Machine Learning, 22, 227-250 (1996)

The Effect of Representation and Knowledge on Goal-Directed Exploration with Reinforcement-Learning Algorithms*

SVEN KOENIG skoenig@cs.cmu.edu
REID G. SIMMONS reids@cs.cmu.edu
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Editor: Leslie Pack Kaelbling

Abstract. We analyze the complexity of on-line reinforcement-learning algorithms applied to goal-directed exploration tasks. Previous work had concluded that, even in deterministic state spaces, initially uninformed reinforcement learning was at least exponential for such problems, or that it was of polynomial worst-case time-complexity only if the learning methods were augmented. We prove that, to the contrary, the algorithms are tractable with only a simple change in the reward structure ("penalizing the agent for action executions") or in the initialization of the values that they maintain. In particular, we provide tight complexity bounds for both Watkins' Q-learning and Heger's Q-hat-learning and show how their complexity depends on properties of the state spaces. We also demonstrate how one can decrease the complexity even further by either learning action models or utilizing prior knowledge of the topology of the state spaces. Our results provide guidance for empirical reinforcement-learning researchers on how to distinguish hard reinforcement-learning problems from easy ones and how to represent them in a way that allows them to be solved efficiently.

Keywords: action models, admissible and consistent heuristics, action-penalty representation, complexity, goal-directed exploration, goal-reward representation, on-line reinforcement learning, prior knowledge, reward structure, Q-hat-learning, Q-learning

1. Introduction

A **goal-directed reinforcement-learning problem** can often be stated as: an agent has to learn an optimal policy for reaching a goal state in an initially unknown state space. One necessary step towards a solution to this problem is to locate a goal state. We therefore analyze the complexity of on-line reinforcement-learning methods (measured in action executions) until the agent reaches a goal state for the first time. If a reinforcement-learning method is terminated when the agent reaches a goal state, then it solves the following **goal-directed exploration problem**: the agent has to find some path to a goal state in an initially unknown or only partially known state space, but it does not need to find a shortest path. Studying goal-directed exploration problems provides insight into the corresponding reinforcement-learning problems. Whitehead (1991a), for example, proved that goal-directed exploration with reinforcement-learning methods can be intractable, which demonstrates that solving reinforcement-learning problems can be intractable as

* This research was supported in part by NASA under contract NAGW-1175 . The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

well. In particular, he showed that the behavior of an agent that is controlled by an initially uninformed (i.e. tabula rasa) reinforcement-learning algorithm can degenerate to a random walk until it reaches a goal state. In this case, it might have to execute a number of actions that is, on average, exponential in the size of the state space and even speed-up techniques such as Lin's action-replay technique (Lin, 1992) do not improve its performance. This theoretical result contrasts sharply with experimental observations of Kaelbling (1990), Whitehead (1991b), Peng and Williams (1992), and Moore and Atkeson (1993b), who reported good performance results.

These results motivated us to study how the performance of reinforcement-learning algorithms in deterministic and non-deterministic state spaces is affected by different representations, where a **representation** determines both the immediate rewards that the reinforcement-learning algorithms receive and the initialization of the values that they maintain. We study versions of both Watkins' Q-learning and Heger's $\hat{Q}$-learning that perform only a minimal amount of computation between action executions, choosing only which action to execute next, and basing this decision only on information local to the current state of the agent. If these inefficient algorithms have a low complexity, then we can expect other reinforcement-learning algorithms to have a low complexity as well. One main result of our analysis is that the choice of representation can have a tremendous impact on the performance of these reinforcement-learning algorithms. If a poor representation is chosen, they are intractable. However, if a good reward structure (such as "penalizing the agent for action executions") or suitable initialization is chosen, the complexity of the same algorithms is only a small polynomial in the size of the state space. Consequently, reinforcement-learning algorithms can be tractable without any need for augmentation. One can decrease their complexity even further by either learning action models or utilizing prior knowledge of the topology of the state space. These results, the proofs of which are found in (Koenig and Simmons, 1995a), provide guidance for empirical reinforcement-learning researchers on how to distinguish hard reinforcement-learning problems from easy ones and how to represent them in a way that allows them to be solved efficiently. They also provide the theoretical underpinnings for the experimental results mentioned above.

2. Notation and Assumptions

We use the following notation: S denotes the finite set of states of the state space, $s_{start} \in S$ is the start state, and G (with $\emptyset \neq G \subseteq S$) is the set of goal states. $A(s)$ is the finite set of actions that can be executed in state $s \in S$. Executing action $a \in A(s)$ causes a (potentially nondeterministic) state transition into one of the states $succ(s, a)$ (with $\emptyset \neq succ(s, a) \subseteq S$). The **size** of the state space is $n := |S|$, and the total number of state-action pairs (loosely called **actions**) is $e := \sum_{s \in S} |A(s)|$. We assume that the state space does not change over time and is completely observable: the agent can always determine its current state with certainty. It also knows at every point in time which actions it can execute in its current state and whether its current state is a goal state.

A state space is **deterministic** iff the cardinality of $succ(s, a)$ equals one for all $s \in S$ and $a \in A(s)$. For deterministic state spaces, we use $succ(s, a)$ to denote both the

set of successor states and the single element of this set. We call a state space **non-deterministic** if we want to stress that we do not require it to be deterministic. In non-deterministic state spaces, we view reinforcement learning as a two-player game: The reinforcement-learning algorithm selects the action to execute and is only constrained by having to choose an action that is applicable in the current state of the agent. The action determines the possible successor states, from which some mechanism, which we call **nature**, chooses one. We do not impose any restrictions on how nature makes its decisions (its **strategy**). In particular, its choice can depend on the state-action history (we do not make the Markov assumption). Nature might, for example, select a successor state randomly or deliberately, in the latter case either to help or hurt the agent. Reinforcement learning in deterministic state spaces is simply a special case of reinforcement learning in non-deterministic state spaces where every action execution results in a unique successor state and nature has no choice which one to select.

The **distance** $d(s, s') \in [0, \infty]$ between $s \in S$ and $s' \subset S$ is defined to be the (unique) solution of the following set of equations

$$d(s, s') = \begin{cases} 0 & \text{for all } s, s' \in S \text{ with } s = s' \\ 1 + \min_{a \subset A(s)} \max_{s'' \in succ(s,a)} d(s'', s') & \text{for all } s, s' \in S \text{ with } s \neq s'. \end{cases}$$

This means that an agent that knows the state space and acts optimally can reach s' from s with at most $d(s, s')$ action executions no matter which strategy nature uses. The **goal distance** $gd(s)$ of $s \in S$ is defined to be $gd(s) := \min_{s' \in G} d(s, s')$. If the agent had to traverse a state more than once in order to reach a goal state, then nature could force it to traverse this cycle infinitely often, which would imply $gd(s) = \infty$. Thus, $gd(s) \leq n - 1$ if $gd(s)$ is finite. We define the **diameter** (depth) of the state space with respect to G as $d := \max_{s \in S} gd(s)$. If d is finite, then $d \leq n - 1$. These definitions correspond to a worst-case scenario, in which nature is an opponent of the agent. For deterministic state spaces, the definitions of $d(s, s')$ and $gd(s)$ simplify to the standard definitions of distance and goal distance, respectively.

We call a reinforcement learning algorithm **uninformed** if it initially does not know which successor states an action can lead to. By its **complexity** we mean an upper bound on the number of actions that an agent controlled by the uninformed reinforcement-learning algorithm can execute in state spaces of a given size until it reaches a goal state for the first time. This bound has to hold for all possible topologies of the state space, start and goal states, tie breaking rules among indistinguishable actions, and strategies of nature. If $gd(s_{start}) = \infty$, then even completely informed reinforcement-learning algorithms have infinite complexity, since there exists a strategy of nature that prevents the agent from reaching any goal state. This problem could be solved by requiring $gd(s_{start}) < \infty$, but this is not a sufficient condition to guarantee the existence of uninformed reinforcement-learning algorithms with finite complexity. We call a state s **lost** iff $gd(s) = \infty$. If the agent enters a lost state during exploration, then nature can prevent it from reaching a goal state. Thus, the complexity can only be finite if no states are lost (i.e. if $d < \infty$).[1] We call state spaces with this property **safely explorable** and limit our analysis to such state spaces. Moore and Atkeson's parti-game algorithm (Moore and Atkeson, 1993a), for example, learns safely explorable abstractions of spatial

Initially, the agent is in state $s_{start} \in S$.

1. Set $s :=$ the current state.
2. If $s \in G$, then stop.
3. Set $a := argmax_{a' \in A(s)} Q(s, a')$.
   (Read: "Choose an action $a \in A(s)$ with the largest $Q(s, a)$ value.")
4. Execute action a.
   (As a consequence, nature selects a state $s' \in succ(s, a)$, the agent receives the immediate reward $r(s, a, s')$, and its new state becomes s'.)
5. Update $Q(s, a)$ using $r(s, a, s')$ and $U(s')$.
6. Go to 1.

where $U(s) := \max_{a' \in A(s)} Q(s, a')$ (i.e. the value of a state is the largest value of its actions).

Figure 1. A framework for 1-step Q-learning

state spaces. Other examples of safely explorable state spaces include two-player zero-sum games where one player can force a win and the game is restarted if this player loses. For deterministic state spaces, researchers such as Whitehead (1991a) and Ishida (1992) usually make the more restrictive assumption that the state spaces are safely explorable for all $s - start \in S$ and $G \subseteq S$, not just the $s - start$ and G given (i.e. that they are strongly connected).

3. A General Q-Learning Framework

Reinforcement learning is learning from positive and negative rewards (costs). When the agent executes action a in state s and successor state s' results, it receives the immediate reward $r(s, a, s') \in \mathcal{R}$. If the agent receives immediate reward r_t when it executes the $(t + 1)$st action, then the **total reward** that it receives over its lifetime for this particular behavior is $\sum_{t=0}^{\infty} \gamma^t r_t$. The **discount factor** $\gamma \in (0, 1]$ specifies the relative value of a reward received after t action executions compared to the same reward received one action execution earlier. We say that **discounting** is used if $\gamma < 1$, otherwise no discounting is used. Reinforcement learning algorithms solve goal-directed reinforcement-learning problems by determining a behavior for the agent that maximizes its total reward. They specify such behaviors as state-action rules, called **policies**.

We analyze reinforcement-learning methods that are variants of Q-learning, probably the most popular reinforcement learning method. In particular, we study 1 step versions of on-line Q-learning. They perform only a minimal amount of computation between action executions, choosing only which action to execute next, and basing this decision only on information local to the current state of the agent. A general framework for 1 step Q-learning (Figure 1) consists of a **termination-checking step** (line 2), an **action-selection step** (line 3), an **action-execution step** (line 4), and a **value-update step** (line 5). The termination-checking step stops the agent when it reaches a goal state.

This is possible, because we have limited ourselves to studying goal-directed exploration problems. The action-selection step determines which action to execute next. This decision is based on values that store information about the relative goodness of the actions, one **Q-value** $Q(s,a) \in \mathcal{R}$ for each action a that can be executed in state s. $Q(s,a)$ approximates the total reward that the agent receives if it starts in s, executes a, and then behaves optimally. The action-selection strategy is greedy: it chooses the action with the largest Q-value in the curent state of the agent. (If several actions tie, an arbitrary one of the equally good actions can be selected.) Because the action-selection step uses only those Q-values that are local to the current state of the agent, there is no need to predict $succ(s,a)$ for any $a \in A(s)$, which means that 1-step Q-learning does not need to learn an **action model** of the state space. After the action-execution step has directed the agent to execute the desired action a in its current state s, nature selects the successor state s' from the states $succ(s,a)$ and the agent receives the immediate reward $r(s,a,s')$. 1-step Q-learning then has temporary access to the Q-values of the agent's former and new state at the same time, and the value-update step adjusts $Q(s,a)$ using $r(s,a,s')$ and the values $Q(s',a')$ for $a' \in A(s')$. This is done because a 1-step look-ahead value of the total reward is more accurate than, and should therefore replace, the current value of $Q(s,a)$. For now, we leave the initial Q-values unspecified.

Two well-known Q-learning algorithms fit this general Q-learning framework:

- A 1-step version (Whitehead, 1991a) of Watkins' **Q-learning algorithm** (Watkins, 1989) can be obtained from the Q-learning framework by making the value-update step

  "Set $Q(s,a) := (1 - \alpha)Q(s,a) + \alpha(r(s,a,s') + \gamma U(s'))$,"

  where $\alpha \in (0,1]$ is called the **learning rate**. This Q-learning algorithm assumes that nature always selects the successor state $s' \in succ(s,a)$ with some time-invariant (but unknown) probability $p(s,a,s')$. Thus, nature selects the successor states probabilistically, and the agent plans for the average (i.e. risk-neutral) case: it tries to maximize its *expected total reward*. Convergence results for Q-learning are given in (Watkins and Dayan, 1992).

  The learning rate determines how much $Q(s,a)$ changes with every update. In order for the Q-values to converge in non-deterministic state spaces to the desired values, it has to approach zero asymptotically – in a manner described in (Watkins and Dayan, 1992). Q-learning needs the learning rate to be able to average the values $r(s,a,s') + \gamma U(s')$ for all successor states $s' \in succ(s,a)$. In deterministic state spaces, however, there is only one successor state and, consequently, averaging is not necessary. Thus, the learning rate can be set to one.

- Heger's **$\hat{\text{Q}}$-learning algorithm** (Heger, 1994) (pronounced "Q-hat-learning") can be obtained from the Q-learning framework by making the value-update step

  "Set $Q(s,a) := \min(Q(s,a), r(s,a,s') + \gamma U(s'))$."

and initializing the Q-values optimistically. $\hat{Q}$-learning assumes that nature always selects the worst successor state for the agent. Thus, nature is an opponent of the agent, and the agent tries to maximize the *total reward that it can receive in the worst case*. Even if the successor states are determined probabilistically, the agent could be extremely risk-averse (in the sense of utility theory; see (Koenig and Simmons, 1994)) and believe in Murphy's law ("anything that can go wrong, will indeed go wrong"). It then assumes that an imaginary opponent exists that always makes the worst possible successor state occur that its action can result in. In this case, $\hat{Q}$-learning can be used to learn a behavior that reflects a completely risk averse attitude and is optimal for the agent provided that it accepts the axioms of utility theory.

$\hat{Q}$-learning needs less information about the state space than Q-learning and converges faster. In particular, it does not need to execute each action $a \in A(s)$ in state s often enough to get a representative distribution over the successor states $succ(s, a)$ and has no need for a learning rate; for details see (Heger, 1996). Convergence results for $\hat{Q}$-learning are given in (Heger, 1994).

If the agent wants to learn an optimal policy, its risk attitude determines which Q-learning algorithm it should use. In the following sections, we analyze the complexity of Q-learning and $\hat{Q}$-learning for different representations of goal-directed exploration problems.

4. Representations

When modeling a goal-directed reinforcement-learning or exploration problem, one has to decide on both the immediate rewards and the initial Q-values. So far, we have left these values unspecified. In this section, we introduce possible reward structures and initializations, and discuss their properties. All of these representations have been used in the experimental reinforcement-learning literature to represent goal-directed reinforcement-learning problems, i.e. for learning shortest paths to a goal state. Since we have restricted our analysis to goal-directed exploration problems, we have to decide on the following values: For the reward structure, we have to decide on the values $r(s, a, s') \in \mathcal{R}$ for $s \in S \setminus G := S \cap \overline{G}$, $a \in A(s)$, and $s' \in succ(s, a)$. For the initial Q-values, we have to decide on the values $Q(s, a)$ for $s \in S \setminus G$ and $a \in A(s)$. In both cases, the values for $s \in G$ do not matter, because the reinforcement-learning methods terminate when the agent reaches a goal state. For goal directed reinforcement learning problems, we define $Q(s, a) = 0$ for $s \in G$ and a $a \in A(s)$.)

4.1. Reward Structures

Developing appropriate reward structures can, in general, be a complex engineering problem (Matarić, 1994). Since goal-directed reinforcement-learning methods determine policies that maximize the total reward, a reward structure for learning shortest paths must guarantee that a state with a smaller goal distance also has a larger optimal total

reward. Two different reward structures have been used in the literature. Neither of them encodes any information about the topology of the state space.

- In the **goal-reward representation**, the agent is rewarded for entering a goal state, but not rewarded or penalized otherwise. This reward structure has been used by Sutton (1990), Whitehead (1991a), Peng and Williams (1992), and Thrun (1992b), among others. Formally,

$$r(s,a,s') := \begin{cases} 1 & \text{for } s \in S \setminus G,\ a \in A(s), \text{ and } s' \in G \\ 0 & \text{for } s \in S \setminus G,\ a \in A(s), \text{ and } s' \in S \setminus G. \end{cases}$$

  Discounting is necessary for learning shortest paths with this reward structure. If no discounting were used, all behaviors that lead the agent eventually to a goal state would have a total reward of one and the reinforcement-learning algorithms could not distinguish between paths of different lengths. If discounting is used, then the goal reward gets discounted with every action execution, and the agent tries to reach a goal state with as few action executions as possible in order to maximize the portion of the goal reward that it receives.

- In the **action-penalty representation**, the agent is penalized for every action that it executes. This reward structure is **denser** than the goal-reward representation (the agent receives non-zero rewards more often) if goals are relatively sparse. It has been used by Barto, Sutton, and Watkins (1989), Koenig (1991), and Barto, Bradtke, and Singh (1995), among others. Formally,

$$r(s,a,s') := -1 \qquad \text{for } s \in S \setminus G,\ a \in A(s), \text{ and } s' \in S.$$

  For learning shortest paths, discounting can be used, but is not necessary. In both cases, the agent tries to reach a goal state with as few action executions as possible in order to minimize the amount of penalty that it receives. The action-penalty representation can be generalized to non-uniform immediate rewards; see Section 7.

4.2. Initial Q-Values

The complexity of Q- or Q-learning depends on properties of the initial Q-values. An important special case of initial Q-values are those that do not convey any information about the state space, such as uniformly initialized Q-values.

Definition. Q- or $\hat{\text{Q}}$-learning is **uniformly initialized** with $q \in \mathcal{R}$ (or, synonymously, q-initialized), iff initially

$$Q(s,a) = \begin{cases} 0 & \text{for all } s \in G \text{ and } a \in A(s) \\ q & \text{for all } s \in S \setminus G \text{ and } a \in A(s). \end{cases}$$

If Q- or $\hat{\text{Q}}$-learning is q-initialized with $q \neq 0$, then goal states are initialized differently from non-goal states. It is important to understand that this particular initialization does not convey any information, since the agent is able to distinguish whether its current

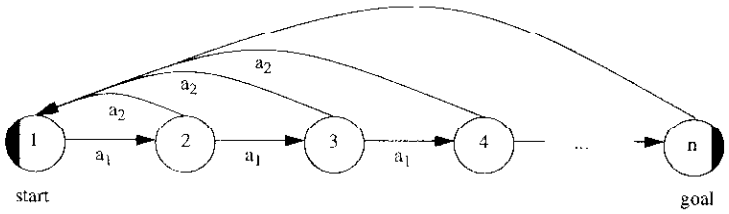

Figure 2. A state space ("reset state space") for which random walks need $3 \times 2^{n-2} - 2$ action executions on average to reach the goal state (for $n \geq 2$)

state is a goal state or not and can therefore initialize the Q-values differently for goal and non-goal states. Consequently, all uniformly initialized Q-learning algorithms are uninformed. This includes both zero-initialized and (minus one)-initialized Q-learning algorithms.

The following definitions characterize properties of Q-values for undiscounted Q- or $\hat{Q}$-learning with action-penalty representation.

Definition. Q-values are **consistent** for undiscounted Q- or $\hat{Q}$-learning with action-penalty representation iff

$$\left.\begin{matrix} 0 \\ -1 + \min_{s' \in succ(s,a)} U(s') \end{matrix}\right\} \leq Q(s,a) < 0 \quad \left\{\begin{matrix} \text{for all } s \in G \text{ and } a \in A(s) \\ \text{for all } s \in S \setminus G \text{ and } a \in A(s). \end{matrix}\right.$$

Consistency means that the triangle inequality holds. It corresponds to the concept of consistency or monotonicity in heuristic search. Both zero-initialized and (minus one)-initialized Q-values are consistent, for example.

Definition. Q-values are **admissible** for undiscounted Q- or $\hat{Q}$-learning with action-penalty representation iff

$$\left.\begin{matrix} 0 \\ -1 - \max_{s' \in succ(s,a)} qd(s') \end{matrix}\right\} < Q(s,a) \leq 0 \quad \left\{\begin{matrix} \text{for all } s \in G \text{ and } a \in A(s) \\ \text{for all } s \in S \setminus G \text{ and } a \in A(s). \end{matrix}\right.$$

Admissibility means that $-Q(s,a)$ never overestimates the number of action executions that the agent needs for reaching a goal state if it knows the state space, starts in state s, executes action a, and then behaves optimally, but nature tries to keep it away from a goal state for as long as possible. This corresponds to the concept of admissibility in heuristic search. All consistent Q-values are admissible.

5. An Intractable Representation

In this section, we assume that Q-learning operates on the goal-reward representation and is zero-initialized. We show that this representation makes the exploration problem intractable.

The agent receives a non-zero immediate reward only if an action execution results in a goal state. Thus, during the search for a goal state, all Q-values remain zero, and the action selection step has no information on which to base the decision which action to execute next.[2] If Q-learning had a systematic bias for actions, for example if it always chose the smallest action according to some predetermined ordering,then the agent could cycle in the state space forever. We therefore assume that it selects among the available actions with uniform probability, in which case the agent performs a random walk. Although random walks reach a goal state with probability one in safely explorable state spaces, the required number of action executions can exceed every given bound. This implies that the complexity of Q-learning is infinite. In deterministic state spaces, the expected number of action executions x_s that the agent needs to reach a goal state from state s can be calculated by solving the following set of linear equations.

$$x_s = \begin{cases} 0 & \text{for } s \in G \\ 1 + \dfrac{1}{|A(s)|} \displaystyle\sum_{a \in A(s)} x_{succ(s,a)} & \text{for } s \in S \setminus G. \end{cases}$$

Although $x_{s_{start}}$ is finite, it can scale exponentially with n. Consider for example the (deterministic) reset state space shown in Figure 2. A **reset state space** is one in which all states (except for the start state) have an action that leads back to the start state. It corresponds for example to the task of stacking n blocks if the agent can, in every state, either stack another block or scramble the stack. Solving the linear equations $x_1 = 1 + x_2$, $x_s = 1 + 0.5x_1 + 0.5x_{s+1}$ for all $s \in \{2, 3, \ldots, n-1\}$, and $x_n = 0$ yields $x_{s_{start}} = x_1 = 3 \times 2^{n-2} - 2$. Since the complexity of Q-learning cannot be lower in non-deterministic state spaces (a superset of deterministic state spaces), the following result holds:

THEOREM 1 *The expected number of action executions that zero-initialized Q-learning with goal-reward representation needs to reach a goal state in deterministic or non-deterministic state spaces can be exponential in n, the size of the state space.*

Whitehead (1991a) made the same observation for the behavior of Q-learning in deterministic state spaces that have the following property: in every state (except for the states on the perimeter of the state space), the probability of choosing an action that leads away from the only goal state is larger than the probability of choosing an action that leads closer to the goal state. This observation motivated him to explore cooperative reinforcement-learning methods in order to decrease the complexity. Subsequently, Thrun (1992a) showed that even non-cooperative reinforcement-learning algorithms can have polynomial complexity if they are extended with a directed exploration mechanism that he calls "counter-based exploration." Counter-based Q-learning, for example, maintains, in addition to the Q-values, a second kind of state-action values, called "counters." These values are used exclusively during learning (exploration) and are meaningless afterwards. Thrun was able to specify action-selection and value-update rules that use these counters and achieve polynomial complexity. There are other techniques that can improve the performance of reinforcement-learning algorithms, such as Lin's action-replay technique (Lin, 1992). However, this method does not change the Q-values before a goal state

has been reached for the first time when it is applied to zero-initialized Q-learning with goal-reward representation and, thus, cannot be used to reduce its complexity.

6. Tractable Representations

In the following, we show that one does not need to augment Q-learning algorithms to reduce their complexity. They are tractable if one uses either the action-penalty representation or initial Q-values different from zero. The intuitive explanation is that, in both cases, the Q-values change immediately, starting with the first action execution. This way, the agent remembers the effects of previous action executions. Our analysis, that formally shows by how much the complexity is reduced, also explains experimental findings by Kaelbling (1990), Whitehead (1991b), Peng and Williams (1992), and Moore and Atkeson (1993b), who reported good performance results for Q-learning when using similar representations.

6.1. Deterministic State Spaces

We first analyze two representations that make Q- or $\hat{Q}$-learning tractable in deterministic state spaces and then discuss how their complexity can be reduced even further. Our analysis of deterministic state spaces is very detailed, because we can then transfer its results to $\hat{Q}$-learning in arbitrary non-deterministic state spaces.

6.1.1. Zero-Initialized Q-Values with Action-Penalty Representation

In this section, we analyze zero-initialized Q- or $\hat{Q}$-learning that operates on the action-penalty representation. Since the agent receives a non-zero immediate reward for every action execution, the Q-values change immediately.

6.1.1.1. Complexity Analysis

We first define admissible Q- or $\hat{Q}$-learning, show that undiscounted admissible Q- or $\hat{Q}$-learning is tractable in deterministic state spaces, and state how its complexity depends on properties of the state space. Then, we show how the analysis can be applied to discounted Q- or $\hat{Q}$-learning.

Definition. Undiscounted Q- or Q-learning with action-penalty representation is **admissible** in deterministic state spaces[3] iff either

1. its initial Q-values are consistent and its value-update step[4] is "Set $Q(s,a) := -1 + U(s')$," or
2. its initial Q-values are admissible and its value-update step is "Set $Q(s,a) := \min(Q(s,a), -1 + U(s'))$."

The value-update step in the first part of the definition is the one of Q-learning with learning rate one. The value-update step in the second part of the definition is the one of $\hat{Q}$-learning. If the Q-values are consistent, then either value-update step can be used, since consistent Q-values are always admissible. The most important property of admissible Q- or $\hat{Q}$-learning is that initially consistent or admissible Q-values remain consistent or admissible, respectively, after every action execution and are monotonically non-increasing.

The correctness of admissible Q- or $\hat{Q}$-learning for goal-directed exploration is easy to show. The argument that it reaches a goal state eventually parallels a similar argument for RTA*-type search algorithms (Korf, 1990) (Russell and Wefald, 1991) and is by contradiction: If the agent did not reach a goal state eventually, it would execute actions forever. In this case, there is a time t from which on the agent only executes those actions that it executes infinitely often. Every time the agent has executed all of these actions at least once, the largest Q-value of these actions has decreased by at least one. Eventually, the Q-values of all these actions drop below every bound. In particular, they drop below the Q-value of an action that Q- or $\hat{Q}$-learning considers infinitely often for execution, but never executes after time t. Such an action exists, since the state space is safely explorable and thus the agent can always reach a goal state. Then, however, Q- or $\hat{Q}$-learning is forced to execute this action after time t, which is a contradiction.

To understand intuitively why admissible Q- or $\hat{Q}$-learning is tractable, assume that it is zero-initialized and consider the set $X := \{s \in S : U(s) = 0\} \supset G$. X is always the set of states in which the agent has not yet executed all of the available actions at least once. At every point in time, the following relationship holds:

$$-1 - \min_{s' \in X} d(succ(s,a), s') < Q(s,a) \leq 0 \qquad \text{for all } s \in S \setminus G \text{ and } a \in A(s).$$

The action-selection step can be interpreted as using $Q(s,a)$ to approximate $-1 - \min_{s' \in X} d(succ(s,a), s'))$. Since it always executes the action with the largest Q-value, it tries (sometimes unsuccessfully) to direct the agent with as few action executions as possible from its current state to the closest state that has at least one **unexplored action** (an action that it has never executed before) and make it then take the unexplored action. Since any unexplored action can potentially lead to a goal state, Q- or $\hat{Q}$-learning always executes the action that appears to be best according to its local view of the state space. This suggests that admissible Q- or $\hat{Q}$-learning might have a low complexity. We therefore conduct a formal complexity analysis. It is centered around the invariant shown in Lemma 1. This lemma states that the number of executed actions is always bounded by an expression that depends only on the initial and current Q-values and, moreover, that "every action execution decreases the sum of all Q-values by one, except for a bounded number of action executions that leave the sum unchanged" (this paraphrase is somewhat simplified). A time superscript of t in Lemmas 1 and 2 refers to the values of the variables immediately before the agent executes the $(t+1)$st action (e.g. $s^0 = s_{start}$).

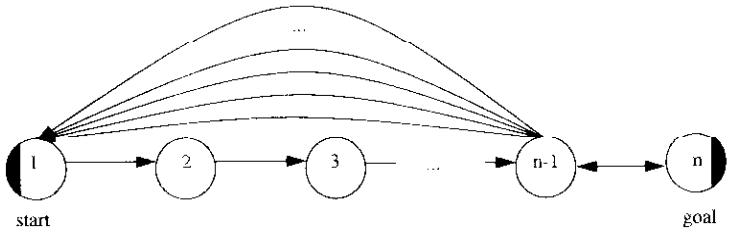

Figure 3. A state space for which uninformed Q- or $\hat{Q}$-learning can need at least $(e - n + 1)(n - 1)$ action executions to reach the goal state (for $n \geq 2$ and $e \geq n$)

LEMMA 1 *For all times $t \in \mathcal{N}_0$ (until termination) of undiscounted admissible Q- or $\hat{Q}$-learning with action-penalty representation in deterministic state spaces, it holds that*

$$U^t(s^t) + \sum_{s \in S} \sum_{a \in A(s)} Q^0(s,a) - t \geq \sum_{s \in S} \sum_{a \in A(s)} Q^t(s,a) + U^0(s^0) - loop^t$$

and

$$loop^t \leq \sum_{s \in S} \sum_{a \in A(s)} Q^0(s,a) - \sum_{s \in S} \sum_{a \in A(s)} Q^t(s,a),$$

where $loop^t := |\{t' \in \{0, \ldots, t-1\} : s^{t'} = s^{t'+1}\}|$ (the number of actions executed before t that did not change the state).

Lemma 2 uses Lemma 1 to derive a bound on t. This is possible, because "the sum of all Q-values decreases by one for every executed action, ..." (according to the invariant), but is bounded from below. That each of the e different Q-values is bounded from below by an expression that depends only on the goal distances follows directly from the definition of consistent or admissible Q-values and the fact that the Q-values maintain this property after every action execution.

LEMMA 2 *Undiscounted admissible Q- or $\hat{Q}$-learning with action-penalty representation reaches a goal state after at most*

$$2 \sum_{s \in S \setminus G} \sum_{a \in A(s)} [Q^0(s,a) + gd(succ(s,a)) + 1] - U^0(s^0)$$

action executions in deterministic state spaces.

Theorem 2 uses Lemma 2 and the fact that $gd(s) \leq d$ for all $s \in S$ to state how the complexity of Q- or $\hat{Q}$-learning depends on e and d. Theorem 2 also guarantees that admissible Q- or $\hat{Q}$-learning terminates in safely explorable state spaces, because d is finite for such state spaces.

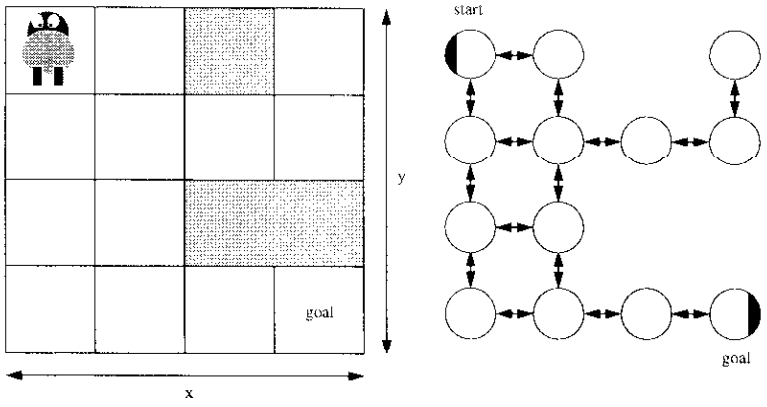

Figure 4. An example of a rectangular gridworld

THEOREM 2 *Undiscounted admissible Q- or Q̂-learning with action-penalty representation reaches a goal state after at most $O(ed)$ action executions in deterministic state spaces.*

Theorem 2 provides an upper bound on the number of action executions that an agent controlled by Q- or Q̂-learning needs to reach a goal state. (Such a worst-case bound is, of course, also a bound on its average-case performance.) To demonstrate that $O(ed)$ is a tight bound for uninformed Q- or Q̂-learning, we show that it is also a lower bound. Lower bounds can be proven by example, one of which is depicted in Figure 3. In this state space, uninformed Q- or Q̂-learning and every other uninformed exploration algorithm can make the agent traverse a supersequence of the following state sequence if ties are broken in favor of actions that lead to states with smaller numbers: $e - n + 1$ times the sequence $123 \ldots n - 1$, and finally n. Basically, all of the $O(e)$ actions in state $n - 1$ are executed once. All of these (but one) lead the agent back to state 1 and therefore force it to execute another $O(d)$ actions before it can execute the next unexplored action, resulting in a total of $O(ed)$ action executions before it reaches the goal state. Thus, the following corollary holds:

COROLLARY 1 *Zero-initialized (or (minus one)-initialized) undiscounted Q- or Q̂-learning with action-penalty representation has a tight complexity of $O(ed)$ action executions for reaching a goal state in deterministic state spaces.*

This complexity can be expressed as a function of the size of the state space: $O(ed) \leq O(en)$, since $d < n - 1$ in all safely explorable state spaces. A state space has no duplicate actions iff either $a = a'$ or $succ(s, a) \neq succ(s, a')$ for all $s \in S$ and $a, a' \in A(s)$. $O(en) < O(n^3)$ for such state spaces, since $e \leq n^2$. The complexity of uninformed Q- or Q̂-learning is even lower in many state spaces that are typically used by reinforcement-learning researchers, because (a) their number of edges often increases only linearly with

the number of states, and/or (b) their diameter increases only sublinearly in n. Consider for example deterministic gridworlds with discrete states. They have often been used in studying reinforcement learning; see (Barto, Sutton, and Watkins, 1989), (Sutton, 1990), (Peng and Williams, 1992), (Singh, 1992), (Thrun, 1992b), or (Whitehead, 1992). In the state space shown in Figure 4, for example, the agent can move from any square to each of its four neighboring squares as long as it stays on the grid and the target square does not contain an obstacle. Thus, the number of actions increases at most linearly with the number of states. A rectangular obstacle-free gridworld of size $x \times y$ has $n = xy$ states, $e = 4xy - 2x - 2y$ state-action pairs, and diameter $d = x + y - 2$. If the gridworld is square ($x = y$), then its diameter increases sublinearly in n, since $d = 2\sqrt{n} - 2$, and $O(ed) = O(x^3) = O(n^{3/2})$. Even for safely-explorable rectangular gridworlds that contain arbitrary obstacles, the complexity of Q- or $\hat{\text{Q}}$-learning cannot be larger than $O(n^2)$ action executions, since $e \leq 4n$, $d \leq n - 1$, and consequently $ed \leq 4n^2 - 4n$. Therefore, Q- or $\hat{\text{Q}}$-learning actually has a very low complexity for finding goal states in unknown gridworlds.

6.1.1.2. Generalization of the Complexity Analysis

We can reduce the analysis of discounted Q- or $\hat{\text{Q}}$-learning with action-penalty representation to the one of undiscounted Q- or $\hat{\text{Q}}$-learning with the same reward structure. Consider the following strictly monotonically increasing bijection from the Q-values of a discounted Q- or $\hat{\text{Q}}$-learning algorithm to the Q-values of the same algorithm with discount rate one: map a Q value $Q_1^t(s,a) \in (1/(\gamma - 1), 0]$ of the former algorithm to the Q-value $Q_2^t(s,a) = -\log_\gamma(1 + (1-\gamma)Q_1^t(s,a)) \in \mathcal{R}_0^-$ of the latter algorithm. If this relationship holds initially for the Q-values of the two algorithms, it continues to hold: if the execution of action a in state s results in successor state s', then either none of the Q-values changes or only the Q-values $Q_1(s,a)$ and $Q_2(s,a)$ change, in which case

$$\begin{aligned}
Q_2^{t+1}(s,a) &= -1 + U_2^t(s') \\
&= -1 + \max_{a' \in A(s')} Q_2^t(s',a') \\
&= -1 + \max_{a' \in A(s')} (-\log_\gamma(1 + (1-\gamma)Q_1^t(s',a'))) \\
&= -\log_\gamma(1 + (1-\gamma)(-1 + \gamma \max_{a' \in A(s')} Q_1^t(s',a'))) \\
&= -\log_\gamma(1 + (1-\gamma)(-1 + \gamma U_1^t(s'))) \\
&= -\log_\gamma(1 + (1-\gamma)Q_1^{t+1}(s,a)).
\end{aligned}$$

Because both algorithms always execute the action with the largest Q-value in the current state, they always choose the same action for execution (if ties are broken in the same way). Thus, they behave identically and the complexity analysis of the latter algorithm also applies to the former algorithm. This means that one can proceed as follows: Given initial Q values $Q_1^0(s,a)$ for Q- or $\hat{\text{Q}}$-learning with discounting, one first

determines the corresponding Q-values $Q_2^0(s,a)$ according to the above formula. These Q-values can then be tested for their consistency or admissibility and finally be used in the formulas of Lemmas 1 and 2 to determine the complexity of discounted Q- or $\hat{\text{Q}}$-learning with initial Q-values $Q_1^0(s,a)$. If the values $Q_1^0(s,a)$ are zero-initialized, then the corresponding undiscounted Q- or $\hat{\text{Q}}$-learning algorithm has the same initialization, which implies that it has a tight complexity of $O(ed)$ action executions. Consequently, the following corollary holds:

COROLLARY 2 *Zero-initialized discounted Q- or $\hat{\text{Q}}$-learning with action-penalty representation has a tight complexity of $O(ed)$ action executions for reaching a goal state in deterministic state spaces.*

The other results from Section 6.1.1.1. can be transferred in the same way.

6.1.2. *One-Initialized Q Values with Goal Reward Representation*

We have shown that uninformed Q- or $\hat{\text{Q}}$-learning with action-penalty representation is tractable in deterministic state spaces, because the Q values change immediately. Since the value-update step of Q- or $\hat{\text{Q}}$-learning updates the Q-values using both the immediate reward and the Q-values of the successor state, this result suggests that one can achieve tractability not only by changing the reward structure, but also by initializing the Q values differently. In this section we show that, indeed, Q- or $\hat{\text{Q}}$-learning is also tractable if goal-reward representation is used (which implies that discounting has to be used as well, i.e. $\gamma < 1$) and the Q-values are $1/\gamma$ or one initialized. The latter initialization has the advantage that it does not depend on any parameters of Q- or $\hat{\text{Q}}$-learning.

The complexity analysis of $1/\gamma$-initialized Q- or $\hat{\text{Q}}$-learning with goal-reward representation can be reduced to the one of zero initialized Q- or $\hat{\text{Q}}$-learning with action penalty representation. Similarly, the complexity analysis of one-initialized Q- or $\hat{\text{Q}}$-learning with goal-reward representation can be reduced to the one of (minus one)-initialized Q- or $\hat{\text{Q}}$-learning with action penalty representation. In both cases, we can proceed in a way similar to the method of Section 6.1.1.2. This time, we consider the following strictly monotonically increasing bijection from the Q-values of a discounted Q- or $\hat{\text{Q}}$-learning algorithm with goal reward representation to the Q values of an undiscounted Q- or $\hat{\text{Q}}$-learning algorithm with action-penalty representation: map a Q-value $Q_1^t(s,a) \in (0, 1/\gamma]$ (or zero) of the former algorithm to the Q-value $Q_2^t(s,a) = -1 - \log_\gamma Q_1^t(s,a) \in \mathcal{R}^-$ (or zero, respectively) of the latter algorithm.[5] Similarly to the proof sketch in the previous section, one can now easily show that this relationship continues to hold if it holds for the initial Q-values. If the values $Q_1^0(s,a)$ are $1/\gamma$ or one-initialized, then the corresponding Q- or $\hat{\text{Q}}$-learning algorithm with action-penalty representation is zero- or (minus one)-initialized, which implies that it has a tight complexity of $O(ed)$ action executions. Consequently, the following corollary holds:

COROLLARY 3 *One-initialized discounted Q- or $\hat{\text{Q}}$-learning with goal-reward representation has a tight complexity of $O(ed)$ action executions for reaching a goal state in deterministic state spaces.*

The other results from Section 6.1.1.1. can be transferred in the same way.

6.1.3. *Decreasing the Complexity Further*

We have shown that finding a goal state with uninformed Q- or $\hat{Q}$-learning is tractable if an appropriate representation is chosen. In particular, we identified both undiscounted or discounted zero-initialized Q-values with action-penalty representation and discounted one-initialized Q-values with goal-reward representation as representations that make these algorithms tractable (**tractable representations**). In this section, we investigate how their complexity can be decreased even further by utilizing prior knowledge of the state space or by learning and using an action model.

6.1.3.1. *Using Prior Knowledge*

The larger the absolute values of consistent or admissible initial Q-values are, the lower the complexity of Q- or $\hat{Q}$-learning is (if we assume without loss of generality that action-penalty representation and no discounting is used), as can be seen from Lemma 2. For example, in the **completely informed** case, the Q-values are initialized as follows:

$$Q(s,a) = \begin{cases} 0 & \text{for } s \in G \text{ and } a \in A(s) \\ 1 - gd(succ(s,a)) & \text{for } s \in S \setminus G \text{ and } a \in A(s). \end{cases}$$

Lemma 2 predicts in this case correctly that the agent needs at most $-U(s) = gd(s)$ action executions to reach a goal state from any given $s \in S$. Often, however, the agent will only have partial prior knowledge of the state space. Heuristic functions that approximate the goal distances and are consistent or admissible for A*-search (Nilsson, 1971) can be used to encode such prior knowledge. Consistent or admissible heuristics are known for many deterministic state spaces, for example for path planning problems in spatial domains (Pearl, 1984). If a heuristic h (with $h(s) > 0$ for all $s \in S$) is consistent or admissible for A*-search, then the following Q-values are consistent or admissible, respectively, as well:

$$Q(s,a) \quad \begin{cases} 0 & \text{for } s \in G \text{ and } a \in A(s) \\ -1 \quad h(succ(s,a)) & \text{for } s \in S \setminus G \text{ and } a \in A(s). \end{cases}$$

Thus, consistent or admissible heuristics can be used to initialize the Q-values, which makes Q- or $\hat{Q}$-learning better informed and lowers its complexity.

6.1.3.2. *Using Action Models*

Although the complexity of uninformed Q- or $\hat{Q}$-learning is a small polynomial in the size of the state space if it operates on a tractable representation, it often directs the agent to execute actions that are suboptimal when judged according to the knowledge that the agent could have acquired had it memorized all of its experience. In particular, the agent

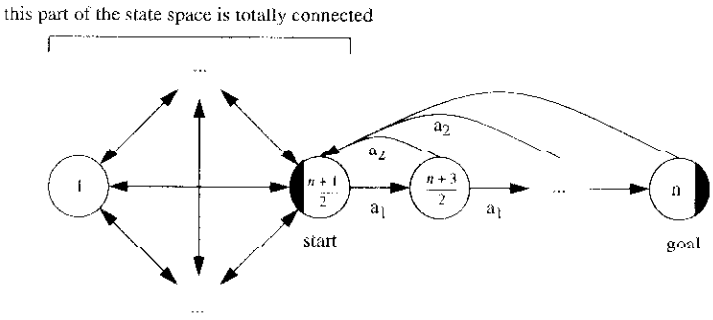

Figure 5. A state space for which uninformed Q- or $\hat{Q}$-learning can need at least $1/16n^3 - 3/16n^2 - 1/16n + 3/16$ action executions to reach the goal state (even if it uses a tractable representation), but Q_{map}-learning needs only at most $3/8n^2 + 3/2n - 23/8$ action executions (for odd $n \geq 3$)

can move around for a long time in parts of the state space that it has already explored. This inefficiency can be avoided by increasing the amount of planning performed between action executions, because it enables the agent to make more informed decisions about which action to execute next. To be able to plan between action executions, the agent has to learn an action model. In this section, we show that learning and using an action model can decrease the complexity of Q- or $\hat{Q}$-learning.

Sutton's **DYNA architecture** (Sutton, 1990 and 1991), for example, implements planning in the framework of Q- or $\hat{Q}$-learning. The learned action model is used to simulate action executions and thereby to create experiences that are indistinguishable from the execution of actions in the world. This way, the world and its model can interchangeably be used to provide input for Q- or $\hat{Q}$-learning. Actions are executed in the world mainly to refine (and, in non-stationary environments, to update) the model. Using the model, the agent can optimize its behavior according to its current knowledge without having to execute actions in the world. In particular, the model allows the agent to simulate any action at any time, whereas the world constrains the agent to execute only actions that are available in its current state. Various researchers, for example Peng and Williams (1992), Yee (1992), and Moore and Atkeson (1993b), have devised strategies for choosing which actions to simulate in order to speed up planning.

Q_{map}-learning is an algorithm that fits the DYNA framework. It remembers the action model of the part of the state space that it has already explored and executes unexplored actions in the same order as Q- or $\hat{Q}$-learning, but always chooses the shortest known path to the next unexplored action. Q_{map}-learning uses its current (incomplete) action model to simulate the behavior of Q- or $\hat{Q}$-learning until it would execute an unexplored action. Then, it uses the same action model to find the shortest known action sequence that leads from its current state in the world to the state in which it can execute this action, directs the agent to execute the action sequence and the unexplored action, and repeats the cycle. Per construction, Q_{map} learning cannot perform worse than Q or $\hat{Q}$ learning

if ties are broken in the same way, but it can be more efficient: Consider, for example, the state sequence that uninformed Q- or $\hat{Q}$-learning on a tractable representation traverses in a reset state space (shown in Figure 2) of size $n = 6$ if ties are broken in favor of actions that lead to states with smaller numbers: 121231234123451212345 6. First, Q- or $\hat{Q}$-learning finds out about the effect of action a_1 in state 1 and then about a_2 in 2, a_1 in 2, a_2 in 3, a_1 in 3, a_2 in 4, a_1 in 4, a_2 in 5, and a_1 in 5, in this order. Q_{map}-learning explores the actions in the same order. However, after it has executed action a_2 in state 5 for the *first time* and, as a *consequence*, got reset into state 1, it reaches state 5 again faster than Q- or $\hat{Q}$-learning: it goes from state 1 through states 2, 3, and 4, to state 5, whereas Q- or $\hat{Q}$-learning goes through states 2, 1, 2, 3, and 4. Thus, Q_{map}-learning traverses the state sequence 12123123412345123456, and is two action executions faster than Q- or $\hat{Q}$-learning. Figure 5 gives an example of a state space for which the big-O complexities of the two algorithms differ: there is a tie-breaking rule that causes Q- or $\hat{Q}$-learning to need $O(n^3)$ action executions to reach the goal state (even if it uses a tractable representation), whereas Q_{map}-learning needs at most $O(n^2)$ action executions no matter how ties are broken; see (Koenig and Simmons, 1992) for the proof. This shows that it is possible to augment Q- or $\hat{Q}$-learning with a component that learns action models such that the resulting algorithm never performs worse than Q- or $\hat{Q}$-learning, but often performs better. In some state spaces, one can reduce the number of action executions even by more than a constant factor. There are "hard" state spaces, however, for which no uninformed algorithm can have a lower big-O complexity than uninformed Q- or $\hat{Q}$-learning with a tractable representation (an example is the state space shown in Figure 3).

Note that we have not included the planning time in the complexity measure. When learning an action model and using it for planning, the agent has to keep more information around and perform more computations between action executions. However, chances are that the increased deliberation time reduces the number of action executions that are needed for reaching a goal state. Consequently, using an action model can be advantageous if executing actions in the world is slow (and expensive), but simulating the execution of actions in a model of the world is fast (and inexpensive). If planning time is not negligible compared to execution time, then there is a trade-off: simulating actions allows the agent to utilize its current knowledge better, whereas executing actions in the world increases its knowledge and allows it to stumble across a goal. Also, the fewer actions are simulated between action executions, *the less opportunity a non-stationary state* space has to change and the closer the model still reflects reality. Planning approaches that take these kinds of trade-offs into account have, for example, been investigated by Boddy and Dean (1989), Russell and Wefald (1991), Goodwin (1994), and Zilberstein (1993).

6.2. *Non-Deterministic State Spaces*

So far, we have studied representations that make Q- or $\hat{Q}$-learning tractable in deterministic state spaces. We now show that our analysis can easily be generalized to $\hat{Q}$-learning

in non-deterministic state spaces. In non-deterministic state spaces, admissible $\hat{Q}$-learning is defined as follows:

Definition. Undiscounted $\hat{Q}$-learning with action-penalty representation is **admissible** in non-deterministic state spaces iff its initial Q-values are admissible and its value-update step is "Set $Q(s,a)$: $\min(Q(s,a), -1 + U(s'))$."

This definition uses the value-update step of undiscounted $\hat{Q}$-learning and is identical to the second part of the definition of admissible Q or $\hat{Q}$ learning in the deterministic case (Page 236). It allows us to transfer the complexity analysis from deterministic to non-deterministic state spaces. In particular, our complexity analysis continues to hold if we replace $gd(succ(s,a))$ with $\max_{s' \in succ(s,a)} gd(s')$, which reflects that we perform a worst-case analysis over all strategies of nature; and undiscounted admissible $\hat{Q}$-learning with action-penalty representation reaches a goal state after at most

$$2 \sum_{s \in S \setminus G} \sum_{a \in A(s)} [Q^0(s,a) + \max_{s' \in succ(s,a)} gd(s') + 1] - U^0(s^0)$$

action executions in non-deterministic state spaces no matter which strategy nature uses. Therefore, its complexity is at most $O(ed)$ action executions, just like in the deterministic case. (This is due to our definition of d that assumes that nature is an opponent of the agent and encompasses deterministic state spaces as a special case.) The complexity is tight for uninformed $\hat{Q}$-learning, because it is tight for deterministic state spaces and cannot be lower in non-deterministic state spaces. We can use the transformations of the Q-values that we used in Sections 6.1.1.2. and 6.1.2 to show that the same complexity result holds for discounted $\hat{Q}$-learning in non-deterministic state spaces if either zero-initialized Q-values and action-penalty representation or one-initialized Q-values and goal-reward representation is used. Consequently, the following corollary holds:

COROLLARY 4 *Zero-initialized undiscounted or discounted $\hat{Q}$-learning with action-penalty representation and one-initialized discounted $\hat{Q}$-learning with goal-reward representation have a tight complexity of $O(ed)$ action executions for reaching a goal state in non-deterministic state spaces.*

This complexity result is fairly general, since it provides an upper bound on the number of action executions that holds for all strategies of nature. For example, it holds for both nature's strategy to select successor states randomly (the assumption underlying Q-learning) and its strategy to select successor states deliberately so that it hurts the agent most (the assumption underlying $\hat{Q}$-learning). But it also applies to scenarios where the agent cannot make assumptions about nature's strategy. Assume, for example, a deterministic world which the agent can only model with low granularity. Then, it might not be able to identify its current state uniquely, and actions can appear to have non-deterministic effects: sometimes the execution of an action results in one successor state, sometimes in another. The agent has no way of predicting which of the potential successor states will occur and could attribute this to nature having a strategy that is unknown to the agent. Our complexity bound holds even in this case.

7. Extensions

Space limitations forced us to limit our presentation to an analysis of the goal-directed exploration behavior of two reinforcement-learning algorithms (Q-learning and $\hat{Q}$-learning) for two different reward structures (action-penalty and goal-reward representation). We have generalized this analysis in three orthogonal directions:

- *Reward Structure*: In our analysis, we have assumed that one can choose an appropriate reward structure when representing a reinforcement-learning or exploration problem. Although this is usually true, sometimes the reward structure is given. In this case, even if the reward structure is dense, the immediate rewards might not all be uniformly minus one, as assumed by our analysis of Q- or $\hat{Q}$-learning with action-penalty representation. The results presented in this article have been generalized to cover dense reward structures with non-uniform costs. The complexity of zero-initialized Q- or $\hat{Q}$-learning with a non-uniform action-penalty representation, for example, is tight at $O(ed)$ action executions, where d is now the weighted diameter of the state space divided by the smallest absolute value of all immediate costs; see (Koenig and Simmons, 1992).

- *Exploration Algorithm*: We have restricted our analysis to two reinforcement-learning algorithms, Q-learning and $\hat{Q}$-learning. Q-learning behaves very similarly to value-iteration (Bellman, 1957), an algorithm that does not use Q-values, but rather the U-values directly. 1-step on-line value-iteration with action-penalty representation behaves in deterministic state spaces identically to Korf's Learning Real-Time A* (LRTA*) search algorithm (Korf, 1990) with search horizon one. Korf showed that LRTA* is guaranteed to reach a goal state and, if it is repeatedly reset into the start state when it reaches a goal state, eventually determines a shortest path from the start state to a goal state. Subsequently, Barto, Bradtke, and Singh (1995) generalized these results to probabilistic state spaces. Since on-line value-iteration and, consequently, LRTA* behave like 1-step Q-learning in a modified state space (Koenig and Simmons, 1992), we have been able to transfer our complexity results to LRTA* and, in the process, generalized previous complexity results for LRTA* by Ishida and Korf (1991), see (Koenig and Simmons, 1995b). The complexity of zero-initialized LRTA* and uninformed value-iteration that operates on a tractable representation is tight at $O(nd)$ action executions; see (Koenig and Simmons, 1995b).

- *Task*: We have analyzed goal-directed exploration problems, because these tasks need to be solved if one wants to solve goal-directed reinforcement-learning problems in unknown state spaces. Our analysis generalizes to finding optimal policies, because this problem can be solved either by repeatedly resetting the agent into its start state when it reaches a goal state (if the task is to find an optimal behavior from the start state) or, if such a reset action is not available, by iteratively executing two independent exploration algorithms: one Q- or $\hat{Q}$-learning algorithm that finds a goal state, and another one that finds a state for which the optimal action assignment has not yet been determined. (The latter method determines optimal behaviors from

all states in strongly connected state spaces.) In both cases, we have been able to show that the complexity of uninformed Q- or $\hat{Q}$-learning that operates on a tractable representation remains tight at $O(ed)$, see (Koenig and Simmons, 1993). (For non-deterministic state spaces, one has to make assumptions about how long nature is allowed to "trick" the agent by never choosing a bad action outcome and thereby hiding its existence.)

8. Conclusion

In this article, we have analyzed how 1-step reinforcement-learning methods solve goal-directed reinforcement-learning problems in safely explorable state spaces – we have studied their behavior until they reach a goal state for the first time. In particular, we studied how the complexity of Q-learning methods (measured in action executions), such as Watkins' Q-learning or Heger's $\hat{Q}$-learning, depends on the number of states of the state space (n), the total number of state-action pairs (e), and the largest goal distance (d). When formulating a goal-directed reinforcement-learning problem, one has to decide on an appropriate representation, which consists of both the immediate rewards that the reinforcement-learning algorithms receive and their initial Q-values. We showed that the choice of representation can have a tremendous impact on the performance of Q- or $\hat{Q}$-learning.

We considered two reward structures that have been used in the literature to learn optimal policies, the goal-reward representation and the action-penalty representation. In the action-penalty representation, the agent is penalized for every action that it executes. In the goal-reward representation, it is rewarded for entering a goal state, but not rewarded or penalized otherwise. Zero-initialized Q-learning with goal-reward representation provides only sparse rewards. Even in deterministic state spaces, it performs a random walk. Although a random walk reaches a goal state with probability one, its complexity is infinite and even its average number of action executions can be exponential in n. Furthermore, speed-up techniques such as Lin's action-replay technique do not improve its performance. This provides motivation for making the reward structure dense. Since the value-update step of Q- or $\hat{Q}$-learning updates the Q-values using both the immediate reward and the Q-values of the successor state, this can be achieved by either changing the reward structure or initializing the Q-values differently. And indeed, we showed that both (undiscounted and discounted) zero-initialized Q- or $\hat{Q}$-learning with action-penalty representation and (discounted) one-initialized Q- or $\hat{Q}$-learning with goal-reward representation are tractable. For the proof, we developed conditions on the initial Q-values, called consistency and admissibility, and - for initial Q-values with these properties - a time invariant relationship between the number of executed actions, the initial Q-values, and the current Q-values. This relationship allowed us to express how the complexity of Q- or $\hat{Q}$-learning depends on the initial Q-values and properties of the state space. Our analysis shows that, if a tractable representation is used, the greedy action-selection strategy of Q- or $\hat{Q}$-learning always executes the action that locally appears to be best, and even uninformed Q- or $\hat{Q}$-learning has a complexity of at most $O(ed)$ action executions in deterministic state spaces. The same result holds for

$\hat{Q}$-learning in non-deterministic state spaces, in which we viewed reinforcement learning as a game where the reinforcement-learning algorithm selects the actions and "nature," a fictitious second agent, selects their outcomes. (The bound holds for all possible outcome selection strategies of nature.)

If a safely explorable state space has no duplicate actions, then $O(ed) < O(n^3)$. The complexity of Q- or $\hat{Q}$-learning is even lower in many state spaces, since e often grows only linearly in n and/or d grows only sublinearly in n. Examples of such state spaces are gridworlds, which are popular reinforcement-learning domains. The complexity can be reduced further by using prior knowledge of the state space and by learning action models. Prior knowledge can be encoded in the initial Q-values by utilizing heuristics that are consistent or admissible for A*-search. Action models predict what would happen if a particular action were executed. We showed how to augment Q- or $\hat{Q}$-learning with a component that learns action models such that the resulting algorithm, which we called Q_{map}-learning, never performs worse than Q- or $\hat{Q}$-learning, but reduces the complexity by at least a factor of n (i.e. by more than a constant factor) in some state spaces.

To summarize, reinforcement learning algorithms are tractable if a suitable representation is chosen. Our complexity results provide guidance for empirical reinforcement-learning researchers on how to model reinforcement learning problems in a way that allows them to be solved efficiently – even for reinforcement-learning tasks that cannot be reformulated as goal-directed reinforcement-learning or exploration problems in safely explorable state spaces: the performance can be improved by making the reward structure dense. Our results also characterize which properties of state spaces make them easy to solve with reinforcement-learning methods, which helps empirical reinforcement-learning researchers to choose appropriate state spaces for their experiments.

Acknowledgments

Avrim Blum, Lonnie Chrisman, Matthias Heger, Long-Ji Lin, Michael Littman, Andrew Moore, Martha Pollack, and Sebastian Thrun provided helpful comments on the ideas presented in this article. Special thanks to Sebastian Thrun for stimulating discussions and to Lonnie Chrisman also for taking the time to check the proofs.

Notes

1. To be precise: it does not matter whether states that the agent cannot possibly reach from its start state are lost. Furthermore, for goal-directed exploration problems, we can disregard all states that the agent can only reach by passing through a goal state, since the agent can never occupy those states. We assume without loss of generality that all such states have been removed from the state space.
2. A similar statement also holds for $\hat{Q}$-learning: it never changes a Q-value. However, since the Q-values are not initialized optimistically, zero-initialized $\hat{Q}$-learning with goal-reward representation cannot be used to learn shortest paths. Studying the goal-directed exploration problem for $\hat{Q}$-learning with this representation therefore does not provide any insight into the corresponding reinforcement-learning problem.

3. Our analysis can be generalized to goal-directed reinforcement-learning problems. For goal-directed exploration problems, the definition of admissible Q- or $\hat{Q}$-learning can be broadened, since one can add the same constant to all Q-values without changing the behavior of Q- or $\hat{Q}$-learning until it reaches a goal state for the first time. Given initial Q-values for a goal-directed exploration problem, one can therefore add a constant to all Q-values before determining whether they are consistent or admissible.
4. In other words: the value-update step is "Set $Q(s,a) := (1-\alpha)Q(s,a) + \alpha(r(s,a,s') + \gamma U(s'))$," where $r(s,a,s') = -1$ and $\alpha = \gamma = 1$.
5. Our analysis can be generalized to goal-directed reinforcement-learning problems. For goal-directed exploration problems, the statement can be broadened as follows: According to Note 3, one can add the same constant to all values $Q_2^0(s,a)$ without changing the behavior of undiscounted Q- or $\hat{Q}$-learning with action-penalty representation until it reaches a goal state for the first time. Since $Q_2^t(s,a) - \log_\gamma c = (-1 + \log_\gamma Q_1^t(s,a)) - \log_\gamma c = -1 + \log_\gamma c Q_1^t(s,a)$, one can multiply all values $Q_1^0(s,a)$ with some positive constant c without changing the behavior of discounted Q- or $\hat{Q}$-learning with goal-reward representation until it reaches a goal state for the first time.

References

Barto, A.G., S.J. Bradtke, and S.P. Singh. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 73(1):81–138.

Barto,A.G., R.S. Sutton, and C.J. Watkins. (1989). Learning and sequential decision making. Technical Report 89–95. Department of Computer Science, University of Massachusetts at Amherst.

Bellman,R. (1957). *Dynamic Programming*. Princeton University Press, Princeton (New Jersey).

Buddy,M. and T. Dean. (1989). Solving time-dependent planning problems. In *Proceedings of the IJCAI*, pages 979–984.

Goodwin, R. (1994). Reasoning about when to start acting. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 86–91.

Heger,M. (1996). The loss from imperfect value functions in expectation-based and minimax-based tasks. *Machine Learning*, pages 197-225.

Heger,M. (1994). Consideration of risk in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 105–111.

Ishida,T. (1992). Moving target search with intelligence. In *Proceedings of the AAAI*, pages 525–532.

Ishida,T. and R.E. Korf. (1991). Moving target search. In *Proceedings of the IJCAI*, pages 204–210.

Kaelbling, L.P. (1990). *Learning in Embedded Systems*. MIT Press, Cambridge (Massachusetts).

Koenig,S. (1991). Optimal probabilistic and decision-theoretic planning using Markovian decision theory. Master's thesis, Computer Science Department, University of California at Berkeley. (Available as Technical Report UCB/CSD 92/685).

Koenig,S. and R.G. Simmons. (1992). Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains. Technical Report CMU–CS–93–106, School of Computer Science, Carnegie Mellon University.

Koenig,S. and R.G. Simmons. (1993). Complexity analysis of real-time reinforcement learning. In *Proceedings of the AAAI*, pages 99–105.

Koenig,S. and R.G. Simmons. (1994). How to make reactive planners risk-sensitive. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 293–298.

Koenig,S. and R.G. Simmons. (1995a). The effect of representation and knowledge on goal-directed exploration with reinforcement learning algorithms: The proofs. Technical Report CMU–CS–95–177, School of Computer Science, Carnegie Mellon University.

Koenig,S. and R.G. Simmons. (1995b). Real time search in non-deterministic domains. In *Proceedings of the IJCAI*, pages 1660–1667.

Korf,R.E. (1990). Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211.

Lin, L.J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8:293–321.

Mataric,M. (1994). *Interaction and Intelligent Behavior*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

Moore,A.W. and C.G. Atkeson. (1993a). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Proceedings of the NIPS*.

Moore,A.W. and C.G. Atkeson. (1993b). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130.

Nilsson,N.J. (1971). *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York (New York).

Pearl,J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Menlo Park (California).

Peng,J. and R.J. Williams. (1992). Efficient learning and planning within the DYNA framework. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 281–290.

Russell,S. and E. Wefald. (1991). *Do the Right Thing – Studies in Limited Rationality*. MIT Press, Cambridge (Massachusetts).

Singh,S.P. (1992). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the AAAI*, pages 202–207.

Sutton,R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the International Conference on Machine Learning*, pages 216–224.

Sutton,R.S. (1991). DYNA, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163.

Thrun,S.B. (1992a). Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon University.

Thrun,S.B. (1992b). The role of exploration in learning control with neural networks. In D.A. White and D.A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages 527–559. Van Nostrand Reinhold, New York (New York).

Watkins,C.J. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge University.

Watkins,C.J. and P. Dayan. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

Whitehead,S.D. (1991a). A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the AAAI*, pages 607–613.

Whitehead,S.D. (1991b). A study of cooperative mechanisms for faster reinforcement learning. Technical Report 365, Computer Science Department, University of Rochester.

Whitehead,S.D. (1992). *Reinforcement Learning for the Adaptive Control of Perception and Action*. PhD thesis, Computer Science Department, University of Rochester.

Yee, R. (1992). Abstraction in control learning. Technical Report 92–16, Department of Computer Science, University of Massachusetts at Amherst.

Zilberstein, S. (1993). *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, Computer Science Department, University of California at Berkeley.

Received November 3, 1994
Accepted March 10, 1995
Final Manuscript October 17, 1995