# Individual Versus Difference Rewards on Reinforcement Learning for Route Choice

Ricardo Grunitzki, Gabriel de O. Ramos, Ana L. C. Bazzan
Instituto de Informática/PPGC
UFRGS, Brazil.
Email: {rgrunitzki, goramos, bazzan}@inf.ufrgs.br

***Abstract*—In transportation systems, drivers usually choose their routes based on their own knowledge about the network. Such a knowledge is obtained from drivers' previous trips. When drivers are faced with jams they may change their routes to take a faster path. But this re-routing may not be a good choice because other drivers can proceed in the same way. Furthermore, such behavior can create jams in other links. On the other hand, if drivers build their routes aiming at maximizing the overall travel time (system's utility), rather than their individual travel time (agents' utility), the whole system may benefit. This work presents two reinforcement learning algorithms for solving the route choice problem in road networks. The I$Q$-learning uses an individual reward function, which aims at finding a policy that maximizes the agents' utility. On the other hand, D$Q$-learning algorithm shapes the agents' reward based on difference rewards function, and aims at finding a route that maximizes the system's utility. Through experiments we show that D$Q$-learning is able to reduce the overall travel time when compared to other methods.**

## I. Introduction

Traffic and mobility present challenging issues to authorities, traffic engineers, and researchers. This is due to the demand for mobility in societies that has grown more than the investments in infrastructure (supply). To deal with the increasing demand, employing techniques and methods to optimize the use of existing road networks is attractive, since these methods mitigate the need for expensive and environmental-impacting changes on the infrastructure.

Traffic assignment (or, alternatively, route assignment or trip assignment) is largely used for traffic modeling in transportation system. Traffic assignment methods select routes and attribute them to travelers aiming at connecting their individual origins with their individual destinations. The output describes the state of a transportation system, which is a relevant input for evaluating the consequences of changes in the infrastructure.

There are many works presenting traffic assignment approaches to model the interaction between the traffic system and drivers decision making. However, approaches such as [1], [2], depend on a central authority to assign a specific route for each driver and do not consider the behavior of the drivers along the trips. The main challenging in modeling and controlling transportation systems is the difficulty (if not impossibility) of a system manager to directly control the behavior of the drivers in terms of their route choices.

In this work, we apply multiagent reinforcement learning to simulate the effects of the agents' route choice based on two reward functions, and apply them using the reinforcement learning (RL) algorithm $Q$-learning. The first model, here called I$Q$-learning (Individual $Q$-learning), shapes the agents' reward based on partial environment perception (current link), and aims at maximizing the individual agents' utility. The second model, D$Q$-learning (Difference $Q$-learning), uses the difference rewards function, proposed by [3], for shaping the agents' reward. This reward function provides a reward that is aligned with the overall system's reward, thus allowing agents to maximize the global system's reward instead of individual agents' reward.

In this paper, agents make decisions dynamically using their experience, which is acquired through their interaction with the environment. The environment is a road network that abstracts some real-world characteristics, such as vehicle movement along roads, allowing us to focus on the main subject, which is to discover the route choice mechanism that maximizes the system's utility (in this case the overall travel time). Results show that the D$Q$-learning improves the travel time in all cases evaluated in this work, when compared against I$Q$-learning, successive averages, incremental assignment, and all-or-nothing assignment. Comparing D$Q$-learning and I$Q$-learning we show that D$Q$-learning has better performance because the reward alignment to the system's utility provided by difference rewards function can distribute the agents over the network in a better manner than I$Q$-learning.

In the next section we review the basics concepts of road traffic modeling (Sect. II-A), reinforcement learning (Sect. II-B), and difference rewards functions (Sect. II-C). Section III presents the related work. Section IV presents our reinforcement learning approach (Sect. IV-A), reward functions (Sect. IV-B), and route choice mechanism (Sect. IV-C). Section V presents the scenario where the approaches are evaluated, the parameters choice and implementation, and discusses the results. Section VI concludes the paper and presents opportunities for further investigation.

## II. Background

### A. Road Traffic Modeling

A road network can be modeled as a graph, with a set of nodes $n \in N$ that represent intersections, and set of links $j \in J$ between these nodes, which represent the roads sections. The weight of a link represents a form of cost associated with the link. For instance, the cost can be travel time, fuel spent, or length.

The road network contains origins and destinations nodes that are subsets of the set of nodes (OD-pairs). A driver's trip consists of a set of links, forming a route between his origin and destination nodes. In a commuting scenario there are repeated trips, e.g., as drivers going from home to work approximately in the same hour of the day during the workdays.

Drivers traveling through the road network generate traffic flow. The simulation of this traffic flow can be done in macroscopic models, which are faster to calculate and run, besides abstracting elements not relevant for this problem (lanes changes, heterogeneous vehicles'

978-1-4799-5618-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BRACIS.2014.53

position and etc.). These models need a suitable cost function that relates traffic flow and link's attributes (such as capacity) to free-flow travel time. In this work, we use a function proposed by [4, Exercise 10.1]:

$$t_j(v) = f_j + 0.02 * v \quad (1)$$

where, $t_j$ is the travel time on link $j$ applied to the number of vehicles $(v)$, and $f_j$ is the free-flow travel time on link $j$. This abstract function increases $t_j$ by 0.02 of a minute for each vehicle/hour of flow.

### B. Reinforcement Learning

Reinforcement learning deals with the problem of making an agent learn a behavior by interacting with the environment. Usually, the RL problem is modeled as a Markov Decision Process (MDP), which is composed by a set of environment states $S$, a set of actions $A$, a state transition function $T : S \times A \to \Pi(S)$, where $\Pi(S)$ is a probability distribution over $S$, and a reward function $R : S \times A \to \Re$ that returns the reward $R(s, a)$ after a given action $a \in A$ has been taken in state $s \in S$. The functions $T$ and $R$ can be defined as the environment model.

The learning job in an MDP is to find a policy $\pi : S \to A$ that maximizes the agent's future reward. To maximize the reward received throughout all interactions, the agent must select each action according to a strategy that balances exploration (gain of knowledge) and exploitation (use of knowledge). At each state the agent must choose whether to exploit an already known action, or explore the new alternative and potentially discover a more beneficial action. When exploiting, the agent must occasionally take random actions to learn their reward, and potentially discover if they lead to states of higher reward. According to [5], the balancing of exploration and exploitation is a key design decision when implementing a RL solution. One known exploration strategy is $\epsilon$-greedy, which consist in choosing random actions (exploration) with probability $\epsilon = [0, 1]$ or choosing best actions (exploitation) with probability $1 - \epsilon$. This way, in the beginning, $\epsilon$ has a high exploration, and decreases exponentially for each episode $\lambda \in \Lambda$ according to Eq. 2, where $\Delta\epsilon$ represents the epsilon decay rate in the interval $[0, 1]$.

$$\epsilon_\lambda = \epsilon * (\Delta\epsilon)^\lambda \quad (2)$$

$Q$-learning is an algorithm that converges towards the optimal policy, given certain conditions [6]. Its update rule is shown in Eq. 3, where $< s, a, s', R >$ is an experience tuple, meaning that the agent performed action $a$ in state $s$, reaching $s'$, and receiving reward $R$. Action $a'$ is one of the possible actions on $s'$, $\alpha \in (0, 1]$ is the learning rate, and $\gamma \in (0, 1]$ is the discount factor. $Q(s, a)$ is an entry indexed by state $s$ and action $a$ in the $Q$-table, which stores the values (called $Q$-values) of each state-action pair. The $Q$-value $Q(s, a)$ is the expected discounted reward for executing action $a$ at state $s$ and following the policy $\pi$ thereafter. A complete description of $Q$-learning can be found in [6].

$$Q(s, a) \to (1 - \alpha) Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q\left(s', a'\right) \right] \quad (3)$$

### C. Difference Rewards

The right selection of the reward structure for the learning agents is an important step during the development of reinforcement learning mechanisms. Most direct approaches let each agent receive the system performance as its reward. However, according to [3], in many domains such reward structure leads to slow learning. Given that reinforcement learning agents aim to maximize their own rewards, a critical task is to create "good" agents' rewards, or rewards that when pursued by the agents lead to good overall system performance. System's utility is a function $G(z)$ that rates the performance of the full system, where $z$ depends on the movement of all agents.

Difference rewards are functions proposed by [3] which aim to provide a reward that is both sensitive to the agents' actions and aligned with the overall system reward. Consider the difference rewards as in Eq. 4, where $z_i$ is the action of the agent $i$. All components of $z$ that are affected by agent $i$ are replaced with the fixed constant $c_i$.

$$D_i \equiv G(z) - G(z - z_i + c_i) \quad (4)$$

Using a $null$ vector in $c_i$ is equivalent to taking agent $i$ out of the system. Intuitively this causes the second term of the difference rewards to evaluate the performance of the system without $i$ and, therefore, D evaluates the agent's contribution to the system's performance. According to [3], there are two advantages using Eq. 4 for $D$. First, the second term removes a significant portion of the impact of the other agents in the system and provides an agent with a "cleaner" signal than $G$. This benefit has been dubbed "learnability" in previous work [7], [8]. Second, because the second term does not depend on the actions of agent $i$, any action by agent $i$ that improves $D_i$, also improves $G$. In other words, any action taken by agent $i$ that improves $D$, also improves $G$. This term measures the amount of alignment between two rewards has been dubbed "factoredness" in previous work [7], [8].

Difference rewards can be applied to any linear or nonlinear system's utility function. However, its effectiveness depends on the domain, and on the interaction among the agents' utility function [3].

## III. RELATED WORK

Several approaches have been proposed to solve route choice problems, many of them considering abstract traffic scenarios. In [9], a two-route scenario was studied to test the effect of real-time information sharing in drivers' behavior. Such approach uses reinforcement learning to reproduce human decision-making in a commuting scenario with 45 agents. The results present relevant aspects of the agents' decision-making process, even though only binary route choice scenarios were studied. In the present work, the interest is in the evaluation of the route choice problem in a more complex network, and with a higher number of agents than in [9]. In addition, instead of using a predefined number of routes, in this paper the routes are built dynamically along the episodes, i.e., during the trip (en-route learning).

In [10] the authors applied multiagent reinforcement learning and difference rewards functions to alleviate the traffic congestion. They proposed two reward functions, the first one aiming to maximize the individual agents' reward, and the second aiming to maximize the system's utility. Although the results shown that the second approach is more efficient, the authors applied their algorithms to a stateless abstract scenario, where there is only one OD-pair, and nine independent routes. That problem is easier to be solved because there is no shared link among routes, and the route choice process does not change during the trip. Furthermore, the impact of an increased number of agents was left aside.

Multiagent reinforcement learning was applied for the route choice problem in [11]. The authors use a re-routing mechanism that

enables agents to change their routes dynamically during the trip. The authors conclude that this approach is useful to compensate the lack of adaptivity regarding traffic management. They use 700 learning agents in the problem to evaluate the approach in an abstract traffic network in format of a $6 \times 6$ grid. The reinforcement learning algorithm used was the $Q$-learning.

In addition to the approaches based on reinforcement learning algorithms cited before, there are centralized traffic assignment approaches for assigning routes to vehicles [1], [2]. The most relevant approaches are presented below. For an extensive explanation, please refer to [4, Chapter 10], as shown next.

A simplest traffic assignment method, called all-or-nothing assignment, is presented by [4, Section 10.3]. This method assigns all trips to the route with the minimum cost. It assumes that there are no congestion effects, that all drivers consider the same attributes for route choice and that they perceive and weigh them in the same way. No feedback loop is assumed between route selection and costs. These assumptions are reasonable in sparse and uncongested networks where there are few alternative routes and they are very different in cost. If we assume congested networks this approach may not be a good choice. For this kind of problem, traffic assignment approaches are proposed, as shown next.

An incremental assignment method is presented by [4, Section 10.5.3]. In this process, fractions of traffic volume are assigned in steps. At each step, each link's travel time is recalculated based on its occupation. When many steps are performed, the flows may represent an equilibrium assignment. However, this method may not yield an equilibrium solution.

Iterative algorithms as the method of successive averages [4, Section 10.5.4] were developed, at least partially, to overcome the problem of allocating too much traffic to low-capacity links. The algorithms find an equilibrium assignment assuming infinite iterations. The load is determined using a repeated all-or-nothing assignment in which costs are determined anew by mean trip distributions over all previous iterations. The convergence may be very slow.

An approach which incrementally loads the traffic demand onto the network, and updates the traffic conditions dynamically is proposed by [1]. The travel time depends of the shortest path algorithms which determines the path with minimum cost for each OD-pair. This method is applied to solve the dynamic solution of the problem incrementally. However, the method depends on a central authority assigning the route for each driver, without the driver learning or adapting to the problem.

A genetic algorithm is proposed in [2] for solving the traffic assignment problem. In this approach, each vehicle has $k$ available routes generated by the Dijkstra algorithm. The vehicles' route are encoded as chromosomes in the algorithm genetic population. A drawback in this approach is that vehicles are not able to change their route during the trip. However, such approach is able to solve problems with millions of trips and thousands nodes and links.

## IV. APPROACH AND SCENARIO

### A. Reinforcement Learning Algorithm

In the present work, the agents (vehicles/drivers) are implemented as independent learners, which ignore the existence of other agents. In other words, joint actions are not modeled in their MDP. This is needed because, in scenarios such as transportation systems, there is a large number of agents interacting with the environment, and the use of joint action learners is infeasible, as remarked in [12].

The MDP model for this problem is as follows. An agent's state is the node at which it is located. The set of the actions are the outbound links from the nodes of the network. In this approach, the agents do not have a set of predefined routes - from their origin to destination -, the route is built dynamically during the episodes' steps based on their MDP (as described in Sect. IV-C). We refer to this approach as en-route learning hereafter. The reinforcement learning algorithm used is the $Q$-learning [6], and the exploration/exploitation policy used is $\epsilon$-greedy, with $\epsilon$ decreasing exponentially.

Here we present two reinforcement learning approaches to solve the route choice problem. Both of them, I$Q$-learning and D$Q$-learning, use the same configuration. However the reward functions are not the same. The I$Q$-learning uses the individual reward function (described in Sect. IV-B1) whereas the D$Q$-learning uses the difference rewards function (described in Sect. IV-B2).

### B. Reward Functions

This section presents the functions used in this work for shaping the reinforcement learners' reward.

*1) Individual Reward Function:* This reward function is used by the I$Q$-learning and is given in Eq. 5, where $t_j$ is the travel time function given in Eq. 1 applied to the number of vehicles $v_j$ on link $j$. The reward increases as travel time decreases, thus drivers will strive to maximize their individual travel times.

$$R = -t_j(v_j) \tag{5}$$

In this work, we call this reward function individual reward because it is based just on the agents' current state, which represents the system portion where the agents have perception (considering they are not receiving information from somebody).

*2) Difference Rewards Function:* The difference rewards function is calculated as in Eq. 4. In this case, $G(z)$ is calculated by Eq. 6. $G(z_{-i})$ is calculated by Eq. 7, and $j_i$ represents the link where the agent $i$ is. Here, the term $c_i$ from Eq. 4 is removed because we used a "null" value for it.

$$G(z) = \sum_{j \in J} -t_j(v_j) \tag{6}$$

$$G(z - z_i) = \sum_{j \in J \setminus i} -t_j(v_j) \tag{7}$$

This way, the reward function is given by $R = D_i$, which is calculated based on the influence of the agent $i$ in the system's utility $G(z)$. In other words, it is analogous to agents knowing their actions' influence over the other agents of the system.

### C. Route Choice Mechanism

In this approach, the agents do not choose one path from their origin to their destination. The route is built dynamically along their interactions with the environment. Compared to implementations like [9], [10], [13], which use a pre-established number of precomputed routes, this problem is harder to be solved. In this work, the number of valid routes for one OD-pair grows with the size of the network, and if we consider that the agents can drive in loops, the number of

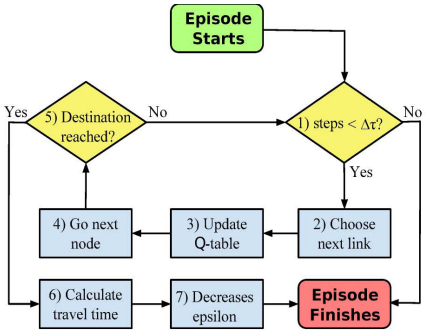


Fig. 1. RL mechanism for route choice flowchart.

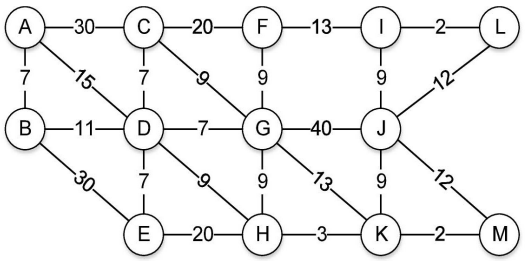


Fig. 2. Abstract road network topology. Adapted from [4, Exercise 10.1].

paths may be infinite. To avoid infinite paths, the number of episode's steps is limited by $\Delta\tau$.

During the simulation, in the initialization process, drivers are created according to the OD-pairs (one driver for each trip), and simulation's parameters are set. After that, for each episode $\lambda$, the drivers' route are built following the flow presented in Fig. 1. When an episode starts, all agents are placed in their origins. After the termination criterion in condition 1, each agent chooses an outbound link (process 2) following the $\epsilon$-greedy policy. At process 3, the agents update their $Q$-table following the reward function used by their RL algorithm (I$Q$-learning or D$Q$-learning). At process 4, agents travel the chosen link and verify if their current node is the destination (condition 5). In the case that this is false they return to condition 1, otherwise, the agent's travel time is calculated on process 6 according to (8). In this equation, each agent $i$ calculates its travel time $t_j$ experienced on its route $P_i$, where $t_j$ is the travel time function given in (1) applied to the number of vehicles on link $j\,(v_j)$. Finally, the $\epsilon_\lambda$ parameter is updated on process 7 according to (2).

$$\widehat{t}_{P_i} = \sum_{j \in P_i} t_j\,(v_j) \tag{8}$$

## V. RESULTS AND DISCUSSIONS

### A. Abstract Network

The scenario used in this work is presented in Fig. 2. This abstract scenario is an adaptation of the Exercise 10.1 presented in [4]. This scenario contains 13 nodes and 24 links. All links are two-way, where the value over the links are their free-flow travel times. In this network we have 2 origins (A and B), representing two residential areas, and two destinations (L and M), representing two major shopping centres. As in [4, Exercise 10.1], there are 1700 vehicles distributed in 4 OD-pair, as shown in Table I.

Free-flow travel times were modified in links AC, BE, CF, EH and GJ (and their opposite direction as well) from the original network to make the problem harder to be solved. Turning these links less attractive drivers need to distribute in a better manner to reduce their travel cost. This is a complex scenario from the point of view of route choice, since the routes for each OD-pair may include loops.

### B. Parameters and Implementation

Before testing the effectiveness of the I$Q$-learning and D$Q$-learning, we ran simulations in the scenario described in Sect. V-A

TABLE I. NUMBER OF TRIPS PER OD-PAIR.

| Origin | Destination | Trips |
|---|---|---|
| A | L | 600 |
| A | M | 400 |
| B | L | 300 |
| B | M | 400 |

for defining parameters $\alpha$, $\gamma$ and $\epsilon$. In order to evaluate our solutions and to compare them against other methods, we measured the average travel time of all the drivers (overall travel time), and the average travel time of all the drivers with the same OD-pair (average travel time per OD-pair).

The choice of parameters $\alpha$ and $\gamma$ was performed individually for both approaches. Empirically, the set of parameters which achieved better results was $\alpha = 0.8$ and $\gamma = 0.9$ in both cases. In this problem domain the reinforcement learning algorithm is most sensitive to discount factor than to the learning rate. For low values of $\gamma$, drivers can not learn a good route to their destination. It occurs because in traffic scenarios, where the next choice option depends on the current, it is relevant to take into account the outcomes of the subsequent choices, otherwise, the next available possibilities will result in a bad overall performance. The learning rate does not have too much impact on the results.

For the $\epsilon$-greedy policy we defined a value for $\Delta\epsilon$ that enables the agents to explore the environment with a higher probability during the initial episodes, and to exploit with a higher probability in the final episodes. By defining $\Delta\epsilon = 0.92$ and $\epsilon = 1.0$, we obtain a good balancing.

The number of episodes and the maximum number of steps per episode ($\Delta\tau$) were empirically set to 150 and 100, respectively. This way, the agents start with an exploration rate of 100%, decreasing until the end of the simulation, where it will have an exploration rate $\approx 0\%$. The exploration curve along the episodes is shown in Fig. 5.

### C. Comparison of Methods: Overall Travel Time

To compare I$Q$-learning and D$Q$-learning, we tested the approaches in the abstract scenario described in Sect. V-A. Each simulation was run 30 times. For the sake of comparison, we also present results for the incremental assignment and successive averages methods described in Sect. III, in addition to the all-or-nothing assignment.

In the following experiments we applied the proposed approaches in the learning agents. In each case, the agents must learn the best

TABLE II. TRAVEL TIMES IN MINUTES AND STANDARD DEVIATION PER OD-PAIR FOR THE COMPARED METHODS.

| | | Method | | | | |
|---|---|---|---|---|---|---|
| OD-pair | Trips | D$Q$-learning | I$Q$-learning | Inc. Assignment | Suc. Averages | All-or-nothing |
| AL | 600 | 79.07 (0.27) | 81.25 (0.23) | 96.20 | 101.20 | 138.00 |
| AM | 400 | 64.41 (0.31) | 67.86 (0.35) | 83.80 | 86.36 | 97.00 |
| BL | 300 | 76.43 (0.44) | 78.56 (0.25) | 93.40 | 97.72 | 128.00 |
| BM | 400 | 61.80 (0.33) | 64.32 (0.40) | 79.80 | 82.88 | 87.00 |
| Overall | 1700 | 71.09 (0.13) | 73.64 (0.23) | 88.93 | 92.78 | 114.59 |

route for their OD-pair, that is the fastest route. In the I$Q$-learning the agent's reward is aligned to the agent's utility. It means that a learned policy may be attractive to the agent but not for the system's utility. On the other hand, in the D$Q$-learning method the agents' reward is aligned to the system's utility. In short, we intend to discover if these reward alignment implies in the quality of the solutions reached by $Q$-learning algorithm, in a dynamic route choice problem, with infinite possible routes and many agents.

The results presented in Table II show that D$Q$-learning yielded better results for all OD-pairs compared to the other methods. The I$Q$-learning was better than incremental assignment, successive averages and all-or-nothing assignment for all OD-pairs. The incremental assignment method was better than successive averages and all-or-nothing assignment. Obviously, the all-or-nothing assignment is not the best method because in this scenario congestion arise in some links due to too many drivers preferences for them. However, it can be considered as a baseline for comparing against other methods.

Compared with other methods, $Q$-learning-based approaches present results significantly better. In addition, they are independent, and drivers do not depend on a central authority assigning their routes, as occurs in successive averages, incremental assignment, and all-or-nothing methods. In RL approaches, the agents are able to adapt their routes dynamically according to the network conditions, whereas in the other methods drivers' route is known to the traffic assigner, which interfere on agents' privacy. Furthermore, the implementation in real world implies in communication between agents and the central authority, which is not trivial.

### *D. Individual Versus Difference Rewards: Convergence Time*

In order to understand the effects of the reward alignment along the simulation, for the reinforcement learning approaches, we also evaluated the convergence time by means of the overall travel time. Fig. 3 shows the convergence time and standard deviation for 30 runs. Results show that the convergence times in main plot are very similar; However, the inset plot shows that D$Q$-learning is able to reach better solutions than I$Q$-learning. More specifically, from episode $\approx 75$ onwards (the point at which the exploration ends) I$Q$-learning stops converging to better solutions, whereas D$Q$-learning keeps improving the overall travel time.

In order to show the improvement provided by D$Q$-learning over I$Q$-learning is statistically significant, we present the following comparison test. Assuming the results reached by D$Q$-learning and I$Q$-learning in Fig. 3 at each point are independents, its values, $q_d$ and $q_i$, respectively can be defined by Gaussian distributions. Thus, the difference $d = |q_d - q_i|$ is also defined as a normal distribution. Fig. 4 shows the confidence interval of $d$. In cases where the confidence interval does not contain the zero value it means that the results are significantly different (considering a confidence level of $95\%$) and consequently D$Q$-learning overcame I$Q$-learning at those points (episode). As we can see in Fig. 4, the significant results reached by D$Q$-learning are more prevalent after the end of exploration.

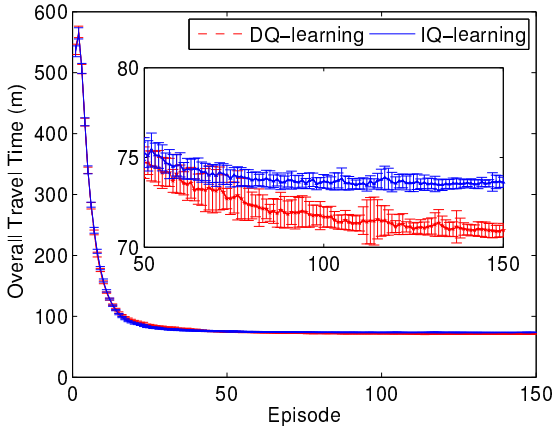


Fig. 3. Convergence times and standard deviation along the episodes (main) and in final episodes (inset).

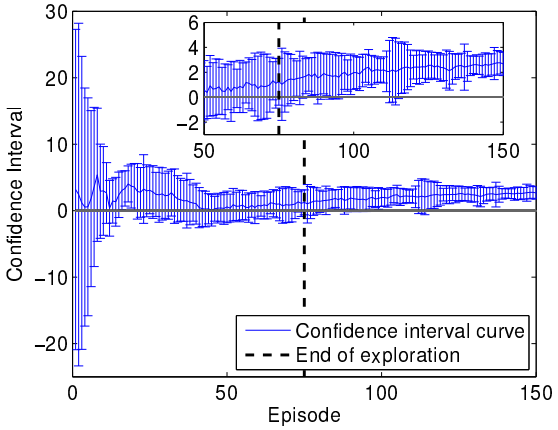


Fig. 4. Confidence interval of $d$ along the episodes (main) and in final episodes (inset).

Assuming that in RL methods the learning process is iterative and needs a significant number of episodes for the exploration period, to consider these initial episodes to define the methods' performance may not be a good choice. Thus, for the interval confidence presented in Fig. 4, we evaluated for groups of 5 episodes the percentage times in which D$Q$-learning is statistically better than I$Q$-learning along the episodes, as shown in Fig. 5. In the initial episodes nothing can be concluded about the algorithms performance because the agents' have a high probability to behave randomly. However, after the end of exploration D$Q$-learning overcomes I$Q$-learning more often. As we can observe, at the final episodes the algorithms' performance

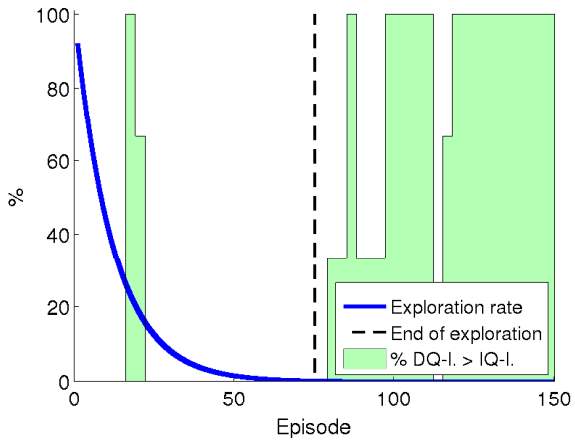


Fig. 5. Percentage time for groups of 5 episodes in which D$Q$-learning overcome I$Q$-learning.

stabilizes. It means that the final learned policies by D$Q$-learning improve the system's performance when compared to I$Q$-learning.

This experiment showed that both methods presented equivalent performance during the exploration period in the evaluated scenario. It occurs because $Q$-learning handles very well this kind of domain. Furthermore, during the exploration period, drivers' may behave randomly. However, after the exploitation period, D$Q$-learning reaches results statistically better than I$Q$-learning. The performance improvement provided by the difference rewards is equivalent to an average reduction of $\cong 3.46\%$ in travel time of all trips. It is equivalent to a reduction of 2 minutes 44 seconds in drivers' travel time.

## VI. CONCLUSIONS

In this work we used a difference rewards function to shape the agents' reward and applied it on $Q$-learning algorithm to solve the route choice problem. Here we call this approach D$Q$-learning. The results obtained were compared against I$Q$-learning, in which the reward function is based on agent's utility. When the difference function is applied to $Q$-learning, the learning process finds a policy that maximizes the system's utility, as opposed to I$Q$-learning, which aims to maximize the agent's utility.

In the experiments we evaluate the proposed approaches in a route choice problem and compared them with three traffic assignment methods: all-or-nothing assignment, successive averages and incremental assignment. In the abstract scenario tested, the D$Q$-learning was $3.46\%$ better than I$Q$-learning. Compared to incremental assignment, successive averages, and all-or-nothing assignment, the D$Q$-learning was between $20\%$ and $40\%$ better. For this same experiment we evaluated the convergence time for $Q$-learning approaches. Results showed that D$Q$-learning overcomes I$Q$-learning in convergence time because the reward alignment to the system's utility provided by difference rewards is able to keep improving the agents' solution along the episodes, whereas I$Q$-learning stops improving agents' solution at the beginning of exploitation period.

Shaping agents' reward with difference utility function makes all agents with a common objective, that is to maximize the system's utility. Even though indirectly, in D$Q$-learning there is cooperation among the agents to improve the system's performance, whereas in I$Q$-learning each agent behaves to improve its own performance. Despite this, I$Q$-learning reached better solutions when compared to other methods.

In a future work, we shall consider evaluating the effect of heterogeneous learners (different learning algorithms) interacting in the same environment. Moreover, due to the large number of episodes needed for the route choice mechanism, knowledge-based algorithms may be applied to speed up the learning process using the two reward functions in the same agent.

## ACKNOWLEDGMENTS

Ricardo Grunitzki and Ana L. C. Bazzan are partially supported by CNPq and FAPERGS. All three authors are partially supported by ITL. Jorge L. A. Samatelo has helped in data analysis.

## REFERENCES

[1] C. Tong and S. Wong, "A predictive dynamic traffic assignment model in congested capacity-constrained road networks," *Transportation Research Part B: Methodological*, vol. 34, no. 8, pp. 625 – 644, 2000.

[2] L. S. Buriol, M. J. Hirsh, P. M. Pardalos, T. Querido, M. G. Resende, and M. Ritt, "A biased random-key genetic algorithm for road congestion minimization," *Optimization Letters*, vol. 4, pp. 619–633, 2010.

[3] K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2007, pp. 1–8.

[4] J. Ortúzar and L. G. Willumsen, *Modelling Transport*, 3rd ed. John Wiley & Sons, 2001.

[5] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[6] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[7] A. Agogino and K. Tumer, "Multi–agent reward analysis for learning in noisy domains," in *AAMAS '05: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, Eds., vol. II. New York, NY: ACM Computing Press, July 2005, pp. 81–88.

[8] K. Tumer and D. Wolpert, "A survey of collectives," in *Collectives and the Design of Complex Systems*, K. Tumer and D. Wolpert, Eds. Springer, 2004, pp. 1–42.

[9] E. Ben-Elia and Y. Shiftan, "Which road do I take? A learning-based model of route-choice behavior with real-time information," *Transportation Research Part A: Policy and Practice*, vol. 44, no. 4, pp. 249–264, 2010.

[10] K. Tumer, Z. T. Welch, and A. Agogino, "Aligning social welfare and agent preferences to alleviate traffic congestion," in *Proceedings of the 7th Int. Conference on Autonomous Agents and Multiagent Systems*, L. Padgham, D. Parkes, J. Müller, and S. Parsons, Eds. Estoril: IFAAMAS, May 2008, pp. 655–662.

[11] A. L. C. Bazzan and F. Klügl, "Re-routing agents in an abstract traffic scenario," in *Advances in artificial intelligence*, ser. Lecture Notes in Artificial Intelligence, G. Zaverucha and A. L. da Costa, Eds., no. 5249. Berlin: Springer-Verlag, 2008, pp. 63–72. [Online]. Available: www.inf.ufrgs.br/maslab/pergamus/pubs/BazzanKluegl.pdf.zip

[12] A. L. C. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multiagent Systems*, vol. 18, no. 3, pp. 342–375, June 2009. [Online]. Available: http://www.springerlink.com/content/j1j0817117r8j18r/

[13] A. R. Tavares and A. L. C. Bazzan, "Independent learners in abstract traffic scenarios," *Revista de Informática Teórica e Aplicada*, vol. 19, no. 2, pp. 13–33, 2012. [Online]. Available: http://seer.ufrgs.br/rita/article/view/rita_v19_n2_p13/23741