Urban Driving with Multi-Objective Deep Reinforcement Learning

Changjian Li
University of Waterloo
Waterloo, Ontario, Canada
changjian.li@uwaterloo.ca

Krzysztof Czarnecki
University of Waterloo
Waterloo, Ontario, Canada
kczarnec@gsd.uwaterloo.ca

ABSTRACT

Autonomous driving is a challenging domain that entails multiple aspects: a vehicle should be able to drive to its destination as fast as possible while avoiding collision, obeying traffic rules and ensuring the comfort of passengers. In this paper, we present a deep learning variant of *thresholded lexicographic Q-learning* for the task of urban driving. Our multi-objective DQN agent learns to drive on multi-lane roads and intersections, yielding and changing lanes according to traffic rules. We also propose an extension for *factored Markov Decision Processes* to the DQN architecture that provides auxiliary features for the Q function. This is shown to significantly improve data efficiency. [1] We then show that the learned policy is able to zero-shot transfer to a ring road without sacrificing performance.

KEYWORDS

reinforcement learning; multi-objective optimization; markov decision process (MDP); deep learning; autonomous driving

1 INTRODUCTION

Deep reinforcement learning (DRL) [13] has seen some success in complex tasks with large state space. However, the task of autonomous urban driving remains challenging, partly due to the many aspects involved: the vehicle is not only expected to avoid collisions with dynamic objects, but also follow all the traffic rules and ensure the comfort of passengers. One motivation for the multi-objective approach is the difficulty in designing a scalar reward that properly weighs the importance of each aspect of driving so that the designer's original intention is reflected. Although there have been attempts to learn the reward function through inverse reinforcement learning [1, 29], these methods add additional computational expenses, and require the availability of demonstrations. Another motivation comes from the problem of exploration [3, 15]. If the agent explores randomly, it might hardly have the chance of reaching the intersection, so the traffic rules at the intersection

might never be learned. In contrast, if each aspect is learned separately, we would have the flexibility of choosing which aspect to explore in a given state. In this paper, we consider a multi-objective RL approach to the problem of urban driving, where each objective is learned by a separate agent. These agents collectively form a combined policy that takes all these objectives into account. In addition to those mentioned above, there are several advantages of the multi-objective approach: (1) Since each objective only considers one aspect of driving, the entire task is divided into smaller, simpler tasks, e.g., smaller state space can be used for each objective. (2) In a new task where only some of the objectives change, the learned policy for other objectives can be reused or transferred. (3) For some of the objectives, the desired behavior might be easy to specify manually. The multi-objective architecture allows these behaviors to be implemented with rule-based systems without creating integration issues.

In this paper, we adopt the thresholded lexicographic Q-learning framework proposed by Gábor et al. [6], and adapt it to the deep learning setting. We set an adaptive threshold for the Q value of each objective, and at each state, the set of admissible actions is restricted by lexicographically applying the threshold to each objective. Therefore, the policy obtained either satisfies all constraints, or, if that's not possible, satisfies the constraints for the more important objectives. We believe that this paradigm is similar to how human drivers drive, e.g., a human driver would aim at guaranteeing safety before considering other aspects such as traffic rules and comfort.

Most existing RL approaches for autonomous driving consider a state space of either raw visual/sensor input [8, 20], or the kinematics of a few immediately surrounding vehicles [14, 27]. Since road and lane information is not explicitly considered, the policy learned using these types of state space in limited scenarios cannot be expected to be transferable to roads with different geometry. In this work, we design a hybrid (of continuous and discrete) state space that not only includes the state of surrounding vehicles, but also geometry-independent road topology information. We show that the policy trained in a four-way intersection using the proposed state space can be zero-shot transferred to a ring road.

As more surrounding vehicles are considered, the complexity of the problem increases exponentially. However, a human learner is able to reduce complexity by focusing on a few important vehicles in a particular situation, presumably because a human learner exploits some sort of structure of the problem. For example, if a human learner is following a car too close, he not only knows the fact that he *is* following too close, but he also knows: (1) *which car* he is following too close; and (2) the car on the other side of intersection has very little, if anything, to do with the situation. In other words,

---

[1]Data efficiency as measured by the number of training steps required to achieve similar performance.

in addition to viewing the state space as a whole, humans are, at the same time, learning on each individual factored state space (the car ahead, and the car on the other side of intersection, etc.) as well, then they use the knowledge learned on the factored state space to help with the original task. To mimic this behaviour, we propose to decompose *factored MDPs* into auxiliary tasks, then the factored Q functions learned on the factored state space can be used as additional features for the original Q function. This is shown to significantly improve data efficiency.

2 RELATED WORKS

There has been emerging research on autonomous driving using reinforcement learning. Sallab et al. [20] compared two DRL-based end-to-end driving models, DQN and DDPG, in TORCS car racing simulator. Wang and Chan [27] proposed to represent the Q function as a quadratic function to deal with continuous action space. Ngai and Yung attempted multi-objective reinforcement learning for learning takeover maneuver [14], where they scalarized the learned Q functions of each objective by weighted sum to form a single policy. The sensor input was quantized into a discrete state space, and tabular Q-learning was used.

Isele et al. [8] trained a DQN agent that can navigate through an intersection, where the scenario is somewhat similar to ours. However, our scenario is much more complex due to several major differences: (1) Their agent is trained in *one single* scenario each time, and the evaluation is done in the same scenario. Our agent is trained in random scenarios within the map, and evaluated in random scenarios. (2) Their agent only deals with the intersection part. The agent starts at one side of the intersection and all it does is to find a gap to enter, the episode ends as soon as it reaches the other side of the intersection. Our agent needs to drive safely to the intersection and if necessary, slow down, yield and change lanes before entering the intersection. (3) No traffic rules are considered in their work.

There have also been research that exploits the temporal structure of the policy (hierarchical RL [5, 16, 23]). Paxton et al. [17] proposed a method where they designed a set of high level *options* [23], and used Monte Carlo tree search and DDPG [10] to learn the high level and low level policy, respectively.

The idea of thresholded lexicographic ordering can be traced back to the *hierarchical optimization criteria* proposed by Waltz [26] for multi-objective optimization. Gábor et al. [6] extended this idea to reinforcement learning. Wray et al. [28] and Pineda et al. [18] considered an adaptive threshold that depends on the learned value function.

The factored Q functions used in this paper can be thought of as a form of generalized value function proposed by Sutton et al. [22]. The auxiliary POMDPs used to train the factored Q functions can be considered a form of unsupervised auxiliary tasks introduced by Jaderberg et al. [9]. The difference is that we exploit the structure of factored MDP and factorize the *original task itself* into auxiliary tasks, then the learned factored Q functions are used directly as features (basis functions) for the original Q function.

An overview of multi-objective optmization can be found in [11], and a summary of multi-objective sequential decision making is given in [19].

3 BACKGROUND

3.1 Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning (MORL) is concerned with multi-objective Markov decision processes (MOMDPs) $(S, A, P, \mathbf{r}, \gamma)$, where S is a finite set of states; A is a finite set of actions; $P(\mathbf{s}'|\mathbf{s}, a)$ is the transition probability from state $\mathbf{s}$ to state $\mathbf{s}'$ taking action a; $\mathbf{r}(\mathbf{s}, a) = [r_1(\mathbf{s}, a), r_2(\mathbf{s}, a), ..., r_k(\mathbf{s}, a)]$ and $\gamma = [\gamma_1, \gamma_2, ..., \gamma_k]$ are the rewards and discount factors for the k objectives, respectively. [2] Fixing some enumeration from 1 to $|S|$ of the finite state space S, we denote $\mathbf{v}_i^\pi$ as a column vector whose jth element is the value function of the ith objective evaluated at the jth state $\mathbf{s}^j$. Precisely:

$$\mathbf{v}_i^\pi = [v_i^\pi(\mathbf{s}^1), v_i^\pi(\mathbf{s}^2), ..., v_i^\pi(\mathbf{s}^{|S|})]^T$$

$$v_i^\pi(\mathbf{s}) = E[\sum_{t=0}^{t=\infty} \gamma_i^t r_i(\mathbf{s}_t, a_t)|\pi, \mathbf{s}], \forall \mathbf{s} \in S, i = 1, 2, ..., k$$

MORL aims to find some policy $\pi : S \to A$, such that

$$\pi(s) = \text{argmax}_\pi \mathbf{V}^\pi$$

$$\mathbf{V}^\pi = [\mathbf{v}_1^\pi, \mathbf{v}_2^\pi, ... \mathbf{v}_k^\pi]$$

Different definition of order relation on the *feasible criterion space* $\{\mathbf{v}|\pi \in S \to A\}$ leads to different MORL algorithms, or in other words, different MORL algorithm implicitly defines such an (partial or total) order relation. In this paper, we consider a thresholded lexicographic approach that we deem suitable for autonomous driving.

3.2 Thresholded Lexicographic Q-learning

Assuming lexicographic ordering $1, 2, ..., k$ on the k objectives of MOMDP $(S, A, T, \mathbf{r}, \gamma)$, and τ_i a *local* threshold that specifies the minimum admissible value for each objective, thresholded lexicographic Q-learning finds k sets of policies $\Pi_i, i = 1, 2, ..., k$ that maximize $\{\hat{Q}_1^*(\mathbf{s}, a), \hat{Q}_2^*(\mathbf{s}, a), ..., \hat{Q}_i^*(\mathbf{s}, a)\}$ in lexicographic order:

$$\Pi_i \overset{def}{=} \left\{ \pi_i \in \Pi_{i-1} \middle| \pi_i(\mathbf{s}) = \text{argmax}_{a \in \{\pi_{i-1}(\mathbf{s})|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(\mathbf{s}, a) \right.$$
$$\left. , \forall \mathbf{s} \in S \right\}, i = 1, 2, ..., k \tag{1}$$

with Π_0 being the set of all deterministic stationary policies, and $\hat{Q}_i^*(\mathbf{s}, a)$ is the Q function rectified to τ_i:

$$\hat{Q}_i^*(\mathbf{s}, a) \overset{def}{=} \min(\tau_i, Q_i^*(\mathbf{s}, a)) \tag{2}$$

Here, $Q_i^*(\mathbf{s}, a)$ is the maximum expected accumulative reward *over all policies* $\pi_{i-1} \in \Pi_{i-1}$ starting from state $\mathbf{s}$ and action a. It follows that

$$\hat{Q}_i^*(\mathbf{s}, a) =$$

$$\min \left(\tau_i, \quad r_i(\mathbf{s}, a) + \gamma_i \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a) \max_{a' \in \{\pi_{i-1}(\mathbf{s})|\pi_{i-1} \in \Pi_{i-1}\}} Q_i^*(\mathbf{s}', a') \right) \geq$$

$$\min \left(\tau_i, \quad r_i(\mathbf{s}, a) + \gamma_i \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a) \max_{a' \in \{\pi_{i-1}(\mathbf{s})|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(\mathbf{s}', a') \right)$$
$$\tag{3}$$

---

[2]$\mathbf{r}(\mathbf{s}, a)$ can be generalized to $\mathbf{r}(\mathbf{s}, a, \mathbf{s}')$.

Gábor et al. [6] propose to approximate $\hat{Q}_i^*(s, a)$ by treating the inequality in Eq. 3 as equality, and do the following value iteration:

$$\hat{Q}_i^*(s, a) :=$$

$$\min\left(\tau_i, \quad r_i(s, a) + \gamma_i \sum_{s'} P(s'|s, a) \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a')\right) \tag{4}$$

Unfortunately, following a policy $\pi \in \Pi_i$ as in Eq. 1 doesn't guarantee that $v_i^\pi(s) > \tau_i$, even if $\exists \pi' \in \Pi_{i-1}, v_i^{\pi'}(s) > \tau_i, \forall s \in S$. In fact, the policies in Π_i can be arbitrarily bad. Therefore the algorithm is only appropriate for problems that are tolerant to small *local* imperfections of the policy, which needs to be kept in mind when designing the reward function.

3.3 Factored MDP

In almost all real-life domains, the state space of MDPs is very large. Fortunately, many large MDPs have some internal structures. The factored MDP framework aims to exploit these internal structures by representing a state $s \in S$ with a set of *state variables* $s = (s_1, s_2, ..., s_m)$. A *locally-scoped* function is defined as a function that depends only on a subset of the state variables [7], and the subset is called the *scope* of the function. There are mainly two types of structures that are covered in literature: the *additive* structure and *context-specific* structure. As an example of context-specific structure, consider the driving domain: the traffic signal is not affected by the current speed of the cars travelling at the intersection. Moreover, reward function might be a sum of a few locally-scoped rewards, such as the (negative) reward for collision and the reward for maintaining steady speed, which is an example of additive structure. In this paper, a method to exploit these structures in DQN is proposed.

4 APPROACH

4.1 Thresholded Lexicographic DQN

Approximating $\hat{Q}_i^*(s, a)$ directly by Eq. 4 as proposed by Gábor et al. [6] has a few drawbacks, especially in the DQN setting:

(1) Eq. 4 is only an approximate fix point equation for the true $\hat{Q}_i^*(s, a)$, because the inequality in Eq. 3 is arbitrarily replaced by equality.

(2) Since

$$\sum_{s'} P(s'|s, a) \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)$$

is estimated by samples of s', and

$$E_{s' \sim P(s'|s, a)}\left[\min\left(\tau_i, r_i(s, a) + \gamma_i \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)\right)\right]$$

$$\leq \min\left(\tau_i, E_{s' \sim P(s'|s, a)}\left[r_i(s, a) + \gamma_i \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)\right]\right) \tag{5}$$

where θ is the parameter of the function approximator, the estimation is biased, similar to the bias introduced by the max operator in DQN as discussed in [25].

(3) Noise in function approximation can create additional bias due to the `min` operator. Consider the `safety` objective where the reward is -1 when ego vehicle collides, 0 otherwise. Assume that $0 \geq \tau_i \geq -1$, and s is a safe state, so that $\exists A_s \neq \emptyset$ s.t. $\hat{Q}_i^*(s, a) = \tau_i, \forall a \in A_s$. The target for $\hat{Q}_i^*(s, a), a \in A_s$ computed from the right-hand-side of Eq. 4 is

$$\min\left(\tau_i, r_i(s, a) + \gamma_i \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)\right) \leq$$

$$\min\left(\tau_i, \gamma_i \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)\right)$$

For the target to be correct,

$$\gamma_i \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta) \geq \tau_i$$

must hold, which means that:

$$\Delta Q = \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a') - \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta)$$

$$\leq \tau_i - \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} \hat{Q}_i^*(s', a'|\theta) \leq (1 - \frac{1}{\gamma_i})\tau_i$$

where $\hat{Q}_i^*(s', a')$ is the true $\hat{Q}_i^*$ function, and ΔQ is the noise of function approximation. In other words, the noise in neural network must be smaller than $(1 - \frac{1}{\gamma_i})\tau_i$ to avoid creating additional bias. If the look-ahead horizon is long, so that $\gamma_i \approx 1$, the margin is very small.

(4) There's no guarantee the DQN will converge to the true Q value [24], and the learned Q function is empirically very inaccurate. Therefore, using a static threshold τ_i might be problematic, and an adaptive threshold that depends on the learned Q function might be preferrable.

Observe that the only purpose of introducing $\hat{Q}_i^*(s, a)$ is to bring some relaxation to $\max_{a \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} Q_i^*(s, a)$ so that all actions in $\{a \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}|Q_i^*(s, a) \geq \tau_i\}$ are treated as equally 'good enough' for that objective. So instead of estimating $\hat{Q}_i^*(s, a)$, which introduces bias through the `min` operator, we can estimate $Q_i^*(s, a)$ directly through the following Bellman equation:

$$Q_i^*(s, a) = r_i(s, a) + \gamma_i \sum_{s'} P(s'|s, a) \max_{a' \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} Q_i^*(s', a') \tag{6}$$

where Π_i is *redefined* as:

$$\Pi_i \overset{def}{=} \left\{\pi_i \in \Pi_{i-1} \middle| Q_i^*(s, \pi_i(s)) \geq \max_{a \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} Q_i^*(s, a) + \tau_i\right.$$

$$\text{or } \pi_i(s) = \text{argmax}_{a \in \{\pi_{i-1}(s)|\pi_{i-1} \in \Pi_{i-1}\}} Q_i^*(s, a)$$

$$\left. , \forall s \in S\right\} \quad , i = 1, 2, ..., k \tag{7}$$

Note that the fixed threshold has been replaced by an adaptive threshold that depends on the learned Q function, and the algorithm essentially becomes the Q-learning version of *lexicographic value iteration* [18, 28]. Here, τ_i has a different meaning. It specifies how much worse than the best action is considered acceptable in each state. With an adaptive threshold of the form of Eq. 7, it's guaranteed that $\forall \pi \in \Pi_i, v_i^\pi(s) > \max_{\pi' \in \Pi_{i-1}} v_i^{\pi'}(s) + \frac{\tau_i}{1-\gamma}$.

The update rule implied by Eq. 6 for objective $i, i = 1, 2, ..., k$ is similar to Q-learning, except that the next action a' is now restricted to those allowed by objective $i-1$ (In the case of $i = 1$, it degenerates to Q-learning). Once objective $i-1$ converges, it becomes regular Q-learning for an MDP whose action space is dependent on s. During training, one of the objectives $i \in \{1, 2, ..., k\}$ can be chosen for exploration at each simulation step. If objective i is chosen for exploration, objectives $j = i + 1, i + 2, ..., k$ are no longer considered for action selection. The action selection procedure is described in algorithm 1.

Algorithm 1 Action Selection

1: **function** SELECT_ACTION(Q^*, **s**)
    // $Q^* = [Q_1^*, Q_2^*, ..., Q_k^*]$ is the list of
    // learned Q functions for each objective
2:     $A_0(\mathbf{s}) := A$
3:     **for** i in $\{1, 2, ..., k\}$ **do**
4:

$$A_i(\mathbf{s}) := \left\{ a \in A_{i-1}(\mathbf{s}) \middle| Q_i^*(\mathbf{s}, a) \geq \max_{a' \in A_{i-1}} Q_i^*(\mathbf{s}, a') + \tau_i \right.$$

$$\left. \textbf{or } a = \operatorname*{argmax}_{a' \in A_{i-1}} Q_i^*(\mathbf{s}, a') \right\}$$

5:         **if** objective i is chosen to be explored **then**
6:             **return** random action from $A_{i-1}(\mathbf{s})$
7:         **end if**
8:     **end for**
9:     **return** random action from $A_k(\mathbf{s})$
10: **end function**

Since the only interface between objectives is the set of acceptable actions for that objective, not all objectives have to be RL agents (some of them can be rule-based agents), as long as they provide the same interface.

4.2 Factored Q Function

Consider the `safety` objective of self-driving, and the factored representation of state $\mathbf{s} = (s_e, s_1, s_2, ..., s_m)$, where s_e is the state variable for ego vehicle, and $s_1, s_2, ..., s_m$ are the state variables for the surrounding vehicles. Informally, the problem has the following internal structure: (1) collision is *directly* related to only a small subset of vehicles (in most cases, ego vehicle and the vehicle ego is crashing into), so it's natural to view the reward as a function of some locally-scoped rewards $r(\mathbf{s}) = f(r(s_e, s_1), r(s_e, s_2), ..., r(s_e, s_m))$ (2) In some cases, $(s_e|_{t+1}, s_i|_{t+1})$ is only weakly dependent on $s_j|_t, j \neq i$, where $s_i|_t$ denotes the value of s_i at time t. For example, a vehicle on the right-turn lane doesn't have much influence on the next state of a vehicle approaching the intersection from the other side. Formal formulation of what it means by being *'weakly'* dependent, and its effect on the value function, is difficult. However, it's reasonable to hypothesize that these structures result in some kind of structure in the value function. In fact, the task of driving safe can be thought of as the composition of a set of smaller tasks: driving safely with regard to *each individual vehicle*. If we learn how to drive safely *with regard to each individual vehicle*, we can use the knowledge to help with the original task of driving safely. In other

Table 1: State Space — Ego State

s_e	ego state
v_e	ego speed
d_e	distance to intersection
$in\_intersection_e$	whether in intersection
$exist\_left\_lane_e$	whether left lane exists
$exist\_right\_lane_e$	whether right lane exists
$lane\_gap_e$	lateral offset from correct (turning) lane

words, we can use the Q functions of the smaller tasks as auxiliary features for the Q function of the bigger original task. This idea can be formalized as follows.

Viewing $(s_e, s_i), i = 1, 2, ..., m$ as observations from the original factored MDP, and the locally-scoped rewards $r(s_e, s_i)$ as rewards corresponding to the observations, we get a set of m smaller auxiliary (partially observable) MDPs. To exploit the structure of the factored MDP, the Q functions of these smaller MDPs (ignoring the partial observability) can be used as features for the Q function of the original factored MDP. To be more specific, instead of approximating the Q function of the factored MDP $Q^*(\mathbf{s}, a|\theta)$ directly, we learn an approximation of the Q functions of the auxiliary MDPs $Q^*((s_e, s_i), a|\theta_i)$, and use these auxiliary Q functions as additional features $\phi(\mathbf{s}) = [Q^*((s_e, s_1), a|\theta_1), ..., Q^*((s_e, s_m), a|\theta_m)]$ for Q function of the factored MDP. Now the original Q function can be approximated using the augmented feature $(\mathbf{s}, \phi(\mathbf{s}))$, so we have $Q^*((\mathbf{s}, \phi(\mathbf{s})), a|\theta')$. The assumption here is that the additional features $\phi(\mathbf{s})$ will help with the learning of $Q^*(\mathbf{s}, a)$. During training, these factored Q functions in $\phi(\mathbf{s})$ are updated according to their own TD errors. Section 4.4 describes this idea in the context of neural networks.

4.3 State Space

The state space needs to include all the necessary information for driving (vehicle kinematics, road information, etc.), and should be at such an abstraction level that policies learned on a particular road are readily transferable to roads with slightly different geometry. Our state space consists of three parts:

(1) ego state (table 1);
(2) state of surrounding vehicles relative to ego (table 2);
(3) road structure, expressed by topological relations between surrounding vehicles and ego (table 3).

Only a subset of the state variables might be needed for each objective, e.g. the `safety` objective does not need to consider road priority information, since the goal of `safety` is to learn a generic collision avoidance policy.

A maximum of m surrounding vehicles are considered. If there are more vehicles in the scene, only the m closest vehicles are considered. $exist\_vehicle_{1...m}$ is included in the state space in case the number of vehicle is fewer than m. In the experiment of this paper $m = 32$.

In order to deal with complex roads with multiple lanes, topological relations between ego and each surrounding vehicle also need to be included. Inspired by the lanelet model introduced by

Table 2: State Space — Surrounding Vehicles

$s_{1\ldots m}$	surrounding vehicles
$exist\_vehicle_{1\ldots m}$	whether vehicle exists
$v_{1\ldots m}$	relative speed to ego
$d_{1\ldots m}$	distance to intersection
$in\_intersection_{1\ldots m}$	whether in intersection
$exist\_left\_lane_{1\ldots m}$	whether left lane exists
$exist\_right\_lane_{1\ldots m}$	whether right lane exists
$x_{1\ldots m}$, $y_{1\ldots m}$	relative position to ego
$\theta_{1\ldots m}$	relative heading to ego
$has\_priority_{1\ldots m}$	whether has right-of-way over ego
$ttc_{1\ldots m}$	time-to-collision with ego
$signal_{1\ldots m}$	brake/turn signal

Table 3: State Space — Topological Relations with Ego

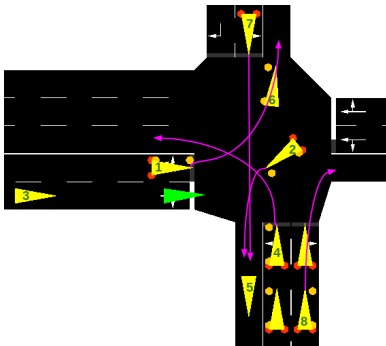|	
merge	merging into the same lane
crossing	routes intersecting each other
left	in left lane
right	in right lane
ahead	ahead in the same or succeeding lane
behind	behind in the same or previous lane
irrelevant	none of the above

Figure 1: Illustration of topological relations. With respect to the green vehicle, vehicle 2, 4, 7 are *crossing*, vehicle 8 is *merge*, vehicle 5 and 6 are *irrelevant*. Vehicle 1 is to the *left* of the green vehicle, and the latter is to the *right* of the former. Vehicle 3 is *behind* the green vehicle, and the latter is *ahead* of the former.

Bender et al. [4], we define seven topological relations between vehicles (table 3), which are illustrated in figure 1. These relations capture the interconnection between roads through vehicles in the scene and their intended path, without explicitly modelling the road structure.

4.4 Network Architecture

The state $s = (s_e, s_1, s_2, \ldots, s_m) \in S$ contains the state variables of m surrounding vehicles s_i, $i = 1, 2, \ldots, m$ (including their topological
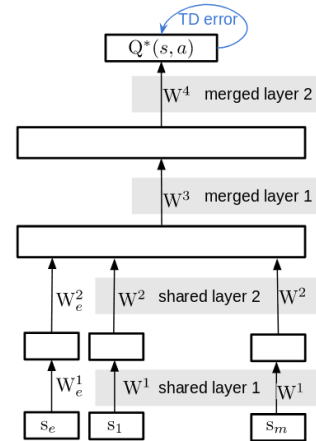

Figure 2: Neural network architecture with built-in invariance to the order of surrounding vehicles in the state. We first pass $s_1, s_2, \ldots, s_m$ through a few shared layers to get the corresponding features. Then these features are merged through addition and activation. After that, the network is fully-connected.

relations with ego). Since swapping the order of two surrounding vehicles in the state doesn't change the scene, the Q value should remain the same:

$$Q((s_e, s_1, \ldots, s_i^u, \ldots, s_j^v, \ldots, s_m), a) = Q((s_e, s_1, \ldots, s_j^v, \ldots, s_i^u, \ldots, s_m), a)$$

where s_i^u denotes the uth possible instantiation of dom(s_i). To build this invariance into the neural network, the network needs to be symmetric with respect to each s_i. In other words, the weights connecting $Q(s, a)$ to each s_i should be the same (shown in Figure 2).

If factored Q function is used, then m additional heads for these value functions are needed (Figure 3). During each update, m Q functions are improved simultaneously in addition to the original Q function, each of which corresponds to learning to avoid collision with each of the m surrounding vehicles, in the case of the `safety` objective. Since the agent utilizes a single scene to learn multiple aspects within the scene, better data efficiency can be expected.

5 EXPERIMENT

SUMO (Simulation of Urban Mobility [2]) traffic simulator is used as the simulation environment for our experiment. We wrote a RL interface similar to OpenAI Gym on top SUMO to provide the state and action space. [3] Given a map, the set of all possible routes a vehicle can travel is predefined. The vehicle needs to control throttle and lane change behavior. The action space is a discrete set of 9 actions:

(1) `max_deceleration`;
(2) `med_deceleration`;
(3) `min_deceleration`;
(4) `maintain_speed`;
(5) `min_acceleration`;
(6) `med_acceleration`;
(7) `max_acceleration`;
(8) `change_to_right_lane`;
(9) `change_to_left_lane`

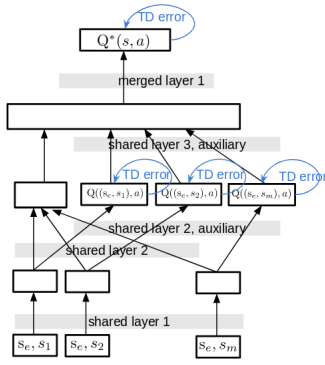[3]Source code can be found at https://gitlab.com/sumo-rl/sumo_openai_gym

Figure 3: Neural network architecture with factored Q function. The shared layer branches off for the factored Q functions (auxiliary branch), which is merged back in higher layers.

The vehicle kinematics follows a point-mass model, and lane changes are instantaneous. Speed is assumed to be maintained during lane changes. The learning scenario is a typical urban four-way intersection of a major road and a minor road with random traffic. Traffic coming from the minor road needs to yield to the major road, and turns need to be made from the correct lane(s).

5.1 Objectives

We consider four objectives in this paper. In lexicographic order, the objectives are:

(1) `lane_change`: rule-based; all it does is to rule out invalid lane change actions, namely: lane change to the left/right when there's no left/right lane, and lane change in intersections. The state space only has three state variables: $exist\_left\_lane_e$, $exist\_right\_lane_e$ and $in\_intersection_e$; thus, it's trivial to learn even if it were implemented as a RL agent.

(2) `safety`: RL-based; it ensures that collision doesn't happen. -1 reward if collision occurs, or if time-to-collision with *at least one* surrounding vehicle is less than 3s and is still decreasing; 0 reward otherwise. [4] The state space includes everything except $lane\_gap_e$ and $has\_priority_{1...m}$. Factored Q functions are learned on the auxiliary MDPs

$$(\text{dom}(s_e, s_i), A, \gamma, r_i), i = 1, 2, ..., m$$

Where r_i is just the locally-scoped version of r: -1 if ego collides with vehicle i or the time-to-collision with *vehicle i* is less than 3s and is still decreasing; 0 reward otherwise. Since up to $m = 32$ vehicles are considered, up to 32 instances of auxiliary POMDPs (which share the state space with the original factored MDP) can be running at the same time. If vehicle i goes out of scene or crashes with ego vehicle, the episode ends for instance i of the auxiliary task. Adaptive

---

[4]This is only a simplified description of the actual reward used. Since we use a simple calculation for time-to-collision, sometimes it's not suitable to make the reward dependent on the (inaccurate) estimates of time-to-collision. In these cases, the reward is set to 0. For the intricacies of the reward function, please refer to the source code.

threshold is used, and τ is set to -0.2 during training; then it's manually fine-tuned on the training set before testing.

(3) `regulation`: RL-based; it makes sure that traffic rules are followed. We consider two traffic rules: (a) to make turns from the correct lane(s); (b) to yield according to right-of-way. A reward of -1 is given for failure to yield right-of-way, -0.02 for failure to proceed when having right-of-way, and up to -1 for staying in the wrong lane (e.g. staying in the left-turn lane, if the assigned route is straight). The state space is comprised of $has\_priority_{1...m}$, $lane\_gap_e$, $in\_intersection_e$, v_e and d_e. Change of right-of-way or change of road is considered end of episode, since these changes would happen regardless of the actions chosen. τ is set to -0.2 during training.

(4) `comfort&speed`: rule-based; prefers acceleration unless speed limit is reached, while avoiding extreme actions (e.g. maximum acceleration) and lane changes.

5.2 Training

The agent is trained on two intersecting roads with random surrounding traffic. Traffic enters the scene with a random probability in each episode. An episode ends either when ego collides with other vehicle(s) or when the timeout is reached. Each surrounding vehicle has a normally distributed maximum speed, and is controlled by SUMO's rule-based behavioral model, which attempts to mimic human drivers. The intersection part of the map is shown in Figure 1. The north/south-bound traffic needs to yield to the east/west-bound traffic. In each episode, ego vehicle is randomly assigned one of the possible routes within the map. Each RL-based objective is trained using double DQN [25] with prioritized experience replay [21]. To speed up training, 10 simulation instances run in parallel, adding experience to the experience replay buffer. Asynchronous [12] update is performed on the Q functions of each objective.

Three models are trained for comparison, which we later refer to as **DQN**, **TLDQN**, and **TLfDQN** respectively:

(1) Scalar-valued DQN: The neural network architecture is as shown in Figure 2, with 4 shared layers and 2 merged layers. Each layer has 64 hidden units. The reward function is a weighted sum of the rewards used for the multi-objective case. The weights are chosen in a way that try to reflect the relative importance of each objective.

(2) Thresholeded lexicographic DQN: The `safety` objective uses the same neural network architecture as above. The `regulation` objective uses a 4-layer fully connected network with 64 hidden units in each layer.

(3) Thresholded lexicographic DQN with factored Q function: The `safety` objective uses the neural network architecture as shown in Figure 3, but with only the auxiliary branch. The auxiliary branch has 4 shared layers, each with 64 hidden units; the merged layer is a fixed `min` layer that takes the minimum of the factored Q functions for each action. $Q(\mathbf{s}, a|\theta) = \min_i Q((s_e, s_i), a|\theta)$ The `regulation` objective uses the same network structure as above.
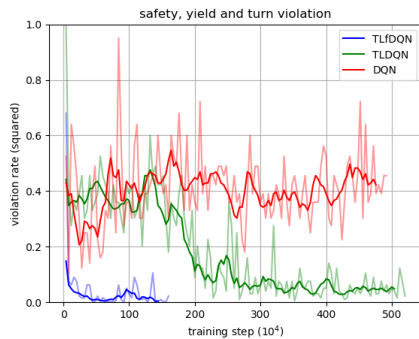
Figure 4: Learning curve of DQN, TLDQN and TLfDQN. The dark curves are the moving averages.

5.3 Results

The three models are first evaluated on the same intersecting roads they've been trained on, with random traffic; then their zero-shot transfer performance is evaluated on a ring road (Figure 8) they've never seen during training. The vehicle can enter the ring road through either right or left turn. Traffic entering the ring road needs to yield to traffic already on the ring road.

Figure 4 shows the learning curve of DQN, TLDQN and TLfDQN. The x-axis is the training step, and the y-axis is the (squared) rate of safety (collisions) and traffic rule (yielding and turning) violations combined. Timeouts are counted as yielding violations. TLDQN and DQN are trained for 30 hours, while TLfDQN is only trained for 18 hours since it has already converged. We see that TLfDQN is able to reach a good policy within $500,000$ training steps, as compared to 3 million training steps for TLDQN, improving the data efficiency by 6 times. It should be noted that the training time of TLfDQN per training step is longer than TLDQN (26 minutes as compared to 14 minutes), mostly due to the computational overhead of the 32 additional targets for the factored Q functions, one for each surrounding vehicle in the scene. However, the overhead can potentially be alleviated by parallelizing the computation of the target. Within 30 hours of training, scalar-valued DQN is not able to learn an acceptable policy, indicating the effectiveness of the multi-objective approach. Different weightings for the objectives in the reward function were tried for scalar-valued DQN, no significantly better result was observed. [5]

Figure 5 shows the learning curves with a breakdown of different types of violation. Ideally, we would like to show how the agent performs on each objective. However, many violations are inter-correlated, e.g., safety violations are usually preceded by failure to yield; improperly stopping in the middle of the road leads to low safety violation rate; high safety violation rate often leads to lower turning violation rate, because the agent simply collides before even reaching the intersection. Therefore, we group the more serious violations — safety, failure to yield and timeouts, into one category; and the less serious violation — failure to change to correct lane, into another category. The blue curves show the first category, and

Table 4: Violation Rate after 30 **Hours of Training**

Model	Collision	Yielding	Turning
DQN	32.9%	8.5%	16.4%
TLDQN	10.9%	0.9%	7.6%
TLfDQN	3.6%	1.0%	2.4%
TLfDQN (transfer)	3.5%	0.4%	N/A

the green curves show both categories. Note that failure to change to correct lane doesn't necessary imply a bad policy, because in some scenarios, the road is just too crowded for lane changes. We see in the figure that in both categories, TLfDQN performs the best. It's worth noting that it might seem that the scalar-valued DQN briefly achieves better performance before getting worse. However, the videos indicate that the lower collision rate is due to the agent learning an incorrect policy that stops abruptly in the middle of the road and waits until all the traffic clears before moving.

Videos of the learned policy of our multi-objective RL agent can be found online [6]. Figure 6 and Figure 7 are some snapshots of the videos. Ego vehicle is colored as green, and vehicles that have right-of-way over ego are colored as orange. In Figure 6, the ego vehicle is assigned a left-turning route, so it needs to first change to the left lane, then take a left turn. The ego vehicle learns to slow down (notice the braking lights) until a gap is found, and then change lane to the left. In Figure 7, the ego vehicle slows down to yield for traffic on the major road, then proceeds to complete the left turn after the road is clear. The vehicle is not yielding for the right-turning vehicle because there's no conflict between them. Figure 8 shows the zero-shot transfer performance on a ring road. The agent is able to drive through the ring road safely and yield to traffic already on the ring road before entering. The performance of the three models after 30 hours of training evaluated on $1,000$ random episodes is shown in Table 4.

6 CONCLUSIONS

We've shown in the context of autonomous urban driving that a multi-objective RL approach — thresholded lexicographic DQN might be effective for problems whose objectives are difficult to express using a scalar reward function due to the many aspects involved. The proposed method of learning factored Q functions, and using them as auxiliary features during training is shown to improve data efficiency. Combining these ideas, we trained an autonomous driving agent that is, most of the time, able to safely drive on busy roads and intersections, while following traffic rules.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Machine Learning, Proceedings of the Twenty-first*

---

[5]A good weighting scheme for the reward might exist, but nevertheless hard to find; and to test a set of new weights, the agent has to be re-trained.

[6]https://www.youtube.com/playlist?list=PLiZsfe-Hr4k9VPiX0tfoNoHHDUE2MDPuQ
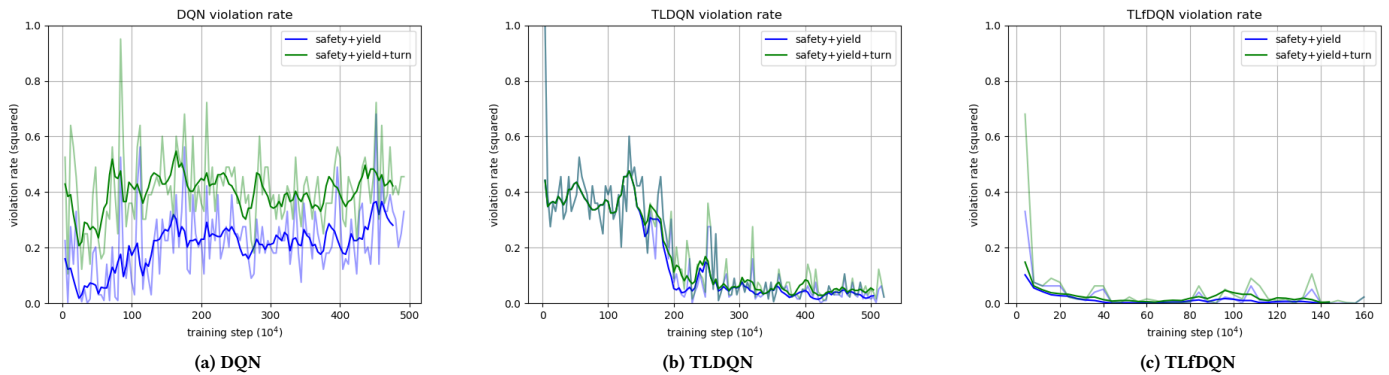
(a) DQN

(b) TLDQN

(c) TLfDQN

Figure 5: Learning curves showing different types of violations. The blue curves show collisions, timeouts and failures to yield combined; the green curves show collisions, timeouts, failures to yield and failures to change to correct lane for turning combined

(a) need to change to left-turn lane

(b) slow down to find a gap

(c) successful lane change
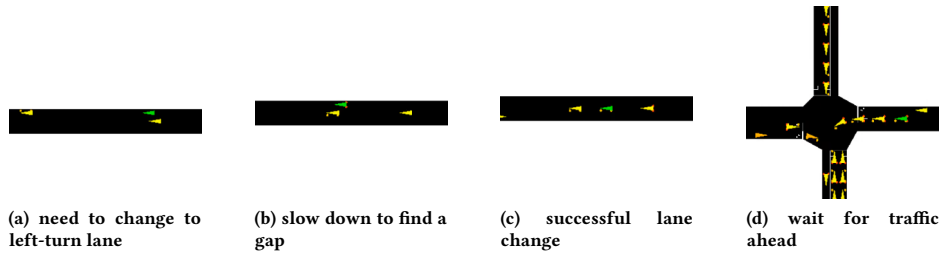
(d) wait for traffic ahead

Figure 6: Ego vehicle (green) is assigned a left-turning route in this episode. The agent slows down to find a gap and successfully changes to the left-turn lane.
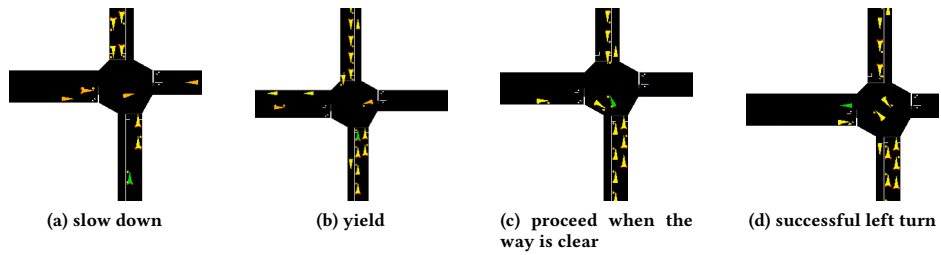

(a) slow down

(b) yield

(c) proceed when the way is clear

(d) successful left turn

Figure 7: The agent (green) yields to the traffic on the main road, and then proceeds when it has right-of-way.

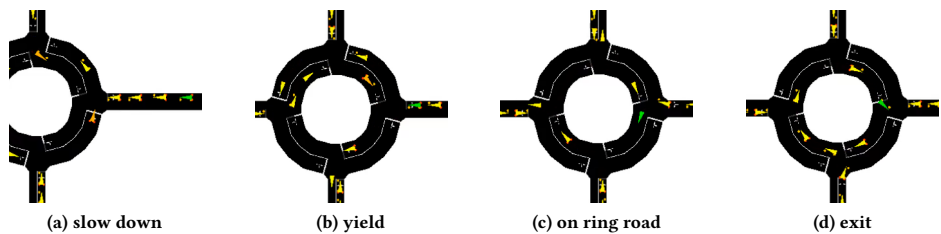(a) slow down

(b) yield

(c) on ring road

(d) exit

Figure 8: Zero-shot transfer performance. The agent (green) is able to drive through the ring road without safety or traffic rule violation.

International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004. https://doi.org/10.1145/1015330.1015430

[2] Michael Behrisch, Daniel Krajzewicz, and Melanie Weber (Eds.). 2014. *Simulation of Urban Mobility - First International Conference, SUMO 2013, Berlin, Germany, May 15-17, 2013. Revised Selected Papers.* Lecture Notes in Computer Science, Vol. 8594. Springer. https://doi.org/10.1007/978-3-662-45079-6

[3] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. 2016. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain.* 1471–1479. http://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation

[4] Philipp Bender, Julius Ziegler, and Christoph Stiller. 2014. Lanelets: Efficient map representation for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, June 8-11, 2014.* 420–425. https://doi.org/10.1109/IVS.2014.6856487

[5] Thomas G. Dietterich. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *J. Artif. Intell. Res.* 13 (2000), 227–303. https://doi.org/10.1613/jair.639

[6] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. 1998. Multi-criteria Reinforcement Learning. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998.* 197–205.

[7] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. 2003. Efficient Solution Algorithms for Factored MDPs. *J. Artif. Intell. Res.* 19 (2003), 399–468. https://doi.org/10.1613/jair.1000

[8] David Isele, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. 2017. Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning. *CoRR* abs/1705.01196 (2017). arXiv:1705.01196 http://arxiv.org/abs/1705.01196

[9] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement Learning with Unsupervised Auxiliary Tasks. *CoRR* abs/1611.05397 (2016). arXiv:1611.05397 http://arxiv.org/abs/1611.05397

[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015). arXiv:1509.02971 http://arxiv.org/abs/1509.02971

[11] R.T. Marler and J.S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization* 26, 6 (01 Apr 2004), 369–395. https://doi.org/10.1007/s00158-003-0368-6

[12] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016.* 1928–1937. http://jmlr.org/proceedings/papers/v48/mniha16.html

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[14] Daniel C. K. Ngai and Nelson Hon Ching Yung. 2007. Automated Vehicle Overtaking based on a Multiple-Goal Reinforcement Learning Framework. In *20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017.* 818–823. https://doi.org/10.1109/ITSC.2007.4357682

[15] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information*

Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. 4026–4034. http://papers.nips.cc/paper/6501-deep-exploration-via-bootstrapped-dqn

[16] Ronald Parr and Stuart J. Russell. 1997. Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997].* 1043–1049. http://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines

[17] Chris Paxton, Vasumathi Raman, Gregory D. Hager, and Marin Kobilarov. 2017. Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017.* 6059–6066. https://doi.org/10.1109/IROS.2017.8206505

[18] Luis Enrique Pineda, Kyle Hollins Wray, and Shlomo Zilberstein. 2015. Revisiting Multi-Objective MDPs with Relaxed Lexicographic Preferences. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015.* 63–68. http://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11678

[19] Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2014. A Survey of Multi-Objective Sequential Decision-Making. *CoRR* abs/1402.0590 (2014). arXiv:1402.0590 http://arxiv.org/abs/1402.0590

[20] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2016. End-to-End Deep Reinforcement Learning for Lane Keeping Assist. *CoRR* abs/1612.04340 (2016). arXiv:1612.04340 http://arxiv.org/abs/1612.04340

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized Experience Replay. *CoRR* abs/1511.05952 (2015). arXiv:1511.05952 http://arxiv.org/abs/1511.05952

[22] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. 2011. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3.* 761–768. http://portal.acm.org/citation.cfm?id=2031726&CFID=54178199&CFTOKEN=61392764

[23] Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[24] J. N. Tsitsiklis and B. Van Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Automat. Control* 42, 5 (May 1997), 674–690. https://doi.org/10.1109/9.580874

[25] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR* abs/1509.06461 (2015). arXiv:1509.06461 http://arxiv.org/abs/1509.06461

[26] Frederick M. Waltz. 1967. An engineering approach: hierarchical optimization criteria. *IEEE Trans. Automat. Control* 12, 2 (1967), 179–180.

[27] Pin Wang and Ching-Yao Chan. 2018. Autonomous Ramp Merge Maneuver Based on Reinforcement Learning with Continuous Action Space. *CoRR* abs/1803.09203 (2018). arXiv:1803.09203 http://arxiv.org/abs/1803.09203

[28] Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. 2015. Multi-Objective MDPs with Conditional Lexicographic Reward Preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* 3418–3424. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9471

[29] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008.* 1433–1438. http://www.aaai.org/Library/AAAI/2008/aaai08-227.php