LEVERAGING DIFFERENCE REWARD IN MULTIAGENT SYSTEMS AT CITY-SCALE NETWORKS USING MESOSCOPIC SIMULATION

Diogo M. F. Rodrigues, M.Eng., Corresponding Author
Artificial Intelligence and Computer Science Laboratory (LIACC)
Department of Informatics Engineering
Faculty of Engineering of the University of Porto
4200-465 Porto, Portugal
diogo.rodrigues@fe.up.pt
ORCiD 0009-0003-7957-0064

Rosaldo J. F. Rossetti, PhD.
Artificial Intelligence and Computer Science Laboratory (LIACC)
Department of Informatics Engineering
Faculty of Engineering of the University of Porto
4200-465 Porto, Portugal
rossetti@fe.up.pt
ORCiD 0000-0002-1566-7006

Word Count: 7214 words + 2 table(s) $\times$ 250 = 7714 words

Submission Date: March 12, 2024

ABSTRACT
People living in cities incur significant costs owed to traffic congestion, meaning that small improvements in drivers' experience in urban traffic can lead to significant economic and health benefits. Drivers are guided by greedy principles as a result of selfish behavior, incomplete and imperfect information, bounded rationality, and lack of a central coordination entity. This results in the sub-optimal User Equilibrium, distinct from the System Optimum where global travel time is minimized. The goal of this work is to mitigate the gap between User Equilibrium and the System Optimum, by performing a city-scale analysis of traffic congestion, modeling drivers as Q-learning agents, and using the concept of difference reward functions to find routes that minimize global costs by internalizing external costs created by excessive congestion. To implement these concepts, a mesoscopic simulator was developed to accurately represent the phenomenon of traffic congestion, while allowing analysis at the scale of a city. Attained results with presented methods revealed that agents were able to learn better paths in a dynamic environment, and point towards reductions in global cost metrics in the order of 0.5% when using less greedy reward functions, approaching the System Optimum.

Keywords: Operations, Traffic assignment, Dynamic traffic assignment, Mesoscopic simulation, Reinforcement Learning, Difference reward

INTRODUCTION

Since the 1950s, the world has gone through a process of rapid urbanization. In 1950 the urbanization rate in Europe was only 51.7% (*1, 2*), which has risen to 74.5% by 2018 (*2*), and is estimated to reach 83.7% in 2050 (*1, 2*). This increasing trend is common to the entire world, with urban population accounting for 29.6% of the world population in 1950, 53.9% in 2015, and 68.4% by 2050 (*1*), and is particularly expressive in Asia and Africa, which are expected to contribute with 90% of the 2.5 billion increase in urban population until 2050 (*2*). Car ownership is also steadily increasing in the European Union (EU), at a yearly rate of 1.2%, totaling 246.3 million passenger cars (*3*) for a population of 446.7 million in 2022 (*4*). These two trends contribute to a high and ever-increasing traffic congestion burden that urban residents incur. The European Commission estimates that total passenger congestion costs amount to 0.98% of the Gross Domestic Product of the EU 28 member states (EU28), or nearly 140 B€/year. And congestion costs are only expected to continue increasing in the EU, by about 50% by 2050 to nearly 200 B€/year (*5*). It is, therefore, paramount to find ever-smarter solutions so that people can live harmoniously in dense communities, as it is only expected that cities will grow in size and population, accentuating congestion problems.

The way people move on a traffic network is guided by greedy principles, where each user selects the most convenient route for themselves. This is the result of applying the concept of Nash equilibrium (*6*) to the traffic assignment problem to arrive at an economic equilibrium. There are some advantages associated with this strategy, namely not requiring communication with fellow users (*7, 8*). It is also intrinsically fair, allowing users to have distinct ways of evaluating possible paths, and entitling every user to make self-interested decisions (*8, 9*). The equilibrium situation in which everyone is greedy is called User Equilibrium (UE), for no user may improve their individual outcomes by unilaterally changing paths (*10*).

This situation differs from the System Optimum (SO) (*10*), where a global system utility function is optimized. This global utility function can be defined so as to attain relevant societal goals such as better infrastructure usage through reduced congestion costs, pollution, and infrastructure maintenance (*11*). A consequence of the definition of SO is that any other solution is suboptimal. In fact, Klein et al. (*12*) assessed that the SO solution is optimal but 'unstable', due to the inherent unfairness (*9*) of some users making concessions for the greater good, and that any of those users' move towards an individually better path would inevitably stray from the SO. The gap between UE and SO is also known as the *price of anarchy*. As pointed out by Lujak et al. (*9*), UE can be arbitrarily more costly than SO, and some studies have noted that this price of anarchy can vary between 0.5% and 15% (*13*).

Given there is a difference between UE and the SO, there is a margin for improvement. This work sets out to propose an agent-based Reinforcement Learning (RL) approach to reduce the gap between UE and SO, using the concepts of Q-learning and *difference reward* on a large-scale mesoscopic simulated environment.

LITERATURE REVIEW

Related work

The first and simplest approach to bridge the UE-SO gap is through tolls imposed on users by an infrastructure management entity or city authority. This approach was first mentioned by Pigou (*14*) and Knight (*15*), which acknowledged that an optimal outcome could only be reached if externalities were properly internalized by the agents responsible for them through taxes. This

implies a user being charged a toll fee equal to the externalities it imposes on all other users. Assuming the time a vehicle takes to cross a road a is a function $t_a(x_a)$ of the vehicular flow x_a through that road, the toll fee T_a to be imposed on a road a so that the optimal outcome is achieved must be the individual penalty $t'_a(x_a)$ that another user j incurs due to the decision of i to use road a, times the number of vehicles x_a using road a:

$$T_a = x_a \cdot t'_a(x_a) \tag{1}$$

In this line of investigation, Ramos et al. (*16*) applied RL to a simple macroscopic traffic model using the Bureau of Public Roads (BPR) (*17*) formula, associated with a tolling scheme based on Equation 1, where agents learn by deciding between the K shortest paths. Recent work also focuses on more realistic alternatives to SO tolling, also known as *first-best* tolling, due to the impracticality of tolling all roads. These tolling schemes are divided into *second-best* tolling, where only a subset of links is subject to toll charge (*18*); and *third-best* tolling, where other goals are set without consideration for the overall impact of tolls on other roads, of which *no-queue* tolling is an example, where only links with frequent queue formation are tolled (*19*).

Other approaches that have found traction in the literature are related to auctions and negotiations, for their versatility and ability to describe feasible and fair distributed multi-agent traffic systems. The basis for current methods on reservations and interactions between vehicle and intersection agents was introduced by Dresner and Stone (*20*). This concept was further expanded by Vasirani (*21*) and Vasirani and Ossowski (*22*) through a distributed, market-inspired mechanism based on combinatorial auctions, where intersection agents participate in the market as suppliers and vehicle agents as buyers. Schepperle and Böhm (*23*) proposed the valuation-aware variant *Initial Time-Slot Auction*, and in a related paper Schepperle et al. (*24*) propose *Time-Slot Exchange*, an optional mechanism through which vehicle agents can bilaterally negotiate slot exchanges after obtaining an initial slot they are not satisfied with. More recently, Rahimi-Farahani et al. (*25*) proposed a *forced-node* route guidance system, where a selected set of intersection and roundabout agents makes binding decisions on which roads vehicles should use so as to avoid congestion, and Elimadi et al. (*26*) proposed an *Agent Node-Centric* approach where vehicle agents book times for crossing intersections while intersection agents adjust booking times according to demand.

Q-learning

Q-learning is a model-free RL algorithm proposed by Watkins (*27*). Provided with a function that maps a (state, action) pair to a reward, it finds an optimal policy that maximizes total reward. Q-learning draws many parallels from the generic paradigm of Dynamic Programming (DP) in that it maintains a data structure that is successively modified to compute optimal policies (*28*).

Q-learning algorithms work by maintaining a structure, or a function $Q : S \times A \to \mathbb{R}$, that is built by the Q-learner from memory of past decisions and is used to decide the next course of action. This function $Q(s,a)$ is used to estimate the quality of the decision for an action $a \in A$ given that the agent is currently at state $s \in S$, so given a set of actions a that an agent may perform starting in state s, the best action is such that maximizes $Q(s,a)$. The agent's learning process consists of progressively updating the Q function according to the rewards that the agent obtains from the actions it takes.

One of the fundamental principles of Q-learning is that the Q function can express not only instantaneous rewards but also future rewards. This is done by updating the Q with the following

formula:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \lambda \cdot \max_a Q(s_{t+1}, a)) \tag{2}$$

where α is the learning rate, r_t is the reward for moving from state s_t to s_{t+1}, $\max_a Q(s_{t+1}, a)$ is an estimate of the maximum possible future reward that can be obtained from transitioning to s_{t+1}, and λ is the *discount factor*, which represents how important future rewards are.

The generic formulation of the Q-learning algorithm leaves the exploration vs. exploitation problem as an open question. A particular type of Q-learning algorithms that addresses this issue is called ε-greedy, because they exploit the action with the largest expected reward with a usually large probability $1-\varepsilon$, but occasionally explores the environment with probability ε to try to find better alternatives (and as such, escape local minimums) by selecting a random action.

Difference reward

Difference reward is a technique introduced by Tumer and Wolpert (*29*) to model a reward function that is sensitive to agent decisions while taking into account the global cost function. A difference reward is the reward that the agent receives as a result of it being a part of the system; so it translates to how much the global utility increased or decreased due to the actions of one agent. Therefore, if rewards are negative costs, the agent will indirectly try to minimize their negative impact on global cost. More formally, a difference reward, or difference utility DU, is defined by Tumer and Wolpert (*29*) for agent η as a utility of the form

$$DU(z) = G(z) - \Gamma(f(z)) \tag{3}$$

where G is the global utility function, and $\Gamma(f)$ does not depend on the decisions of agent η. Tumer and Agogino (*30*) suggest a way in which this generic framework can be materialized in a more understandable way, with DU expressed by:

$$D_i \equiv G(z) - G(z - z_i + c_i) \tag{4}$$

where z is a global solution, z_i is the action of agent i, and c_i is a fixed constant. Under this notation system, $z - z_i + c_i$ is the global state if the action of agent i is replaced by a constant c_i; this constant is typically a neutral element such as a zero vector, so that $G(z)$ reflects the global utility including the actions of agent i, and $G(z - z_i + c_i)$ represents the global utility if agent i was completely removed from the system.

This reward function was created so agents could rapidly converge toward a global optimum. The intuitive reason behind its effectiveness is that the agent is aware of its actions' influence over other agents in the system (*31*) and will aim to maximize its positive impact in the global utility function; or otherwise, if rewards are negative, the agent will seek to minimize its negative impact in the system. In sum, difference reward internalizes externalities caused by the agent's decisions, and as such constitutes a form of Pigouvian tax imposed on agents to prevent an unwanted market outcome of excessive congestion (*14, 15*).

According to Tumer and Wolpert (*29*), the quality of a reward function g_i for agent i on a system with a global cost function G depends on its *factoredness* and *learnability*. Factoredness is associated with the alignment between the reward function and the global utility; i.e., if the designed reward function g frequently increases when G also increases, then the reward function has a high degree of factoredness. Learnability is associated with agents easily understanding their

roles in the greater picture of the system; it is akin to a signal-to-noise ratio, since high learnability corresponds to the agent getting a "cleaner" signal (*31*).

The application of difference reward to alleviate traffic congestion was spearheaded by Tumer et al. (*32*) using a simple static scenario with one origin-destination (OD) pair and 9 time slots (so, 9 possible "routes" in total). Grunitzki et al. (*31*) further expanded this analysis by considering a dynamic traffic assignment scenario with 13 junctions and 24 two-way links, with a significantly larger number of possible routes. Both are relatively small and simple scenarios, which cannot be compared with the real-world scenario of a large city with thousands of junctions and roads, that have to move hundreds of thousands of vehicles per hour.

METHODOLOGY

Data

The traffic network under analysis is the traffic network of the city of Porto[1]. It is geographically bounded by the borders of the municipality of Porto, plus complementary roads and major highways that cross municipality borders. This data was obtained from OpenStreetMap (OSM) (*33*), using JOSM (*34*) to download a local working version.

All available OSM data was downloaded for the bounding box with coordinates (41.05°, −8.70°) and (41.20°, −8.50°) (latitude, longitude respectively), yielding a total of 212.2 MB raw OSM data including versioning metadata (`porto-unbounded.osm`). This data was then filtered to remove less important roads outside the municipality of Porto, so that an OSM file was obtained containing all data inside the municipality of Porto plus motorways in the vicinity (`porto.osm`), totaling at 100.8 MB. Finally, the OSM file is converted to the SUMO (Simulation of Urban MObility) (*35*) format and stored as `porto.net.xml`, with 18.1 MB. The network file includes all roads traversable by motorized vehicles, except for roads classified in OSM as `living_street` and `service` since they have very low through traffic.

Some manual fixes were made to improve the quality of the network data. The majority of fixes were made directly to the public OSM dataset, and those that could not be done in OSM were made in the resulting SUMO files.

The traffic network that was used in this study (see Figure 1) contains:

- 4690 junctions;
- 8749 edges;
- 11 544 lanes, each lane with an average length of 105.42 m;
- 16 823 connections;
- 404 traffic light programs and 54 roundabouts.

Note that, as per the SUMO network format, a traffic light program usually controls one junction but may simultaneously control multiple junctions that are geographically near each other.

[1]The data used in this study is available at `https://github.com/dmfrodrigues/dynaminator-data`

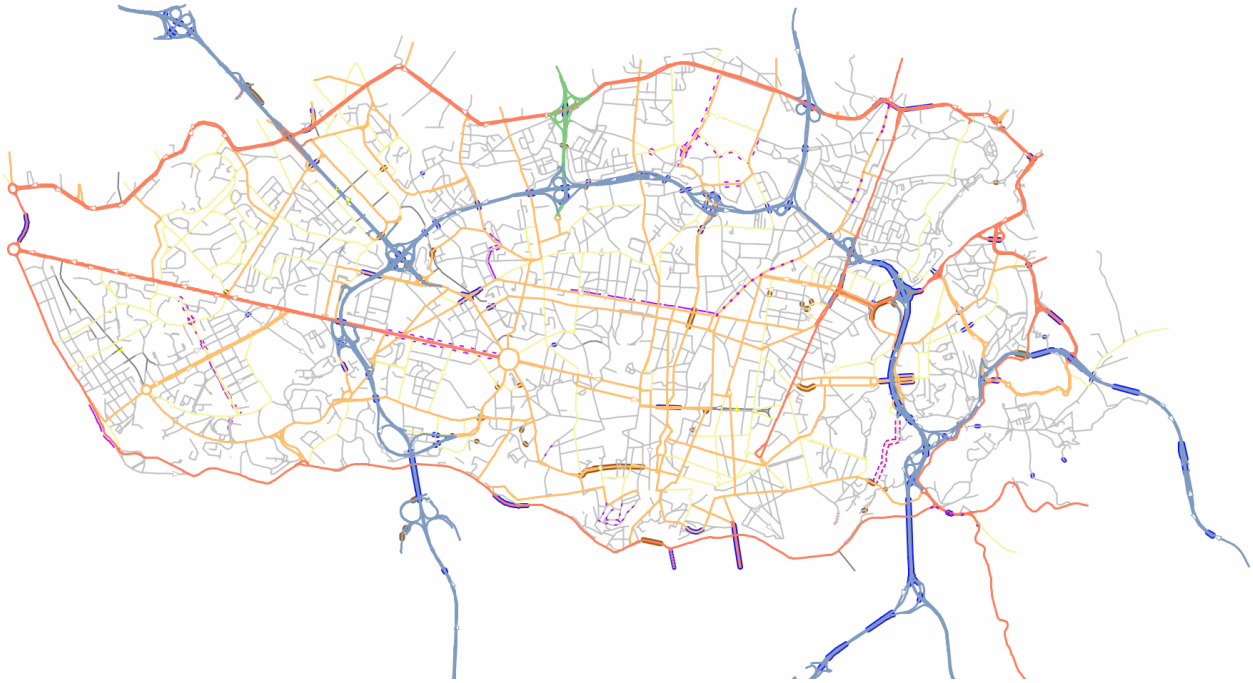

FIGURE 1 Traffic network of the municipality of Porto with relevant road classes. Rendered with JOSM (*34*).

The Traffic Assignment Zones (TAZ) and demand data were provided by the municipality of Porto. This dataset includes:

- A SUMO network file, named `porto-armis.net.xml`.
- A SUMO TAZ file, named `porto-armis.taz.xml`, where TAZ edges are identified with respect to network file `porto-armis.net.xml`.
- Two files of demand data corresponding to demand estimates between 09:00-10:00 and 10:00-11:00, named `matrix.9.0.10.0.armis.fma` and `matrix.10.0.11.0.armis.fma` respectively. The demand data files are in VISUM (*36*) O-format (*37*) to represent origin-destination (OD) matrices, with estimated vehicle counts between each pair of TAZs over a 1 h period. The TAZs are identified with respect to TAZ file `porto-armis.taz.xml`.

The TAZ file identifies 119 TAZs in the city of Porto and its vicinity, and also identifies the edges of `porto-armis.net.xml` that are part of each TAZ; in total, there are 1086 edges from `porto-armis.net.xml` that are part of TAZs. Files `matrix.9.0.10.0.armis.fma` and `matrix.10.0.11.0.armis.fma` contain 102 731 and 96 431 OD pairs respectively. Figure 2 presents an exploratory visualization of demand data from 09:00-10:00.

Since the roads and IDs used in `porto-armis.net.xml` are different from those in `porto.net.xml`, it was necessary to identify the edges in `porto.net.xml` that corresponded to edges in TAZs in `porto-armis.net.xml`, and create a new SUMO TAZ file `porto.taz.xml` that uses the edge IDs in `porto.net.xml`.

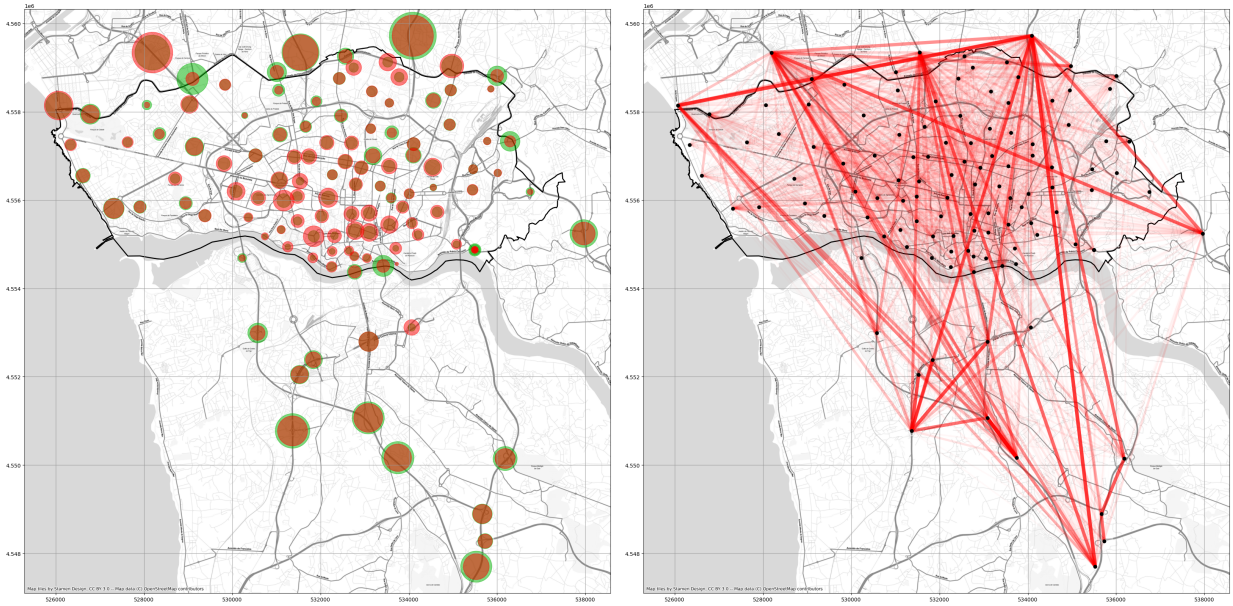

FIGURE 2 Exploratory visualization of demand data from 09:00-10:00. Left: TAZs mapped by the number of departures (green) and arrivals (red). Right: Overview of major vehicle flows between TAZs.

DynamiNATOR

A new mesoscopic, event-driven dynamic traffic simulator, named DynamiNATOR (*38*), was developed to perform the experiments in this work[2]. The simulator implements congestion effects, stopping at traffic lights, right-of-way rules, queues, and queue spill-back. This simulator was inspired on SUMO (*39*), a microscopic timestep-based traffic simulator that uses a car-following algorithm, considered the most widely used open-source traffic simulator in the scientific area. To contextualize the developed simulator, Figure 3 shows an overview of the data types that DynamiNATOR consumes and produces, and how that data is obtained and visualized, respectively. It receives a network description, composed of files `*.net.xml` and `*.taz.xml` describing the network and the TAZs, both in SUMO format (*40, 41*). Demand is provided in a `*.fma` file, in VISUM O-format (*37*). Outputs are in the form of: *route* files, containing the list of edges that vehicles traverse, and can be provided to SUMO for further processing or to visualize vehicle routes; *edge-data* files, containing time-variable edge data, and can be visualized with SUMO GUI (Graphical User Interface); and *netstate* files, also known as *raw dumps*, contain a list of timesteps, each of which is a snapshot of the network state at a time instant, containing the position and velocity of each vehicle.

This model requires three parameters. *Average vehicle gap* l_v is the average value of the sum of the vehicle length and the distance between the front of the vehicle and the back of the next vehicle in a traffic jam. It is therefore the effective space that vehicles take up in a queue. Its value was empirically estimated as $l_v = 6.52\,\mathrm{m}$ using JOSM (*34*) and Bing aerial imagery (*42*). *Queue dissipation speed* v_q is the average speed of all vehicles in a dissipating queue. Its default value was estimated to be $v_q = 3.44\,\mathrm{m/s}$ from $v_q = c \cdot l_v$, where $c = 1900\,\mathrm{h}^{-1}$ is the default value for

[2]The simulator was made publicly available at `https://github.com/dmfrodrigues/dynaminator`

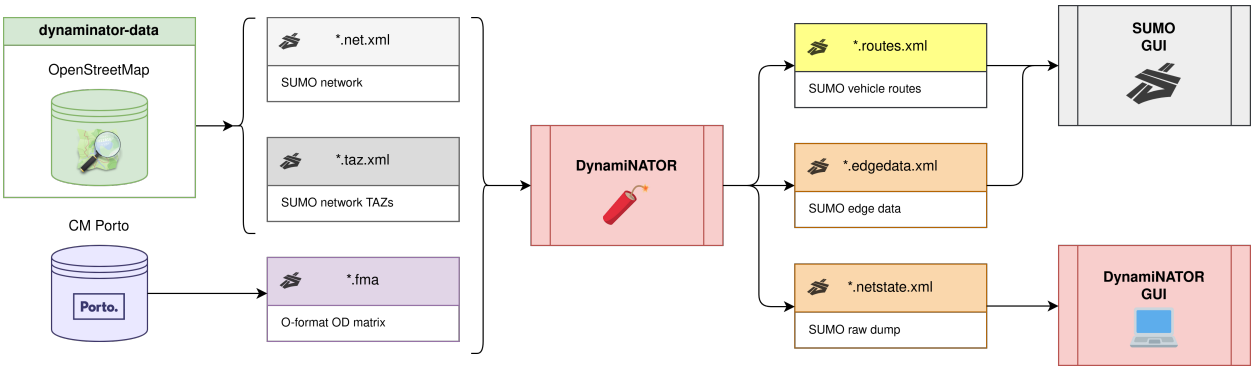

FIGURE 3 Data flow diagram of DynamiNATOR.

base saturation flow rate provided in the Highway Capacity Manual (*43*) without any adjustments. *Critical gap* t_{cr} is the minimum headway between consecutive vehicles that a less-priority vehicle can accept when joining a higher-priority flow. Ashalatha and Chandra (*44*) point to typical values of the critical gap between 2 s to 3 s, so it was decided to use $t_{cr} = 2.5\,\text{s}$.

Model

The traffic network is modeled as a set of junctions and edges. An edge is a road with one or more lanes in a single direction. A junction is a space-limited area that serves as an interchange where vehicles may change between edges, and it may be controlled by traffic lights. The atomic unit of road in this model is the *lane*, and the atomic unit of traffic flow is the *vehicle*. A vehicle is described by its position (lane and offset since the beginning of the lane), speed, and state. Its state is one of either:

- `MOVING`: the vehicle is moving under uncongested flow;
- `STOPPED`: the vehicle is stopped at a queue or is part of a queue that is dissipating;
- `LEFT`: auxiliary state, indicating the vehicle left the environment.

Uncongested flow occurs when vehicles travel relatively unimpeded at a constant self-selected speed which depends on the vehicle density K in the current lane. Greenshields (*45*) model was adopted for its simplicity to determine vehicle speed choice under uncongested flow, following Equation 5 where K_{jam} is the jam density (calculated as $K_{\text{jam}} = 1/l_v$) and v_0 is the free-flow speed, assumed to be the speed limit of the corresponding lane. The speed that a vehicle adopts when traversing a lane is constant and determined when the vehicle enters the lane. This means any vehicle j entering a lane after vehicle i has no impact on the speed of vehicle i. Additionally, if v is calculated to be less than v_q, v is set to v_q so as to avoid issues when the vehicle density K approaches the jam density K_{jam}.

$$v = v_0 \left(1 - \frac{K}{K_{\text{jam}}}\right) \tag{5}$$

A traffic queue can occur if the traffic light controlling a connection is red, or if a vehicle must yield to another vehicle according to right-of-way rules. A queue is represented as a First-In First-Out (FIFO) data structure, and may only form at the end of a lane. The offset of vehicle i that is in a queue is given by

$$x_i = L - n_i^q \cdot l_v \tag{6}$$

where L is the edge length, and n_i^q is the ordinal position of the vehicle in the queue, where $n_i^q = 0$ would indicate that vehicle i is at the front of the queue. Similarly, the length q and the position x_q of the end of a queue of size Q are given by

$$q = Q \cdot l_v \tag{7}$$

$$x_q = L - q \tag{8}$$

If the obstacle that created the queue disappears, the queue dissipates at constant speed v_q. The *capacity* of a queue is the number of vehicles that can fit in that queue, and is calculated from Equation 9. This formula prevents the edge case where the capacity may be less than one vehicle.

$$C_q = \left\lceil \frac{L}{l_v} \right\rceil \tag{9}$$

Queue spill-back was also implemented so that if a vehicle tries to move to a lane with a full queue, the queue spills back to the current lane.

If a vehicle in a low-priority connection approaches a prioritized intersection, it must yield if there are any vehicles in a higher-priority connection. Specifically, each connection c keeps a timestamp t_c of the last time it was used, and a vehicle trying to use a low-priority connection must wait at least until a time interval of t_{cr} has passed since the most recent usage of a connection with higher priority; in other words, if a vehicle is trying to use a connection c, and R_c is the set of connections c' with higher priority than c, the vehicle must wait until at least the time instant t described in Equation 10.

$$t = t_{cr} + \max_{c' \in R_c} t_{c'} \tag{10}$$

When a vehicle enters a lane, it can change to any other lane on the same edge. A vehicle is never prohibited from changing lanes, but if the destination lane is full, the vehicle is enqueued at the previous junction. Lane changing does not incur any penalty since it would not significantly impact travel time; additionally, penalties are already incurred at traffic lights, prioritized intersections, and high-density uncongested lanes.

DynamiNATOR is an event-driven simulator, meaning the environment state is only updated when significant events occur. The two most significant events that are implemented in the simulator are: *EventUpdateVehicle*, which is scheduled when a vehicle is under uncongested flow, and is triggered when the vehicle reaches the end of a lane and intends to change lanes or be enqueued; *EventPopQueue*, which is triggered when a queue must be popped. Event-driven design leads to a very high-performance simulator, as shown in the results section on the comparison with SUMO.

Reinforcement Learning

Under the developed model, a state $s(l)$ consists of being at the very end of a lane l, and an action $a(c, l'')$ consists of the physical action of using connection $c(l, l')$ to move from lane l to lane l', followed by changing lanes in the same edge e' from lane l' to lane l'', and in the meantime traversing the full length of edge e'. For a state-action pair to be valid, the action connection must start at the same lane state.

The ε-greedy Q-learning algorithm was implemented as described by Equation 2. We assumed that if s_f is a final state, then $\forall a, Q(s_f, a) = 0$. As such, if we have a state-action pair

(s_{f-1}, a_{f-1}) that leads to s_f, the reward is the cost that the vehicle incurred to go from state s_{f-1} to s_f through action a_{f-1} (which is the cost of traversing e_f plus the cost of changing lanes). This reasoning hints at the general conclusion that an initial estimate for $Q(s,a)$ can be made by the following process. Consider $a = a(c(s,l^*),s^*)$, such that s^* is the result of executing action a from state s. We additionally define that l_i^* is the initial position at lane l^*. We then have for the initial estimate of Q,

$$Q_0(s,a) = |c| + |l_i^* \rightarrow s^*| + |s^* \rightsquigarrow s_f| \tag{11}$$

where $|c|$ is the cost of traversing connection c; $|l_i^* \rightarrow s^*|$ is the cost of changing lanes from the start of lane l^* until reaching state s^*, while traversing the edge e that both lanes belong to; and $s^* \rightsquigarrow s_f$ is the shortest path from s^* to s_f, making $|s^* \rightsquigarrow s_f|$ the cost of this path. $|c|$ was set to 0 in prioritized intersections; for traffic light-controlled intersections, it was estimated from the red period r and cycle length C by using the expression $|c| = r^2/2C$ (*46*). In order to reduce memory usage and speed up the learning process, it was decided to use the same Q-matrix for all vehicles that have the same goal TAZ.

The learning rate α should ideally be set to a small value, which we suggest to be no larger than $\alpha = 0.003$ so as to avoid excessive instabilities in the Q-values during the time period of a traffic light cycle. However, in practice these relatively small values of α caused excessively slow learning, and values of α as high as 0.5 led to faster learning without significant detected Q-value instabilities.

For this problem we know that there are final states, because a vehicle strives to maximize its rewards but the ultimate goal is to reach its destination. Additionally, all rewards are negative and the second main goal is to optimize the total rewards. Therefore, a value of $\lambda = 1$ is not problematic from a theoretical perspective, and experiments have shown that this value of the discount factor did not cause difficulties during the learning process.

The metrics that were used to evaluate the performance of the learning process and agents' decisions were the instantaneous number of vehicles in the network, and the travel time of the vehicles that left the simulation in the previous Δt seconds. In all the experiments that were conducted, the value $\Delta t = 600\,\text{s} = 10\,\text{min}$ was considered.

Reward functions

Difference reward reflects the negative impact the agent has on the overall system performance, so it includes two components. The first component is the time it took for the vehicle to take action a from state s, similar to the greedy reward function. This is considered a negative impact on overall system performance because the agent does not have an option not to travel, so while having to travel, it should still strive to pick the shortest path. The second component is the additional costs incurred by other vehicles in the network that are associated to the choice of action a; in other words, the negative effects of the agent on other vehicles. This component is the most complex to quantify, since determining the exact value of the negative effects of an agent's decisions can only be done by performing the entire simulation without the considered vehicle; this is not computationally feasible in a useful time frame. The alternative, as was adopted for this model, is to estimate the negative impact by making some simplifications. The negative effects of a vehicle on others is one of the following: slowing down other vehicles in the uncongested flow section, since the vehicle's presence increases vehicle density in the lane; or causing a longer queue, so vehicles in the queue take more time to leave the queue.

As for the first component, it was estimated by determining the vehicle density with and without the vehicle i at lane l'' where the vehicle is, and then the speed of vehicles with and without vehicle i is calculated. The time it would take to travel the length of the lane is then determined for the cases with and without vehicle i, and the difference in travel times is considered to be the negative impact of vehicle i on each individual vehicle that was at lane l'' at the time vehicle i leaves it.

For the second component, all the vehicles in the queue at lane l'' as of the time vehicle i leaves l'' are considered to incur a time delay of T_q, because all those vehicles were forced to leave the queue T_q seconds later than they would if vehicle i was not in that queue. The time penalty was estimated to be $T_q = l_v/v_q$.

Thus, the process to calculate the difference reward is as follows:

$$K_{\mathrm{w}} = \frac{N}{L} \tag{12}$$

$$K_{\mathrm{wo}} = \frac{N-1}{L} \tag{13}$$

$$v_{\mathrm{w}} = \max\left\{v_q, v_0\left(1 - \frac{K_{\mathrm{w}}}{K_{\mathrm{jam}}}\right)\right\} \tag{14}$$

$$v_{\mathrm{wo}} = \max\left\{v_q, v_0\left(1 - \frac{K_{\mathrm{wo}}}{K_{\mathrm{jam}}}\right)\right\} \tag{15}$$

$$t_{\mathrm{w}} = L/v_{\mathrm{w}} \tag{16}$$

$$t_{\mathrm{wo}} = L/v_{\mathrm{wo}} \tag{17}$$

$$\Delta t_{\mathrm{moving}} = t_{\mathrm{w}} - t_{\mathrm{wo}} \tag{18}$$

$$\Delta t_{\mathrm{queue}} = T_q \tag{19}$$

$$r = -t - t_{\mathrm{externalities}} = -t - w\left[(N-1)\Delta t_{\mathrm{moving}} + (Q-1)\Delta t_{\mathrm{queue}}\right] \tag{20}$$

where L is the length of the lane, N is the number of vehicles in the lane, Q is the number of vehicles in the queue, and w is the *greediness factor*, a weighing factor that defines how concerned an agent is with the negative effect it has on other agents; $w = 0$ is equivalent to a purely greedy agent, while $w = 1$ refers to an agent that is very concerned with its impact on the system and with the way its actions will contribute toward the SO. For comparison, the reward function of a greedy agent would be $r = -t$, which is equivalent to Equation 20 with $w = 0$.

RESULTS & DISCUSSION

Comparison with SUMO

Three experiments were conducted to compare DynamiNATOR with SUMO in terms of performance. Each experiment contains three trials, and was conducted as follows:

1. The scenario was simulated with DynamiNATOR, using the Q-learning algorithm with greedy reward. Route files were obtained from DynamiNATOR, and execution time was recorded.
2. Route files obtained from DynamiNATOR were supplied to SUMO, running those routes in a similar scenario. The execution time was recorded.

Table 1 presents the results of these experiments, showing that DynamiNATOR was significantly faster than SUMO in all the experiments, with speedup ratios between 22.52 and 32.51. It can be

concluded that DynamiNATOR meets the goal of high performance, particularly when compared with SUMO.

TABLE 1 Execution times of simulators DynamiNATOR and SUMO for similar scenarios.

Scale	Simulation time (h)	DynamiNATOR (s)	SUMO (s)	Speedup
0.2	1	4.853 ± 0.083	128.045 ± 2.921	26.39
0.1	2	4.310 ± 0.023	140.130 ± 2.683	32.51
0.05	4	5.076 ± 0.007	114.307 ± 2.065	22.52

Reinforcement Learning

To evaluate different assignment strategies, a common scenario was established (Figure 4). This scenario includes a warm-up process where the demand scale is progressively increased over the simulation, allowing agents to gain some insight into the traffic network before loading it with a higher number of vehicles, preventing gridlocks. Warm-up occurs until $t = 100\,\text{h}$, after which point the scale is fixed at 0.45. Higher values of the demand scale caused simulations to diverge due to excessive congestion and simulator limitations.

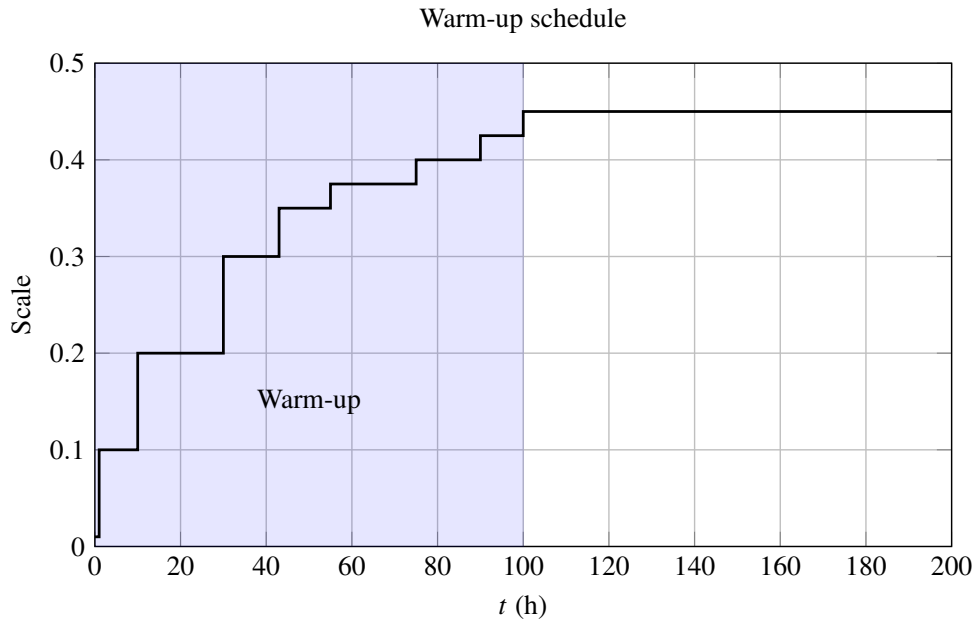

FIGURE 4 Scenario used to evaluate assignment strategies, showing warm-up schedule.

Figure 5 shows the number of vehicles and average travel time of vehicles leaving the simulation. Data points were sampled at in-simulation periods of $p = 600\,\text{s}$, and the average travel time of vehicles leaving the simulation at each sample was calculated from the travel time of vehicles that left the simulation since the last sample, during time period $(t - p, t]$.

As presented in Figure 5, the number of vehicles in the simulation initially shows a step increase as a natural consequence of the warm-up program designed for this experiment. Once the program reaches maximum scale at $t = 100\,\text{h}$, the number of vehicles slowly decreases from 6253 to 5651, a 10% decrease. Another pattern that can be observed is that, as the number of vehicles

in the simulation increases, the slope of the number of vehicles is increasingly negative. This is because vehicles with the same goal share the same Q-matrix, so the more vehicles there are in the network, the more updates to the Q-matrix there are, so agents learn faster.

Average travel time is initially large, reaching 15.23 min, which can be linked to an initial stage where a few Q-value estimates differ significantly from observed rewards, so the agents go through a very rapid adaptation process. After that, average travel time presents a more stable behavior. Travel time increases immediately after the demand scale increases, since agents are using knowledge of a less congested network to navigate a more congested network, and are taking longer paths but also adapting to the new congestion conditions. After the increase in travel time caused by increasing scale, travel time slowly decreases as agents learn better actions to navigate the congested network more efficiently. When the last stage starts, average travel time rises up to 7.785 min, and then slowly decreases until it reaches the final value of 7.031 min.

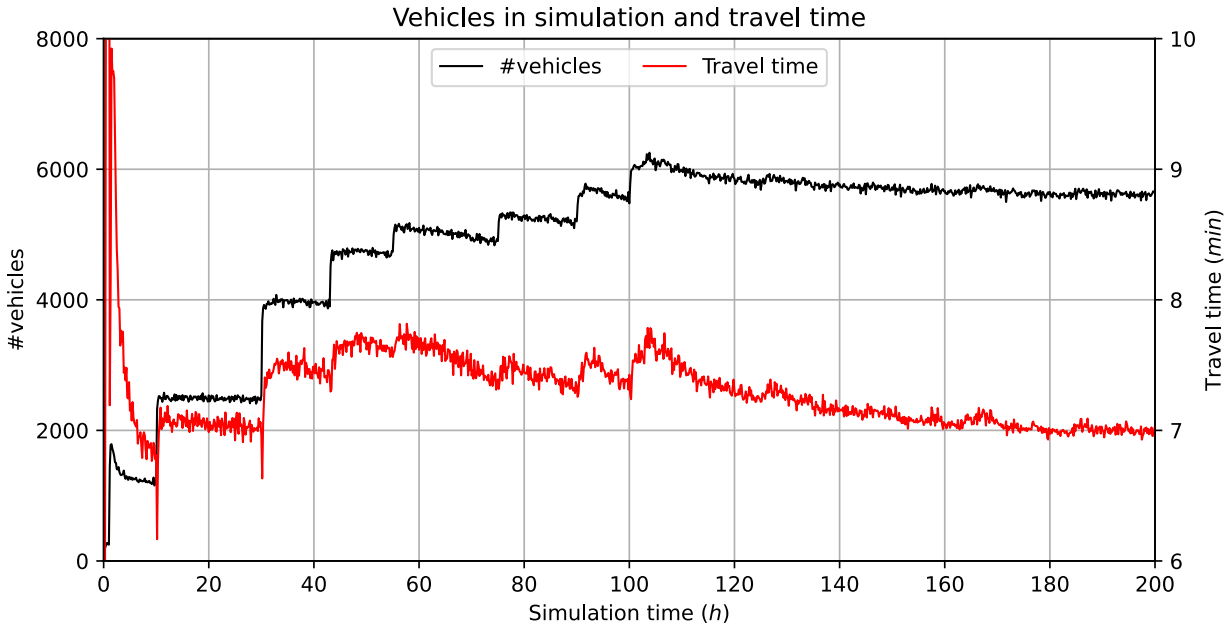

FIGURE 5 Number of vehicles and average travel time of vehicles leaving the simulation.

Both of these results clearly show that the agents are learning, albeit at a slow pace; note that the simulation spans over a period of 200 h. This can be explained by the high level of detail that was chosen for the network, which means vehicles have to take a much larger number of actions to reach their final destinations, so expected future reward values also take longer to propagate to previous actions' Q-values.

Five experiments were then conducted with Q-learners using the difference reward function, for distinct values of the greediness factor $w \in \{0.0, 0.1, 0.5, 0.75, 0.9\}$. The higher the greediness factor corresponding to the reciprocal of greediness (so, the more concerned agents are with the overall system performance), the more learning difficulties agents had at the beginning of the simulation (up to $t = 100$ h), Figure 6. At the later stage $t > 100$ h, the travel time presents a similar trend for all the experiments. The final average travel times in Table 2 were calculated by averaging the travel times of vehicles leaving the simulation in the period $[198, 200]$ h. There is a decreasing trend of the final average travel time as w increases in agreement with the literature (*31*). However,

the relative differences of 0.4% are small when compared to the mean values, and these differences smaller than the values around 1% in literature (*13*) under similar conditions. The SO - UE gap is most noticeable in high-load situations, with the considered scenario presenting medium loads so as to avoid divergent behavior of the model under high loads.

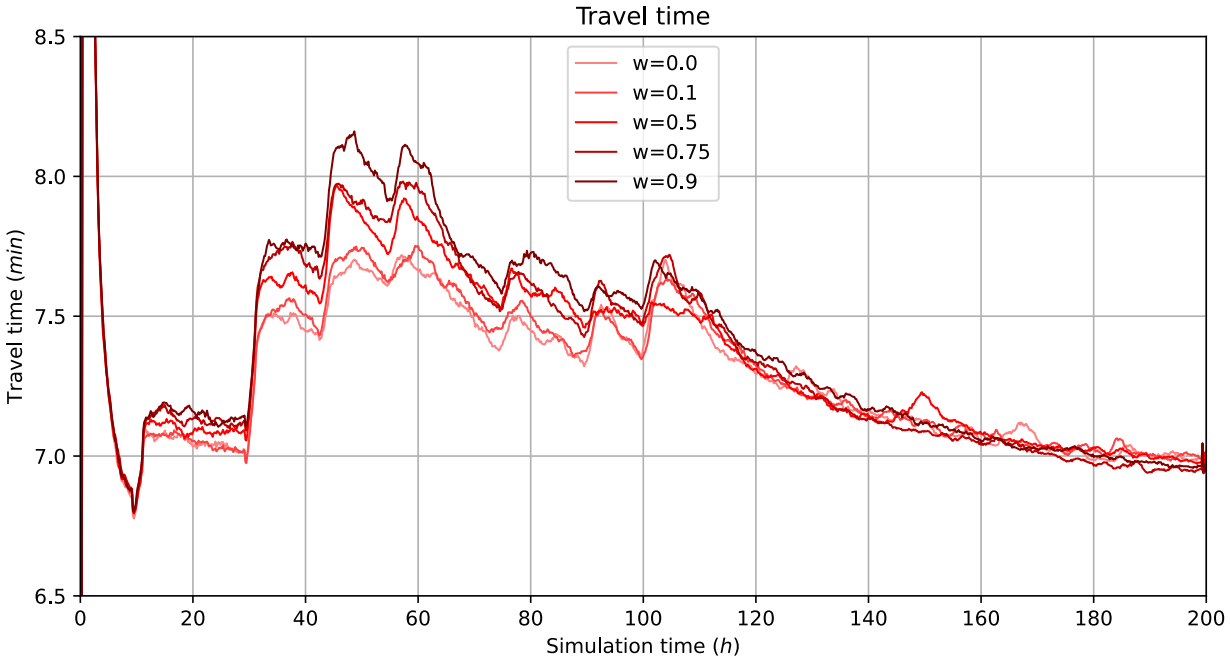

FIGURE 6 Travel time for agents using difference demand with distinct values of greediness factor w.

TABLE 2 Execution time and final travel time for different values of greediness factor w.

w	Execution time (s)	Final average travel time (s)
0.00	4411.5	419.680
0.10	4043.3	419.319
0.50	3576.3	418.512
0.75	3585.1	417.344
0.90	3738.7	417.836

CONCLUSION

The results of this work show that the Q-learning agents were able to learn in a large-scale, complex, and noisy scenario of a large city, and that the employment of difference reward functions with different greediness factors yielded better global cost results than purely greedy agents by a factor of about 0.5% in medium-congestion scenarios. The significance of this result is owed to its cumulative effect over a large number of vehicles, which turns this apparently small relative gain into a large absolute gain in the order of 700 M€/year in the EU28. These results also show that DynamiNATOR is significantly faster than SUMO, meaning the conducted study would only be possible by using DynamiNATOR, as these simulations would require about 27 times more execution time using SUMO.

This work makes contributions at three levels. The scientific contributions are in the form of studying and applying new traffic assignment strategies from a theoretical standpoint by building on existing scientific literature. The deliverables that resulted from this work, including a mesoscopic large-scale traffic model and simulator, and implementations of different assignment strategies, constitute valuable technological contributions. From a more practical perspective, the simulator and assignment strategies can be used as a virtual laboratory so that stakeholders can use them for pure or applied research and to guide public policy on infrastructure. Furthermore, this strategy can be extended in the future as a routing service for drivers, providing directions to individual users that make the overall system more efficient. This vision is increasingly a reality, with the increasing incorporation of technological devices and user interfaces in vehicles that allow users to obtain directions and alternative paths.

Regarding future work, implementing a mechanism to resolve gridlocks automatically would greatly enhance the quality of this virtual laboratory, by allowing simulations under higher traffic demand. A natural extension of this model would be towards multimodal transportation with the inclusion of a diverse mix of vehicles, including most notably trucks, buses, and bikes, as well as pedestrian traffic to simulate the impedance they exert on vehicles when crossing roads. The adopted approach based on ε-greedy Q-learning can be extended to use other techniques, namely with SARSA (*47*) that can be effortlessly implemented in DynamiNATOR, as well as updating Q-values at the end of each vehicle's route so that rewards and expected future rewards can readily propagate to upstream states, and evaluate knowledge-based approaches, for potential improvements in learning speed. The extensible way in which the code was developed also allows other promising distributed multi-agent approaches, such as auctions and negotiation, to be applied in future studies at the scale of a city. Besides, this methodology can be redesigned for better suitability in real-world scenarios, since learning can be modified from a centralized to a distributed process.

ACKNOWLEDGMENTS

The authors would like to thank the Municipality of Porto, for providing project DynamiCITY with their traffic demand estimates, which this work greatly benefited from. This work is a result of project DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions, ref. NORTE-01-0145-FEDER-000073, supported by the Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: Diogo Rodrigues, Rosaldo Rossetti; data collection and processing: Diogo Rodrigues; analysis and interpretation of results: Diogo Rodrigues; draft manuscript preparation: Diogo Rodrigues, Rosaldo Rossetti. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

1. United Nations, *World Urbanization Prospects: The 2018 Revision, Online Edition*, 2018.
2. United Nations, *World Urbanization Prospects 2018: Highlights*. United Nations, Department of Economic and Social Affairs, Population Division, New York, NY, 2019.
3. ACEA, *Vehicles in Use Europe 2022*. Yearly report, European Automobile Manufacturers Association (ACEA), 2022.
4. Eurostat, *Population on 1 January by Age and Sex*, 2023.
5. European Comission, *Impact Assessment - Accompanying Document to the Roadmap to a Single European Transport Area – Towards a Competitive and Resource Efficient Transport System*. Commission Staff Working Paper SEC(2011) 358 final, European Comission, Brussels, Belgium, 2011.
6. Nash, J., Non-Cooperative Games. *The Annals of Mathematics*, Vol. 54, No. 2, 1951, p. 286.
7. Speranza, M. G., Trends in Transportation and Logistics. *European Journal of Operational Research*, Vol. 264, No. 3, 2018, pp. 830–836.
8. Morandi, V., Bridging the User Equilibrium and the System Optimum in Static Traffic Assignment: A Review. *4OR - A Quarterly Journal of Operations Research*, Vol. postprint, 2023.
9. Lujak, M., S. Giordani, and S. Ossowski, Route Guidance: Bridging System and User Optimization in Traffic Assignment. *Neurocomputing*, Vol. 151, 2015, pp. 449–460.
10. Wardrop, J. G., Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineers*, Vol. 1, No. 3, 1952, pp. 325–362.
11. Morandi, V., Bridging the User Equilibrium and the System Optimum in Static Traffic Assignment: How the Cooperation among Drivers Can Solve the Congestion Problem in City Networks. *arXiv: Computer Science > Computer Science and Game Theory*, 2021.
12. Klein, I., N. Levy, and E. Ben-Elia, An Agent-Based Model of the Emergence of Cooperation and a Fair and Stable System Optimum Using ATIS on a Simple Road Network. *Transportation Research Part C: Emerging Technologies*, Vol. 86, 2018, pp. 183–201.
13. Peeta, S. and H. S. Mahmassani, System Optimal and User Equilibrium Time-Dependent Traffic Assignment in Congested Networks. *Annals of Operations Research*, Vol. 60, No. 1, 1995, pp. 81–113.
14. Pigou, A. C., *The Economics of Welfare*. MacMillan, London, UK, 1920.
15. Knight, F. H., Some Fallacies in the Interpretation of Social Cost. *The Quarterly Journal of Economics*, Vol. 38, No. 4, 1924, pp. 582–606.
16. Ramos, G. d. O., B. C. Da Silva, R. Rădulescu, A. L. C. Bazzan, and A. Nowé, Toll-Based Reinforcement Learning for Efficient Equilibria in Route Choice. *The Knowledge Engineering Review*, Vol. 35, 2020, p. e8.
17. Bureau of Public Roads, *Traffic Assignment Manual for Application with a Large, High Speed Computer*. U.S. Department of Commerce, 1964.
18. Yang, H. and X. Zhang, Optimal Toll Design in Second-Best Link-Based Congestion Pricing. *Transportation Research Record*, Vol. 1857, No. 1, 2003, pp. 85–92.
19. de Palma, A., M. Kilani, and R. Lindsey, Comparison of Second-Best and Third-Best Tolling Schemes on a Road Network. *Transportation Research Record*, Vol. 1932, No. 1, 2005, pp. 89–96.

20. Dresner, K. and P. Stone, Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. *New York*, 2004.
21. Vasirani, M., *Vehicle-Centric Coordination for Urban Road Traffic Management: A Market-Based Multiagent Approach*. Ph.D. thesis, Rey Juan Carlos University, Madrid, 2009.
22. Vasirani, M. and S. Ossowski, A Market-Inspired Approach for Intersection Management in Urban Road Traffic Networks. *Journal of Artificial Intelligence Research*, Vol. 43, 2012, pp. 621–659.
23. Schepperle, H. and K. Böhm, Agent-Based Traffic Control Using Auctions. In *Cooperative Information Agents XI* (J. G. Carbonell, J. Siekmann, M. Klusch, K. Hindriks, M. P. Papazoglou, and L. Sterling, eds.), Springer, Delft, The Netherlands, 2007, Vol. 4676 of *Lecture Notes in Artificial Intelligence*, pp. 119–133.
24. Schepperle, H., K. Böhm, and S. Forster, Traffic Management Based on Negotiations between Vehicles – A Feasibility Demonstration Using Agents. In *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis* (W. van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, J. Collins, P. Faratin, S. Parsons, J. A. Rodriguez-Aguilar, N. M. Sadeh, O. Shehory, and E. Sklar, eds.), Springer, Berlin, 2008, Vol. 13 of *Lecture Notes in Business Information Processing*, pp. 90–104.
25. Rahimi-Farahani, H., A. A. Rassafi, and B. Mirbaha, Forced-node Route Guidance System: Incorporating Both User Equilibrium and System Optimal Benefits. *IET Intelligent Transport Systems*, Vol. 13, No. 12, 2019, pp. 1851–1859.
26. Elimadi, M., A. Abbas-Turki, and A. Koukam, A Novel Approach for Dynamic Traffic Assignment Based on Multi-Agent Node Reservation: Comparative Study on Two Competing Roads. *Procedia Computer Science*, Vol. 191, 2021, pp. 320–327.
27. Watkins, C. J. C. H., *Learning from Delayed Rewards*. Ph.D. thesis, King's College, Cambridge, UK, 1989.
28. Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2nd ed., 2020.
29. Tumer, K. and D. Wolpert, A Survey of Collectives. In *Collectives and the Design of Complex Systems* (K. Tumer and D. Wolpert, eds.), Springer, New York, NY, 2004, pp. 1–42.
30. Tumer, K. and A. Agogino, Distributed Agent-Based Air Traffic Flow Management. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, Honolulu, HI, 2007, AAMAS'07, pp. 342–349.
31. Grunitzki, R., G. d. O. Ramos, and A. L. C. Bazzan, Individual versus Difference Rewards on Reinforcement Learning for Route Choice. In *2014 Brazilian Conference on Intelligent Systems*, IEEE, São Paulo, Brazil, 2014, pp. 253–258.
32. Tumer, K., Z. T. Welch, and A. Agogino, Aligning Social Welfare and Agent Preferences to Alleviate Traffic Congestion. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (L. Padgham, D. Parkes, J. Müller, and S. Parsons, eds.), International Foundation for Autonomous Agents and Multiagent Systems, Estoril, Portugal, 2008, Vol. 2 of *AAMAS'08*, pp. 655–662.
33. OpenStreetMap contributors, *OpenStreetMap*, 2023.
34. Scholz, I. and D. Stöcker, *JOSM V18746*, 2023.

35. Lopez, P. A., E. Wiessner, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, and P. Wagner, *Microscopic Traffic Simulation using SUMO*. Maui, HI, 2018.
36. PTV, *VISUM*. Planung Transport Verkehr GmbH, Karlsruhe, Germany, 1992.
37. PTV, *VISUM User Manual Version 7.50*. Karlsruhe, Germany, 2001.
38. Rodrigues, D. M. F., *DynamiNATOR v0.1.0*. DynamiCITY, Porto, Portugal, 2023.
39. German Aerospace Center, *SUMO*. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany, 2023.
40. German Aerospace Center, *SUMO - PlainXML*. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany, 2023.
41. German Aerospace Center, *SUMO Road Networks*. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany, 2023.
42. Microsoft and Vexcel Imaging, *Bing Aerial Imagery*, 2023.
43. Transportation Research Board, *Highway Capacity Manual*. Transportation Research Board, National Research Council, Washington, DC, 2000.
44. Ashalatha, R. and S. Chandra, Critical Gap through Clearing Behavior of Drivers at Unsignalised Intersections. *KSCE Journal of Civil Engineering*, Vol. 15, No. 8, 2011, pp. 1427–1434.
45. Greenshields, B. D., A Study of Traffic Capacity. *Highway Research Board*, Vol. 14, 1935, pp. 448–477.
46. Sheffi, Y., *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
47. Rummery, G. A. and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems*. Engineering Department - Cambridge University, Cambridge, UK, 1994.