Exploration with and without a Map*

Sven Koenig and Reid G. Simmons
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
skoenig@cs.cmu.edu, reids@cs.cmu.edu

Abstract

We investigate the problem for an agent to reach one of a number of goal states by taking actions, where actions cause a deterministic state change. Initially, the topology of the state space and its size are unknown to the agent. We compare a zero-initialized version of Q-learning, that minimizes deliberation time between action executions, with other uninformed search algorithms (i.e. where the effects of actions are initially unknown). We show that the big-O worst-case complexity of every uninformed search algorithm over *all* domains is at least as large as the big-O worst-case complexity of Q-learning. However, learning and subsequently using a map of the state space can provide a search algorithm with an advantage over Q-learning in *some* (but not all) domains. Formally, we show that there exists an uninformed search algorithm that dominates Q-learning, i.e. always performs no worse, but performs strictly better in at least one case. In particular, there is at least one domain in which the number of action executions can be reduced by more than a constant factor.

Introduction

Consider the problem for an agent of reaching one of a number of goal states by taking actions, where actions cause a deterministic state change. Initially, the topology of the state space is unknown to the agent, and the agent therefore has to explore it until it finds a goal state.

We are interested in the worst-case complexity of uninformed search algorithms for this problem. We define an **uninformed search algorithm** to be an algorithm that does not know the effect of an action before it has executed it at least once and observed its effect. By **worst-case complexity** we mean, for a state space of a given size, an upper bound on the number of action executions (steps), until a goal state is reached that holds for all possible topologies of state spaces, start states, goal states, and tie-breaking rules among unexplored actions. Clearly, in order to have a worst-case complexity smaller than infinity, an initially uninformed search algorithm must learn something about the effects of actions.

Zero-initialized Q-learning [Watkins, 1989] is one popular example of such uninformed search algorithms. Its application to path planning problems in discrete state spaces has been studied by [Peng and Williams, 1992], [Sutton, 1990b], [Thrun, 1992], [Whitehead, 1992], and others.

We study a version of zero-initialized Q-learning with very restricted capabilities: 1-step Q-learning [Whitehead, 1991] does not learn a map of the state space. Also, it performs only minimal computation between action executions, choosing only which action to execute next, and basing this decision only on information local to the current state. This way, the time between action executions is linear in the number of actions available in the current state, and the number of steps executed is a good measure for the run-time of the algorithm.

A highly **reactive** algorithm such as 1-step Q-learning often executes actions that are suboptimal (even when judged according to the knowledge that the agent could have when remembering its experiences). The agent can move around for a long time in parts of the state space that it has already explored, thus neither exploring unknown parts of the state space nor having a chance to find a goal. One could hope that this can be avoided when **planning** further into the future: the agent learns a map, i.e. a model of the state space, and uses it subsequently to predict the effects of action executions. This enables the agent to choose more carefully which action to execute next, although it comes at the expense of a larger amount of computations between action executions.

In the framework of Q-learning, for example, Sutton proposed the DYNA architecture [Sutton, 1990a]

*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.

[Sutton, 1990b], which learns a "mental" model of the world (map). Actions are executed in the real world mainly to refine the model. The model is used to simulate the execution of actions and thereby create experiences that are indistinguishable from the execution of real actions. This way, the real world and the model can interchangeably be used to provide input for Q-learning. Using the model, the agent can optimize its behavior (according to its current knowledge) without having to execute actions in the real world. Also, the agent can simulate the execution of arbitrary (not just local) actions at any time. Various researchers, for example [Moore and Atkeson, 1992] and [Peng and Williams, 1992], have devised strategies that determine which actions to simulate in order to speed up planning if the deliberation time between action executions is limited.

To summarize, when learning a map and using it for planning, the agent has to keep more information around and perform more computations between action executions. This paper investigates whether planning decreases the number of steps needed to reach a goal state and, if so, by how much.

In the following, we compare the worst-case complexity of the 1-step Q-learning algorithm, that does not learn a map, to the worst-case complexity of any other uninformed search algorithm, for example one that learns a map and uses it for planning.

Notation and Assumptions

We use the following notation. S denotes the finite set of states of the state space, and G (with $\emptyset \neq G \subseteq S$) is the non-empty set of goal states. $s_{start} \in S$ is the start state of the agent. The size of the state space is $n := |S|$. $A(s)$ is the finite set of deterministic actions that can be executed in $s \in S$. $succ(s, a)$ is the uniquely determined successor state when $a \in A(s)$ is executed in $s \in S$. A **domain** is a state space together with a start state and a set of goal states.

We assume that the state space is strongly connected (or, synonymously, irreducible), i.e. every state can be reached from every other state. We also assume that the state space is totally observable, i.e. the agent can determine its current state s and $A(s)$ with certainty and knows whether it is in a goal state. Furthermore, the domain is single agent and stationary, i.e. it does not change over time. To simplify the following presentation, we also assume that the state space has no duplicate actions, i.e. for all $s \in S$ and $a, a' \in A(s)$, either $a = a'$ or $succ(s, a) \neq succ(s, a')$. This assumption can easily be dropped.

We say that the agent knows a **map** if it is able to predict $succ(s, a)$ for all $s \in S$ and $a \in A(s)$, no matter which state it is in. We say that the agent knows a **distributed map** if it is able to predict $succ(s, a)$ for all $a \in A(s)$ in its current state $s \in S$, but is not able to predict the outcomes of actions that are executed in other states than its current state.

```
Initially, $Q(s,a) = 0$ for all $s \in S$ and $a \in A(s)$. The agent is in state $s_{start}$.
1. Set $s :=$ the current state.
2. If $s \in G$, then stop.
3. Set $a := \text{argmax}_{a' \in A(s)} Q(s,a')$, i.e. select an $a \in A(s)$ with $Q(s,a) = max_{a' \in A(s)} Q(s,a')$. (Ties can be broken arbitrarily.)
4. Execute action $a$. (As a consequence, the agent receives reward $-1$ and is in state $succ(s,a)$.)
5. Set $Q(s,a) := -1 + \gamma \times U(succ(s,a))$.
6. Go to 1.
where $U(s) := \max_{a \in A(s)} Q(s,a)$ at every point in time and $\gamma \in (0,1]$.
```

Figure 1: The 1-step Q-learning algorithm

Q-Learning

The uninformed 1-step **Q-learning algorithm** (in the following just called Q-learning) is shown in Figure 1. It is a dynamic programming algorithm that 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 learning rate α is set to one, since the domain is deterministic.)

With every state-action pair $s \in S$ and $a \in A(s)$, there is one Q-value $Q(s, a)$ associated. $-Q(s, a)$ approximates the smallest number of action executions that is necessary to reach a goal state if the agent starts in s, executes a, and then behaves optimally (provided that the discount factor γ is 1). The Q-values are zero initially, reflecting that the algorithm has no prior knowledge of the state space.

The action selection step always selects the most promising action, which is the action with the largest Q-value. It does not need to predict the effects that actions have. Thus, Q-learning neither learns a map nor a distributed map.

The value update step adjusts $Q(s, a)$, after the selected action a has been executed in state s. The 1-step look-ahead value $-1 + \gamma U(succ(s, a))$ is more accurate than, and therefore replaces, $Q(s, a)$. The -1 is the immediate reward for each action. We use this formulation because, according to a recent result, this Q-learning algorithm has a worst-case complexity of $O(n^3)$ for reaching a goal state, see [Koenig and Simmons, 1993] and for the proofs [Koenig and Simmons, 1992].

Worst-Case Complexity of Uninformed Search Algorithms

In the following, we compare the worst-case complexity of Q-learning to the worst-case complexity of any other uninformed search algorithm.

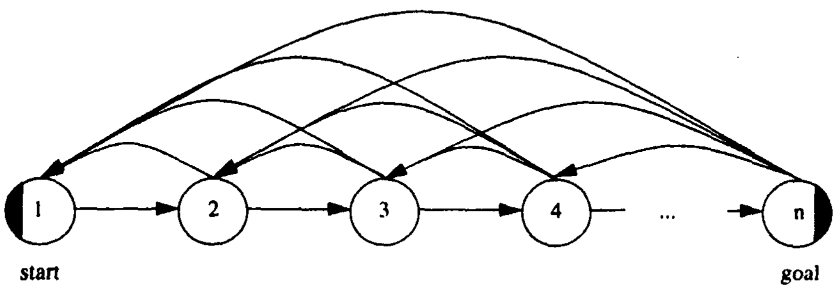

Figure 2: A state space for which every uninformed search algorithm can need at least $1/6n^3 - 1/6n$ steps to reach the goal state (for $n \geq 1$)

Figure 2 shows that *every* uninformed search algorithm has a worst-case complexity of at least $O(n^3)$. If ties are broken in favor of actions that lead to states with smaller numbers, then every uninformed search algorithm can traverse a superset of the following state sequence: $1, 2, 1, 2, 3, 2, 3, 1, 2, 3, 4, \ldots i, i-1, i, i-2, i-1, i, \ldots, 1, 2, 3, \ldots, i-1, i, i+1, \ldots, n-1, n$. Initially, the agent does not know what the effect of each action is. Therefore, it is indifferent between all actions in its current state that it has not executed at least once. In the worst case, it executes all of the actions that lead away from the goal state before the (only) action that leads one step closer to the goal state. Thus, it executes all of the $O(n^2)$ actions in non-goal states at least once. Most of these lead the agent back to a part of the state space that it has already explored and therefore force it to execute $O(n)$ explored actions (namely the ones that let it approach the goal state again) on average before it can explore a new action.

This result shows that the worst-case bound of $O(n^3)$ for zero-initialized Q-learning is tight. Furthermore, planning cannot decrease the big-O worst-case complexity in this particular state space, since the agent can only plan in the part of the state space for which it knows the effects of actions. Theorem 1 follows immediately.

Theorem 1 *No uninformed search algorithm has a smaller big-O worst-case complexity (for arbitrary state spaces) than Q-learning.*

In the following, we will demonstrate that this result *does not* imply that Q-learning is the best possible algorithm for reaching a goal state. To do that, we will show that there exists an uninformed search algorithm that strictly dominates Q-learning. An algorithm X **strictly dominates** an algorithm Y, if X always performs no worse (i.e. needs no more action executions) than Y, and performs strictly better in at least one case.

Consider an algorithm that maintains a map of the part of the state space that it has already explored. It explores unknown actions in the same order as Q-learning, but always chooses the shortest known path to the next unexplored action: It uses its current map and its knowledge of the Q-values to simulate its behavior under the Q-learning algorithm until it would execute an unexplored action. Then, it uses the map to find the shortest known action sequence that leads from its current state in the world to the unexplored action, executes this action sequence followed by the unexplored action, and repeats the cycle. We call this algorithm the **Q_{map}-learning algorithm**. Per construction, it cannot perform worse than Q-learning if ties are broken in the same way (no matter what the tie-breaking rule is). Thus, the worst-case complexity of Q_{map}-learning over all tie-breaking rules of a given domain cannot be worse than that of Q-learning. Consider, for example, the state sequence that Q-learning traverses in the state space shown in Figure 3 of size $n = 6$ if ties are broken in favor of actions that lead to states with smaller numbers: $1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 4, 5, 6$. First, 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. The Q_{map}-learning algorithm explores the actions in the same order. However, after it has executed action a_2 in state 5 for the first time, it knows how to reach state 5 again faster than Q-learning: it goes from state 1 through states 2, 3, and 4, to state 5, whereas Q-learning goes through states 2, 1, 2, 3, and 4. Thus, the Q_{map}-learning algorithm traverses the following state sequence: $1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 6$. and is two steps faster than Q-learning. Figure 4 gives an example of a state space for which the big-O worst-case complexities of the two algorithms are different: Q-learning can require $O(n^3)$ steps to reach the goal state, whereas Q_{map}-learning reaches a goal state with at most $O(n^2)$ steps no matter how ties are broken, see [Koenig and Simmons, 1992] for the proof.

Theorem 2 *There exist uninformed search algorithms that dominate Q-learning, i.e. always perform no worse, but perform strictly better in at least one*

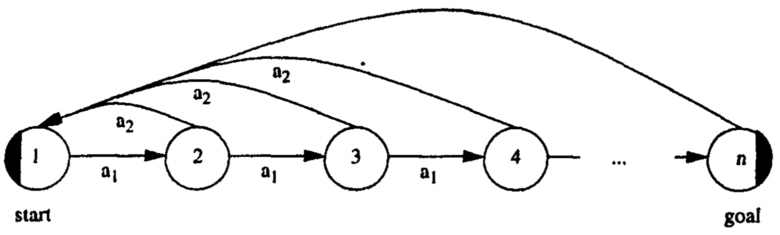

Figure 3: A state space used to compare the behavior of Q-learning and Q_{map}-learning

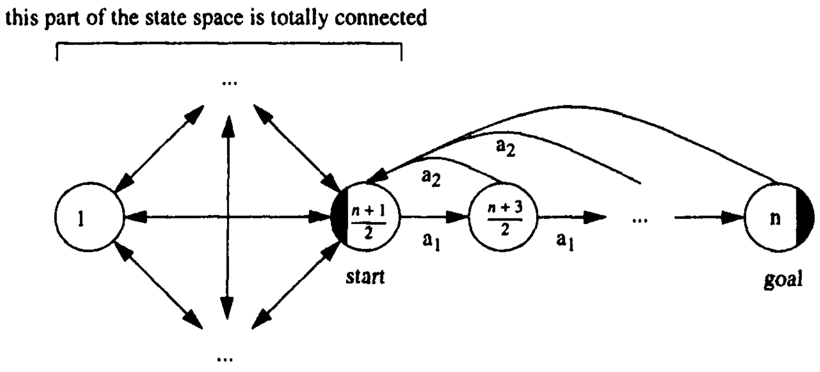

Figure 4: A state space for which the Q-learning algorithm can need at least $1/16n^3 - 3/16n^2 - 1/16n + 3/16$ steps to reach the goal state, but Q_{map}-learning needs only at most $3/8n^2 + 3/2n - 23/8$ steps (for odd $n \geq 3$)

case.

The Q_{map}-learning algorithm is mainly of theoretical interest, because it demonstrates that there exist algorithms that are better than the Q-learning algorithm, and its domination proof is trivial. However, algorithms whose behaviors resemble that of the Q_{map}-learning algorithm are not only of theoretical interest: The idea behind the DYNA architecture is precisely that executing actions in the real world is slow, whereas simulating the execution of actions in a model of the world is fast. Once an action is explored in the real world, it can be integrated in the model. If possible, planning should exclusively be done in the model. Consequently, actions should only be executed in the real world

1. to find out about the effect of an unexplored action,
2. to get the agent into a state in which it can find out about the effect of an unexplored action, or
3. to get to a goal.

Therefore, reinforcement learning researchers, for example [Moore and Atkeson, 1992], have proposed real-time schemes for approximating the following behavior of the agent: "If the current world state is a goal state, stop. Otherwise, go to the closest state with an unexplored action, execute the unexplored action, and repeat." This way, one prevents the agent from unnecessarily executing actions that it has already explored, which is also the objective of the Q_{map}-learning algorithm.

Conclusion

The relationships between the Q-learning algorithm, the Q_{map}-learning algorithm, and uninformed search algorithms in general are summarized in Figure 5. Every uninformed search algorithm (even if it is more powerful than Q-learning in that it keeps more information around and performs more computations between action executions) has at least the same big-O worst-case complexity as the Q-learning algorithm. However, it can be misleading to focus only on the big-O worst-case complexity of a search algorithm *over all domains*, as the Q_{map}-learning algorithm shows. Learning (and subsequently using) a map of the state space can provide the agent with an advantage over Q-learning for some, but not all domains if one is willing to tolerate an increase in deliberation time between action executions. The Q_{map}-learning algorithm, that

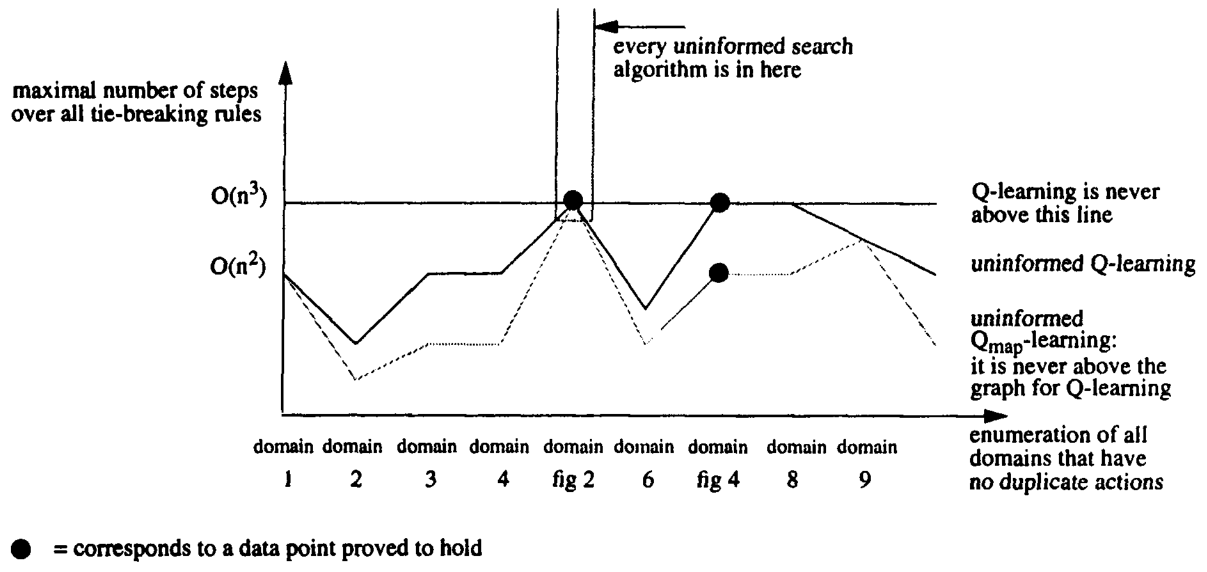

Figure 5: A diagram showing the relationships between the Q-learning algorithm, the Q_{map}-learning algorithm, and uninformed search algorithms in general (data points for domains other than the ones shown in Figures 2 and 4 are fictitious)

learns a map, dominates Q-learning: it always performs no worse, but reduces the number of action executions by more than a constant factor in at least one domain.

Acknowledgments

The Reinforcement Learning Study Group at Carnegie Mellon University provided a stimulating research environment. Avrim Blum, Lonnie Chrisman, Long-Ji Lin, Michael Littman, Joseph O'Sullivan, Martha Pollack, and Sebastian Thrun provided helpful comments on the ideas presented in this paper. In addition, Lonnie Chrisman provided extremely detailed technical comments on the proofs in [Koenig and Simmons, 1992], which improved their presentation.

References

Koenig, Sven and Simmons, Reid G. 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, Sven and Simmons, Reid G. 1993. Complexity analysis of real-time reinforcement learning. In *Proceedings of the AAAI.*

Moore, Andrew W. and Atkeson, Christopher G. 1992. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In *Proceedings of the NIPS.*

Peng, Jing and Williams, Ronald J. 1992. Efficient learning and planning within the Dyna framework. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats.*

Sutton, Richard S. 1990a. First results with DYNA. In *Proceedings of the AAAI Spring Symposium.*

Sutton, Richard S. 1990b. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning.*

Thrun, Sebastian B. 1992. The role of exploration in learning control with neural networks. In White, David A. and Sofge, Donald A., editors 1992, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Van Nostrand Reinhold, Florence, Kentucky.

Watkins, Christopher J. 1989. *Learning from Delayed Rewards.* Ph.D. Dissertation, King's College, Cambridge University.

Whitehead, Steven D. 1991. A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the AAAI.* 607–613.

Whitehead, Steven D. 1992. *Reinforcement Learning for the Adaptive Control of Perception and Action.* Ph.D. Dissertation, Department of Computer Science, University of Rochester.