilde{r}_{i,a}^s \quad (6)$$

$$\tilde{\mathcal{R}}_i^t = \max_{a \in A_i} \frac{1}{t} \sum_{s=1}^t \tilde{r}_{i,a}^s - \frac{1}{t} \sum_{s=1}^t r^s(a_i^s) \quad (7)$$

## 4 Learning Through Regret

In this section, we present a simple algorithmic solution for the agents to learn using the estimated action regret of Equation (6). To this end, we employ the Q-learning algorithm (as presented in Section 2.1). We highlight, however, that any other reinforcement learning algorithm could be used as well.

Every driver  $i \in D$  is represented by a Q-learning agent. The route choice problem can then be modelled as a stateless MDP. As such, the states and the transition functions can be ignored. Let  $A = \{A_i \mid i \in D\}$  be the set of agents' actions, where  $A_i$  is the set of routes of agent  $i$ , with  $A_i = A_j$  if agents  $i$  and  $j$  belong to the same OD pair. The reward for taking action  $a$  at time  $t$  is given by  $r^t(a)$ .

The learning process works as follows. At each episode  $t \in [1, T]$ , each agent  $i \in D$  chooses an action  $a_i^t \in A_i$  using the  $\epsilon$ -greedy strategy. After taking the chosen action, the agent receives a reward of  $r^t(a_i^t)$ . Afterwards, the agent updates its history  $H_i$  using Equation (5) and calculates the estimated regret of action  $a_i^t$  using Equation (6). Finally, the agent updates  $Q_i(a_i^t)$  using Equation (3). This process is repeated for each episode, aiming at minimising agents' regret.

Recall that the original definition of regret of Equation (4) measures the regret of the agent, not of his actions. However,

<sup>4</sup>As initial value, one can consider the minimum possible reward, which is computed using the links' free flow travel times (as described in Section 2.1). From a practical perspective, such information could be easily obtained using any offline navigation device.

the agent regret is not useful in the learning process because it does not consider how much the reward of a single action contributes to the regret. In other words, as we consider the average regret, the reward of a good-performing action could be penalised by those of bad-performing actions. Moreover, the learning process works by adjusting the expected value (or regret) of each action of the agent. In this sense, our estimated action regret definition isolates the regret per action, thus allowing the actions to be evaluated singly. The estimated action regret is more suitable to evaluate how promising a given action is as compared to the others.

Finally, it is worth noting that, although each driver minimises its actions' regret, this is equivalent to minimising its total regret. Recall that the estimated action regret measures how much worse an action performs as compared to the best one. By employing such a value in the learning process, the agent puts more weight on the actions with smaller regret. Moreover, using the  $\epsilon$ -greedy strategy, the agent tends to choose the action with the smallest regret with a higher probability. Consequently, we can state that minimising the estimated action regret along time is equivalent to minimising the estimated agent regret.

## 5 Experimental Evaluation

### 5.1 Setup

In order to evaluate our approach, we employ five different road networks that are available in the literature.

**Pigou** : this is a classical network introduced in [Pigou, 1920]. It comprises only two links  $l_1$  and  $l_2$ , with  $c_{l_1}(f_{l_1}) = 1.0$  and  $c_{l_2}(f_{l_2}) = f_{l_2}/d$ . We set the number of drivers to  $d = 100$ , all of them belonging to the same OD pair. In this scenario, there are only two actions (one corresponding to each link), i.e.,  $|A| = 2$ .

**OW** : is a small, synthetic network, with  $|N| = 13$ ,  $|L| = 48$ , and  $d = 1700$  [Ortúzar and Willumsen, 2011, exercise 10.1]. The vehicles are distributed among four OD pairs. The costs of the links are calculated using Equation (1). In this network, the number of possible routes for each OD pair is high. To this regard, we employ the KSP algorithm [Yen, 1971] to find the  $k$  shortest routes for each OD pair, i.e.,  $|A| = k$ .

**Braess graphs** : these are expanded versions of the network introduced in [Braess, 1968]. Specifically, let  $p \in \{1, 2, \dots\}$  be the  $p^{th}$  expansion of such graph, where 1st Braess graph is equivalent to the original graph [Roughgarden, 2006]. We generalise such model to consider a discrete set of drivers. The complete description of these graphs is available in Appendix A. We employ the 1st Braess graph, 2nd Braess graph and 3rd Braess graph, and define  $d = 4000$ , with all drivers belonging to the same OD pair, and, by definition,  $|A| = 2p + 1$ .

An experiment corresponds to a complete execution, with 1000 episodes, of the Q-learning algorithm on a single network. After an execution is completed, we measure its performance by means of the average travel time (*avg-tt* hereafter, measured in minutes) and the average estimated agent

Table 1: Performance of our approach in different road networks. For each tested network, we show: the average travel time (which, ideally, should approximate the UE), the UE (as reported in the literature), the average estimated agent regret (Equation (7)), the average real agent regret (Equation (4)), and the relative difference between the estimated and real agent regrets.

network	<i>avg-tt</i>	UE	estimated regret	real regret	relative difference (%)
Pigou	1.000	1.000	0.0136	0.0135	4.11
OW	67.156	67.157	0.0031	0.0045	8.02
1st Braess graph	1.988	2.000	0.0245	0.0224	8.25
2nd Braess graph	2.832	3.000	0.0393	0.0221	41.66
3rd Braess graph	3.701	4.000	0.0882	0.0293	64.64

regret (using Equation (7)), both regarding the last episode. We tested different combinations for the Q-learning parameters. For each such combination, 30 repetitions were performed. The best<sup>5</sup> combination found was  $\alpha = 0.5$ ,  $\epsilon = 1.0$  and  $\lambda = 0.99$ . Moreover, in the case of the OW network, we also tested different values for  $k$  (the KSP algorithm is run once for each OD pair, in the beginning of the experiment). The best results were achieved for  $k = 8$ . The results of such configurations were selected for further analysis in the next subsection.

The main hypotheses of our work are that: (i) the results approach the user equilibrium (UE), and (ii) the regret estimations are reasonably precise. In the next subsection, we analyse how successful our approach performed regarding our initial hypotheses.

## 5.2 Results

The performance of our approach in all tested road networks is presented in Table 1. In the table, we show the two main performance metrics *avg-tt* and average estimated agent regret. Additionally, in order to analyse such results, we present the UE (as reported in the literature), the average real agent regret (using Equation (4)<sup>6</sup>), and the relative difference between the estimated and real regrets.

Our first hypothesis states that the *avg-tt* results are close to the UE. As shown in Table 1, such results are indeed close to the UE values reported in the literature. The results become slightly far from the UE for the Braess graphs, especially the larger ones ( $p > 1$ ). This phenomenon can be explained by the nature of such graphs. Under UE, only the so-called type-P routes are used (see Appendix A for a detailed description). However, such routes have very similar costs. Consequently, it becomes harder for the agents to choose which route to take. The problem becomes even harder for larger Braess graphs because the number of type-P routes also increases with  $p$ . Furthermore, the Braess graphs were designed so that the UE values are the farthest possible from the System Op-

timum (SO). Nonetheless, observe that, for these particular graphs, our *avg-tt* results are closer to the SO than those of the UE. Therefore, such results evidence that our approach provides good approximations to the UE.

Regarding our second aforementioned hypothesis, its validation involves evaluating how precise the regret estimations are. To analyse such precision, we compare the real and estimated agent regrets by means of their relative difference. The lower such difference, the better. Of course, such difference cannot be computed by the agents, otherwise the regret estimation would not be necessary. As can be seen in Table 1, the estimated regrets are reasonably close to the real ones, especially for the networks Pigou, OW and 1st Braess graph. For the larger Braess graphs, the results were somewhat worse. The point here, again, is that the Braess graphs are more challenging because they were designed to be among the hardest networks. As the agents have more difficulty to learn their best routes, the network takes longer to reach a more stable point. Consequently, the agents estimations need to be updated more frequently to account for the high variations in the routes costs. However, despite the difficulties, the estimations were reasonable. We highlight that such estimations could be greatly improved by adopting more sophisticated estimation methods (e.g., using a nonlinear regressor). Thus, the present results also evidence that our regret estimation mechanism reaches a reasonable level of precision.

Along these lines, we can conclude, at least experimentally, that the agents can, in fact, estimate their regret locally and use such information to learn their best routes. Obviously, these results are not definitive. As initially mentioned, this work represents a very first step towards a more formal investigation regarding formal guarantees for RL algorithms, which is our medium-term objective.

## 6 Concluding Remarks

In this paper, we addressed the route choice problem by minimising the drivers' regret. This problem concerns how rational drivers learn which route to take so as to minimise their expected travel costs. To this regard, we developed methods for learning agents to estimate their regret locally (i.e., based exclusively on their experience) and to learn using such estimations. Specifically, each agent keeps a history of experimented rewards, which is used to compute the so-called estimated action regret.

<sup>5</sup>The best value for  $\alpha$  varied slightly from one network to another. However, such a value was reasonably close to 0.5 in all tested cases. Thus, for uniformity, we chose  $\alpha = 0.5$  for all networks.

<sup>6</sup>In order to compute the real regret of Equation (4), we considered that the space of policies consists of a simple mapping from routes to deterministic policies. In fact, ignoring mixed policies over the set of available actions is a common practice in the literature [Banerjee and Peng, 2005; Zinkevich *et al.*, 2008].

Based on experiments, we have shown that our approach provides reasonably precise estimations of the agents' regret and that such estimations are useful in the learning process. We recall that this work represents an initial effort towards a more formal investigation of efficiency guarantees for RL algorithms, which is our medium-term objective.

For future work, we would like to investigate formally how precise the regret estimations might be. We expect that more sophisticated methods could be employed to estimate the agents' regret (e.g., using a nonlinear regressor). Moreover, we would like to study how much our approach benefits when the agents can communicate to improve their estimations. We also consider investigating the benefits of employing an online service for providing global information for the agents. Regarding the learning process, a promising direction would be adopting algorithms that learn mixed policies over the actions rather than only the best action. Furthermore, considering this work is a proof-of-concept, no comparison was made against other methods in the literature. Thereby, making such a comparison is an obviously important step to provide a more complete analysis of our approach. Last but not least, it would be interesting to explore how the learning process could be shaped towards globally efficient routes.

## Acknowledgments

We are very grateful to the anonymous reviewers for their valuable comments. The authors are partially supported by CNPq and CAPES grants.

## References

- [Banerjee and Peng, 2005] Bikramjit Banerjee and Jing Peng. Efficient no-regret multiagent learning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 41–46. AAAI Press, 2005.
- [Bazzan and Klügl, 2013] Ana L. C. Bazzan and Franziska Klügl. *Introduction to Intelligent Systems in Traffic and Transportation*, volume 7 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan and Claypool, 2013.
- [Bowling and Veloso, 2002] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.
- [Bowling and Zinkevich, 2012] Michael Bowling and Martin Zinkevich. On local regret. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1631–1638, New York, USA, 2012. Omnipress.
- [Braess, 1968] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258, 1968.
- [Buşoniu *et al.*, 2008] L. Buşoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.
- [Hannan, 1957] James Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.
- [Koutsoupias and Papadimitriou, 1999] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science (STACS)*, pages 404–413, Berlin, Heidelberg, 1999. Springer-Verlag.
- [Ortúzar and Willumsen, 2011] Juan de Dios Ortúzar and Luis G. Willumsen. *Modelling transport*. John Wiley & Sons, Chichester, UK, 4 edition, 2011.
- [Pigou, 1920] A. Pigou. *The Economics of Welfare*. Palgrave Classics in Economics. Palgrave Macmillan, 1920.
- [Powers and Shoham, 2005] Rob Powers and Yoav Shoham. New criteria and a new algorithm for learning in multi-agent systems. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1089–1096. MIT Press, 2005.
- [Roughgarden, 2006] Tim Roughgarden. On the severity of Braess's paradox: Designing networks for selfish users is hard. *Journal of Computer and System Sciences*, 72(5):922–953, 2006.
- [Shoham *et al.*, 2007] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, May 2007.
- [Stone and Veloso, 2000] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.
- [Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [Wardrop, 1952] John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institution of Civil Engineers*, volume 1, pages 325–362, 1952.
- [Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.
- [Waugh *et al.*, 2015] Kevin Waugh, Dustin Morrill, J Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2138–2144. AAAI Press, 2015.
- [Yen, 1971] Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.
- [Zinkevich *et al.*, 2008] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In J C Platt, D Koller, Y Singer, and S T Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1729–1736. Curran Associates, Inc., 2008.
- [Zinkevich, 2003] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 928–936, Menlo Park, USA, 2003. AAAI Press.

## A Expanding the Braess Graphs

The original Braess graph was designed to illustrate the counter-intuitive phenomenon that removing a link from a road network may improve its outcome [Braess, 1968]. This is the so-called Braess paradox. In this paper, we are not interested in the paradox itself. However, we employed the Braess graph for validating our approach given it poses some interesting challenges in the drivers' decision process (as seen in Section 5.2). Roughgarden devised a method for generating the  $p^{\text{th}}$  expansion of the original Braess graph [Roughgarden, 2006]. Nonetheless, the proposed modelling required the flow (i.e., the number of vehicles) to be normalised in the interval  $[0, p]$ , with  $p$  the order of the Braess graph. In order to overcome such limitation, we extend Roughgarden's modelling dropping such a requirement, thus delivering a more general model. Moreover, we provide updated theoretical results for the System Optimal (SO) and User Equilibrium (UE) solution concepts, as well as the results for the Braess paradox and the price of anarchy.

### A.1 Graphs Generation Process

Consider the modelling introduced in Section 2.1. Let  $B^p$  be the  $p^{\text{th}}$  Braess graph, with  $B^1$  being equivalent to the original Braess graph. The set of nodes can be described as  $N^p = \{s, n_1, \dots, n_p, o_1, \dots, o_p, t\}$ , with  $|N^p| = 2p + 2$  and  $s \in N$  and  $t \in N$  representing the source and target nodes, respectively, for all  $d$  drivers (i.e., all drivers share the same OD pair). Let  $(i, j)$  represent a link from  $i \in N$  to  $j \in N$ . The set of links can be formulated as  $L^p = \{(s, n_i), (n_i, o_i), (o_i, t) : 1 \leq i \leq p\} \cup \{(n_i, o_{i-1}) : 2 \leq i \leq p\} \cup \{(n_1, t), (s, o_p)\}$ , with  $|L^p| = 4p + 1$ . The links are grouped into three distinct types, each with a corresponding cost function, as follows.

**type-A** : for all links on the form  $(n_i, o_i)$ , we use  $c_i^p(f_i) = 0$ ;

**type-B** : for all links on the form  $(n_i, o_{i-1})$ ,  $(s, o_p)$ , and  $(n_1, t)$ , we use  $c_i^p(f_i) = 1$ ;

**type-C** : for all links on the form  $(o_i, t)$  and  $(s, n_{p-i+1})$ , with  $i \in \{1, \dots, p\}$ , we use a piecewise, continuous, non-decreasing function as in Equation (8). Using this function, the maximum possible cost of a type-C link (when  $f_i = d$ ) is  $ip^2$ . The shape of the type-C cost function is illustrated in Figure 1.

$$c_i^p(f_i) = \begin{cases} 0 & \text{if } f_i \leq d/(p+1) \\ \frac{ip(pf_i + f_i - d)}{d} & \text{otherwise} \end{cases} \quad (8)$$

The routes are divided into two groups. Let  $P$  denote<sup>7</sup> the set of routes without any type-C link, i.e.,  $P = \{P_i = (s, n_i, o_i, t) \mid i \in [1, p]\}$ , with  $|P| = p$ . All the other routes, with type-C links, are then represented by  $Q = \{(s, n_1, t)\} \cup \{Q_i = (s, n_i, o_{i-1}, t) \mid i \in [2, p]\} \cup \{(s, o_p, t)\}$ , with  $|Q| = p + 1$ . We will distinguish the routes from these two groups as type-P and type-Q routes.

<sup>7</sup>We slightly abuse notation here, using  $(s, \dots, t)$  to represent a sequence of connected links  $\{(s, \cdot), \dots, (\cdot, t)\}$  that form a route.

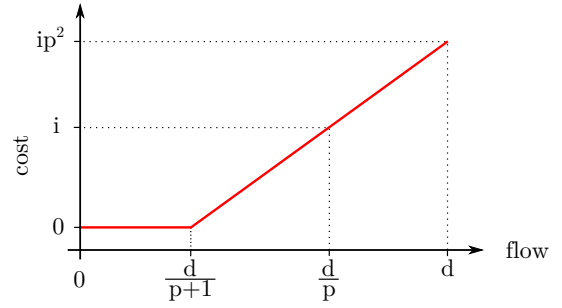
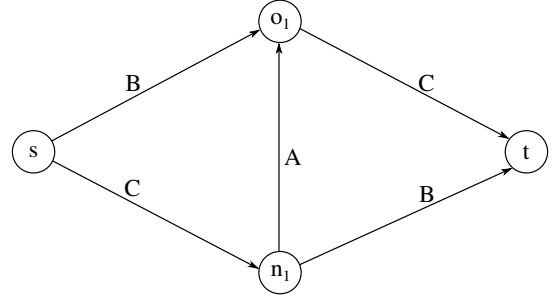
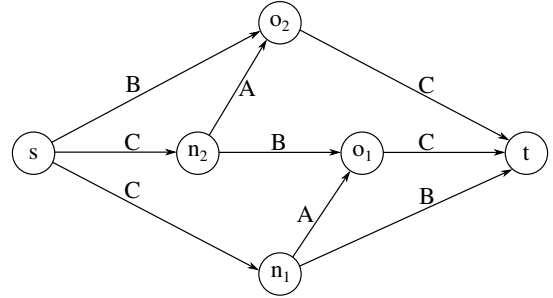


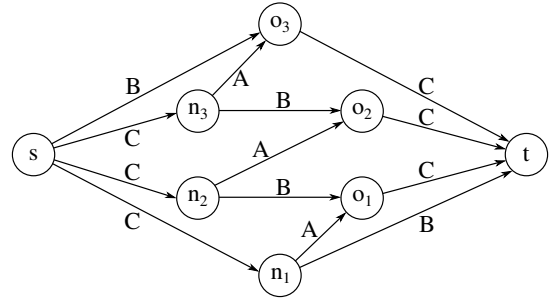
Figure 1: Shape of type-C cost functions.



(a) 1st Braess graph



(b) 2nd Braess graph



(c) 3rd Braess graph

Figure 2: The first, second, and third Braess graphs. Links are labelled with their types.

### A.2 Theoretical Results

Given the above formulation, we can define theoretical results for the System Optimum (SO) and the User Equilibrium (UE) as follows. The SO is achieved when the total flow  $d$  is equally divided among the  $p + 1$  type-Q routes. In this case,

each such route receives a portion  $d/(p + 1)$  of the flow and the type-P routes are not used. Under such conditions, each route has a cost of 1 and the *avg-tt* is also 1.

The UE, on the other hand, is achieved when only the  $p$  type-P routes are used. In this case, each such route receives a flow of  $d/p$ , thus costing  $p + 1$ . Under such circumstances, the *avg-tt* is also  $p + 1$ . By comparing the SO and UE results, we can define the price of anarchy to be  $p + 1$  [Koutsoupias and Papadimitriou, 1999].

Observe that, in both solution concepts, the *avg-tt* and the route costs are always the same. This is due to the fact that, in both cases, all routes being used have precisely the same flow and cost, i.e., under SO all used routes have a flow of  $d/(p+1)$  and cost 1 each, and under UE all used routes have a flow of

$d/p$  and cost  $p + 1$ . Consequently, as all vehicles experiment the same cost, we have that the *avg-tt* and the route costs are always the same.

Regarding the Braess paradox, our modelling does not invalidate it. Observe that, whenever the type-A links (those in the form  $(n_i, o_i)$ ) are removed, all type-P routes are also eliminated. Moreover, recall that, under UE, the cost of the flow is  $p + 1$ . However, after the type-P routes are removed, the UE is achieved when the flow is equally divided among the type-Q routes, which is precisely the SO. Thus, removing the type-P routes leads to an improvement in the overall performance, meaning that the Braess paradox exists.

# An Empirical Investigation of Adaptive Traffic Control Parameters

Jeffery Raphael<sup>1</sup> and Elizabeth I. Sklar<sup>2</sup> and Simon Maskell<sup>3</sup>

<sup>1</sup>Dept of Computer Science, University of Liverpool, UK

<sup>2</sup>Dept of Informatics, King’s College London, UK

<sup>3</sup>Dept of Electrical Engineering and Electronics, University of Liverpool, UK

jeffery.raaphael@liverpool.ac.uk, elizabeth.sklar@kcl.ac.uk, s.maskell@liverpool.ac.uk

## Abstract

The goal of *adaptive traffic management* is to adjust the timing of traffic signals at intersections in order to dynamically adapt, in real time, to traffic conditions. The SCOOT system, a commercial product widely deployed around the world, focuses on adjusting three traffic signal control parameters: *split*, *cycle* and *offset*. By responding to data collected from sensors embedded in roadways, SCOOT can effectively adjust to expected fluctuations in traffic, such as those that occur regularly during commuting hours. However, SCOOT does not perform optimally when there are unexpected disruptions in traffic flow, such as after the occurrence of an accident or during events that cause traffic conditions to deviate from the norm. The work presented here outlines an empirical study of the three SCOOT parameters, comparing the adjustment algorithm employed by SCOOT to a number of different adaptive methodologies, including two novel schemes. Experimental results, analysed across a range of different traffic flows, demonstrate that the novel methods perform as well as SCOOT under normal conditions and better under disruptive conditions.

## 1 Introduction

The notion of *adaptive traffic management* has been considered in a range of fields, from traffic control engineering to intelligent systems science. The goal is to maximise the throughput of vehicles across networks of roadways: reducing travel times for individuals, minimising wait times at intersections and avoiding collisions. There are a number of desirable subgoals, such as reducing the amount of pollution created by decreasing travel times, lowering petrol costs by shortening idle times and diminishing stress on commuters.

Within the *multi-agent systems* (MAS) community, a popular approach is to represent each vehicle as an *autonomous agent* and employ mechanisms that require the vehicles to negotiate with each other [Carlino *et al.*, 2013; Dresner and Stone, 2004; Vasirani and Ossowski, 2012]. However, widespread deployment of *autonomous vehicles* in real-world environments is not a near-term reality. There are many challenges that remain before self-driving cars will be used by

the masses. First, there is the development and deployment of the cars themselves. Google’s self-driving cars are widely talked about, with a fleet of autonomous cars that have collectively covered over 700K miles [Gomes, 2014]. Yet, these cars navigate using special maps that have enhanced information, such as location of traffic signals and driveways. As well, they cannot avoid unmarked potholes and would not be able to obey commands from a traffic officer [Gomes, 2014]. Second, there is the current state of connectivity. The communication infrastructure necessary for broad vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) currently does not exist. In the USA, the National Highway Traffic Safety Administration (NHTSA) is currently pushing for the use of V2V technology nationwide, arguing that it could dramatically reduce accidents by warning of dangers ahead; but, to date, there is no nationwide agreement or timeline for implementation. It is estimated that self-driving cars will not completely supercede human-driven cars until at least the year 2040 [Litman, 2015; Shanker *et al.*, 2013].

Motivated by these practical constraints, our work does not rely on the presence of autonomous vehicles—instead, we focus on adaptive solutions to traffic problems that can be deployed within today’s infrastructure. An *intersection* is a prominent feature of existing infrastructure, where roads cross each other and the need to coordinate access to the intersection is vital for preventing collisions. The task of *intersection management* is primarily achieved using traffic signals, familiar artifacts that are well integrated into road infrastructures world-wide<sup>1</sup>. Traditionally, intersection management by traffic signals is implemented as *fixed* periods of green, amber and red lights. In an effort to improve on the performance of fixed traffic signals, *adaptive Urban Traffic Controllers* (UTCs) have been developed and deployed in many cities around the world [Wang, 2005; Mladenovic and Abbas, 2013; Papageorgiou *et al.*, 2003]. Adaptive UTCs use information about current road conditions and determine, some in real-time, the best signal settings. These systems attempt to harmonise the interplay between all aspects of traffic (private cars, public transporta-

<sup>1</sup>In some countries, another common feature of the infrastructure is a *roundabout* (also called a *rotary* or *traffic circle*); but these are controlled through norms and driver behaviour, and do not fall into the category of technologies that are controlled through infrastructure external to the driver, which is what we consider here.

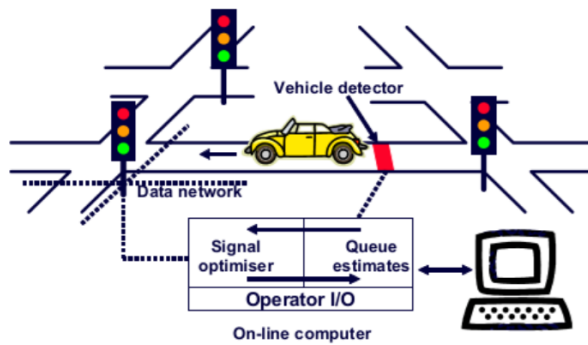


Figure 1: The SCOOT Model [Limited, 2016]. The red stripe across the road behind the yellow car in the figure illustrates an induction-loop road sensor.

tion, cyclists and pedestrians) in areas ranging in size from a few city blocks to entire cities. The majority of adaptive UTCs employ optimisation algorithms which are costly to develop, calibrate, maintain and expand [Wang, 2005]. Examples of deployed UTCs include: SCOOT<sup>2</sup> [Hunt *et al.*, 1981], RHODES [Mirchandani and Wang, 2005] and OPAC [Gartner *et al.*, 2001]. We focus on SCOOT because it is a popular system, it is deployed in our local city and we have access to data for modelling. The remainder of this paper is organised as follows. Section 2 describes how SCOOT works. Our approach is presented in Section 3 and experiment design in Section 4. Our results are presented in Section 5 and discussed in Section 6. Section 7 reviews other adaptive approaches to traffic control, and Section 8 closes with a summary and directions for future research.

## 2 SCOOT

SCOOT (*Split, Cycle and Offset Optimisation Technique*) is a centralised, real-time system that minimises delay and prevents congestion by coordinating small sets of traffic signals, called *regions*. The intersections within a region always form a linear path, i.e., signal timings are optimised to improve traffic flow in a single direction. SCOOT responds to data collected from *induction-loop sensors* embedded in roadways (Figure 1), which are simple counter devices that trigger when vehicles drive over them. Using the sensor data, SCOOT responds effectively to expected fluctuations in traffic.

Traffic can flow into an intersection from multiple directions, each of which is called a *link*. The *degree of saturation* of an intersection is a measure of its level of use, i.e., the amount of traffic demand compared to its maximum capacity. The traffic signal has a *phase* for each link which sequences through a period of *green time*, followed by a period of *red time*<sup>3</sup>. SCOOT adjusts three traffic signal control parameters, as follows:

<sup>2</sup><http://www.scoot-utc.com>

<sup>3</sup>Typically, red time is preceded by a short period of amber (yellow) time; in some countries, green time is preceded by a short period of joint amber and red time.

- **split**—The amount of green time allocated to each individual link is called *split*. Five seconds before a phase change, SCOOT considers the effect on the degree of saturation caused by *advancing* (terminating the phase), *retarding* (extending the phase) or *holding* (allowing the phase to continue to termination). SCOOT selects the option that reduces the degree of saturation the most. The split is adjusted in increments/decrements of 4 seconds.
- **cycle**—*Cycle length* is the total amount of time it takes for every link to receive its complement of green time. SCOOT optimises cycle length by examining the roadway with the highest degree of saturation. If that is greater than 90%, then the cycle length (for the entire region) is increased. SCOOT decreases the cycle length if every roadway entering the intersection has a degree of saturation greater than 90%. Cycle length changes are made in increments (or decrements) of 4, 8, 16, and 32 seconds (the shorter the cycle, the smaller the change) [Halkias, 1997].
- **offset**—A *green wave* is a phenomenon that occurs when a vehicle crosses many intersections in a row and all the traffic signals show green, so the vehicle does not have to stop at each intersection. In order for a green wave to occur, the traffic signals at adjacent intersections in a given path must be synchronised. The *offset* parameter represents the difference between the start of green time at two consecutive intersections. SCOOT checks the offset once at the end of every cycle and attempts to minimise the number stops required per vehicle by adjusting the offset in increments/decrements of 4 seconds.

Although SCOOT responds well to expected changes, such as regular increases in directional traffic flows during commuting times, SCOOT does not perform optimally when there are *unexpected disruptions* in traffic flow, such as when there are accidents or entertainment events that suddenly cause patterns to deviate from the norm. In our work, we have developed a set of traffic patterns that test the efficacy of SCOOT under different conditions. We use these patterns to compare several different parameter adjustment policies to the SCOOT benchmark, including two novel schemes that take a market-based approach. Experimental results, analysed across different traffic flows, demonstrate that our novel methods perform as well as SCOOT under normal conditions and better than SCOOT under disruptive conditions.

## 3 Our Approach

Our approach to traffic control parameter optimisation considers the three SCOOT parameters described above. In order to tune these parameters for real-time traffic control, we address a number of questions: *Which parameters should be adjusted? When should the parameters be adjusted? What data is used to inform an adjustment? and How should the parameters be adjusted?*

Our approach to traffic control revolves around the notion that traffic control is a *coordination* problem where intersections work together to minimise delay. Thus, we decompose the intersection into a *multi-agent system* and utilise an



*auction-based approach* to facilitate coordination amongst its agents. Our approach shares some similarities with SCOOT: it manages traffic flow using the same three parameters (cycle, split and offset), uses degree of saturation to measure road usage and uses transportation technology (vehicle detectors) that is currently available. However, our approach has a many significant differences. Adjustments to the traffic control parameters are made periodically and intersections are not clustered into fixed, pre-defined regions. Without these restrictions, our approach allows our mechanism to function on a much larger scale than SCOOT.

We first experimented with the idea of intersections as agents, informed in real-time by road sensors, in Raphael *et al.* [2015], where we presented our SAT mechanism. Here we expand upon that work in several ways. First, we present two new strategies for the behaviour of our traffic control agents. Second, we present experimental results that demonstrate the robustness of our approach in the face of unexpected disruptions in traffic flow. Finally, we compare our approach with a broad set of alternate strategies.

In both approaches, we use an *intersection agent* as an auction manager and *traffic signal agents* that represent the traffic signal phases. A phase represents multiple traffic streams. A single phase can service multiple vehicle manoeuvres. For example, the first phase of a traffic signal may allow through traffic and left turns. We use a two-phase signal plan: one light phase for north/south-bound traffic and the other phase for west/east-bound traffic. Thus, at every intersection, there is an intersection agent working in concert with two traffic signal agents. Our traffic signal control mechanism employs a first-price, single-item auction. As traffic flows through an intersection, auctions take place at fixed intervals<sup>4</sup>. The traffic signal agents bid against each other; the winner is the agent with the highest bid. The winning agent then makes a single adjustment to its traffic signal timing.

### 3.1 GRACE

Our initial investigation into traffic control mechanisms [Raphael *et al.*, 2015] was limited in its ability to react to changing traffic conditions because only green time was adjusted (in 5-second segments). Our new method presented here, **GeneRAL Purpose Auction-based Traffic ControllER** (GRACE), allows traffic signal agents to change all three variables. Adjustments are made in discrete steps,  $s$  (measured in seconds), defined as:

$$s = \langle \Delta_{green\_time}, \Delta_{offset}, \Delta_{cycle\_length} \rangle$$

For example, if  $s = \langle 3, -4, 10 \rangle$ , then the green time would be increased by  $3s$ , the offset reduced by  $4s$  and the cycle length increased by  $10s$ . A finite set of possible adjustment values is defined, specific to each mechanism (see below).

In [Raphael *et al.*, 2015], we measured the level of use of a roadway by calculating *saturation*, the ratio of the volume

<sup>4</sup>The optimal length of the fixed interval varies with each mechanism, and the values we use in our work were determined experimentally. Detailed discussion of these results is beyond the scope of this paper, but can be found in our technical reports.

of traffic (as measured by road sensors) to its estimated maximum capacity. However, this ratio does not quantify how a change to green time (or cycle length) effects the level of use in a lane( $s$ ), so GRACE uses *degree of saturation* [Lee *et al.*, 2002; Roess *et al.*, 2009],  $X$ , which is defined as:

$$X = \frac{v}{c} * \frac{L}{g} \quad (1)$$

where:  $v$  is the volume of traffic read by the traffic signal agent;  $c$  is the maximum possible volume of traffic (in vehicles per hour);  $L$  is cycle length; and  $g$  is green time. Traffic signal agents in GRACE are characterised by their utility function and their bidding rule. Next, we present two different GRACE-based traffic signal agents: DCF and MMDOS.

### 3.2 DCF

In *Dynamic Coalition Formation* (DCF), traffic signal agents find the best offset to reduce the number of vehicles that will have to stop for the red light and a green time that will minimise the maximum degree of saturation. At an intersection, each lane of traffic flow may have a different degree of saturation. DCF attempts to minimise the degree of saturation of the lane experiencing the highest level of use. The utility of adjustment  $s$  is given by:

$$U(s) = -[X + D(s)] \quad (2)$$

where the values for the degree of saturation  $X_t$  and estimated number of stopped vehicles  $D(s)$  reflect the adoption of adjustment  $s$ . The bidding rule for DCF is:

$$b = X \quad (3)$$

The possible adjustment values for DCF are:  $\Delta_{green\_time} \in \{0 \dots 5\}$ ,  $\Delta_{offset} \in \{-4, 0, 4\}$ , and  $\Delta_{cycle\_length} = 0$  (i.e., cycle length does not change).

### 3.3 MMDOS

In *Minimise Maximum Degree Of Saturation* (MMDOS), traffic signal agents minimise the *degree of saturation* of the lane experiencing the highest level of use. The utility of adjustment  $s$  in MMDOS is given by:

$$U(s) = -[X] \quad (4)$$

The biddings rule for MMDOS is:

$$b = X + u \quad (5)$$

where  $u$  is the length of the queue of cars on the roadway associated with the phase under the agent's control. The possible adjustment values for MMDOS are:  $\Delta_{green\_time} \in \{1 \dots 5\}$ ,  $\Delta_{offset} = 0$ , and  $\Delta_{cycle\_length} = 0$  (i.e., offset and cycle length do not change).

## 4 Experiments

We evaluated GRACE in a simulated  $5 \times 5$  grid-based city plan (Figure 2). Our traffic control experiments were conducted on *Simulation of Urban MObility* (SUMO) [Krajzewicz *et al.*, 2012], an open source microscopic traffic simulator. All traffic signals used a two-phase signal plan: during one

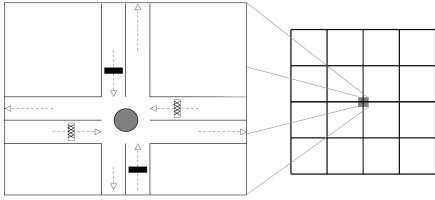


Figure 2: Grid-based city plan with intersection layout.

phase, north/south bound traffic passed through the intersection, while west/east bound traffic passed in the other phase. The signal plan did not include dedicated turning (right or left) phases, therefore left and right turns were given lower priority than through movements, i.e., vehicles turning left or right waited until it is safe to do so. All the roads were fitted with road sensors to collect traffic volume data. Also, the four corner traffic signals were disabled because there were no conflicting traffic movements at those intersections. Thus, in our experiments, GRACE adaptively controls twenty-one of the intersections.

#### 4.1 Traffic Conditions

For the experiments described here, we utilised three different traffic scenarios to evaluate the performance of our market-based mechanism. The scenarios employed sudden increases in traffic volume (or *intensity*) to disrupt traffic flow. The final scenario replicated traffic conditions that may occur during a sporting event. The scenarios are:

- **Structured** is traffic that flows through the network with an identifiable (e.g., commuter) path with heavy flow;
- **Unstructured** is traffic flow with no identifiable path with heavy flow; and
- **Football** emulated traffic conditions before, during and after a football match. The traffic flow represented a worst-case scenario where there is a sudden sharp increase in traffic demand. There are two disruptions: first, fans enter the area of the arena (30 minutes after the simulation started); and second, fans exit the arena (approximately 90 minutes later).

We raised the intensity of traffic at the one-hour mark during Structured and Unstructured traffic conditions. Structured represents the traffic pattern that is ideal for an adaptive urban controller such as SCOOT. Each set of experimental conditions were repeated 30 times to attain suitable statistics.

We evaluated the performance of the traffic controllers using the metric *travel time*. Travel time is by far the most common way of measuring the effectiveness of traffic controllers. We examined *travel time* in several different forms. First, we looked at the average travel time of all the vehicles across the 30 simulations. Second, we collected data on the average travel time of vehicles as they finished their journey at each time step. We compare the performance of our market-based controller to SCOOT (described in Section 1), fixed-time traffic signals (Section 4.2) and an auction-based traffic controller that learns a bidding strategy (Section 4.3).

#### 4.2 Fixed-time Signals

We also implemented a fixed-time traffic signal controller, FXM. The fixed-time traffic signal controllers represented traditional, non-adaptive, traffic signal devices. In the case of fixed-time traffic signal controllers, all three traffic control parameters remain constant. The traffic signals displayed the same light sequences for the same duration every cycle. We chose to use the initial traffic signal timing settings used by the adaptive mechanisms as the settings for the fixed-time traffic signals. Thus, any differences in performances can be attributed to the adaptive nature of the controller (and not initial signal timings). The fixed-time traffic signals have a cycle length of 80 seconds, and 87.5% of that is allotted to the split.

#### 4.3 Learning to Bid

We implemented a version of the auction-based traffic control mechanism of Mashayekhi and List [2015] in our SUMO traffic controller evaluation testbed. Of the three parameters adjusted by SCOOT, Mashayekhi and List modify only one, the split (green time). Their auction determines the amount of green time in a phase as well as the order of the phases. Mashayekhi and List used *Reinforcement Learning* (RL) to learn a bidding strategy. The only major difference between their implementation and ours was the number of *movement managers*. In their work, each movement manager was associated with a single stream of traffic. In our version, there were fewer movement managers because our test network did not have dedicated turning lanes. Furthermore, Mashayekhi and List did not specify an action space. Therefore, we discretised the bidding space to values  $[0 \dots 10]$  as our action space. That is, whenever an agent bids, its bid amount is some value between 0 and 10.

### 5 Results

Policy	Average Travel Time ( <i>std.</i> )		
	Traffic Pattern		
	Structured	Unstructured	Football
SAT	160.22 (8.22)	623.64 (42.31)	150.66 (9.10)
MMDOS	169.50 (7.31)	652.09 (48.57)	137.36 (5.35)
DCF	158.37 (4.98)	<b>609.22 (32.80)</b>	<b>135.15 (4.84)</b>
FXM	165.93 (1.38)	927.47 (107.39)	184.34 (7.13)
SCOOT	<b>143.66 (4.85)</b>	1931.35 (225.81)	233.42 (9.42)
RL	302.82 (17.70)	1038.09 (266.38)	200.89 (10.59)

Table 1: Average travel time of vehicles under different methods of traffic control.

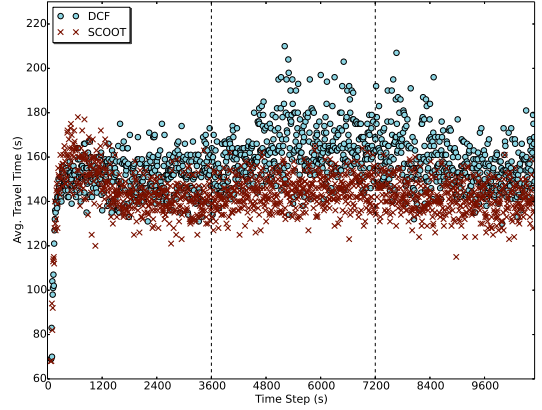
We simulated our three scenarios using six different traffic control methods: our earlier mechanism (SAT, from [Raphael *et al.*, 2015]), two new GRACE mechanisms (MMDOS and DCF), and three baselines: a fixed-time traffic signal (FXM), SCOOT, and the RL controller. In this section, we describe our

results, primarily the difference in performance of the controllers with *patterned* traffic (e.g., Structured traffic) versus *non-patterned* traffic (Unstructured and Football traffic).

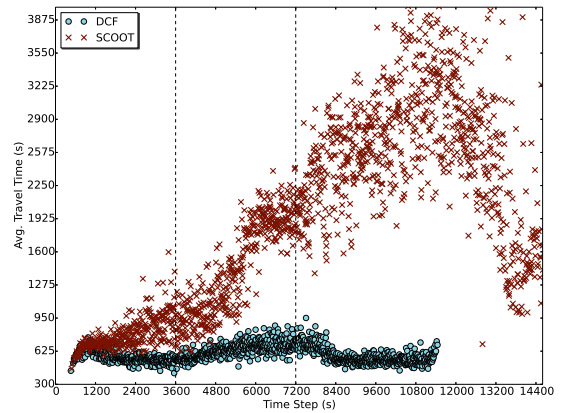
Average travel times reflect time saved (or incurred) at intersections due to adequate traffic flow. With Unstructured and Football traffic, our market-based approaches outperformed all the other traffic controllers (Table 1). The worst performing mechanism from our approaches did better than FXM. DCF had the best overall average travel time in both the Unstructured and Football traffic. In Unstructured traffic, DCF reduced average travel time by 34.3% and 68%, compared to FXM and SCOOT, respectively. For the simulated football event, DCF reduced average travel time by 26.7% and 42%, compared to FXM and SCOOT, respectively. SCOOT had the worst performance with the two non-patterned traffic scenarios. With Unstructured traffic, SCOOT increased average travel time by over 100% and with the football match traffic it increased travel time by 26% (this is compared to FXM). However, SCOOT had the best performance with Structured traffic (the second best time was achieved by DCF). RL performed slightly worse than FXM with Unstructured and Football traffic; it increased travel time by nearly 10% in both cases.

Figure 3 provides a more detailed picture of travel time under SCOOT control versus our DCF controller. At each time step, as vehicles completed their journey, we captured their average travel time. With Unstructured traffic, SCOOT’s travel time begins to increase even before the occurrence of the disruption at the 3600<sup>th</sup> second (Figure 3b). Under SCOOT, there is a sharp increase in travel times during the Unstructured disruption and it never recovers until the very end of the simulation. During the half-hour influx of drivers beginning at the 1800<sup>th</sup> second (Figure 3c), cars under DCF experienced significantly less delay than vehicles controlled by SCOOT. Immediately after the disruption ends, the average travel time peaks for both DCF and SCOOT, but SCOOT had the highest increase in average travel times. Both methods return to normal day-to-day travel times soon after the influx ends. Again, for the second disruption, starting at the 9000<sup>th</sup> second, traffic under SCOOT experienced far more delays than DCF. Although SCOOT did better than DCF in overall performance with Structure traffic, we find that there was significant overlap (Figure 3a) in travel times between vehicles under SCOOT control and vehicles controlled by DCF. In other words, there were many vehicles under DCF control that experienced travel time as short as those found in SCOOT. In Figures 3b and 4b, the SCOOT and RL simulations required more time steps than the other traffic controllers. The difference in the simulation horizon is due to how SUMO (the traffic simulator) works. SUMO does not terminate a simulation until all the vehicles that have been spawned complete their assigned trip. In all our simulations, the same number of vehicles were spawned but delay caused by the traffic controllers (e.g., SCOOT) resulted in a significant increase in the simulation horizon.

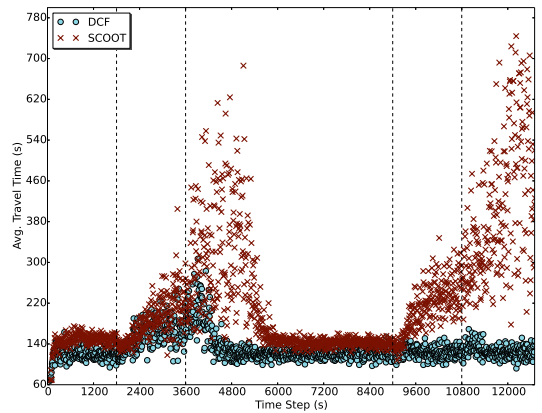
We also collected cumulative averages as the simulations ran (Figures 4). With Unstructured and Football traffic (Figures 4b and 4c), we see how quickly SCOOT’s performance diverges from the market-based approaches. Our market-based



(a) Structured

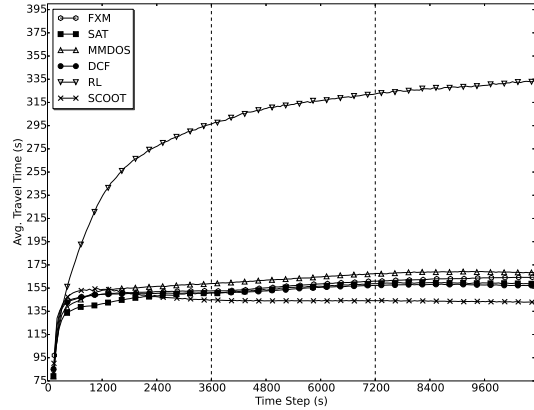


(b) Unstructured

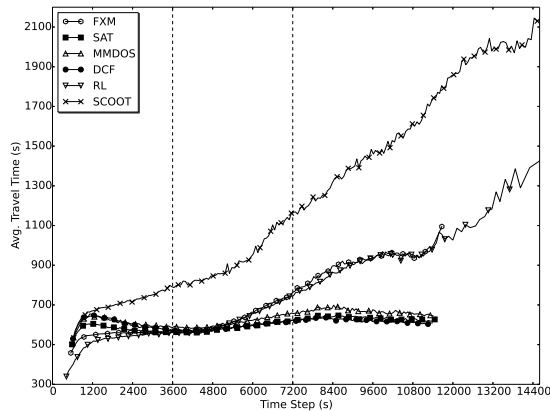


(c) Football

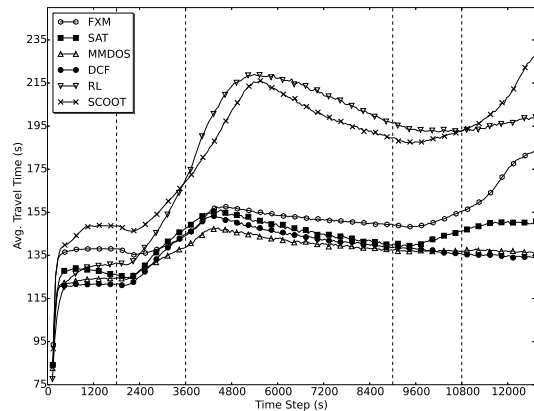
Figure 3: A comparison of average travel times of vehicles that have completed their journey at each time step. Beginning and end of disruptions are marked by dotted lines.



(a) Structured



(b) Unstructured



(c) Football

Figure 4: Cumulative average travel times. Beginning and end of disruptions are marked by dotted lines.

approaches did experience some increase in travel time during disruptions (e.g., the period from 1800<sup>th</sup> second to the

3600<sup>th</sup> second in Figure 4c), but never peaked as high as SCOOT. With Structured traffic, the traffic scenario where SCOOT had the best performance, we find that our approach closely matched FXM (Figure 4a). RL had the worst performance under Structured traffic. In Figure 4a, we see that RL never showed any signs of adapting to the traffic demands. Also, in Unstructured traffic (Figure 4b) RL’s performance closely mirrors FXM but in Football traffic (Figure 4c) it behaved more like SCOOT.

## 6 Discussion

Our results clearly demonstrate the dramatic effect traffic disruptions may have on the performance of SCOOT. Although our market-based approach utilises the same traffic parameters as SCOOT, we manipulate the split, offset and cycle time in a completely different manner. SCOOT is simply unable to satisfy the changing traffic demands and conflicting intersection manoeuvres (it is the latter that our approach excels at). SCOOT was designed to optimise the signal timing of small sets of traffic signals (that form a linear path). This severely restricts the ability of SCOOT to adapt to unexpected cross traffic. SCOOT performed well with traffic that had some established pattern of behaviour such as Structured, but could not cope with the Unstructured and Football scenarios. In Structured traffic (and the other scenarios like it) the scope of the control problem is more manageable than in other traffic scenarios.

RL did not perform as well as expected and our results did not resemble those found in [Mashayekhi and List, 2015]. There are a number of factors inherent to reinforcement-learning that could have contributed to its poor performance. For example, state space size (and representation) can affect learning, i.e., convergence to an optimal policy [Bakker *et al.*, 2010; Sutton and Barto, 1998].

Lastly, DCF and MMDOS represents our latest efforts to expand the capabilities of our market-based traffic controllers. One of the most important improvements to our approach is the new way in which it selects green time shifts. SAT can only make changes to green time in 5 second increments. DCF and MMDOS can make smaller adjustments, if necessary, to fine-tune green time allocations. Although DCF does attempt to form *green waves*, this ability does not always provide much of an advantage over SAT. DCF does use a constant cycle length and this may have negatively effected its performance. We will investigate this question in future work.

## 7 Related Work

Our approach is inspired by the work of Tumer and Agogino [2007], who applied MAS to the problem of air traffic control. Rather than modelling airplanes as autonomous agents, the authors made a counter-intuitive choice and defined *waypoints*—intermediate positions in an airplane’s flight path—as the agents. These static waypoints negotiated for the “right” to accept a plane at a particular instance in time. We adopt a similar approach to traffic control and select geographically fixed agents whose behaviour is influenced by traffic conditions. This is very different from many other traffic control systems that view the vehicles—rather than the

intersections—as their focus. To address the parameter adjustment questions from Section 3, we employ auctions to expedite parameter adjustments and coordinate intersections.

The variety of approaches to auction-based traffic control demonstrates the versatility of auctions as a means of resource allocation. Dresner and Stone [2004] did away with traffic lights entirely; relying instead on a reservation system to work out when it is safe to enter an intersection. Auctions can be deployed as a tool to determine road pricing (or congestion charge) in order to optimise route selection [Iwanowski *et al.*, 2003; Markose *et al.*, 2007]. Auctions can also be used as complete, intersection-level, traffic controllers. Carlino *et al.* [2013] described a traffic control system where second-price sealed bid auctions were used at intersections to determine order of use. Vehicles have an embedded agent bidding on their behalf, which is referred to as the *wallet agent*. A *system agent* also bids in a manner that facilitates traffic flow beneficial to the entire transportation system—while the *wallet agent* is solely (selfishly) concerned with getting its vehicle to its destination in the least expensive and quickest way. The authors tested different modes and found that the typical fixed-length traffic signal performed the worst in terms of reducing trip times.

One of the more interesting properties of utilising an auction mechanism as a component of traffic control is that it allows the intersection to consider the needs of individual drivers. Schepperle *et al.* [2007] described an intersection controller called *Initial Time-Slot Auction (ITSA)* which is *valuation-aware*—a mechanism that takes into consideration the individual’s cost of waiting at an intersection. In ITSA, vehicles approach and *register* with an intersection. An *intersection agent* executes a second-price sealed-bid auction for the most current time slot available. The authors also described two variants of ITSA: a mechanism is included to prevent *starvation*<sup>5</sup> where auctions are suspended if vehicle waiting time has reached some fixed limit; and ITSA+SUBSIDIES, which considers subsidies where vehicles that have not participated in an auction yet can influence the auction of the vehicles in front of them. The authors compared their traffic controller to the reservation-based system in Dresner and Stone [2004]. Both ITSA and ITSA+SUBSIDIES were able to reduce average travel time while minimising average weighted waiting time, as compared to the reservation-based system. ITSA+SUBSIDIES was better at reducing average weighted waiting time.

Vasirani *et al.* [2012] expanded on Dresner and Stone’s [2004] work by examining the performance changes to a reservation-based system where time slots were allocated using a *combinatorial auction (CA)*. As drivers approached the intersection, reservations were awarded through the auction, instead of simply handed out in order of arrival (the Dresner and Stone approach). In this way, drivers express their true valuation for a contested reservation. In a network with a single intersection, the authors looked at the delay experienced by drivers based on the amount they were willing

<sup>5</sup>In this context, *starvation* refers to one traffic flow being given a green signal for (too) long periods, and the other (stopped) traffic flows are “starved” for green time.

to “pay” to use the intersection. They found that initially having a willingness to pay does decrease delay, but eventually this levels off. However, CA was found to increase overall delay. As the intensity of traffic increased, CA experienced far more delays and rejected reservations than the first-come, first-served approach. Both reservation-based systems described in [Dresner and Stone, 2004; Vasirani and Ossowski, 2012] rely on vehicle agents having the capability to communicate with each other.

Other researchers have investigated approaches similar to our auction-based mechanism. Mashayekhi & List [Mashayekhi and List, 2015] designed a multi-agent auction-based traffic controller. The major difference between our approach and [Mashayekhi and List, 2015] is in the bidding strategy. We designed our bidding strategy from common traffic engineering practices while Mashayekhi & List used Reinforcement Learning to acquire a bidding strategy. Another significant difference is their traffic controller needs vehicle-to-infrastructure communication: as vehicles approach an intersection, they must report their presence to the movement managers via *tokens*. Our methods do not rely on such technologies.

## 8 Summary

We have presented our exploratory work on automated traffic control systems that do not require the existence of vehicle agents and can adjust dynamically as road conditions change. Moreover, our approach uses local traffic state information gathered from induction-loop vehicle detectors. As a result, our market-based traffic control methods are not constrained by the lack of transportation communication devices and protocols. Locally acting agents provide a robust traffic control system that maintains performance gains during and after traffic flow disruptions.

In patterned traffic, such as Structured, SCOOT performs well, but so do fixed-time signals. Thus, when recognised, these traffic patterns can be exploited; but this is not always the case in large cities where traffic disruptions (such as accidents or local events) can easily perturb the norm. Through a broad series of experiments, we have demonstrated the efficacy of our new approach, in comparison with our earlier work and several benchmarks (SCOOT, fixed-time signals and a reinforcement learning approach). The experimental results highlight the impact of including offset and fine-tuned green time adjustments in bidding, which produce improvements in travel time. Our next steps with this work involve incorporating elements in the bidding to improve green waves. We will also continue evaluating the traffic parameters discussed in this paper with the aim of developing a clearer picture of the impact that adjusting split, cycle and offset (and various combinations thereof) has on travel time.

## References

- [Bakker *et al.*, 2010] Bram Bakker, Shimon Whiteson, Leon J. H. M. Kester, and Frans C. A. Groen. Traffic Light Control by Multiagent Reinforcement Learning Systems. In Robert Babuska and Frans C. A. Groen, editors, *Interactive Collaborative Information Systems*, volume 281

- of *Studies in Computational Intelligence*, pages 475–510. Springer, 2010.
- [Carlino *et al.*, 2013] Dustin Carlino, Stephen D. Boyles, and Peter Stone. Auction-based autonomous intersection management. In *Proceedings of the 16th IEEE Intelligent Transportation Systems Conference (ITSC)*, 2013.
- [Dresner and Stone, 2004] Kurt M. Dresner and Peter Stone. Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. In *Proceedings of the Third International Joint Conference on AAMAS*, pages 530–537. IEEE Computer Society, 2004.
- [Gartner *et al.*, 2001] Nathan H. Gartner, Farhad J. Pooran, and Christina M. Andrews. Implementation of the OPAC Adaptive Control Strategy in a Traffic Signal Network. In *IEEE Intelligent Transportation Systems*. IEEE, 2001.
- [Gomes, 2014] Lee Gomes. *Hidden Obstacles For Google's Self-Driving Cars: Impressive Progress Hides Major Limitations Of Google's Quest For Automated Driving*. MIT Technological Review, (www.technologyreview.com), 2014.
- [Halkias, 1997] John A. Halkias. Advanced transportation management technologies : participant reference guide : Demonstration Project No. 105. Tech Report FHWA-SA-97-058, United States. Federal Highway Administration, 1997.
- [Hunt *et al.*, 1981] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton. SCOOT - A traffic responsive method of coordinating signals. Technical Report 1014, Transport and Road Research Laboratory, 1981.
- [Iwanowski *et al.*, 2003] S. Iwanowski, W. Spring, and W.J. Coughlin. Road traffic coordination by electronic trading. *Transportation Research Part C: Emerging Technologies*, 11(5):405–422, 2003. cited By 0.
- [Krajzewicz *et al.*, 2012] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.
- [Lee *et al.*, 2002] Sang Soo Lee, Seung Hwan Lee, Young Tae Oh, and Kee Choo Choi. Development of Degree of Saturation Estimation Models for Adaptive Signal Systems. *RSCE Journal of Civil Engineering*, 6(3):337–345, 2002.
- [Limited, 2016] T. R. L. Limited. *SCOOT Advice Leaflet 1: The SCOOT urban traffic control system*. April 2016.
- [Litman, 2015] Todd Litman. Autonomous Vehicle Implementation Predictions. Technical report, Victoria Transport Policy Institute, December 2015.
- [Markose *et al.*, 2007] S. Markose, A. Alentorn, D. Koesrindartoto, P. Allen, P. Blythe, and S. Grosso. A smart market for passenger road transport (SMPRT) congestion: An application of computational mechanism design. *Journal of Economic Dynamics and Control*, 31(6):2001–2032, 2007. cited By 0.
- [Mashayekhi and List, 2015] Mehdi Mashayekhi and George List. A Multi-agent Auction-based Approach for Modeling of Signalized Intersections. In *Second Workshop on Synergies Between Multiagent Systems, Machine Learning and Complex Systems (TRI 2015)*, Buenos Aires, Argentina, July 2015.
- [Mirchandani and Wang, 2005] Pitu Mirchandani and Fei-Yue Wang. RHODES to Intelligent Transportation Systems. *IEEE Intelligent Systems*, 20(1):10–15, 2005.
- [Mladenovic and Abbas, 2013] Milos Mladenovic and Montasir Abbas. A Survey of Experiences with Adaptive Traffic Control Systems in North America. *Journal of Road and Traffic Engineering*, 59(2):5–11, 2013.
- [Papageorgiou *et al.*, 2003] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, December 2003.
- [Raphael *et al.*, 2015] Jeffery Raphael, Simon Maskell, and Elizabeth Sklar. From Goods to Traffic: First Steps Toward an Auction-based Traffic Signal Controller. In *13th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'15)*. Springer, 2015.
- [Roess *et al.*, 2009] Roger P. Roess, Elena Prasas, and William R. McShane. *Traffic Engineering: International Edition, 4th Edition*. Pearson Education, Inc., 4th edition, 2009.
- [Schepperle and Böhm, 2007] Heiko Schepperle and Klemens Böhm. Agent-Based Traffic Control Using Auctions. In Matthias Klusch, Koen V. Hindriks, Mike P. Papazoglou, and Leon Sterling, editors, *CIA*, volume 4676 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2007.
- [Shanker *et al.*, 2013] Ravi Shanker, Adam Jonas, Scott Devitt, Katy Huberty, Simon Flannery, William Greene, Benjamin Swinburne, Gregory Locraft, Adam Wood, Keith Weiss, Joseph Moore, Andrew Schenker, Paresh Jain, Yejay Ying, Shinji Kakiuchi, Ryosuke Hoshino, and Andrew Humphrey. Autonomous Cars, Self-Driving the New Auto Industry Paradigm. Technical report, Morgan Stanley Research, November 2013.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.
- [Tumer and Agogino, 2007] K. Tumer and A. Agogino. Distributed Agent-Based Air Traffic Flow Management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 330–337, May 2007.
- [Vasirani and Ossowski, 2012] Matteo Vasirani and Sascha Ossowski. A Market-Inspired Approach for Intersection Management in Urban Road Traffic Networks. *Journal Artificial Intelligence Research*, 43:621–659, 2012.
- [Wang, 2005] Fei-Yue Wang. Agent-Based Control for Networked Traffic Management Systems. *IEEE Intelligent Systems*, 20(5):92–96, 2005.



# An Architecture for Safe Evacuation Route Recommendation in Smart Spaces

Marin Lujak<sup>a</sup>, Stefano Giordani<sup>b</sup>, and Sascha Ossowski<sup>a</sup>

<sup>a</sup> CETINIA, University King Juan Carlos, Madrid, Spain

<sup>b</sup> University of Rome “Tor Vergata”, Italy

## Abstract

In this paper, we treat pedestrian evacuation in emergency scenarios of networked smart spaces. Personal safety may be jeopardized due to natural catastrophes (e.g., hurricanes, earthquakes, etc.) and/or adversarial actions of intentional enemies. During evacuation, the severity of emergency may increase causing partial or complete blockage of some evacuation routes. Thus, it is of the highest importance to (re)route evacuees based on updated real-time structure safety conditions. In this paper, we propose a multi-agent based architecture for dynamic route safety optimization in large smart space evacuation. The objective of the model is to ensure that the smart space network gets evacuated securely while aptly responding to unpredictable contingencies in the network safety.

## 1 Introduction

The objective of an evacuation is to relocate evacuees from hazardous to safe areas or the areas where the life-threatening risk is minimal while providing them with safe routes. Present building evacuation approaches are mostly static and preassigned. Frequently, no coordination is available except for predefined evacuation maps. With sufficient estimated time to calamity and in case of larger evacuations, human coordinators are introduced mostly in isolated critical evacuation points. Due to uncertainty related with emergencies, there is a need for a real-time route recommendation system for dynamically determining evacuation routes in inner spaces based on the imminent or ongoing emergency.

Some typical reasons for evacuation include natural disasters like hurricanes, earthquakes, and wildfire, and adversarial actions like biological, nuclear, or chemical attacks. Evacuation routes may be subject to damage and destruction that may arise from natural catastrophes or action of intentional enemies. Due to the lack of the overall evacuation network information, there might be casualties caused by a too slow evacuation on hazardous routes. To avoid casualties and facilitate evacuation, we propose the usage of smart space technology for the introduction of route recommender systems into inner spaces. Smart spaces are spaces equipped with information processing, sensing and actuation facilities. These

systems can provide assistance and facilitate the distribution of real-time evacuation information to evacuees through, e.g., LCD displays and smartphones.

A smart space can be modelled as an agent able to acquire and apply knowledge about itself and about its inhabitants in order to improve their well-being in the same. Moreover, a network of smart spaces can be implemented not only in buildings, but also at an urban scale. A city may be seen as a network of smart spaces and their inhabitants. In such a complex system, by using the information of the both, intelligent evacuation route recommendation is aimed at guiding people to safe areas considering individually optimal routes while optimizing global people flow based on safety conditions. The resulting interaction of a multitude of space agents and humans requires a scalable and responsive evacuation coordination approach.

In this paper, we propose a multi-agent based architecture for evacuation safety optimization that considers personal safety requirements in the recommended routes and ensures dynamic route update based on safety conditions within buildings and on the road infrastructure. The proposed model reduces exposure to hazard by dynamically updating evacuees' routes in real time thus leading them to safe areas. Routes, evacuation areas, and safe areas are dynamically calculated and recalculated based on additional data, either real-time, historical, or other data added to the system, to compute optimal initial routes and redirect evacuees if changes in the emergency situation occur.

The rest of the paper is organized as follows. In Section 2, we consider crowd dynamics related with velocity, density and flow of pedestrians in inner spaces and State-of-the-art evacuation control approaches. The proposed route-recommender architecture is presented in Section 3 with necessary details on its functioning when recommending safe and efficient evacuation routes. In Section 4, we formally define the distributed evacuation safety optimization problem and in Section 5, we describe the optimization approach. We conclude the paper in Section 6.

## 2 Crowd dynamics

Total capacity is traditionally used to measure a building safety related with panic. It determines the total number of people who can fit in an edifice due to the physical space

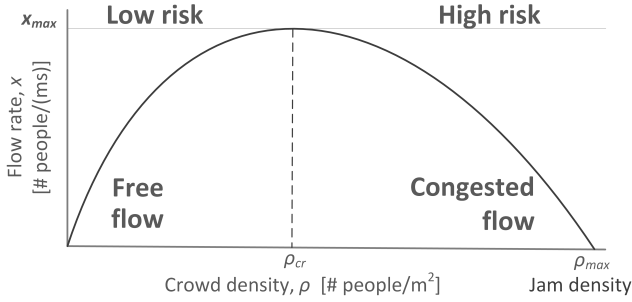


Figure 1: Risk levels in free flow and congestion: crowd flow rate - density relationship

available or limitations set by law. However, it is not a sufficient parameter to avoid panic-related casualties in larger spaces since the capacity should be controlled for every larger constituent space in the building.

Evacuation routes may pass from larger to smaller spaces where overcrowding may occur. The formation of crowds, their size and granularity, and in general dynamics of crowds are crucial parameters in panic tolerant evacuation systems, see, e.g., Lujak and Ossowski [2016, In press 2016]. Overcrowding is the main reason for crowd crushing, injuries and mass fatalities that can be avoided by keeping density and velocity of the crowd under critical values. These values are influenced by multiple factors like, e.g., crowd profile (average age, physical conditions, presence of families with children and people with physical disabilities, etc.), nature of surface (e.g., concrete, mud, sand), presence of depressions in the walking surface or debris, gravel, rocks, mud, slopes, steps, etc.

Similar to vehicle flow, a macroscopic fundamental diagram for pedestrian traffic involves crowd traffic flow, density and velocity. The relationship between crowd density (number of people per square metre [ $\#people/m^2$ ]) and crowd flow (number of people per metre per second [ $\#people/(m \cdot s)$ ]) is as follows:  $x = v(\rho) \cdot \rho$ , where  $x$  is unit flow rate,  $\rho$  is pedestrian density, and  $v(\rho)$  is pedestrian velocity [ $m/s$ ], which in general depends on the pedestrian density  $\rho$ , Figure 1.

One of the assumptions under which a proper shape of the fundamental diagram for pedestrian traffic is found, is that the congestion is spread homogeneously over the network, see, e.g., Knoop and Hoogendoorn [2013]. However, crowds rarely pack in regular formation. Knoop and Hoogendoorn Daamen *et al.* [2015] show the effect of inhomogeneity by deriving the so-called generalised macroscopic fundamental diagram. Hoogendoorn *et al.* [2011] have shown that a similar relation exists between the number of pedestrians in an area and the average flow in that area.

When there are few pedestrians on a walkway, i.e., low flow levels, there is space available to choose higher walking speeds. As crowd density increases, crowd flow increases only until critical density  $\rho_{cr}$  is reached, Figure 1. When a critical level of crowding occurs, maximal flow  $x_{max}$  oc-

currs at some critical combination of velocity and density and separates the free flow ( $x \leq x^{cr}$ ) from the congested one ( $x > x^{cr}$ ). With the increase of density above  $\rho_{cr}$ , people flow decreases until jam density where there is no more flow.

The critical density can be different for different events/crowds, see, e.g., Helbing and Johansson [2011]. Pedestrians can only circulate freely when crowds are no denser than approximately 10-15 persons per  $10 m^2$ . After this point, as crowd density increases the crowd flow rate falls. As individual movement becomes effortful because of closer interactions among evacuees, consequently also crowd velocity falls.

At high density, the crowd moves at the pace of the slowest individuals and there is the potential for overcrowding and personal injury. Evacuees' safety decreases due to a higher possibility of panic related behaviors such as herding and stampeding. This is why we should aim not to let the people density pass the critical value at any area.

Regarding velocity, people should avoid running to avoid panic. Human walking speed can vary depending on various factors such as, e.g., height, age, terrain, weight, effort, etc. The average human walking speed is about 5.0 kilometres per hour and it ranges from 4.51 to 5.43 kilometres per hour, see, e.g., Rastogi *et al.* [2010]. This means that every space should be dynamically controlled detecting group formations that should not surpass these values at any position.

The crowd is unlikely to be evenly distributed throughout an open space. This can make it difficult to estimate the point at which the space is reaching its capacity limit. This is why, at high risk people densities, it is important to monitor and control the crowd movement in all constituent areas of the space of interest at all times.

Before the crowd reaches jam density  $\rho_{max}$ , we can detect spaces between evacuees by people tracking technologies. Tracking refers to data output from the technologies that capture the evacuees' walking paths, e.g., WiFi by tracking their mobile phone signals, monocular and 3D stereo video, thermal imaging, infrared beams, and beacons. Each technology has its own set of challenges and benefits. For example, Wi-Fi and beacons are based on radio wave technologies, and are distinct by range and the accuracy of the signal capture process.

## 2.1 Evacuation control in smart spaces

By the use of ambient intelligence, we can both monitor and influence crowd actions during evacuation. The space access restrictions can be changed dynamically depending on the area safety status. The information about the number of people to evacuate and their behaviour facilitates successful planning of evacuation and assessing necessary emergency services.

Application of ambient intelligence to evacuation control is a dynamic research area. In Mittleton-Kelly *et al.* [2013], a review on the utilisation of Aml (Ambient Intelligence) technology in providing support and enhancing crowd evacuation during emergencies and improving traffic management is presented. While most of the approaches treat congested networks and related k-shortest path problem, to the best of our knowledge, there is little work on dynamic real-time route op-

timization based on the safety of the paths' constituent arcs, e.g., Stepanov and Smith [2009]. Most of the approaches take the binary approach for safety: the route is safe or not. In this paper, we go a step forward and offer the optimization of the routes when the route safety is represented by a continuous variable.

Azhar Mohd *et al.* [2016] provide a review of intelligent evacuation management systems covering the aspects of crowd monitoring, crowd disaster prediction, evacuation modelling, and evacuation path guidelines. While the review deals with video and nonvideo based aspects of crowd monitoring and crowd disaster prediction, evacuation techniques are reviewed via the theme of soft computing, along with a brief review on the evacuation navigation path.

A literature review of network emergency evacuation modeling was presented in Xiongfei *et al.* [2010]. The linear programming approach uses time-expanded networks to compute the optimal evacuation plan and requires a user-provided upper bound on evacuation time. It suffers from high computational cost and may not scale up to large transportation networks in urban scenarios. In Lu *et al.* [2005], a capacity constrained route planner (CCRP) was proposed. It is a heuristics that produces sub-optimal solution for the evacuation planning problem. The CCRP models capacity as a time series and uses a capacity constrained routing approach to incorporate route capacity constraints. It addresses the limitations of the linear programming approach by using only the original evacuation network and it does not require prior knowledge of evacuation time. The CCRP algorithm produces high quality solutions and significantly reduces the computational cost compared to the linear programming approach that produces optimal solutions. CCRP is also scalable to the number of evacuees and the size of the network.

Desmet and Gelenbe [2013] propose an approach to the design and optimisation of emergency management schemes that offers fast estimates based on graph and probability models. They show that graph models can offer insight into the critical areas in an emergency evacuation and that they can suggest locations where sensor systems are particularly important and may require hardening.

In Bruce *et al.* [2008], a GIS-based system that determines evacuation routes for specific areas requiring evacuation is presented. Routes, evacuation areas, and safe areas are dynamically calculated and recalculated based on additional data to compute optimal initial routes and redirect evacuees if changes in the emergency situation occur. However, the model includes only two operative states of the roads: open, closed, and their travel time if open. The proposed system does not take into account relative safety variation of the route.

One possible way of personalizing evacuation notifications and communicating evacuation routes in indoor work environments over smartphones was presented in Aedo *et al.* [2012]. The paper considers efficient communication of predefined evacuation routes that can be personalized based on a type of the evacuee. However, this paper does not consider autonomous smart space route update based on the evacuation route real time safety conditions.

### 3 Architecture for safe evacuation routes' recommendation

Safety conditions in the infrastructure change due to the evacuees' behavior and the safety conditions caused by the hazard. The proposed architecture for safe evacuation routes' recommendation integrates real-time evacuation route computation and situational awareness both at the evacuee and infrastructure level. The proposed architecture is made of the evacuee's route recommender and overall smart route evacuation system, both relying on smart space technologies, Figure 2. In more detail:

- **Evacuee's route recommender** is meant as a mobile app that serves as an evacuee's evacuation guide and an interaction bridge between the evacuee and the smart space while increasing situational awareness of the evacuee and recommending him/her evacuation route that avoids unsafe and highly congested spaces. The situation awareness solution should take into account data received through relevant sensors, evacuee's current mental state and the capacity to follow the recommended route based on the momentary GPS coordinates and the actual area safety state, the evacuation infrastructure complexity (e.g., through Google Services), sensor readings and actual smart phone's state (acceleration, velocity dynamics, orientation, etc.).

It uses smart phone sensors for knowledge extraction and communicates with nearby smart space infrastructure. Evacuee's personal route recommender system (EPRS) is a CPS that works as an evacuee's assistant that mediates the interaction between the evacuee and the Smart Space. The EPRS's objective is that the evacuation be safe in complex evacuation situations so it adapts the evacuation route to the profile of the evacuee. Moreover, evacuee's route recommender informs the evacuee about evacuation safety conditions and its malfunctions, battery, his/her performance, security alerts, crowdedness and related risks, alternative routes, etc.

- **Smart Route Evacuation System (SRES)** monitors and manages the strategic behavior of the smart space network and in the case of necessity, performs corrective actions on the spaces in real-time. SRES informs the evacuee's route recommender about the state of the evacuee's physical environment, eventual contingencies, and evacuation performance. It establishes a personal evacuee profile record (based on personal data, presence of mobility disabilities, affiliate ties with other evacuees etc.). If necessary, it undertakes corrective actions on the evacuees and minimizes the performance degradation during sudden changes of safety conditions. Moreover, it monitors in real time and acts upon human-factor processes (presence of panic and related herding and stampeding behaviors) and predicts possible such states. If necessary, it reassigns routes in real-time to overcome contingencies, e.g., accidents and overcrowding.
- **Smart space** is a Cyber-Physical System that integrates a series of sensors for obtaining data that passes through several levels of processing: data filtering by noise elim-

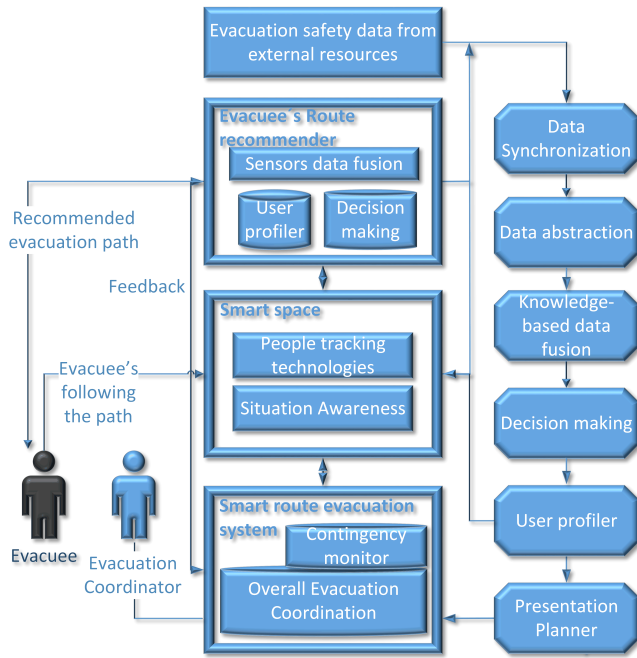


Figure 2: Proposed architecture for safe evacuation routes recommendation

ination, synchronization, abstraction at a semantic level, and data stream reasoning and knowledge extraction. The result of these processes is a situation awareness of the evacuees present in the smart space and knowledge sharing with other smart spaces and the smart route evacuation system. Some of the exemplary smart space situation awareness processes are: forecasting the hazard and evacuation dynamics with the specific evacuees' profiles and hazard description, and networking with other smart spaces in the system for optimal route computation and contingency coverage. The identification of the evacuation situation is possible through image recognition, fusion of data received from different sensors, and sensor knowledge extraction. Due to increased energy, computational and memory requirements, those operations are performed in a distributed manner by infrastructure node agents connected with a computational cloud.

There are services available at the overall architecture level for knowledge extraction integrating the situation awareness from the evacuees' route recommenders, the network of smart spaces, and the smart route evacuation system. These services serve for knowledge fusion from different databases and bottleneck routes' resolution at the system's level. They also keep track and evaluate evacuees' profiles based on their historical data and present behavior. After knowledge-based data fusion, safety classification of scenarios gives us numeric values for each safety condition, Figure 2.

### 3.1 Proposed multi-agent system for safe evacuation

The proposed multi-agent system model is composed of four different agent categories:

- **Evacuee agent** is implemented on evacuees' smart phones within an evacuee's route recommender and it represents each evacuee in the evacuation process.

- **Node agent** represents a physical node of the smart space network on which it is installed and controls the evacuation flow on it. Node agents interact with their neighboring node agents and in a distributed way monitor and control smart space network and, if necessary, compute the safest efficient evacuation routes for evacuees in a distributed way. Moreover, node agents are situated in the smart space and serve as its computational nodes. Each node agent senses its assigned physical node and its incoming arcs. Furthermore, it can open and close automatic exit doors and broadcast information to evacuees within its realm.

Each node broadcasts its incoming arc travel times in regular intervals such that any node in the network has a complete information about arcs' safety and costs. If a node detects the outage of one of its incident incoming arcs or neighbor nodes, it evacuates these areas and informs of the accident all neighboring nodes to deviate all traffic that has to be sent over this failed element.

- **Origin agent** is created on demand whenever there is at least one evacuee present in the realm of a node agent. It is a part of the smart route evacuation system that interacts with the evacuee agent through the evacuee's route recommender, Figure 2. Origin agents perform the shortest safe route computation for the evacuees positioned in their realm of influence. This computation can be made in a centralized or distributed manner with infrastructure node agents.

Once when the safest efficient routes are computed, each origin agent assigns them to its evacuees based on individual evacuee's characteristics (e.g., mobility disabilities, presence of families with children, etc.). Evacuees exchange the information only with their origin agent. As evacuees move in the infrastructure, their assigned origin agents change respectively.

- **Evacuation coordinator agent** represents a human evacuation manager or management team that has a broader knowledge of evacuation reasons and purposes. Their role is the description of key performance indicators based on the evacuation strategy.

No a priori global assignment information is available and the information is exchanged among these four agent types through neighbor to neighbor communication. In this way, we obtain a dynamic communication network operating in a multi-hop manner, which can recalculate evacuation routes based on the actual infrastructure safety conditions, evacuee congestion, and evacuation demand.

## 4 Finding safe and efficient evacuation routes

In this Section, we concentrate on finding the safest temporally efficient paths for each evacuee within the decision making module of the evacuee's route recommender. With this aim, we consider a network of smart spaces in flow condi-

tions where flow represents people transit pattern at steady state.

If real-time infrastructure information is available to evacuees and they can negotiate their routes (paths), it becomes possible to provide a selection of safe fair routes considering individual safety requirements. Therefore, we assume that the building and evacuees are monitored by strategically positioned sensors like, e.g., cameras, beacons, etc. The monitoring permits us both to recognize the evacuees' behavior in respect to the suggested route and time window as to perceive the congestion and safety conditions of the infrastructure.

Furthermore, we assume that the people flow demand (i.e., evacuation requests) is known at the beginning of the time window. Based on the population density data, we determine the evacuation demand in the case of regional evacuation, while in smart building evacuation, we use the number of persons in each node to enumerate the requests.

In this way, each individual is seen as a unit element (particle) of the total people flow. We assume, furthermore, that the variations of the evacuation requests are negligible in an observed time window.

Starting from the above stated assumptions, let us define the infrastructure from which the people need to evacuate. Let  $G = (N, A)$  be a connected digraph representing the smart space network where  $N$  is the set of  $n$  vertices representing rooms, offices, halls, and in general, any portion of space within a building or other structure, separated by walls or partitions from other parts. In the case of larger spaces, for simplicity, the same are divided into regions represented by nodes completely connected by arcs  $a \in A$ , where  $A$  is the set of  $m$  arcs  $a = (i, j)$ ,  $i, j \in N$  and  $i \neq j$ , representing corridors or passages connecting nodes  $i$  and  $j$ . To simplify the notation, we assume that there is at most one arc in each direction between any pair of nodes.

Let  $O \subseteq N$  and  $D \subseteq N$  be the set of all origins and destinations respectively. We assume that there are  $n_O$  origin nodes  $o \in O$  disjoint from  $n_D$  destination nodes  $d \in D$ , where  $n_O + n_D \leq n$ . Here, origins are all areas with evacuees inside the smart space network while destinations are their near safe exits.

In the definition of evacuation requests, we introduce fictitious sink node  $\hat{d} \in N$  that is adjacent to all the destination nodes (safe exits) by fictitious (dummy) arcs. In this way, we assume that graph  $G$  includes (together with actual nodes) also fictitious node  $\hat{d}$  and its incoming dummy arcs. Then, let  $w \in W$  be a generic evacuation request from node  $o \in O$  to fictitious sink node  $\hat{d}$ , where  $W$  is the set of all evacuation requests. Moreover, let  $R$  be a vector of cardinality  $n_O$  representing evacuation demands from origins  $O$  towards fictitious safe exit  $\hat{d}$ , where  $R_{o\hat{d}} = R_w$  entry indicates the demand of evacuees in unit time period who request to leave origin node  $o \in O$  to go to any of the safe exits  $d \in D$  and, hence, to fictitious destination  $\hat{d}$ .

Our objective is, thus, to safely evacuate all the evacuees and if not possible, then as many people as possible within the allotted time period. To this aim, we should find optimal paths toward safe exits that minimize the evacuation time considering safety of the evacuation areas and thus avoiding

the hazardous conditions that might result in fatalities and/or panic.

Let  $\bar{P}_w$  denote the set of available (simple) paths acceptable in terms of duration cost for each evacuation request  $w \in W$  from origin  $o_w \in O$  to fictitious sink  $\hat{d}$ . By acceptable in terms of duration cost, we mean the paths from an origin  $o \in O$  to safe exits  $d \in D$  considering the upper bound in respect to the minimum duration among the paths for that origin. Furthermore, let  $\bar{P}_W$  be the set of all such paths.

Moreover, let us assume that safety status  $S_a$  is given for each arc  $a \in A$  as a function of safety conditions that can be jeopardized by hazardous conditions as, e.g., natural disaster or terrorist attacks. We normalize it to the range  $[0, 1]$ , such that 1 represents perfect conditions while 0 represents conditions impossible for survival, with a critical level for survival  $0 < S_a^{cr} < 1$  depending on the combination of the previously mentioned parameters. The data quantizing and fusion whose result is the arc safety status is not a topic of this paper. More details can be found in, e.g., Khaleghi *et al.* [2013]; Zervas *et al.* [2011].

The safety optimization problem is related with minimizing the risks caused by possible threats present on the arcs of the paths towards evacuees' safe areas. If each constituent arc  $a$  of path  $k$ ,  $k \in \bar{P}_w$ ,  $w \in W$  has safety  $S_a \geq S^{cr}$ , then path  $k$  is considered to be safe. On the contrary, when safety  $S^k$  on path  $k \in \bar{P}_w$  falls behind threshold value  $S^{cr}$ , its harmful effects may threaten the evacuees' lives. Thus, path  $k$  is considered unsafe and is jeopardized by the safety of its constituent unsafe arcs  $A_k^{cr} = \{a : a \in k, S_a < S^{cr}\}$ .

We are concerned about the number of these unsafe arcs and their safety values in the proposed paths. The proposed paths  $k \in \bar{P}_w$  for  $w \in W$  should all satisfy safety conditions  $S^k \geq S^{cr}$ . However, when such a path is not available, a path with the maximal safety should be proposed where the travel time passed in the safety jeopardized areas should be minimized. Since arcs' safety  $S_a$  can vary significantly within a proposed shortest path, we introduce a normalized path safety that maintains balance between the minimal and average arcs' safety values:

$$S^k = \sqrt[|a \in k|]{\prod_{a \in k} S_a}, \forall k \in \bar{P}_w, w \in W. \quad (1)$$

We want to find a path  $k \in \bar{P}_w$  for each  $w \in W$  that maximizes (1) and minimizes path's evacuation time  $t_k$ , where  $t_k = \sum t_{a \in k}$  and  $t_{a \in k}$  is the travel time of each arc  $a \in k$ . Since the longest path problem is NP hard, we convert the safety maximization to jeopardy minimization problem, where jeopardy  $U^k$  of path  $k$  is defined as  $U^k = 1 - S^k$ .

Then, the objective is to find a temporally efficient path with minimized jeopardy. For this reason, we search for a path with both minimized path's jeopardy and the evacuation time.

Overall path safety for the evacuation request of each origin agent can then be computed by a product of the constituent paths' safeties, Formula 2.

$$\mathbf{S}_w = \sqrt[|k|]{\prod_{k \in w} S^k}, \forall k \in \bar{P}_w, w \in W, \quad (2)$$

## 5 Routes' safety optimization model

Route resilience to contingencies should be provided through the computation of  $k$ -shortest paths in regular time intervals such that evacuees may be simply redirected to a backup path if the proposed path gets dangerous at some node. By computing  $k$  shortest paths from each origin and any intermediate node towards safe exits, we guarantee that the evacuees will be given viable alternatives based on the real-time safety updates. In this light, each origin agent computes  $k$  shortest paths towards safe exits that comply with the requirements on the maximal evacuation time. If an arc or node failure occurs, the route of affected evacuees is changed locally by the node agent that detects the failure.

In the case there are no available safe shortest routes for some origin node, it remains isolated. To resolve this issue, and to maintain the connectivity of origin nodes with safe exits at all times, in the shortest path computation, we multiply the travel time of unsafe arcs for which  $S_a < S^{cr}$  by  $M^{-S_a}$ , where  $M$  is a very large number. In this way, the unsafe arcs will be included in the shortest paths only if there is no alternative path composed of safe arcs. Moreover, the number of the unsafe arcs will be minimal and their safety value will be maximal.

The dynamic component of the evacuation should be included in the computation since the original demand gets lower as the time passes. In this respect, we can assume that an arc is loaded with flow until all the evacuees haven't evacuated the arc.

In the computation of  $k$  shortest paths, we use Yen's algorithm. The time complexity of Yen's algorithm is dependent on the shortest path algorithm used in the computation of the spur paths. For this purpose, we use Dijkstra algorithm. Dijkstra's algorithm has a worse case time complexity of  $O(n^2)$ , but using a Fibonacci heap it becomes  $O(m + n \log n)$ .

After the shortest paths are found for each origin agent, the latter can decide of the evacuees' assignment to the paths based on relevant personal characteristics that guarantee equality through an iterative auction. The negotiation through auctions is local between each origin agent and the evacuees starting their travel at that origin, similar to Lujak *et al.* [2014].

## 6 Conclusions

In this work we studied crowd evacuation coordination problem with the focus on smart spaces. We considered how route safety affects the selection of evacuation routes and their re-configuration in the case of contingencies. In this context, we proposed an architecture for evacuation route safety optimization in large smart spaces that recommends safe and efficient situationally aware routes for evacuation.

If we consider multiple communicating open and closed spaces, this evacuation coordination approach can be potentially applied to different scales in emergency evacuation at a building, district, and urban level. In the future work, we plan to validate the model in relevant simulated scenarios.

## Acknowledgments

This work has been partially supported by the Autonomous Region of Madrid through grant "MOSI-AGIL-CM" (P2013/ICE-3019) co-funded by EU Structural Funds FSE and FEDER, "SURF" (TIN2015-65515-C4-4-R) funded by the Spanish Ministry of Economy and Competitiveness, and through the Excellence Research Group GES2ME (Ref. 30VCPIGI05) co-funded by URJC and Santander Bank.

## References

- Ignacio Aedo, Shuxin Yu, Paloma Díaz, Pablo Acuña, and Teresa Onorati. Personalized alert notifications and evacuation routes in indoor environments. *Sensors*, 12(6):7804–7827, 2012.
- Ibrahim Azhar Mohd, Ibrahim Venkat, KG Subramanian, Ahamad Tajudin Khader, and Philippe De Wilde. Intelligent evacuation management systems: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):36, 2016.
- Alan E Bruce, Kenneth A Cobleigh, and Pauline Joe. Evacuation route planning tool, March 25 2008. US Patent 7,349,768.
- Winnie Daamen, Victor L Knoop, and Serge P Hoogendoorn. Generalized macroscopic fundamental diagram for pedestrian flows. In *Traffic and Granular Flow'13*, pages 41–46. Springer, 2015.
- Antoine Desmet and Erol Gelenbe. Graph and analytical models for emergency evacuation. *Future Internet*, 5(1):46–55, 2013.
- Dirk Helbing and Anders Johansson. *Pedestrian, Crowd and Evacuation Dynamics*, pages 697–716. Springer New York, New York, NY, 2011.
- SP Hoogendoorn, MC Campanella, and W Daamen. Fundamental diagrams for pedestrian networks. In *Pedestrian and Evacuation Dynamics*, pages 255–264. Springer, 2011.
- Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013.
- Victor Knoop and Serge Hoogendoorn. Empirics of a generalized macroscopic fundamental diagram for urban free-ways. *Transportation Research Record: Journal of the Transportation Research Board*, (2391):133–141, 2013.
- Qingsong Lu, Betsy George, and Shashi Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. In *Advances in spatial and temporal databases*, pages 291–307. Springer, 2005.
- Marin Lujak and Sascha Ossowski. Intelligent people flow coordination in smart spaces. In Michael Rovatsos *et al.*, editors, *Multi-Agent Systems and Agreement Technologies: 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Revised Selected Papers*, volume 9571 of *LNCS*, pages 34–49. Springer, 2016.



- Marin Lujak and Sascha Ossowski. On avoiding panic by pedestrian route recommendation in smart spaces. In *Communications and Networking (IEEE BlackSeaCom), 2016 IEEE 4th Int. BlackSea Conf. on*. IEEE, (In press) 2016.
- Marin Lujak, Stefano Giordani, and Sascha Ossowski. Fair route guidance: Bridging system and user optimization. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1415–1422. IEEE, 2014.
- Eve Mitleton-Kelly, Ivan Deschenaux, Christian Maag, et al. *Co-evolution of Intelligent Socio-technical Systems: Modelling and Applications in Large Scale Emergency and Transport Domains*, chapter Enhancing Crowd Evacuation and Traffic Management Through AmI Technologies: A Review of the Literature, pages 19–41. Springer, 2013.
- Rajat Rastogi, Ilango Thaniarasu, and Satish Chandra. Design implications of walking speed for pedestrian facilities. *Journal of transportation engineering*, 137(10):687–696, 2010.
- Alexander Stepanov and James MacGregor Smith. Multi-objective evacuation routing in transportation networks. *European Journal of Operational Research*, 198(2):435–446, 2009.
- Zhang Xiongfei, Shi Qixin, He Rachel, and Ran Bin. Network emergency evacuation modeling: A literature review. In *Optoelectronics and Image Processing (ICOIP), 2010 International Conference on*, volume 2, pages 30–34. IEEE, 2010.
- Evangelos Zervas, A Mpimpoudis, Christos Anagnostopoulos, Odysseas Sekkas, and Stathes Hadjiefthymiades. Multisensor data fusion for fire detection. *Information Fusion*, 12(3):150–159, 2011.

# Urban Traffic Control Assisted by AI Planning and Relational Learning

Alberto Pozanco and Susana Fernández and Daniel Borrajo

Departamento de Informática, Universidad Carlos III de Madrid  
Avda. de la Universidad, 30. 28911 Leganes (Madrid). Spain  
apozanco@pa.uc3m.es, sfarregu@inf.uc3m.es, dborrajo@ia.uc3m.es

## Abstract

Urban Traffic Control is a key problem for most big cities. An inefficient traffic control system can lead to increased traffic congestions that degrade city quality metrics such as average travel time or city pollution. Most common approaches focus on controlling traffic by appropriately setting traffic lights. Current systems in operation range from static control of traffic light phases to adaptive systems based on numeric models. In this paper, we propose an autonomic approach based on declarative automated planning to generate control plans only when the default behavior should be overridden. Planning is complemented with plan execution control and monitoring, replanning, as well as self-adaptive behavior using Relational Learning. Learning is used to anticipate the appearance of congestions and correctly solve them. Our system outperforms static approaches as well as a planning-based system that recently won a competition on autonomic behavior in Urban Traffic Control.

## 1 Introduction

Traffic efficient management and control in urban networks is an important challenge for city authorities. They usually want to achieve a variety of policy-based objectives, such as reducing atmospheric pollution or mitigating the effects of unexpected situations like accidents or road closure. There are many ways to set the traffic lights programs, ranging from early static off-line approaches, to most recent adaptive approaches that change the programs according to the state of the city. The reader is directed to surveys in the area [Papa-georgiou *et al.*, 2007; Hamilton *et al.*, 2013].

From a centralized perspective, Automated Planning (AP) has been recently shown to perform well in this kind of tasks [Gulić *et al.*, 2015; Vallati *et al.*, 2016]. The main advantage of using AP is that the domain and problem descriptions are specified in a declarative language. Thus, even traffic engineers can easily include new actions, sensor information or metrics. Also, these models can be automatically updated by using learning techniques. In this paper we propose an approach that integrates a planning system for controlling traffic lights with a learning system that predicts when

a street density is going to be high in the near future. In those cases, our system anticipates future problems by generating new goals to the planning module and starts a planning-execution-monitoring process. The proposed system can be seen as an instance of a full autonomic (autonomous) system, given that it incorporates many self-\* properties, as self-monitoring (continuous observation), self-diagnosis (detects undesired behavior), self-optimization (planning), self-healing (executes actions) and self-adaptation (learning).

The paper is organized as follows: the next section describes the system architecture that integrates learning with AP; the third section formally defines AP tasks and describes the traffic-control domain; the fourth section briefly describes the learning system; the fifth section presents the experimental results; and the last section draws conclusions and outlines future work.

## 2 Architecture

We propose to use a planning-execution-monitoring architecture called PELEA to provide a framework that can integrate the various components of our system [Guzmán *et al.*, 2012]. Figure 1 shows a sketch of the architecture. At start, the *Execution* module receives an AP domain and problem. Then, it captures the current state of the world, *state*, and sets the problem initial state. The initial goal set could be also set by the *Goal&Metrics Generation* module. The *Monitoring* module calls the *Planning* module to obtain a plan whose actions are sent back to the *Execution* module. Once the actions are executed, the *Monitoring* module receives the necessary knowledge (current state, problem and domain) from the *Execution* module to initialize a new planning-execution-monitoring cycle. If the execution did not produce the expected changes (reduction in traffic density in some streets), it will result in the generation of new goals and a new initial state for a new call to the planner. The *Goal&Metrics Generation* module combines these goals with possible external ones (as the ones given directly by traffic controllers) to update the problem. The environment can be substituted by a *Simulator* in some domains, as the one we focus in this paper.

One of the greatest challenges in the proposed architecture is the generation of new goals. Here, we propose to apply machine learning techniques to infer when new goals should be generated to anticipate future problematic streets. In a training step, examples are generated by observing the traffic be-

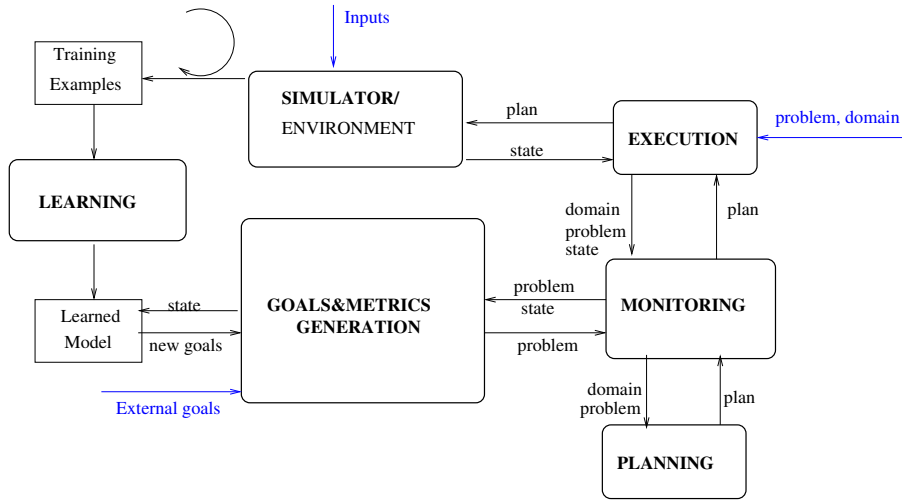


Figure 1: Planning and execution architecture that includes learning capabilities.

havior during some time periods, under different traffic conditions. Then, a learning algorithm can generate a model from those examples, such that given any new state it returns new goals. We are assuming here that the learning process is performed off-line, prior to the actual use of the AP-based system, but it could also be done on-line. The following section formally defines AP tasks and describes the Urban Traffic Control (UTC) domain we are using on this work.

### 3 Planning Tasks

In order to represent planning tasks compactly, the AP community uses the standard language PDDL (Planning Domain Description Language) [Fox and Long, 2003]. Most planners automatically generate an instantiated planning task from the PDDL declarative description of a domain  $D$  and a problem  $P$ . The domain defines the predicates for representing states and the actions that agents can perform. Figure 2 shows an example of an action in the domain definition. The problem describes the task to be solved at each reasoning step; i.e., the objects involved (e.g., streets, traffic lights), the initial state and the set of goals to achieve. Figure 3 shows a subset of a problem definition. The planner will receive both the domain and the problem files as input and it will try to find a solution plan for the given problem. In this case, the output of the planner will be a set of actions to be performed over the traffic lights, such that these actions override the default control program for a certain time period. If the planner has solved the congestion at the next reasoning step, the default program will take the control again. Otherwise, the next actions of the previously generated plan are executed.

This planning model assumes the world is deterministic and the agent has full observability, among other assumptions. In most real-world environments, this is not the case. Actions have stochastic outcomes (the traffic density is not always reduced in the same way when setting a longer green phase in a traffic light), and agents have partial observability (they do not know what the density due to new vehicles

entering the city is going to be in the following time steps). There have mainly been two ways to handle uncertainty. In the first type of models, uncertainty is represented explicitly in the planning model and planners reason with those stochastic models [Bonet and Geffner, 2005]. In the second, planners reason with deterministic world models and when execution of some actions fails, the agent replans [Yoon *et al.*, 2007]. In this paper, we will use the second alternative given that, from a practical perspective, it is good enough for the domain we are focusing on.

### 4 Learning Traffic Behavior

In this section we define the task of learning when goals will arrive; that is, predicting the density level of the streets so we can anticipate their congestion, generating the appropriate goals for the planner. We formulate this problem as a time series prediction one, using Relational Learning in this case. Relational Learning is a Machine Learning technique that can capture the correlations between connected elements. In our case, we conjecture that the structured layout of a city can influence the density levels of some streets based on the ones that are connected to some others. Thus, it is a relational domain. Relational Learning also suits AP, because it allows induction over structured examples that can include first-order logical representations, like the ones used in PDDL.

#### 4.1 Representation

The representation is based on a subset of the predicates we use in the planning traffic domain. In order to represent the time steps, we modify some of these predicates, adding the corresponding time steps. The predicates used for the learning task are shown in Table 1.

We distinguish two types of predicates: the static and the dynamic ones. The static part of the city is represented by the *connection* predicate, that indicates that a vehicle can move from one street section to another. All the *connection* predicates together represent the entire city network. The dynamic

```

(:action hm-green-to-all-ways
:parameters (?t - traffic-light ?c - crossing ?sin - street
             ?sout1 - street ?sout2 - street ?sout3 - street)
:precondition (and (goes-into ?sin ?c)
                  (goes-out ?sout1 ?c)
                  (traffic-lights-from-street ?t ?c ?sin)
                  (not (opposite-direction ?sin ?sout1))
                  (densityLevel ?sout1 moderate)...)
:effect (and (not (state-to-street ?t ?sout1 red))
            (densityLevel ?sin low)...)

```

Figure 2: Part of an example description of a PDDL action.

```

(define (problem traffic1) (:domain traffic)
  (:objects s1 ... s566 - street
            c1 ... c30 - crossing
            t11 ... t110 - traffic-light)
  (:init (goes-into s1 c3)
         (opposite-directions s5 s7)
         (state-from-street t11 s7 green)
         (densityLevel s1 high)...)
  (:goal (and (densityLevel s4 low)
              (densityLevel s35 low) ...)))

```

Figure 3: Part of an example PDDL problem file.

Predicate	Type
density(st,l)	Dynamic
connection(st,st)	Static
open $X$ (tl,st)	Dynamic
density $LX$ (st)	Dynamic

Table 1: Predicates used in the learning task.  $X$  represents the time step.  $L$  represents the density level.

part of the city is formed by the state of the traffic lights and the density of the streets. The  $openX(tl, st)$  predicate represents a green traffic light  $tl$  located at street  $st$  at time step  $X$ . In our approach,  $X$  can take the values from one to three ( $X$  previous time steps, or time windows), but it is a parameter that can be modified to extend or reduce the prediction horizon. The  $densityLX(st)$  predicate indicates that a street  $st$  has a density level  $L$  at time step  $X$ .  $L$  can take the values *veryhigh*, *high*, *moderate*, *low* and *verylow*. The last predicate of each example,  $density(st, l)$ , represents the current density level  $l$  of the street  $st$ . This will represent the class of each example.

## 4.2 Algorithms

We are using TILDE [Blockeel and De Raedt, 1998] to learn relational decision trees. It receives two files as input: the settings file, where the user can specify the algorithm parameters, as well as defining the predicates and classes; and the knowledge base file, where both the training and test data are included. The output of the learning algorithm is a file containing the resulting relational tree and its translation into rules. It also contains the confusion matrix for the training and test sets. An example output of TILDE is shown in Figure 4,

where  $A$  represents the example id and the other letters the predicates' arguments ( $B$  is the street whose density level,  $C$ , we want to predict). A minus symbol predating a variable means that it is new in the tree, while when the variable appears alone, it has to be referenced before. The classes to predict appear in the leaf nodes of the tree between brackets. For example, in the model shown in Figure 4, a high density would be predicted for a street  $B$  in two cases: (1) if its density was low two time steps ago, but there exists another street  $D$  connected to  $B$  whose density was high three time steps ago and was not low in the last time step; and (2) if its density was not low neither two time steps ago nor one time step ago.

```

density (-A, -B, -C)
densityLow2 (A, B) ?
+-yes: densityHigh3 (A, -D) ?
      +-yes: connection (A, B, D) ?
            +-yes: densityLow1 (A, D) ?
                  +-yes: [low]
                  +-no: [high]
            +-no: [low]
      +-no: [low]
+-no: densityLow1 (A, B) ?
      +-yes: [low]
      +-no: [high]

```

Figure 4: Example of TILDE output.

## 5 Experiments and results

On this work we use SUMO [Behrisch *et al.*, 2011], an open source traffic simulator developed by the German Aerospace Center (DLR). It allows to import or generate not only road networks, but also traffic demand. And it also allows users to define traffic lights control programs. We want to test first if we are able to build a model to predict the appearance of goals in advance, and then we try to apply the created model to several urban traffic control scenarios.

### 5.1 Results on Learning Goals

We are using a real city network in our learning experiments; a grid-like section of Houston downtown, shown in Figure 5. It is composed of 35 junctions, 140 traffic lights and 164 street sections. We have selected five particular street sections to learn from (A to E). We chose these city points due to their different traffic characteristics. C and D are street sections close to a Job Center. B is a point between the Job Center and the main exit of the city. E represents a street section far from the main traffic, while A is a random point with no specific features.

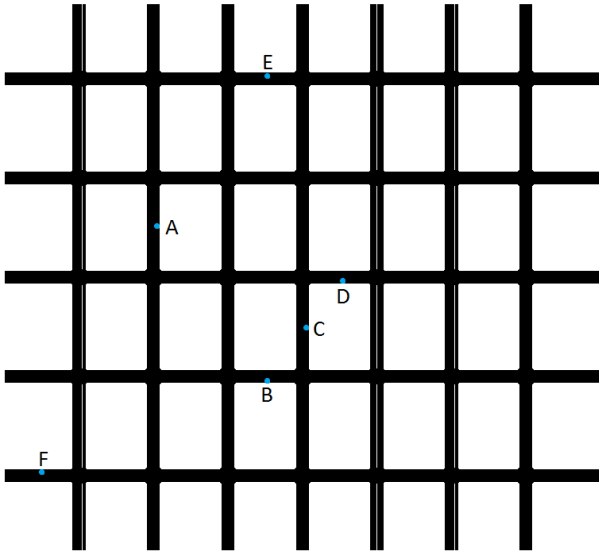


Figure 5: Benchmark network in SUMO. Models are created for points A, B, C, D and E. We assume that a Job Center is located on D. F corresponds to the main exit point of the city.

We have also defined a traffic demand that tries to emulate the real traffic flow of a city for an entire week. So, we define lower vehicles traffic at night, more traffic at rush hours, and higher traffic during week days than in the weekend. The Job Center is included, where most of the cars want to go during the work hours and also a main exit point, to go out of the city at the end of the workday. The rest of the routes are randomly generated. The vehicles may enter the city by any street section and can finish their trip in an inner (parking, mall, office...) or outer point of the network. A summary of the full traffic demand specification is shown in Figure 6.

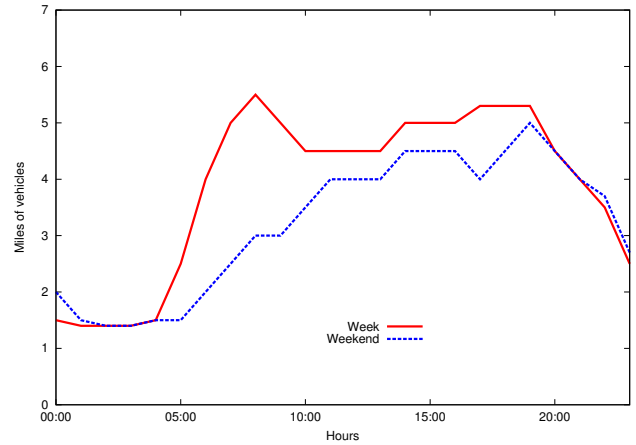


Figure 6: Summary of the generated traffic flows on weekdays and weekends. The y axis represents the number of vehicles that enter the network at each hour, in thousands, and the x axis represents the hours.

Data is collected every five minutes for the learning task, which means 2013 instances for the whole week. Five minutes is what we call “time step”, the sample frequency. We have chosen this sample time as we want to collect traffic data from an entire week, and, at the same time, we want to keep a not very high number of instances so that TILDE is able to handle them. In our experimental setting, a step in the simulation corresponds to a second. Each instance stores the static part of the city previously described, as well as the dynamic component of the state in the last three time steps. We learn one relational model for each street section shown in Figure 5, and then we test with data of the other street sections.

We have also varied the density levels, both in the classes to predict and the predicates used on each instance. We have used two approaches. One is based on five density levels: *veryhigh*, *high*, *moderate*, *low* and *verylow*. A second version uses only two: *high* and *low*. All the generated models are pre-pruned, limiting the creation of new branches when the node has less than 10 instances.

In the first experiment, we generated five different models using data from the five selected street sections and the five density levels approach. And we tested these models in the five street sections to check accuracy and generality of the learned models. The results for this first configuration are on Table 2.

We can observe that the accuracy is similar for all the street sections except for B, whose behaviour seems to be more difficult to predict. A and E, the two points away from downtown and the Job Center, present a similar behaviour as expected.

In the second experiment, the problem is simplified with only two density levels both for the class and the state predicates. The results for this last configuration are on Table 3.

We can observe that as we decrease the number of density levels, the complexity of the problem decreases too and the prediction task becomes easier. With only two levels, the density of a street knowing the state of the city in the last time

	A	B	C	D	E
A	0.90	0.68	0.85	0.77	0.83
B	0.82	0.72	0.79	0.77	0.80
C	0.83	0.66	0.88	0.77	0.81
D	0.80	0.66	0.83	0.85	0.81
E	0.87	0.66	0.85	0.78	0.89

Table 2: Accuracy results using the model obtained with five density levels. Each cell  $(i, j)$  represents the estimated accuracy of learning a model with the data extracted at point  $i$  in the city and testing that model against the data collected at point  $j$ .

	A	B	C	D	E
A	0.99	0.94	0.99	0.97	0.99
B	0.99	0.95	0.99	0.98	0.99
C	0.96	0.93	0.99	0.97	0.99
D	0.96	0.93	0.99	0.98	0.99
E	0.96	0.93	0.99	0.97	0.99

Table 3: Accuracy results using two density levels for the class and the predicates. Each cell  $(i, j)$  represents the estimated accuracy of learning a model with the data extracted at point  $i$  in the city and testing that model against the data collected at point  $j$ .

steps can be predicted with a high accuracy, even in street sections that have very different behavior. The final model that will be used in our architecture corresponds to the one learned with the data of point B, which on average performs best. The relational tree was shown in Figure 4.

## 5.2 Results on Traffic Management

Finally, we want to test whether a traffic control system would improve its performance if it had some predictive model of the traffic. To do so, we will use several simulation scenarios where we vary the size of the network (medium and large), the fluency of traffic (fluent or congested) and the evaluated time period (an hour and a day).

When using the learned model, it predicts the density at each street at each time step, using the previous  $X$  time steps as input. If it detects a high density at any subset of the street sections, it generates goals to lower the density of those street sections. These new goals, together with the current state of the traffic, create a PDDL planning problem that is given as input to the planner. Therefore, the system is predicting the appearance of goals in the next  $X$  time steps, and the planning process can anticipate to the congestions. We will call this new approach `Learning`. In [Pozanco *et al.*, 2016], we show that if the system uses a short-horizon prediction, having the same time steps for both building the model and checking for goals is not that important. So, our system checks for new goals every fifty seconds using the prediction model built with the five minutes time step previously described.

We compare our system with a `Static` one, that corresponds to the default system used by SUMO. We also compare our approach with a `Reactive` system, that acts locally on each traffic light and sets a longer green phase on those

whose their corresponding street density is currently high. We also compare with the AP approach proposed in [Gulić *et al.*, 2015], co-winner of the ARTS-COST competition on *Increasing the resilience of road traffic support systems by the use of autonomics*<sup>1</sup>. That planning system does not have any learning component and only calls the planner when a vehicle has been stopped for a long time. We will call it `Planning`. This system is the starting point of our approach, so we use the same planning domain and planner, LAMA [Richter and Westphal, 2010]. The last system we introduce in the tests combines the `Planning` approach and the `Learning` one. It calls the planner when a goal (high density) is predicted or the current density of a street is high. We will refer to it as `Combined`.

We use the following metrics to measure the performance of each system: the number of steps it takes all cars to reach their destination; the total amount of  $C0_2$  emitted by the vehicles; the average waiting time (AWT); the average travel time (ATT); and, if it applies, the number of planner executions (PE) and the mean planner execution time (MPE). We choose them simply for comparison, none of the systems explicitly reasons on optimizing these metrics.

### Experiments in a Medium-Sized City Network

We created a fluent traffic scenario for the first experiment by introducing 5300 cars in 3600 steps in the same city network we used in the learning goals experiments. The simulation finishes if all cars reach their destination, or after 5000 steps. The results are shown in Table 4. We can see that there is no substantial difference when the traffic is fluid among the different systems. But the `Learning` approach outperforms the others on most metrics. So, when the traffic is fluent, one expects that even the `Static` control program will perform well. In this traffic situation, the time spent on average per vehicle in a traffic light (AWT) is approximately half of the total time spent in their complete travel (ATT). Given the size of the example network, ATT is around three minutes, while AWT is around a minute and a half. The number of planner executions is low in the `Planning` and `Learning` systems, and it becomes very high when using the `Combined` approach. The number of times it calls the planner is much higher than in the two other approaches, as expected.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	3969	1103	93	172		
Reactive	4059	1137	100	181		
Planning	4070	1117	95	175	22	10
Learning	3881	1090	88	167	15	10
Combined	4104	1193	115	197	61	10

Table 4: Performance of the different control systems with a fluent traffic situation in a medium-sized city. Steps, AWT and ATT are given in steps (seconds), while  $C0_2$  is in kg. MPE is in seconds.

In the second experiment, we test the systems performance on a very congested traffic scenario using the same city network. It was created by introducing 6000 cars in one hour

<sup>1</sup><https://helios.hud.ac.uk/cost/comp2.php>

(3600 steps). The results are reported in Table 5. The columns report the same metrics as the one before.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	-	2553	582	638		
Reactive	4106	1262	119	202		
Planning	-	2187	435	506	48	11
Learning	4070	1265	121	204	46	10
Combined	4244	1301	128	212	68	11

Table 5: Performance of the different control systems with a very congested traffic situation in a medium-sized city. Steps, AWT and ATT are given in steps (seconds), while  $C0_2$  is in kg. MPE is in seconds.

As we can see, even if the `Planning` approach outperforms the `Static` system, it performs worse than the `Reactive` mechanism and the two other autonomic approaches. Both `Learning` and `Combined` can completely solve the traffic congestion. The vehicles spend much more time waiting on average than travelling in this scenario (relation between ATT and AWT). However, the `Learning` system is able to reduce the waiting time to half of the travel time, as in a fluent traffic situation. Thus, it is effectively converting a congested situation into a fluent traffic scenario. The reduction of the pollution achieved by `Learning` is quite substantial too: half of the  $C0_2$  levels of the static approach. In fact, they are close to those generated in a fluent traffic scenario. `Reactive` obtains practically the same results than the `Learning` approach, even if it only acts locally at each traffic light without considering the whole network.

### Dense Traffic in a Large Size City Network

This experiment tests the scalability of the proposed model to larger city networks. The benchmark network in this case is composed of 130 junctions, 520 traffic lights and 566 streets. This can be considered as a large network in relation to most papers in the field, specially considering that our approaches perform centralized planning. The network is shown in Figure 7. We introduce 13,000 cars in one hour in order to create a dense traffic situation. As the city is bigger than the previous one, a experiment will finish when all cars reach their destination or after 6,000 time steps. Table 6 reports the results.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	-	6649	439	549		
Reactive	-	7676	605	709		
Planning	-	5520	341	468	50	46
Learning	5837	5231	321	445	47	44
Combined	-	6279	518	633	64	54

Table 6: Performance of the different control systems with a dense traffic situation in a large-sized city network. Steps, AWT and ATT are given in steps (seconds), while  $C0_2$  is in kg. MPE is in seconds.

In this case, `Learning` outperforms the rest and it is the only one that can finish the simulation before 6,000 steps. The model we learned with the medium-sized urban network

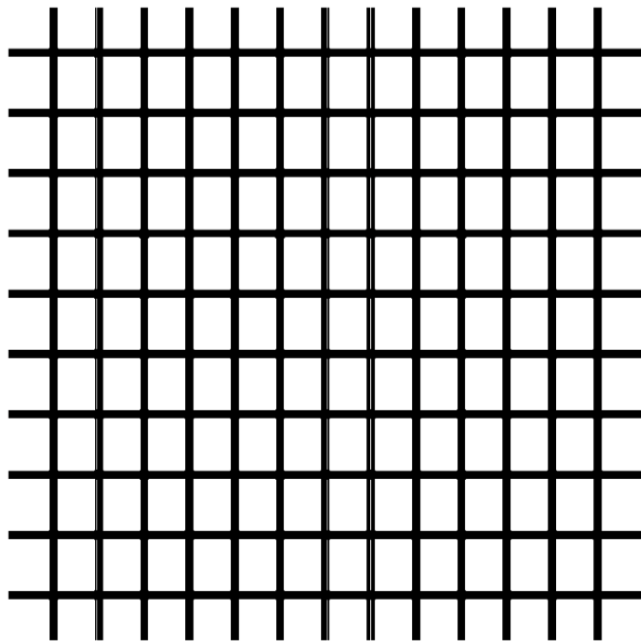


Figure 7: Large city network used in the second type of experiments.

is able to generalize to this larger city. Our system scales quite well even in a large network; it can find a plan in less than fifty seconds, the checking-for-goals sample period. The performance of `Planning` is quite good in this case and it almost solves the congestion. Thus, this only-planning approach works well when we have a reasonably high traffic density (as in this experiment or in the first one), but not too high (as in the previous experiment). The `Reactive` method does not scale up well to the large city network. When trying to locally reduce the congestion, it ends up generating traffic jams and performing even worse than the default, `Static`.

### Full day experiment

The last experiment focuses not only on trying to handle a traffic peak, but also to test whether a system can deal with a full day traffic flow. In these cases, the decisions spread over time. We use the medium-sized city network and a traffic demand specification similar to the one presented on Figure 6 for the week days. In this experiment we only measure the AWT per hour. The other metrics could be irrelevant for the 24 hours case. The results are reported on Figure 8. Vehicles routes remain static in SUMO. A car will always try to reach its destination following the shortest path. If this route is congested, the vehicle will not choose another one, but it will stand still waiting for the route to be free. That is the reason why, when using some systems, the network can get congested at some time point and become congested for the whole day. We can see this effect when a given curve in the graphic reaches 200 s. When using this metric, a traffic system performs better if the area under its curve is smaller.

As we can see, only our `Learning` system is able to finish the simulation properly. The AWT grows up in the morn-



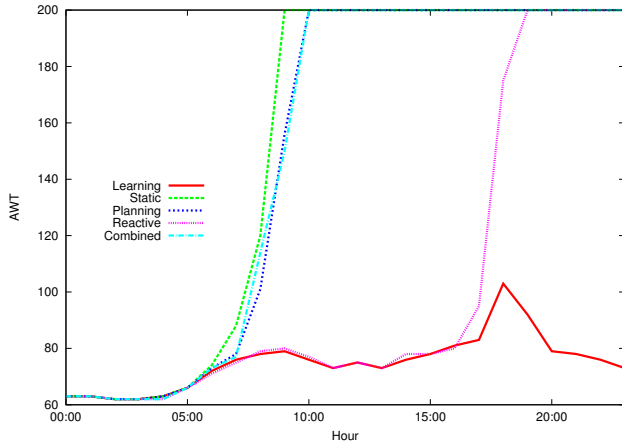


Figure 8: Average waiting time in the city network per hour.

ing, when the cars go to the Job Center, but it does not get fully congested. The AWT remains around 80 s throughout the morning and it starts growing again by the end of the workday. The metric reaches a peak around 18:00 where the AWT is 103 seconds at the most congested traffic situation of the day, which is still a reasonable behavior. After that time period, the system is able to reduce the congestion and the AWT starts to decrease. The *Reactive* system, which showed good performance in the medium-sized city network, can solve the early morning traffic problem. It obtains similar results to the ones of *Learning* until the end of the workday. However, it cannot deal correctly with the end of the day traffic. The other systems can not face the morning rush hour. Even if the *Planning* system is still better than the other two, it does not solve the congestion.

## 6 Related work

The first UTC models in the 1950s and 1960s, were based on fixed-time traffic lights control mechanisms. Actions were predefined following an off-line optimization using historical data of demand levels. TRANSYT [Robertson, 1969] is one of the most well developed and widely used control systems that uses these techniques. These approaches could even generate “green waves”, simple coordination of neighbouring traffic lights in order to increase the traffic fluidity. The problem of early systems is that they can age rapidly due to the continuous evolution of the traffic flows in a city. The benefits may be lost in some years if the control plans are not updated. Our proposed system overcomes this situation, as it not only can react to the current traffic scenario, but it can anticipate and adapt to future ones.

In the last years, the use of new and better sensor systems has allowed engineers to implement traffic-responsive systems that use the data provided by the detectors in an on-line way. These techniques range from centralized approaches, as SCOOT [Bretherton *et al.*, 1998] and SCATS [Lowrie, 1990] to distributed ones as UTOPIA [Donati *et al.*, 1984]. As most other traffic-responsive systems, they use a mathematical framework to compute the optimal time allocation of each

traffic light. A weak point of these systems is that they cannot predict incidents and they do not deal well with them. Also, their models are not defined declaratively. Thus, our models are easier to update with new types of information, or new metrics to be taken into account when optimizing.

Other AI-related approaches have appeared in recent years. The main goal is to build semi- or fully autonomous systems with little human assistance. Most of them address traffic management from a multi-agent perspective. A single agent acts over a single junction or subset of junctions and then several agents collaborate, discuss and negotiate with the rest [Ossowski *et al.*, 1998]. In [Box and Waterson, 2012], the authors propose a model based on logistic regression and neural networks to learn over time how to better control the traffic signals. Other approaches focus on multi-agent reinforcement learning [Kuyer *et al.*, 2008], distributed geometric fuzzy systems [Gokulan and Srinivasan, 2010] or creating a multi-agent model predictive control [de Oliveira and Camponogara, 2010]. New approaches for efficient UTC are arising in the last years using vehicle communication as the core of the control process [Ferreira *et al.*, 2010]. But, these methods are still far from being implemented in real cities and controlling traffic lights remains the most widespread way to handle urban traffic.

## 7 Conclusions and Future work

In this paper we have presented a dynamic approach for UTC based on Automated Planning and Relational Learning. As we have shown, by adding a learning component that can predict the city state to a planning system, we can highly increase its autonomy. It can automatically generate its own goals, in addition to letting the planner starts the planning process sooner. We have tested our model in several traffic control scenarios, showing that the ability to anticipate goals can lead to better control performance than using only static traffic lights programs. Our system also outperforms the *Planning* system and overcomes its limitations, as *Planning* needs to know when a vehicle has been stopped for a long time. Instead, our model only needs the street density levels, which are easier to obtain from current sensor systems. By just knowing density levels, we are able to model a wide variety of circumstances that affect traffic behavior such as adverse weather conditions or different days and hours. Also, since other types of incidents (e.g., road-blocking or big accidents) indirectly affect the density levels, we believe our approach could also work to alleviate congestions caused by them.

In future work, we would like to integrate the ability to learn how to anticipate goals with externally supplied goals (e.g., by traffic controllers), reactively generated ones (e.g., reactively generating goals), or internally supplied ones (e.g., generated by internal motivations of the system). Although the proposed system scales up, we would also like to apply a multi-agent approach by dividing the city in sections in which an agent can apply the system in an autonomous way. We think this could lead to similar performance with lower execution times. We would also like to compare our system with other state of the art methods on traffic control, such as

model predictive control (e.g., SCOOT), or other AI-based approaches (e.g., reinforcement learning). Finally, we want to test the proposed system in irregular city networks such as European ones and build the learning model on-line in order to show the system's real-world applicability.

## Acknowledgements

This work has been partially supported by MINECO project TIN2014-55637-C2-1-R.

## References

- [Behrisch *et al.*, 2011] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo—simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, 2011.
- [Blockeel and De Raedt, 1998] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial intelligence*, 101(1):285–297, 1998.
- [Bonet and Geffner, 2005] Blai Bonet and Héctor Geffner. mGPT: A probabilistic planner based on heuristic search. *JAIR*, 24:933–944, 12 2005.
- [Box and Waterson, 2012] Simon Box and Ben Waterson. An automated signalized junction controller that learns strategies from a human expert. *Engineering applications of artificial intelligence*, 25(1):107–118, 2012.
- [Bretherton *et al.*, 1998] R. Bretherton, K. Wood, and G.T. Bowen. Scoot version 4. In *Proceedings of 9th International Conference on Road Transport Information and Control*, 1998.
- [de Oliveira and Camponogara, 2010] Lucas de Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.
- [Donati *et al.*, 1984] F Donati, Vito Mauro, G Roncolini, and M Vallauri. A hierarchical decentralized traffic light control system. the first realisation "progetto torino". In *Proceedings of the 9th World Congress of the International Federation of Automotive Control*, pages 2853–2858, 1984.
- [Ferreira *et al.*, 2010] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNetworking*, pages 85–90. ACM, 2010.
- [Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of AI Research*, 20:61–124, 2003.
- [Gokulan and Srinivasan, 2010] Balaji Parasumanna Gokulan and Dipti Srinivasan. Distributed geometric fuzzy multiagent urban traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 11(3):714–727, 2010.
- [Gulić *et al.*, 2015] Matija Gulić, Ricardo Olivares, and Daniel Borrajo. Using automated planning for traffic signals control. In *Working Notes of ARTS-COST 2nd competition*, 2015.
- [Guzmán *et al.*, 2012] César Guzmán, Vidal Alcázar, David Prior, Eva Onaindía, Daniel Borrajo, Juan Fdez-Olivares, and Ezequiel Quintero. PELEA: a domain-independent architecture for planning, execution and learning. In *Proceedings of ICAPS'12 Scheduling and Planning Applications workshop (SPARK)*, pages 38–45, Atibaia (Brazil), 2012. AAAI Press.
- [Hamilton *et al.*, 2013] Andrew Hamilton, Ben Waterson, Tom Cherrett, Andrew Robinson, and Ian Snell. The evolution of urban traffic control: changing policy and technology. *Transportation planning and technology*, 36(1):24–43, 2013.
- [Kuyer *et al.*, 2008] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Machine learning and knowledge discovery in databases*, pages 656–671. Springer, 2008.
- [Lowrie, 1990] PR Lowrie. Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.
- [Ossowski *et al.*, 1998] Sascha Ossowski, José Cuenca, and Ana García-Serrano. A case of multiagent decision support: Using autonomous agents for urban traffic control. In *Progress in Artificial Intelligence—IBERAMIA 98*, pages 100–111. Springer, 1998.
- [Papageorgiou *et al.*, 2007] M Papageorgiou, M Ben-Akiva, Jon Bottom, Piet HL Bovy, SP Hoogendoorn, Nick B Hounsell, Apostolos Kotsialos, and M McDonald. Its and traffic management. *Handbooks in Operations Research and Management Science*, 14:715–774, 2007.
- [Pozanco *et al.*, 2016] Alberto Pozanco, Susana Fernández, and Daniel Borrajo. On learning planning goals for traffic control. In *4th Workshop on Goal Reasoning (IJCAI'16)*, 2016.
- [Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.
- [Robertson, 1969] Dennis I Robertson. Transyt: a traffic network study tool. 1969.
- [Vallati *et al.*, 2016] M. Vallati, D. Magazzeni, B. De Schutter, L. Chrapa, and T.L. McCluskey. Efficient macroscopic urban traffic models for reducing congestion: a pddl+ planning approach. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.
- [Yoon *et al.*, 2007] Sungwook Yoon, Alan Fern, and Robert Givan. FF-replan: A baseline for probabilistic planning. In *ICAPS*, pages 352–360, 2007.

# A Novel Approach of Cognitive Base Station with Dynamic Spectrum Management For High-speed Rail \*

Qingting Wu, Yiming Wang, Zhijie Yin, Hongyu Deng, Cheng Wu<sup>†</sup>  
School of Urban Rail Transportation, Soochow University, Suzhou, China

## Abstract

The characteristic of fast movement in high-speed rail seriously affects the stability of vehicular wireless communication. Applying cognitive technology to individual users often brings frequent channel switch and inefficient blind learning. To address these issues this paper proposes a novel concept of Cognitive Base Station (CBS), which has the capability of forecasting spectrum holes and assigning spectrum to individuals. We then give the model of cognitive base station and evaluate the performance in our simulation platform within high-speed rail environment. The experiment results further prove that the model can significantly improve the performance of vehicular communication.

## 1 Introduction

With the development of era, the demand for rail transit is rapidly increasing. When travelling on train, the passengers always hope to enjoy better communication quality and faster data access service. European Rail Traffic Management System (ERTMS) is a revolution in railways to guarantee the communication, which is consist of European Train Control System (ETCS) and a mobile-communications network optimized for railways called GSM-R.

GSM-R is the Global System for Mobile Communications-Railway in the worldwide and is dedicated to provide the bidirectional radio bearer for the train signaling systems, which operates in a 4MHz band (876-880 MHz for uplink and 921-925 MHz for downlink) [Sniady and Soler, 2012]. It is possible to divide the authorized band into 19 channels of 200KHz width in each GSM-R group. The rail line is covered with GSM-R groups and each consists of many GSM-R cells. A single GSM-R cell can use only few of the channels in a round robin manner, because the same channel cannot be reused by neighboring cells due to interference. Each cell is equipped with a base station. The base station is made up of building baseband unit (BBU) and radio remote unit (RRU). RRU is

always deployed outside along the railway and BBU is inside. One BBU is connected to multiple RRUs. BBU and RRU are used to process baseband signal and radio frequency signal, respectively. To ensure the communication between RRU and passengers, two vehicular stations (VS) are installed on the top and final carriages of the train. The network architecture is illustrated in Fig. 1 [Isheng Zhao *et al.*, 2013], [Tian *et al.*, 2012]. The GSM-R system consists of base transceiver stations (BTS) along the railway lines and embedded GSM-R mobiles connected to antennas on the roof of the trains. The train has to be permanently connected to the trains control center. This connection has a high priority level, and if the modem connection is lost, the train stops automatically [Dudoyer *et al.*, 2012].

However, under the circumstance of high-speed railway [Zhang *et al.*, 2012], vehicular communication often shows unstable, even sometime dreadful [Ai *et al.*, 2014]. Usually, when the speed is up to 350 kilometers per hour, there unavoidably arises some issues, such as Doppler shifts, fast cell switching and the penetration loss [Zhou and Ai, 2014]. The Doppler shifts results from the relative motion between a vehicle and a base station. Doppler Effect becomes another pivotal factor degrading system performance, which increases randomness of received signal [Liu *et al.*, 2011], [Li and Zhao, 2012], [Dybala and Radkowski, 2013]. The high speed operation of the train leads to fast cell switching. As a train moves across the footprint of the satellite beam, the receiving signal level may vary, especially towards the edge of the beam, which significantly impacts service rates even causing service drops [Li *et al.*, 2013], [Alkayal and Saada, 2013]. The fully enclosed body structure with good sealing property of the high-speed train results in penetration loss. Typically, the terminals inside the train connect to the base stations along the railway tracks via wireless links, in which the large penetration loss will directly degrade the communication link quality and decrease the cell coverage [Zhu *et al.*, 2013], [Liu *et al.*, 2012]. Furthermore, Federal Communications Commission (FCC) released the investigation on the usage of spectrum In 2003. It suggested that the authorized band in 3 – 6GHz range is less than 0.5% utilized on average. And so is the band below 3GHz, which is less than 35% [Commission and others, 2003]. Just based on these viewpoints, it is necessary to introduce a novel architecture for high-speed vehicular communication to address the issues

\*Project supported by the National Nature Science Foundation of China (No. 61471252) and the Natural Science Foundation of Jiangsu Province (No. BK20130303).

<sup>†</sup>Corresponding Author: cwu@suda.edu.cn

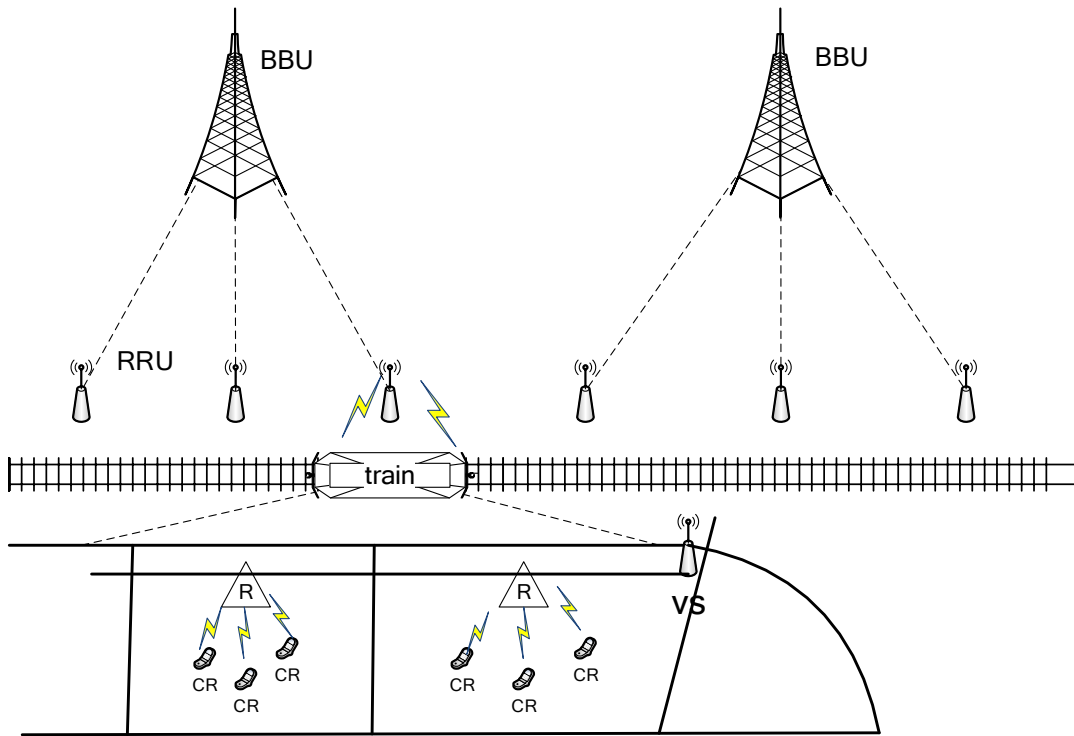


Figure 1: Networks architecture for the high-speed rail communication system.

from individual user's high-speed movement along the rails and the inefficiency in the spectrum usage.

In recent years, a lot of researchers used cognitive radio (CR) to improve the performance of wireless communication. The basic idea of CR networks is that the unlicensed devices (also called cognitive radio users or secondary users) need to vacate the spectrum band once they detect the licensed devices (also known as primary users). Simon Haykin defined the CR as an intelligent wireless communication system that is aware of its environment and uses the methodology of understanding-by-building to learn from the environment and adapt to statistical variations in the input stimuli [Haykin, 2005]. Letaief presented a cognitive space-time-frequency coding technique that can opportunistically adjust its coding structure by adapting itself to the dynamic spectrum environment [Letaief and Zhang, 2009]. Soyeon Kim proposed a CR operational algorithm for mobile cellular systems, which was applicable to the multiple secondary user environment [Kim and Sung, 2014]. These results proved CR technology can significantly reduce interference to licensed users, while maintaining a high probability of successful transmissions in a cognitive radio (CR) ad hoc network.

There are few publications about applying CR to the field of urban rail transit. Wu proposed a wireless cognitive model for high-speed individuals' spectrum management and show a small performance improvement in wireless communication [Wu *et al.*, 2015]. Although using cognitive radio in high-speed-railway has improved the performance, there are still so many issues that are open to address:

(1) Most of the cognitive radio users usually sense in the

same environment and each user is independent. So they compete each other for the spectrum resources, which leads to blind learning and frequent conflicts.

- (2) The rail transit contains a large number of CR users. While every user senses the environment, the system works with heavy workload and high computational complexity.
- (3) The operations of mutual competition and cooperation between the CR users interfere with not only primary users, but also themselves and their neighbors.
- (4) Spectrum holes in each base station are different. It would inevitably occur spectrum handoff.

For addressing the above issues, we try to propose a novel model of cognitive base station in the paper. Our proposed CBS attempts to use the authorized bands for railway without interrupting PUs. The CBS model should satisfy the following conditions:

- (1) The CBS can forecast spectrum holes according to its experience and assign spectrum to individuals within its range of coverage. In this way, the computational complexity of the entire network can be reduced.
- (2) The rail transit runs daily over a fixed route according to its timetable. The CBS can take the advantage of these characteristics, cooperate with each other to forecast spectrum holes on the whole route.

This paper is organized as follows. We first introduce the concept of cognitive base station and its mathematical model in Section 2. Section 3 then applies the novel CBS model

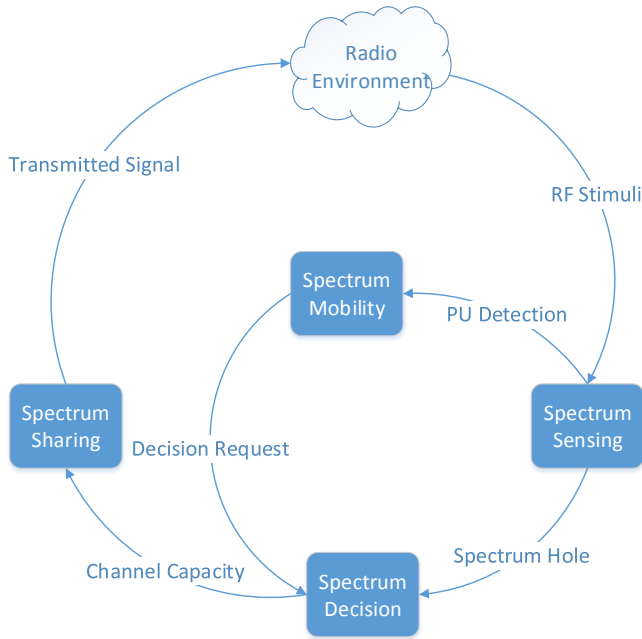


Figure 2: The cognitive cycle of a cognitive base station.

with RL into the scenario of high-speed rail, and propose the cooperation mechanism of multiple CBS agents. The experimental simulation results are given in Section 4. We conclude this paper in Conclusion.

## 2 Cognitive Base Station Model

Our proposed CBS is deployed along the railway, which works as a spectrum assigner. It learns from feedback received through interactions with an external environment and assigns spectrum to the passengers in the range of coverage. We consider each CBS to be an agent, which has four spectrum management functions: spectrum sensing, spectrum mobility, spectrum decision and spectrum sharing [Chkribene and Hamdi, 2015], [Lee and Akyildiz, 2012]. Fig. 2 gives the steps of the cognitive cycle within the framework of CBS, which is formed by the spectrum-aware operations. Each CBS agent uses reinforcement learning to operate spectrum management. All of the agents can sense the environment, obtain its own current state about spectrum usage, and communicate with each other for the purpose of cooperation. They then make decision according to its own state and the whole network situation, then use spectrum mobility to choose actions. Finally, these CBS agents continue to send its new state to the other neighbor CBS agents.

We assume that our cognitive radio network along high-speed rail consists of a collection of CBS agents and CR user agents. Each CBS agent has its own PUs and available spectrums. The CBS agents undertake decisions on choosing the spectrum independently of the CR user agents in the range. A choice of spectrum by the CBS agent  $i$  is essentially the choice of the frequency represented by  $f^i \in F$ . The CR user agents continuously monitor the spectrum that the CBS agent choose in each slot time. We assume perfect sensing,

in which, the CBS agents correctly infer the presence of the PUs if the former lies within the PUs' transmission range.

- Long-term Awareness of Spectrum Usage

Characterizing the spectrum bands based on their activity, and in particular, learning about the utilization of the channel is a key function of the CR users. Online learning algorithms must be developed that allow the CBS agents to continuously gather information about its radio environment, and construct a utilization function. Apart from simply classifying the spectrum as *busy* or *available*, it is beneficial if a probability distribution of the anticipated transmission/silent durations of the PUs can be derived. We propose a tightly integrated reinforcement learning equipped link layer protocol to schedule the transmissions between CBS agents and CR user agents over time.

- End-to-End Learning

Distributed networks rely on multihop forwarding of packets between a source-destination pair. Each CBS agent on this path learns of its own spectrum environment over time, and this information can be leveraged at the start and end points of the path to make optimal decisions regarding the spectrum choices and routing options. As an example, spectrum switching costs locally at a node affects end-to-end delays. While spectrum characteristics can be locally inferred, the specific choice of the spectrum at each link to minimize intra-path switching must be undertaken at the end points of the path. We explore ways to share this learning and spectrum awareness obtained by a node between its local neighbors, and subsequently over multiple hops to the destination. The cost of this learning and the benefits are investigated as part of this project.

## 3 SPECTRUM MANAGEMENT BASED COGNITIVE BASE STATION

### 3.1 The Q-Learning

Reinforcement learning, which is inspired by psychological learning theory from biology [Waltz and Fu, 1965], enables the agent to learn behavior through trial-and error interactions with a dynamic environment [Sutton and Barto, 1998]. The classical reinforcement algorithm is Q-Learning, the process of which is as follows [Puterman, 1994]. On each step of interaction the agent chooses an action according to the external environment based on its current state. As a result, the action changes the environment and receives a reward. The agent need to develop a policy, that maximizes the long-run measure of reinforcement.

The classic reinforcement learning algorithm is formulated as follows. At each time  $t$ , the agent perceives its current state  $s_t \in S$  and the set of possible actions  $A_{s_t}$ . The agent chooses an action  $a \in A_{s_t}$  and receives from the environment a new state  $s_{t+1}$  and a reward  $r_{t+1}$ . Based on these interactions, the reinforcement learning agent must develop a policy  $\pi : S \rightarrow A$  which maximizes the long-term reward  $R = \sum_t \gamma r_t$  for MDPs, where  $0 \leq \gamma \leq 1$  is a discounting factor for subsequent rewards. The long-term reward is

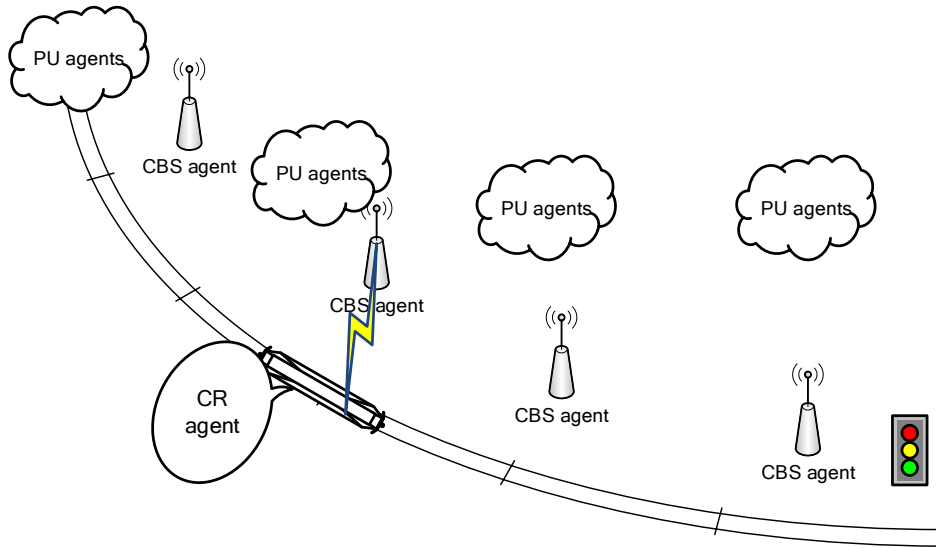


Figure 3: The cognitive base station within the high-speed-rail transportation.

the expected accumulated reward that the agent expects to receive in the future under the policy, which can be specified by a *value function*. In this way, the Q-learning can calculate an update to its expected discounted reward,  $Q(s_t, a_t)$  as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where  $\gamma$  is the discount factor such that  $0 \leq \gamma < 1$ . The agent stores the state-action values in a table  $Q$  [Wu *et al.*, 2010], [Jiang *et al.*, 2011], [Bkassiny *et al.*, 2013].

Recently the reinforcement learning has attracted increasing interest in the machine learning and artificial intelligence communities. Kadam *et al.* applied the Q-Learning into routing data in Wireless Sensor Network scenario to route data efficiently from one source to multiple mobile sinks [Kadam and Srivastava, 2012]. It turned out that the algorithm can extend the network lifetime.

### 3.2 Application to Cognitive Base Station

We illustrate the high-speed railway environment with CBS agents along the way in Fig. 3. We further model a cognitive radio network as consisting of a set of Cognitive Base Stations, denoted *CBS*, a set of primary users, denoted *PU*, and a set of available frequencies, denoted *SP*. We assume that the topological structure of a given network is fixed.

Spectrum holes vary due to the behavior of PUs, which causes the change of environment. CBS agents can perceive the states within the environment. The state of an CBS agent is the current spectrum of its transmission. The state of the multi-agent system includes the state of every CBS agent. We therefore define the state of the system at time  $t$ , denoted  $s_t$ , as

$$s_t = (\vec{s}_p)_t$$

, where  $\vec{s}_p$  is a vector of spectrums across all agents. Here  $sp_i$  are the spectrum on the  $i$ th agent and  $sp_i \in \vec{SP}$ . Nor-

mally, if there are  $m$  spectrums, we can use the index to specify these spectrums. In this way, we have  $\vec{SP} = \{SP_1, SP_2, \dots, SP_m\}$ .

At a particular time and a particular state, the CBS will take action according to learning results to either switch channel or transmit. At time  $t$  we define  $a_t = k$ , where  $k$  is the action that CBS chooses at time  $t$  and

$$k \in \{switch\_to\_channel_1, switch\_to\_channel_2, \dots, switch\_to\_channel_m, transmit\_data\}.$$

Once the CBS agent has detected any active PU, it would take action to channel switching. We use the  $Q$  table to store state-action values. At time  $t$ , the state is  $sp_t$  and the action is  $k$ , then we can calculate the value  $Q(sp_t, k)$  by the above Q-learning formulas. If PU is detected, the CBS agent would switch to the other available spectrum with the largest  $Q$ -value.

The reward is the estimate for spectrum usage availability on a CBS agent. The different network situation results in different rewards as follows.

- *CR-PU interference*: If a PU's activity occurs in the spectrum shared by any CR user, and in the slot same selected for transmission, then a high penalty of  $-15$  is assigned. The intuitive meaning of this is as follows: We can avoid the collisions among the CR users using the mediation from the CBS agents. However, the concurrent use of the spectrum with a PU goes against the principle of protection of the licensed devices, and hence, must be strictly avoided.
- *Successful Transmission*: If none of the above conditions are observed to be true in the given transmission slot, then packet is successfully transmitted from the sender to receiver, and a reward of  $+5$  is assigned, which is found experimentally to give the best results.



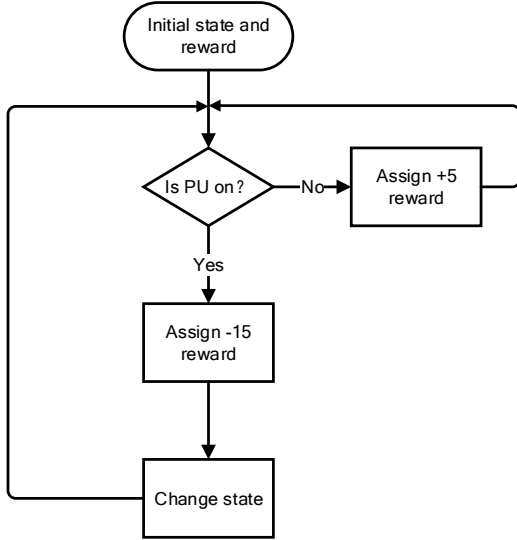


Figure 4: The Q-learning process on CBS model.

Once detected the primary user, a harsh punishment will be given. Otherwise, a positive reward will be assigned. Fig. 4 illustrates the proposed process, and Algorithm 1 describes our algorithm for implementing the Q-learning on CBS agent.

## 4 EXPERIMENTAL SIMULATION

### 4.1 Experimental Design

In this section, we describe preliminary results from applying our reinforcement learning based approach to the cognitive radio model. To detect the PUs correctly is the necessary prerequisite. The overall aim of our proposed learning based approach is to allow the CBS agents to decide on an optimal choice of spectrum so that (i) PUs are not affected, and (ii) CR users share the spectrum in a fair manner. These two rules are to simulate the public's behaviors in Urban Rail Transit Environment. That is, those bands that are frequently occupied by licensed users are rarely utilized because of open areas or relatively closed environment, and the public can opportunistically use band resources with a same probability.

Our novel CBS network simulator within the framework of high-speed rail has been designed to investigate the effect of the proposed reinforcement learning technique on the network operation. The implemented *ns-2* model is composed of several modifications to the physical, link and network layers in the form of stand-alone C++ modules. The PU Activity Block describes the activity of PUs based on the on-off model, including their transmission range, location, and spectrum band of use. The Channel Block contains a channel table with the background noise, capacity, and occupancy status. The Spectrum Sensing Block implements the energy-based sensing functionalities, and if a PU is detected, the Spectrum Management Block is notified. This, in turn causes the device to switch to the next available channel, and also alert the upper layers of the change of frequency. The Spectrum Sharing Block coordinates the distributed channel access, and calculates the interference at any given node due to the ongoing

---

### Algorithm 1 Pseudo code of Q-learning on CBS

---

**Main()**

Initialize state  $s_t$  and action  $a_t$  and their  $\vec{Q}$  value;

**repeat**

  Q-learning( $s_t, a_t, \vec{Q}$ )

**until** all episodes are traversed

**Q-with-Kanerva**( $s_t, a_t, \vec{Q}$ )

**repeat**

  Take action  $s_t$ , observe reward  $r_t$ , get next state  $s_{t+1}$

  Get  $Q(s_t a_t)$  from the  $Q$ -table;

**for** all actions  $a^*$  under new state  $s_{t+1}$  **do**

    Generate the state-action pair  $s_{t+1} a_{t+1}$  from state  $s_{t+1}$  and action  $a^*$

    Get  $Q(s_{t+1} a_{t+1})$  from the  $Q$ -table;

**end for**

$\delta = r + \gamma * \max Q(s_{t+1} a_{t+1}) - Q(s_t a_t)$

$\Delta \vec{Q} = \alpha * \delta$

$\vec{Q} = \vec{Q} + \Delta \vec{Q}$

$s_t = s_{t+1}$

**if** random probability  $\leq \epsilon$  **then**

**for** all actions  $a^*$  under current state  $s_t$  **do**

$a_t = \text{argmax}_a Q(s_t a_t)$

**end for**

**else**

$a_t = \text{random action}$

**end if**

**until**  $s_t$  is terminal

---

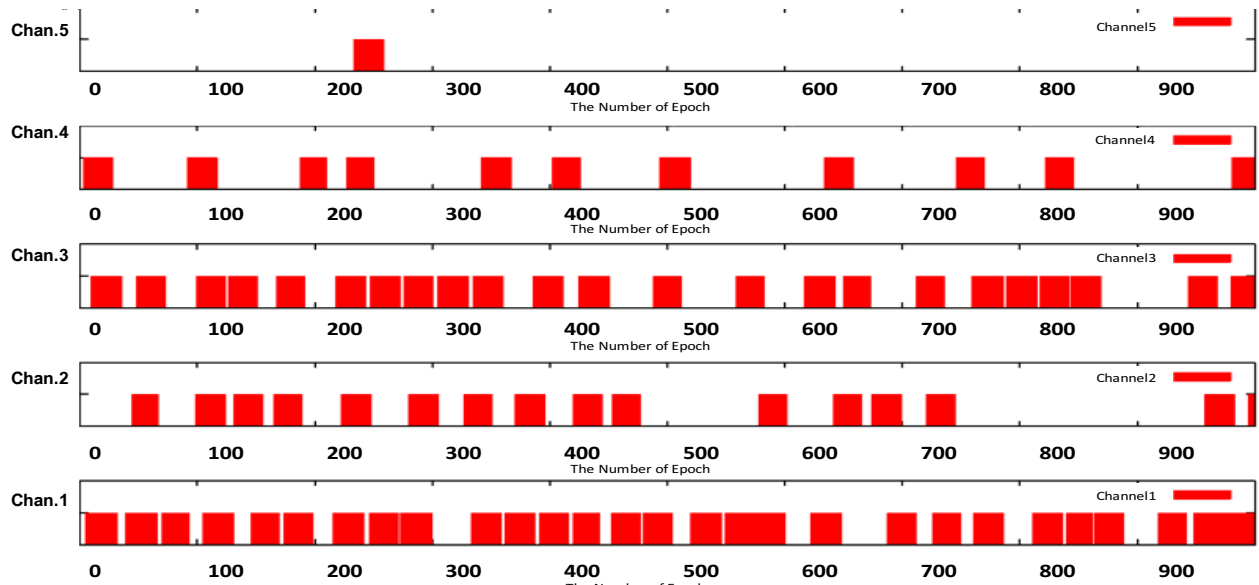
transmissions in the network. The Cross Layer Repository facilitates the information sharing between the different protocol stack layers.

We conduct our experiment in the following scenario: there are 2 trains which take on 21 passengers for each and 5 CBS agents aside the railway. The average speed of train is  $10m/s$ . We have 10 primary users in the range of each CBS. The activity of primary users is based on ON-OFF model and each primary user is assigned the spectrum randomly from 5 spectrums (small network) or 10 spectrums (large network). The CBS agent senses the spectrum holes per 0.1 second and assigns available spectrum to CR user agent. The simulation parameters are summarized in Table 1.

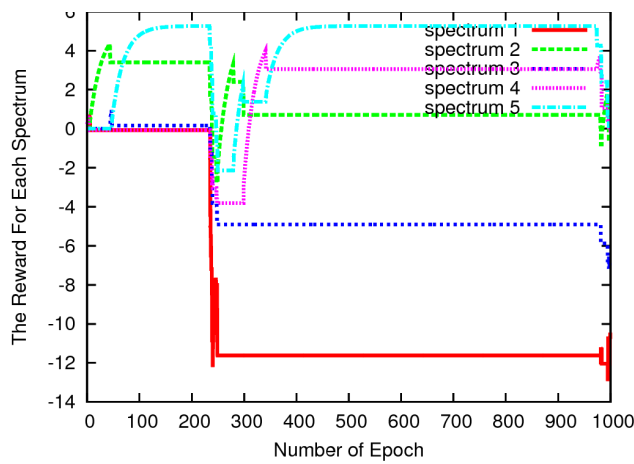
### 4.2 Experimental results

We compare the performance of our CBS with reinforcement learning (CBS-RL) scheme with the CBS with Round-Robin scheme (CBS-RR), which is a typical way in GSM-R system. The Round-robin (RR) scheme employs the principle that once a spectrum is not available, the agent switches to next channel in equal portions and in circular order, handling all switches without priority (also known as cyclic executive). This method is simple, easy to implement, and starvation-free. In our RL-based scheme, the exploration rate  $\epsilon$  is set to 0.2, which we found experimentally to give the best results. The initial learning rate  $\alpha$  is set to 0.8, and it is decreased by a scaling factor of 0.995 after each time slot.

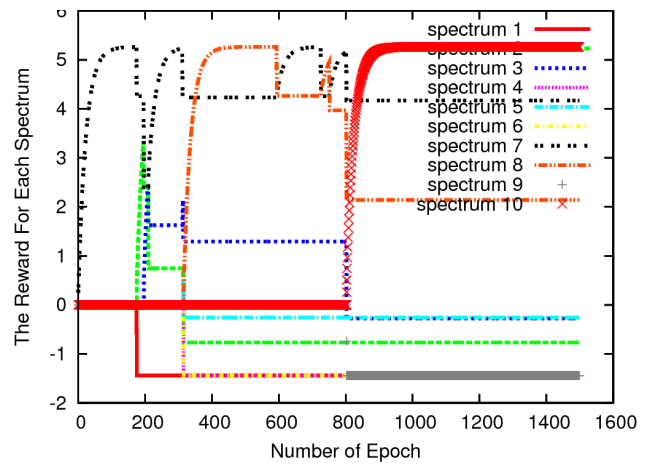
Figure 5(a) shows an example about the distribution of



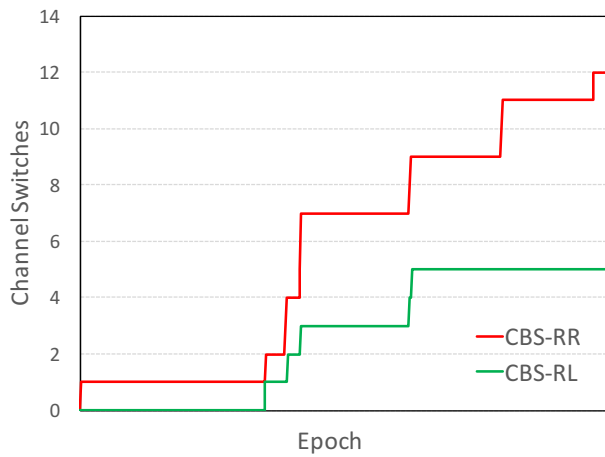
(a) An example about the distribution of spectrums occupancy on CBS with 5 spectrums.



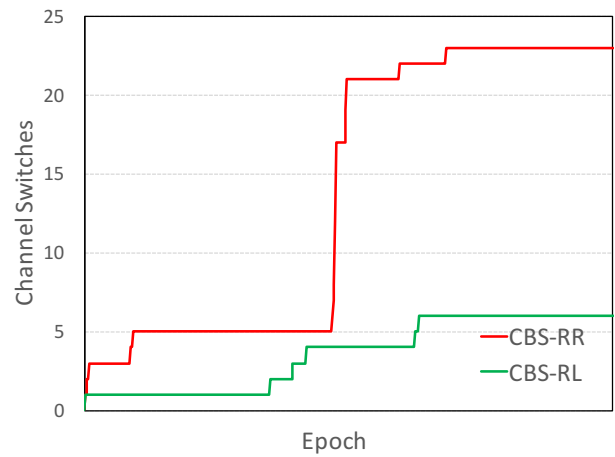
(b) Average rewards for 5 spectrum bands



(c) Average rewards for 10 spectrum bands



(d) Cumulative number of channel switching for 5 spectrum bands



(e) Cumulative number of channel switching for 10 spectrum bands

Figure 5: CBS simulations with RL and RR schemes.

Table 1: Simulation Parameters

Parameters	Values
Topology size	X:7000m Y:500m
Number of passengers	42
Number of primary users	50
Number of cognitive base station	5
Speed	10m/s
Number of spectrums	6
Bandwidth	2000000Hz
Simulation time	1000s

spectrums occupancy on the CBS with 5 spectrums. Spectrums occupancy on CBS follows the ON-OFF model: the ON mode is in the normal distribution with the parameter  $\mu = 25$ , and the OFF mode is in the exponential distribution with the parameter  $\beta$ . the value of which is randomly generated.

Figure 5(b) and 5(c) show the average rewards received by CBS agent across all spectrums using the CBS-RL scheme. The result in Figure 5(b) shows that after learning over 1000 epochs, Channel 5 receives the largest positive reward of approximately +5.5, while Channel 1, 2, 3 and 4 gets a reward of approximately -11.8, +0.7, -5.1 and +3.3. The results indicate that our approach pushes the CBS agents to gradually achieve higher positive rewards and choose more suitable spectrum for their transmission. The results also indicate that the reward tends to be suitable to the distribution of spectrums occupancy. A similar trend is observed in Figure 5(c), with Channel 10 receiving the highest average reward of approximately +5.2.

Figure 5(d) and 5(e) show the cumulative number of channel switching using CBS-RL and CBS-RR schemes. The result in Figure 5(d) shows the average number of channel switches for the small topology. We observe that after learning, the CBS-RL scheme tends to decrease number of channel switching to 5, while CBS-RR keeps the channel switches to approximately 12. For the large topology in Figure 5(e), the CBS-RL scheme reduces the channel switches to 6, while CBS-RR keeps the channel switches approximately 23. The results indicate that our proposed CBS-RL approach can keep the channel switches lower than the CBS-RR approach and converge to an optimal solution.

## 5 CONCLUSIONS

To address the issues of frequent channel switches and inefficient blind learning in high-speed rail, we propose a novel concept of Cognitive Base Station, which has the capability of forecasting spectrum holes and assigning spectrum to individuals. Our simulation results prove that after autonomous learning, the CBS-RL scheme can forecast spectrum holes. In this way, our proposed model can significantly improve the performance of vehicular communication, which can decrease cell-switching and unsuccessful transmission.

## References

[Ai *et al.*, 2014] Bo Ai, Xiang Cheng, Thomas Kurner, Zhangdui Zhong, Ke Guan, Ruisi He, Lei Xiong, David W

Matolak, David G Michelson, and Cesar Briso-Rodriguez. Challenges toward wireless communications for high-speed railway. *Intelligent Transportation Systems, IEEE Transactions on*, 15(5):2143–2158, 2014.

- [Alkayal and Saada, 2013] Faisal Alkayal and Johnny Bou Saada. Compact three phase inverter in silicon carbide technology for auxiliary converter used in railway applications. In *Power Electronics and Applications (EPE), 2013 15th European Conference on*, pages 1–10. IEEE, 2013.
- [Bkassiny *et al.*, 2013] Mario Bkassiny, Yang Li, and Sudharman K Jayaweera. A survey on machine-learning techniques in cognitive radios. *Communications Surveys & Tutorials, IEEE*, 15(3):1136–1159, 2013.
- [Chkirkbene and Hamdi, 2015] Zina Chkirkbene and Nouredine Hamdi. A survey on spectrum management in cognitive radio networks. *International Journal of Wireless and Mobile Computing*, 8(2):153–165, 2015.
- [Commission and others, 2003] Federal Communications Commission et al. Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies. *Et docket*, (03-108):05–57, 2003.
- [Dudoyer *et al.*, 2012] Stephen Dudoyer, Virginie Deniau, Ricardo Adriano, MN Ben Slimen, Jean Rioult, Benoît Meyniel, and Marion Berbineau. Study of the susceptibility of the gsm-r communications face to the electromagnetic interferences of the rail environment. *Electromagnetic Compatibility, IEEE Transactions on*, 54(3):667–676, 2012.
- [Dybala and Radkowski, 2013] Jacek Dybala and Stanislaw Radkowski. Reduction of doppler effect for the needs of wayside condition monitoring system of railway vehicles. *Mechanical Systems and Signal Processing*, 38(1):125–136, 2013.
- [Haykin, 2005] Simon Haykin. Cognitive radio: brain-empowered wireless communications. *Selected Areas in Communications, IEEE Journal on*, 23(2):201–220, 2005.
- [isheng Zhao *et al.*, 2013] isheng Zhao, Xi Li, Yi Li, and Hong Ji. Resource allocation for high-speed railway downlink mimo-ofdm system using quantum-behaved particle swarm optimization. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2343–2347. IEEE, 2013.
- [Jiang *et al.*, 2011] Tianzi Jiang, David Grace, and Paul D Mitchell. Efficient exploration in reinforcement learning-based cognitive radio spectrum sharing. *Communications, IET*, 5(10):1309–1317, 2011.
- [Kadam and Srivastava, 2012] Kaveri Kadam and Navin Srivastava. Application of machine learning (reinforcement learning) for routing in wireless sensor networks (wsns). In *Physics and Technology of Sensors (ISPTS), 2012 1st International Symposium on*, pages 349–352. IEEE, 2012.
- [Kim and Sung, 2014] Soyeon Kim and Wonjin Sung. Operational algorithm for wireless communication systems using cognitive radio. In *Communication, Networks and Satellite (COMNETSAT), 2014 IEEE International Conference on*, pages 29–33. IEEE, 2014.

- [Lee and Akyildiz, 2012] Won-Yeol Lee and Ian F Akyildiz. Spectrum-aware mobility management in cognitive radio cellular networks. *Mobile Computing, IEEE Transactions on*, 11(4):529–542, 2012.
- [Letaief and Zhang, 2009] Khaled Ben Letaief and Wei Zhang. Cooperative communications for cognitive radio networks. *Proceedings of the IEEE*, 97(5):878–893, 2009.
- [Li and Zhao, 2012] Jinxing Li and Youping Zhao. Radio environment map-based cognitive doppler spread compensation algorithms for high-speed rail broadband mobile communications. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):1–18, 2012.
- [Li et al., 2013] Ying Li, Lei Lei, Zhangdui Zhong, and Siyu Lin. Performance analysis for high-speed railway communication network using stochastic network calculus. In *Wireless, Mobile and Multimedia Networks (ICWMMN 2013), 5th IET International Conference on*, pages 100–105. IET, 2013.
- [Liu et al., 2011] Qiuyan Liu, Miao Wang, and Zhangdui Zhong. Statistics of capacity analysis in high speed railway communication systems. *Tamkang Journal of Science and Engineering*, 14(3):209–215, 2011.
- [Liu et al., 2012] Liu Liu, Cheng Tao, Jiahui Qiu, Houjin Chen, Li Yu, Weihui Dong, and Yao Yuan. Position-based modeling for wireless channel on high-speed railway under a viaduct at 2.35 ghz. *Selected Areas in Communications, IEEE Journal on*, 30(4):834–845, 2012.
- [Puterman, 1994] M. L. Puterman. Markov decision processes. In *Wiley*, 1994.
- [Sniady and Soler, 2012] Aleksander Sniady and Jose? Soler. An overview of gsm-r technology and its shortcomings. In *ITS Telecommunications (ITST), 2012 12th International Conference on*, pages 626–629. IEEE, 2012.
- [Sutton and Barto, 1998] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Bradford Books, 1998.
- [Tian et al., 2012] Lin Tian, Juan Li, Yi Huang, Jinglin Shi, and Jihua Zhou. Seamless dual-link handover scheme in broadband wireless communication systems for high-speed rail. *Selected Areas in Communications, IEEE Journal on*, 30(4):708–718, 2012.
- [Waltz and Fu, 1965] M. D. Waltz and K. S. Fu. A heuristic approach to reinforcement learning control systems. In *IEEE Transactions on Automatic Control*, 10:390-398., 1965.
- [Wu et al., 2010] Cheng Wu, Kaushik Chowdhury, Marco Di Felice, and Waleed Meleis. Spectrum management of cognitive radio using multi-agent reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry track*, pages 1705–1712. International Foundation for Autonomous Agents and Multiagent Systems, 2010.
- [Wu et al., 2015] Cheng Wu, Yiming Wang, Xiang Qiang, and Zhaoyang Zhang. Adaptive spectrum management of cognitive radio in intelligent transportation system. In *Applied Mechanics and Materials*, volume 743, pages 765–773. Trans Tech Publ, 2015.
- [Zhang et al., 2012] Jiayi Zhang, Zhenhui Tan, Xiaoxi Yua, Haibo Wang, and Linwen Zhang. Review of public broadband access systems for high-speed railways and key technologies. *Journal of the China Railway Society*, 34(1):46–53, 2012.
- [Zhou and Ai, 2014] Yuzhe Zhou and Bo Ai. Quality of service improvement for high-speed railway communication. *Communications, China*, 11(11):156–167, 2014.
- [Zhu et al., 2013] Xiangqian Zhu, Shanzhi Chen, Haijing Hu, Xin Su, and Yan Shi. Tdd-based mobile communication solutions for high-speed railway scenarios. *Wireless Communications, IEEE*, 20(6):22–29, 2013.

# Iterative committee elections for collective decision-making in a ride-sharing application\*

Sophie L. Dennisen, Jörg P. Müller

Institut für Informatik

Technische Universität Clausthal

38678 Clausthal-Zellerfeld, Germany

sophie.dennisen@tu-clausthal.de, joerg.mueller@tu-clausthal.de

## Abstract

We investigate the use of voting methods for multi-agent decision-making in cooperative traffic applications. We consider a ride-sharing problem in which passengers use committee elections to collectively decide on sets of points of interest to visit. In this paper, we propose an iterative voting protocol for the committee voting rules Minisum Approval and Minimax Approval. Using this protocol, voters can leave a group if their dissatisfaction with the election result exceeds a threshold value. We evaluate the rules for the ride-sharing problem using an agent-based simulation. Our results indicate that for initial group sizes around 20, both rules tend to require equal numbers of iterations for dissatisfaction threshold values around zero. We showed that Minisum Approval needs distinctly fewer iterations than Minimax Approval for values between zero and half the size of the candidate set. In some cases, for values around half the size of the candidate set, Minimax needs fewer iterations. For higher values, both rules need tendentially the same number of iterations. When aiming at minimising the number of iterations, we recommend to apply Minisum Approval for threshold values between zero and half the size of the candidate set. For higher values, we recommend to use Minimax Approval.

## 1 Introduction

There are diverse approaches for vehicle routing and ride-sharing problems. In vehicle routing, the goal is to design a low-cost route so that each node is visited by exactly one vehicle. In ride-sharing, it is usually assumed that each passenger has exactly one desired destination.

Here, we consider another approach for the situation that the passengers of the shared vehicles submit their preferences

---

\*This research has been supported by the German Research Foundation (DFG) through the Research Training Group SocialCars: Cooperative (De-)centralized Traffic Management (GRK 1931). The focus of the SocialCars Research Training Group is on significantly improving the city's future road traffic, through cooperative approaches. This support is gratefully acknowledged.

on all possible destinations. In this context, the question arises how to agree on a common subset of destinations to visit.

We propose to use committee voting rules to design an initial solution and to allow dissatisfied passengers to leave the group and to apply iterative winner determination until all remaining passengers are satisfied with the selected subset of destinations.

The trend in transportation systems goes towards automation, i.e. non-automated vehicles will be replaced by autonomous vehicles over time, increasing safety and decreasing environment pollution.

Our model is motivated by a future scenario in which traditional urban traffic is replaced by autonomous vehicles (AVs), where the city provides AVs for visitors of the city. Once a visitor boards an AV, s/he transmits his/her preferences regarding the possible destinations etc. to the AV, e.g. via smartphone app.

From the perspective of the urban traffic management, pooling the visitors into as large groups as possible is desirable as it reduces energy consumption and increases safety. Thus, we assume that the urban traffic management encourages the visitors to group together in autonomous vehicles as soon as they enter the intraurban area.

We assume that the visitors are grouped together in autonomous shared vehicles (ASVs) provided by the city at predefined boarding points, as proposed by Dennisen and Müller [2015].

We focus on visitors who submit their preferences regarding all possible destinations in the respective urban area.

This approach raises several questions.

1. How can the passengers of a shared vehicle agree on a common route?
2. How should one deal with passengers who are not satisfied with the selected route?

### 1.1 Outline

The remainder of the paper is structured as follows. In Section 2, the state-of-the-art is depicted, and in Section 3 the research gap is described. Section 4 gives an overview on the definitions, our solution approach and the voting architecture used for the simulations. Section 5 describes an example scenario. Section 6 includes the experimental settings, the results and the discussion, and Section 7 concludes the paper.

## 2 State-of-the-art

There is a range of works on the areas Ride-Sharing, Vehicle Routing and Transportation Systems. Here, we discuss a selection of those works.

### 2.1 The Vehicle Routing Problem

According to the review paper by Laporte [1992], the Vehicle Routing Problem (VRP) is defined as follows.

Input:  $G = (V, A)$  graph where  $V = \{1, \dots, n\}$  is a set of vertices representing cities with the depot located at vertex 1, and  $A$  is the set of arcs. With every arc  $(i, j), i \neq j$ , is associated a non-negative distance matrix  $C = (c_{ij})$ . In some contexts,  $c_{ij}$  can be interpreted as travel cost or travel time.

The goal is to design a set of least-cost vehicle routes so that

- each city except for the depot is visited by exactly one vehicle
- all vehicle routes start and end at the depot
- some side constraints are satisfied

### 2.2 Dynamic ride-sharing

In the review article by Agatz et al. [2012], the authors refer by dynamic ride-sharing to a system where an automated system made available by a ride-sharer provider matches up drivers and riders on short notice.

Most studies on ride-sharing consider one of the following specific objectives when determining ride-sharing matches.

- Minimise system-wide vehicle-miles
- Minimise the system-wide travel time
- Maximise the number of participants

In ride-sharing, it is assumed that each rider wants to travel from his/her origin to his/her destination.

### 2.3 Sharing Rides with Friends

One aspect which has been considered regarding ride-sharing is the constraints of the social network connecting the commuters. Bistaffa et al. [2014] consider the Social Ridesharing Problem, where a set of commuters, connected through a social network, arrange one-time rides at short notice. They focus on the associated optimisation problem of forming the cars to minimise the travel cost of the overall system, modelling the problem as a graph constrained coalition formation (GCCF) problem, where the set of feasible coalitions is restricted by a graph, i.e. the social network.

They assume real-time ride-sharing, arranging one-time rides with private cars and focus on providing an approach that, given the desired starting points and destinations of a community of commuters, can share cars to lower associated transportation costs, i.e. travel time and fuel, while considering the constraints imposed by the social network that connects such commuters.

### 2.4 Rural Flexible Transport Systems

Velaga et al. [2012] developed a passenger-centric agent-based flexible transport systems (FTS) platform using argumentation theory. Each passenger provides the following information:

- origin
- destination
- travel time window
- order of preference among: travel cost, number of changes and journey length.

The first three items are requirements, i.e. conditions that must be fulfilled for a journey to be a candidate.

The brokering subsystem gathers all the plausible journeys and composes a certain number of allocations, i.e. an assignment of passengers to sequences of vehicles.

The final step is to choose the globally preferred allocation from this set. For this, Velaga et al. [2012] use a variation of the Borda voting rule; each of the passenger agents votes by assigning a rating to each candidate allocation, and the allocation with the best rating wins. Velaga et al. [2012] do not consider committee elections.

## 3 Research Gap

According to Laporte [1992], in the VRP, the objective function is usually dependent on travel time or travel cost, depending on the edges. In our approach, we focus on agreeing on a common subset of POIs, disregarding the routing problem in the first phase.

In the RFTS model by Velaga et al. [2012], the candidates are a number of plausible assignments of passengers to journeys, not the points of interest (POIs), i.e. the construction of the journeys is independent from the voting process. In our approach, the voting process is necessary for the construction of the routes.

Bistaffa et al. [2014] do not consider the preferences of passengers over several destinations, but assume that each passenger has exactly one desired destination and generate coalitions with minimal cost routes.

None of these approaches focuses on how to agree on a common subset of destinations to visit based on the passengers' preferences over all possible destinations.

In this paper we propose that the passengers of a shared vehicle agree on a common subset of destinations to visit via committee election and to apply iterative winner determination until all remaining passengers are satisfied with the selected subset of destinations, i.e. in each iteration, the most dissatisfied passenger leaves the group and the remaining votes are re-evaluated.

From the operative perspective, a small number of iterations is desirable: On the one hand, the fewer iterations are conducted, the more passengers are left and the better the capacity of the shared vehicle is utilised. On the other hand, in each iteration communication between the visitors and the chair is required, i.e. minimising the number of iterations reduces the communication expense.

Thus, in this paper, we focus on the following research questions.



Assuming that visitors group together dependent on their time of arrival (i.e. in a random fashion) and only change to other shared vehicles at the starting point(s), decide on a common route via committee election:

1. How do different committee voting rules under an iterative protocol compare regarding the number of iterations?
2. Given a committee voting rule, how many iterations will tendentially be conducted until the committee election is terminated?

## 4 Definitions and Methods

### 4.1 Definitions

#### Election

Here, we follow the definition in Rothe et al. [2012]. An election is defined as a tuple  $(C, V)$  where  $C = \{c_1, \dots, c_m\}$  is the set of candidates and  $V = \{v_1, \dots, v_n\}$  is the list of votes over  $C$ . Each voter is represented via his/her vote which specifies his/her preferences over the candidates in  $C$ . Which form the votes take depends on the voting rule.

#### Voting Rule

Following Rothe et al. [2012], given a candidate set  $C$ , a voting rule is a social-choice correspondence  $f : \{(C, V) | (C, V) \text{ is a valid election}\} \rightarrow \mathcal{P}(C)$  which assigns to each valid election  $(C, V)$  a set of winning candidates. To determine a unique winner, it can be necessary to apply a tie-breaking rule.

#### Committee election

Analogously to the above definition, a committee election can be defined as a tuple  $(C, V, k)$  with non-negative integer  $k \leq m = |C|$ .

#### Committee voting rule

Analogously to voting rules, one can define committee voting rules. For a given candidate set  $C$  and non-negative integer  $k \leq m = |C|$ , a committee voting rule is a function which assigns to each valid committee election  $(C, V, k)$  a set of winning committees. To determine a unique winning committee, it can be necessary to apply a tie-breaking rule.

Following the definition in Baumeister et al. [2015], a committee voting rule is a mapping  $g : \{(C, V, k) | (C, V, k) \text{ is a valid committee election}\} \rightarrow F_k(C)$  with  $F_k(C)$  the set of all committees from  $C$  of size  $k$ .

#### Voting protocol

Here, we use the notion of voting protocols in the sense that a voting protocol defines the communication processes between the agents involved in the election.

#### Voting mechanism

In the context of this paper, a voting mechanism consists of a voting protocol and a voting rule or committee voting rule.

### Committee voting rules for scenario

Both committee voting rules considered for the ride-sharing scenario assume Approval vectors, i.e. votes from  $\{0, 1\}^n$ , where a “0” at  $i$ -th position stands for disapproval and a “1” at  $i$ -th position for approval of the  $i$ -th candidate.

Following Brams et al. [2007a,b], the dissatisfaction of a voter  $v$  with a selected committee  $com$  is measured via the Hamming distance  $HD(v, com)$ .

**Minisum Approval** Minisum Approval selects a committee for which the sum of the Hamming distances between all votes and the committee is minimal. This corresponds to a utilitarian approach.

**Minimax Approval** Minimax Approval as proposed by Brams et al. [2007a,b]; Kilgour et al. [2006] selects a committee for which the maximum Hamming distance between a vote and the committee is minimal. This corresponds to an egalitarian approach.

### 4.2 Approach

Under the assumption that visitors of a city are encouraged to conduct round-trips in shared vehicles provided by the city, there is the question how the passengers of a shared vehicle agree on a common route. We assume that the vehicles can rank in size from taxi size to bus size.

We propose to use committee elections to agree on an initial solution. When considering the initial solution, it is possible that some passengers are dissatisfied. Such dissatisfied passengers can be allowed to leave the shared vehicle at the start point(s) and change to other vehicles. This leads to iterative winner determination. We propose to use an iterative voting protocol as depicted in Figure 1. The figure depicts the steps for the non-iterative protocol as described in Dennisen and Müller [2015] in solid lines and the additional steps for the iterative protocol in dashed lines.

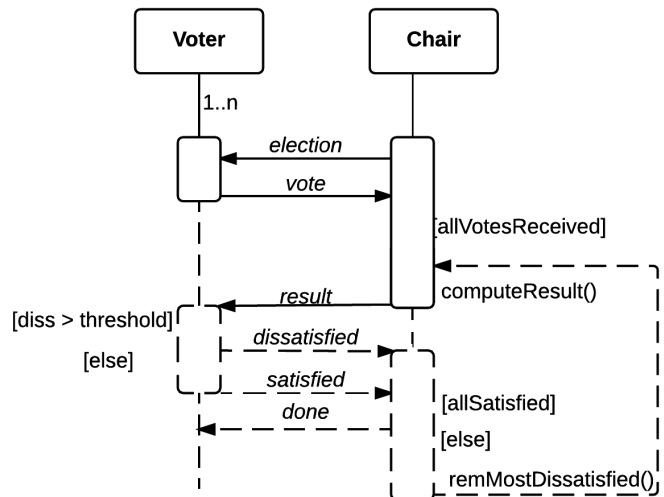


Figure 1: Centralised iterative protocol

In the non-iterative protocol, the chair starts the election by sending an `election` message to all voters and the voters respond by submitting their votes to the chair. As soon as the chair has received all votes, s/he computes the result of the election according to the given voting rule and sends the result to all voters.

In the iterative protocol, after receiving the result, the voters check via a dissatisfaction threshold if they are dissatisfied with the result. Based on their (unaltered) votes, they submit a `satisfied` or a `dissatisfied` message to the chair. If there is at least one dissatisfied voter, the chair removes the most dissatisfied voter and computes the result for the remaining voters. Otherwise, the election is terminated.

### 4.3 Voting Architecture

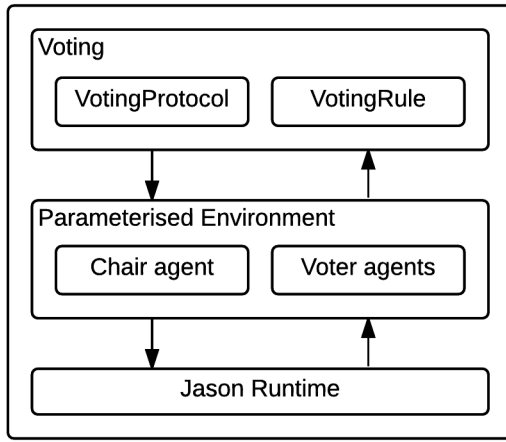


Figure 2: Voting Architecture

We decided to evaluate the behaviour of different voting mechanisms via multiagent-based simulation. This allows for testing diverse input combinations and later extension to dynamic traffic simulations.

As a voting architecture, we developed an adaptation of J-MADeM, an agent-based architecture implemented in Jason by Grimaldo et al. [2010]. Jason is an interpreter developed by Bordini et al. [2005], written in Java for an extended version of AgentSpeak, a logic-based agent-oriented programming language that is suitable for the implementation of reactive planning systems according to the Belief-Desire-Intention (BDI) architecture.

Currently, in our voting architecture, two voting protocols and two committee voting rules are implemented. The architecture allows for extension to further voting protocols and voting rules such as decentralised non-iterative protocols, decentralised iterative protocols, the voting rules Condorcet (based on pairwise comparisons), Borda and the committee voting rule Minisum-Ranksum (based on positional scores) as described in Baumeister and Dennisen [2015].

The architecture is structured as depicted in Figure 2. The Jason Runtime handles the agent cycles and the communication between the agents. The chair and voter agents are located in a parameterised environment. They receive the simulation parameters in form of initial beliefs and call the voting

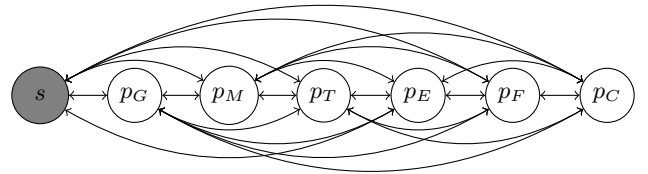
protocol/rule; this is realised via customisation of the Agent Architecture class.

Which modules in the Agent Architecture class are used by the respective agent depends on the role of the agent. The chair agent uses the election launcher module responsible for starting the election, the votes manager module which collects the votes and the winner determination module which computes the result of the votes according to the voting rule. The voter agents use the voting module responsible for submitting votes.

## 5 Example scenario

Consider as example the following scenario. Four visitors  $t_1, t_2, t_3, t_4$  who want to visit Manhattan, NY form together to a group at a predefined point  $s$  in Northern Manhattan. Each of them submits his/her preferences regarding all possible POIs. Due to time constraints on the operative side, their common route can only cover exactly three POIs. For simplicity, we assume that there are six possible POIs: Guggenheim Museum ( $p_G$ ), MoMA ( $p_M$ ), Times Square ( $p_T$ ), Empire State Building ( $p_E$ ), Flatiron Building ( $p_F$ ) and Chinatown ( $p_C$ ). The corresponding graph is depicted in Figure 3.

Figure 3: Graph for illustrating example



In the simplest model, the passengers of a shared vehicle submit their preferences in form of approval votes, i.e. they indicate approval of a candidate by assigning a “1” to it and disapproval of a candidate by assigning a “0” to it.

In the illustrating scenario, we assume that the visitors submit their preferences as approval vectors. In this case, one can apply the committee voting rule Minisum Approval. The votes and the scores are depicted in Table 1.

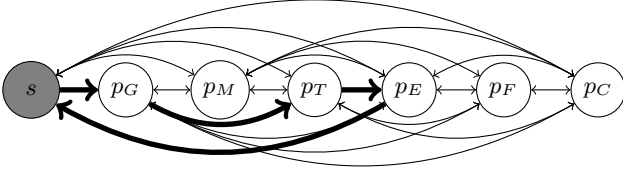
Table 1: Approval scores for illustrating example

POI	$p_G$	$p_M$	$p_T$	$p_E$	$p_F$	$p_C$
$t_1$	1	0	1	1	0	1
$t_2$	1	0	1	1	0	0
$t_3$	1	0	1	0	1	0
$t_4$	0	0	0	1	1	1
Score	3	0	3	3	2	2

For committee size  $k = 3$ , the winning committee is  $K = \{p_G, p_T, p_E\}$ . Assuming that the shared vehicle drives from North to South until it heads back to its starting point, the shared vehicle would take the route  $s \rightarrow p_G \rightarrow p_T \rightarrow p_E \rightarrow s$ , as depicted in Figure 4.

For approval vectors, the straightforward approach to measure the dissatisfaction with an elected committee is to con-

Figure 4: Resulting route



consider the Hamming distance between the respective vote and the elected committee.

The Hamming distances between the votes and the elected committee  $p_G, p_T, p_E$  are depicted in Table 2

Table 2: Hamming distances for illustrating example

	1	0	1	1	0	0	Hamming distance
$t_1$	1	0	1	1	0	1	1
$t_2$	1	0	1	1	0	0	0
$t_3$	1	0	1	0	1	0	2
$t_4$	0	0	0	1	1	1	4

Assuming dissatisfaction threshold  $t = 2$ ,  $t_4$  leaves the group and looks for another shared vehicle.

Table 3: Approval scores for illustrating example, second iteration

POI	$p_G$	$p_M$	$p_T$	$p_E$	$p_F$	$p_C$
$t_1$	1	0	1	1	0	1
$t_2$	1	0	1	1	0	0
$t_3$	1	0	1	0	1	0
Score	<b>4</b>	0	<b>4</b>	<b>3</b>	1	1

In this case, the removal of the dissatisfied voter does not alter the outcome of the committee election, so the route stays the same.

## 6 Evaluation

We investigated the impact of the dissatisfaction threshold for the visitors  $t$ , the number of POIs to be visited  $k$  and the number of offered POIs  $m$  on the number of iterations needed by the voting rules Minisum and Minimax Approval.

### 6.1 Experimental Settings

Our simulations were conducted with the following technical settings.

- Jason-1.4.2
- Java 1.8.0.65
- Windows 8.1
- HDF5 for storing input and output data
- R x64 3.2.3 for evaluation

As configuration parameters for the simulation, we have

- the number  $n = 20$  of voters (visitors):  $n = 20$  is oriented towards bus sizes

- the number  $m$  of candidates (POIs)
- the size  $k$  of the committee to be elected
- the dissatisfaction threshold  $t$
- the committee voting rules
- the voting protocol(s)

For each run, the votes are generated as follows: For each position in each vote, a “1” or a “0” is selected with equal probability, resulting in homogenous electorates.

In each run, the number of necessary iterations is saved.

We measured and compared the number of iterations under the iterative protocol for Minimax Approval and Minimax Approval for several input combinations  $(n, m, k, t)$ .

In each simulation, we conduct 100 runs and measure the median of the numbers of iterations for Minisum and Minimax as well as the median of differences  $iteration\_minisum - iterations\_minimax$ . In our simulations, we consider three different settings.

1. Vary the values for dissatisfaction threshold  $t$
2. Vary the values for committee size  $k$
3. Vary the values for number of candidates  $m$

### 6.2 Results

#### Exploring the impact of dissatisfaction threshold $t$ on iteration numbers

In our first setting, we considered all possible values of dissatisfaction threshold  $t$  for number of voters  $n = 20$ , number of candidates  $m = 10$ , committee size  $k = 5$ , i.e.  $0 \leq t \leq m = 10$ . The results are depicted in Figure 5. The figure shows that the number of iterations decreases for both voting rules with increasing dissatisfaction threshold  $t$ . For values of  $t$  around 0, it is impossible to satisfy the voters, so that both voting rules need 20 iterations, creating empty groups. For smaller values of  $t$  above zero, i.e. for hard-to-please voters, Minisum needs tendentially fewer iterations than Minimax. For values of  $t$  above  $m/2$  and near  $m/2$ , i.e. for more tolerant voters, Minimax needs fewer iterations than Minisum. For higher values, both rules need zero iterations.

Consider the input combination  $n = 20, m = 10, k = 5, t = 4$ . The boxplot for the differences between Minisum and Minimax is depicted in Figure 6. The median of the differences is  $-3$ , i.e. Minisum needs tendentially 3 iterations fewer than Minimax for dissatisfaction value  $t = 3$ .

For  $t = 4$ , the Wilcoxon rank sum test with continuity correction yields  $W = 988.5$  and p-value  $< 2.2e - 16 < 0.05$ , i.e. we can reject the null hypothesis that there is no statistical difference between the distributions.

Consider the input combination  $n = 20, m = 10, k = 5, t = 6$ . The boxplot for the differences between Minisum and Minimax is depicted in Figure 7. The median of the differences is 1, i.e. Minimax needs tendentially one iterations fewer than Minisum.

For  $t = 6$ , the Wilcoxon rank sum test with continuity correction yields  $W = 8734.5$  and p-value  $< 2.2e - 16 < 0.05$ , i.e. we can reject the null hypothesis that there is no statistical difference between the distributions.

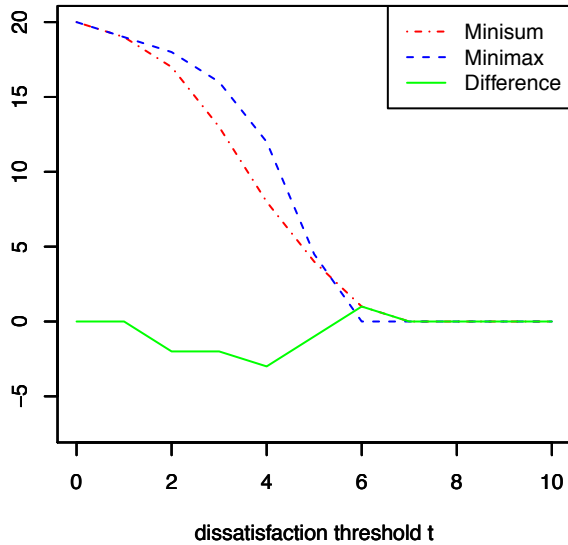


Figure 5: Median of iteration numbers for Minisum and Minimax and median of differences (Minisum-Minimax) for  $n = 20, m = 10, k = 5$  against dissatisfaction threshold  $t$ .

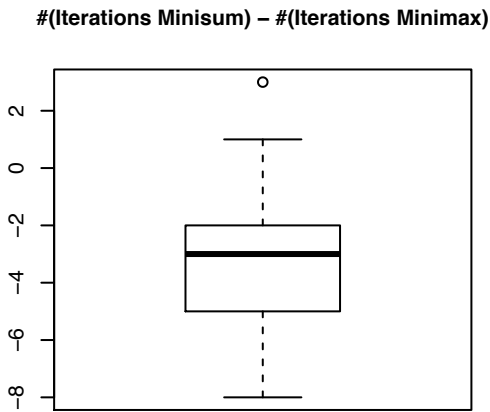


Figure 6: Boxplot of differences for  $n = 20, m = 10, k = 5, t = 4$

### Exploring the impact of committee size $k$ on iteration numbers

In the second setting, we fixed the number of candidates  $m = 10$  the number of voters  $n = 20$ , varied committee size  $k$ , i.e. the number of effectively visited POIs and measured the median of differences for dissatisfaction threshold values  $t = 5$  and  $t = 6$ .

The results for  $t = 5$  are depicted in Table 4 and Figure 8, the results for  $t = 6$  in Table 5 and Figure 9.

Table 4 shows that both voting rules need fewer iterations for medium values of  $k$ , i.e. the closer  $k$  is to  $m/2$ , the fewer iterations are needed. For  $t = 5$ , the median of the differences between Minisum and Minimax is  $-1$ , i.e. Minisum needs tendentially one iteration fewer than Minimax.

Table 5 and Figure 9 show a similar trend. Both voting

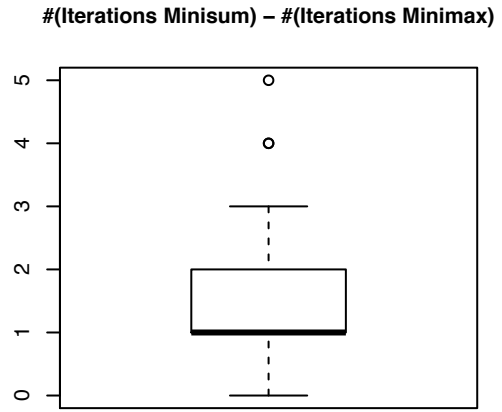


Figure 7: Boxplot of differences for  $n = 20, m = 10, k = 5, t = 6$

rules need fewer iterations for medium values of  $k$ . Here, Minimax needs one iteration fewer than Minisum for medium values of  $k$ . For other values of  $k$ , both rules need tendentially the same number of iterations.

Table 4: Medians for  $m = 10, n = 20, t = 5$  and varying  $k$

$k$	Iterations Minisum	Iterations Minimax	Difference
1	6	7	-1
2	5	6	-1
3	5	5	-1
4	5	5	-1
5	4	5	-1
6	4	5	-1
7	5	6	-1
8	5	6	-1
9	7	9	-1

Table 5: Medians for  $m = 10, n = 20, t = 6$  and varying  $k$

$k$	Iterations Minisum	Iterations Minimax	Difference
1	2	2	0
2	2	1	0
3	2	0	1
4	1	0	1
5	1	0	1
6	1	0	1
7	2	0	1
8	2	1	1
9	2	3	0

Consider the input combination  $n = 20, m = 10, k = 5, t = 6$ . The boxplot for the differences between Minisum and Minimax is depicted in Figure 10. The median of the differences is 1, i.e. Minimax needs tendentially one iterations fewer than Minisum.

The Wilcoxon rank sum test with continuity correction yields  $W = 8667.5$  and  $p\text{-value} < 2.2e - 16 < 0.05$ , i.e. we can reject the null hypothesis that there is no statistical difference between the distributions.

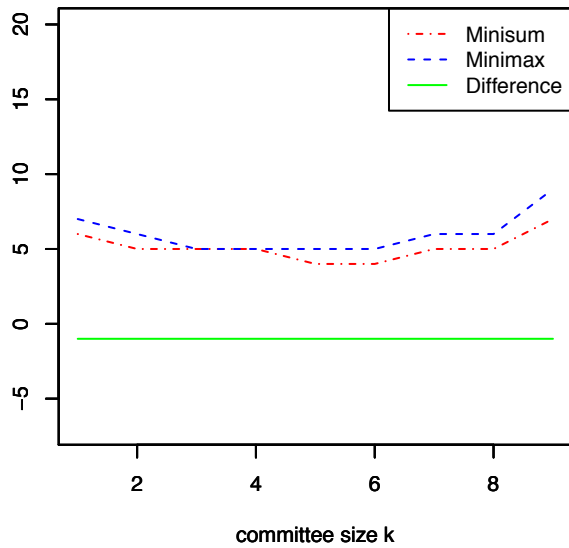


Figure 8: Median of iteration numbers for Minisum and Minimax and median of differences (Minisum-Minimax) for  $n = 20, m = 10, t = 5$  against committee size  $k$

### Exploring the impact of number of candidates $m$ on iteration numbers

In the third setting, we fixed  $n = 20, k = 5, t = \lfloor m/2 \rfloor, \lceil m/2 \rceil$  and measured the median of differences for different numbers of offered POIs  $m = 10, 15, 20$ . The results are depicted in Table 6.

Table 6: Median of differences for  $m = 10, 15, 20, n = 20, k = 5, t = \lfloor m/2 \rfloor, \lceil m/2 \rceil$

	Minisum	Minimax	Difference
$m = 10, k = 5, t = 5$	4	4.5	-1
$m = 15, k = 5, t = 7$	6	6	0
$m = 15, k = 5, t = 8$	3	1	2
$m = 20, k = 5, t = 10$	5	4	1

Table 6 shows no clear relation between the number of the candidates and the iteration numbers for Minisum and Minimax Approval. For  $m = 15, k = 5$ , one can again see the influence of  $t$  on the numbers of iterations.

### 6.3 Discussion

Assuming shared vehicles with a capacity of 20 and numbers of offered POIs up to 20, our results indicate that a favourable constellation from operative perspective would be to offer two times as many POIs as can be visited by a shared vehicle - the voting rules need fewer iterations if the number of effectively visited POIs lies around half the number of offered POIs.

The dissatisfaction threshold has an considerable impact: For threshold values around 0, Minisum and Minimax tend to need the same number of iterations. For higher values below  $m/2$ , i.e. if the voters are hard to please, Minisum needs distinctly fewer iterations than Minimax.

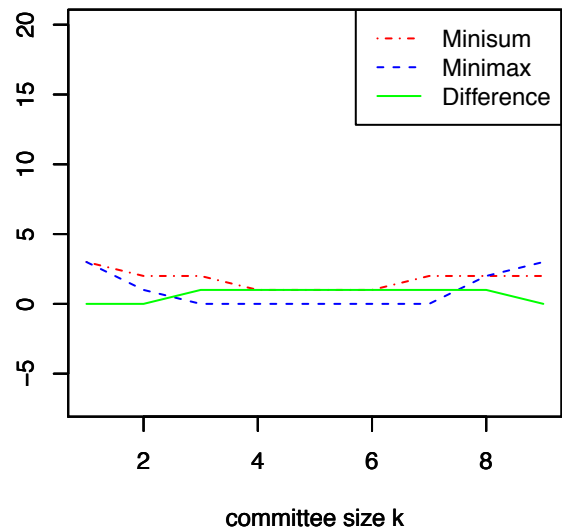


Figure 9: Median of iteration numbers for Minisum and Minimax and median of differences (Minisum-Minimax) for  $n = 20, m = 10, t = 6$  against committee size  $k$

#(Iterations Minisum) - #(Iterations Minimax)

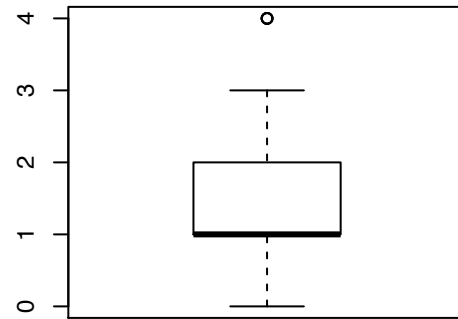


Figure 10: Boxplot of differences for  $n = 20, m = 10, k = 5, t = 6$

If  $k$  lies around  $m/2$ , Minimax needs tendentially fewer iterations than Minisum for values of  $t$  close to  $m/2$ , i.e. for more tolerant voters.

For higher values of  $t$ , the difference decreases until both committee voting rules tend to need the same number of iterations.

In practice, there are several motives for minimising the number of iterations.

- Capacity utilisation: The fewer iterations are conducted, the more visitors remain in the shared vehicle
- Communication expense: In each iteration, the visitors have to communicate with the chair in order to indicate if they are satisfied or dissatisfied.

In order to minimise the number of iterations, we recommend to apply Minisum Approval for the case that it is ex-

pected that the visitors are hard-to-please. If you expect that the visitors are more tolerant, we recommend to use Minimax Approval.

So far, Computational Social Choice methods have been largely subject to theoretical analysis. There are hardly any attempts to use them in the engineering of socio-technical multiagent systems such as traffic modeling and management. The ride-sharing scenario is relatively simple but we believe it is yet suitable as an experimental scenario due to its relative generality and the relevance (and hardness) of the underlying optimisation problems. The concept is applicable for non-autonomous driving as well: In the case of non-autonomous driving, one could equate the chair agent with the owner/driver.

Our next step will be to reproduce our results for further input combinations  $(n, m, k, t)$ . Note that we conducted investigations for relative small numbers of available POIs. For larger numbers of POIs, we aim to compare the properties of Minisum Approval and Minimax Approximation algorithms.

In the setting considered in this paper, we used a fixed dissatisfaction threshold to determine the dissatisfaction of the visitors. In a dynamic scenario, it would make sense to let the visitors decide individually if they are satisfied or dissatisfied. To simulate this, one would need a stochastic model to determine the dissatisfaction thresholds.

Furthermore, we will consider the situation that the voters cannot only leave their initially assigned groups but change to another groups.

Also, a challenge for future research is to study the runtime performance of the voting mechanisms taking the time requirements of collective decision situations in real traffic into account.

## 7 Conclusion

In this paper, we investigated the usability of methods known from the area of computational social choice in future cooperative traffic environments consisting of automated or human-operated vehicles, able to communicate with each other, e.g. using Vehicle-to-X communication technologies. In particular, we considered a ride-sharing scenario where visitors of a city share vehicles with seating capacities similar to buses to visit points of interest.

We proposed an iterative voting protocol based on the well-known Minisum Approval and Minimax Approval committee voting rules, allowing dissatisfied travellers to leave a group and join a different one. Using an agent-based simulation, we compared iterative Minisum Approval and Minimax Approval with respect to their convergence properties.

The main result is that iterative Minisum Approval outperforms Minimax approval in this respect for threshold values higher than 0 and lower than  $m/2$ . If  $k$  lies around  $m/2$ , there is a slight advantage to Minimax Approval for values of  $t$  close to  $m/2$ .

## References

Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review.

*European Journal of Operational Research*, 223(2):295–303, 2012.

Dorothea Baumeister and Sophie Dennisen. Voter Dissatisfaction in Committee Elections. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1707–1708. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

Dorothea Baumeister, Sophie Dennisen, and Lisa Rey. Winner Determination and Manipulation in Minisum and Minimax Committee Elections. In *Algorithmic Decision Theory*, pages 469–485. Springer, 2015.

Filippo Bistaffa, Alessandro Farinelli, and Sarvapali D Ramchurn. Sharing rides with friends: a coalition formation algorithm for ridesharing. 2014.

Rafael H Bordini, Jomi F Hübner, and Renata Vieira. Jason and the golden fleece of agent-oriented programming. In *Multi-agent programming*, pages 3–37. Springer, 2005.

Steven J Brams, D Marc Kilgour, and M Remzi Sanver. A Minimax Procedure for Electing Committees. *Public Choice*, 132(3-4):401–420, 2007a.

Steven J Brams, D Marc Kilgour, and M Remzi Sanver. A Minimax Procedure for Negotiating Multilateral Treaties. In *Diplomacy games*, pages 265–282. Springer, 2007b.

Sophie L Dennisen and Jörg P Müller. Agent-Based Voting Architecture for Traffic Applications. In *Multiagent System Technologies*, pages 200–217. Springer, 2015.

Francisco Grimaldo, Miguel Lozano, Fernando Barber, and Alejandro Guerra-Hernández. J-MADeM v1. 1: A full-fledge AgentSpeak (L) multimodal social decision library in Jason. In *The 8th European Workshop on Multi-Agent Systems (EUMAS 2010)*, 2010.

D Marc Kilgour, Steven J Brams, and M Remzi Sanver. How to Elect a Representative Committee using Approval Balloting. In *Mathematics and Democracy*, pages 83–95. Springer, 2006.

Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

Jörg Rothe, Dorothea Baumeister, Claudia Lindner, and Irene Rothe. *Einführung in Computational Social Choice: Individuelle Strategien und kollektive Entscheidungen beim Spielen, Wählen und Teilen*. Springer-Verlag, 2012.

Nagendra R Velaga, Nicolás D Rotstein, Nir Oren, John D Nelson, Timothy J Norman, and Steve Wright. Development of an integrated flexible transport systems platform for rural areas using argumentation theory. *Research in Transportation Business & Management*, 3:62–70, 2012.



# Model-Driven Engineering of Simulations for Smart Roads

Alberto Fernández-Isabel, Rubén Fuentes-Fernández

Department of Software Engineering and Artificial Intelligence

Universidad Complutense de Madrid

Madrid, Spain

afernandezisabel@ucm.es, ruben@fdi.ucm.es

## Abstract

Smart Roads (SRs) are systems that provide traffic-related services, based on a combination of sensor and actuator networks deployed in roads, vehicles, and surrounding elements. They are complex distributed systems that involve multiple heterogeneous components and technologies. This makes their development a challenging and costly process. Simulations are a key tool to deal with these issues, as they allow developing and testing in fully controlled environments with simplified software components. Nevertheless, they still need to consider multiple perspectives (e.g. experts and designers), which frequently cause problems to understand and validate them. Model-Driven Engineering of simulations appears as a solution. It uses models to represent explicitly these perspectives, and transformations to link them and generate new artifacts (including code). This paper presents a framework to develop simulations of SRs following this approach. Its base is an existing modeling language related to road traffic which is adapted to specify the aspects of these systems (i.e. sensors, networks, and services), and their context (i.e. users, vehicles, and their environment). A process guides its use in the transition from abstract models to code supported by tailored tools. A case study on a system to track vehicles using sensors in roads illustrates its use.

## 1 Introduction

The availability of affordable sensors and actuators suitable for traffic settings is leading experts to redesign the related facilities. The goal is that they become *smart* environments, able to gather and analyze information, and react to it [Figueiredo *et al.*, 2001]. Intelligent Transportation Systems (ITSs) integrate these environments to provide services like [Figueiredo *et al.*, 2001] [Varaiya, 1993] vehicle tracking, congestion detection, or identification of road conditions. In this context, Smart Roads (SRs) [Wang *et al.*, 2006] are systems where the key devices are those deployed in roads and their elements.

The development of systems for SRs (and in general of ITSs) presents multiple challenges [Figueiredo *et al.*, 2001] [Varaiya, 1993] [Wang *et al.*, 2006]. First, these are complex and distributed systems. They comprehend multiple and heterogeneous software and hardware components. Their operating conditions are changing and demanding, as they are frequently deployed over wide geographical areas and partly outdoors. Among other issues, this implies that they have to deal with the failure and redeployment of components, energy saving, and limited and intermittent connectivity. Second, they affect activities that involve living beings and uncontrolled environments, so carrying out realistic and exhaustive testing is difficult and expensive.

Simulations help to mitigate the previous problems [Pursula, 1999]. With them, experts can control and observe the relevant variables of a problem, and designers perform an incremental development of systems. However, simulations also present some important drawbacks [Axtell and Epstein, 1994]. People involved in their development have different backgrounds, e.g. authorities, traffic experts, systems designers, or programmers of simulations and control. It is difficult for them having a complete understanding of the simulation at the different levels of abstraction and its multiple facets. For this reason, there are frequent problems to validate that simulation results correspond to the initial requirements.

Model-Driven Engineering (MDE) [Schmidt, 2006] can be used to address these issues [Fuentes-Fernández *et al.*, 2012]. In these developments, participants specify systems mainly using *models*. *Transformations* perform recurrent modifications of models and other artifacts, and describe mappings among them. In the case of simulations for SRs, traffic experts would model the abstract system (i.e. independently of specific platforms), and designers would ground it to specific devices and target simulation platforms. Part of the transition between both groups of models could be automated with transformations. For instance, abstract sensors and actuators usually correspond to certain classes in the target platform. In this way, the development of simulations becomes an iterative and incremental process of refining models and transformations where all the information is explicit. Since models have a higher level of abstraction than code, and transformations describe the relevant correspondences, this approach facilitates the exchange and discussion of information on simulations and artifact reutilization.

The adoption of MDE to develop SR simulations needs to have available an infrastructure that includes several elements. Domain Specific Modeling Languages (DSMLs) define the vocabulary to specify models in a determined context. There must be also languages to describe transformations, though here both specific Transformation Languages (TLs) and general-purpose programming languages are used. Participants need tools to work with these elements, such as model editors, transformation engines, or code generators. Finally, processes guide participants in developments with these elements. This work introduces a DSML for the high-level specification of simulations of SRs. Work with it is based on a process with tailored tools.

The DSML is adapted from a previous one related to road traffic [Fernandez-Isabel and Fuentes-Fernandez, 2015]. It is formed by three clusters according to the context they consider: a *behavioral cluster*, an *environment cluster* and an *interactive cluster*. The first describes the profiles and behavior of individuals, while the second takes into account the place where the simulation occurs and its elements. The last one uses elements commonly used by Agent-Oriented Software Engineering (AOSE) [Argente *et al.*, 2009] (i.e. goals and tasks) and a perception, evaluation, action cycle to represent the decision-making of individuals.

New primitives are introduced to model the main *elements* of SRs. These are the sensors and actuators that provide the interface of the system with the external world. That world includes people, their vehicles, and the environment. The environment in turn includes, at least, roads, signals, and general conditions (e.g. weather, daytime, or type of road). Part of these concepts are extracted from research in related domains, including Agent-Based Modeling (ABM) [Axtell and Epstein, 1994], traffic simulations [Fernandez-Isabel and Fuentes-Fernandez, 2015], and sensor networks [Fuentes-Fernández *et al.*, 2009].

Most of elements in these problems are represented in the DSML as *model elements*. These are related to the original DSML or to SRs components, having the latter an internal state, and an interface with methods to consult and manipulate them. Examples of components are sensors, actuators, and vehicles. A particular type of element are the *spots*. They represent components in the environment that can be observed and manipulated by devices (i.e. sensors and actuators). For instance, a tracking magnetic sensor can detect the passing of a bodywork *spot* of a vehicle.

SRs elements with complex behaviors are modeled as *agents*. They are defined in a similar way of individuals involved in road traffic (i.e. in terms of the goals they pursue and the information they have), and their capabilities to manipulate both their internal and the environment states.

The language also includes general mechanisms of inheritance between concepts and definition of instances of types in the adaptation to SRs. They facilitate the modification to different modeling needs through extensions of the language and the specification of simulations using models.

The related tools are a model editor and a code generator. Experts use the first one to specify graphically models compliant with the DSML. It is based on the INGENME [Pavón *et al.*, 2011] meta-editor. Designers and program-

mers use the generator (adapted from [Fernandez-Isabel and Fuentes-Fernandez, 2015] and based on Eclipse [Steinberg *et al.*, 2008] frameworks) to map model elements to code templates. These templates are fragments of code with marks corresponding to primitives of the DSML. Then, the code generator reads the models and mappings, instantiates the templates, and generates the code of simulations.

The case study that illustrates this approach is the simulation of the system in [Karpiriski *et al.*, 2006]. That work presents an architecture with road sensors to track vehicles. This case study models the system and generates its code for the JADE agent platform [Bellifemine *et al.*, 2007]. This illustrates how working in this way facilitates understanding the different aspects of the simulation and reduces the effort to code it.

The rest of the paper is organized as follows. Section 2 makes an introduction to MDE. Section 3 presents the DSML, while Section 4 the development guidelines of the proposal. Section 5 describes the support tools, and Section 6 applies the framework to the case study of tracking vehicles. Then, Section 7 compares the approach and its results with related work. Finally, Section 8 discusses the conclusions and future work.

## 2 Background

MDE [Schmidt, 2006; Kent, 2002] is an approach to software development based on *models* and *transformations*. Models are specifications of information regarding the system to build. Transformations are automated modifications of models and other artifacts to generate new products. In this context, developers work specifying their models incrementally, and running transformations to integrate models or perform certain modifications (e.g. adding design information or generating code). All the information relevant for the development is thus presented as models or transformations, so developers have it explicitly described. This improves traceability between artifacts across development. Working effectively in this way requires having support tools for certain tasks.

Models are described following Modeling Languages (MLs) that establish their primitives and constraints, so all developers can interpret them in similar ways. Model editors support developers when specifying models, and guarantee the compliance of models with their MLs. In order to allow this functionality, MLs are defined formally. There are alternatives for this definition depending on the ML features and the context and needs of its use. Domain Specific Modeling Languages (DSML) [Luoma *et al.*, 2004] are MLs oriented only to one context.

Graphical graph-oriented MLs are the most popular ones in contexts such as Software Engineering and graphical simulations. Their models specify graphs where entities are connected by links, and all of them can have related properties. Metamodels are the most widely used means to specify these languages [Steinberg *et al.*, 2008]. The description of metamodels relies on meta-modeling languages such as the Meta-Object Facility (MOF), Ecore [Steinberg *et al.*, 2008], or Graph-Object-Property-Relationship-Role (GOPRR) [Smolander, 1993]. MOF is used by the Object-

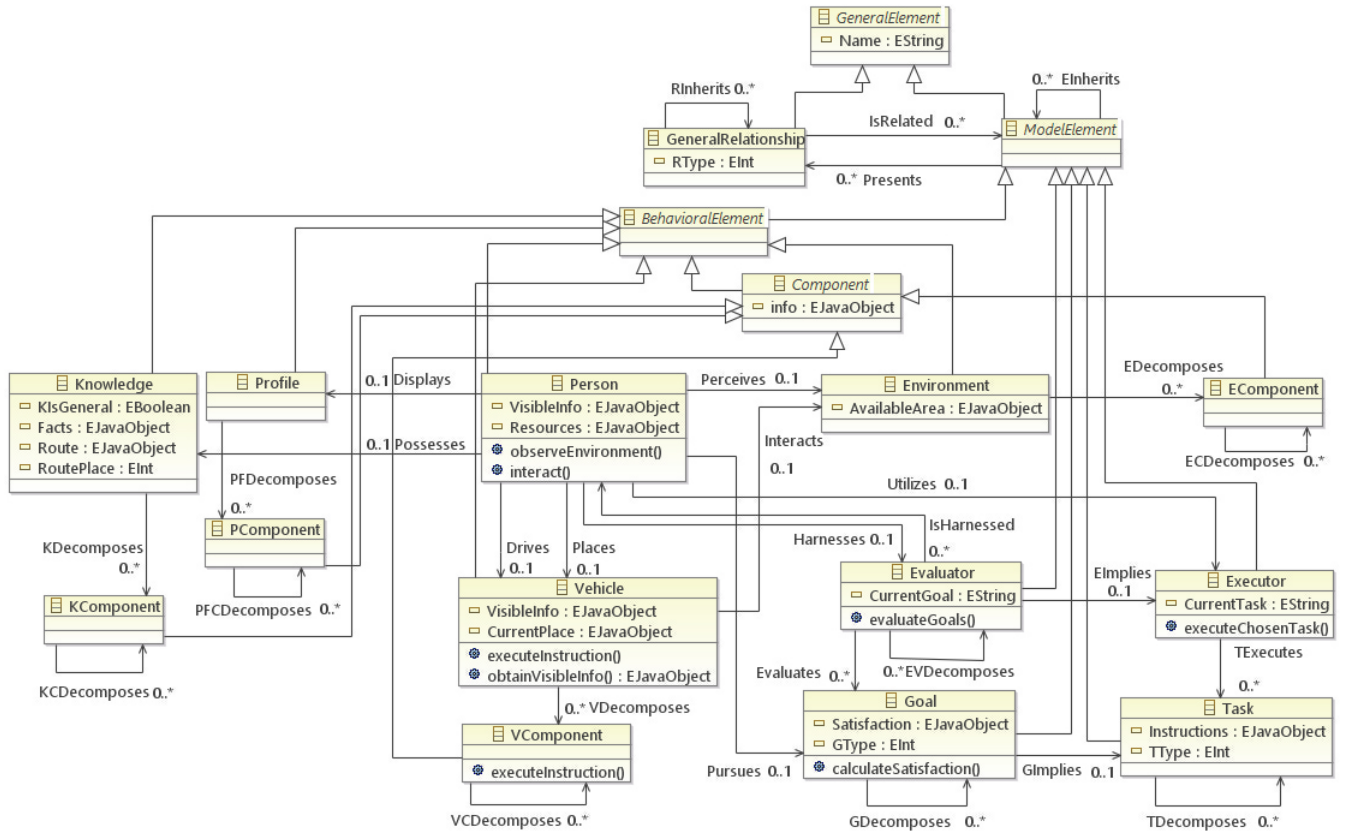


Figure 1: Excerpt of the road traffic metamodel.

Management Group (OMG) to define standards such as the Unified Modeling Language (UML). Ecore is almost aligned with Essential MOF, a subset of MOF. It is supported by the Eclipse communities related to MDE with a complete and widely supported set of tools. GOPRR has a richer set of primitives than the previous two languages, as it allows for instance the direct definition of n-ary relationships. INGENME [Pavón *et al.*, 2011] is a framework for it.

The implementation of transformations has two main approaches. They can be implemented as modules in mainstream programming languages; or they can be described with specific transformation languages and executed by an engine. The first approach reuses existing expertise and resources, and it is usually more efficient. Examples of it are INGENME [Pavón *et al.*, 2011] and the traffic simulation framework in [Fernandez-Isabel and Fuentes-Fernandez, 2015], both based on Java and XML. The second approach makes easier to examine the mappings between the source and target artifacts of the transformation. Examples of it are Eclipse projects such as JET and ATL [Steinberg *et al.*, 2008].

This work adopts GOPRR as its meta-modeling language. Its tools are based on the INGENME MDE framework.

### 3 Domain specific modeling language

The foundations of the DSML for SRs is adopted from another DSML focused on general purpose road traffic

[Fernandez-Isabel and Fuentes-Fernandez, 2015]. It is enhanced modifying its structure and introducing various abstract entities. The *model element* is the main one. It allows describing elements from road traffic (e.g. profile and knowledge of individuals involved in traffic) and specific components related to SRs (e.g. sensors and spots).

Regarding the naming, nodes are the meta-classes and links are meta-relationships among them. Meta-relationships with triangles represent inheritance and with filled diamonds aggregation. The attributes and adornments of the previous elements are meta-properties. It is specified with a GOPRR metamodel, being this notation similar to that also used in MOF and Ecore [Steinberg *et al.*, 2008].

The original metamodel is introduced in Section 3.1 where its structure and meta-classes are explained. The DSML extension developed to represent the SRs components is described in Section 3.2.

#### 3.1 Traffic DSML

The DSML is originally focused on modeling the behavior of individuals involved in road traffic (i.e. drivers, pedestrians and passengers). It is a flexible language that presents specific mechanisms to ease its fitness to the large amount of theories evaluated by traffic studies.

It is described by a metamodel [Steinberg *et al.*, 2008] which provides a set of meta-entities in order to represent the

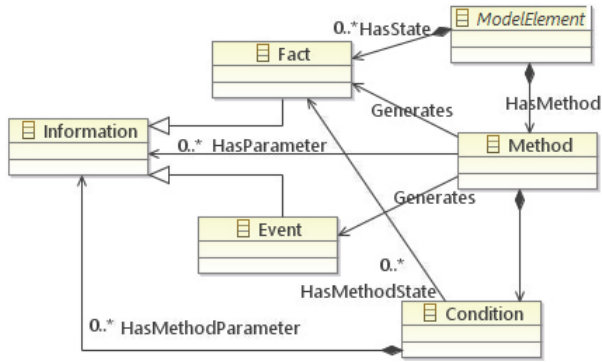


Figure 2: *Information* and *ModelElement* related elements.

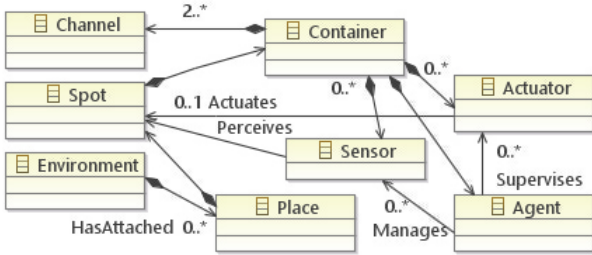


Figure 3: *Environment* and *Container* related elements.

notions, relationships, properties and explicit constraints.

Metamodel concepts are inspired in AOSE [Argente *et al.*, 2009] and are classified into three clusters. The *mental cluster* (i.e. *profile*, *knowledge* and their *components*) is based on [Shinar, 1978] and considers the features and internal state of individuals. The *environment cluster* (i.e. *environment*, *vehicle* and their respective *components*) describes the elements extracted from DVE model [Amditis *et al.*, 2010]. The *interactive cluster* (i.e. *evaluator*, *executor*, *goal* and *task*) represents the decision-making of individuals. It includes a perception, evaluation, and acting cycle.

Regarding the metamodel bedrock, it revolves around the *person* notion (see Fig. 1). It symbolizes a kind of human being involved in road traffic. According to their means of transport and how they interact with them, these people can take different roles (i.e. drivers, passengers, or pedestrians). Thus, *person* are able to interact with an *environment*. This interaction is immediate (for pedestrians) or indirect (in the case of drivers and passengers). The information people have is illustrated with the *knowledge*, while their features are represented by the *profile*. The purposes of people involved in traffic are described by *goals*, and the actions to carry out them by *tasks*. *Evaluators* study the information obtained from the *environment* and decide how the individuals must act according to it. *Tasks* for achieving their implicit instructions are picked up by an *executor*.

The metamodel uses inheritance hierarchies with the purpose of providing notion specializations and a flexible struc-

ture. The main one is the *general element* from which the *model element* and the *general relationship* extend (see Fig. 1). The first acts as a basis for the traffic DSML and the SRs DSML parts and is extended to the *behavioral element* and the meta-classes involved in the *interactive cluster*. The *behavioral element* is the parent meta-class of the main elements (i.e. not *components*) that compound the *mental* and *environment* clusters. *Component* extended from it and is also the parent of the *components* of both clusters. The second supports introducing relations (e.g. affect or impact) between the rest of entities that are extended from *model element*.

Another kind of hierarchies are considered in this part of the metamodel. In the both *mental* and *environment* clusters composition hierarchies are introduced between main elements (e.g. *profile* or *vehicle*) and their respective *components* (e.g. *pcomponent* or *vcomponent*). These *components* can be decomposed into others of the same type, promoting the creation of complex structures.

### 3.2 DSML extension to SRs

This module of the DSML is mainly focused on people moving in their vehicles in roads unless there are other elements in the environment such as traffic signals, obstacles, and weather. These elements can be observed with sensors, and systems that can actuate on them using actuators. The DSML makes of these concepts its core categories.

Elements are modeled in terms of the information they manage. There are two basic types of *information*: *facts* are internal to elements, and *events* can be perceived from outside.

The root concept that embraces both modules of DSML is the *ModelElement* entity (see Fig. 2). In this case, it is characterized in terms of an identifier, an internal *state*, an interface compose of *methods*, and the *events* it can generate. The internal state is a set of *facts*. A method is defined by its parameters and results, which are *information*. It can also have execution *conditions* defined in terms of their parameters and the internal state of its model element. Methods can be internal (only accessible from the component) or external (accessible from other components).

The *environment* meta-class of the traffic DSML is related to a set of elements over a *map*. These are the *places*, and can be located in *sections* or *junctions*. Examples of places are things in the environment, like the road surface or protective fence. *Environment* has attributes (e.g. *AvailableArea*) to store the relevant information of the *map*. This latter is represented through a graph that describes the road *sections* that link two *junctions* (one when the section in an entering or exit point).

*Places* contain *spots* (see Fig. 3). These are the components that sensors can actually observe and where actuators can act upon. For instance, a vehicle have several *spots*, e.g. the bodywork, the electronic system, or the engine. *Sensors* and *actuators* constitute the interface of systems with the external environment. They run on containers attached to *spots*. Besides this, a *sensor* is linked by the *perceives* relationship to the *spot* it observes, and an *actuator* by the *actuates* relationship to the *spot* it affects. In both cases, *sensors* and *actuators* access to the external interface of the *spot*, i.e. they

use its external *methods*. For instance, a *rain sensor* runs in a *container* of the bodywork, where it perceives the raindrops from an *abstract weather spot*. The *containers* of a system are linked through *communication channels*.

Beyond these elements, complex entities related to SRs are modeled as *agents*. Typical agents are the controllers of sensors and actuators. The control of components can be represented directly with their methods, but controller agents are recommended when there are complex algorithms and communication with other controllers.

Similarity to a *person* (see Fig. 1), an *agent* has an identifier, and *goals* that it can achieve through *tasks* that manipulate information. As the environment of SR systems is unpredictable, the execution of a task can fail or not to produce the expected results. Thus, a goal defines satisfaction conditions in terms of information to indicate when it has been fulfilled.

*Tasks* can be organized (or decomposed) into others according to the traffic DSML (see Fig. 1). Also, they can be linked to *methods* related to the information they produce and consume. In this way, tasks can use methods of components, including sensors and actuators.

Agents communicate among them using *notifications*. These are a type of event addressed to a certain *agent* identifier.

Agents behave internally following a perceive-reflect-act cycle. First, they execute those tasks that imply access to external components (e.g. sensors and other elements related to SRs) to gather data. Then, they update their internal state, both *facts* and *goals*. Finally, they pre-select for execution those *tasks* that can satisfy some of their still non-fulfilled goals. Among them, they choose one to actually execute.

The external elements that *agents* can access are those linked to them using *manages* relationships. For controllers, these relationship can be only with other elements in their *containers*. In contrary, *persons* can relate only to *sensors* and *actuators* from *containers*, or other elements outside *containers* like *vehicles* or *components* in the *environment*.

This part of the language also includes some general mechanisms applicable to most of the previous concepts. It supports inheritance of concepts and relationships to allow their specialization. For instance, the concept of component can have additional and different features according to the target simulation platform.

## 4 Development guidelines

The framework to develop SR simulations provides guidelines to model using the previous DSML and support tools. They include 11 activities. The process is decomposed in two different stages. The first one (nodes 1-9) is focused on the expert work and specifications with the proposed DSML. The second one (nodes 10-11) deals with the design of simulation. After concluding the first one with the specific infrastructure of this work, the second part can be addressed with a MDE methodology for general software development. Given that our DSML oriented to SRs follows ABM [Axtell and Epstein, 1994], methodologies from AOSE are a suitable choice [Argente *et al.*, 2009]. Both ABM and AOSE make of agents their core concept. Though there are differences among spe-

cific works, most of them conceptualize agents in terms of mental entities and communication capabilities, and consider the existence in their environment of artifacts they can use. This common core facilitates the transition from abstract to design models with transformations.

The first stage is organized around the services that the SR should provide. It starts identifying potential services pending to specify (activity 1). If there are any, work follows with its definition in terms of the information it needs and it provides, and the actions it should take (activity 2). This information and actions appear in elements of the system and its environment that next activities specify.

The service interacts with *spots*, either observing or changing them. Experts identify them and their potential *containers*, and specify them as *model elements* (activity 3).

The service also communicates with *spots* using the system *sensors* (described in activity 4) and *actuators* (in activity 5). These devices are initially specified as *model elements*. In case that their functioning needs complex control or communications with other containers, they also need controller *agents*. *Channels* must be added for those *containers* that need to be linked.

The *spots* identified in previous activities are located in elements of the SR environment. These elements are *places*, *vehicles* (considered in activity 6), *persons* (activity 7) and (in activity 8) other *components* from environment (i.e. *ecomponents*). All of them are specified as *model elements*. They also have a location in the environment. Thus, these activities also define the *map* and locate the places in it.

The last element to specify is the behavior of individuals (in activity 9). It is defined in terms of the *goals* and *tasks* adapting existing road traffic theories [Fernandez-Isabel and Fuentes-Fernandez, 2015], and the steps of the perception, evaluation and acting cycle. As people act on the environment, tasks to check the actual result of their actions and update its information must be included (e.g. route path or position in the environment).

When all the services have been identified, the process can move to the design of the simulation. Transformations map abstract to design models (activity 10). These transformations can be reused when they are available from other projects with the same target AOSE methodology. Then, the design models act as the initial specification for the simulation in that methodology (activity 11). For instance, our work can be easily linked to the INGENIAS AOSE methodology [Pavón *et al.*, 2005]. The INGENME [Pavón *et al.*, 2011] infrastructure our work uses is the same of INGENIAS, and both share similar definitions of concepts such as *agent*, *goal*, *task*, and *fact*.

## 5 Support tools

This MDE approach uses two development tools to achieve the different steps of the process (see Section 4). A model editor based on INGENME supports the specification of models compliant with the DSML. It is the main tool of the first stage. For the second stage, most of AOSE methodologies have their own tailored model editors and transformation tools. Here, the last steps of code generation are achieved with a



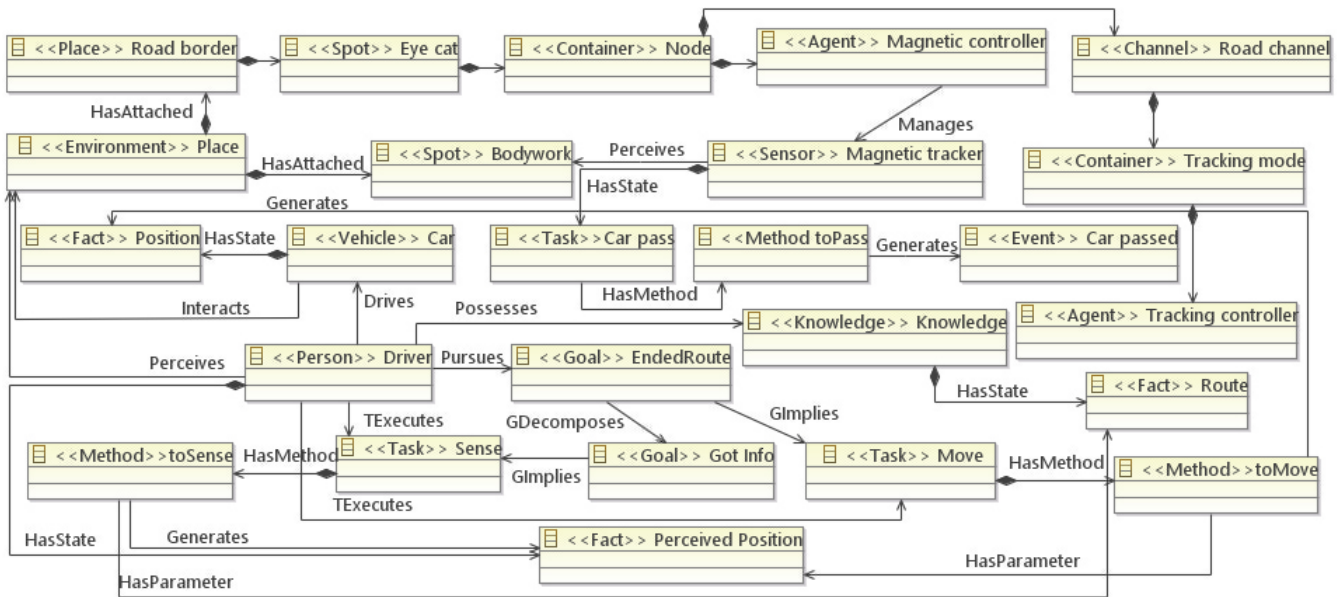


Figure 4: Excerpt of the simulation model for the system to track vehicles. Stereotypes indicate DSML types.

code generator adapted from [Fernandez-Isabel and Fuentes-Fernandez, 2015]. Then, designers specify graphically the mappings from elements in models to classes in the target platform. The generator outputs the simulation code from this information.

The INGENME model editor is based on INGENIAS methodology. It present a graphical canvas where models can be captured. The resulting outcome can be exported as a XML file. It promotes the modularity of the proposal and the ability to maintain the independence among the different artifacts created.

The code generator is a graphical tool implemented in Java which provides an engine to generate source code from a model specification. In this case, the specification comes in form of XML file from the model editor. This tool is mainly focused on easing the work to developers. In order to that, it provides an intuitive navigation through the elements of the models, and uses multiple wizards to guide the users in the achievement of the most complex tasks. It also takes as input a metamodel, code templates compliant with it and libraries from the target simulation platform. The templates support the preliminary automated code generation while the libraries can be used to adapt the simulation platform to the models requirements. To achieve it, the tool offers the possibility of creating new classes (empty or extended) through which producing a new simulation platform specialized in specific models [Fernandez-Isabel and Fuentes-Fernandez, 2015].

## 6 Case study

This section applies the previous framework (see Sections 3 and 4) to develop the simulation of one of the services for SRs described in [Karpiriski *et al.*, 2006]. It is a vehicle tracking service based on magnetic sensors located in nodes of cat eyes every few meters in road borders. These sensors are able to

perceive car passing. Nodes know their relative order and distance to others. They make up an ad-hoc network to exchange information, so they can determine the position and speed of vehicles.

The first stage of the process generates the abstract model of the simulation with the DSML. Fig. 4 shows part of it.

Activities 1 and 2 identify the services the SR offers. The definition of the problem points out only to the *tracking vehicle* service. Activity 3 identifies the *spots* related to it. There are two. The sensors are placed in *eye cats* in the road borders. These sensors track the passing of vehicles sensing their metal, for instance in their *bodywork*. Activities 4 and 5 consider the related *sensors* and *actuators*. There are *magnetic tracker sensors*, but no *actuators*. The specification of *car vehicles* in activity 6 does not consider any particular feature of them. In the proposed DSML, decisions regarding maneuvers are placed in *persons*. Activity 7 models them. The original work does not introduce any specific model of drivers. The problem also identifies elements in the environment (see activity 8) and models them in particular as *ecomponents* (see Section 3.1) or *places* (e.g. *road borders*).

The previous first round allows identifying the main concepts of the problem, and their types and relationships. A second round is focused on their state and functionality. Given that individuals with driver role and their actions trigger most of activities in the system, the analysis starts with them.

A simple path-following behavior is proposed for *persons* with driver role. Following a perception, reflect and acting cycle (no evaluation is considered), there is a first step of calculating position and a second of moving. The first one corresponds to *goal got info* and *task sense*, and the second to *goal ended route* and *task move* (see Fig. 4). *Sense* generates the *fact perceived position* consulting the *method toSense*. This fact is compared with the *fact route* to determine if the car has

arrived to its destination. That is the satisfaction condition of *goal ended route*. If the goal is unfulfilled, the *task move* can be triggered. It calls a *method toMove* that updates the *car* and its *fact position*.

In the system, several elements do not have specific state or methods, as only their location is relevant. This is the case of *road border* and *eye cat*. In order to have a precise location of sensors, road borders are modeled as multiple elements of this type, each one with its own location.

The *eye cat spot* has *node containers* for the sensors and controllers of the system. The *magnetic tracker sensor* triggers events when it perceives a car. Its *magnetic controller* has a *task pass car* to sense that event (see Fig. 4), and generates through the *toPass method* a *car passed event* addressed to a *central tracking controller* (not shown in the diagram). This would implement the services for end users described in [Karpiriski *et al.*, 2006]. Given the constraints imposed by the DSML, the model needs to introduce a *tracking node container* for this last controller. In order to enable communication between controllers in different containers, these are connected with a *road channel*.

The previous discussions has not considered the *map*. The studied work focuses on two-way single carriageways. With the DSML, these correspond to *junctions* that connect at least two *sections*, one for each direction. Crossroads are *junctions* that connect more *sections*. The previous *model elements* (vehicles and road borders) are placed in them.

These steps complete the first stage and the abstract model of the simulation. Activity 10 maps the abstractions of that model to those of INGENIAS [Pavón *et al.*, 2005]. Then, activity 11 follows the steps of this methodology adding the design information required for the JADE platform [Bellifemine *et al.*, 2007]. There are several entities of the DSML with direct mappings to JADE: agent to agent and task to behavior. Others need to be mapped to specific ad-hoc classes. This is the case of the model elements, facts, goals, spots, sensors and actuators. Their implementation only requires attributes and methods to get and set their values, as agents manage the updates of the simulation state.

## 7 Related work

The presented framework is mainly related to the modeling and development of SRs. This section discusses existing alternatives for them.

Under the label of SRs literature presents a variety of complex heterogeneous systems [Figueiredo *et al.*, 2001] [Varaiya, 1993] [Wang *et al.*, 2006]: they provide different services, for a wide range of users, and integrating multiple devices. Nevertheless, several common elements can be abstracted in most of them [Varaiya, 1993] [Wang *et al.*, 2006] [Sun *et al.*, 2006]. There are sensors and actuators embedded in an environment that includes roads and their elements, weather, vehicles, and sometimes people. Vehicles and people are usually modeled focusing on their movement. In the case of simulation [Kotusevski and Hawick, 2009], works offer more complex models of vehicles and people moving, with paths to follow and principles of movement (e.g. collision avoidance or observe traffic norms). These elements are

considered in the proposed DSML, though the use of *components*, *agents*, and *information* makes possible setting up richer models than in other works. The DSML also offers extension mechanisms frequently disregarded in other works.

In most cases, there is no information about the adopted development process (see for instance the already mentioned works). However, this is a key aspect to evaluate approaches regarding, for instance, ease of adoption and modeling or costs. When there is some information on that [Pursula, 1999] [Kotusevski and Hawick, 2009], it usually shows approaches with a manual transition from abstract models to code, and focused on the later. The advantages of MDE in this context were pointed out in the introduction: explicit definition of all the information related to that transition, which facilitates its discussion and reutilization, and support tools for it. There are already some works in this line in the context of traffic studies [Fernandez-Isabel and Fuentes-Fernandez, 2015] [Vangheluwe and De Lara, 2004]. Their main differences with our work are their focus and purposes (general traffic versus SRs) and the use of infrastructures less adopted than ours (in [Vangheluwe and De Lara, 2004] graph rewriting grammars for transformations), which hinders their adoption.

## 8 Conclusions and future work

This paper has introduced a framework for the model-driven development of simulations of SRs. It is based on a DSML and tailored standard infrastructures.

The DSML adopts a previous one focused on road traffic that integrates concepts from studies and simulations of traffic [Fernandez-Isabel and Fuentes-Fernandez, 2015]. It is adapted through ABM concepts (e.g. agents) [Axtell and Epstein, 1994] and elements related to sensor networks [Fuentes-Fernández *et al.*, 2009]. Based on a general concept of *model element* (could be a modeling entity or a component with state and an interface that manipulates information), it introduces new concepts related to sensors, actuators, spots in the environment and vehicles. A higher level of abstraction comes from agents, which are used to describe complex controllers.

The framework also proposes a development process based on this DSML. It comprehends a specific first stage to specify the abstract models of simulations, and connects with an AOSE MDE methodology [Argente *et al.*, 2009] for the low level design and code generation. A model editor based on INGENME [Pavón *et al.*, 2011] and a code generator supports it. This latter is a graphical engine based on wizards that allow guiding users in some of the complex steps of the development process.

The case study has shown how a simulation can be specified to a large extent with the DSML, including the behavior of its components. Only algorithms to manipulate information are demoted to code templates. This approach focus development efforts on models and transformation, which can be reused more easily than code.

The previous work has still several open issues. The SR module of the DSML needs extensions to consider aspects such as time or specific constraints. There are studies on



these issues but further work is required to integrate them. The process needs to incorporate additional advice on how to use the DSML and deal with the design. In particular, that design should be consistent with the expected semantics of the DSML. Finally, additional experiments are needed with other works and target platforms to validate the approach. Specifically, the combination of road traffic theories and services for SRs taking advantage of the potential of the DSML described here (it is a general purpose traffic DSML adapted to SRs) is another important next step to consider.

## Acknowledgments

This work has been done in the context of the project Social Ambient Assisting Living - Methods (SociAAL) (TIN2011-28335-C02-01) supported by the Spanish Ministry for Economy and Competitiveness. Also, we acknowledge support from the Programa de Creacin y Consolidacin de Grupos de Investigacin (UCM-BSCH GR35/10-A).

## References

- [Amditis *et al.*, 2010] Angelos Amditis, Katia Pagle, Somya Joshi, and Evangelos Bekiaris. Driver-vehicle-environment monitoring for on-board driver support systems: Lessons learned from design and implementation. *Applied Ergonomics*, 41(2):225–235, 2010.
- [Argente *et al.*, 2009] Estefanía Argente, Ghassan Beydoun, Rubén Fuentes-Fernández, Brian Henderson-Sellers, and Graham Low. Modelling with agents. In *Agent-Oriented Software Engineering X*, pages 157–168. Springer, 2009.
- [Axtell and Epstein, 1994] Robert L Axtell and Joshua M Epstein. Agent-based modeling: understanding our creations. *The Bulletin of the Santa Fe Institute*, 9(2):28–32, 1994.
- [Bellifemine *et al.*, 2007] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing multi-agent systems with JADE*, volume 7. John Wiley & Sons, 2007.
- [Fernandez-Isabel and Fuentes-Fernandez, 2015] Alberto Fernandez-Isabel and Ruben Fuentes-Fernandez. Developing an integrative modelling language for enhancing road traffic simulations. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1745–1756. IEEE, 2015.
- [Figueiredo *et al.*, 2001] Lino Figueiredo, Isabel Jesus, JA Tenreiro Machado, J Ferreira, and JL Martins De Carvalho. Towards the development of intelligent transportation systems. In *Intelligent Transportation Systems*, volume 88, pages 1206–1211, 2001.
- [Fuentes-Fernández *et al.*, 2009] Rubén Fuentes-Fernández, María Guijarro, and Gonzalo Pajares. A multi-agent system architecture for sensor networks. *Sensors*, 9(12):10244–10269, 2009.
- [Fuentes-Fernández *et al.*, 2012] Rubén Fuentes-Fernández, Samer Hassan, Juan Pavón, José M Galán, and Adolfo López-Paredes. Metamodels for role-driven agent-based modelling. *Computational and Mathematical Organization Theory*, 18(1):91–112, 2012.
- [Karpiriski *et al.*, 2006] M Karpiriski, Aline Senart, and Vinny Cahill. Sensor networks for smart roads. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 5–pp. IEEE, 2006.
- [Kent, 2002] Stuart Kent. Model driven engineering. In *Integrated formal methods*, pages 286–298. Springer, 2002.
- [Kotusevski and Hawick, 2009] G Kotusevski and KA Hawick. A review of traffic simulation software. *Computer Science. Institute of Information and Mathematical Sciences, Massey University*, 2009.
- [Luoma *et al.*, 2004] Janne Luoma, Steven Kelly, and Juha-Pekka Tolvanen. Defining domain-specific modeling languages: Collected experiences. In *4 th Workshop on Domain-Specific Modeling*, 2004.
- [Pavón *et al.*, 2005] Juan Pavón, Jorge J Gómez-Sanz, and Rubén Fuentes. The ingenias methodology and tools. *Agent-oriented methodologies*, 9:236–276, 2005.
- [Pavón *et al.*, 2011] Juan Pavón, Jorge Gómez-Sanz, and Adolfo López Paredes. The sicossys approach to sos engineering. In *System of systems engineering (SoSE), 2011 6th international conference on*, pages 179–184. IEEE, 2011.
- [Pursula, 1999] Matti Pursula. Simulation of traffic systems-an overview. *Journal of Geographic Information and Decision Analysis*, 3(1):1–8, 1999.
- [Schmidt, 2006] Douglas C Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY*, 39(2):25, 2006.
- [Shinar, 1978] David Shinar. *Psychology on the Road. The Human Factor in Traffic Safety*. John Wiley & Sons, 1978.
- [Smolander, 1993] Kari Smolander. Goprr: a proposal for a meta level model. *University of Jyväskylä, Finland*, 1993.
- [Steinberg *et al.*, 2008] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.
- [Sun *et al.*, 2006] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, 2006.
- [Vangheluwe and De Lara, 2004] Hans Vangheluwe and Juan De Lara. Computer automated multi-paradigm modelling for analysis and design of traffic networks. In *Proceedings of the 36th conference on Winter simulation*, pages 249–258. Winter Simulation Conference, 2004.
- [Varaiya, 1993] Pravin Varaiya. Smart cars on smart roads: problems of control. *Automatic Control, IEEE Transactions on*, 38(2):195–207, 1993.
- [Wang *et al.*, 2006] Fei-Yue Wang, Daniel Zeng, and Li-ying Yang. Smart cars on smart roads: an ieeec intelligent transportation systems society update. *IEEE Pervasive Computing*, (4):68–69, 2006.