Autonomous Driving: A Multi-Objective Deep Reinforcement Learning Approach

by

Changjian Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

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. It's representative of complex reinforcement learning tasks humans encounter in real life. The aim of this thesis is to explore the effectiveness of multi-objective reinforcement learning for such tasks characterized by autonomous driving. In particular, it shows that:

- Multi-objective reinforcement learning is effective at overcoming some of the difficulties faced by scalar-reward reinforcement learning, and a multi-objective DQN agent based on a variant of thresholded lexicographic Q-learning is successfully trained to drive on multi-lane roads and intersections, yielding and changing lanes according to traffic rules.
- Data efficiency of (multi-objective) reinforcement learning can be significantly improved by exploiting the factored structure of a task. Specifically, factored Q functions learned on the factored state space can be used as features to the original Q function to speed up learning.
- Inclusion of history-dependent policies enables an intuitive exact algorithm for multi-objective reinforcement learning with thresholded lexicographic order.

Acknowledgements

I would like to thank my supervisor Prof. Krzysztof Czarnecki, for the patient guidance, encouragement and advice he has provided throughout my master's studies. I would also like to thank Sean Sedwards, Jaeyoung Lee, and my colleagues at Waterloo Intelligent System Engineering Lab (WISELab) for discussions and feedback.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Deep reinforcement learning [33] has seen some success in complex tasks with large state space. From Atari [33] to the ancient game of Go [52], reinforcement learning (RL) offers an intriguing promise: the agent learns how to behave through trial-and-error with the environment, without human experts explicitly specifying how the task is to be achieved. It is therefore natural to ask the question of how to build a RL agent for real-world tasks such as autonomous driving, where it is potentially infeasible for humans to program the optimal behavior. These tasks are especially challenging, partially because they entails many, sometimes conflicting, aspects. For example, in autonomous driving, 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 comes from the difficulty of 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 infer the reward function from demonstrations (inverse RL [1, 66]), supplying sufficient amount of good demonstrations for (all the corner cases) might not be feasible. Another motivation comes from the problem of exploration [6, 36] in RL. The multi-objective approach endows an agent the flexibility of choosing which aspect of the task to explore. For example, if an autonomous driving 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 the safety aspect is learned separately from traffic rules, the agent has the potential of exploring traffic rules among the safe actions only. We therefore consider a multi-objective RL approach to the problem of urban autonomous 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 other advantages of the multi-objective approach:

- 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, accelerating learning.
- In a new task where only some of the objectives change, the learned policy for other objectives can be reused or transferred.

1.2 Overview

Reinforcement learning dates back to the research on optimal control, psychology and neuro-science [23]. Over the years, two main classes of algorithms have been developed for reinforcement learning, namely the value function approaches and the policy gradient approaches. Value function approaches maintain an estimate of the optimal *action-value function* (Q function) — the maximum expected cumulative reward starting from a state taking a particular action, and the optimal policy is implied by the action that maximize the Q function for each state. Well known approaches of this class include Q-learning and SARSA. However, due to the 'curse of dimensionality', success had mostly been limited to simple toy-domain scenarios until the introduction of *deep Q network* (DQN) [33] where the classic Q-learning algorithm was combined with deep neural networks. This was followed by many improvements, including double DQN, dueling networks, *deep deterministic policy gradient* (DDPG), prioritized experience replay, etc. The first policy gradient algorithm, REINFORCE, was first introduced by Williams [63]. The algorithm suffers from high variance of gradient, and is slow to learn in practice. Several algorithms have been proposed to alleviate the issue, including actor-critic algorithms [27], natural policy gradient [24], and more recent approaches such as trust region policy optimization (TRPO) [48] and proximal policy optimization (PPO) [47]. Apart from the two main classes of algorithms, evolution strategies (ES) [42, 53] have recently been shown to be a promising direction for reinforcement learning.

Multi-objective reinforcement learning [41] was preceded by early research on multi-objective MDP by White [62]. Instead of having a scalar reward as in (single-objective) reinforcement learning, in multi-objective reinforcement learning, the agent receives a vector reward at each time step, with each dimension for one criterion of the task. Since the value function is now also a vector, a partial/total order needs to be defined. Different definitions of partial/total order lead to different multi-objective reinforcement learning algorithms. In single-policy methods, a total order is defined, and the goal is to find the maximum value function. In multi-policy methods, the order relation is partial, so not all realizable value functions can be compared. The goal is thus to find the *frontier* (typically the Pareto frontier) of the set of all realizable value functions.

There has been emerging research on autonomous driving using RL. Sallab et al. [43] compared two deep RL-based end-to-end driving models, DQN and DDPG, in TORCS car racing simulator. Wang and Chan [60] proposed to represent the Q function as a quadratic function to deal with continuous action space. Ngai and Yung attempted multi-objective RL for learning takeover maneuver [35], 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. [21] trained a DQN agent that can navigate through an intersection. However, these research either considers a scenario where safety is not critical (car racing game [43]), or a specific maneuver in a very specific scenario (highway ramp merge in [60], takeover maneuver in [35], and intersection in [21]). Furthermore, none of these works consider traffic rules. The limitations of these agents suggest that the standard Markov decision process (MDP) formulation of RL might not capture the structure of the tasks in an effective way. Several research has been focusing on reducing the complexity by exploiting structure of the problem, which includes: 1. hierarchical RL [37, 54, 11] that uses the temporal structure of the task; 2. factored MDPs [8] that aim to leverage *context-specific* and *additive* structure of the MDP; 3. multi-objective RL that exploits the structure of the agent's goals. The first of them — hierarchical RL — has been explored in the context of autonomous driving by Paxton et al. [38], where they designed a set of high level *options* [54], and used Monte Carlo tree search and DDPG [29] to learn the high level and low level policy, respectively. The thesis demonstrates how the other two directions — factored MDPs and multi-objective RL — can be adapted and applied to the task of autonomous driving. Specifically:

- Chapter 2 introduces the relevant background;
- Chapter 3 describes the proposed approach, *thresholded lexicographic factored DQN*, which combines multi-objective RL with ideas from factored MDP;
- Chapter 4 shows the corresponding experimental result, where the proposed agent successfully learns to drive on multi-lane roads and intersections, yielding and changing lanes according to traffic rules.

- Chapter 5 revisits thresholded lexicographic Q learning, and analyzes its limitations. Two new algorithms, one model-based, one model-free, are proposed.

Chapter 2

Background

This chapter lays out the foundations for later chapters. We start with Makov Decision Process — the commonly adopted mathematical framework for stochastic sequential decision making problems, and review its exact and approximate solution methods. Then we briefly go through RL and multi-objective RL, in particular, thresholded lexicographic Q-learning.

2.1 Markov Decision Process

A finite (single-objective) *Markov decision process (MDP)* can be represented by a 5-tuple (S, A, P, r, γ), where S is a finite set of states; A is a finite set of actions; $P(\mathbf{s}'|\mathbf{s}, a) \in [0, 1]_{\mathbb{R}}$ is the transition probability from state $\mathbf{s}$ to state $\mathbf{s}'$ taking action a; $r(\mathbf{s}, a, \mathbf{s}') \in \mathbb{R}$ is reward for taking action a in state $\mathbf{s}$ and ending up in $\mathbf{s}'$; and γ is the discount factor. A *policy* π tells the agent what action to take at each time step. In the most general case, a policy can be history-dependent and random, in the form of $\pi = (\pi^0, \pi^1, ..., \pi^t, ...)$, where a *decision rule* $\pi^t(h^t, \mathbf{s}, a) \in [0, 1]_{\mathbb{R}}$ is the probability of taking action a in state $\mathbf{s}$ with history h^t. A history [1] is a sequence of past states, actions and decision rules $h^t = (\mathbf{s}^0, a^0, \pi^0, \mathbf{s}^1, a^1, \pi^1, ..., \mathbf{s}^{t-1}, a^{t-1}, \pi^{t-1})$. A policy is said be *deterministic* if $\pi^t(h^t, \mathbf{s}, a) = 1$ for only one action, in which case we can use a simplified notation $\pi = (d^0, d^1, ..., d^t, ...)$, where $d^t(h^t, \mathbf{s}) \in A$. Correspondingly, if the policy is deterministic, a history can be represented with $h^t = (\mathbf{s}^0, d^0, \mathbf{s}^1, d^1, ..., \mathbf{s}^{t-1}, d^{t-1})$. A policy is said to be *stationary* if the

[1] This definition of history considers decision rules in addition to actions, which makes it slightly different from the usual definition in literature.

decision rule only depends on the current state $\mathbf{s}$, and does not change with time, i.e., $\pi^t(h^t, \mathbf{s}, a) = \pi(\mathbf{s}, a)$. The set of all history-dependent random policies, history-dependent deterministic policies, stationary random policies, and stationary deterministic policies are denoted by Π_{HR}, Π_{HD}, Π_{SR} and Π_{SD}, respectively.

The goal of a MDP is to find the policies that maximizes the expected discounted cumulative reward [2]:

$$\max_{\pi \in \Pi} E[\sum_{t=0}^{\infty} \gamma^{(t)} r^t | \pi, \mathbf{s}^{t=0} = \mathbf{s}], \mathbf{s} \in S \tag{2.1}$$

where $\Pi \in \{\Pi_{\text{HR}}, \Pi_{\text{HD}}, \Pi_{\text{SR}}, \Pi_{\text{SD}}\}$ denotes the policies we would like to consider. We call a task an *episodic* task with *horizon* T if the state space is augmented with time; $\gamma = 1$; and $r(s, a, s') = 0, \forall t \geq T$.

To measure of how good a policy is, it is convenient to define the *value function* and the *action-value function*. The value function of a policy π is defined as the expected cumulative reward starting from state $\mathbf{s}$ and following policy π:

$$v^{\pi}(\mathbf{s}) \overset{def}{=} E[\sum_{t=0}^{t=\infty} \gamma^{(t)} r^t | \pi, \mathbf{s}^{t=0} = \mathbf{s}], \mathbf{s} \in S \tag{2.2}$$

The action-value function (or *Q function*) of a policy π is defined as the expected cumulative reward starting from state $\mathbf{s}$ and action a, then following policy π:

$$Q^{\pi}(\mathbf{s}, a) \overset{def}{=} E[\sum_{t=0}^{\infty} \gamma^{(t)} r^t | \pi, \mathbf{s}^{t=0} = \mathbf{s}, a^{t=0} = a], \mathbf{s} \in S, a \in A \tag{2.3}$$

The optimal value function, optimal policy and optimal Q function are denoted by v^*, π^*, and Q^*, respectively:

$$v^*(\mathbf{s}) \overset{def}{=} \max_{\pi \in \Pi} v^{\pi}(\mathbf{s}), \forall \mathbf{s} \in S \tag{2.4}$$

$$\pi^* \overset{def}{=} \arg\max_{\pi \in \Pi} v^{\pi}(\mathbf{s}), \forall \mathbf{s} \in S \tag{2.5}$$

$$Q^*(\mathbf{s}, a) \overset{def}{=} E[\sum_{t=0}^{\infty} \gamma^{(t)} r^t | \pi^*, \mathbf{s}^{t=0} = \mathbf{s}, a^{t=0} = a], \mathbf{s} \in S, a \in A \tag{2.6}$$

[2] The (t) at the upper right corner of γ represents exponential.

For single-objective MDPs, there exists a stationary deterministic optimal policy [40]. That is

$$\exists \pi \in \Pi_{\mathrm{SD}}, \forall s \in S, v^{\pi}(s) = \arg\max_{\pi' \in \Pi_{\mathrm{HR}}} v^{\pi'}(s) \tag{2.7}$$

Unless otherwise noted, we assume a policy is stationary deterministic.

2.1.1 Value Iteration

There are mainly three classes of exact solution methods for MDPs: value iteration, policy iteration and linear programming. All three algorithms assume complete knowledge of the MDP — the rewards and the transition probabilities. Here, we only go through value iteration, which forms the foundation for Q-learning in the next section. Value iteration is an iterative method that start with some arbitrary estimate of value function. At each step, we update the estimate according to the *Bellman equation*:

$$v^*(\mathbf{s}) = \max_a \sum_{\mathbf{s}' \in S} P(\mathbf{s}'|\mathbf{s}, a)\left[r(\mathbf{s}, a, \mathbf{s}') + \gamma v^*(\mathbf{s}')\right], \forall \mathbf{s} \in S \tag{2.8}$$

So the algorithm works as follows:

Algorithm 1 Value Iteration

```
for s in $S$ do
    $\underline{v}^*(\mathbf{s}) := 0$
end for
while stopping condition is not met do
    for s in $S$ do
        $\underline{v}^*(\mathbf{s}) := \max_a \sum_{\mathbf{s}' \in S} P(\mathbf{s}'|\mathbf{s}, a)\left[r(\mathbf{s}, a, \mathbf{s}') + \gamma \underline{v}^*(\mathbf{s}')\right]$
    end for
end while
```

where := denotes assignment. Throughout the thesis, we use underline to represent the current estimate of the corresponding variable without an underline. For example, $\underline{v}^*(\mathbf{s})$ denotes the current estimate of $v^*(\mathbf{s})$. If a parametric function approximator is used, $\underline{v}^*_\theta(\mathbf{s})$ or $\underline{v}^*(\mathbf{s}|\theta)$ denotes the current estimate of $v^*(\mathbf{s})$ with parameter θ.

2.1.2 Factored MDP

The complexity of the above exact solution methods increases exponentially with the number of state variables. Fortunately, many large MDPs have some internal structures. The factored MDP framework aims to exploit these internal structures by explicitly representing a state $\mathbf{s} \in S$ with a set of *state variables* $\mathbf{s} = (s_1, s_2, ..., s_m)$. For a set $C \subseteq \{1, 2, ..., m\}$, $\mathbf{s}[C]$ denotes the subset of state variables in $\mathbf{s}$ with index that belongs to C. A *locally-scoped* function is defined as a function that depends only on a subset of the state variables [16], and the subset is called the *scope* of the function. There are mainly two types of structures discussed in literature: the *context-specific* structure and *additive* structure. Context-specific structure refers to the conditional independence between the state variables. For example, $\mathbf{s}^{t+1}[C]$ might only depend on a subset $\mathbf{s}^t[C']$ of $\mathbf{s}^t$, this subset is called the parents of $\mathbf{s}[C]$, and is denoted by $\mathbf{s}[\Gamma(C)]$.

$$P(\mathbf{s}^{t+1}[C]|\mathbf{s}^t, a^t) = P(\mathbf{s}^{t+1}|\mathbf{s}^t[\Gamma(C)], a^t), \forall \mathbf{s}^t, \mathbf{s}^{t+1} \in S, \forall a^t \in A \tag{2.9}$$

This kind of conditional dependency can be graphically described by a *dynamic Bayesian network* (DBN). The other structure — the additive structure refers to the fact that the reward function might be the sum of a few locally-scoped rewards:

$$r(\mathbf{s}, a, \mathbf{s}') = \sum_i r_i(\mathbf{s}[\Gamma(C_i)], a, \mathbf{s}'[C_i]) \tag{2.10}$$

We may hope the value function to have similar structure as well. However, such structure is, in general, not preserved in the value function. Nevertheless, we can still take advantage of these structures to derive approximate algorithms. Here, we consider an approximate value iteration method where the value function is approximated by a linear combination of a set of k basis functions $\{h_1(\mathbf{s}), h_2(\mathbf{s}), ..., h_k(\mathbf{s})\}$:

$$\underline{v}^*_{\mathbf{w}}(\mathbf{s}) = \sum_{i=1}^{k} w_i h_i(\mathbf{s}) \tag{2.11}$$

Denoting $\mathbf{h}_i$ as a column vector whose mth element is the h_i evaluated at the m^{th} state ${}^m\mathbf{s}$, i.e. $h_i({}^m\mathbf{s})$, and $\mathbf{H}$ as a matrix whose columns are $\mathbf{h}_i$, we can compactly express the above equation as $\underline{\mathbf{v}}^*_{\mathbf{w}} = \mathbf{Hw}$. According to the Bellman equation, we need to find $\mathbf{w}$ such that:

$$\mathbf{w} = \arg\min_{\mathbf{w}} ||\mathbf{Hw} - \max_{\mathbf{a}}(\text{diag}(\mathbf{P^a R^a}) + \gamma \mathbf{P^a H w})|| \tag{2.12}$$

where $\mathbf{max}$ is element-wise maximization, and $\mathbf{a}$ is a vector whose mth element is the action that maximize the mth element of $\text{diag}(\mathbf{P^a R^a}) + \gamma \mathbf{P^a H w}$. $\mathbf{P^a}$ is a matrix with $\mathbf{P}^{\mathbf{a}}_{mn} = P({}^n\mathbf{s}|{}^m\mathbf{s}, \mathbf{a}_m)$, and $\mathbf{R}^{\mathbf{a}}_{nm} = r({}^m\mathbf{s}, \mathbf{a}_m, {}^n\mathbf{s})$.

As with exact value iteration, the approximate version also greedily improves $\mathbf{v}$ in each iteration:

$$\underline{\mathbf{v}}^* := \max_{\mathbf{a}} \{\text{diag}(\mathbf{P^a R^a}) + \gamma \mathbf{P^a H w}\} \tag{2.13}$$

The difference is that there is an extra step that projects $\underline{\mathbf{v}}^*$ back to the linear space spanned by $\mathbf{H}$.

$$\mathbf{w} := \arg\min_{\mathbf{w}} ||\mathbf{Hw} - \underline{\mathbf{v}}^*|| \tag{2.14}$$

To utilized the additive and context-specific structure, *factored value iteration* considers the case where $h_i(\mathbf{s})$ are locally-scoped, i.e. $h_i(\mathbf{s}) = h_i(\mathbf{s}[C_i])$. By exploiting the conditional independence as in Eq. 2.9, each element of the right-hand-side of Eq. 2.13 can be simplified as:

$$\begin{aligned}
&\max_a \sum_{s'} P(\mathbf{s}'|\mathbf{s}, a) \left[\sum_i r_i(\mathbf{s}[\Gamma(C_i)], a, \mathbf{s}'[C_i]) + \gamma \sum_i w_i h_i(\mathbf{s}'[C_i]) \right] \\
=&\max_a \sum_i \sum_{\mathbf{s}'[C_i]} P((\mathbf{s}'[C_i]|\mathbf{s}[\Gamma(C_i)], a) \Big[r_i(\mathbf{s}[\Gamma(C_i)], a, \mathbf{s}'[C_i]) + \gamma w_i h_i(\mathbf{s}'[C_i]) \Big]
\end{aligned} \tag{2.15}$$

Note that the summation over $\mathbf{s}'$ has been reduced to $\mathbf{s}'[C_i]$, and the results for states $\mathbf{s}$ with the same $\mathbf{s}[\Gamma(C_i)]$ can be reused. The above is just one example of how the structure of the MDP can be exploited with this factored representation. Although there are many other methods for factored MDP, the essence stays the same.

2.2 Reinforcement Learning

The MDP solution methods outlined in the previous section assume knowledge of the environment model in the form of transition probabilities and rewards. Reinforcement learning, on the other hand, does not make such assumptions. Of course, the models can be learned first through execution, so that the methods in the previous section can be applied, which leads to *model-based RL*. However, it might be non-trivial to learn and solve the exact model. In contrast, *model-free RL* learns the optimal policy directly without learning the model, which usually leads to simpler algorithms.

2.2.1 Q-learning

Q-learning is a popular model-free RL algorithm. It works by updating the Q function according to sampled Bellman error (*temporal difference error* or *TD error*) in each step. Denoting the TD error by δ:

$$\delta(\mathbf{s}, a) = r(\mathbf{s}, a, \mathbf{s}') + \gamma \max_a \underline{Q}^*(\mathbf{s}', a) - \underline{Q}^*(\mathbf{s}, a) \tag{2.16}$$

where $\mathbf{s}' \sim P(\mathbf{s}'|\mathbf{s}, a)$, Q-learning updates the current Q estimate so that the TD error is reduced:

$$\underline{Q}^*(\mathbf{s}, a) := \underline{Q}^*(\mathbf{s}, a) + \alpha\delta(\mathbf{s}, a) \tag{2.17}$$

where α is the learning rate. Since $\delta(\mathbf{s})$ is an unbiased estimate of the true Bellman error

$$\sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a) \left[r(\mathbf{s}, a, \mathbf{s}') + \gamma \max_a \underline{Q}^*(\mathbf{s}', a) - \underline{Q}^*(\mathbf{s}, a) \right] \tag{2.18}$$

If all the states are visited infinitely often, Eq. 2.17 is, in the long term, equivalent to value iteration, and if α decays at a proper rate, it can be proved that Eq. 2.17 will converge to the correct Q value.

With function approximation where $\underline{Q}^*_\theta(\mathbf{s}, a)$ is parameterized by θ, we can no longer directly apply Eq. 2.17. Our goal, however, is still to push $\underline{Q}^*_\theta(\mathbf{s}, a)$ in the direction that the TD error in Eq. 2.16 is reduced. This can be achieved by applying gradient descent on $\frac{1}{2}\delta^2$:

$$\begin{aligned} \nabla_\theta \left[\frac{1}{2}\delta^2 \right] &= \delta\nabla_\theta \left[r(\mathbf{s}, a, \mathbf{s}')) + \gamma \max_a \underline{Q}^*(\mathbf{s}', a) - \underline{Q}^*_\theta(\mathbf{s}, a) \right] \\ &= -\delta\nabla_\theta \underline{Q}^*_\theta(\mathbf{s}, a) \end{aligned} \tag{2.19}$$

So the update rule now becomes:

$$\theta := \theta + \alpha\delta\nabla_\theta \underline{Q}^*_\theta(\mathbf{s}, a) \tag{2.20}$$

It is important to notice that in Eq. 2.19 we ignored the fact that $\underline{Q}^*_\theta(\mathbf{s}', a)$ is also parameterized by θ, and treated it as fixed. Therefore the gradient in Eq. 2.19 is not the full gradient, and it is instead called semi-gradient.

Vanilla Q-learning with function approximation is often unstable in practice, and several techniques have been introduced to alleviate the issue:

- The convergence result for stochastic gradient descent relies on the assumption that the data is independent and identically distributed. However, the samples in Q learning are temporally correlated — the distribution over the next state $\mathbf{s}'$ depends heavily on the current state $\mathbf{s}$. *Experience Replay* attempts to break the correlation between samples by saving the transition tuple $(\mathbf{s}, a, \mathbf{s}', r)$ in an experience replay buffer. In each Q update, a mini-batch of experiences are sampled randomly from the replay buffer. This has the additional benefit of being able to reuse past experience, thus improving sample efficiency.

- Another source of instability for Q learning with function approximation is the constantly moving *target* $r(\mathbf{s}, a, \mathbf{s}') + \gamma \max_a \underline{Q}^*_\theta(\mathbf{s}', a)$. When θ is updated, the change in the learned Q function is not restricted to that specific state and action, $\underline{Q}^*_\theta(\cdot, \cdot)$ changes for all states and actions. This results in θ chasing a target that is itself moving according to θ. To alleviate this issue, Mnih et al. [33] introduced a separate *target network* that is parameterized by another set of parameters θ', which is used for computing the target. The target network is synchronized with the action-selection network from time to time, so that the target is still correct, but not moving too fast.

The TD error (Eq. 2.16) is an unbiased estimate of the Bellman error (Eq. 2.18) only if $\underline{Q}^*(\mathbf{s}', a)$ is considered as constant. If $\underline{Q}^*(\mathbf{s}', a)$ is considered as a random variable, however, the estimation is biased, because the $\max_a Q^*(\mathbf{s}', a)$ in Eq. 2.18 should now be written as $\max_a E[\underline{Q}^*(\mathbf{s}', a)]$, while Eq. 2.16 is actually an estimator for $E[\max_a \underline{Q}^*(\mathbf{s}', a)]$, and

$$E[\max_a \underline{Q}^*(\mathbf{s}', a)] \geq \max_a E[\underline{Q}^*(\mathbf{s}', a)] \tag{2.21}$$

Q-learning with function approximation therefore tends to suffer from overestimation, where considering $\underline{Q}^*(\mathbf{s}', a)$ as a random variable is more appropriate. Double Q-learning [17, 18] solves this by introducing a second estimator for $\underline{Q}^*(\mathbf{s}, a)$. The `max` over actions $a \in A$ taken using one estimator, but the value is taken from the other estimator. This results in an estimator that underestimates Eq. 2.18 rather than overestimating it.

2.2.2 Generalized Value Function

The value function we discussed above is directly related to learning the optimal policy for the problem at hand, which is basically what the agent needs to do. However, interacting with the environment provides much more knowledge beyond the control policy alone, that might or might not be directly useful for solving the task. For example, in a task where a robot is asked to pick up an apple from the floor, it does not matter whether the robot learns how to jump, so the knowledge of how to jump is not useful for this particular task. However, in a related task where the apple is hanged on the ceiling, the knowledge becomes useful. Such knowledge (as jumping in the example) has a separate reward function and termination condition from the original task, and thus has a separate value function associated with it, and the associated value function is called a *generalized value function* [55]. A generalized value function can be used in various ways, but the idea is that value functions based on pseudo-reward functions on top the original task, can be helpful for the task or some similar tasks.

2.3 Multi-Objective Reinforcement Learning

In some cases [41], it is preferable to consider different aspects of a task separately as different objectives. *Multi-objective RL* is concerned with *multi-objective Markov decision processes* (multi-objective MDPs) $(S, A, P, \mathbf{r}, \boldsymbol{\gamma}, \geq_v)$, where S, A and $P(\mathbf{s}'|\mathbf{s}, a)$ are respectively the state space, action space, and the transition probability as in single-objective MDPs; Now the rewards $\mathbf{r}(\mathbf{s}, a, \mathbf{s}') = [r_1(\mathbf{s}, a, \mathbf{s}'), r_2(\mathbf{s}, a, \mathbf{s}'), ..., r_k(\mathbf{s}, a, \mathbf{s}')]$ and discount factors $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_k]$ are vectors, the i^{th} element for the i^{th} objective. We denote $\mathbf{v}_i^\pi$ as a column vector whose jth element is the value function of the ith objective evaluated at the j^{th} state $\mathbf{s}^j$. To be precise:

$$\begin{aligned} \mathbf{v}_i^\pi &= [v_i^\pi({}^1\mathbf{s}), v_i^\pi({}^2\mathbf{s}), ..., v_i^\pi({}^{|S|}\mathbf{s})]^T, i = 1, 2, ..., k \\ v_i^\pi(\mathbf{s}) &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma_i^{(t)} r_i(\mathbf{s}^t, a^t, \mathbf{s}^{t+1})|\pi, \mathbf{s}^{t=0} = \mathbf{s}], \forall s \in S \end{aligned} \tag{2.22}$$

Multi-objective RL aims to find some policy $\pi \in \Pi_{\text{SD}}$, such that

$$\begin{aligned} \pi^*(s) &= \arg\max_{\pi \in \Pi} \mathbf{V}^\pi \\ \mathbf{V}^\pi &= [\mathbf{v}_1^\pi, \mathbf{v}_2^\pi, ...\mathbf{v}_k^\pi] \end{aligned} \tag{2.23}$$

$\geq_v$ is a partial order defined on the space of value functions $\{\mathbf{V}^\pi | \pi \in \Pi\}$. It's worth noting that the max in the arg max here refers to maximal elements [3] (not the maximum) . Different choice of order relations leads to different multi-objective RL algorithms. In this thesis, we adopt a thresholded lexicographic approach that we deem suitable for autonomous driving. However, as we shall see in Chapter 5, thresholded lexicographic Q-learning does not

[3] a maximal element of a subset X of some partially ordered set is an element of X that no other element in X is larger

strictly fit into the framework of Eq. 2.22 and Eq. 2.23, and shall only be treated as an approximate algorithm.

Although for single-objective MDPs, the optimal value can be attained by a deterministic stationary policy, this is in general not true for multi-objective MDPs. White [62] showed that history-dependent deterministic policies can dominate stationary deterministic policies; Chatterjee, Majumdar, and Henzinger [9] proved for the case $\gamma_1 = \gamma_2 = ... = \gamma_k$ that there exists a stationary random optimal policy. Although random policies can dominate deterministic policies, they do not necessarily add value to the solution as far as a single trial is concerned. Suppose that we have two policies π and π' for a task with two objectives, with value function $[v_1^{\pi}, v_2^{\pi}]$ and $[v_1^{\pi'}, v_2^{\pi'}]$, respectively. By mixing π and π' with probability α and $1-\alpha$, the value function of the mixture policy is $[\alpha v_1^{\pi} + (1-\alpha)v_1^{\pi'}, \alpha v_2^{\pi} + (1-\alpha)v_2^{\pi'}]$. Depending on the partial order $\geq_v$, the mixture policy might in expectation 'outperforms' the pure policies over multiple trials. However, as far as one trial is concerned, the executed policy is either π or π', so no better result is gained. In this thesis, we restrict our discussion to deterministic policies.

2.3.1 Thresholded Lexicographic Q-learning

Assuming lexicographic ordering $1, 2, ..., k$ on the k objectives of multi-objective MDP $(S, A, P, \mathbf{r}, \gamma)$, and τ_i a *local* threshold that specifies the minimum admissible Q value for each objective, thresholded lexicographic Q-learning finds k sets of policies $\hat{\Pi}_i, i = 1, 2, ..., k$ that maximize

$$\{\hat{Q}_1^{\hat{\Pi}_0}(\mathbf{s}, a), \hat{Q}_2^{\hat{\Pi}_1}(\mathbf{s}, a), ..., \hat{Q}_i^{\hat{\Pi}_{k-1}}(\mathbf{s}, a)\} \tag{2.24}$$

in lexicographic order:

$$\hat{\Pi}_i \overset{def}{=} \left\{ \pi_i \in \hat{\Pi}_{i-1} \middle| \pi_i(\mathbf{s}) = \text{argmax}_{a \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} \hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) \quad , \forall \mathbf{s} \in S \right\}, i = 1, 2, ..., k \tag{2.25}$$

with $\hat{\Pi}_0 \overset{def}{=} \Pi_{\text{SD}}$ being the set of all deterministic stationary policies, and $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ is the Q function rectified to τ_i:

$$\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) \overset{def}{=} \min(\tau_i, Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)) \tag{2.26}$$

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

$$\begin{aligned}\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) = \min\left(\tau_i, \quad \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)\Big[r_i(\mathbf{s}, a, \mathbf{s}') + \gamma_i \max_{a' \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}', a')\Big]\right) \geq \\ = \min\left(\tau_i, \quad \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)\Big[r_i(\mathbf{s}, a, \mathbf{s}') + \gamma_i \max_{a' \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} \hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}', a')\Big]\right)\end{aligned} \tag{2.27}$$

Gábor, Kalmár, and Szepesvári [13] propose to approximate $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ by treating the inequality in Eq. 2.27 as equality, and do the following value iteration:

$$\underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}, a) := \min\left(\tau_i, \quad \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)\Big[r_i(\mathbf{s}, a, \mathbf{s}') + \gamma_i \max_{a' \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \underline{\hat{\Pi}}_{i-1}\}} \underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}', a')\Big]\right) \tag{2.28}$$

which we refer to as *thresholded lexicographic Q-learning* [4]. The framework allows some relaxation to each objective so that other objectives can be considered, which intuitively resembles how humans balance different aspects of a task.

[4] We made some slight changes from the original formulation given in [13].

Chapter 3

Methods

3.1 Thresholded Lexicographic DQN

In many applications, we have several objectives with different priorities. We would like to guarantee the more important objective to a certain degree before considering other objectives. Autonomous driving is one such application: we would like to guarantee safety before considering other objectives such as traffic rules; and only among the policies that follow the traffic rules, we consider even less important aspects such as passenger comfort, etc. Thresholded lexicographic learning follows a similar procedure. However, approximating $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ (Eq. 2.26) directly by Eq. 2.28 as proposed by Gábor, Kalmár, and Szepesvári [13] has a few drawbacks, especially in the DQN setting:

1. Eq. 2.28 is only an approximate fix point equation for the true $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$, because the inequality in Eq. 2.27 is arbitrarily replaced by equality.

2. Since

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

is estimated by samples of $\mathbf{s}'$, and

$$\begin{aligned} &E_{\mathbf{s}'\sim P(\mathbf{s}'|\mathbf{s},a)}\left[\min\left(\tau_i, r_i(\mathbf{s},a,\mathbf{s}') + \gamma_i \max_{a'\in\{\pi_{i-1}(\mathbf{s})|\pi_{i-1}\in\underline{\hat{\Pi}}_{i-1}\}} \underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}',a'|\theta)\right)\right] \\ &\leq \min\left(\tau_i, E_{\mathbf{s}'\sim P(\mathbf{s}'|\mathbf{s},a)}\left[r_i(\mathbf{s},a,\mathbf{s}') + \gamma_i \max_{a'\in\{\pi_{i-1}(\mathbf{s})|\pi_{i-1}\in\underline{\hat{\Pi}}_{i-1}\}} \underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}',a'|\theta)\right]\right) \end{aligned} \tag{3.1}$$

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 [18].

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 $\mathbf{s}$ is a safe state, so that $\exists A_s \neq \emptyset$ s.t. $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s},a) = \tau_i, \forall a \in A_s$. The target for $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s},a), a \in A_s$ computed from the right-hand-side of Eq. 2.28 is

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

For the target to be correct,

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

must hold, which means that:

$$\begin{aligned} \Delta Q &= \max_{a'\in\{\pi_{i-1}(\mathbf{s})|\pi_{i-1}\in\hat{\Pi}_{i-1}\}} \hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}',a') - \max_{a'\in\{\pi_{i-1}(\mathbf{s})|\pi_{i-1}\in\underline{\hat{\Pi}}_{i-1}\}} \underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}',a'|\theta) \\ &\leq \tau_i - \max_{a'\in\{\pi_{i-1}(\mathbf{s})|\pi_{i-1}\in\underline{\hat{\Pi}}_{i-1}\}} \underline{\hat{Q}}_i^{\underline{\hat{\Pi}}_{i-1}}(\mathbf{s}',a'|\theta) \leq (1-\frac{1}{\gamma_i})\tau_i \end{aligned}$$

where $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}',a')$ is the true $\hat{Q}_i^{\hat{\Pi}_{i-1}}$ 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 is no guarantee the DQN will converge to the true Q value [58], and the learned Q-value are 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 $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ is to bring some relaxation to $\max_{a \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ so that all actions in $\{a \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\} | Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) \geq \tau_i\}$ are treated as equally 'good enough' for that objective. So instead of estimating $\hat{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$, which introduces bias through the `min` operator, we can estimate $Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ directly through the following Bellman equation:

$$Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) = r_i(\mathbf{s}, a, \mathbf{s}') + \gamma_i \sum_{s'} P(\mathbf{s}' | \mathbf{s}, a) \max_{a' \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}', a') \tag{3.2}$$

where $\hat{\Pi}_i$ is *redefined* as:

$$\hat{\Pi}_i \overset{def}{=} \left\{ \pi_i \in \hat{\Pi}_{i-1} \middle| Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, \pi_i(\mathbf{s})) \geq \max_{a \in \{\pi_{i-1}(\mathbf{s}) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) - \tau_i \right\}, i = 1, 2, ..., k \tag{3.3}$$

As in the previous chapter, $\hat{\Pi}_0 \overset{def}{=} \Pi_{\text{SD}}$, $Q_i^{\Pi_{i-1}}(\mathbf{s}, a)$ is a shorthand for $\max_{\pi \in \Pi_{i-1}} Q_i^{\pi}(\mathbf{s}, a)$, and $v_i^{\Pi_{i-1}}(\mathbf{s})$ is a shorthand for $\max_{\pi \in \Pi_{i-1}} v_i^{\pi}(\mathbf{s})$. 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* [64, 39]. Here, we restrict τ_i to be non-negative, and thus give it a different meaning: it specifies how much worse than the best action is considered acceptable in each state.

The update rule implied by Eq. 3.2 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 2.

Algorithm 2 Action Selection

```
function SELECT_ACTION($\underline{Q}, \mathbf{s}$)
// $\underline{Q} = [\underline{Q}_1^{\hat{\Pi}_0}, \underline{Q}_2^{\hat{\Pi}_1}, ..., \underline{Q}_k^{\hat{\Pi}_{k-1}}]$ is the list of
// learned Q functions for each objective
    $A_0(\mathbf{s}) := A$
    for $i$ in $\{1, 2, ..., k\}$ do
        $A_i(\mathbf{s}) := \left\{ a \in A_{i-1}(\mathbf{s}) \middle| \underline{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a) \geq \max_{a' \in A_{i-1}} \underline{Q}_i^{\hat{\Pi}_{i-1}}(\mathbf{s}, a') - \tau_i \right\}$
        if objective $i$ is chosen to be explored then
            return random action from $A_{i-1}(\mathbf{s})$
        end if
    end for
    return random action from $A_k(\mathbf{s})$
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.

3.2 Factored Q Function

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, 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.

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 is 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 does not have much influence on the next state of a vehicle approaching the intersection from the opposite side. Formal formulation of what it means by being *'weakly'* dependent, and its effect on the value function, is difficult. However, it is 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 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 estimating the Q function of the factored MDP $Q^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$ directly, we learn an estimation of each of the Q functions of the auxiliary MDPs $Q^{\hat{\Pi}_{i-1}}((s_e, s_i), a)$, and use these auxiliary Q functions as additional features $\phi(\mathbf{s}) = [Q^{\hat{\Pi}_{i-1}}((s_e, s_1), a), ..., Q^{\hat{\Pi}_{i-1}}((s_e, s_m), a)]$ for estimating the Q function of the factored MDP. Now the original Q function can be approximated using the augmented feature $(\mathbf{s}, \phi(\mathbf{s}))$. The assumption here is that the additional features $\phi(\mathbf{s})$ will help with the learning of $Q^{\hat{\Pi}_{i-1}}(\mathbf{s}, a)$. These factored Q functions in $\phi(\mathbf{s})$ essentially fall into the framework of generalized value functions, and are updated according to their own TD errors during training. Section 3.4 describes this idea in the context of neural networks.

3.3 State Space

Most existing RL approaches for autonomous driving consider a state space of either raw visual/sensor input [43, 21], or the kinematics of a few immediately surrounding vehicles [35, 60]. 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. To overcome this limitation, the state space not only needs to include all the necessary information for driving (vehicle kinematics, road information, etc.), but should also 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 3.1);
2. state of surrounding vehicles relative to ego (table 3.2);
3. road structure, expressed by topological relations between surrounding vehicles and ego (table 3.3).

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

Table 3.1: 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

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

Table 3.2: 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
$brake_{1\ldots m}$	brake signal
$left\_turn_{1\ldots m}$	left turn signal
$right\_turn_{1\ldots m}$	right turn signal

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 Bender, Ziegler, and Stiller [7], we define seven topological relations between vehicles (table 3.3), which are illustrated in figure 3.1. These relations capture the interconnection between roads through vehicles in the scene and their intended path, without explicitly modelling the road structure.

Table 3.3: 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

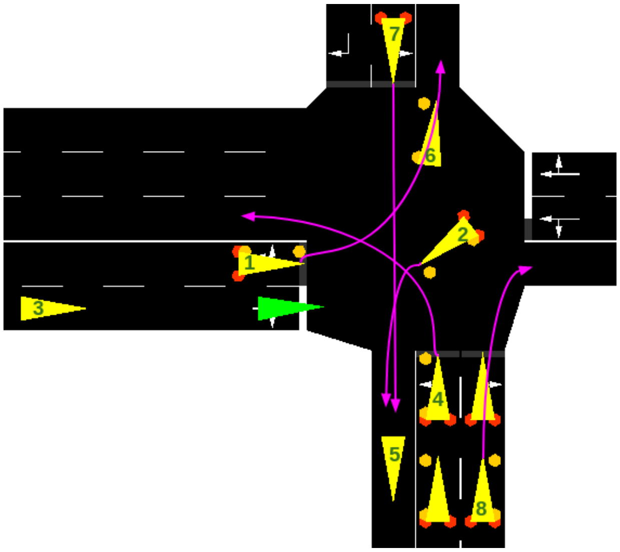


Figure 3.1: Illustration of topological relations. With respect to the green vehicle, vehicle $2, 4, 7$ are *crossing*; vehicle 8 is *merge*; vehicle 5 and 6 are *irrelevant*. Vehicle 1 is to the *left* of the green vehicle, and the latter is to the *right* of the former. Vehicle 3 is *behind* the green vehicle, and the latter is *ahead* of the former.

3.4 Network Architecture

The state $s = (s_e, s_1, s_2, ..., s_m) \in S$ contains the state variables of m surrounding vehicles $s_i, i = 1, 2, ..., m$ (including their topological relations with ego). Since swapping the order of two surrounding vehicles in the state does not change the scene, the Q value should remain the same:

$$Q((s_e, s_1, ..., {}^u s_i, ..., {}^v s_j, ..., s_m), a) = Q((s_e, s_1, ..., {}^v s_j, ..., {}^u s_i, ..., s_m), a)$$

where ${}^u s_i$ denotes the uth possible instantiation of $\mathrm{dom}(s_i)$. 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 3.2). The loss function at time step t is the usual TD loss:

$$L^t(\theta) = E_{s,a \sim \rho(\cdot)}[r(\mathbf{s}) + \gamma \max_{a'} \underline{Q}_\theta(\mathbf{s}', a') - \underline{Q}_\theta(\mathbf{s}, a)] \tag{3.4}$$

where $\rho(s, a)$ is the probability distribution (s, a) pair.

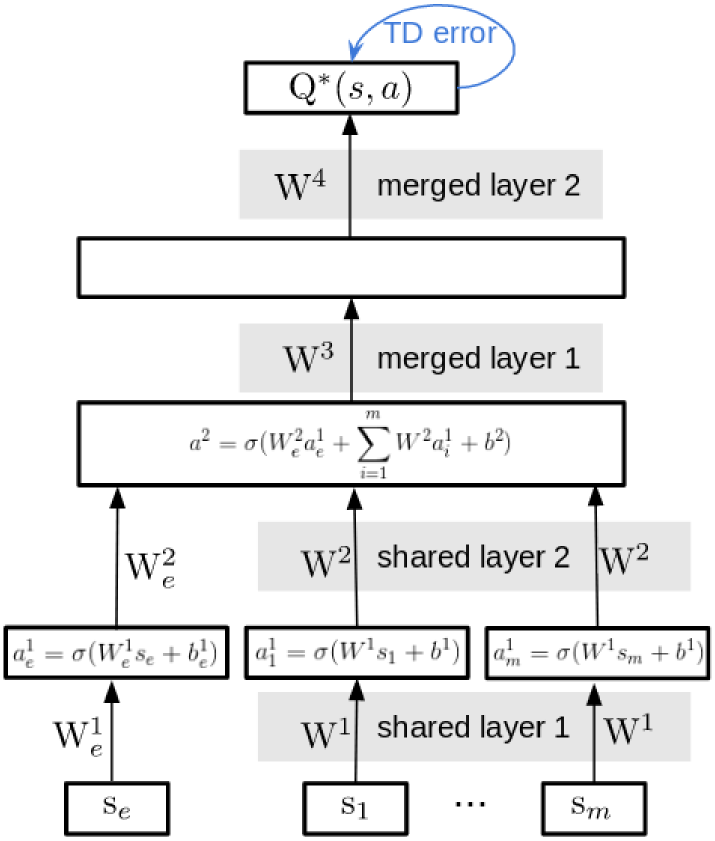

TD error
Q*(s, a)
W4
merged layer 2
W3
merged layer 1
$a^2 = \sigma(W_e^2 a_e^1 + \sum_{i=1}^{m} W^2 a_i^1 + b^2)$
W2e
W2
shared layer 2
W2
$a_e^1 = \sigma(W_e^1 s_e + b_e^1)$
$a_1^1 = \sigma(W^1 s_1 + b^1)$
$a_m^1 = \sigma(W^1 s_m + b^1)$
W1e
W1
shared layer 1
W1
se
s1
...
sm

Figure 3.2: Neural network architecture with built-in invariance to the order of surrounding vehicles in the state. We first pass $s_1, s_2, ..., s_m$ 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.

If factored Q function is used, then m additional heads for these value functions are needed (Figure 3.3). 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. The loss function for each auxiliary task is thus $L_i^t(\theta) = E_{s,a \sim \rho(\cdot)}[r(s_e', s_i') + \gamma \max_{a'} \underline{Q}_{\theta}((s_e', s_i'), a') - \underline{Q}_{\theta}((s_e', s_i'), a)]$. Since the agent utilizes a single scene to learn multiple aspects within the scene, better data efficiency can be expected.

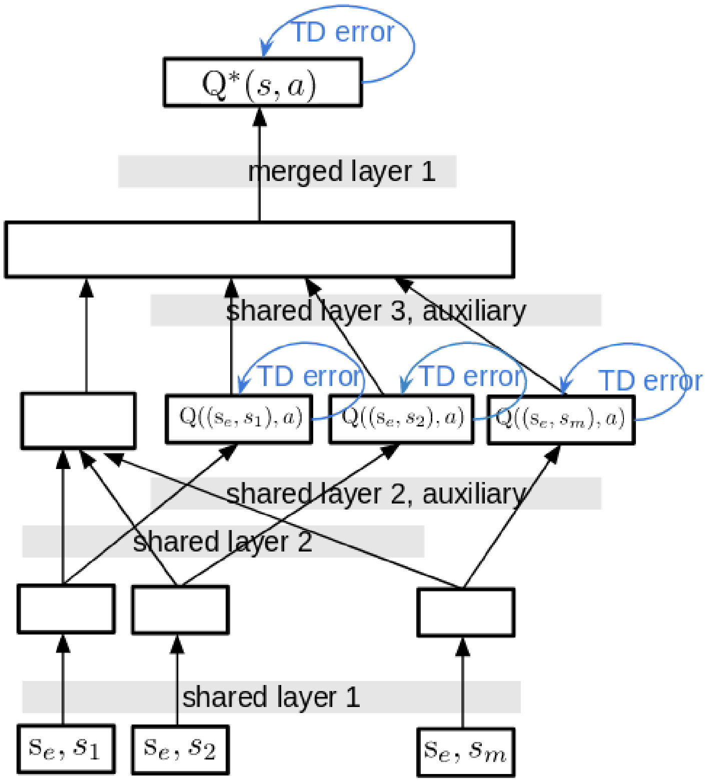


Figure 3.3: Neural network architecture with factored Q function. The shared layer branches off for the factored Q functions (auxiliary branch), which is merged back in higher layers.

Chapter 4

Experiments

4.1 Environment

SUMO (Simulation of Urban Mobility [5]) traffic simulator is used as the simulation environment for our experiment. A RL interface similar to OpenAI Gym is written on top SUMO to provide the state and action space. [1] 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`

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).

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

4.2 Objectives

We consider four objectives in this thesis. 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 is no left/right lane, and lane change in intersections. The state space only has three state variables: $\mathit{exist\_left\_lane}_e$, $\mathit{exist\_right\_lane}_e$ and $\mathit{in\_intersection}_e$; thus, it is trivial to learn even if it were implemented as a RL agent.

2. `safety`: RL-based; it ensures that collision does not 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. [2] The state space includes everything except $\mathit{lane\_gap}_e$ and $\mathit{has\_priority}_{1..m}$. Factored Q functions are learned on the auxiliary MDPs

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

   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 threshold is used, and τ is set to 0.2 during training; then it is 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 $\mathit{has\_priority}_{1...m}$, $\mathit{lane\_gap}_e$, $\mathit{in\_intersection}_e$, 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.

[2] This is only a simplified description of the actual reward used. Since we use a simple calculation for time-to-collision, sometimes it is 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.

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

4.3 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 when ego collides with other vehicle(s), when ego goes out of scene, 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 3.1. 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 [18] with prioritized experience replay [46]. To speed up training, 10 simulation instances run in parallel, adding experience to the experience replay buffer. Asynchronous [32] 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 3.2, 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 3.3, 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.

4.4 Results

The three models are first evaluated on the same intersecting roads they have been trained on, with random traffic; then their zero-shot transfer performance is evaluated on a ring road (Figure 4.5) they have 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 4.1 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. [3]

Figure 4.2 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 the green curves show both categories. Note that failure to change to correct lane does not 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 is worth noting that it might seem that the scalar-valued DQN briefly achieves better performance before getting worse. However, the videos indicate that the

[3] 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.

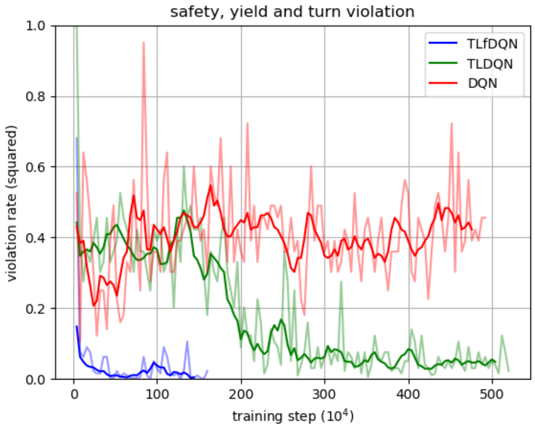


Figure 4.1: Learning curve of DQN, TLDQN and TLfDQN. The dark curves are the moving averages.

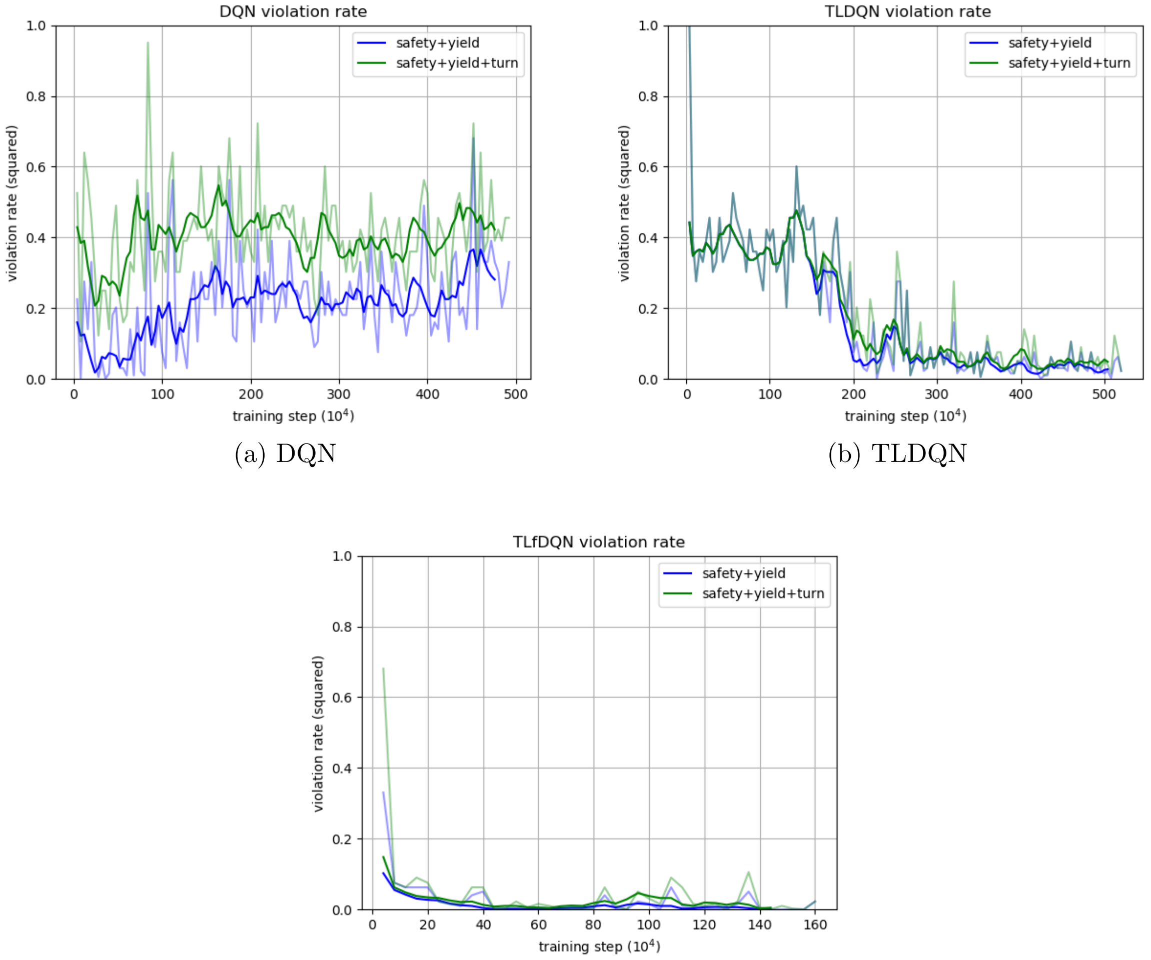
Table 4.1: Violation Rate after 30 Hours of Training

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

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 [4]. Figure 4.3 and Figure 4.4 are some snapshots from the videos. Ego vehicle is colored as green, and vehicles that have right-of-way over ego are colored as orange. In Figure 4.3, 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 4.4, 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 is no conflict between them. Figure 4.5 shows the zero-shot transfer performance on a ring road.

[4] https://www.youtube.com/playlist?list=PLiZsfe-Hr4k9VPiX0tfoNoHHDUE2MDPuQ



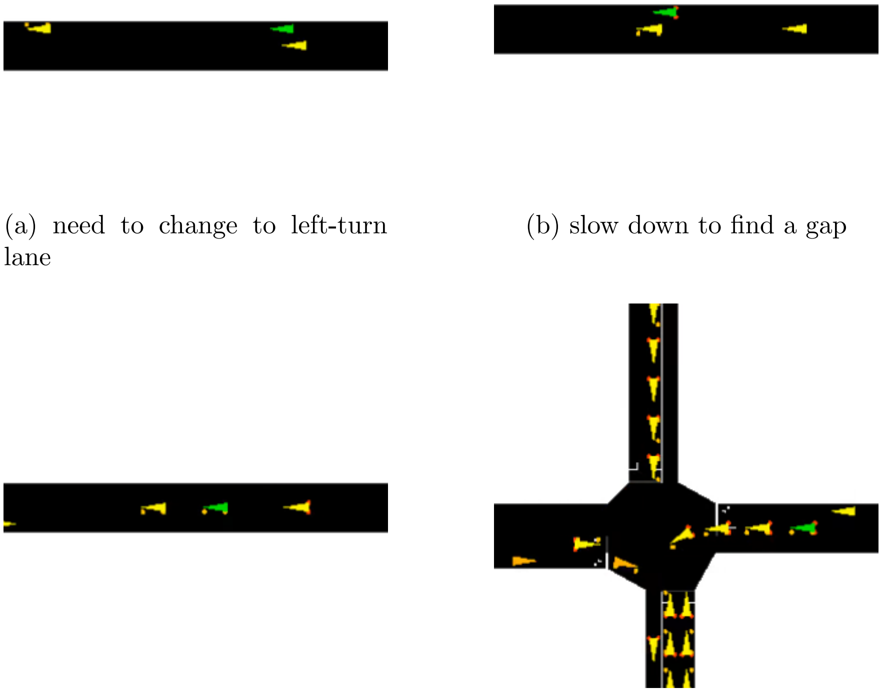
(a) DQN

(b) TLDQN

(c) TLfDQN

Figure 4.2: 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

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 4.1.



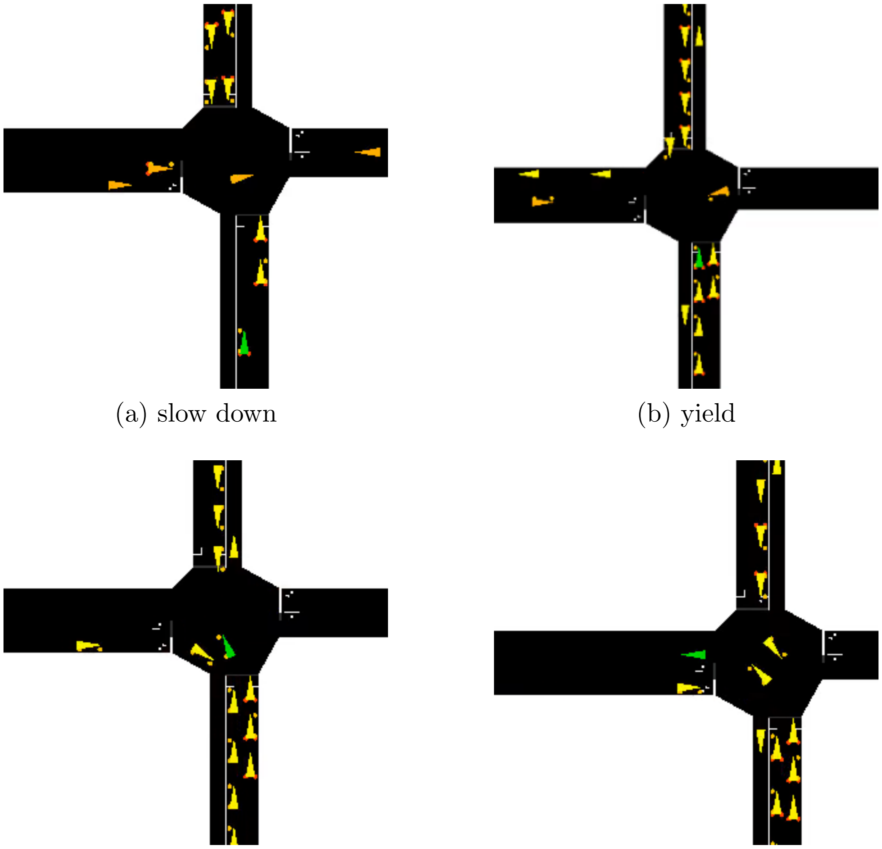
(a) need to change to left-turn lane

(b) slow down to find a gap

(c) successful lane change

(d) wait for traffic ahead

Figure 4.3: 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.



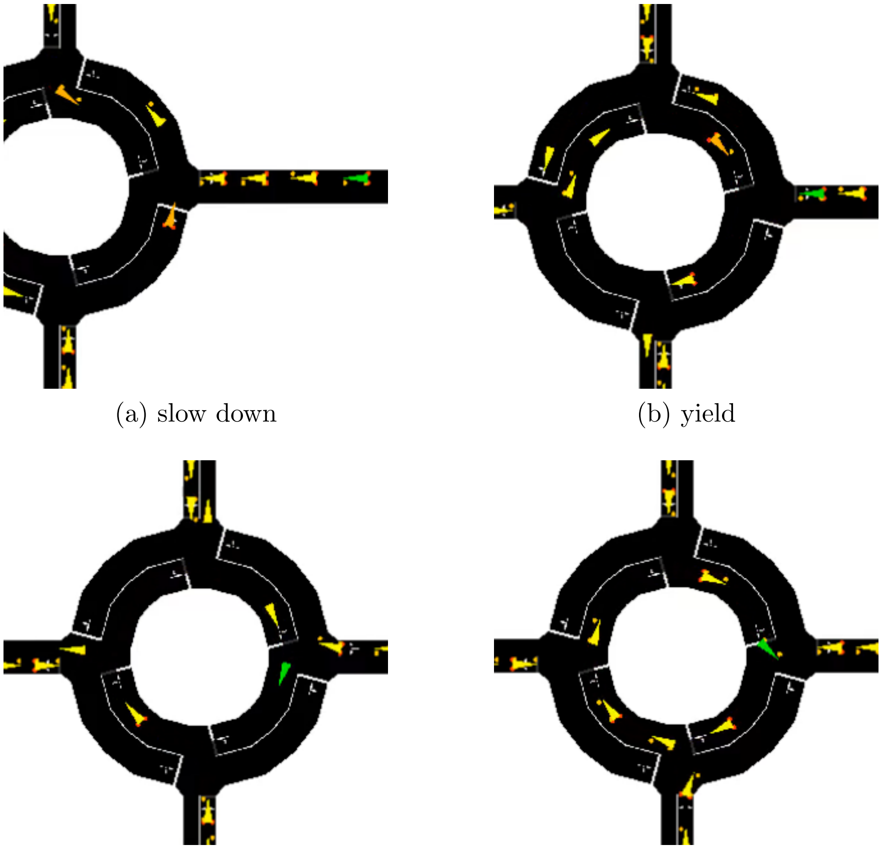
(a) slow down

(b) yield

(c) proceed when the way is clear

(d) successful left turn

Figure 4.4: 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 4.5: Zero-shot transfer performance. The agent (green) is able to drive through the ring road without safety or traffic rule violation.

Chapter 5

Revisiting Multi-Objective RL with Thresholded Lexicographic Order

5.1 A Deeper Look at Thresholded Lexicographic Q-Learning

So far we have introduced thresholded lexicographic Q-learning (Eq. 2.25) and its variant (Eq. 3.3) without examining the order relations they define on the space of value functions $\{\mathbf{V}^\pi | \pi \in S \to A\}$. Unfortunately, it turns out that these algorithms do not define an order relation on $\{\mathbf{V}^\pi | \pi \in S \to A\}$ for all multi-objective MDPs, thus do not strictly fit into the multi-objective RL formulation of Eq. 2.22 and Eq. 2.23. Due to the issues with the original thresholded lexicographic Q-learning, here we only focus on its variant Eq. 3.3, and in the following text, thresholded lexicographic Q-learning refers to this variant. Since we no longer consider the factored representation, a state will be denoted by s instead of $\mathbf{s}$.

By defining the optimal policy as $\arg\max_\pi \mathbf{V}^\pi$ (Eq. 2.23), it is implicitly assumed that:

$$\pi \geq_p \pi' \iff \mathbf{V}^\pi \geq_v \mathbf{V}^{\pi'} \tag{5.1}$$

In other words, we are assuming that the order on $\{\pi | S \to A\}$ is defined by the order of the corresponding value function $\mathbf{V}^\pi$. However, thresholded lexicographic Q-learning works differently, it defines the order relation on $\{\pi | S \to A\}$ through

$$\begin{aligned}\pi \geq_q \pi' \iff & \exists i \in \{0, 1, ..., k\}, \\ & \pi \in \hat{\Pi}_i \wedge \pi' \in \hat{\Pi}_i \wedge \pi' \notin \hat{\Pi}_{i+1} \wedge \\ & [\forall s \in S, (Q^{\Pi_i}_{i+1}(s, \pi(s)) \geq Q^{\Pi_i}_{i+1}(s, \pi'(s)) \wedge \\ & Q^{\Pi_i}_{i+1}(s, \pi'(s)) < \operatorname*{arg\,max}_{a \in \{\pi_{i-1}(s) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q^{\Pi_i}_{i+1}(s, a) - \tau_{i+1}) \vee \\ & Q^{\Pi_i}_{i+1}(s, \pi(s)) \geq \operatorname*{arg\,max}_{a \in \{\pi_{i-1}(s) | \pi_{i-1} \in \hat{\Pi}_{i-1}\}} Q^{\Pi_i}_{i+1}(s, a) - \tau_{i+1})]\end{aligned} \tag{5.2}$$

where $\hat{\Pi}_i, i = 1, 2, ..., k$ is defined by Eq. 3.3. $\hat{\Pi}_{k+1}$ and $Q^{\Pi_k}_{k+1}(s, a)$ are only technical, and are defined to be $\emptyset$ and 0 respectively. Now it comes down to whether there exists an order relation $\geq_v$, such that

$$\pi \geq_p \pi' \iff \pi \geq_q \pi' \tag{5.3}$$

If such an order relation exists for all multi-objective MDPs, then thresholded lexicographic Q-learning can be thought of as an exact solution algorithm for Eq. 2.23 with order relation $\geq_v$ on $\{\mathbf{V}^\pi | \pi \in S \to A\}$ implicitly defined through Eq. 5.2. However, it is easy to construct an example where no such $\geq_v$ exists. Consider the following single-objective problem (which is a special case of multi-objective problems) as shown in Figure 5.1 where there are three states: s_1, s_2 and s_3. s_2 and s_3 are absorption states with reward of 0. From s_1 there are three actions: a_1 leads the state back to s_1 with a reward of -1, a_2 leads to s_2 with reward -10, a_3 leads to s_3 with reward of 0. The discount factor is $\gamma = 0.9$. There are only three stationary deterministic policies: $\pi(s_1) = a_1$ and $\pi(s_1) = a_2$ and $\pi(s_1) = a_3$, denoted as π_1, π_2 and π_3 respectively. The optimal Q value for the problem is: $Q^*(s_1, a_1) = -1$, $Q^*(s_1, a_2) = -10$, $Q^*(s_1, a_3) = 0$. Assume that we apply thresholded lexicographic Q-learning with threshold $\tau = 2$. According to Eq. 5.2, we have $\pi_1 \geq_q \pi_2 \wedge \pi_2 \not\geq_q \pi_1$ because $Q^*(s_1, a_1) = -1 > v^*(s_1) - \tau = 0 - 2 = -2$, while $Q^*(s_1, a_2) = -10 < -2$. However, $v^{\pi_1}(s_1) =_v v^{\pi_2} = -10$, so $\pi_1 =_p \pi_2$. Therefore, Eq. 5.3 cannot hold for any order relation $\geq_v$ on $\{\mathbf{V}^\pi | \pi \in S \to A\}$.

The above analysis suggests that the intuitively elegant thresholded lexicographic Q-learning may not have strong theoretical results. To gain deeper understanding, we need to reexamine the motivation behind the algorithm — we *wish* to find the set of policies $\Pi_1 \subset \Pi$ such that policies $\pi \in \Pi_1$ are not worse than the best policy to a certain degree, or formally, $v^{\pi_1}_1(h^0, s) \geq v^{\Pi}_1(h^0, s) - \delta_1(s)$ if and only if $\pi_1 \in \Pi_1$, where h^0 is the empty history, $v^\pi(h, s)$ denotes the expected cumulative reward starting from history h and s following policy π, and $v^{\Pi_{i-1}}_i(\cdot)$ a shorthand for $\max_{\pi \in \Pi_{i-1}} v^\pi_i(\cdot)$. $\delta_1(s)$ is the slackness we allow for this objective, and can be tighter or looser for different starting states. Then

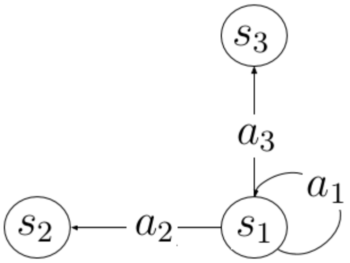


Figure 5.1: Thresholded lexicographic Q-learning is not an exact algorithm for multi-objective MDP.

among this set of policies Π_1, we further find the set of policies $\Pi_2 \subset \Pi_1$, such that $v_2^{\pi_2}(h^0, s) \geq v_2^{\Pi_1}(h^0, s) - \delta_2(s)$ if and only if $\pi_2 \in \Pi_2$, and so on. This can be represented by a multi-objective MDP with the following partial order on $\{\mathbf{V}^\pi | \pi \in S \to A\}$:

$$\begin{aligned}
&\mathbf{V}^\pi \geq_v \mathbf{V}^{\pi'} \iff \exists i \in \{0, 1, ..., k\}, \\
&[\forall 0 < j \leq i, \forall s \in S, v_j^\pi(h^0, s) \geq v_j^{\Pi_{j-1}}(h^0, s) - \delta_j(s) \wedge \\
&v_j^{\pi'}(h^0, s) \geq v_j^{\Pi_{j-1}}(h^0, s) - \delta_j(s)] \wedge \\
&[\exists s \in S, v_{i+1}^{\pi'}(h^0, s) < v_{i+1}^{\Pi_i}(h^0, s) - \delta_{i+1}(s)] \wedge \\
&[\forall s \in S, (v_{i+1}^{\pi}(h^0, s) \geq v_{i+1}^{\pi'}(h^0, s) \wedge \\
&v_{i+1}^{\pi'}(h^0, s) < v_{i+1}^{\Pi_i}(h^0, s) - \delta_{i+1}(s)) \vee \\
&v_{i+1}^{\pi}(h^0, s) \geq v_{i+1}^{\Pi_i}(h^0, s) - \delta_{i+1}(s)]
\end{aligned} \tag{5.4}$$

where Π_i is defined as:

$$\Pi_i \overset{def}{=} \{\pi \in \Pi_{i-1} | v_i^\pi(h^0, s) \geq v_i^{\Pi_{i-1}}(h^0, s) - \delta_i(s), \forall s \in S\} \tag{5.5}$$

and $\Pi_0 \overset{def}{=} \Pi$. Such a multi-objective MDP is called *thresholded lexicographic MDP* (TLMDP) [64, 39]. However, finding Π_i is non-trivial, and that's where thresholded lexicographic Q-learning comes into play. Consider a TLMDP with state-independent slackness $\delta_i(s) = \delta_i$, by setting $\tau_i = (1 - \gamma_i)\delta_i$, it is guarantee that $\hat{\Pi}_i \subset \Pi_i$ [64, 39]. In this sense, thresholded lexicographic Q-learning is an approximate algorithm for TLMDP.

5.2 Pseudo-Markov Thresholded Lexicographic Value Iteration

In this section, we provide further insights into multi-objective RL with thresholded lexicographic value iteration by introducing two algorithms. One is an exact algorithm, but is impractical because it requires knowledge of the transition probabilities; the other is a practical model-free approximate algorithm, which is a direct extension of thresholded lexicographic Q-learning, and gives strictly better approximation.

Reiterating Eq. 3.3, thresholded lexicographic Q-learning finds the $\hat{\Pi}_i$:

$$\hat{\Pi}_i = \{\pi \in \hat{\Pi}_{i-1} | Q_i(s, \pi(s)) > v_i^{\hat{\Pi}_{i-1}}(s) - (1 - \gamma_i)\delta_i, \forall s \in S\} \tag{5.6}$$

Here, τ_i has been substituted by $(1 - \gamma_i)\delta_i$ so that $\hat{\Pi}_i \subset \Pi_i$. For episodic tasks, this translates to:

$$\hat{\Pi}_i = \{\pi \in \hat{\Pi}_{i-1} | Q_i(s, \pi(s)) > v_i^{\hat{\Pi}_{i-1}}(s) - \frac{1}{T_i}\delta_i, \forall s \in S\} \tag{5.7}$$

In essence, the algorithm spreads the slackness δ_i *evenly* throughout the episode, thus leading to an incomplete solution: the slackness does not have to be spread *evenly* across states — we can decide to use more slackness in some time steps or in some states, and less slackness in others, so that the overall deviation of $v_i^{\pi}(s^0)$ from $v_i^{\Pi_{i-1}}(s^0)$ is within δ_i. This would potentially (but not necessarily) result in history-dependent policies, because we might need to keep track of how much slackness has already been used. We are, however, not interested the most general form of history-dependent policy that depends arbitrarily on the history — for one thing, storing such a policy would require space exponential to the time horizon. We are interested in policies that depends on a fixed-size Markov summary of the history $b^t = g(h^t, s^t), h^t \in H^t$, where g is the summary function. The fixed-size summary should be Markov, meaning that $P(b^t | h^{t-1}, s^{t-1}, d^{t-1}) = P(b^t | b^{t-1}, d^{t-1})$. Let B denote the set of all summaries, we call this type of policy $\pi = (d^0, d^1, ..., d^t, ...), d^t : B \to A$ a *pseudo-Markov* [1] deterministic policy.

Now we introduce an exact solution algorithm for TLMDP as defined by Eq. 5.4. Since it considers pseudo-Markov policy and requires the knowledge of the transition prababilities, we call it *pseudo-Markov thresholded lexicographic value iteration* (PsM-TLVI).

[1] The term *semi-Markov policy* would have been more appropriate, but is already taken in [54] for a different meaning.

Algorithm 3 Pseudo-Markov TLVI

```
function PsM-TLVI($Q, s^0$)
    // $Q^* = [Q_1^{\Pi_0}, Q_2^{\Pi_1}, ..., Q_k^{\Pi_{k-1}}]$ is the list of Q functions for each objective.
    // $s^0$ is the initial state.
    // $D_0^t$ denotes the set of all deterministic decision rules $\{d : S \rightarrow A\}$.
    // $\eta_i^{\Pi}(h, s, d)$ denotes $v_i^{\Pi}(h, s) - Q_i^{\Pi}(h, s, d(s))$
    for $i$ in $\{1, 2, ..., k\}$ do
        $D_i^0 := \{d \in D_{i-1}^0 | \eta_i^{\Pi_{i-1}}(h^0, s, d) \leq \delta_i(s), \forall s \in S\}$
    end for
    choose $d^0 \in D_k^0$
    for $i$ in $\{1, 2, ..., k\}$ do
        $\epsilon_i^0 := \delta_i(s^0) - \eta_i^{\Pi_{i-1}}(h^0, s^0, d^0)$
    end for
    for $t$ in $\{0, 1, ...\}$ do
        take action $a^t = d^t(s^t)$, and arrive at state $s^{t+1}$
        for $i$ in $\{1, 2, ..., k\}$ do
            $D_i^{t+1} := \left\{ d \in D_{i-1}^{t+1} \middle| \mathbb{E}_{s \sim p(s|s^t, a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d)] \leq \frac{1}{\gamma_i}\epsilon_i^t \right\}$
        end for
        choose $d^{t+1} \in D_k^{t+1}$
        for $i$ in $\{1, 2, ..., k\}$ do
            $\epsilon_i^{t+1} := \frac{1}{\gamma_i}\epsilon_i^t - \mathbb{E}_{s \sim p(s|s^t, a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d^{t+1})]$
        end for
    end for
end function
```

To go through the algorithm, we start with the first objective, and *for now* we assume that $v_i^{\Pi_{i-1}}(h, s)$ is somehow available. The allowed slackness can be thought as a quantity to be consumed, and we want to make sure that in expectation, the discounted cumulative slackness consumed does not exceed $\delta_i(s)$. Suppose that the remaining slackness is ϵ_1^t after executing action a^t in state s^t according to the decision rule d^t, we are faced with the choice of the decision rule for the next step d^{t+1}. For that we need to make sure that in expectation, the slackness we use at time $t+1$ does not exceed $\frac{1}{\gamma_1}\epsilon_1^t$. Since the slackness we will use in s^{t+1} following d^{t+1} would be:

$$\begin{aligned} \eta_1^{\Pi_0}(h^{t+1}, s^{t+1}, d^{t+1}) = v_1^{\Pi_0}(h^{t+1}, s^{t+1}) - \\ Q_1^{\Pi_0}(h^{t+1}, s^{t+1}, d^{t+1}(s^{t+1})) \end{aligned} \tag{5.8}$$

d^{t+1} needs to satisfy:

$$\frac{1}{\gamma_1}\epsilon_1^t \geq \sum_s p(s|s^t, a^t)\left[\eta_1^{\Pi_0}(h^{t+1}, s, d^{t+1})\right] \tag{5.9}$$

The decision rule for the initial time step d^0 should instead guarantee that:

$$\delta_1(s) \geq \eta_1^{\Pi_0}(h^0, s, d^0), \forall s \in S \tag{5.10}$$

The set of decision rules d^{t+1} that satisfy Eq. 5.9 (or Eq. 5.10 if $t+1=0$) can then be passed down to the second objective for further selection following the same procedure, and so on, till the k^{th} objective. After selecting d^{t+1}, we need to calculate the remaining slackness for the next step ϵ_i^{t+1}, according to how much slackness d^{t+1} uses at time $t+1$ in expectation:

$$\epsilon_i^{t+1} := \frac{1}{\gamma_i}\epsilon_i^t - \mathbb{E}_{s\sim p(s|s^t,a^t)}\left[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d^{t+1})\right] \tag{5.11}$$

If $t+1=0$, the remaining slackness should be calculated according to:

$$\epsilon_i^0 := \delta_i(s^0) - \eta_i^{\Pi_{i-1}}(h^0, s^0, d^0) \tag{5.12}$$

The complete algorithm is described in Algorithm 3. At each time step t, the algorithm returns k sets of feasible decision rules $D_i^t \subset \{S \to A\}$ (for the first i objectives). This essentially implies k sets of history-dependent policy $\tilde{\Pi}_i$ by:

$$\pi = (d^0, d^1, ..., d^t, ...) \in \tilde{\Pi}_i \iff d^t \in D_i^t, \forall t \tag{5.13}$$

As can be seen from Line 12 of the algorithm, D_i^t is dependent on the history only through s^{t-1}, a^{t-1}, and the remaining slackness of the first i objectives $\boldsymbol{\epsilon}_{1:i}^{t-1} = [\epsilon_1^{t-1}, \epsilon_2^{t-1}..., \epsilon_i^{t-1}]$, so we have:

$$D_i^t = D_i(\boldsymbol{\epsilon}_{1:i}^{t-1}, s^{t-1}, a^{t-1}) \tag{5.14}$$

Notice that $b^t = (\boldsymbol{\epsilon}^{t-1}, s^{t-1}, a^{t-1}, s^t)$ is a fixed-sized summary of the history, and is Markov (Eq. 5.11), therefore the policies $\pi \in \tilde{\Pi}_k$ are pseudo-Markov.

Proposition 1. *PsM-LVI is an exact algorithm for TLMDP as formulated by Eq. 5.4, i.e.,* $\tilde{\Pi}_k = \Pi_k$

Proof. We first prove $\tilde{\Pi}_k \subset \Pi_k$. In other words, we need to prove that any $\pi = (d^1, d^2, ...)$ with $d^t \in D_k^t, \forall t$, satisfies $v_i^{\pi}(h^0, s) \geq v_i^{\Pi_{i-1}}(h^0, s) - \delta_i(s), \forall i, \forall s \in S$. This can be shown by induction. Let $Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1, ..., d^{t-1})$ be the maximum expected cumulative reward that can be achieved following a policy in Π_{i-1}, after acting according to $d^0, d^1, ..., d^{t-1}$ in the first t time step starting from empty history and state s^0. We start with $Q_i^{\Pi_{i-1}}(h^0, s^0, d^0), \forall s^0 \in S$. Since d^0 satisfies $\eta_i^{\Pi_{i-1}}(h^0, s^0, d^0) \leq \delta_i(s^0)$ (Line 3), and $\epsilon_i^0 = \delta_i(s^0) - \eta_i^{\Pi_{i-1}}(h^0, s^0, d^0)$ (Line 7), it follows directly that:

$$\begin{gathered} Q_i^{\Pi_{i-1}}(h^0, s^0, d^0) = v_i^{\Pi_{i-1}}(h^0, s^0) - (\delta_i(s^0) - \epsilon_i^0) \\ \epsilon_i^0 \geq 0 \end{gathered}$$

Assume

$$\begin{gathered} Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1, ..., d^t) = v_i^{\Pi_{i-1}}(h^0, s^0) - (\delta_i(s^0) - \gamma^{(t)}\epsilon_i^t) \\ \epsilon_i^t \geq 0 \end{gathered}$$

Since $\epsilon_i^t \geq 0$, Line 12 and Line 16 ensures that

$$\epsilon_i^{t+1} \geq 0 \tag{5.15}$$

and that d^{t+1} satisfies

$$\mathbb{E}_{s\sim p(s|s^t,a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d^{t+1})] = \frac{1}{\gamma_i}\epsilon_i^t - \epsilon_i^{t+1}$$

We have

$$\begin{aligned} & Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1 ..., d^t) - Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1, ..., d^{t+1}) \\ =& \gamma^{(t+1)}\mathbb{E}_{s^1,...,s^t}\mathbb{E}_{s\sim p(s|s^t,a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+t}, s, d^{t+1})] \\ =& \gamma^{(t)}\epsilon_i^t - \gamma^{(t+1)}\epsilon_i^{t+1} \end{aligned}$$

Therefore

$$\begin{aligned} & Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1, ..., d^{t+1}) \\ =& Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1 ..., d^t) - \gamma^{(t)}\epsilon_i^t + \gamma^{(t+1)}\epsilon_i^{t+1} \\ =& v_i^{\Pi_{i-1}}(h^0, s^0) - (\delta_i(s^0) - \gamma^{(t)}\epsilon_i^t) - \gamma^{(t)}\epsilon_i^t + \gamma^{(t+1)}\epsilon_i^{t+1} \\ =& v_i^{\Pi_{i-1}}(h^0, s^0) - (\delta_i(s^0) - \gamma^{(t+1)}\epsilon_i^{t+1}) \end{aligned} \tag{5.16}$$

Eq. 5.15 and Eq. 5.16 completes the induction. Therefore:

$$v_i^{\pi}(h^0, s^0) = Q^{\Pi_{i-1}}(h^0, s^0, d^0, d^1, ...) \geq v_i^{\Pi_{i-1}}(h^0, s^0) - \delta_i(s^0)$$

finishing the proof.

Now we prove $\Pi_k \subset \tilde{\Pi}_k$, for which we need to show that every policy π that satisfies $v_i^\pi(h^0, s) \geq v^{\Pi_{i-1}}(h^0, s) - \delta_i(s), \forall i, \forall s \in S$ are in $\tilde{\Pi}_k$. If $\pi \notin \tilde{\Pi}_k$, it means that for some i and t, d^t violates either Line 3 or Line 12. According to Line 7 and Line 16, this would result in $\epsilon_i^t < 0$. So we have

$$\begin{aligned} v_i^\pi &\leq Q_i^{\Pi_{i-1}}(h^0, s^0, d^0, d^1 ..., d^t) \\ &= v_i^{\Pi_{i-1}}(h^0, s^0) - (\delta_i(s^0) - \gamma^{(t)}\epsilon_i^t) \\ &< v_i^{\Pi_{i-1}}(h^0, s^0) - \delta_i(s^0) \end{aligned}$$

Therefore $\pi \notin \Pi_k$. □

So far we have assumed $v_i^{\Pi_{i-1}}(h^t, s^t)$ is already available. Now we discuss how $v_i^{\Pi_{i-1}}(h^t, s^t)$ can be learned. $v_i^{\Pi_{i-1}}(h^t, s^t)$ is influenced by h^t only through the set of allowed decision rules $D_{i-1}^\tau, \tau \geq t$. As discussed above, the policies in Π_{i-1} are pseudo-Markov and $D_{i-1}^t = D_{i-1}(\boldsymbol{\epsilon}_{1:i-1}^{t-1}, s^{t-1}, a^{t-1})$. For each objective, we construct a MDP with state $s_{i+}^t = (\boldsymbol{\epsilon}_{1:i-1}^{t-1}, s^{t-1}, a^{t-1}, s^t)$ and action set $a_+ \in D_{i-1}(s_+) \subset \{S \to A\}$ (Note that now the actions are decision rules). The transition probabilities of this MDP are implied by the underlying MDP and Eq. 5.11. Denoting the optimal value function of this new MDP by v_{i+}^*, it is clear that:

$$v_i^{\Pi_{i-1}}(h^t, s^t) = v_{i+}^*(s_{i+}^t) \tag{5.17}$$

Therefore $v_i^{\Pi_{i-1}}(h^t, s^t)$ can be learned by doing regular value iteration or Q-learning on this new MDP. Since the action set D_i depends on D_{i-1}, the convergence of $v_i^{\Pi_{i-1}}(h^t, s^t)$ is sequential, from objective 1 to objective k.

Although Algorithm 3 is exact, it requires knowledge of the model. In Line 12 and Line 16, we need the transition probabilities to calculate $\mathbb{E}_{s\sim p(s|s^t,a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d)]$. A convenient approximation would be to use $\eta_i^{\Pi_{i-1}}(h^{t+1}, s^{t+1}, a^t)$ in place of $\mathbb{E}_{s\sim p(s|s^t,a^t)}[\eta_i^{\Pi_{i-1}}(h^{t+1}, s, d)]$. This would lead to an incomplete solution, but obviates the need for the transition probability. An added benefit is that we no longer need to think in terms of the decision rules d^t both in policy selection and the learning of $v_i^{\Pi_{i-1}}(h, s)$, just as in thresholded lexicographic Q-learning. We call the resulting algorithm *pseudo Markov thresholded lexicographic Q-Learning.*

Chapter 6

Conclusions

A RL problem consists of two parts: the environment model, and the interface between the agent and the task designer. The vast majority of RL literature restricts the agent-designer interface to a scalar reward that is a function of the transition tuple (s^t, a^t, s^{t+1}). However, more often than not, it is hard for humans to tell what reward they receive at a certain point of time, especially when there are multiple aspects involved. Since the reward function ultimately needs to be specified by a human, this creates a challenge for complex tasks such as autonomous urban driving. This thesis explored a more general form of agent-designer interface that allows multi-dimensional rewards — multi-objective RL. Each dimension of the reward deals with one aspect of the task, and a partial order relation is defined on the space of multi-dimensional value functions. Particularly,

- We have shown that an approximate algorithm for multi-objective RL with lexicographic order, thresholded lexicographic Q-learning, can be successfully applied to autonomous driving in a simulated environment.
- The limitation of thresholded lexicographic Q-learning is analyzed and further insight is provided by introducing a novel exact algorithm. A tractable form of history-dependent policy, pseudo-Markov policy is shown to be the key concept of this exact algorithm.

This thesis focused on a value function based approach to multi-objective RL. However in practice, Q-learning, especially when combined with function approximation, tends to give inaccurate estimates of the real Q function, which poses a challenge for value function based approaches. One future direction would be to explore policy gradient methods for multi-objective reinforcement learning with relaxed lexicographic order.

References

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. 2004. DOI: 10.1145/1015330.1015430. URL: http://doi.acm.org/10.1145/1015330.1015430.

[2] Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999. ISBN: 9780849303821.

[3] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[4] Andrew G. Barto and Sridhar Mahadevan. "Recent Advances in Hierarchical Reinforcement Learning". In: *Discrete Event Dynamic Systems* 13.1-2 (2003), pp. 41–77. DOI: 10.1023/A:1022140919877. URL: https://doi.org/10.1023/A:1022140919877.

[5] Michael Behrisch, Daniel Krajzewicz, and Melanie Weber, eds. *Simulation of Urban Mobility - First International Conference, SUMO 2013, Berlin, Germany, May 15-17, 2013. Revised Selected Papers*. Vol. 8594. Lecture Notes in Computer Science. Springer, 2014. ISBN: 978-3-662-45078-9. DOI: 10.1007/978-3-662-45079-6. URL: https://doi.org/10.1007/978-3-662-45079-6.

[6] Marc G. Bellemare et al. "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*. 2016, pp. 1471–1479. URL: http://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation.

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

[8] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. "Stochastic dynamic programming with factored representations". In: *Artif. Intell.* 121.1-2 (2000), pp. 49–107. DOI: 10.1016/S0004-3702(00)00033-3. URL: https://doi.org/10.1016/S0004-3702(00)00033-3.

[9] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. "Markov Decision Processes with Multiple Objectives". In: *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*. 2006, pp. 325–336. DOI: 10.1007/11672142_26. URL: https://doi.org/10.1007/11672142%5C_26.

[10] Peter Dayan and Geoffrey E. Hinton. "Feudal Reinforcement Learning". In: *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*. 1992, pp. 271–278. URL: http://papers.nips.cc/paper/714-feudal-reinforcement-learning.

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

[12] Eugene A. Feinberg. "Constrained Discounted Markov Decision Processes and Hamiltonian Cycles". In: *Math. Oper. Res.* 25.1 (Feb. 2000), pp. 130–140. ISSN: 0364-765X. DOI: 10.1287/moor.25.1.130.15210. URL: http://dx.doi.org/10.1287/moor.25.1.130.15210.

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

[14] Peter Geibel and Fritz Wysotzki. "Risk-Sensitive Reinforcement Learning Applied to Control under Constraints". In: *CoRR* abs/1109.2147 (2011). arXiv: 1109.2147. URL: http://arxiv.org/abs/1109.2147.

[15] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1994.

[16] Carlos Guestrin et al. "Efficient Solution Algorithms for Factored MDPs". In: *J. Artif. Intell. Res.* 19 (2003), pp. 399–468. DOI: 10.1613/jair.1000. URL: https://doi.org/10.1613/jair.1000.

[17] Hado van Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* 2010, pp. 2613–2621. URL: http://papers.nips.cc/paper/3964-double-q-learning.

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

[19] Ping Hou, William Yeoh, and Pradeep Varakantham. “Revisiting Risk-Sensitive MDPs: New Algorithms and Results”. In: *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014.* 2014. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7798.

[20] Leemon C. Baird III. “Residual Algorithms: Reinforcement Learning with Function Approximation”. In: *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995.* 1995, pp. 30–37.

[21] David Isele et al. “Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning”. In: *CoRR* abs/1705.01196 (2017). arXiv: 1705.01196. URL: http://arxiv.org/abs/1705.01196.

[22] Max Jaderberg et al. “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *CoRR* abs/1611.05397 (2016). arXiv: 1611.05397. URL: http://arxiv.org/abs/1611.05397.

[23] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. “Reinforcement Learning: A Survey”. In: *CoRR* cs.AI/9605103 (1996). URL: http://arxiv.org/abs/cs.AI/9605103.

[24] Sham Kakade. “A Natural Policy Gradient”. In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada].* 2001, pp. 1531–1538. URL: http://papers.nips.cc/paper/2073-a-natural-policy-gradient.

[25] Donald Knuth. *The TEXbook*. Reading, Massachusetts: Addison-Wesley, 1986.

[26] Sven Koenig and Reid G. Simmons. "Risk-Sensitive Planning with Probabilistic Decision Graphs". In: *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994*. 1994, pp. 363–373.

[27] Vijay R. Konda and John N. Tsitsiklis. "Actor-Critic Algorithms". In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. 1999, pp. 1008–1014. URL: http://papers.nips.cc/paper/1786-actor-critic-algorithms.

[28] Leslie Lamport. *LaTeX — A Document Preparation System*. Second. Reading, Massachusetts: Addison-Wesley, 1994.

[29] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *CoRR* abs/1509.02971 (2015). arXiv: 1509.02971. URL: http://arxiv.org/abs/1509.02971.

[30] Chunming Liu, Xin Xu, and Dewen Hu. "Multiobjective Reinforcement Learning: A Comprehensive Overview". In: *IEEE Trans. Systems, Man, and Cybernetics: Systems* 45.3 (2015), pp. 385–398. DOI: 10.1109/TSMC.2014.2358639. URL: https://doi.org/10.1109/TSMC.2014.2358639.

[31] R.T. Marler and J.S. Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and Multidisciplinary Optimization* 26.6 (Apr. 2004), pp. 369–395. ISSN: 1615-1488. DOI: 10.1007/s00158-003-0368-6. URL: https://doi.org/10.1007/s00158-003-0368-6.

[32] Volodymyr Mnih et al. "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*. 2016, pp. 1928–1937. URL: http://jmlr.org/proceedings/papers/v48/mniha16.html.

[33] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[34] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping". In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287. ISBN: 1-55860-612-2. URL: http://dl.acm.org/citation.cfm?id=645528.657613.

[35] Daniel C. K. Ngai and Nelson Hon Ching Yung. “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*. 2007, pp. 818–823. DOI: 10.1109/ITSC.2007.4357682. URL: https://doi.org/10.1109/ITSC.2007.4357682.

[36] Ian Osband et al. “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*. 2016, pp. 4026–4034. URL: http://papers.nips.cc/paper/6501-deep-exploration-via-bootstrapped-dqn.

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

[38] Chris Paxton et al. “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*. 2017, pp. 6059–6066. DOI: 10.1109/IROS.2017.8206505. URL: https://doi.org/10.1109/IROS.2017.8206505.

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

[40] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1994. ISBN: 0471619779.

[41] Diederik Marijn Roijers et al. “A Survey of Multi-Objective Sequential Decision-Making”. In: *CoRR* abs/1402.0590 (2014). arXiv: 1402.0590. URL: http://arxiv.org/abs/1402.0590.

[42] Tim Salimans et al. “Evolution Strategies as a Scalable Alternative to Reinforcement Learning”. In: *CoRR* abs/1703.03864 (2017). arXiv: 1703.03864. URL: http://arxiv.org/abs/1703.03864.

[43] Ahmad El Sallab et al. “End-to-End Deep Reinforcement Learning for Lane Keeping Assist”. In: *CoRR* abs/1612.04340 (2016). arXiv: 1612.04340. URL: http://arxiv.org/abs/1612.04340.

[44] Atri Sarkar et al. “Trajectory prediction of traffic agents at urban intersections through learned interactions”. In: *20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017*. 2017, pp. 1–8. DOI: 10.1109/ITSC.2017.8317731. URL: https://doi.org/10.1109/ITSC.2017.8317731.

[45] Makoto Sato, Hajime Kimura, and Shibenobu Kobayashi. “TD Algorithm for the Variance of Return and Mean-Variance Reinforcement Learning”. In: *Transactions of the Japanese Society for Artificial Intelligence* 16.3 (2001), pp. 353–362. DOI: 10.1527/tjsai.16.353.

[46] Tom Schaul et al. “Prioritized Experience Replay”. In: *CoRR* abs/1511.05952 (2015). arXiv: 1511.05952. URL: http://arxiv.org/abs/1511.05952.

[47] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[48] John Schulman et al. “Trust Region Policy Optimization”. In: *CoRR* abs/1502.05477 (2015). arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477.

[49] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving”. In: *CoRR* abs/1610.03295 (2016). arXiv: 1610.03295. URL: http://arxiv.org/abs/1610.03295.

[50] Craig Sherstan et al. “Comparing Direct and Indirect Temporal-Difference Methods for Estimating the Variance of the Return”. In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*. 2018, pp. 63–72. URL: http://auai.org/uai2018/proceedings/papers/35.pdf.

[51] Olivier Sigaud and Olivier Buffet. *Markov Decision Processes in Artificial Intelligence*. Wiley-IEEE Press, 2010. ISBN: 1848211678, 9781848211674.

[52] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (Oct. 2017), p. 354. ISSN: 1476-4687. DOI: 10.1038/nature24270. URL: https://doi.org/10.1038/nature24270.

[53] Felipe Petroski Such et al. “Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning”. In: *CoRR* abs/1712.06567 (2017). arXiv: 1712.06567. URL: http://arxiv.org/abs/1712.06567.

[54] Richard S. Sutton, Doina Precup, and Satinder P. Singh. “Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning”. In: *Artif. Intell.* 112.1-2 (1999), pp. 181–211. DOI: 10.1016/S0004-3702(99)00052-1. URL: https://doi.org/10.1016/S0004-3702(99)00052-1.

[55] Richard S. Sutton et al. “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*. 2011, pp. 761–768. URL: http://portal.acm.org/citation.cfm?id=2031726%5C&CFID=54178199%5C&CFTOKEN=61392764.

[56] Richard S. Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. 1999, pp. 1057–1063. URL: http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.

[57] Aviv Tamar, Dotan Di Castro, and Shie Mannor. “Temporal Difference Methods for the Variance of the Reward To Go”. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. 2013, pp. 495–503. URL: http://jmlr.org/proceedings/papers/v28/tamar13.html.

[58] J. N. Tsitsiklis and B. Van Roy. “An analysis of temporal-difference learning with function approximation”. In: *IEEE Transactions on Automatic Control* 42.5 (May 1997), pp. 674–690. ISSN: 0018-9286. DOI: 10.1109/9.580874.

[59] Frederick M. Waltz. “An engineering approach: hierarchical optimization criteria”. In: *IEEE Transactions on Automatic Control* 12.2 (1967), pp. 179–180.

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

[61] Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. “A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers”. In: *CoRR* abs/1804.07871 (2018). arXiv: 1804.07871. URL: http://arxiv.org/abs/1804.07871.

[62] D.J White. “Multi-objective infinite-horizon discounted Markov decision processes”. In: *Journal of Mathematical Analysis and Applications* 89.2 (1982), pp. 639–647. ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(82)90122-6. URL: http://www.sciencedirect.com/science/article/pii/0022247X82901226.

[63] Ronald J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine Learning* 8.3 (May 1992), pp. 229–256. ISSN: 1573-0565. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

[64] Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. “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*. 2015, pp. 3418–3424. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9471.

[65] Stella X Yu, Yuanlie Lin, and Pingfan Yan. “Optimization Models for the First Arrival Target Distribution Function in Discrete Time”. In: *Journal of Mathematical Analysis and Applications* 225.1 (1998), pp. 193–223. ISSN: 0022-247X. DOI: https://doi.org/10.1006/jmaa.1998.6015. URL: http://www.sciencedirect.com/science/article/pii/S0022247X98960152.

[66] Brian D. Ziebart et al. “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*. 2008, pp. 1433–1438. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-227.php.