On the Role of Reward Functions for Reinforcement Learning in the Traffic Assignment Problem

Ricardo Grunitzki
Mobile Innovation Lab
SIDIA Institute of Science and Technology
Manaus, Brazil
ricardo.grunitzki@sidia.com

Gabriel de Oliveira Ramos
Graduate Program in Applied Computing
Universidade do Vale do Rio dos Sinos - UNISINOS
São Leopoldo, Brazil
gdoramos@unisinos.br

***Abstract*—The traffic assignment problem (TAP) consists of assigning routes to road users in order to minimize traffic congestion. Traditional methods for solving the TAP assume the existence of a central authority who computes and dictates routes to road users. Multi-agent reinforcement learning (MARL) approaches are more realistic in solving this kind of problem because they consider that road users (agents) have complete autonomy for choosing routes. However, MARL approaches usually require a long training period in order to compute the optimal routes, which could be a major limitation in more realistic traffic scenarios. In this paper, we tackle this problem by evaluating the performance of three conceptually different reward functions, namely: expert-designed rewards, difference rewards, and intrinsically motivated rewards. In particular, our focus lies on providing a deeper understanding of the impact of these reward functions on the agents' performance, thus contributing towards reducing congestion levels. To this end, we perform an extensive experimental evaluation on different road networks, including up to 360,600 concurrently learning agents. Our results show that, although the adopted reward functions were not able to speed up the learning process, the correct reward function choice plays an important role in the quality of the learned solution.**

I. INTRODUCTION

Traffic assignment plays a role in the classical planning and modeling of transportation systems [22]. A transportation system can be described by two elements: supply and demand. The former represents the transportation infrastructure (i.e., road network), while the latter represents the users (i.e., drivers) of such infrastructure. The traffic assignment problem (TAP) aims at connecting supply and demand while minimizing a particular objective function (e.g., average travel time). Specifically, traffic assignment methods select routes and assign them to road users considering their origins and destinations.

Several efficient methods for dealing with the TAP are available in the literature, including TAPAS [3] and algorithm B [9]. However, such methods assume the existence of a central authority who computes and assigns routes to the road users. In real scenarios, however, such an assumption is not valid because a traffic authority cannot directly control the behavior of self-interested road users regarding route choice [25].

An alternative for dealing with individual decision-making in the TAP is provided by multi-agent reinforcement learning (MARL), where road users are modeled as independent learning agents. Reinforcement learning (RL) deals with the problem of an agent learning a behavior to accomplish a task through successive interactions with the environment [34]. When multiple agents are learning concurrently in a shared environment, the problem is known as MARL. In any such case, the agent's behavior involves mapping situations (states) to actions, in a manner that maximizes some numerical utility or reward. Differently from other machine learning methods, an RL agent is not told what action to take in each situation. Rather, it must discover what actions maximize its reward by experiencing them.

Existing MARL-based approaches for the TAP differ primarily in the way agents' knowledge-base is modelled and in the employed MARL algorithm. This paper focuses on two approaches: route-based Q-learning and edge-based Q-learning. Both methods deal with the TAP in a decentralized perspective, in which each driver individually learns its route. These methods consider that agents neither have a complete observation of the network traffic condition nor consider the choices of other agents in their decision-making process. This discussion is necessary because complex scenarios, such as transportation, may present the following challenges: (i) a large number of concurrently learning agents—implying that agents must handle the influence of the actions of other agents in their decision-making process; (ii) large networks (in terms of topology)—increasing the search space associated with agents' choices. We remark that both approaches use the Q-Learning algorithm [37] to learn optimal policies. However, they differ in the way the search space is structured. The route-based Q-learning (route-based-QL) learns from a pre-computed subset of routes. On the other hand, the edge-based Q-Learning (edge-based-QL) learns a route from scratch (i.e., without prior knowledge about the road network), which may result in a larger space of possible routes.

In previous works [12], [25], [2], [24], edge-based-QL and route-based-QL were compared against classical centralized methods for the TAP, showing that their performance is sensitive to the features relative to the scenarios. As the number of agents and the search space increase, the learning task becomes harder for the agents. This is because agents' actions are highly coupled and because they share a common environment, thus making their reward signals highly noisy. One way to mitigate such problems and to improve the performance of MARL agents—as well as reduce the amount of sub-optimal actions taken by them—is through the correct definition of the reward functions.

In this paper, we investigate ways of improving the agents performance and convergence in solving the TAP through the design of better reward functions. We use three strategies for rewarding edge-based and route-based-QL agents. The first strategy uses expert-designed reward functions, which is the strategy most typically adopted by MARL system designers. The second strategy uses intrinsically-motivated reward functions, obtained by solving the extended-optimal reward problem (EORP) [12]. The third strategy uses difference rewards (DR) [36] to promote cooperation among the agents. These methods are evaluated in three scenarios that differ in the number of agents (including up to 360,600 concurrently learning agents) and the network topology (including up to 528 combinations of origins and destinations). Experimental results show that, for the evaluated scenarios, the three reward functions are very similar concerning convergence time. However, in terms of resulting average travel time, the difference rewards function has shown superior in smaller scenarios.

The rest of this paper is organized as follows. In Section II, the background and literature review are presented. Section III details the learning algorithms. In Section IV, the reward functions, scenarios, and experimental results are presented. Main conclusions and future work directions are discussed in Section V.

II. BACKGROUND AND RELATED WORK

This section discusses the theoretical background upon which we build our work. Section II-A introduces the TAP and related concepts. Section II-B presents the MARL concept as well as its main approaches for solving the TAP. Section II-C discusses the automated strategies for designing reward functions.

A. *Traffic assignment problem*

A transportation system is composed by supply and demand. The supply can be modeled as a directed graph $G\left(V,E\right)$, with the set of vertices V representing intersections, and the set of edges (or links) E representing the roads between these intersections. The weight of an edge $e \in E$ represents the cost c_e associated with crossing it (e.g., the cost could represent travel time, fuel spent, travel distance). In general, the cost of crossing an edge is abstracted by a volume-delay function (VDF), which maps the flow of vehicles using an edge to a travel time. An example of a VDF is shown in Equation 1 [6], where c_e represents the travel time on edge e, t_e^0 is the free-flow travel time on edge e (i.e., the minimum travel time assuming a zero flow), f_e is the flow of vehicles using edge e, C_e is the edge capacity, and $\mathcal{A}$ and $\mathcal{B}$ are parameters defined for each edge, giving its physical characteristics.

$$c_e(f_e) = t_e^0 \left[1 + \mathcal{A} \left(\frac{f_e}{C_e} \right)^{\mathcal{B}} \right] \tag{1}$$

The second part of the transportation system—the demand—represents the users of the infrastructure. The demand can be represented by an origin-destination (OD) matrix. An OD-matrix T contains I lines (origin vertices/zones) and J columns (destination vertices/zones). Each element T_{ij} represents the number of trips from vertex i to vertex j in a given time interval. It is said that $i \in I$ and $j \in J$ form an OD-pair.

The TAP consists of connecting the supply and demand by assigning routes to each road user. A route consists of a sequence of connected edges, forming a route between an origin and a destination. The cost of route p is defined by $c\left(p\right) = \sum_{e \in E_p} c_e$, which represents the sum of costs c_e of all edges $e \in E_p$ comprising route p.

The output of traffic assignment methods describes the state of a transportation system, which is a relevant input for traffic authorities to evaluate the dynamics of traffic and the possible consequences of changes in the physical infrastructure [22]. To evaluate the quality of an assignment, the literature presents two main solution concepts. The first is the user equilibrium (UE), which states that under UE condition, no road user can unilaterally reduce her travel time by shifting to another route. The UE does not ensure the system travel time is minimum. The total system travel time can be fund minimum under the system optimum (SO) condition, which is desirable from the traffic authorities point of view.

B. *Multi-agent reinforcement learning*

MARL deals with the problem of multiple agents learning in a shared environment. A well-know and more scalable MARL technique is that of multiple independent learners (MIL) [7]. This technique implements the agents' decision-making process in an individual manner, i.e., joint-actions are not considered [8]. In particular, agents consider each others' behaviour as the environment's underlying dynamics. This kind of modelling is particularly suitable in traffic settings, where drivers have no knowledge about other drivers' behaviour and about the overall traffic conditions, thus requiring them to make decisions independently, concurrently, and in the absence of a central authority dictating them what to do.

The agent's decision-making process can be modeled as a Markov decision process (MDP). An MDP is composed by a set of states S, a set of actions A, a state transition function $T : S \times A \rightarrow \Pi\left(S\right)$, where $\Pi\left(S\right)$ is a probability distribution over S, and a reward function $R : S \times A \rightarrow \Re$ that returns the reward $R\left(s,a\right)$ received after a given action $a \in \mathcal{A}\left(s\right)$ has been taken in state $s \in S$, where $\mathcal{A}\left(s\right)$ is the set of available actions in s.

The learning task in an MDP is to find a policy $\pi : S \rightarrow A$ that maximizes the agent's cummulative 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). A well-known exploration strategy is the ϵ-greedy (with a decreasing rate ϵ), which consists in choosing random actions (exploration) with probability $\epsilon \in [0, 1]$ or choosing best actions (exploitation) with probability $1 - \epsilon$. In general, ϵ starts with a high value (say, 1.0), which translates into high exploration, and decreases exponentially for each episode $\lambda \in \Lambda$ according to $\epsilon_\lambda = \epsilon * (\Delta\epsilon)^\lambda$, where $\Delta\epsilon$ represents the epsilon decay rate in the interval $[0, 1]$.

The Q-learning algorithm, presented by [37], is a traditional model-free algorithm used to learn value functions independently of the policy being followed. Its update rule is shown in Equation 2, 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 MDP, which stores the value functions (or 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 policy π thereafter.

$$Q(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') \right] \quad (2)$$

Transportation scenarios represent a challenging testbed for MARL algorithms given its distributed nature and the dynamics resulting from agents self-interestedness. The most common learning tasks in a transportation system are the route learning [12], [5], [14] (as in the case of the TAP) and the traffic signal timing learning [18], [23], [10].

C. Methods for reward function design

The reward function is the element of the RL framework that defines and represents a specific learning task [19]. The system designer[1] is responsible for designing a function to efficiently guide the learning process. There is no general rule for the definition of a reward function. In practice, the system designer defines a reward function empirically, based on her intuition. In this present paper, we will refer to this kind of reward functions as *expert-designed* reward functions.

The literature also presents methods such as the extended optimal reward problem (EORP) [13], difference rewards (DR) [36], inverse reinforcement learning [39], and reward shaping [19], which can all support, at least partially, the system designer in the task of modeling reward functions. However, inverse reinforcement learning and reward shaping do not fit our goals of designing efficient reward functions for the TAP due to the following reasons. Reward shaping methods just provide additional rewards to agents, through an additional reward function, in order to speed-up the learning convergence without modifying the optimal policy [19]. It assumes that a reward function is given by the system designer, while EORP and DR provide such a reward function. The inverse RL technique deals with the problem of identifying the reward function being optimized by an agent, given observations of its activity [28]. We do not have such a set of observations representing the trajectory of the agents performing the TAP task. Therefore, the only methods that fit our needs are EORP and DR. The following sections describe them in detail.

[1]The system designer here represents the expert or team who is modeling the MARL solution for TAP.

1) Extended-optimal reward problem: In the classical RL framework the reward function represents two purposes: (i) defining the preferences of the system designer by representing a *fitness function*; and (ii) guiding the behavior of the agent by representing the guidance *reward function*. The use of a single function has shown to confound these two purposes [4], [32].

By definition, the standard RL framework assumes that goals of both agent and designer should be the same. Some works [30], [31] suggested that both reward function and fitness function must be separated. During the modeling stage of an RL solution, it is assumed that the fitness function is known by the designer. However, the correct definition of reward functions is not so direct in complex problems. The authors present the Optimal Reward Problem (ORP), that is the problem of finding an optimal reward function (ORF) that maximizes the fitness function.

The work of [30], [31] focuses on the theoretical formulation of ORP. Following works on ORP present methods to find approximated solutions for the ORP [21], [33], but they were limited to single-agent problems. Further works [17], [27] emerged to fell the gap of multi-agent problems. From these two approaches, we choose the extended-ORP (EORP) [27], because this is the only ORP variation capable of dealing with multi-agent non-cooperative tasks such as the TAP.

The EORP is formally defined as follows. At each time step, an agent $i \in I$ receives an observation $o \in \mathcal{O}$ from its environment $\mathcal{M}$, takes action $a \in A$, produces a history h_i, and repeats this process for a certain time horizon. The agent's goals are represented by a reward function R. The designer's goals are represented by a multi-objective evaluation function $\mathcal{F}$, which produces a return vector $\mathcal{F}_R(H)$ for a reward function R over a set of histories H. This set of histories represents all histories h_i generated by agents $i \in I$ learning with a given reward function R. The EORP is formalized in Equation 3, where R^* is an optimal reward function that maximizes the designer expected evaluation function $\mathcal{F}$ of a set of agents I in some environment $\mathcal{M}$; the optimization is over a set of reward functions in a reward design space $\mathcal{R}(J)$. J is the set of reward features that compose the search space of reward functions. The expectation operator is denoted by $\mathbb{E}$. The notation $H \sim \mathcal{M}\langle I(R)\rangle$ denotes that H is a set of all sample histories generated when agents $i \in I$ acts in environment $\mathcal{M}$. This formulation provides two main changes with respect to the original ORP: i) a reward design space, $\mathcal{R}(J)$, corresponding to the space of reward functions spanned by a given set of feature states J; and ii) a multi-objective function $\mathcal{F}$, called evaluation function.

$$R^* = \underset{R \in \mathcal{R}(J)}{\arg\max} \mathbb{E}\left[\mathcal{F}_{R_0}(H) \mid H \sim M\langle I(R)\rangle\right] \quad (3)$$

In the reward design space, $\mathcal{R}(J)$, more than one function $R \in \mathcal{R}(J)$ can produce the same fitness F_R, but under different learning effort. For instance, one reward function may allow the agent to learn an optimal policy in fewer steps because it provides the agent with more informative guidelines about the effectiveness of its current behavior. For the designer, it is interesting to identify a reward function that, when optimized, results in the agent more rapidly learning to solve the task. The EORP considers that the designer has multiple goals to be maximized, such as fitness and learning effort. The function $\mathcal{F}$ evaluates the fitness (f_1) and effort (f_2) produced by H and returns a 2-dimensional vector $\mathcal{F}(H) = [f_1(H), f_2(H)]$. The fitness function measures the quality of the final learned behavior achieved when maximizing a given reward function, while the learning effort function evaluates the amount of effort spent by the agent (e.g., time until convergence) during its lifetime. The designer must define both functions to represent his goals for a given learning task.

The reward design space $\mathcal{R}(J)$ represents the set of all possible reward functions spanned by a given set of features J. A feature can be seen as a situation in which the agent may be rewarded. A reward function $R \in \mathcal{R}(J)$ is represented as $R = \sum_{j \in J} s(j)\, w(j)\, P(j)$, where for each feature j, $s(j) \in \{0, 1\}$ indicates that the feature j is activated or not in state s, $w(j) = \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$ represents the contribution to the reward signal of j, and $P(j) = \{x \in \mathbb{N} \mid 0 \leq x \leq 1\}$ is an indicator function reflecting if j is used to compose or not R. Given a set of n features, a search space $\mathcal{R}(J) \subset \mathcal{R}^{2n} = [\{w(j_0), \ldots, w(j_n)\}, \{p(j_0), \ldots, p(j_n)\}]$ contains $n = |J|$ indicator features $P(j)$ and n reward signals, $w(j)$. The number of decision variables of an EORP with such search space is $2n$. Our method optimizes a single reward function that is used by all collective of learning agents I interacting in a given problem. This way, independently of the amount of agents, the dimensionality of the optimization problem is always given by the amount of features ($|J|$) in the state space of one individual agent.

Solving the EORP consists in finding the set of Pareto optimal solutions, $R^* \in \mathcal{R}(J)$, that maximize $\mathcal{F}$. Any multi-objective optimization algorithm that deals with real and integer decision variables can be used. EORF produces only one EORP solution per set of agents, so that every agent $i \in I$ will learn using an instance of this function.

2) Difference rewards: In multi-agent cooperative problems, the function normally rewards each agent according to the overall system performance. However, according to [36], such reward structure may lead to slow learning in some domains. Given that reinforcement learning agents aim at maximizing 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's performance. The full performance of the system can be measured by a system's utility function $G(z)$, where z is a variable that represents the state-action pairs of all agents in the system.

DR are functions proposed by [36], which aim at providing reward signals that are both sensitive to the agents' actions and aligned with the overall system's reward. Consider the DR as in Equation 4, z_{-i} is the state-action pairs for a theoretical system without the contribution of agent i. All components of z that are affected by agent i are replaced with the fixed constant c_i.

$$D_i \equiv G(z) - G(z_{-i} + c_i) \quad (4)$$

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 DR to evaluate the performance of the system without i. Therefore, D_i can be seen as the contribution of agent i to the system's performance. According to [36], there are two advantages using Equation 4 for rewarding agents. 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* [1], [35]. Second, the second term does not depend on the actions of agent i. In other words, any action taken by agent i that improves D, also improves G. This term measures the alignment between the two rewards. As such, it has been dubbed *factoredness* [1], [35].

DR 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 [36]. In addition, this method deals only with cooperative tasks.

III. LEARNING ALGORITHMS

In this paper, we address the TAP from the MARL perspective using two approaches: edge-based-QL [12], [2] and route-based-QL [25], [24], [2]. Both methods model the agent's decision-making process as an MDP. The main difference between these two approaches is that route-based-QL restricts the agents' action space to a pre-computed subset of shortest routes. Although this modelling may seem realistic (since real drivers usually know a few shortest routes for their daily trips), it relies on prior knowledge about the road network to generate the set of routes. On the other hand, edge-based-QL does not restrict the agents' action space. Rather, it allows agents to explore roads in particular, thus enabling agents to explore all possible routes within the road network.

As typical in the literature, we assume that each learning agent is a road user associated with a trip in an OD-matrix. Each agent can learn a route between its origin state (s_-) and destination state (s_+). For both edge-based-QL and route-based-QL, the learning process is organized in episodes (trials) and time steps (time unit). An episode i ends in $t_{\max}$ time steps or when all agents reach they terminal state s_+, whichever comes first.

Also following the literature (which abstracts the network as a graph), we model traffic from a macroscopic perspective [22], [29], where each unit of time step $t \leq t_{\max}$ represents a hop in the graph. For instance, the route $p = (v_0, v_1, v_2)$ that connects v_0 to v_2, takes $t = (|p| - 1)$ time steps to be traveled.

The following sections detail each MARL approach.

A. Edge-based Q-Learning

In the *edge-based* approach, agent's actions correspond to the edges of the road network. When an agent is learning with this method, it can find a policy from the set of all routes available in the search space. The agent's MDP is modeled as follows. A state s is a vertex $v \in V$ of the road network in which the agent is located. Each state $s \in S$ has a set of available actions $\mathcal{A}(s)$, which is represented by the outgoing edges of the corresponding vertex of s. The reward received by the agent after taking action a in state s and transitioning to state s' is given by $R(s, a, s')$.

If we consider that agents can drive in loops, this approach may present infinite routes. To handle this issue, the agent's search space is limited to a maximum number of time step $t_{\max}$. Thus, if the agents do not find a route from s_- to s_+ in $t_{\max}$ time steps, then the current episode is stopped. Even though the stopped route belongs to an invalid set of routes P_{invalid}, it is also a possible solution experienced by the agent in search space. In this manner, the set of all possible routes for a edge-based-QL agent is defined as in Equation 5, where $P_{\text{valid}} = \left\{ p_{i,j}^{(s_-,s_+)}(t_i) \mid i = 1, \ldots, M; j = 1, \ldots, N_i \right\}$, where M is the current simulation time step, and N_i is the number of routes from s_- to s_+ in t_i steps; and $P_{\text{invalid}} = \left\{ p_{i,j}^{(S,V_{t_i})}(t'_i) \mid i = 1, \ldots, M'; j = 1, \ldots, N'_i \right\}$, where M' is the current simulation time step, and N'_i is the number of routes from s_i to $v_{t'_i}$ in t'_i steps. An important property of P for this approach is that $P_{\text{valid}} \cap P_{\text{invalid}} = \emptyset$, i.e., every route is either valid or invalid.

$$P^{(s_-,s_+,t_{\max})} = P_{\text{valid}} \cup P_{\text{invalid}} \tag{5}$$

B. Route-based Q-Learning

As opposed to edge-based-QL, in route-based-QL the agent's action represents a complete route connecting its origin to its destination. The search space P is restricted to a predefined subset of possible routes, as shown in Equation 6, where j is the index of the j-th lowest cost route. This subset of routes is pre-processed before the learning process begins. Thus, each agent receives $|P|$ pre-computed routes from s_- to s_+.

$$P^{(s_-,s_+)} = \{p_i, p_2, \ldots, p_j\} \tag{6}$$

The set of actions is defined according to the number of predefined routes used. A parameter $K = |P|$ must be defined to determine the number of routes to be calculated. These routes are the K-lowest cost routes of the agent's OD-pair under no congestion. These routes are computed in the graph using the algorithm K Shortest Loopless Paths [38], which can find the K shortest routes without loops for an OD-pair.

The agent's MDP is modeled as follows. The set of states contains only the initial and terminal states. At its initial state, the agent has $\mathcal{A}(s_-) = P$ actions. Each action is a route that connects s_- to s_+. When the agent reaches the terminal state, it receives a reward according to the cost of traveling the route, which is given by $R(s_-, p_a, s_+)$, where p_a represents the route taken from $\mathcal{A}(s_-)$.

In this approach, an episode finishes when all agents reach a terminal state. The $t_{\max}$ parameter is disregarded here because no route has loops. This approach presents a major disadvantage when compared against edge-based-QL, which is the extra parameter ($|P|$). Furthermore, even though the search space restriction can simplify the learning process, it may underestimate the ability of the agent to learn the most appropriate policy if the search space was set incorrectly.

IV. EXPERIMENTS

We now present an experimental evaluation of the methods presented above. In particular, the aim here is to show that:

- The proper choice of the learning algorithm impacts the quality of the learned behaviour as well as in the learning period it takes to converge;
- The proper choice of the reward function can improve the performance of existing MARL algorithms for traffic assignment;
- The results obtained by difference rewards can be achieved by more simpler methods that do not make use of global information; and
- The use of intrinsically-motivated reward functions is beneficial in traffic assignment domain.

A. Methodology

In order to evaluate the proposed approaches for the TAP, we use three scenarios that differ regarding network topology and number of agents. The first scenario, named OW, was proposed in [22, Chapter 10]. Its network topology contains 13 vertices and 24 edges, which is used by a constant flow of 1,700 trips distributed over 4 OD-pair. The cost function for each edge e is $c_e = t_e^0 + V_e \times 0.02$, where c_e is the travel time in minutes to cross edge e; t_e^0 is the free-flow travel time for edge e; V_e is the flow using e. The travel time increases linearly in 0.02 minutes per vehicle using the edge.

The second scenario is an adaptation of the network presented by [20], named ND. We modified the original road network so that all roads are two-way to provide more options of routes. The resulting graph has 13 vertices and 38 edges. The demand is composed of 2,000 trips distributed over 4 OD-pairs.

The third and larger scenario is the Sioux Falls (SF) [16]. This is a well-known transportation problem used in the literature as a testbed for traffic assignment methods. The road network has 24 vertices and 76 edges. The demand is comprised of 360,600 trips distributed among 528 OD-pairs. The cost function for the edges is the VDF presented in Equation 1.

TABLE I
RELEVANT ASPECTS OF SCENARIOS OW, ND AND SF.

Aspect	OW scenario	ND scenario	SF scenario
Trips	1,700	2,000	360,600
OD-pairs	4	4	528
Vertices	13	13	24
Edges	24	38	76
UE	≈ 67.16	≈ 50.28	≈ 20.78
SO	≈ 66.92	≈ 50.01	≈ 19.95

Relevant aspects of the scenarios are summarized in Table I[2]. This table also present for each scenario their average travel time (in minutes) under user equilibrium (UE). Note that SF represents a more challenging scenario for MARL than OW and ND, since it has a larger network topology, a demand up to 200 times larger, and a higher number of OD-pairs than others scenarios. From the point of view of individual decision-making, SF scenario is harder than OW and ND scenario because agents need to learn in a larger and more dynamic environment.

B. Reward functions

This section presents the definition of the three reward functions considered in the present work.

1) Expert-designed: For this class of reward function, we will use the reward functions presented in [12], [14], [11], [5], [26]. In this strategy, the travel time resulting by an agent's action is used to compose the reward signal. For edge-based-QL, the expert-designed reward function is given as in Equation 7. By learning with this reward function, at each action performed, the agent is rewarded by the negative travel cost (c_e) of the traveled edge corresponding to agent's action.

$$R = -c_e \tag{7}$$

The expert-designed reward function for route-based-QL is defined as in Equation 8, where E_p is the set of edges that composes route p. This is similar to the reward function in Equation 7. The difference is that, in route-based-QL, the feedback received by an agent's action refers to the negative travel cost of a route and not of a single edge.

$$R = \sum_{e \in E_p} -c_e \tag{8}$$

2) Difference rewards: For both edge-based and route-based-QL, the DR function in given as in Equation 4, where $G(z) = \dfrac{\sum_{i \in I} c(p_i)}{|I|}$ represents the average travel time of the traveled routes of all agent in I; and $G(z_{-i})$ represents the average travel time of the routes of all agent in $I \setminus \{i\}$, i.e., the travel time of agent i is disregarded from the average travel time in $G(z_{-i})$.

[2]All data sets containing networks, demands and cost functions for the three scenarios are available at http://github.com/goramos/transportation_networks.

3) Extended-optimal reward problem: For obtaining optimal reward functions through EORP, we need to: (i) define the reward design space $\mathcal{R}(J)$; (ii) define the evaluation function $\mathcal{F}(H)$; and (iii) solve the EORP.

The reward design space for the TAP is defined as follows. In Table II, we present the set of reward features (column J) manually extracted for each learning algorithm. This space captures the most common situations faced by the agents in the TAP. The set of reward functions available for edge-based-QL is different from the one of route-based-QL. This is because edge-based-QL abstracts less environment information in its action structure than route-based-QL does. These features represent real or binary variables (column *type*). The reward design space of both algorithms is given by $\mathcal{R}(J) \subset \mathcal{R}^{|J|}$, where J is the set of reward features. The reward function $R = \sum_{k=1}^{|J|} s(j_k)\, w(j_k)\, P(j_k)$ returns a reward signal based on the values of the reward features activated in a given state.

We define the multi-objective evaluation function $\mathcal{F}(H) = [f_1(H), -f_2(H)]$ as follows. The fitness function of each agent i is given by $f_1(h_i) = -C(p_i)$, which represents the negative cost of its learned route p_i. This function considers only the fitness produced in a given episode. By using this function, the fitness is maximized when the travel time of the agent is minimized. The effort function is given by $f_2(h_i) = \frac{\sum_{\lambda=0}^{|\Lambda|} t_\lambda}{|\Lambda|}$, which represents the average number of decisions taken during the lifetime of the agent. Both MARL algorithms use such evaluation function.

The next step is solving the EORP. We opted for using a multi-objective version of the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)[15] to solve the EORP because it has been shown successful in solving other RL applications [13]. We defined the set of CMA-ES parameters as in the work of [13], which uses a population size of 20 elements and a stopping criteria in 1000 evaluations.

Solving the EORP for edge-based-QL produced 25 Pareto-optimal reward functions. For comparison purposes, Table III shows one (arbitrarily chosen) of such reward functions. For route-based-QL, the EORP returned just a single reward function because in edge-based-QL the learning agents always performs just one action per episode, resulting in learning effort always equals to 1.

C. Parameters setting

For the Q-learning algorithm and the employed exploration strategy, four parameters have to be set: learning rate (α), discount factor (γ), number of episodes (Λ), and exploration rate (ϵ). This paper considers a fixed learning horizon of $\Lambda = 1000$ episodes. The rest of the parameters were set empirically. Consider $\alpha = 0.5$ and $\gamma = 0.99$. For route-base-QL, consider $|P| = 10$. The exploration policy utilized is the ϵ-decreasing, with $\epsilon = 1.0$ and $\Delta\epsilon = 0.99$. All experiments presented in this paper were repeated 30 times. In order to measure the performance of an algorithm, we used the normalized average travel time (NATT), as follows:

$$\phi(v, v^*) = \frac{v}{v^*}, \tag{9}$$

TABLE II

SET OF REWARD FEATURES (J) AVAILABLE FOR EDGE-BASED AND ROUTE-BASED-QL. COLUMN *type* DESCRIBES THE VARIABLE TYPE THAT REPRESENTS EACH REWARD FEATURE.

J	Description	Type	Edge-based-QL	Route-based-QL
j_0	edge takes the agent to its origin vertex?	Binary	✓	✗
j_1	edge takes the agent to its destination vertex?	Binary	✓	✗
j_2	edge was already traveled by the agent?	Binary	✓	✗
j_3	traveled distance	Real	✓	✓
j_4	travel cost under free-flow condition	Real	✓	✓
j_5	flow of vehicles	Real	✓	✓
j_6	travel time	Real	✓	✓

TABLE III

DETAIL OF AN OPTIMAL REWARD FUNCTION FOR EDGE-BASED AND ROUTE-BASED-QL IN THE THREE TAP SCENARIOS.

reward feature	edge-based-QL	route-based-QL
j_0	0.52	N/a
j_1	-0.55	N/a
j_2	-0.58	N/a
j_3	-	-
j_4	0	-0.28
j_5	-	-
j_6	-1	-0.61

TABLE IV

AVERAGE PERFORMANCE (AND STANDARD DEVIATION) OF THE CONSIDERED METHODS ON DIFFERENT SCENARIOS IN TERMS OF THE NORMALIZED AVERAGE TRAVEL TIME. LOWER IS BETTER. BEST RESULTS HIGHLIGHTED IN BOLD. ON AVERAGE, ROUTE-BASED APPROACHES FARED BETTER THAN EDGE-BASED ONES IN MOST CASES.

Method	Scenario		
	OW	ND	SF
EBQL-ED	1.0034 (0.0003)	1.0045 (0.0008)	1.0998 (0.0049)
EBQL-DR	**1.0009 (0.0004)**	1.0068 (0.0011)	1.0967 (0.0060)
EBQL-EORP	1.0058 (0.0005)	1.0062 (0.0009)	1.0987 (0.0045)
RBQL-ED	1.0035 (0.0002)	1.0049 (0.0001)	**1.0323 (0.0034)**
RBQL-DR	**1.0015 (0.0031)**	**1.0004 (0.0000)**	1.0391 (0.0049)
RBQL-EORP	1.0038 (0.0011)	1.0049 (0.0001)	**1.0305 (0.0030)**

where v denotes the average travel time at the last episode of the simulation, and v^* denotes the optimal reference value, as reported in Table I. The lowest the value of ϕ, the better.

D. Results

To evaluate and compare the performance of edge-based-QL and route-based-QL under the guidance of expert-designed (ED), DR and EORP reward functions, we use the three scenarios presented in Section IV-A. Two analyses are performed here. In the first analysis, we compare the obtained results of all methods regarding the normalized average travel time (NATT). In the second analysis, the convergence curve of each algorithm is analyzed in a representative scenario. The results concerning NATT for each approach are presented in Table IV.

In the OW scenario, the performance of all methods was closer to the optimum. In all cases, however, the DR reward function yielded slightly better results. As the OW scenario has only a few options of possible routes, it creates a lot of competition among selfish-agents for specific parts of the road network. In such situations, the DR function is capable of achieving a better NATT because it elicits cooperation among agents. Such cooperation reduces the competition for the most attractive edges and benefits the system as a whole.

For the ND scenario, again all approaches achieved close-to-optimum NATT. The best performance was achieved using the route-based-QL approaches, especially with DR. This is due to the fact that the ND scenario is more challenging than OW. In particular, as the network topology and demand in the ND scenario are greater than in OW, the learning task becomes harder for edge-based-QL, since the search space is considerably larger and the reward signals are noisier.

In the SF scenario, the learning algorithm played a major role in performance than the reward function did. Regardless of the reward function used, route-based-QL yielded a NATT closer to the optimum. The massive demand and number of OD-pairs make the reward signals extremely noisy for the agents. However, even under such circumstances, the search space restriction provided by route-based-QL made it possible for the agents to learn more appropriate routes. For edge-based-QL, the three reward functions yielded a NATT worse than the ones obtained by route-based-QL. This is because the larger network topology of scenario SF increases the search space in the case of edge-based-QL.

The results presented so far indicate that the three reward functions are robust in solving the TAP. In larger scenarios, as in the SF presented here, regardless of the employed reward function, the edge-based-QL ends up limiting the learning capacity of agents. It is harder for the learning agents to learn optimal policies in the larger and highly dynamic search space proved by edge-based-QL. For overcoming such limitation, these results suggest that, in larger scenarios, it is more appropriate to use route-based-QL than edge-based-QL.

In order to better evaluate the proposed approaches, the learning curves—NATT along episodes—of the learning methods and reward functions are presented in Figure 1 for the SF. As can be observed, route-based-QL converges faster than edge-based-QL, independently of the reward function being used. In the early episodes, edge-based-QL agents have no knowledge in their MPD, and the exploration rate is very high. Consequently, agents have a high probability of converging to invalid policies. Despite this, after some episodes, the learning curves of both methods converge to similar solutions in all scenarios.

In the first experiment, we showed that the three used reward functions are robust in guiding the learning process of TAP agents. In small scenarios such as OW and ND, the

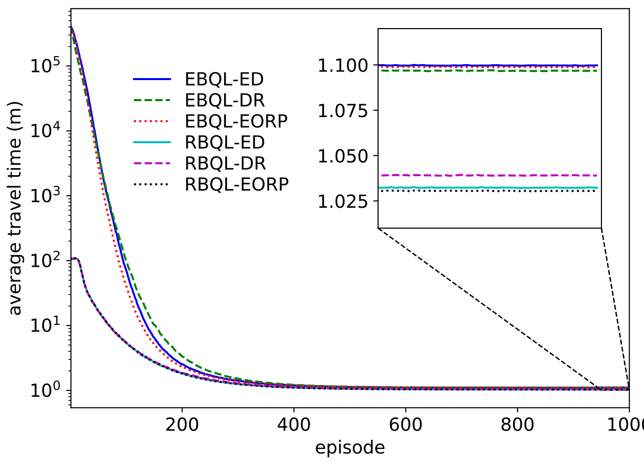

Fig. 1. Performance (based on normalized average travel time) of the tested methods along episodes on the SF scenario. Lower is better. Route-based approaches fared better than edge-based ones. EORP reward functions lead to best results.

DR function was able to yield slightly better NATT. In spite of this, it was evident that the learning algorithm has a more considerable influence on the performance of the agents than the reward functions we used. The second experiment further concluded that all reward functions have similar performance regarding of convergence.

The three reward functions evaluated here are very similar concerning the obtained NATT and convergence time. However, EORP and DR reward functions are not trivially applicable in TAP. EORP reward functions require a lot of work during the definition and resolution of the EORP, while DR functions assume a communication channel among agents by which they obtain the necessary information for the calculation of system's performance. The expert-designed reward function seems to be a more adequate reward function for TAP because its definition is more straightforward than others.

V. CONCLUDING REMARKS

This paper deals with the traffic assignment problem (TAP) in a distributed perspective, in which the road users are capable of learning the most appropriate routes by their own. Two multi-agent reinforcement learning (MARL) approaches that differ in search space restriction are considered. The first, edge-based-QL, uses the whole search space in agents decision-making process. The second, route-based-QL, restricts the search space of the agents to a subset of precomputed solutions. Both learning algorithms were evaluated under the guidance of three different strategies for defining reward functions: expert-designed, difference rewards and extended-optimal reward problem-based (EORP). These approaches are evaluated in scenarios that differ in the number of agents (road users) and search space size (network topology).

Experimental results showed that, for the evaluated scenarios, the three reward functions are very similar concerning their obtained ATT and convergence time. However, expert-designed reward functions are simpler and more straightforward to be acquired and/or computed than DR and EORP-based reward functions. Independently of the reward function being used, the learning capability of the agents is sensitive to some features of the problem such as the number of agents and size of the search space. Any of the three reward function strategies used here was able to mitigate this problem in edge-based-QL, suggesting that the search space restriction provided by route-based-QL is still a more efficient alternative to deal with such larger scenarios.

ACKNOWLEDGEMENTS

The authors are partially supported by CNPq, FAPERGS and SIDIA Institute of Science and Technology.

REFERENCES

[1] Adrian Agogino and Kagan 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*, eds., Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, volume II, pp. 81–88, New York, NY, (July 2005). acm.

[2] Baruch Awerbuch and Robert D. Kleinberg, ‘Adaptive routing with end-to-end feedback: distributed learning and geometric approaches’, in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing - STOC ’04*, pp. 45–53, New York, New York, USA, (2004). ACM Press.

[3] Hillel Bar-Gera, ‘Traffic assignment by paired alternative segments’, *Transportation Research Part B: Methodological*, **44**(8-9), 1022–1046, (sep 2010).

[4] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez, ‘Intrinsically motivated learning of hierarchical collections of skills’, in *Proc. 3rd Int. Conf. Development Learn*, pp. 112–119, (2004).

[5] Ana L. C. Bazzan and R. Grunitzki, ‘A multiagent reinforcement learning approach to en-route trip building’, in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 5288–5295, (July 2016).

[6] of Public Roads Bureau, *Bureau of Public Roads: Traffic Assignment Manual*, 1964.

[7] 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).

[8] Caroline Claus and Craig Boutilier, ‘The dynamics of reinforcement learning in cooperative multiagent systems’, in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 746–752, (1998).

[9] Robert B. Dial, ‘path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration’, *Transportation Research Part B: Methodological*, **40**(10), 917–936, (dec 2006).

[10] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, ‘Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto’, *Intelligent Transportation Systems, IEEE Transactions on*, **14**(3), 1140–1150, (Sept 2013).

[11] Ricardo Grunitzki and Ana L. C. Bazzan, ‘Combining car-to-infrastructure communication and multi-agent reinforcement learning in route choice’, in *Proceedings of the Ninth Workshop on Agents in Traffic and Transportation (ATT-2016)*, eds., Ana L. C. Bazzan, Franziska Klügl, Sascha Ossowski, and Giuseppe Vizzari, New York, (July 2016). CEUR-WS.org.

[12] Ricardo Grunitzki and Ana L. C. Bazzan, ‘Comparing two multiagent reinforcement learning approaches for the traffic assignment problem’, in *Intelligent Systems (BRACIS), 2017 Brazilian Conference on*, (Oct 2017).

[13] Ricardo Grunitzki, Bruno C. da Silva, and Ana L. C. Bazzan, 'A flexible approach for designing optimal reward functions', in *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, eds., S. Das, E. Durfee, K. Larson, and M. Winikoff, pp. 1559–1560, São Paulo, (May 2017). IFAAMAS.

[14] 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*, pp. 253–258, (Oct 2014).

[15] Christian Igel, Nikolaus Hansen, and Stefan Roth, 'Covariance matrix adaptation for multi-objective optimization', *Evolutionary computation*, **15**(1), 1–28, (2007).

[16] 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).

[17] Bingyao Liu, S Baveja, RL Lewis, and Shiyin Qin, 'Optimal Rewards for Cooperative Agents', *Autonomous Mental Development, IEEE Transactions on*, **11**(4), (2014).

[18] Patrick Mannion, Jim Duggan, and Enda Howley, 'An experimental review of reinforcement learning algorithms for adaptive traffic signal control', in *Autonomic Road Transport Support Systems*, eds., Leo Thomas McCluskey, Apostolos Kotsialos, P. Jörg Müller, Franziska Klügl, Omer Rana, and René Schumann, 47–66, Springer, (2016).

[19] Andrew Y. Ng, Daishi Harada, and Stuart Russell, 'Policy invariance under reward transformations: Theory and application to reward shaping', in *In Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287. Morgan Kaufmann, (1999).

[20] Sang Nguyen and Clermont Dupuis, 'An efficient method for computing traffic equilibria in networks with asymmetric transportation costs', *Transportation Science*, **18**(2), 185–202, (1984).

[21] Scott Niekum, A.G. Barto, and Lee Spector, 'Genetic Programming for Reward Function Search', *IEEE Transactions on Autonomous Mental Development*, **2**(2), 83–90, (2010).

[22] Juan de Dios Ortúzar and Luis G. Willumsen, *Modelling transport*, John Wiley & Sons, Chichester, UK, 4 edn., 2011.

[23] KJ Prabuchandran, AN Hemanth Kumar, and Shalabh Bhatnagar, 'Multi-agent reinforcement learning for traffic signal control', in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 2529–2534. IEEE, (2014).

[24] Gabriel Ramos, Roxana Radulescu, and Ann Nowé, 'A Budged-Balanced Tolling Scheme for Efficient Equilibria under Heterogeneous Preferences', in *Proceedings of the Adaptive Learning Agents Workshop 2019 (ALA-19)*, Montreal, (2019).

[25] Gabriel de O. Ramos, Bruno C. da Silva, and Ana L. C. Bazzan, 'Learning to minimise regret in route choice', in *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pp. 846–855, São Paulo, (2017). IFAAMAS.

[26] Gabriel de O. Ramos and Ricardo Grunitzki, 'An improved learning automata approach for the route choice problem', in *Agent Technology for Intelligent Mobile Services and Smart Societies*, eds., Fernando Koch, Felipe Meneguzzi, and Kiran Lakkaraju, volume 498 of *Communications in Computer and Information Science*, 56–67, Springer Berlin Heidelberg, (2015).

[27] Ricardo Grunitzki, Bruno C. da Silva, Ana L. C. Bazzan, and Jorge C. Chamby-Diaz, 'Towards Designing Optimal Reward Functions in Multi-Agent Reinforcement Learning Problems', in *2018 IEEE Joint Conference on Neural Networks*, Rio de Janeiro, (2018).

[28] Stuart Russell, 'Learning agents for uncertain environments (extended abstract)', in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 101–103, New York, NY, USA, (1998). ACM.

[29] Guni Sharon, Michael W. Levin, Josiah P. Hanna, Tarun Rambha, Stephen D. Boyles, and Peter Stone, 'Network-wide adaptive tolling for connected and automated vehicles', *Transportation Research Part C: Emerging Technologies*, **84**, 142 – 157, (2017).

[30] Satinder Singh, Richard L Lewis, and Andrew G Barto, 'Where do rewards come from', in *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606, (2009).

[31] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg, 'Intrinsically motivated reinforcement learning: An evolutionary perspective', *Autonomous Mental Development, IEEE Transactions on*, **2**(2), 70–82, (2010).

[32] Satinder P Singh, Andrew G Barto, and Nuttapong Chentanez, 'Intrinsically motivated reinforcement learning.', in *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, volume 17, pp. 1281–1288, (2004).

[33] Jonathan Sorg, Richard L Lewis, and Satinder P Singh, 'Reward Design via Online Gradient Ascent', in *Advances in Neural Information Processing Systems 23*, eds., J D Lafferty, C K I Williams, J Shawe-Taylor, R S Zemel, and A Culotta, 2190–2198, Curran Associates, Inc., (2010).

[34] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[35] K. Tumer and D. Wolpert, 'A survey of collectives', in *Collectives and the Design of Complex Systems*, eds., K. Tumer and D. Wolpert, 1–42, Springer, (2004).

[36] Kagan Tumer and Adrian Agogino, 'Distributed agent-based air traffic flow management', in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 1–8, New York, NY, USA, (2007). ACM.

[37] Christopher J. C. H. Watkins and Peter Dayan, 'Q-learning', *Machine Learning*, **8**(3), 279–292, (1992).

[38] Jin Y. Yen, 'Finding the k shortest loopless paths in a network', *Management Science*, **17**(11), 712–716, (1971).

[39] Shao Zhifei and Er Meng Joo, 'A review of inverse reinforcement learning theory and recent advances', in *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE, (jun 2012).