Reinforcement Learning in Partially Observable Decision Processes

Roy Fox

למידת חיזוקים בתהליכי החלטה נצפים חלקית

רועי פוקס

Reinforcement Learning in Partially Observable Decision Processes

Research Thesis

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Roy Fox

Submitted to the Senate of the Technion – Israel Institute of Technology
Adar II 5768 Haifa April 2008

למידת חיזוקים בתהליכי החלטה נצפים חלקית

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר במדעי המחשב

רועי פוקס

הוגש לסנט הטכניון - מכון טכנולוגי לישראל
אדר ב' ה'תשס"ח חיפה אפריל 2008

The research thesis was done under the supervision of Prof. Moshe Tennenholtz
of the Faculty of Industrial Engineering and Management

I sincerely thank Prof. Moshe Tennenholtz for all his support and help, and all that he taught me

The generous financial help of the Technion is gratefully acknowledged

המחקר נעשה בהנחיית פרופ' משה טננהולץ מהפקולטה להנדסת תעשייה וניהול

תודתי נתונה לפרופ' משה טננהולץ על תמיכתו ועל עזרתו, ועל כל שלימדני

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

תקציר

תהליכי החלטה מרקוביים הם מודלים מתמטיים של יחסי-הגומלין של סוכן עם סביבתו, ושל שאיפתו להשיג תועלת מאינטראקציה זו. הסביבה מתוארת על ידי מצב, המשתנה במהלך האינטראקציה. יתכן שמצב זה נצפה לגמרי ע"י הסוכן (fully observable), או לא נצפה כלל (unobservable). במקרים המורכבים יותר, הסוכן יכול לצפות חלקית במצב הסביבה, ואז נאמר שהתהליך **נצפה חלקית** (partially observable). בפעולותיו הסוכן משפיע על מצב הסביבה, ומקבל תגמול כתלות בהשפעה זו.

במקרים מסוימים הסוכן אינו יודע מלכתחילה את החוקים השולטים ביחסי-הגומלין, ומה עשויות להיות תוצאות החלטותיו. אם המודל ידוע מראש לסוכן, השגת תועלת מיר-בית מהתהליך היא בעית תכנון (planning). אחרת, על הסוכן ללמוד את המודל מתוך האינטראקציה, לפחות במידה שתאפשר לו להשיג תועלת קרובה למירבית. סוג למידה זה קרוי **למידת חיזוקים** (reinforcement learning).

בעבודה זו אנו דנים במספר היבטים של למידת חיזוקים בתהליכי החלטה מרקובי-ים שהנצפות בהם חלקית. אנו מציגים את מושג **סיבוכיות האינטראקציה** (interaction complexity), שהוא מידה לקצב בו אלגוריתם למידת חיזוקים מפיק מהסביבה מידע מוע-יל. בעוד סיבוכיות זמן הריצה של האלגוריתם מודדת את כמות הזמן הנדרשת לפעולתו, סיבוכיות האינטראקציה מודדת את כמות פעולות הגומלין הנדרשת ללמידה. סיבוכיות האינטראקציה היא מספר צעדי האינטראקציה המזערי שאלגוריתם למידה זקוק להם על מנת שהתועלת לסוכן תהיה קרובה למירבית. אנו מנתחים את סיבוכיות האינטראקציה של מספר מחלקות של תהליכי החלטה נצפים חלקית.

תהליכי החלטה מרקוביים נצפים חלקית (Partially Observable Markov Decision Pro-cesses, POMDPs) מהווים מחלקה חשובה של תהליכי החלטה, הנמצאת במוקד מחקר רב. ב- POMDPs מסוימים אין די (או אין כלל) יכולת לצפות במצב הסביבה, ולמידת חיזוקים כלל אינה אפשרית. אנו מראים תת-מחלקה של POMDPs שניתן ללמוד באמצעות חיזוקים, הדורשת באופן כללי סיבוכיות אינטראקציה מעריכית.

אנו דנים בתת-מחלקה של POMDPs, הנקראת OPOMDPs, בה נחקרו בעבר הן בעיות תכנון והן למידה. ב- OPOMDPs, צפיה במצב הסביבה אינה בלתי אפשרית, אך היא

עשויה להיות יקרה. לסוכן פעולה המגלה לו לחלוטין את המצב, אך עולה לו בהקטנת התועלת מהתהליך. אנו מראים כי למידת חיזוקים ב- OPOMDPs אפשרית תמיד בסי-בוכיות אינטראקציה פולינומיאלית. בעית למידה זו דומה במהותה ללמידה בתהליכי החל-טה מרקוביים נצפים לגמרי (Fully Observable Markov Decision Processes, MDPs), ופתרונה מורכב יותר אך במעט.

לאחר מכן אנו מציגים תת-מחלקה של POMDPs הדומה ל- OPOMDPs, אותה אנו מכנים OCOMDPs. תת-מחלקה זו תופסת את המורכבות של למידה בתהליכים נצפים חלקית יותר מאשר OPOMDPs, ולמידה יעילה בה קשה יותר לביצוע ולניתוח. ב- OCOMDPs, כמו ב-OPOMDPs, צפיה במצב הסביבה אפשרית באמצעות פעולת צפיה מיוחדת, העשויה להיות יקרה. אלגוריתם הלמידה נדרש לשקול היטב מתי לבצע פעולת צפיה זו, ולתאר כיצד להפיק ממנה די אינפורמציה על המודל.

הבדל אחד בין המודלים הוא שב- OCOMDPs, מלבד פעולת הצפיה המלאה, שאר הפעולות אינן מספקות כל צפיה על מצב הסביבה. הבדל נוסף הוא שב- OCOMDPs מצב הסביבה עשוי להשתנות בעקבות הפעולה. מתברר כי הבדל נוסף זה אחראי לעלית המדרגה במורכבות הבעיה. כעת לא ניתן לברר את מצב הסביבה בבואנו לבצע פעולת למידה, שכן המצב לאחר הבירור עשוי להיות בלתי רלבנטי למה שרצינו ללמוד. אנו נדרשים למצוא הזד-מנויות למידה גם כאשר מצב הסביבה אינו ידוע בוודאות, ועל כן כל פעולת למידה מורכבת יותר. כמו כן מורכבת יותר בעית הזיהוי של החלק מהמודל שכבר נלמד, כמו גם ניתוח קצב הלמידה.

התוצאה העיקרית של תזה זו היא הצגת אלגוריתם UR-MAX ללמידת חיזוקים ב-OCOMDPs, שהוא בעל סיבוכיות אינטראקציה פולינומיאלית. אנו מציגים את האלגוריתם ודנים בו ובאינטואיציה שמאחוריו, וכן מוכיחים את נכונותו ואת יעילותו.

Contents

List of Figures

תוכן עניינים

רשימת איורים

Abstract

Markov decision processes are mathematical models of an agent's interaction with its environment, and its attempt to obtain some benefit from this interaction. In the more complicated cases, the agent can only partially observe the state of the environment, and the process is called *partially observable*.

It may also be the case that the agent doesn't initially know the rules that govern the interaction, and what the results of its decisions may be. Then the agent has to learn the model from the interaction, a type of learning called *reinforcement learning*.

In this work we discuss several aspects of reinforcement learning in Markov decision processes in which observability is partial. We introduce the concept of *interaction complexity*, which is a measure of the rate in which reinforcement learning algorithms extract from the environment useful information. We then explore the interaction complexity of several classes of partially observable decision processes.

Partially Observable Markov Decision Processes (POMDPs) are a central and extensively studied class of decision processes. In some POMDPs, observability is too scarce, and reinforcement learning is not possible. We show a subclass of POMDPs which can be learned by reinforcement, but requires exponential interaction complexity.

We discuss a subclass of POMDPs, called OPOMDPs, in which both planning and learning have been previously studied. We show that reinforcement learning in OPOMDPs is always possible with polynomial interaction complexity.

We then introduce a closely related subclass of POMDPs which we call OCOMDPs. OCOMDPs capture more of the complexities of learning in POMDPs than OPOMDPs do, and efficiently learning them is considerably more difficult to perform and analyze than learning OPOMDPs.

The main result of this thesis is the UR-MAX algorithm for reinforcement learning in OCOMDPs with polynomial interaction complexity. We introduce and discuss the algorithm and some of the intuition behind it, and prove its correctness and efficiency.

1 Introduction

The widely applicable subject of decision making has attracted attention in many fields of research for many decades. Its normative aspect is the desire to make "good" choices. We identify an *agent*, which is the deciding entity, and an *environment*, which consists of everything else relevant to the decision problem. We may think of the environment as having some *state*, which is a complete description of these relevant features. The agent has some perception of that state, formed by an *observation* the agent makes of the state.

In its decisions, or *actions*, the agent may affect the state of the environment, as well as the nature of the observation. Some of these influences may be more desirable to the agent than others, and this serves as an incentive to the agent to prefer some decisions to others. This benefit, or *reward*, which the agent obtains, together with the observation, thus closes a loop of action and reaction between the agent and the environment (Figure 1).

If we wish to apply analytic tools to measure the quality of decisions and, hopefully, to automate the process of making good decisions, we can define the concepts above quantitatively, and look for algorithms which interact with the environment to obtain provably high rewards.

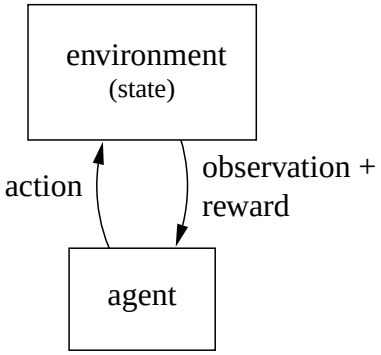

Figure 1: Scheme of interaction between agent and environment

Mathematical models of decision making are usually classified by the amount of information which the observation holds of the state. In *fully observable* models the entire state is observed. In *unobservable* models no useful information about the state is revealed by the observation. Generalizing both are *partially observable* models, in which the amount of information about the state provided by the observation can be anything between none and all.

The problem of making good decisions has another important aspect. Prior to the interaction, the agent may or may not know the model which will governs the environment – the way the actions affect the state, the way observations are made and so on. If the model is known beforehand, the question of making good choices becomes a calculation problem, and is named *planning*. If, however, the model is initially unknown, the agent has to interact with the environment, in order to learn enough of the model to eventually decide well. This *reinforcement learning* problem sets the challenge of extracting information from the interaction, in addition to the calculation problem of planning. Our focus in this work is addressing this challenge when observability is partial.

In chapter 2 below we provide formal definitions of the models which were briefly described here. In chapter 3 we define the problems of planning and learning and discuss some of their features. We also define the concept of *interaction complexity*, an important and useful measure for the efficiency of reinforcement learning algorithms, which has been implicit in other publications. In chapter 4 we briefly discuss previous research which relates to this work.

The last three chapters analyze the interaction complexity of reinforcement learning in partially observable decision processes. Chapter 5 includes a hardness result for POMDPs, the most general class of models we discuss. In chapter 6 we show how R-MAX, a previously published algorithm for efficient reinforcement learning in fully observable models, can be extended efficiently to some class of partially observable processes called OPOMDPs.

In chapter 7 we introduce another class of partially observable decision processes called OCOMDPs. Despite the similarity in definition to OPOMDPs, learning in this class is considerably more complicated, and demonstrates some of the difficulties of learning under partial observability. Our main result in this work is UR-MAX, an efficient algorithm for reinforcement learning in OCOMDPs.

2 Models

We present some well studied mathematical models of an environment with which a decision making agent is interacting. Time in these models advances discretely, and in each time step the agent performs an action of its choice. The state of the environment following this action depends stochastically only on the action and the previous state, which is known as the *Markov property*.

In *Markov Decision Processes (MDPs)*, the agent fully observes the state in each step before taking an action. This greatly simplifies the model, its solvability and learnability.

In *Partially Observable MDPs (POMDPs)*, the agent doesn't have direct access to the state. Rather, in each step it makes an observation which may reveal information about the state. This is a much more general and complex model.

Unobservable MDPs (UMDPs) take this handicap to the extreme, where no useful information about the state is available to the agent.

Following are definitions of these models.

2.1 Markov Decision Processes

Definition 1 (MDP)**.** A Markov Decision Process (MDP) is a tuple $M = \langle S, A, t, r \rangle$ where

- S is a finite non-empty set of possible environment states,
- A is a finite non-empty set of possible agent actions,
- $t : S \times A \to \Delta(S)$ is a distribution-valued state transition function, and
- $r : S \to [0, 1]$ is an agent's reward function.

The *interaction* of an agent with an MDP is a stochastic process $(s_i, a_i)_{i \geqslant 0}$, where the random variables s_i and a_i are distributed over the domains S and A, respectively.

Three elements induce this process. First, the initial state s_0 is distributed according to some initial state distribution $b_0 \in \Delta(S)$. This distribution is known to the agent, and arbitrary unless otherwise defined.

Second, the process is affected by the agent's choice of action in each step. We define $H = \bigcup_{i \geqslant 0} (S \times A)^i \times S$ to be the set of *observable histories*. Each time the agent is required to choose an action, there's a history $h \in H$ which describes the sequence of past events the agent has witnessed – the previous states and actions. The agent's *policy* is then a function $\pi : H \to \Delta(A)$, which determines, for every $i \geqslant 0$, the distribution of $a_i \sim \pi(h_i)$, where $h_i = \left((s_j, a_j)_{j=0}^{i-1}, s_i\right)$ is the observable history after i steps.

Third, the state transitions are determined by the model, with $s_{i+1} \sim t(s_i, a_i)$, for $i \geqslant 0$.

r is of no consequence for the process itself. It only becomes important when we use it to score policies in the following section. Its definition as a function from S to $[0, 1]$ may seem overly restrictive, but is in fact equivalent to any other common definition, as explained in the following section.

2.2 Scoring Policies

We are interested in making "good" decisions. The quality of a policy, for a given model and a given initial distribution, is determined by some quantity of the stochastic process, or more specifically, of the sequence of rewards $(r(s_i))_{i>0}$. Some commonly used expressions for this measure are:

- For a given finite *horizon* T:
  1. The expected average of the first T rewards, $\mathbf{E}\,\frac{1}{T}\sum_{i=1}^{T} r(s_i)$.
  2. The expected discounted sum, $\mathbf{E}\sum_{i=1}^{T}\gamma^i r(s_i)$, for some discount $\gamma \in (0,1)$.
- For infinite horizon:
  1. The infimum limit of the expected average, $\liminf_{T\to\infty} \mathbf{E}\,\frac{1}{T}\sum_{i=1}^{T} r(s_i)$.
  2. The expected discounted sum, $\mathbf{E}\sum_{i>0}\gamma^i r(s_i)$, for some $\gamma \in (0,1)$.

In the infinite horizon case, the second expression may make more sense than the first, in which the contribution of each reward tends to 0. In this work we are primarily concerned with the finite horizon case, and therefore use the simpler undiscounted expected average. This measure we call the *return* of the policy, and define it to be

$$U_M(b_0, \pi, T) = \mathbf{E}\,\frac{1}{T}\sum_{i=1}^{T} r(s_i)$$

for an initial state distribution b_0, a policy π and a finite horizon T, and

$$U_M(b_0, \pi) = \liminf_{T\to\infty} U_M(b_0, \pi, T)$$

for infinite horizon.

We conveniently use a rather specific definition of the reward function, $r : S \to [0,1]$. However, other functions can be represented in this formulation, in a way which preserves the meaning of the return as a quality measure. To be exact, let $M' = \langle S, A, t, r'\rangle$ be a model where the reward function r' that is not from S to $[0,1]$, and U' a return which will be defined appropriately below. We reduce M' to a model $\tilde{M}$ with rewards $\tilde{r} : S \to [0,1]$ such that the return $U_{\tilde{M}}$ is monotonic in ${U'}_{M'}$.

- Suppose that $r' : S \to \mathbb{R}$ is a reward function which is not restricted to $[0,1]$, and U' is the standard return formula.

  Since S is finite, r' is bounded, $r' : S \to [R_{\min}, R_{\max}]$, with $R_{\min} < R_{\max}$. Then we use $\tilde{r} = {}^{(r'-R_{\min})}\!/_{(R_{\max}-R_{\min})}$, which is restricted to $[0,1]$ while affecting U monotonically.

- Suppose that rewards are stochastic, $r' : S \to \Delta(\mathbb{R})$, and U' replaces each $r(s_i)$ in the standard formula with a random variable $r_i \sim r'(s_i)$.

  Since the return only depends on the expected reward, we take $\tilde{r} = \mathbf{E}\,r'$, normalized as in the previous item.

- Suppose that rewards depend on the actions, $r' : S \times A \to [0, 1]$. U' uses $r'(s_i, a_i)$ instead of $r(s_i)$, so that rewards are obtained for acting well, not for merely reaching good states. Also, the summation in U' should start from $i = 0$, instead of 1.

  The trick in this reduction is to define a new set of states, which remember not only the current state, but also the previous state and action. Formally, let $\tilde{M} = \left\langle \tilde{S}, A, \tilde{t}, \tilde{r} \right\rangle$ with $\tilde{S} = S^2 \times A$. The transition function $\tilde{t}$ remembers one previous state and action,

  $$\tilde{t}((s_{\text{curr}}, s_{\text{prev}}, a_{\text{prev}}), a_{\text{curr}}) = (t(s_{\text{curr}}, a_{\text{curr}}), s_{\text{curr}}, a_{\text{curr}}),$$

  and the reward for the reached state is actually the reward for the previous state and action

  $$\tilde{r}((s_{\text{curr}}, s_{\text{prev}}, a_{\text{prev}})) = r'(s_{\text{prev}}, a_{\text{prev}}).$$

  If we let the initial distribution be $\tilde{b}_0((s_0, \hat{s}, \hat{a})) = b_0(s_0)$ for some fixed $\hat{s} \in S$ and $\hat{a} \in A$, and $\tilde{\pi}$ be the policy in $\tilde{M}$ which corresponds to a policy π in M', then

  $$U_{\tilde{M}}(\tilde{b}_0, \pi, T) = \mathbf{E}\,\frac{1}{T}\sum_{i=1}^{T} \tilde{r}(\tilde{s}_i) =$$

  $$= \mathbf{E}\,\frac{1}{T}\sum_{i=1}^{T} r'(s_{i-1}, a_{i-1}) =$$

  $$= \mathbf{E}\,\frac{1}{T}\sum_{i=0}^{T-1} r'(s_i, a_i) = {U'}_{M'}(b_0, \pi, T).$$

2.3 Partially Observable and Unobservable Markov Decision Processes

Definition 2 (POMDP)**.** A Partially Observable Markov Decision Process (POMDP) is a tuple $M = \langle S, A, O, t, r, \theta \rangle$ where

- S, A, t and r are as in an MDP,
- O is a finite non-empty set of possible agent observations, and
- $\theta : S \times A \to O$ is an agent's observation function.

In POMDPs the agent doesn't observe the state at each step, and therefore can't base its actions on it. Instead, the agent's action includes some sensory power, which provides the agent with an observation of some features of the state. The set of observable histories here is $H = \bigcup_{i \geqslant 0} (A \times O)^i$, and the agent's policy is $\pi : H \to \Delta(A)$, such that $a_i \sim \pi(h_i)$, where

$$h_i = \left((a_j, \theta(s_j, a_j))_{j=0}^{i-1}\right)$$

is the sequence of past actions and observations. The return function U_M is defined here exactly as in MDPs.

In most publications, the observation depends on the state and the action stochastically. This has the interpretation that the value of the observed feature is generated "on demand" rather than being intrinsic to the state. But these classes of models are equivalent, in the sense that each can be reduced to the other. We can even restrict the model not to allow the observation to depend on the action, or otherwise extend it to allow the observation to depend on (or have joint distribution with) the following state, and equivalence is still preserved, as discussed in Appendix A.

An *Unobservable Markov Decision Process (UMDP)* is a POMDP in which there are no observations, or more precisely, observations provide no useful information. Formally, the easiest way to define a UMDP is as a POMDP in which $O = \{\perp\}$, so the constant observation $\perp$ provides no information of the state. Note that in both POMDPs and UMDPs, the rewards are generally not considered to be observable, and the agent can't use them in its decisions, unless they are explicitly revealed by θ.

2.4 POMDPs as MDPs over Belief-States

When observability is partial, the agent may not know the true state of the environment. In each step, the agent can calculate the probability distribution of the state, given the observable history, a distribution referred to as the agent's *belief* of what the state may be.

It's sometimes useful to view POMDPs as "MDPs" in which beliefs play the role of states. For a POMDP $M = \langle S, A, O, t, r, \theta \rangle$, consider the model $M' = \langle S', A, t', r' \rangle$. M' is like an MDP in every aspect, except that $S' = \Delta(S)$ is not finite. To define the transition function $t' : S' \times A \to \Delta(S')$, consider the belief $b \in S'$ which is the distribution of the state given some observable history h. Now suppose that a is taken, and that the observation is o, i.e. the next history is (h, a, o). Then the next belief is

$$b' = \frac{\sum_{s:\theta(s,a)=o} b(s)t(s,a)}{\sum_{s:\theta(s,a)=o} b(s)}.$$

$t'(b, a)$ is the distribution of b' (a distribution over distributions of states), and it receives the value above with probability

$$\Pr(o|h, a) = \sum_{s:\theta(s,a)=o} b(s).$$

Note that when the observation is constant, as in UMDPs, the belief-state transition is deterministic. This fact will prove very useful in chapter 7.

Using t', and given the initial state distribution (which is the initial belief-state in M'), we can translate observable histories into beliefs. Then policies in M translate into policies in M'. Since the return of a policy only involves the expected rewards, taking $r'(b) = \sum_s b(s)r(s)$ will give identical returns for a translated policy in M' as for the original policy in M.

3 Reinforcement Learning (RL)

3.1 Planning and Learning

Two objectives naturally arise when considering models of decision making. The first, *planning*, is the automated solution of the model.

Definition 3 (Finite Horizon Planning)**.** The finite horizon POMDP planning problem is to find an algorithm $\mathcal{A}$ which, given a model M, an initial state distribution b_0, and a horizon T, finds a policy $\pi_{\mathcal{A}} = \mathcal{A}(M, b_0, T)$ which, when used for T steps, maximizes the T-step return,

$$U_M(b_0, \pi_{\mathcal{A}}, T) = \max_{\pi} U_M(b_0, \pi, T).$$

Definition 4 (Infinite Horizon Planning)**.** The infinite horizon POMDP planning problem is to find an algorithm $\mathcal{A}$ which, given a model M, an initial state distribution b_0 and a margin $\epsilon > 0$, finds a policy $\pi_{\mathcal{A}} = \mathcal{A}(M, b_0, \epsilon)$ which, when used indefinitely, approaches the supremum return,

$$U_M(b_0, \pi_{\mathcal{A}}) \geqslant \sup_{\pi} U_M(b_0, \pi) - \epsilon.$$

The margin is required because generally a maximizing policy may not exist. As an example where no optimal policy exists, suppose that $S = \{s^0, s^1, s^{\text{good}}, s^{\text{bad}}\}$, $A = \{a^0, a^1, a^?\}$, and $O = \{o^0, o^1\}$ (Figure 2). We start in either s^0 or s^1 with equal probability. In these states, the actions a^0 and a^1 are an attempt to guess the state, and take us deterministically to s^{good} or s^{bad}, respectively if the guess is right (i.e. a^i is taken from s^i) or wrong (i.e. a^{1-i} is taken from s^i). The action $a^?$ stays in the same state s^0 or s^1, but provides an observation which helps determine the state – in state s^i the observation is o^i with probability $2/3$ and o^{1-i} with probability $1/3$. Once we take a guess, though, and reach s^{good} or s^{bad}, we always stay in the same state. The rewards are $r(s^0) = r(s^1) = 0$, $r(s^{\text{bad}}) = 1/2$ and $r(s^{\text{good}}) = 1$. Since the process continues indefinitely, making the right guess makes the difference between approaching an average reward of 1 or $1/2$, so any finite effort to improve the chances is justified. However, these chances improve indefinitely whenever $a^?$ is taken. Since there's no largest finite number of times to take $a^?$ before guessing, there's no optimal policy. Some related results can be found in [13].

It should also be noted that it's enough to focus on *deterministic* policies, i.e. ones in which the action is determined by the history, $\pi : H \to A$. Since the return in POMDPs is multilinear in the agent's policy, there always exists an optimal deterministic policy.

Matters become more complicated when the real model of the environment's dynamics is unknown, as is often the case in actual real-life problems. The second objective, *learning*, is the automated discovery of the model, to an extent which enables approaching the maximal return. Here the model isn't given as input to the algorithm. Instead, the agent must interact with the environment which implements the model, and extract from this interaction information which will allow it to devise a near-optimal policy. This type of learning is called *reinforcement learning*, and will be defined formally later. Note that it's possible for the model never to become fully known to the agent, but enough of it must be revealed in order to plan near-optimally.

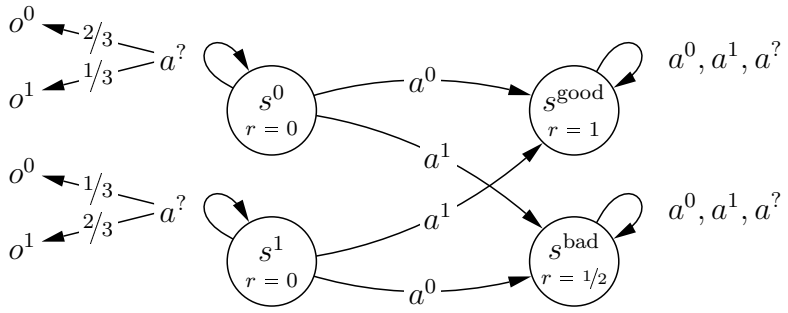

Figure 2: A POMDP with no optimal infinite horizon policy

3.2 Tradeoffs in Planning and Learning

In planning, a balance must be struck between short-term and long-term considerations of return maximization. This stems from the fact that the identity of the state affects simultaneously the reward, the observation and the following state. In the short term, rewards count directly towards the return. Obviously, a high return cannot be achieved without high rewards. In the long term, future rewards depend on future states, as some states may lead to better rewards than others. Therefore in choosing an action it's essential to consider not only the expected reward of the immediately next state, but also the possibility and probability of reaching future states with high expected rewards.

When observability is partial, the agent may not know the true state of the environment. We are still considering planning, and the model is completely known, but not the state at each step. The agent can calculate its belief, the probability distribution of the state given the observable history. This distribution is the best one for the agent to use in its future planning, and generally speaking, the less uncertainty this belief contains – the better expected reward the agent can achieve, as the example of Figure 2 shows. Hence long term planning also involves the observations and the certainty they give of the identity of states. The agent must therefore balance all three aspects of the effects its actions have: the magnitude of rewards obtained, the potential created to reach better future states, and the amount of certainty gained, in a three-way tradeoff.

In learning, these considerations are valid as well. Initially, however, the model in unknown, and a balancing policy cannot be calculated. As information regarding the model is collected, another kind of balance must be reached, that between *exploration* and *exploitation*. On the one hand, in order to improve the return, the agent must choose actions which it has tried before and found to effectively balance the short-term and the long-term rewards. On the other hand, poorly tested actions may turn out to be even more effective, and must be checked. The tradeoff between exploiting known parts of the model and exploring unknown parts is an important aspect of reinforcement learning.

3.3 Learning and Complexity

In all but trivial cases, at least some exploration must precede exploitation. The unavoidable delay in implementing a near-optimal policy has significant consequences on the kind of solutions to the learning problem we may expect to exist.

First, by the time knowledged planning can take place, the agent's belief may irrecoverably become arbitrarily worse than the initial distribution. In order to reveal the effects of actions, the agent must try every action. In fact, it has to try them many times, in order to estimate the transition distribution in possibly many states. This exploration process may put the environment through a variety of states, and the agent through a variety of beliefs. Therefore, a learning agent cannot be expected to maximize a return which depends on the initial belief. Many publications assume some conditions of connectedness under which the initial belief may always be returned to (for example, see [5]), but we shall not restrict our models as much. The agent's objective in learning will therefore be to maximize a version of the return function which is independent of the initial belief. The optimal T-step return will be

$$U_M^*(T) = \min_{\tilde{b}_0} \max_{\tilde{\pi}} U_M(\tilde{b}_0, \tilde{\pi}, T),$$

i.e. the return of the best policy from the worse initial belief, when the policy is aware of the belief. Similarly, the optimal infinite horizon return will be

$$U_M^*(\infty) = \min_{\tilde{b}_0} \sup_{\tilde{\pi}} U_M(\tilde{b}_0, \tilde{\pi}).$$

Second, the unavoidable delay in producing a good plan may prevent learning in the same time frame as planning. The learning agent must be given more time to interact with the environment.

Definition 5 (Finite Horizon Learning)**.** The finite horizon POMDP learning problem is to find an algorithm $\mathcal{A}$ which, given a horizon T and a margin $\epsilon > 0$, interacts with the environment for some $T_{\mathcal{A}}$ steps or more using a policy $\pi_{\mathcal{A}} = \mathcal{A}(T, \epsilon)$, to gain a return which approaches the optimal T-step return, i.e.

$$\forall T' \geqslant T_{\mathcal{A}} \quad U_M(b_0, \pi_{\mathcal{A}}, T') \geqslant U_M^*(T) - \epsilon,$$

where M is the actual model of the environment and b_0 is the actual initial distribution.

Definition 6 (Interaction Complexity)**.** For a reinforcement learning algorithm $\mathcal{A}$, the minimal number $T_{\mathcal{A}}$ of interaction steps after which near-optimal return is achieved for any M and b_0 is called the *interaction complexity* of $\mathcal{A}$.

$T_{\mathcal{A}}$ may be well over T, and keeping it small is an important aspect of the learning problem. For example, it's interesting to ask whether obtaining near-optimal return is possible in

$$T_{\mathcal{A}} = \text{poly}\left(T, \tfrac{1}{\epsilon}, |M|\right)$$

steps, i.e. polynomial interaction complexity. Here $|M|$ is the size of M's representation, which is polynomial in $|S|$, $|A|$ and $|O|$. This notion of complexity doesn't replace, but adds to the classic notion of computation complexity, expressed by $\mathcal{A}$'s running time and space requirements.

It's clear that for POMDPs such a reinforcement learning algorithm doesn't always exist, even if complexity is not bounded. For example, if the environment is a UMDP, the "learning" algorithm is equivalent to a predetermined sequence of actions, since no useful information is gained during interaction. Clearly, for each such sequence there exists a UMDP in which the optimal return is 1, but the return obtained by that particular sequence is arbitrarily close to 0.

3.4 Learning in Infinite Horizon

Definition 7 (Infinite Horizon Learning)**.** The infinite horizon POMDP learning problem is to find a policy $\pi_{\mathcal{A}}$ which, when used indefinitely, yields a return which converges at least to the optimal overall return, i.e. to have

$$U_M(b_0, \pi_{\mathcal{A}}) \geqslant U_M^*(\infty),$$

where M is the actual model of the environment and b_0 the actual initial distribution.

The achieved return can be better than "optimal", because $U_M^*(\infty)$ refers to the worst initial distribution.

The convergence rate of $\pi_{\mathcal{A}}$'s return to $U_M^*(\infty)$ is usually described in terms of a measure called *return mixing time*.

Definition 8 (Return Mixing Time)**.** The *ϵ-return mixing time* of a policy π, for an initial distribution b_0, denoted $T_{\text{mix}}(b_0, \pi, \epsilon)$, is the minimal horizon such that

$$\forall T' \geqslant T_{\text{mix}}(b_0, \pi, \epsilon) \quad U_M(b_0, \pi, T') \geqslant U_M(b_0, \pi) - \epsilon,$$

i.e. the time it takes the policy to approach its asymptotic return.

It's possible for a near-optimal policy to have much lower return mixing times than any strictly optimal policy. When the convergence rate of $\pi_{\mathcal{A}}$ matters, it may be beneficial to compare it not to an optimal policy, but to the best policy with a given mixing time. So we define

$$U_M^{*\text{mix}}(\epsilon, T) = \min_{\tilde{b}_0} \sup_{\tilde{\pi}: T_{\text{mix}}(\tilde{b}_0, \tilde{\pi}, \epsilon) \leqslant T} U_M(\tilde{b}_0, \tilde{\pi})$$

to be the optimal infinite horizon return, from the worst initial distribution, among policies whose ϵ-return mixing time for that distribution is at most T.

It's interesting to ask whether it's possible for $\pi_{\mathcal{A}}$ to approach $U_M^{*\text{mix}}(\epsilon, T)$, for every $\epsilon > 0$ and T, within a number of steps which is polynomial in T. Formally, we would like, for every $\epsilon, \epsilon' > 0$ and T, to have

$$T_{\mathcal{A}} = \text{poly}\left(T, \tfrac{1}{\epsilon'}, |M|\right),$$

such that

$$\forall T' \geqslant T_{\mathcal{A}} \quad U_M(b_0, \pi_{\mathcal{A}}, T') \geqslant U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon', \qquad (*)$$

where b_0 is the actual initial distribution. Then we can say that the optimal policy $\pi_{\mathcal{A}}$ converges efficiently.

Almost all relevant publications aim to solve the infinite horizon learning problem, rather than the finite one. This is mostly due to historical reasons, as earlier research was primarily concerned with whether convergence to optimal return was at all possible. More recent works (see [11] and [4]) define the return mixing time in order to analyze the convergence rate. However, these papers actually reduce the problem to its finite horizon form and solve it in that form. As it turns out, the finite horizon learning problem is both easier to work with and at least as general as its infinite counterpart.

Suppose that $\mathcal{A}'$ is an algorithm which efficiently solves the finite horizon learning problem. Recall that $\mathcal{A}'$ receives two arguments, a horizon and an error margin. We show that the policy $\pi_{\mathcal{A}}$ which sequentially runs $\mathcal{A}'(1,1), \mathcal{A}'(2,1/2), \mathcal{A}'(3,1/3), \ldots, \mathcal{A}'(x,1/x), \ldots$ indefinitely, efficiently solves the infinite horizon learning problem.

Fix some $\epsilon, \epsilon' > 0$ and T. We need to find $T_{\mathcal{A}}$ such that the requirement $(*)$ is satisfied. Let b_{x-1} be the actual belief when $\mathcal{A}'(x,1/x)$ starts, and let π_x and $T_x \geqslant x$ be the policy and the number of steps, respectively, used by $\mathcal{A}'(x,1/x)$. Take $x_0 = \max(T, \lceil 2/\epsilon' \rceil)$.

Now suppose that $\pi_{\mathcal{A}}$ is used for T' steps, which include exactly x whole runs of $\mathcal{A}'$, i.e. $\sum_{y=1}^{x} T_y \leqslant T' < \sum_{y=1}^{x+1} T_y$. Then

$$U_M(b_0, \pi_{\mathcal{A}}, T') \geqslant \mathbf{E}_{b_1,\ldots,b_{x-1}} \frac{1}{T'} \sum_{y=1}^{x} T_y U_M(b_{y-1}, \pi_y, T_y) \geqslant$$

(by $\mathcal{A}'$'s near-optimality)

$$\geqslant \frac{1}{T'} \sum_{y=1}^{x} T_y \max(0, U_M^*(y) - 1/y) \geqslant$$

$$\geqslant \frac{1}{T'} \sum_{y=x_0}^{x} T_y \max\left(0, \min_{\tilde{b}_0} \max_{\tilde{\pi}} U_M(\tilde{b}_0, \tilde{\pi}, y) - 1/y\right) \geqslant$$

$$\geqslant \frac{1}{T'} \sum_{y=x_0}^{x} T_y \max\left(0, \min_{\tilde{b}_0} \max_{\tilde{\pi}: T_{\text{mix}}(\tilde{\pi}, \tilde{b}_0, \epsilon) \leqslant T} U_M(\tilde{b}_0, \tilde{\pi}, y) - 1/y\right) \geqslant$$

$(y \geqslant x_0 \geqslant T \geqslant T_{\text{mix}}(\tilde{\pi}, \tilde{b}_0, \epsilon))$

$$\geqslant \frac{1}{T'} \sum_{y=x_0}^{x} T_y \max\left(0, \min_{\tilde{b}_0} \sup_{\tilde{\pi}: T_{\text{mix}}(\tilde{\pi}, \tilde{b}_0, \epsilon) \leqslant T} U_M(\tilde{b}_0, \tilde{\pi}) - \epsilon - 1/y\right) \geqslant$$

$(1/y \leqslant 1/x_0 \leqslant \epsilon'/2)$

$$\geqslant \frac{1}{T'} \sum_{y=x_0}^{x} T_y \max\left(0, U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon'/2\right).$$

We assume that $\mathcal{A}'$ has polynomial interaction complexity, so without loss of generality $\sum_{y=1}^{x} T_y = c_1 x^{c_2}$ for $c_1 = \text{poly}\,|M|$ and constant $c_2 \geqslant 1$. For $x \geqslant \frac{2}{\epsilon'}(c_2 + x_0)$ we have

$$\frac{1}{T'} \sum_{y=x_0}^{x} T_y \geqslant \tfrac{c_1 x^{c_2} - c_1 x_0{}^{c_2}}{c_1 (x+1)^{c_2}} =$$

$$= \left(1 - \tfrac{1}{x+1}\right)^{c_2} - \left(\tfrac{x_0}{x+1}\right)^{c_2} \geqslant$$

$$\geqslant 1 - \tfrac{c_2}{x} - \tfrac{x_0}{x} \geqslant 1 - \epsilon'/2,$$

and

$$U_M(b_0, \pi_{\mathcal{A}}, T') \geqslant$$

$$\geqslant \frac{1}{T'} \sum_{y=x_0}^{x} T_y \max\left(0, U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon'/2\right) \geqslant$$

$$\geqslant (1 - \epsilon'/2) \max\left(0, U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon'/2\right) >$$

$$(U_M^{*\text{mix}}(\epsilon, T) \leqslant 1)$$

$$> U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon',$$

as required. To have such value of x it's enough to have

$$T_{\mathcal{A}} = c_1 \left\lceil \tfrac{2}{\epsilon'}(c_2 + x_0) \right\rceil^{c_2} =$$

$$= c_1 \left\lceil \tfrac{2}{\epsilon'}(c_2 + \max\left(T, \left\lceil \tfrac{2}{\epsilon'} \right\rceil\right)) \right\rceil^{c_2} =$$

$$= \text{poly}\left(T, \tfrac{1}{\epsilon'}, |M|\right).$$

To show that $\pi_{\mathcal{A}}$'s return also converges eventually to an infinite horizon return of at least $U_M^*(\infty)$, it's sufficient to show that $U_M^{*\text{mix}}(\epsilon, T)$ tends to $U_M^*(\infty)$ as ϵ tends to 0 and T to infinity, since then

$$U_M(b_0, \pi) \geqslant \lim_{\substack{\epsilon \to 0 \\ \epsilon' \to 0 \\ T \to \infty}} U_M^{*\text{mix}}(\epsilon, T) - \epsilon - \epsilon' = U_M^*(\infty).$$

Showing this is not straightforward, because although the return mixing time is always finite, it may not be uniformly bounded. This difficulty can be overcome by discretizing the distribution space $\Delta(S)$.

4 Related Work

Markov Decision Processes and the problem of planning optimally in them was introduced in 1957 by Bellman [3], and was further studied by Howard [9]. They were able to formulate a polynomial-time algorithm for planning in MDPs, using Bellman's novel method of *Dynamic Programming*.

Partially Observable Markov Decision Processes were later introduced as a natural extension of MDPs, with more realistic and much stronger expressiveness, but one in which optimal planning is a much harder problem. It's of course at least as hard as the problem of deciding if a policy exists which surpasses a given return, which is NP-Complete for UMDPs and PSPACE-Complete for general POMDPs ([14] and [7]). Focus has therefore shifted to finding algorithms, mainly heuristic ones, which calculate near-optimal solutions. A survey of these methods can be found in [1].

Reinforcement Learning was conceived as a tool for "implicitly programming" an agent to perform a task, by specifying what the agent should desire to achieve, rather than how it should achieve it. Although it dates back to the early days of Artificial Intelligence and the intimately related Optimal Control Theory, it started receiving increasing attention only in the 1980's and the 1990's ([15] and [12]).

Several near-optimal reinforcement learning methods were developed in those years, but it wasn't until 1998 that an efficient algorithm, E^3, for learning in MDPs was introduced [11]. In 2002 another algorithm, R-MAX, was introduced [4], which was both simpler and more general, extending to multi-agent decision processes (called *Stochastic Games*).

Learning in POMDPs is not always possible. In UMDPs, for example, even if we assume that the reward function r is known, no information about the transition function t is obtained from interaction, and there's no way to improve, in the worst case, on a random walk which chooses actions uniformly at random.

Even in a subclass of POMDPs in which learning is possible, a simple Karp reduction shows that it's computationally at least as hard as planning. Despite that, or perhaps because of that, Reinforcement Learning in POMDPs has been the focus of much research, and many algorithms have been published which achieve it under different conditions (for a survey see [10], and more recently [5]).

The concept of interaction complexity gives the problem a new dimension. These aforementioned algorithms for learning in POMDPs all have at least exponential interaction complexity, in addition to exponential computation complexity. The following chapter will show that this is unavoidable in general POMDPs.

However, later chapters will explore subclasses of the POMDP class in which learning with polynomial interaction complexity is possible, along with algorithms which do so. To the best of our knowledge, this is the first published result of this kind. Some of it we previously published in [6].

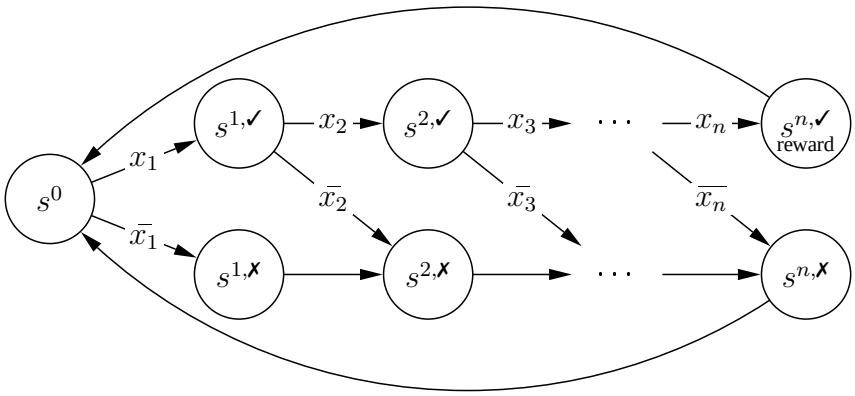

Figure 3: Password Guessing Problem

5 Hardness of RL in POMDPs

For a given reinforcement learning algorithm and a given model (unknown to the agent), for the algorithm to work, there must be a finite number of steps after which a near-optimal return is guaranteed. The interaction complexity of the algorithm is a worst case measure, taking the maximum of those guarantee-times over all models in a given class, if the maximum exists. The interaction complexity of a class of models can then be defined as the complexity of the best algorithm for learning it.

As explained in the previous chapter, there are POMDPs which no algorithm can learn, so the interaction complexity of POMDPs is undefined. But we can take a subclass of POMDPs in which learning is possible, such that its interaction complexity is at least exponential in the size of the model.

To see this, consider a Password Guessing Problem, which is the class of POMDPs of the form illustrated in Figure 3. The password is a fixed string $x_1 x_2 \dots x_n$ of n bits. The agent repeatedly tries to guess the password. If it succeeds, it receives a positive reward. All other rewards are 0, and the rewards are the only observations.

Planning in this model is easy, because knowing the model means knowing the password. The agent can then repeatedly guess the right password. Learning is possible, by guessing all passwords and observing which one yields a reward.

But this learning method takes more than $n2^n$ interaction steps in the worst case, while the number of states is $2n + 1$ and the number of actions and observations is constant. In fact, any learning algorithm will require exponential interaction complexity for this class, a result relied upon by nearly every security system in existence today.

To see this, consider the steps before a reward is first obtained. The observation is fixed in these steps, so the policy is equivalent to a predetermined sequence of actions, which is executed until the first reward. Clearly, for any such sequence there exists a model in which the password is not among the first $2^n - 1$ guesses.

6 Efficient RL in OPOMDPs

For our first positive result of reinforcement learning under partial observability, we consider a class of models in which full observability is available, but comes with a cost. This class is originally due to Hansen et al. [8]. It was studied recently in the context of approximate planning [2] under the name Oracular Partially Observable MDPs (OPOMDPs), which we adopt.

First, some useful notation. We define b_s to be the pure distribution with support s, i.e. $b_s(s) = 1$. We define I_P to be the indicator of predicate P, i.e. $I_P = 1$ if P is true and 0 if false.

In OPOMDPs there exists a special action α which allows the agent to observe the state of the environment. Although, technically, the agent spends a step taking α, this action is regarded as instantaneous, and the environment's state doesn't change while being observed.

Definition 9 (OPOMDP)**.** An Oracular Partially Observable Markov Decision Process (OPOMDP) is a tuple $M = \langle S, A, O, t, r, \theta, C \rangle$ where

- $\langle S, A, O, t, r, \theta \rangle$ is a POMDP,
- $S \subseteq O$,
- there exists some $\alpha \in A$, such that $\theta(s, \alpha) = s$ and $t(s, \alpha) = b_s$ for every $s \in S$, i.e. α observes the state without changing it, and
- action α incurs a special cost $C \geqslant 0$.

The agent's return is now given by

$$U_M(b_0, \pi, T) = \mathbf{E}\, \frac{1}{T} \sum_{i=1}^{T} \left(r(s_i) - C \cdot I_{[a_{i-1}=\alpha]} \right),$$

so that a cost C is deducted whenever α is taken.

This full-observability feature is invaluable for learning, but may be useless for planning, as can be seen in the following reduction from the learning problem of POMDPs to that of OPOMDPs. We simply add to the model an action α with the required properties but an inhibitive cost. If the original model is $M = \langle S, A, O, t, r, \theta \rangle$ and the horizon is T, we define $M' = \langle S, A', O', t', r, \theta', 2T \rangle$ with $A' = A \cup \{\alpha\}$, $O' = O \cup S$, where t' and θ' extend t and θ, respectively, as required. Taking α in M', even only once, reduces the T-step return by $C/T = 2$, to a negative value. Since the return in M is never negative, clearly an optimal policy in M' doesn't use α. This policy is then optimal in M as well.

Learning in OPOMDPs, on the other hand, is only slightly more complicated than learning in MDPs, if computation is not limited and interaction complexity is the only concern. We wish to focus on the differences, so instead of presenting an R-MAX-like algorithm for learning in OPOMDPs, we describe the important aspects of R-MAX, and explain how to manipulate it to handle OPOMDPs. The reader is referred to [4] for a more formal and complete presentation of R-MAX and a proof of its correctness and efficiency.

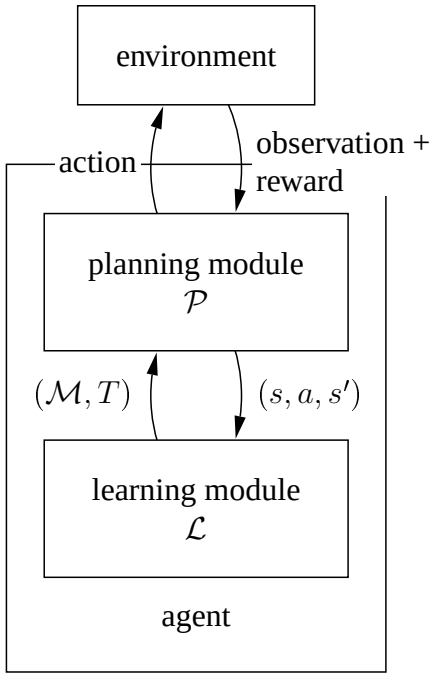

Figure 4: Scheme of a model-based RL algorithm

We assume that the reward function r is known to the agent. See section 7.2 for a discussion on how rewards can be learned.

R-MAX is a *model-based* reinforcement learning algorithm, which means we can identify in it a planning module and a learning module (Figure 4). The learning module $\mathcal{L}$ keeps track of a hypothetical model $\mathcal{M}$ of the environment. $\mathcal{M} = \langle \mathcal{S}, A, \tau, \rho \rangle$ is a standard MDP, except that for some of the pairs (s, a), the transition $\tau(s, a)$ is not guaranteed to resemble the real transition function t. These pairs are marked *unknown* in $\mathcal{M}$.

$\mathcal{L}$ sends to the planning module $\mathcal{P}$ planning commands of the form $(\mathcal{M}, T)$, meaning that $\mathcal{P}$ should calculate an optimal T-step policy in $\mathcal{M}$ and execute it. Occasionally, however, this is interrupted when $\mathcal{P}$ wishes to take an action a from a state s, where the transition for (s, a) is unknown in $\mathcal{M}$. In that case, $\mathcal{P}$ will take a, and upon reaching the next state s' will return to $\mathcal{L}$ the triplet (s, a, s'). From time to time, $\mathcal{L}$ will use these triplets to improve the model, all the while issuing more planning commands.

$\mathcal{P}$'s responsibility is to make policies which either yield near-optimal return or stumble upon an unknown pair (s, a). This is the *implicit explore or exploit* property of policies. The optimal policy in $\mathcal{M}$ has this property if $\mathcal{M}$ meets the following requirement:

- $\mathcal{M}$ correctly estimates the return of policies which have negligible probability to go through an unknown pair (s, a), and
- when $\mathcal{M}$ fails to estimate the return of a policy, it can only overestimate it.

This requirement is inspired by the principle of *optimism under uncertainty*.

Now if we can only guarantee that polynomial interaction is enough to rid $\mathcal{M}$ of all unknown pairs, the algorithm will reach near-optimality efficiently. This is what $\mathcal{L}$ does. For every unknown pair (s, a) which $\mathcal{P}$ encounters, it reports to $\mathcal{L}$ a sample (s, a, s') with $s' \sim t(s, a)$. Then $\mathcal{L}$ is required to update $\mathcal{M}$ in such a way that

- the optimism requirement mentioned above is kept, and
- there can never be more than polynomially many commands $(\mathcal{M}, T)$ and results (s, a, s') such that (s, a) is unknown in $\mathcal{M}$.

If all these requirements are satisfied, the agent will learn efficiently. A proof of this claim can be found in [4], together with a proof that R-MAX meets these requirements.

How can we apply such a scheme when observability is partial? Even if we think of a POMDP as an MDP over belief-states, there's still a basic difference between the two. In a POMDP, the environment doesn't tell the agent what the belief-state is, only what the observation is. We can't learn the model from samples of the form (b, a, b'), because we can't calculate b' before the relevant part of the model is learned.

In our solution, $\mathcal{L}$ won't learn belief transitions, but will learn state transitions just as it does in R-MAX. In fact, we'll make $\mathcal{L}$ completely oblivious to the fact that the model is no longer an MDP. We'll modify $\mathcal{P}$ slightly, to interact with $\mathcal{L}$ as if it's learning the state transition function t of an MDP.

It's interesting to observe that, if we "trick" $\mathcal{L}$ in this manner, almost everything else remains the same as it is in R-MAX. Partial observability limits the variety of policies available to $\mathcal{P}$, to those whose information of the state is that included in the observations, but the return of any fixed policy has nothing to do with θ, and is only defined in terms of t and r. This means that an optimistic hypothetical model still entails the implicit explore or exploit property. Note that partial observability does, however, make $\mathcal{P}$ computationally more complex.

One difference in learning that does exist, is that in R-MAX $\mathcal{P}$ can easily identify when (s, a) is unknown for the current state s. With POMDPs, we say that (b, a) is known if the probability is negligible that (s, a) is unknown, when s is distributed according to the belief b. $\mathcal{P}$ keeps track of an approximation of the current belief, and when it wishes to take an action a from a belief b such that (b, a) is unknown, it does the following. It takes α to reveal the current state $s \sim b$, then if (s, a) is unknown it takes a and then α again, to reveal the next state $s' \sim t(s, a)$. It passes to $\mathcal{L}$ the triplet (s, a, s'). Such a triplet is generated with non-negligible probability for every encounter of an unknown pair (b, a), so there will be at most a polynomial number of such encounters.

$\mathcal{P}$'s approximation of the current belief can degrade for two reasons. One is that the transition from the previous belief may be unknown. When it is known, there's the inaccuracy of the learned transition function, causing accumulated error in the approximation. In both cases, $\mathcal{P}$ will take α to reveal the state and make the belief pure and known again. The first type occurs at most a polynomial number of times, and the second is far enough between to have the cost of these α actions marginal.

7 Efficient RL in OCOMDPs

We now consider the case where α is not considered an oracle invocation, but an action inside the environment. Whereas in OPOMDPs α is guaranteed not to affect the state, now we won't expect the environment to stand still while being fully observed. Instead, α is allowed to affect the state, just like any action is. We call this class Costly Observable MDPs (COMDPs).

Definition 10 (COMDP)**.** A Costly Observable Markov Decision Process (COMDP) is a tuple $M = \langle S, A, O, t, r, \theta, C \rangle$ where

- $\langle S, A, O, t, r, \theta \rangle$ is a POMDP,
- $S \subseteq O$,
- there exists some $\alpha \in A$, such that $\theta(s, \alpha) = s$ for every $s \in S$, i.e. α observes the state, and
- action α incurs a special cost $C \geqslant 0$.

The return is defined as in OPOMDPs.

In this work we only discuss a subclass of COMDPs called Only Costly Observable MDPs (OCOMDPs). In OCOMDPs, α is the only observing action, and other actions provide no useful observability. That is, OCOMDPs extend UMDPs by adding α to the set of actions.

Definition 11 (OCOMDP)**.** An Only Costly Observable Markov Decision Process (OCOMDP) extending a UMDP $M = \langle S, A, \{\bot\}, t, r, \theta \rangle$ is a tuple $M' = \langle S, A', O, t', r, \theta', C \rangle$ where

- $A' = A \cup \{\alpha\}$,
- $O = S \cup \{\bot\}$,
- $t' : S \times A' \to \Delta(S)$ extends t,
- $\theta' : S \times A' \to O$ extends θ with $\theta'(s, \alpha) = s$, for every $s \in S$, and
- action α incurs a special cost $C \geqslant 0$.

The return is defined as in OPOMDPs.

COMDPs generalize OPOMDPs by removing the restriction that α doesn't change the state. OCOMDPs then specialize OCOMDPs by restricting the observability of other actions. It may be surprising that the apparently small generalization is enough to make learning in OCOMDPs considerably more complicated than in OPOMDPs. The additional difficulties will be discussed as we introduce UR-MAX, an efficient reinforcement learning algorithm for OCOMDPs.

It should be noted that, strictly speaking, the rewards in an OCOMDP are not bounded by $[0, 1]$, because α inflicts a negative reward of C. Rewards could be translated and scaled to follow our standard notation, but the analysis is cleaner if we keep this anomaly. Scaling would shrink ϵ by a factor of $C + 1$ too, so it's reasonable to allow the interaction complexity to be polynomial in C as well.

7.1 Learning with respect to the Underlying UMDP

7.1.1 Outline

For ease of exposition, we first learn with respect to the underlying UMDP. That is, the return actually achieved in an OCOMDP M' extending a UMDP M, given a horizon T and a margin $\epsilon > 0$, is required to be at least $U^*_M(T) - \epsilon$. For the moment we also assume that S, A, r and C are given as input, and that only t is initially missing.

In R-MAX [4] the transition t for unknown pairs (s, a) is learned when there are enough times in which s is reached and a taken from it. When we extended R-MAX to OPOMDPs in the previous chapter, we were able to do the same by taking α before a, reaching some pure belief b_s, and learning the effects of a on this s. However in OCOMDPs, since α changes the state, taking it before a may result in a belief irrelevant to anything we wish to learn.

But pure beliefs are not required for learning. If some mixed belief is reached enough times, we could learn the effects of some action on that belief. Suppose that the belief b is reached again and again indefinitely. Suppose that in enough of these cases we decide to take the action $a \neq \alpha$. If we take α after a, we observe the following state, which is distributed according to

$$b' = t(b, a) \stackrel{\text{def}}{=} \sum_s b(s)t(s, a).$$

The point here is that, since there are no observations, the belief b' which follows b when a is taken is determined only by b and a. Having repeated this strategy a large number of times, we have enough independent samples of b' to estimate it well.

But how can a belief b be reached many times? We have already established in section 2.4 that when the observation is constant the belief transition is deterministic. If we manage to reach some belief many times, and then take a fixed sequence of actions with constant observations, the belief at the end of that sequence is also the same every time. In addition, note that the belief after α is always $t(b_s, \alpha)$ for some observed $s \in S$, where in our notation $b_s(s) = 1$. So taking α enough times results in some (possibly unknown) belief appearing many times. We now combine these two facts into a systematic way to repeatedly reach the same beliefs.

UR-MAX, the algorithm we present here, operates in *phases*, which for convenience are of a fixed length T_1. Consider a T_1-step deterministic policy π, in which α is used as the first action but never again later. The initial state σ of a phase may be unknown when the phase begins, but since α is always the first action taken, σ is then observed. Given this initial state σ, the whole phase then advances deterministically:

- All the observations are determined by σ, since all are $\perp$ except the first, which is σ.
- It follows that all the actions are determined by σ, since π is deterministic.
- It then follows that all the beliefs in the phase are determined by σ.

Using the estimation scheme described above, each of these beliefs can eventually be estimated well if σ is the initial state of enough phases.

The policy π employed is decided on using a *hypothetical model* $\mathcal{M}$, which represents the part of the real model currently known. Since learning now involves beliefs instead of states, merely representing this model is considerably more complicated than in MDPs or in OPOMDPs, and an exact definition is kept for later. For now, we only give a general description.

The hypothetical model $\mathcal{M}$ is not a standard POMDP, but it is an MDP over belief-states. $\mathcal{M}$ keeps track of sets $\mathcal{B}_a \subseteq \Delta(S)$, for every $a \in A'$. $\mathcal{B}_a$ is the set of beliefs b for which the transition $\tau(b, a)$ is considered known, where τ is the transition function of $\mathcal{M}$. As part of the correctness of the algorithm, we'll show that, when $\mathcal{M}$ considers a transition known, it indeed approximates well the real transition t'.

As in R-MAX, $\mathcal{M}$ follows the principle of *optimism under uncertainty*, in that whenever the transition is unknown, it's assumed to result in an optimistic fictitious state $s^* \in \mathcal{S}$, where $\mathcal{S}$ is $\mathcal{M}$'s set of states (pure belief-states). When an action a is taken from a belief $b \notin \mathcal{B}_a$, $\mathcal{M}$ predicts that the state will become s^*. Reaching s^* in $\mathcal{M}$ actually means that $\mathcal{M}$ doesn't know the real belief reached in the real model M'. Once s^* is reached, $\mathcal{M}$ predicts a maximal reward for all future steps. This induces a bias towards exploring unknown parts of the model.

Some more explanation and definitions are in order before presenting the algorithm formally, but we are now ready to present a sketch, using the parameters T_1, the number of steps in a phase, and K, the number of samples of a distribution sufficient to estimate it well.

UR-MAX (sketch)

1. Initialize a completely unknown model $\mathcal{M}$, i.e. $\mathcal{B}_a = \varnothing$ for every $a \in A$
2. Let π be a deterministic T_1-step policy, which is optimal in $\mathcal{M}$ among those which use α in the first step and only then
3. Repeat for each T_1-step phase:
   - 3.1. "Forget" the history, set it to the initial history Λ
   - 3.2. For T_1 steps, but only while the current belief is known, use π
   - 3.3. If an unknown belief b is reached, i.e. s^* is reached in $\mathcal{M}$
     - Take α, and observe a state distributed according to b
   - 3.4. If there are now K samples of b
     - Update $\mathcal{M}$ (more on this below)
     - Find an optimal policy π in the new model $\mathcal{M}$

Note that the planning problem in line 3.4 may require exponential time to solve, but here we focus on the interaction complexity.

This sketch is missing some important details, such as how records are kept, when exactly a belief b is unknown, and how $\mathcal{M}$ is updated. Before we discuss these issues more formally and prove that UR-MAX works, here's some more intuition for why it works.

We can think of the initial α action of each phase as a reset action, which retroactively sets the initial belief to be the pure distribution b_σ, where σ is the initial state of the phase. As explained earlier, beliefs in the phase only depend on σ and on the actions taken, which means there's a finite (although possibly exponential) number of beliefs the agent can reach in T_1-step phases which start with α. Reaching a belief enough times allows the algorithm to sample and approximate it. Eventually, there will be phases in which all the beliefs are approximately known.

In this exposition, we assume that r is known. In phases where the beliefs calculated by $\mathcal{M}$ approximate the real beliefs in M', the returns promised by $\mathcal{M}$ also approximate the real returns. So if ever the return of some policy in $\mathcal{M}$ fails to approximate the real return of that policy, it's because some belief is not approximated well, i.e. some transition is not approximated well. As mentioned above, this only happens when an unknown real belief is reached, which is to say s^* is reached in $\mathcal{M}$. But then $\mathcal{M}$ promises maximal rewards, which are surely at least as high as the real rewards. This means that $\mathcal{M}$ can either estimate a return correctly or overestimate it, but never underestimate a policy's return by more than some margin.

Compare the policy π used by the algorithm with the optimal policy π^*. The return π^* provides in $\mathcal{M}$ can't be significantly lower than its real return, the optimal return. π maximizes the return in $\mathcal{M}$, so its return in $\mathcal{M}$ is also approximately at least as high as the optimal return. Finally, recall that, since we're discussing phases without unknown beliefs, the real return obtained from using π in the real model is approximated by $\mathcal{M}$, and can't be significantly lower than optimal.

$$\text{real return of } \pi \approx \text{return of } \pi \text{ in } \mathcal{M} \geqslant \text{return of } \pi^* \text{ in } \mathcal{M} \gtrsim \text{optimal return.}$$

This means the algorithm eventually achieves near-optimal return.

But UR-MAX promises more than that. It promises a near-optimal return after polynomially many steps. Naïvely, near-optimal phases will only be reached after sampling exponentially many beliefs. This is where we start filling in the blanks in the sketch above, and show how to use and update a hypothetical model in a way that requires only polynomial interaction complexity. We start with a formal definition of the hypothetical model $\mathcal{M}$.

7.1.2 Hypothetical Model

Definition 12 (PK-OCOMDP)**.** For an OCOMDP $M' = \langle S, A', O, t', r, \theta', C\rangle$, a Partially Known OCOMDP (PK-OCOMDP) is a tuple $\mathcal{M} = \langle \mathcal{S}, A', O, \tau, \rho, \theta', C, \mathcal{W}, \epsilon'\rangle$ where

- $\mathcal{S} = S \cup \{s^*\}$ contains a special learning state $s^* \notin S$,
- $\rho : \mathcal{S} \to [0, 1]$ extends r with $\rho(s^*) = 1$,
- $\tau : S \times A' \to \Delta(S)$ partially approximates t' (τ, $\mathcal{W}$ and ϵ' are explained later),
- $\mathcal{W} = \{\mathcal{W}_a\}_{a\in A'}$ is a collection of sets of beliefs, $\mathcal{W}_a \subseteq \Delta(S)$ for every $a \in A'$, which serve a base to calculate $\mathcal{B}_a$, the part of the model considered known, and
- $\epsilon' \geqslant 0$ is the tolerance of what is considered known.

We need a few more definitions in order to explain how $\mathcal{M}$ is used. This will also explain what τ, $\mathcal{W}$ and ϵ' mean in PK-OCOMDPs.

For every action $a \in A'$, the elements of $\mathcal{W}_a$ are beliefs b for which the effects of a, namely $t'(b,a)$, have been sampled by the algorithm. Recall that the belief which deterministically follows b when $a \neq \alpha$ is taken is defined as $t'(b,a) = \sum_s b(s)t'(s,a)$, and similarly for $a = \alpha$ if b is pure. This linearity of transition allows the agent to deduce the effects of a on beliefs which are linear combinations of elements of $\mathcal{W}_a$.

For every $a \in A'$, we define $Sp(\mathcal{W}_a)$ to be the minimal affine subspace of $\mathbb{R}^S$ which includes $\mathcal{W}_a$

$$Sp(\mathcal{W}_a) = \left\{ \sum_{w \in \mathcal{W}_a} c_w w \middle| \sum_{w \in \mathcal{W}_a} c_w = 1 \right\}.$$

In particular, $Sp(\varnothing) = \varnothing$. So for $a \in A'$ and $b = \sum_w c_w w \in Sp(\mathcal{W}_a)$, where $a \neq \alpha$ or b is pure, we have

$$t'(b,a) = \sum_s b(s)t'(s,a) = \sum_{s,w} c_w w(s)t'(s,a) = \sum_w c_w t'(w,a).$$

If $t'(w,a)$ are estimated by sampling them, an estimate for $t'(b,a)$ can be extrapolated from them.

However, $t'(w,a)$ can't be exactly revealed by the algorithm, except in degenerate cases. In order to bound the error in $t'(b,a)$, we must also bound the coefficients in these combinations. For every $a \in A'$ and $c > 0$, $Sp(\mathcal{W}_a, c) \subseteq Sp(\mathcal{W}_a)$ is the set of linear combinations of $\mathcal{W}_a$ with bounded coefficients

$$Sp(\mathcal{W}_a, c) = \left\{ \sum_w c_w w \in Sp(\mathcal{W}_a) \middle| \forall w \in \mathcal{W}_a \quad c_w \leqslant c \right\}.$$

Now we allow some tolerance as, for every $a \in A'$, we define $\mathcal{B}_a^{\mathcal{W},\delta}$ to be the set of beliefs which are δ-close to $Sp(\mathcal{W}_a, e)$, in the L_1 metric, where e is the base of the natural logarithm.

$$\mathcal{B}_a^{\mathcal{W},\delta} = \left\{ b \in \Delta(S) \middle| \exists b' \in Sp(\mathcal{W}_a, e) \quad \|b' - b\|_1 \leqslant \delta \right\},$$

where

$$\|b' - b\|_1 = \sum_{s \in S} |b'(s) - b(s)|.$$

e is not special here, and any constant larger than 1 would do, but e will be convenient later when the logarithm of this number will be used. We always use a single hypothetical model, so we will sometimes omit $\mathcal{W}$ from this notation.

In $\mathcal{M}$, belief-state transitions are defined similar to the belief-state transitions in OCOMDPs, except that an action which is unknown for the belief-state from which it is taken always leads deterministically to the special learning state $s^* \in \mathcal{S}$. We can finally define exactly what we mean by an *unknown transition*, and that's taking an action a from a belief $b \notin \mathcal{B}_a^{\epsilon'}$. Then the belief

which follows $\bar{b} \in \Delta(\mathcal{S})$ when $a \in A'$ is taken and o is observed is[1]

$$\tau(\bar{b}, a) = \begin{cases} \sum_{s \in S} \bar{b}(s)\tau(s, a), & \text{if } a \neq \alpha \text{ and } \bar{b} \in \mathcal{B}_a^{\epsilon'}; \\ \tau(o, \alpha), & \text{if } a = \alpha \text{ and } b_o \in \mathcal{B}_\alpha^{\epsilon'}; \\ b_{s^*}, & \text{otherwise.} \end{cases}$$

Note that $\tau(\bar{b}, a) = b_{s^*}$ whenever $\bar{b}(s^*) > 0$, i.e. once s^* is reached it's never left.

Defining the stochastic process $(s_i, a_i)_{i \geqslant 0}$ in $\mathcal{M}$ is a bit tricky, because it's just a hypothetical model, and it doesn't actually govern the real distributions of events. When $\mathcal{M}$ is used for planning, the process is defined by having, for every $i \geqslant 0$, $s_i \sim \bar{b}_i$, and $\bar{b}_{i+1} = \tau(\bar{b}_i, a_i)$, where the observation is $o_i = \theta'(s_i, a_i)$.

However when $\mathcal{M}$ is used to simulate the actual interaction with an OCOMDP M', s_i is not really distributed according to the belief in $\mathcal{M}$, but rather the belief in M'. This is a different stochastic process, which may be unknown to the agent, but is still well-defined.

If there's going to be any correlation between using $\mathcal{M}$ for planning and for simulating, we must define a way in which $\mathcal{M}$ must resemble the real model. The following definition asserts when a PK-OCOMDP is usable as an approximation of an OCOMDP.

Definition 13 (OCOMDP Approximation)**.** Let $M' = \langle S, A', O, t', r, \theta', C \rangle$ be an OCOMDP and $\mathcal{M} = \langle \mathcal{S}, A', O, \tau, \rho, \theta', C, \mathcal{W}, \epsilon' \rangle$ a PK-OCOMDP, such that $\mathcal{S} = S \cup \{s^*\}$ and ρ extends r. $\mathcal{M}$ is called a μ-approximation of M' if, for every $0 \leqslant \delta' \leqslant \epsilon'$, $a \in A'$ and $b \in \mathcal{B}_a^{\mathcal{W},\delta'} \subseteq \mathcal{B}_a^{\mathcal{W},\epsilon'}$

$$\left\| \sum_{s \in S} b(s)(\tau(s, a) - t'(s, a)) \right\|_1 \leqslant \mu + 2\delta'.$$

Note the involvement of the parameter δ', which demonstrates the linear degradation of the transition functions' proximity, as the belief becomes more distant from the span of learned beliefs.

7.1.3 Updating the Model

As new knowledge of the real model is gained, it is incorporated into the hypothetical model. Updating the model basically means adding a belief b into some $\mathcal{W}_a$. It's convenient for many parts of the algorithm and its analysis to have $\mathcal{W}_a$ linearly independent, i.e. not to have any of its members expressible, or even nearly expressible, by a linear combination of the other members. Therefore, when the newly added b is δ-close to a member of $Sp(\mathcal{W}_a)$, for some learning tolerance $\delta > 0$, some member of $\mathcal{W}_a$ is dropped from it, to maintain this strong form of independence. In any case, updating the model changes $\mathcal{B}_a$, and τ is also updated so that $\mathcal{M}$ remains an approximation of the real model.

Before presenting the Update procedure, we must address another important technical issue. Maintaining $\mathcal{M}$ as an approximation of the real model relies on having $\tau(w, a)$ approximate

[1] We slightly abuse the notation for the sake of clarity. The beliefs $b \in \Delta(S)$ and $b' \in \Delta(\mathcal{S})$, where $\forall s \in S \quad b(s) = b'(s)$, are interchangeable in our notation.

$t'(w,a)$. There are two sources for error in this approximation. One is the error in sampling the distribution after taking a, which can be made small by taking large enough samples. The other source is error in the estimation of w, which may be caused by the actual belief being different than the one calculated in $\mathcal{M}$. This error is the accumulation of approximation errors in previous steps of the phase. When the next belief is calculated, it incorporates both the error in τ and the error in the previous belief, so the latter may actually be counted twice. For this reason, if error is allowed to propagate, it could increase exponentially and prevent approximation. The solution is to have the belief w sampled again before using it in $\mathcal{W}_a$. In this way, its error is not affected by previous steps, and doesn't increase from step to step.

An invocation of the Update procedure with parameters $\mathrm{sample}_{\mathrm{before}}, \mathrm{sample}_{\mathrm{after}} \in \Delta(S)$ and $a \in A'$ is supposed to mean that $\mathrm{sample}_{\mathrm{before}}$ and $\mathrm{sample}_{\mathrm{after}}$ approximate some beliefs b and b', respectively (presumably because they are large samples of these distributions), such that $b' = t'(b,a)$. In the following procedure, δ is a parameter of learning tolerance and μ' a parameter of allowed approximation error. Update is only invoked for $\mathrm{sample}_{\mathrm{before}} \notin \mathcal{B}_a^{\sqrt{n}\delta}$, where $n = |S|$. In addition, we use the notation $f_{w,a}$, for every $w \in \mathcal{W}_a$, to refer to previously used samples, and so in each run we set $f_{\mathrm{sample}_{\mathrm{before}},a} = \mathrm{sample}_{\mathrm{after}}$.

Update($\mathbf{sample_{before}}$, a, $\mathbf{sample_{after}}$)

1. Set $f_{\mathrm{sample}_{\mathrm{before}},a} = \mathrm{sample}_{\mathrm{after}}$.

2. If $\mathcal{W}_a \neq \varnothing$, let $b' = \sum_{w \in \mathcal{W}_a} c_w w$ be $\mathrm{sample}_{\mathrm{before}}$'s projection on $Sp(\mathcal{W}_a)$.

3. If $\mathcal{W}_a = \varnothing$, or if $\|b' - \mathrm{sample}_{\mathrm{before}}\|_2 > \delta$, then let $\mathcal{W}'_a = \mathcal{W}_a \cup \{\mathrm{sample}_{\mathrm{before}}\}$.

4. Otherwise, there exists some $w' \in \mathcal{W}_a$ such that $c_{w'} > e$ (we prove this in Lemma 1), and let $\mathcal{W}'_a = \mathcal{W}_a \backslash \{w'\} \cup \{\mathrm{sample}_{\mathrm{before}}\}$.

5. Find functions $\tau'_s : S \to \mathbb{R}$, for every $s \in S$, which satisfy all of the following linear constraints:

   - For every $s \in S$, $\tau'_s \in \Delta(S)$ is a probability distribution function.
   - For every $w \in \mathcal{W}'_a$, $\|\sum_s w(s)\tau'_s - f_{w,a}\|_1 \leqslant 2\mu'$.

6. If a solution exists:

   - Set $\mathcal{W}_a$ to $\mathcal{W}'_a$.
   - For every $s \in S$, set $\tau(s,a)$ to τ'_s.

7.1.4 Analysis of the Update Procedure

In this section we prove the correctness and efficiency of the Update procedure, under some conditions. This analysis has 3 parts:

- Lemma 1 relates the number of invocations of the Update procedure to a measure of how much $\mathcal{M}$ knows of the real model.
- Corollary 2 bounds this measure, and therefore also shows that the Update procedure can only be invoked polynomially many times.
- Lemma 3 then proves that these updates produce good approximations of the real model.

We need to make several assumptions at this point, one is that the samples are good representations of the beliefs they sample.

Definition 14 (Sample Compatibility)**.** A triplet $(\mathrm{sample}_{\mathrm{before}}, a, \mathrm{sample}_{\mathrm{after}})$ is said to be *compatible* if

$$\left\| \mathrm{sample}_{\mathrm{after}} - \sum_s \mathrm{sample}_{\mathrm{before}}(s) t'(s,a) \right\|_1 \leqslant 2\mu'.$$

If $a = \alpha$, it's also required that $\mathrm{sample}_{\mathrm{before}}$ is a pure distribution.

Compatibility means that $\mathrm{sample}_{\mathrm{before}}$ and $\mathrm{sample}_{\mathrm{after}}$ could reasonably be samples of some beliefs b and b', respectively, before and after taking action a. When we assume *compatibility*, we mean that all invocations of the Update procedure have compatible arguments.

In addition, since we didn't yet provide the precise context of the Update procedure, we assume for now that $\mathcal{M}$ is initialized empty (i.e. $\mathcal{W}_a = \varnothing$ for every $a \in A'$), that $\mathcal{M}$ is only changed by the Update procedure, and that, as mentioned before, the Update procedure is only invoked when $\mathrm{sample}_{\mathrm{before}} \notin \mathcal{B}_a^{\mathcal{W}, \sqrt{n}\delta}$, where $n = |S|$.

For an action $a \in A'$ such that $\mathcal{W}_a \neq \varnothing$, let $d(\mathcal{W}_a) = \dim Sp(\mathcal{W}_a)$ be the dimension of $Sp(\mathcal{W}_a)$. Let $H(\mathcal{W}_a) \subseteq Sp(\mathcal{W}_a)$ be the convex hull of $\mathcal{W}_a$, that is, the set of linear combinations of $\mathcal{W}_a$ with non-negative coefficients which add up to 1. Let $v(\mathcal{W}_a)$ be the $d(\mathcal{W}_a)$-dimensional volume of $H(\mathcal{W}_a)$, that is, the measure of $H(\mathcal{W}_a)$ in $Sp(\mathcal{W}_a)$. For $\delta > 0$, we define

$$F_\delta(\mathcal{W}_a) = d(\mathcal{W}_a) + \ln \frac{d(\mathcal{W}_a)! v(\mathcal{W}_a)}{\delta^{d(\mathcal{W}_a)}} + 1.$$

For convenience, we define $d(\varnothing) = -1$ and $F_\delta(\varnothing) = 0$.

$F_\delta(\mathcal{W}_a)$ is a measure of how much is known for a, which combines the dimension of $\mathcal{W}_a$ and the volume of its convex hull. Following the two cases in lines 3 and 4 of the Update procedure, if $\mathrm{sample}_{\mathrm{before}}$ is not within δ of being spanned by $\mathcal{W}_a$, then including it in $\mathcal{W}_a$ will increase the dimension of its convex hull. Otherwise, replacing some w' for $\mathrm{sample}_{\mathrm{before}}$ will increase the volume significantly, which is where δ becomes important. In either case, $F_\delta(\mathcal{W}_a)$ will increase by at least 1, as shown in the following lemma.

Lemma 1 (Update Effectiveness)**.** *Assuming compatibility, after D invocations of the Update procedure $d(\mathcal{W}_a) = |\mathcal{W}_a| - 1$ (i.e. $\mathcal{W}_a$ is linearly independent) for every $a \in A'$, and $\sum_{a \in A'} F_\delta(\mathcal{W}_a) \geqslant D$.*

Proof. The proof is by induction on D. For $D = 0$, $d(\varnothing) = |\varnothing| - 1$, and $\sum_a F_\delta(\varnothing) = 0$.

Now we assume that the lemma holds for some $D - 1$, and prove it for D. Let the Dth invocation be with parameters $\mathrm{sample}_{\mathrm{before}}$, a and $\mathrm{sample}_{\mathrm{after}}$. During the invocation, $\mathcal{W}_{a'}$ remains the same for every $a' \neq a$. We shall show that, assuming compatibility, $\mathcal{W}'_a$ is well-defined and satisfies $d(\mathcal{W}'_a) = |\mathcal{W}'_a| - 1$ and $F_\delta(\mathcal{W}'_a) \geqslant F_\delta(\mathcal{W}_a) + 1$. We shall further show that, assuming compatibility, the linear constraint satisfaction problem of the Update procedure has a solution, so that the update actually takes place.

b' is the projection of $\mathrm{sample}_{\mathrm{before}}$ on $Sp(\mathcal{W}_a)$, if one exists. Suppose first that $\mathcal{W}_a = \varnothing$, or otherwise that $\|b' - \mathrm{sample}_{\mathrm{before}}\|_2 > \delta > 0$ (Figure 5). Then $\mathrm{sample}_{\mathrm{before}} \notin Sp(\mathcal{W}_a)$, and so

$$d(\mathcal{W}'_a) = d(\mathcal{W}_a \cup \{\mathrm{sample}_{\mathrm{before}}\}) = d(\mathcal{W}_a) + 1 = |\mathcal{W}_a| = |\mathcal{W}'_a| - 1.$$

If $\mathcal{W}_a = \varnothing$, then $\mathcal{W}'_a = \{\mathrm{sample}_{\mathrm{before}}\}$, and

$$F_\delta(\mathcal{W}'_a) = 0 + \ln 1 + 1 = F_\delta(\mathcal{W}_a) + 1.$$

If $\mathcal{W}_a \neq \varnothing$, then $H(\mathcal{W}'_a)$ is a $d(\mathcal{W}'_a)$-dimensional hyper-pyramid, whose head is $\mathrm{sample}_{\mathrm{before}}$, whose base is $H(\mathcal{W}_a)$, and whose height is $\|b' - \mathrm{sample}_{\mathrm{before}}\|_2 > \delta$. The formula for the volume of the pyramid is

$$v(\mathcal{W}'_a) = \frac{\|b' - \mathrm{sample}_{\mathrm{before}}\|_2 \, v(\mathcal{W}_a)}{d(\mathcal{W}'_a)},$$

and

$$F_\delta(\mathcal{W}'_a) = d(\mathcal{W}'_a) + \ln \frac{d(\mathcal{W}'_a)! v(\mathcal{W}'_a)}{\delta^{d(\mathcal{W}'_a)}} + 1 >$$

$$> d(\mathcal{W}'_a) + \ln \frac{d(\mathcal{W}'_a)! \cdot \delta v(\mathcal{W}_a)}{\delta^{d(\mathcal{W}'_a)} \cdot d(\mathcal{W}'_a)} + 1 =$$

$$= d(\mathcal{W}_a) + 1 + \ln \frac{d(\mathcal{W}_a)! v(\mathcal{W}_a)}{\delta^{d(\mathcal{W}_a)}} + 1 = F_\delta(\mathcal{W}_a) + 1.$$

The remaining case (Figure 6) is when $\mathcal{W}_a \neq \varnothing$ and

$$\sqrt{n}\delta \geqslant \sqrt{n} \, \|b' - \mathrm{sample}_{\mathrm{before}}\|_2 \geqslant \|b' - \mathrm{sample}_{\mathrm{before}}\|_1 .$$

Recall that we assume that $\mathrm{sample}_{\mathrm{before}} \notin \mathcal{B}_a^{\mathcal{W}, \sqrt{n}\delta}$, so that

$$b' = \sum_w c_w w \in Sp(\mathcal{W}_a) \backslash Sp(\mathcal{W}_a, e),$$

and there must be some $w' \in \mathcal{W}_a$ such that $c_{w'} > e$. In this case, $\mathcal{W}'_a = \mathcal{W}_a \backslash \{w'\} \cup \{\mathrm{sample}_{\mathrm{before}}\}$. Under the assumptions, $\mathcal{W}'_a$ is well-defined in the Update procedure.

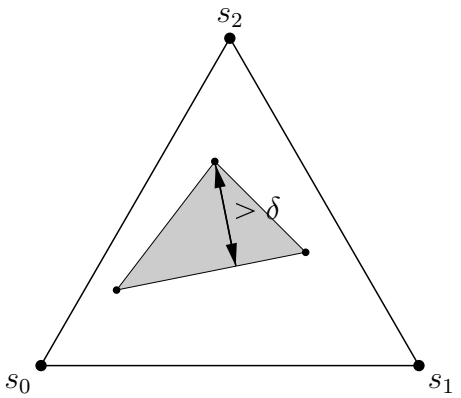

Figure 5: First case of the Update procedure (line 3).
The topmost belief is not spanned by the other two,
and adding it to $\mathcal{W}_a$ increases $d(\mathcal{W}_a)$.

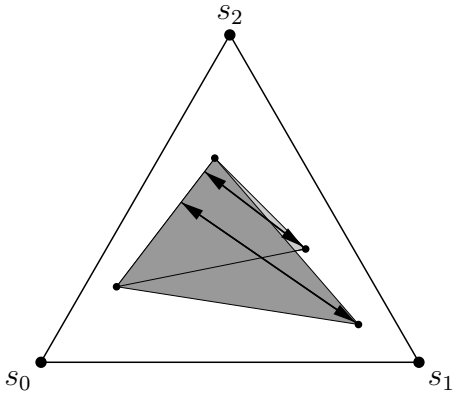

Figure 6: Second case of the Update procedure (line 4).
The rightmost belief is spanned by the other three,
but one coefficient is too large (for illustration, we use > 1.5 instead of $> e$);
adding the rightmost belief to $\mathcal{W}_a$ instead of the one with large coefficient
multiplies $v(\mathcal{W}_a)$ by the ratio of heights.

Let b'' and w'' be the projections of b' and w', respectively, on $Sp(\mathcal{W}_a \backslash \{w'\})$. b'' and w'' exist, since $\sum_w c_w = 1$ and $c_{w'} > e$ imply $|\mathcal{W}_a| > 1$. Observe that

$$b'' - b' = c_{w'}(w'' - w'),$$

and that b'' is also the projection of $\mathrm{sample}_{\mathrm{before}}$ on $Sp(\mathcal{W}_a \backslash \{w'\})$. Then

$$\|b'' - \mathrm{sample}_{\mathrm{before}}\|_2 = \sqrt{\|b'' - b'\|_2^2 + \|b' - \mathrm{sample}_{\mathrm{before}}\|_2^2} \geqslant$$

$$\geqslant \|b'' - b'\|_2 = c_{w'} \|w'' - w'\|_2 > e \|w'' - w'\|_2 > 0.$$

By the induction assumption, $\mathcal{W}_a$ is linearly independent, and therefore

$$d(\mathcal{W}'_a) = d(\mathcal{W}_a \backslash \{w'\}) + 1 = d(\mathcal{W}_a) = |\mathcal{W}_a| - 1 = |\mathcal{W}'_a| - 1,$$

and

$$v(\mathcal{W}'_a) = \frac{\|b'' - \mathrm{sample}_{\mathrm{before}}\|_2 \, v(\mathcal{W}_a \backslash \{w'\})}{d(\mathcal{W}'_a)} >$$

$$> \frac{e \|w'' - w'\|_2 \, v(\mathcal{W}_a \backslash \{w'\})}{d(\mathcal{W}_a)} = e v(\mathcal{W}_a),$$

which gives

$$F_\delta(\mathcal{W}'_a) = d(\mathcal{W}'_a) + \ln \frac{d(\mathcal{W}'_a)! v(\mathcal{W}'_a)}{\delta^{d(\mathcal{W}'_a)}} + 1 >$$

$$> d(\mathcal{W}_a) + \ln \frac{d(\mathcal{W}_a)! \cdot e v(\mathcal{W}_a)}{\delta^{d(\mathcal{W}_a)}} + 1 = F_\delta(\mathcal{W}_a) + 1.$$

We now turn to handle the constraint satisfaction problem. Consider the "correct" solution in which, for every $s \in S$, $\tau'_s = t'(s, a)$. Then of course τ'_s is indeed a probability distribution over S. As for the second group of constraints, for every $w \in \mathcal{W}'_a$ there had to be some invocation of the Update procedure in which $\mathrm{sample}_{\mathrm{before}}$ was that w and $\mathrm{sample}_{\mathrm{after}}$ was $f_{w,a}$. The constraints are then exactly equivalent to the compatibility condition, which we assume to hold. □

In the following, $|S| = n$ and $|A'| = k$.

Corollary 2 (Update Invocations)**.** *Assuming compatibility, less than* $2nk \ln \frac{1}{\delta}$ *invocations of the Update procedure are ever made.*

Proof. Since $\mathcal{W}_a \subseteq \Delta(S)$ for every $a \in A'$, we have $d(\mathcal{W}_a) < n$. The convex hull $H(\mathcal{W}_a)$ is bounded by the belief space. The largest convex hull of $|\mathcal{W}_a|$ linearly independent points bounded by a simplex is a $d(\mathcal{W}_a)$-face of the simplex.

In order to see that consider, among the sets $\mathcal{W}_a$ with maximal $v(\mathcal{W}_a)$, one which contains the most pure beliefs, i.e. vertices of the belief space. Assume in contradiction that some belief is not pure. If we fix the others, then the volume, as a function of the position of that last point, is the absolute value of a linear function, and is therefore maximal in a vertex of the simplex, i.e. in a pure belief, in contradiction.

A $d(\mathcal{W}_a)$-face of the belief space is itself a simplex with $\sqrt{2}$-length edges. So

$$v(\mathcal{W}_a) \leqslant \frac{\sqrt{d(\mathcal{W}_a)+1}}{d(\mathcal{W}_a)!},$$

which is the volume of that polytope. Therefore, for $0 < \delta \leqslant e^{-2}$,

$$F_\delta(\mathcal{W}_a) \leqslant n + \ln \frac{\sqrt{n}}{\delta^{n-1}},$$

and assuming compatibility, by Lemma 1 the number of invocations is at most

$$\sum_a F_\delta(\mathcal{W}_a) \leqslant k\left(n + \ln \frac{\sqrt{n}}{\delta^{n-1}}\right) =$$
$$= k\left(n + \tfrac{1}{2}\ln n + (n-1)\ln \tfrac{1}{\delta}\right) <$$
$$< nk\left(\ln \tfrac{1}{\delta} + 2\right) \leqslant 2nk \ln \tfrac{1}{\delta}.$$

□

Lemma 3 (Update Correctness)**.** *Assuming compatibility, $\mathcal{M}$ is always a μ-approximation of M', where $\mu = 8en\mu'$.*

Proof. The empty initialization of $\mathcal{M}$ is a μ-approximation of any real model.

Consider the values of τ and $\mathcal{W}_a \neq \varnothing$ after they are updated. Let $0 \leqslant \delta' \leqslant \epsilon'$, $b \in \mathcal{B}_a^{\mathcal{W},\delta'}$ and $b' = \sum_w c_w w \in Sp(\mathcal{W}_a, e)$ such that $\sum_w c_w = 1$ and $\|b' - b\|_1 \leqslant \delta'$. Since $c_w \leqslant e$ for every $w \in \mathcal{W}_a$, $\sum_w |c_w|$ is maximal when one of the c_ws is $1 - e(|\mathcal{W}_a| - 1)$ and the others are e. $|\mathcal{W}_a| = d(\mathcal{W}_a) + 1 \leqslant n$, so $\sum_w |c_w| \leqslant \max(2e(|\mathcal{W}_a| - 1) - 1, 1) < 2en$. Compatibility and the second group of constraints in the Update procedure then give

$$\left\| \sum_s b(s)(\tau(s,a) - t'(s,a)) \right\|_1 \leqslant$$
$$\leqslant \left\| \sum_s (b(s) - b'(s))\tau(s,a) \right\|_1 + \left\| \sum_s b'(s)(\tau(s,a) - t'(s,a)) \right\|_1 + \left\| \sum_s (b'(s) - b(s))t'(s,a) \right\|_1 \leqslant$$
$$\leqslant \sum_w |c_w| \left\| \sum_s w(s)(\tau(s,a) - t'(s,a)) \right\|_1 + 2\|b' - b\|_1 \leqslant$$
$$\leqslant \sum_w |c_w| \left(\left\| \sum_s w(s)\tau(s,a) - f_{w,a} \right\|_1 + \left\| f_{w,a} - \sum_s w(s)t'(s,a) \right\|_1 \right) + 2\delta' \leqslant$$
(constraints and compatibility)
$$\leqslant \sum_w |c_w| 4\mu' + 2\delta' <$$
$$< 8en\mu' + 2\delta' = \mu + 2\delta'.$$

□

7.1.5 The Main Procedure

Bearing in mind that the Update procedure requires independent samples of the beliefs *before* as well as *after* the action, we revise the main procedure to produce both these samples. Recall that for a fixed initial state and a fixed deterministic policy which takes α only initially, the belief in every step is also fixed. The algorithm will sample these beliefs, starting with the first step and iterating for later and later steps. The variable steps_σ is the number of steps, in phases starting in state σ, for which the beliefs were already sampled. If at any step an Update is required, that is, $\mathrm{sample}_{\mathrm{before}} \notin \mathcal{B}_a^{\sqrt{n}\delta}$, then the Update procedure is invoked, a new policy is calculated, and the samples, which are no longer meaningful, are discarded, setting steps_σ to 0.

A record rec_x is a collection, possibly with repetitions, of elements of S. Let $\mathrm{freq}_x \in \Delta(S)$ be the relative frequencies of elements in rec_x, i.e. $\mathrm{freq}_x(s) = \frac{\#_s(\mathrm{rec}_x)}{|\mathrm{rec}_x|}$, where $\#_s(\mathrm{rec}_x)$ is the number of occurrences of s in rec_x.

UR-MAX(S, A, r, C, T, ϵ)

1. Initialize a completely unknown model $\mathcal{M}$, i.e. let $\mathcal{W}_a = \varnothing$ for each $a \in A'$
2. Let π be a deterministic T_1-step policy, which is optimal in $\mathcal{M}$ among those which use α in the first step and only then
3. Let $\mathrm{rec}_{s,\mathrm{before}} = \{K \text{ copies of } s\}$, let $\mathrm{rec}_{s,\mathrm{after}} = \varnothing$, and $\mathrm{steps}_s = 0$, for each $s \in S$
4. Repeat for each T_1-step phase:
   - 4.1. Empty the history used for planning, set it to the initial history Λ
   - 4.2. Take α and observe the initial state σ
   - 4.3. Take steps_σ more steps according to π
   - 4.4. If $\mathrm{steps}_\sigma < T_1 - 1$
     - 4.4.1. Let a be the last action taken
     - 4.4.2. Take α and add the observed state to $\mathrm{rec}_{\sigma,\mathrm{after}}$
     - 4.4.3. If $|\mathrm{rec}_{\sigma,\mathrm{after}}| = K$
       - If $\mathrm{freq}_{\sigma,\mathrm{before}} \notin \mathcal{B}_a^{\sqrt{n}\delta}$
         - Update($\mathrm{freq}_{\sigma,\mathrm{before}}$, a, $\mathrm{freq}_{\sigma,\mathrm{after}}$)
         - Do lines 2 and 3
       - Otherwise
         - Let $\mathrm{rec}_{\sigma,\mathrm{before}} = \mathrm{rec}_{\sigma,\mathrm{after}}$ and $\mathrm{rec}_{\sigma,\mathrm{after}} = \varnothing$
         - Increase steps_σ by 1
     - 4.4.4. Take $T_1 - \mathrm{steps}_\sigma - 2$ arbitrary non-α actions (fixed-length phases for easier analysis)

First note that the assumptions made in the last subsection about the context of the Update procedure are indeed correct: $\mathcal{M}$ is initialized empty, only updated by the Update procedure, and only when $\mathrm{sample}_{\mathrm{before}} \notin \mathcal{B}_a^{\sqrt{n}\delta}$. That doesn't include compatibility, which only occurs with high probability, as discussed below. Also note again that an optimal policy in $\mathcal{M}$ may require exponential time to calculate, but the interaction complexity will be polynomial nonetheless.

Bounding the interaction complexity means bounding the number of T_1-step phases the algorithm requires to reach near-optimality. Let T_2 be that minimal number of phases. The interaction complexity of the algorithm will then be $T_1 T_2$, and both these parameters will have to be polynomial. The parameters use by the algorithm are T_1 and K of the main procedure, δ and μ' of the Update procedure, ϵ' of the hypothetical model, and the minimal number T_2 of phases needed for near-optimality. Sufficient values for these parameters will be determined as we analyze the algorithm.

7.1.6 Analysis of UR-MAX

We shall now prove the correctness and efficiency of the UR-MAX algorithm. We argue that, with high probability, in almost every phase, the algorithm implements a near-optimal policy. This involves three propositions:

- With probability at least $1 - \frac{\epsilon}{4}$ the records are accurate enough representations of the beliefs according to which their entries are distributed. This includes compatibility of the samples, and also another requirement which we describe below. This proposition is proved in Corollary 6.

- Given these adequate records, at least a $1 - \frac{\epsilon}{4}$ fraction of the phases are *exploiting*, i.e. have $\mathrm{steps}_\sigma = T_1 - 1$. This proposition follows directly from Corollary 4.

- These exploiting phases yield a return of at least $U^*_M(T) - \frac{\epsilon}{4}$. This proposition is proved in Lemma 8.

We then combine these in Theorem 9 to show that running UR-MAX for $T' \geqslant T_1 T_2$ steps yields a return of at least $U^*_M(T) - \epsilon$, while T_1 and T_2 are polynomial in T, ϵ, $|M|$ and C.

Here, again, $|S| = n$ and $|A'| = k$.

Corollary 4 (Number of Entries)**.** *Assuming compatibility, at most $2n^2 k K T_1 \ln \frac{1}{\delta}$ entries are ever added to the records.*

Proof. Each time steps_σ is increased by 1, a complete record is thrown away. We can think of steps_σ as representing the number of such records "in sight".

When the Update procedure is invoked, we lose sight of these records, as well as the current ones. There are at most $n(T_1 - 1)$ records represented by the n variables $\mathrm{steps}_s \leqslant T_1 - 1$, and $2n$ records of the forms $\mathrm{rec}_{s,\mathrm{before}}$ and $\mathrm{rec}_{s,\mathrm{after}}$. From this number we should deduct the n records which were initialized to size K in line 3 without their entries actually being added. This makes a total of at most nT_1 records. Records are bounded in size by K, so at most nKT_1 entries are added between consecutive Update invocations.

After D invocations of the Update procedure, before the entry which would cause another invocation, at most $nKT_1(D+1)$ entries could have ever been added. Since by Corollary 2, assuming compatibility, $D+1 \leqslant 2nk\ln\frac{1}{\delta}$, at most $2n^2kKT_1\ln\frac{1}{\delta}$ entries are ever added to the records. □

The compatibility of samples doesn't tie them to any particular real belief. It merely states that the belief $\mathrm{sample}_{\mathrm{after}}$ approximates the one which follows the belief $\mathrm{sample}_{\mathrm{before}}$ when a is taken. This is good enough for the Update procedure, but we require more when discussing the source of these samples in the main procedure. In particular, we need to assert the relevance of checking whether a sample is in $\mathcal{B}_a^{\sqrt{n}\delta}$.

The record $\mathrm{rec}_{\sigma,\mathrm{after}}$, which is the one receiving new entries in phases starting with σ, is emptied whenever π is recalculated or steps_σ is changed. This means that in any single record, the initial state and the policy are the same for all entries, and so is the number of steps taken in the phase before generating the entry. As explained earlier, this makes all the entries in a record independently and identically distributed (i.i.d.) according to the belief in that step of the phase, and the next action fixed. Note the special case in which the record $\mathrm{rec}_{\sigma,\mathrm{before}}$, when initialized, contains K occurrences of σ, which are of course i.i.d. according to b_σ.

Definition 15 (Successful Record Completion)**.** For a sample $\mathrm{freq}_{\sigma,\mathrm{before}}$, let b be the belief according to which the entries of $\mathrm{rec}_{\sigma,\mathrm{before}}$ are i.i.d., and a the action that follows. The completion of $\mathrm{rec}_{\sigma,\mathrm{after}}$ to size K is *successful* if at that time $(\mathrm{freq}_{\sigma,\mathrm{before}}, a, \mathrm{freq}_{\sigma,\mathrm{after}})$ is compatible, and also

$$\left\|\mathrm{freq}_{\sigma,\mathrm{before}} - b\right\|_1 \leqslant \mu'.$$

Although $\mathrm{rec}_{\sigma,\mathrm{after}}$ is the one being completed, it's convenient to tie $\mathrm{freq}_{\sigma,\mathrm{before}}$ to the real belief b. $\mathrm{freq}_{\sigma,\mathrm{after}}$ is then tied too, since the samples are compatible.

Lemma 5 (Completion Success)**.** *A record completion is unsuccessful with probability at most* $4ne^{-2K\mu'^2/n^2}$.

Proof. Consider the completion to size K of $\mathrm{rec}_{\sigma,\mathrm{after}}$, for some $\sigma \in S$, and let b be the belief according to which entries in $\mathrm{rec}_{\sigma,\mathrm{before}}$ are i.i.d., and a the following action. Then $a \neq \alpha$ or b is pure, because α only occurs initially, when b is chosen by the algorithm to be pure. So entries in $\mathrm{rec}_{\sigma,\mathrm{after}}$ are i.i.d. according to $b' = \sum_s b(s)t'(s,a)$. Since $|\mathrm{rec}_{\sigma,\mathrm{before}}| = |\mathrm{rec}_{\sigma,\mathrm{after}}| = K$, we expect that $\tilde{b} = \mathrm{freq}_{\sigma,\mathrm{before}}$ and $\tilde{b}' = \mathrm{freq}_{\sigma,\mathrm{after}}$ should approximate b and b', respectively.

So we have

$$\Pr\left(\left\|\tilde{b}' - \sum_s \tilde{b}(s)t'(s,a)\right\|_1 > 2\mu' \vee \left\|\tilde{b} - b\right\|_1 > \mu'\right) \leqslant$$

$$(b' = \textstyle\sum_s b(s)t'(s,a))$$

$$\leqslant \Pr\left(\left\|\tilde{b}' - b'\right\|_1 + \left\|\sum_s \left(b(s) - \tilde{b}(s)\right)t'(s,a)\right\|_1 > 2\mu' \vee \left\|\tilde{b} - b\right\|_1 > \mu'\right) \leqslant$$

$$\leqslant \Pr\left(\left\|\tilde{b}' - b'\right\|_1 + \left\|b - \tilde{b}\right\|_1 > 2\mu' \vee \left\|\tilde{b} - b\right\|_1 > \mu'\right) \leqslant$$

$$\leqslant \Pr\left(\left\|\tilde{b}' - b'\right\|_1 > \mu' \vee \left\|\tilde{b} - b\right\|_1 > \mu'\right) \leqslant$$

$$\leqslant \Pr\left(\bigvee_s \left(\left|\tilde{b}'(s) - b'(s)\right| > \mu'/n \vee \left|\tilde{b}(s) - b(s)\right| > \mu'/n\right)\right) \leqslant$$

$$\leqslant \sum_s \left(\Pr\left(\left|\tilde{b}'(s) - b'(s)\right| > \mu'/n\right) + \Pr\left(\left|\tilde{b}(s) - b(s)\right| > \mu'/n\right)\right) \leqslant$$

(Hoeffding's inequality)

$$\leqslant 4ne^{-2K\mu'^2/n^2}.$$

□

Combining Corollary 4 and Lemma 5, we have

Corollary 6 (Total Success)**.** *With probability at least* $1 - 8n^3kT_1e^{-2K\mu'^2/n^2}\ln\frac{1}{\delta}$*, all record completions are successful.*

Proof. A union bound guarantees that of at most $2n^2kT_1\ln\frac{1}{\delta}$ completions of a record to size K, each unsuccessful with probability at most $4ne^{-2K\mu'^2/n^2}$, the probability for any completion to be unsuccessful is at most $8n^3kT_1e^{-2K\mu'^2/n^2}\ln\frac{1}{\delta}$. □

If we set

$$K = \left\lceil \frac{n^2}{2\mu'^2}\ln\left(32n^3kT_1\frac{1}{\epsilon}\ln\frac{1}{\delta}\right)\right\rceil,$$

we get that the probability of total success is at least $1 - \frac{\epsilon}{4}$, as promised.

In UR-MAX, *exploring* phases are those in which $\text{steps}_\sigma < T_1 - 1$, and therefore an entry is added to the records. By Corollary 4, if we set

$$T_2 = \left\lceil 8n^2kKT_1\frac{1}{\epsilon}\ln\frac{1}{\delta}\right\rceil,$$

at most a $\frac{\epsilon}{4}$ fraction of the phases are exploring, and the others are *exploiting* phases, in which $\text{steps}_\sigma = T_1 - 1$.

Before our main theorem, it remains to be seen that these exploiting phases yield near-optimal return. For that we need a lemma which bounds the divergence of the beliefs in M' and in $\mathcal{M}$.

Lemma 7 (Belief Approximation)**.** *Let* $\mathcal{M} = \langle \mathcal{S}, A', O, \tau, \rho, \theta', C, \mathcal{W}, \epsilon'\rangle$ *be a* μ*-approximation of an OCOMDP* $M' = \langle S, A', O, t', r, \theta', C\rangle$*,* $0 \leqslant \delta' \leqslant \epsilon'$*,* a *an action and* b *and* $\bar{b}$ *the beliefs reached in* M' *and in* $\mathcal{M}$*, respectively. Assume that* $\bar{b} \in \mathcal{B}_a^{\mathcal{W},\epsilon'}$*, and that either* b *or* $\bar{b}$ *(or both) are in* $\mathcal{B}_a^{\mathcal{W},\delta'} \subseteq \mathcal{B}_a^{\mathcal{W},\epsilon'}$*. If* $a = \alpha$*, assume further that* b *and* $\bar{b}$ *are pure. Then, the beliefs* b' *and* $\bar{b}'$ *reached after taking* a *in* M' *and in* $\mathcal{M}$*, respectively, satisfy*

$$\left\|\bar{b}' - b'\right\|_1 \leqslant \left\|\bar{b} - b\right\|_1 + \mu + 2\delta'.$$

Proof. If $b \in \mathcal{B}_a^{\mathcal{W},\delta'}$,

$$\left\|\bar{b}' - b'\right\|_1 = \left\|\sum_s \left(\bar{b}(s)\tau(s,a) - b(s)t'(s,a)\right)\right\|_1 \leqslant$$

$$\leqslant \left\|\sum_s \left(\bar{b}(s) - b(s)\right)\tau(s,a)\right\|_1 + \left\|\sum_s b(s)\left(\tau(s,a) - t'(s,a)\right)\right\|_1 \leqslant$$

$$\leqslant \left\|\bar{b} - b\right\|_1 + \mu + 2\delta'.$$

If $\bar{b} \in \mathcal{B}_a^{\mathcal{W},\delta'}$, the proof is symmetrical. □

In Lemma 8 we need to assert

- $T \,\Big|\, T_1 - 1$,
- $\frac{\epsilon}{4} \geqslant \frac{C+2}{T_1} + 4\epsilon' T_1$, and
- $\epsilon' \geqslant (9en\mu' + 2\sqrt{n}\delta)T_1$.

So we set

$$T_1 = \left\lceil \frac{8(C+2)}{\epsilon T} \right\rceil T + 1,$$

$$\epsilon' = \frac{\epsilon}{32T_1},$$

$$\delta = \frac{\epsilon'}{4\sqrt{n}T_1},$$

and

$$\mu' = \frac{\epsilon'}{18enT_1}.$$

Lemma 8 (Exploiting Phases)**.** *Assuming successful record completions, every exploiting phase has a return of at least* $U_M^*(T) - \frac{\epsilon}{4}$.

Proof. This is a version of the Implicit Explore or Exploit Lemma of R-MAX [4].

Consider the values of $\mathcal{M}$, σ and π in a phase in which $\mathrm{steps}_\sigma = T_1 - 1$, so that no entries are added to the records, and π is implemented for T_1 steps. Let b_i and $\bar{b}_i$, for $0 \leqslant i \leqslant T_1$ be the beliefs which deterministically result in the real model M' and in $\mathcal{M}$, respectively, after using π for i steps, starting from the belief b_σ.

Consider the $T_1 - 1$ times in which steps_σ had to be increased to reach $T_1 - 1$. The fact that these increases occurred indicates that b_i has been sampled K times when steps_σ was $i - 1$, for $1 \leqslant i < T_1$. The sample for $b_0 = b_\sigma$ is explicitly initialized. Let sample_i, for $0 \leqslant i < T_1$, be these samples.

In addition, the algorithm verifies that $\mathrm{sample}_i \in \mathcal{B}_{a_i}^{\sqrt{n}\delta}$, where a_i is the action in step i, for $0 \leqslant i < T_1 - 1$. Note that this excludes the last step, because b_{T_1} is not sampled and $\mathrm{sample}_{T_1-1} \in \mathcal{B}_{a_{T_1-1}}^{\sqrt{n}\delta}$ is not verified and may be false.

We now prove by induction on i that $\left\|\bar{b}_i - b_i\right\|_1 \leqslant \frac{\epsilon' i}{T_1}$, for $0 \leqslant i < T_1$. As a base for the induction, $\left\|\bar{b}_0 - b_0\right\|_1 = 0$.

Now assume the proposition holds for some $0 \leqslant i < T_1 - 1$, and prove it for $i+1$. Let $b' \in Sp(\mathcal{W}_{a_i}, e)$ be such that $\|b' - \text{sample}_i\|_1 \leqslant \sqrt{n}\delta$. By the assumption of completion success

$$\|b' - b_i\|_1 \leqslant \|b' - \text{sample}_i\|_1 + \|\text{sample}_i - b_i\|_1 \leqslant \sqrt{n}\delta + \mu',$$

and by the induction assumption

$$\left\|b' - \bar{b}_i\right\|_1 \leqslant \|b' - b_i\|_1 + \left\|b_i - \bar{b}_i\right\|_1 \leqslant$$

$$\leqslant \sqrt{n}\delta + \mu' + \tfrac{\epsilon' i}{T_1} < \epsilon'.$$

Therefore, $b_i \in \mathcal{B}_{a_i}^{\sqrt{n}\delta+\mu'}$, and $\bar{b}_i \in \mathcal{B}_{a_i}^{\epsilon'}$. By taking $\delta' = \sqrt{n}\delta + \mu' < \epsilon'$ in Lemma 7, we have

$$\left\|\bar{b}_{i+1} - b_{i+1}\right\|_1 \leqslant \left\|\bar{b}_i - b_i\right\|_1 + \mu + 2\sqrt{n}\delta + 2\mu' <$$

$$< \tfrac{\epsilon' i}{T_1} + 9en\mu' + 2\sqrt{n}\delta = \tfrac{\epsilon'(i+1)}{T_1},$$

which completes the induction.

Note that $\bar{b}_i \in \mathcal{B}_{a_i}^{\epsilon'}$, for every $0 \leqslant i < T_1 - 1$, means that s^* is not reached in $\mathcal{M}$, except maybe in the last step. Now since α is used exactly once

$$U_{M'}(b_\sigma, \pi, T_1) = \frac{1}{T_1} \sum_{i=1}^{T_1} \sum_{s \in S} b_i(s) r(s) - \frac{C}{T_1},$$

$$U_{\mathcal{M}}(b_\sigma, \pi, T_1) = \frac{1}{T_1} \sum_{i=1}^{T_1} \sum_{s \in \mathcal{S}} \bar{b}_i(s) \rho(s) - \frac{C}{T_1},$$

and

$$|U_{\mathcal{M}}(b_\sigma, \pi_\sigma, T_1) - U_{M'}(b_\sigma, \pi_\sigma, T_1)| \leqslant$$

$$\leqslant \frac{1}{T_1} \sum_{i=1}^{T_1} \left| \sum_{s \in \mathcal{S}} \left(\bar{b}_i(s) \rho(s) - b_i(s) r(s) \right) \right| \leqslant$$

(s^* is not reached for $i < T_1$)

$$\leqslant \frac{1}{T_1} \sum_{i=1}^{T_1 - 1} \left| \sum_{s \in S} \left(\bar{b}_i(s) - b_i(s) \right) r(s) \right| + \frac{1}{T_1} \leqslant$$

$$\leqslant \frac{1}{T_1} \sum_{i=1}^{T_1 - 1} \left\| \bar{b}_i - b_i \right\|_1 + \frac{1}{T_1} \leqslant$$

$$\leqslant \frac{1}{T_1} \sum_{i=1}^{T_1 - 1} \frac{\epsilon' i}{T_1} + \frac{1}{T_1} \leqslant \frac{\epsilon'}{2} + \frac{1}{T_1}.$$

In order to compare to the optimal T-step policy, we consider the following structure. We start the phase with α, and continue with a sequence of deterministic T-step policies. Each policy is optimal for the belief it starts with, among policies which don't use α.

Let $b_1^* = b_1$ and $\bar{b}_1^* = \bar{b}_1$. We define the policies and the beliefs recursively, for l from 0 to $\frac{T_1-1}{T} - 1$:

- π_l^* is a deterministic policy in M which maximizes $U_M(b_{lT+1}^*, \pi_l^*, T)$,
- $(a_i^*)_{i=lT+1}^{(l+1)T} \in A^T$ is the sequence of actions taken by π_l^*, and
- b_i^* and $\bar{b}_i^*$, for $lT+1 < i \leqslant (l+1)T+1$, are the beliefs which deterministically result in M and in $\mathcal{M}$, respectively, after taking the actions $(a_j^*)_{j=lT+1}^{i-1}$ from the beliefs b_{lT+1}^* and $\bar{b}_{lT+1}^*$, respectively.

Let π^* be the concatenation of these π_l^*s, i.e. the deterministic T_1-step policy in M' which takes the actions $(a_i^*)_{i=0}^{T_1-1}$, with $a_0^* = \alpha$. Where $b_0^* = \bar{b}_0^* = b_\sigma$, let i_0 be the least $0 \leqslant i < T_1$ such that $\bar{b}_i^* \notin \mathcal{B}_{a_i^*}^{\epsilon'}$, or T_1 if no such index exists. By Lemma 7, $\left\|\bar{b}_i^* - b_i^*\right\|_1 \leqslant (\mu + 2\epsilon')i$, for every $0 \leqslant i \leqslant i_0$. Then

$$U_{\mathcal{M}}(b_\sigma, \pi^*, T_1) = \frac{1}{T_1}\sum_{i=1}^{T_1}\sum_{s\in\mathcal{S}}\bar{b}_i^*(s)\rho(s) - \frac{C}{T_1} =$$

$$= \frac{1}{T_1}\left(\sum_{i=1}^{i_0}\sum_{s\in S}\bar{b}_i^*(s)r(s) + \sum_{i=i_0+1}^{T_1}\rho(s^*)\right) - \frac{C}{T_1} \geqslant$$

$$\geqslant \frac{1}{T_1}\left(\sum_{i=1}^{i_0}\sum_{s} b_i^*(s)r(s) + \sum_{i=i_0+1}^{T_1}\rho(s^*)\right) - \frac{C}{T_1} - \frac{1}{T_1}\sum_{i=1}^{i_0}\left|\sum_s\left(\bar{b}_i^*(s) - b_i^*(s)\right)r(s)\right| \geqslant$$

$$(\rho(s^*) \geqslant r(s))$$

$$\geqslant \frac{1}{T_1}\sum_{i=2}^{T_1}\sum_{s} b_i^*(s)r(s) - \frac{C}{T_1} - \frac{1}{T_1}\sum_{i=1}^{i_0}\left\|\bar{b}_i^* - b_i^*\right\|_1 \geqslant$$

$$\geqslant \frac{1}{T_1}\sum_{l=0}^{\frac{T_1-1}{T}-1}\sum_{i=1}^{T}\sum_{s} b_{lT+1+i}^*(s)r(s) - \frac{C}{T_1} - \frac{1}{T_1}(\mu+2\epsilon')\frac{(i_0+1)^2}{2} \geqslant$$

$$\geqslant \frac{1}{T_1}\sum_{l=0}^{\frac{T_1-1}{T}-1} TU_M(b_{lT+1}^*, \pi_l^*, T) - \frac{C}{T_1} - (8en\mu' + 2\epsilon')\frac{(T_1+1)^2}{2T_1} >$$

$$(T_1 \geqslant 2)$$

$$> \frac{1}{T_1}\sum_{l=0}^{\frac{T_1-1}{T}-1} TU_M^*(T) - \frac{C}{T_1} - (\tfrac{\epsilon'}{2T_1} + 2\epsilon')\tfrac{9}{8}T_1 >$$

$$> \frac{T_1-1}{T_1}U_M^*(T) - \frac{C}{T_1} - 3\epsilon' T_1.$$

Observe that π^* satisfies the constraints under which π is optimal for $\mathcal{M}$. Summing it all up, we have

$$\begin{aligned}
&U_{M'}(b_\sigma, \pi, T_1) \geqslant \\
&\geqslant U_{\mathcal{M}}(b_\sigma, \pi, T_1) - \frac{\epsilon'}{2} - \frac{1}{T_1} \geqslant \\
&\geqslant U_{\mathcal{M}}(b_\sigma, \pi^*, T_1) - \frac{\epsilon'}{2} - \frac{1}{T_1} \geqslant \\
&\geqslant \frac{T_1 - 1}{T_1} U^*_M(T) - \frac{C}{T_1} - 3\epsilon' T_1 - \frac{\epsilon'}{2} - \frac{1}{T_1} \geqslant \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (U^*_M(T) \leqslant 1) \\
&\geqslant U^*_M(T) - \frac{C+2}{T_1} - 4\epsilon' T_1 \geqslant U^*_M(T) - \frac{\epsilon}{4}.
\end{aligned}$$

□

Finally we are ready for our main theorem, for which we summarize our choice of parameters.

$$\begin{aligned}
T_1 &= \left\lceil \tfrac{8(C+2)}{\epsilon T} \right\rceil T + 1 & \mu' &= \tfrac{\epsilon'}{18enT_1} \\
\epsilon' &= \tfrac{\epsilon}{32T_1} & K &= \left\lceil \tfrac{n^2}{2\mu'^2} \ln\left(32n^3kT_1 \tfrac{1}{\epsilon} \ln \tfrac{1}{\delta}\right) \right\rceil \\
\delta &= \tfrac{\epsilon'}{4\sqrt{n}T_1} & T_2 &= \left\lceil 8n^2kKT_1 \tfrac{1}{\epsilon} \ln \tfrac{1}{\delta} \right\rceil
\end{aligned}$$

Theorem 9 (UR-MAX Correctness)**.** *Let $M' = \langle S, A', O, t', r, \theta', C\rangle$ be an OCOMDP which extends a UMDP $M = \langle S, A, \{\perp\}, r, t, \theta\rangle$. Let b_0 be some initial state distribution, T a horizon, and $\epsilon > 0$ a margin. Let $\pi_{\mathcal{A}}$ and $T_{\mathcal{A}}$ be the policy and the number of steps, respectively, used by* UR-MAX*(S, A, r, C, T, ϵ). Then*

$$T_{\mathcal{A}} = \text{poly}\left(T, \tfrac{1}{\epsilon}, |M|, C\right),$$

and for every $T' \geqslant T_{\mathcal{A}}$

$$U_{M'}\left(b_0, \pi_{\mathcal{A}}, T'\right) \geqslant U^*_M(T) - \epsilon.$$

Proof. According to our choice of parameters, $T_{\mathcal{A}} = T_1 T_2 = \text{poly}\left(T, \frac{1}{\epsilon}, |M|, C\right)$.

Suppose that the interaction lasts $T' \geqslant T_{\mathcal{A}}$ steps. For convenience of the proof, append to T' one dummy step, and then some more to complete the last T_1-step phase. For these dummy steps there's of course no reward. Then the algorithm goes through $\left\lceil \frac{T'+1}{T_1} \right\rceil \geqslant T_2 + 1$ phases.

By Corollary 6, there's a probability of at least $1 - \frac{\epsilon}{4}$ that all record completions are successful. Assuming that, by Corollary 4 the fraction of non-exploiting phases (including the dummy phase) is at most

$$\frac{2n^2kKT_1 \ln \frac{1}{\delta}}{\left\lceil 8n^2kKT_1 \frac{1}{\epsilon} \ln \frac{1}{\delta} \right\rceil + 1} < \frac{\epsilon}{4}.$$

So the probability that a phase chosen uniformly at random is exploiting is at least $(1-\frac{\epsilon}{4})^2$. If it is exploiting, Lemma 8 guarantees a return of at least $U_M^*(T)-\frac{\epsilon}{4}$. Otherwise, it may utilize α twice, and yield as worse a return as $-\frac{2C}{T_1}$. We assume $\epsilon \leqslant U_M^*(T) \leqslant 1$ because otherwise the learning problem is trivial. It follows that $U_M^*(T)-\frac{\epsilon}{4} > 0 > -\frac{2C}{T_1}$.

So we have

$$U_{M'}(b_0,\pi_{\mathcal{A}},T') \geqslant$$

$$\geqslant \left(1-\tfrac{\epsilon}{4}\right)^2\left(U_M^*(T)-\tfrac{\epsilon}{4}\right)-\left(1-\left(1-\tfrac{\epsilon}{4}\right)^2\right)\tfrac{2C}{T_1} \geqslant$$

$$\geqslant U_M^*(T)-\tfrac{\epsilon}{2}U_M^*(T)-\tfrac{\epsilon}{4}+\tfrac{\epsilon^2}{8}-\tfrac{\epsilon^3}{64}-\tfrac{\epsilon C}{T_1} >$$

$$(T_1 > \tfrac{8C}{\epsilon} > 4C)$$

$$> U_M^*(T)-\tfrac{\epsilon}{2}-\tfrac{\epsilon}{4}-\tfrac{\epsilon}{4} = U_M^*(T)-\epsilon.$$

□

7.2 Learning States and Rewards

Obviously the agent must be given the set A of possible actions, for otherwise it can't even begin to interact. As for S, r and C, those can sometimes be learned from the interaction, instead of being given as input.

C can simply be learned if we assume that it can be observed when first payed.

For learning r, we assume that when α observes a state, its reward is visible as well. It can be verified that the rewards of states which are never observed by α play absolutely no role in UR-MAX. These states are missing from the records when learning, and therefore aren't reachable when planning in $\mathcal{M}$. It's enough then to take note of the value of $r(s)$ when s is first observed.

The names of the states are similarly learned when observed. However, $n=|S|$ is required for the calculation of parameters. As n grows, so does $T_{\mathcal{A}}$, the number of steps required to ensure learning. One could imagine a model in which larger and larger subsets of states can be reached with lower and lower probability. Intuitively speaking, an algorithm guessing a bound N on the number of states in such a model will have to use a number of steps which will take it through much more than N states, with high probability. If the algorithm iteratively updates the bound, it may have to do so a large number of times. Compensating for interaction spent with the bound too low may then be a problem.

There are two approaches for dealing with this problem.

The first approach makes additional assumptions which restrict the class of models we try to learn. For example, suppose that non-zero transition probabilities can't be arbitrarily small, but rather are bounded from below by a known constant $\gamma > 0$. Then if transition probabilities are estimated, in all samples, with error less than γ, the states which aren't yet reached simply can't be reached. The following algorithm enables learning in such a scenario, using as a building block UR-MAX, altered to estimate probabilities to within an error of γ or lower.

Suppose that up to some point only $n' < n$ states of M are observed (initially, $n' = 0$). Imagine a model $\bar{M}$ with $n' + 1$ states, in which the set S^u of all unobserved states of M is replaced with a single state. This state has maximal reward and leads deterministically to itself, for any action. The probability in $\bar{M}$ of reaching this fictitious state from an observed state $s \notin S^u$ is the probability in M of reaching any $s' \in S^u$ from s. Other transitions remain the same.

The algorithm runs UR-MAX with $\bar{M}$'s states as input. If new states are revealed, the process is repeated, again and again, with increasing values of n'. Eventually UR-MAX is run without revealing new states. At that point the algorithm is unable to tell if the real model is M or $\bar{M}$. Since UR-MAX competes with the optimal return in $\bar{M}$, which is at least as high as the optimal return of M, it is indeed near-optimal. The algorithm can then continue for some more steps to compensate for the interaction spent with too few states. According to our assumptions, during this period no new states can be reached.

The other approach sacrifices complexity for generality. Even if we don't make any assumptions on the model, we can learn with interaction complexity which is not much worse than polynomial. For example, suppose that we use the same algorithm as above, but this time we start with $n' = 2$, and square n' every time we reach a previously undiscovered state. That is, for $i = 0, 1, 2, \ldots$ we run UR-MAX with 2^{2^i} states and margin $\epsilon/2$.

When this number of states eventually exceeds the real number of states n, the return of the last iteration is at most $\epsilon/2$ less than optimal. The overall return may be $\epsilon/2$ lower than that, due to the interaction spent with too few states. To guarantee this, we don't simply run UR-MAX for the number of steps it suggests, but rather for a larger number of steps. If T_i is the number of steps required by UR-MAX in iteration i, we actually use it for T'_i steps, where

$$T'_i = \max\left(T_i, \left(\tfrac{2}{\epsilon} - 1\right) \sum_{j=0}^{i-1} T'_j\right),$$

so we have

$$\frac{\sum_{j=0}^{i-1} T'_j}{\sum_{j=0}^{i} T'_j} \leqslant \frac{\sum_{j=0}^{i-1} T'_j}{\sum_{j=0}^{i-1} T'_j \left(1 + \frac{2}{\epsilon} - 1\right)} = \tfrac{\epsilon}{2},$$

and

$$T'_i < T_i + \left(\tfrac{2}{\epsilon} - 1\right) \sum_{j=0}^{i-1} T'_j < \sum_{j=0}^{i} \left(\tfrac{2}{\epsilon} - 1\right)^{i-j} T_j.$$

The algorithm terminates in an iteration where no new states are encountered, which happens after at most $\lceil \log_2 \log_2 n \rceil + 1$ iterations (for $n > 1$). In addition, never more than n^2 states are assumed by the algorithm, so T_i is still polynomial. It follows that the interaction complexity of this algorithm is

$$\left(\tfrac{2}{\epsilon} - 1\right)^{\log_2 \log_2 n} \operatorname{poly}\left(T, \tfrac{1}{\epsilon}, |M|, C\right).$$

7.3 Learning a UMDP Policy

OCOMDPs extend UMDPs by adding a fully-observing action α. In a sense, this enables learning in UMDPs. As an example, consider the medical treatment of outpatients. In some medical fields, the patients' own feelings are not considered significant enough for diagnosis, and when they are not under medical observation, a UMDP can be suggested to describe the process they are going through. The states of the UMDP are possible prognoses, and the actions are possible medical interventions. The reward can be a measure of the patient's well-being, and can also incorporate the cost of treatment. Transition probabilities represent the distribution in the population of unpredictable factors, such as whether a given patient will react positively to some drug. The initial state (or distribution) is the patient's state when the condition is first diagnosed.

Observability in many cases of medical treatment is not impossible, but costly. It may require, for example, expensive hospital equipment. The model then becomes an OCOMDP, probably one in which an optimal policy still lacks observations. However, the approach taken by UR-MAX makes little sense in this case. In the setting of section 7.1, UR-MAX treats one particular patient repeatedly, applying various treatments again and again, while occasionally monitoring the state, until it concludes that the patient's condition can't be further helped.

What a medical researcher would do, simplistically, is take a large number of patients, divide them into groups, and subject each group to a different treatment. By monitoring each patient's reaction to treatment, the researcher would then be able to determine the evolution of the condition under different interventions, and recommend a good policy for cases where such monitoring is too costly.

To express this methodology in our notation, we make two adjustments. First, the UMDP will include a *reset* action, which always leads back to the initial state distribution. In the medical domain example, this action means switching to the next patient. Second, it's not enough for the algorithm to compete with the optimal policy of the underlying UMDP, just as it's not enough to cure patients during research. It must provide a near-optimal T-step policy that doesn't use α, so that after the learning period α can eventually be dropped.

This can be easily achieved. Note that in UR-MAX, α is used in exploiting phases only as a reset action, to provide a known belief which the agent has visited repeatedly and learned to act in. We simply change the way an optimal policy π is calculated in the hypothetical model $\mathcal{M}$. Instead of a deterministic T_1-step policy which is optimal among those which use α in the first step and only then, we calculate a deterministic T-step policy π' which is optimal among those which start with the reset action and never use α. Then we expand it to a deterministic T_1-step policy which is optimal among those which start with π' and never use α.

The proof that the algorithm is still near-optimal, compared with the optimal T-step policy in the underlying UMDP, is essentially identical to the proof of Theorem 9. A similar analysis shows that in exploiting phases (which eventually occur) π' is a near-optimal T-step policy in the UMDP.

When the UMDP is lacking a reset action, but satisfies some condition of connectedness, an *approximate reset strategy* [5] can be used to simulate such reset, and still learn a near-optimal UMDP policy. The interaction complexity, however, is generally not polynomial.

7.4 Learning with respect to the OCOMDP

The title of this chapter promises learning in OCOMDPs. Approaching the optimal return of the underlying UMDP, as interesting in itself as it is, was just an introduction. We still have to compete with the optimal return of the OCOMDP M', which is the actual model of the environment. If the cost C of taking α is small enough, there may be a policy which uses α to improve its return, beyond the optimal return of the UMDP.

The reason we had to compare the policy of UR-MAX to the optimal policy of the UMDP is that its planning module wasn't allowed to use α anywhere inside the policy. We did that in order to have deterministic phases, given the initial state. Let's see what is changed when we remove this restriction.

First, the computation complexity of planning may be larger now, but is not our concern here.

Note that nothing is changed in the Update procedure or in its analysis, as long as the assumptions it makes on its input are kept by the main procedure.

Phases now don't follow deterministically from the initial state, because of the observations of the in-phase α actions. But determinism is still preserved between α actions, in the following manner.

A sequence of steps starting with an α action and ending before the next will be called a *sub-phase*. Every phase can be broken into a sequence of consecutive sub-phases. Suppose that in two different sub-phases the α action gives the same observation, and then the same sequence of actions is taken. The beliefs will deterministically be the same in both sub-phases, and they can therefore contribute entries to the same records.

Formerly, the observable history after a given number of steps of a phase could take at most $|S|$ different values, one for each initial state. Now that taking α is allowed, and observation can be made in any step, the number of observable histories can become exponential. Since actions depend on histories, this may result in exponentially many different sub-phases, and require exponentially many records.

To address this difficulty, we restrict our attention to *stationary* policies, without loss of generality. Consider a policy π which maximizes $U_{M'}(b_0, \pi, T_1)$. After taking $T_1 - i$ steps, and reaching some belief b, the restriction of π to the remaining steps, $\pi_{b,i}$, has to maximize $U_{M'}(b, \pi_{b,i}, i)$. There may be more than one maximizing sub-policy $\pi_{b,i}$, and it's possible for π to choose different ones in different histories (if different histories lead to the same belief b). But it wouldn't harm optimality to choose the same sub-policy regardless of the history. A stationary policy uses actions based only on the current belief and the remaining horizon. Note that there always exists an optimal stationary policy.

Now, consider what happens when a stationary deterministic policy is used in an OCOMDP. Suppose that at some point α is taken. The next belief is determined by the observation σ, and the next action is determined by σ and the remaining horizon ι. Suppose that that action is not α, then the belief and action following it are also determined by σ and ι. This continues for the entire sub-phase, up until the next α action. The records we keep will reflect this new version of determinism.

The revised algorithm may seem a little more complicated, but it operates in essentially the same way.

UR-MAX(S, A, r, C, T, ϵ) (revised)

1. Initialize a completely unknown model $\mathcal{M}$, i.e. let $\mathcal{W}_a = \varnothing$ for each $a \in A'$
2. Let π be a stationary deterministic T_1-step policy, which is optimal in $\mathcal{M}$ among those which use α in the first step
3. Let $\mathrm{rec}_{s,i,\mathrm{before}} = \{K \text{ copies of } s\}$, let $\mathrm{rec}_{s,i,\mathrm{after}} = \varnothing$, and $\mathrm{steps}_{s,i} = 0$, for each $s \in S$ and $0 < i < T_1$
4. Repeat for each T_1-step phase:
   - 4.1. Take α and observe the initial state σ
   - 4.2. Let $\iota = T_1 - 1$ be the remaining horizon in the phase
   - 4.3. While $\iota > 0$, do another sub-phase:
     - 4.3.1. Take $\mathrm{steps}_{\sigma,\iota}$ steps according to π
     - 4.3.2. Decrease ι by $\mathrm{steps}_{\sigma,\iota}$
     - 4.3.3. Let a be the last action taken
     - 4.3.4. If ($a \neq \alpha$ or $\mathrm{steps}_{\sigma,\iota} = 0$) and $\iota > 0$
       - 4.3.4.1. Take α and add the observed state to $\mathrm{rec}_{\sigma,\iota,\mathrm{after}}$
       - 4.3.4.2. If $|\mathrm{rec}_{\sigma,\iota,\mathrm{after}}| = K$
         - If $\mathrm{freq}_{\sigma,\iota,\mathrm{before}} \notin \mathcal{B}_a^{\sqrt{n}\delta}$
           - Update($\mathrm{freq}_{\sigma,\iota,\mathrm{before}}$, a, $\mathrm{freq}_{\sigma,\iota,\mathrm{after}}$)
           - Do lines 2 and 3
         - Otherwise
           - Let $\mathrm{rec}_{\sigma,\iota,\mathrm{before}} = \mathrm{rec}_{\sigma,\iota,\mathrm{after}}$ and $\mathrm{rec}_{\sigma,\iota,\mathrm{after}} = \varnothing$
           - Increase $\mathrm{steps}_{\sigma,\iota}$ by 1
       - 4.3.4.3. Take ι arbitrary non-α actions and let $\iota = 0$

The explanation of line 4.3.4 is as follows. If $a = \alpha$, we go to the next sub-phase. The exception is when $\mathrm{steps}_{\sigma,\iota} = 0$, which means we're now sampling the beginning of a sub-phase. We also make sure that the phase hasn't ended, $\iota > 0$.

Only two modifications are required in the analysis.

First, there are more records to take into account in Corollary 4. This makes all the parameters a little larger, but still polynomial.

Second, in Lemma 8 we need to observe that, among the policies available to the planning module, is a policy which starts with α, and then repeatedly uses T-step policies which are optimal in M' for their respective initial beliefs. The rest of the proof remains the same.

8 Summary

In this thesis we have discussed the interaction complexity of several classes of decision processes in which observability is partial: POMDPs, OPOMDPs and OCOMDPs. The relationships between these and other important classes are illustrated in Figure 7.

One contribution of this thesis has been defining the concept of interaction complexity as an entity separate from the standard concept of computation complexity. Previously, publications concerned with complexity sought computationally efficient reinforcement learning algorithms, and viewed the efficiency of their interaction as a necessary and insufficient condition. We have argued that the interaction complexity is interesting in itself, and focused on reinforcement learning algorithms with polynomial interaction complexity.

Another attribute of most previous research is that it aims to solve the infinite horizon learning problem, but actually switches to the finite horizon form in algorithms and proofs. This reduction is often only implicit or discussed informally. Since the finite horizon form is more general and easier to work with, we have provided and proved a general Turing reduction from the infinite horizon form, which has allowed the rest of our work to apply equally well to both problems.

A major contribution of this thesis has been introducing classes of partially observable decision processes with provably polynomial interaction complexity. Importantly, the proofs have been constructive, i.e. we have provided algorithms for efficient reinforcement learning in these classes. One reason that this is interesting is that it indeed separates the interaction complexity from the computation complexity, by demonstrating learning problems which can be solved with polynomial interaction complexity, but not with efficient computation (according to standard complexity assumptions).

We have shown that learning in POMDPs, when possible, generally can't be achieved with polynomial interaction complexity. That is, we have shown a learnable subset of POMDPs which requires a number of interaction steps exponential in the size of the model.

We have then introduced two subclasses, OPOMDPs and OCOMDPs, the latter being our own contribution. Both these classes are too general to allow either efficient planning or computationally efficient learning. Computationally, the former class is at least as hard as POMDPs, and the latter at least as hard as UMDPs. However, we have shown that learning with polynomial interaction complexity is possible in both these classes.

Our algorithm for learning in OPOMDPs is only slightly more complicated than R-MAX, which learns efficiently in MDPs. OCOMDPs, on the other hand, have turned out to pose a considerable challenge, and the UR-MAX algorithm for efficiently learning them has been another major contribution of this thesis.

The basic version of the UR-MAX algorithm interacts with an unknown OCOMDP to obtain a return which is nearly the optimal return of the underlying UMDP. We have then extended the algorithm in two important ways: to produce a stand-alone policy for the underlying UMDP, and to compete with the OCOMDP itself. We have also dealt with the case where the number of states is unknown.

The problem of efficient learnability in partial observability is far from solved, and should be further investigated in future work. It is our hope that these first positive results of efficient learning in general classes of partially observable decision processes will inspire further research.

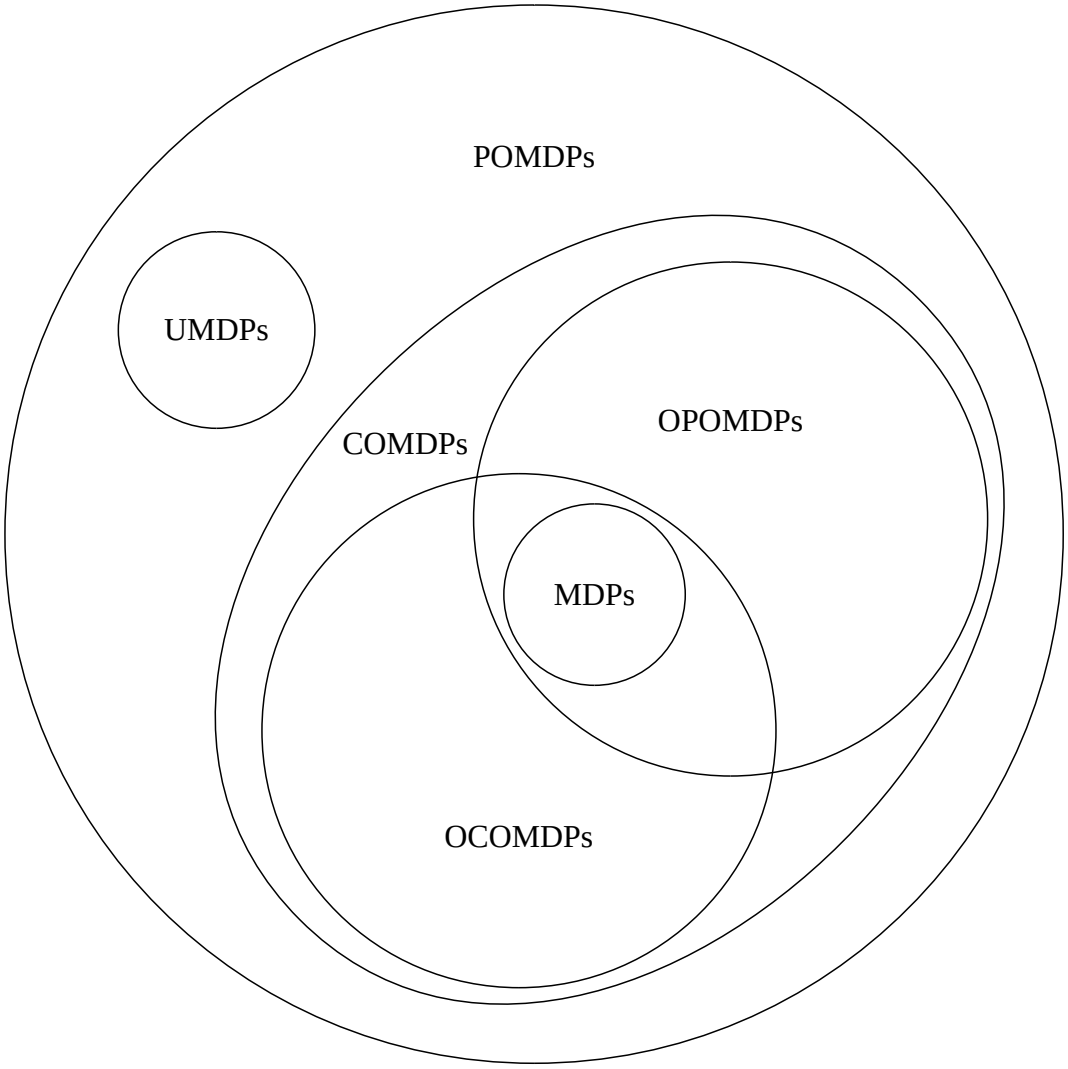

Figure 7: Classes of decision processes

A Appendix: Equivalence of Observation Models

In section 2.3 we argued that most common definitions of the observation model produce classes which are equivalent for planning. We now show why this is so.

First, having stochastic observations doesn't extend the class. As explained earlier, this just means that observations are generated "on demand" instead of just revealing a feature of the state. For example, compare $\theta : S \to \Delta(O)$ to $\theta' : S' \to O$. Of course, a model of the second type can easily be represented in the form of the first. For the reduction in the other direction, let $S' = S \times O$. Whenever a new state $s \in S$ is drawn, the observation $o \sim \theta(s)$ is immediately drawn too, to produce $(s, o) \in S'$. Then simply $\theta'(s, o) = o$ reveals this observation.

Now we show that it doesn't matter what part of the recent history the observation depends on. The types we consider are

1. $S \to O$, the current observation depends on the current state,

2. $S \times A \to O$, the observation is triggered by an action, and depends on the action and the state before it, and

3. $S \times A \times S \to O$, the action-triggered observation also depends on the state reached thereafter.

We show that these observation models are equivalent, in terms of planning in the classes they produce. We start with equivalence of the first and third types, and follow with equivalence of the second and third.

There are two differences between the first and third types of models. They both make an observation each time a state is reached, but only in the latter may the observation depend on the previous state and action. Reducing one model to the other is simply a matter of ignoring these variables, in one direction, or extending the state to include them and use them, in the other direction.

The other difference between these models is that in the first type an observation is made before the first action. When a model of the third type is reduced to one of the first type, the initial observation can simply be made constant. In the other direction, the reduction is not fixed. Instead, we first make the first observation, and only then we can calculate the reduction. The result will have different initial beliefs for different observations.

An observation model of the second form can easily be represented in the third form, simply by not having the observation depend on the next state after all. The other direction is a little more challenging. Suppose that we have a model M with states S and observation function $\theta : S \times A \times S \to O$. To reduce it to a model M' with states S' and observation function $\theta' : S \times A \to O$, we want a state in M' to represent now not only the current state in M, but also the next state. However, the current state is decided before the action, so it needs to represent a function from the action to the following state. So $S' \subseteq S \times A^S$, where $(s, u) \in S'$ indicates that the current state in M is s, and the next will be $u(a)$ if a is taken. But we have to avoid $|S'| = |S| \cdot |A|^{|S|}$, to keep the reduction polynomial.

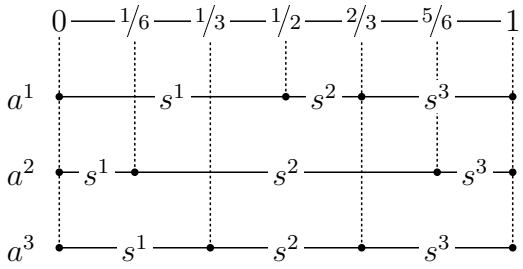

Figure 8: Efficient representation of a function from action to following state

A distribution over an ordered set can be naturally represented by a partition of $[0, 1)$ into finitely many intervals. For example, the partition $\{[0, 1/2), [1/2, 2/3), [2/3, 1)\}$ may represent the distribution in which $\Pr(1) = 1/2$, $\Pr(2) = 1/6$, and $\Pr(3) = 1/3$.

If we intersect the partitions which represent $t(s, a)$, for a fixed state s and every action $a \in A$, we get a new partition which refines each of them (Figure 8). Each interval in each original partition corresponds to a state, so each interval in the new partition corresponds to a function that matches a state to each action. For example, in Figure 8, the interval $[2/3, 5/6)$ corresponds to u such that $u(a^1) = u(a^3) = s^3$ and $u(a^2) = s^2$.

The new partition represents a distribution over a subset of these functions, of size less than $|S| \cdot |A|$. When the state s is reached in M, we reach (s, u) in M', where u is drawn according to this distribution. Then we have $u(a) \sim t(s, a)$, and $\theta'((s, u), a) = \theta(s, a, u(a))$, as required.

References

[1] Douglas Aberdeen. A (revised) survey of approximate methods for solving Partially Observable Markov Decision Processes. Technical report, National ICT Australia, December 2003.

[2] Nicholas Armstrong-Crews and Manuela Veloso. Oracular Partially Observable Markov Decision Processes: A very special case. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 2477–2482. IEEE Press, April 2007.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, June 1957.

[4] Ronen Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, October 2002.

[5] Eyal Even-Dar, Sham Kakade, and Yishay Mansour. Reinforcement learning in POMDPs without resets. In *Proc. International Joint Conf. on Artificial Intelligence*, pages 690–695. IJCAI Press, August 2005.

[6] Roy Fox and Moshe Tennenholtz. A reinforcement learning algorithm with polynomial interaction complexity for Only-Costly-Observable MDPs. In *Proc. 22nd AAAI Conf. on Artificial Intelligence*, pages 553–558. AAAI Press, 2007.

[7] Judy Goldsmith and Martin Mundhenk. Complexity issues in Markov Decision Processes. In *Proc. 13th Annual IEEE Conf. on Computational Complexity*, pages 272–280. IEEE Press, June 1998.

[8] Eric Hansen, Andrew Barto, and Shlomo Zilberstein. Reinforcement learning for mixed open-loop and closed-loop control. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 1026–1032. The MIT Press, 1997.

[9] Ronald Howard. *Dynamic Programming and Markov Processes*. Technology Press Research Monographs. Technology Press of the Massachusetts Institute of Technology and John Wiley & Sons, June 1960.

[10] Leslie Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, January–June 1996.

[11] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In Jude W. Shavlik, editor, *Proc. 15th International Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, July 1998.

[12] P. R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23(3):329–380, May 1985.

[13] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proc. 16th National Conf. on Artificial Intelligence*, pages 541–548. AAAI Press, 1999.

[14] Christos Papadimitriou and John Tsitsiklis. The complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[15] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning, Bradford Books. The MIT Press, 1998.