Instituto de Sistemas e Robótica

Pólo de Lisboa

# THE USE OF TRANSITION ENTROPY IN PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES[1]

Francisco A. Melo
M. Isabel Ribeiro
January 2005
RT-601-05

ISR Torre Norte
Av. Rovisco Pais
1049 - 001 Lisboa
Portugal

[1] Work partially supported by Programa Operacional Sociedade de Informação (POSI) in the frame of QCA III. The first author acknowledges the PhD grant SFRH/BD/3074/2000.

# The Use of Transition Entropy in Partially Observable Markov Decision Processes

Francisco A. Melo  M. Isabel Ribeiro
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-001 Lisboa,
PORTUGAL
e-mail: {fmelo,mir}@isr.ist.utl.pt

**Abstract**

In this report we describe a new POMDP algorithm, denoted TE$Q$-MDP, which computes the optimal policy of a modified MDP and uses the obtained optimal solution to compute the action for the POMDP, as a function of the belief-state. The modified MDP includes state entropy information (transition entropy) in its reward structure so as to value actions that gather information. This algorithm is suitable for real-time implementation, since the main computational burden can be done off-line.

We present the results from several tests in example-environments from the literature and compare the performance of the TE$Q$-MDP algorithm with the performance of another heuristic method ($Q$-MDP). In the various situations where $Q$-MDP presents near-optimal behaviour, the TE$Q$-MDP algorithm performs no worse. Furthermore, TE$Q$-MDP also presents near-optimal performance in particular cases where the $Q$-MDP algorithm clearly fails.

## 1 Introduction

When interacting with an environment, an agent is often faced with the situation where a decision must be taken in order to complete some task. This happens in different research fields, from classical control theory (where the agent is the controller and the environment is the controlled system) to intelligent robotics (where the agent is the robot and the environment is its physical surrounding).

If the completion of such task critically depends on the decision, this becomes a critical step in the process. In order to take the best possible action, the agent should consider all the consequences of each possible action, so as to choose properly. The "usefulness" of an action depends on two main aspects: the effects of such action on the environment, and how these effects contribute (or not) to the correct execution of the task.

In order to predict the effects of a particular action in the environment, the agent must have access to a model of the environment. However, in many situations, even the most elaborate model will not be able to predict completely the consequences of every action. As such, some models were introduced which make use of probabilities to describe the transitions on the system.

In some cases, the complete past history of the system is summarized in its current state. The systems in which this property is verified are said to have the Markov Property and are named Markov Chains[1]. This property is not an unreasonable demand, and experience shows that many system models actually verify the Markov Property. A sequential task in an environment which can be modeled as a Markov Chain, is called a Markov Decision Process (MDP)[2].

[1] For a formal definition of Markov Chains, see [CL99].
[2] For a complete review on Markov Processes, see [Mur02].

In a Markov Decision Process, the agent acts in successive time instants based in the current state of the environment in order to meet some optimality criterium. However, if the current state is unknown and the agent has available only an indirect observation of it, the elegant theory and effective algorithms developed for Markov Decision Processes are, in general, not applicable.

The control of one of such systems, where the agent has available only partial information regarding the state of the environment, is referred to as Partially Observable Markov Decision Processes (POMDP). POMDPs model a wider range of systems, in a more realistic setting than that of simple MDPs. However, this added modelling ability comes at a significant cost in complexity.

In this report, we propose a new algorithm, named as TE$Q$-MDP, that computes a heuristic POMDP policy, using a modified concept of *entropy*. The algorithm seeks to cope with the computational drawback issues regarding POMDPs presented by most of the methods proposed in the literature. This is done by using the solution of a fully observable Markov Decision Process to compute the policy in the POMDP.

As already referred, in a POMDP, the actual state of the system is not fully available to the agent and, sometimes, the best action to take seeks to determine with higher degree of accuracy the current state of the system (which is unnecessary in any MDP). In other words, sometimes the best action to take is an information-collecting action. As such, we have used a modified MDP which includes what we called *transition entropy* in its reward structure. Transition entropy assigns, in our modified MDP, a "value" to each observation which, in turn, allows the agent in the modified MDP to distinguish between two actions not only based on their reward but also based on the information they gather. The optimal solution of this modified MDP is then used to compute the action that the agent should take in each time instant. In this computation, the algorithm proposed in this report applies a procedure similar to the $Q$-MDP algorithm procedure, using the transition entropy in the modified MDP. This motivates the name of the algorithm, Transition Entropy $Q$-MDP, TE$Q$-MDP for short.

Notice that, since the MDP solution can be computed straightforwardly from the MDP parameters, it can be performed off-line. This means that, since the determination of this solution is responsible for the main computational load of the proposed algorithm, the TE$Q$-MDP algorithm is suitable for real-time implementations.

We present the results of several tests, where the proposed algorithm was used in some known literature examples, and compare its performance with other heuristic approaches also from the literature. The various examples illustrate the near-optimality of the TE$Q$-MDP algorithm in the different settings tested. In particular, the TE$Q$-MDP algorithm is shown to overcome obvious limitations presented by the $Q$-MDP algorithm, proving to provide near-optimal results in all examples tested.

There is a fair amount of work on MDPs and POMDPs. In **Section 2** we provide a brief overview of the work in this area, outlining the major contributions and drawbacks of several methods proposed in the literature. In Subsection 2.1 we present some notation used in the report.

**Section 3** provides a quick overview on Markov Decision Processes and related notation. It also briefly reviews the concepts of policy and value-function and discusses solution techniques for MDPs.

**Section 4** reviews a more general control setting for Markov Chains, the Partially Observable MDPs. The related notation is presented, and a brief discussion on POMDP solutions is conducted.

**Section 5** presents a detailed description of the approach followed in order to solve POMDPs, by exploring known solutions, and then formally introduces the new proposed TE$Q$-MDP algorithm, where the concept of transition entropy is defined and applied.

**Section 6** presents some experimental results of tests conducted on several different partially observable environments. The performance of the proposed TE$Q$-MDP algorithm is discussed.

Finally, in **Section 7**, we conclude the report by presenting the most important conclusions of our work as well as directions for future research.

# 2 Previous Work

This section intends to provide a non extensive review on MDP and POMDP literature. The POMDP literature presented covers most of the work in the area and the most significant POMDP results. We also present some references of generic review papers on MDPs and MDP solution techniques, since MDP results will play an important role in the later developments.

Markov Decision Processes have given rise to much research work, as they are the basic framework for most of the significant work on Reinforcement Learning (RL). In fact, Reinforcement Learning aims to develop algorithms for agents to experimentally "learn" the solution for a given MDP. The paper [Mur02] provides a good review on Markov Decision Processes and corresponding algorithmic solutions. For a good introduction to Reinforcement Learning and Markov Decision Processes, see the book by Barto and Sutton [SB98] or the review paper [KLM96].

If MDPs are a thoroughly studied control framework, with good known algorithmic solutions, its Partially Observable counterparts present a much more complex challenge to the research community. In fact, the introduction of partial observability leads to a significant increase in complexity, and the wide knowledge regarding MDPs brings no help to this situation.

General reviews of POMDP solution techniques can be found in [Cas94, Abe03a] and [Mur00]. Cassandra's PhD thesis, [Cas98], provides a good technical coverage of POMDPs and corresponding solution techniques.

Algorithmic solutions for POMDPs can be divided in two main categories: exact solutions, which seek to exactly determine the optimal policy, and approximate solutions, which use approximations in intermediate steps in order to simplify the computational burden.

Examples of exact solution techniques can be found in many POMDP works, such as [Cas94, Lit94, LCK95b, CLZ97]. However, exact solution methods for POMDPs generally make use of more or less complex dynamic-programming algorithms in each iteration, and have revealed to be computationally untractable for systems with more than a few dozen states.

Approximate solution methods seek to overcome this evident problem, either by using approximations in the intermediate computations or by using simplifying assumptions. These methods, although yielding solutions with sub-optimal performance or limited applicability, seek to overcome the prohibitive computational complexity presented by exact methods. Examples of some approximate methods can be found in [Abe03a, Abe03b, PR95, CKL94, MS99, GKP01, ST01].

Both the exact and the approximate methods seek to develop algorithms based on well derived mathematical foundations. However, in the literature, some more heuristic algorithms have also been developed. Some examples can be found in [CKK87, RT99, RGT05].

Finally, the adequacy of Reinforcement Learning methods to MDPs has also lead the RL community to take interest on extending RL methods to the POMDP framework. Different Reinforcement Learning approaches to POMDPs can be found in [Has02, Per02, LCK95a]. Additionally, [SJJ94] introduces a slightly modified POMDP formulation, leading to a Reinforcement Learning algorithm in [JSJ95].

## 2.1 Notation on Sets and Probabilities

In this subsection we introduce notation used in the remaining of this report.

Let $X$ be a discrete set. A probability function on $X$ is a function $P : X \longrightarrow [0, 1]$ such that, for $U \subset X, P(x \in U)$ is the probability of occurrence of the event $U$, i.e., the probability of the random variable $x$ taking a value in $U$. This probability $P(x \in U)$ will be denoted as $P(U)$. In particular, if $U = \{u\}$, this will often be denoted as $P(u)$.

Suppose, now, that $X$ is a finite set $X = \{x_1, \ldots, x_n\}$. Let $P$ be a probability function on $X$ and define the probability vector $\pi_P \in [0, 1]^n$ such that its $i$th component $\pi_P(i)$ is given by $\pi_P(i) = P(x_i)$. Since the vector $\pi_P$ allows the computation of the probability of any event $U \subset X$, it will abusively be confounded with the probability function $P$. Hence, we will write $\pi_P(x)$ or simply $\pi(x)$ when $P$ is clear from the context, to represent $P(x)$.

The set of all probability functions over a set $X$ will be denoted $\Pi(X)$. Clearly, any probability vector $\pi$ on $X$ is an element of $\Pi(X)$.

# 3 Markov Decision Processes

A Markov Decision Process (MDP) is a control framework where an intelligent agent is faced with a sequential task in an environment modeled as a Markov Chain. Since the understanding of Markov Decision Processes is of vital importance in the upcoming developments, in this section we provide a brief formal overview of such processes.

## 3.1 The Decision Process

Consider an agent interacting with an environment in order to complete some predefined task. This task will, indirectly, dictate the behavior of the agent. A parallel between this nomenclature and the control nomenclature is illustrated in Figure 1.

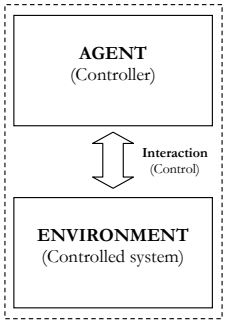



Figure 1: Parallel between the AI and the Control nomenclature.

The system[3] can be in one of a discrete number of states. The set of all possible states will be denoted as $S$. At each time instant, the agent will choose an action $a$ from a set $A$ of possible actions. Both sets $S$ and $A$ are considered finite. In this report, we will refer to state $s \in S$ as the state of the system, the state of the process, the state of the environment or the state of the agent, interchangeably.

The actions taken by the agent obviously have some effect on the state of the environment. That influence is described by a probability function $P$, where

$$P[s(t+1) = s' \mid \mathbf{a}(0, \ldots, t), \mathbf{s}(0, \ldots, t)] \tag{1}$$

denotes the probability of the system being at state $s'$ at time instant $t+1$, given that the past history of actions from time instants 0 to $t$ is $\mathbf{a}(0, \ldots, t)$ and the past history of states is $\mathbf{s}(0, \ldots, t)$.

However, in a Markov Decision Process, the environment is modeled as a *Markov Chain*, i.e., it is assumed that the probability in (1) verifies the *Markov property*,

$$P[s(t+1) = s' \mid \mathbf{a}(0, \ldots, t), \mathbf{s}(0, \ldots, t)] = P[s(t+1) = s' \mid a(t), s(t)]. \tag{2}$$

Equality (2) means that the state at time $t+1$ depends only on the state at time $t$ and on the action taken at time $t$. The probabilities in (2) are called *transition probabilities* and are denoted in a more compact way as $P(s'|s, a)$ instead of $P[s(t+1) = s' \mid s(t) = s, a(t) = a]$. Given the transition probabilities, define the transition probability function

$$\begin{aligned} &T : S \times A \times S \longrightarrow [0, 1] \\ &T(s, a, s') = P(s'|a, s). \end{aligned} \tag{3}$$

The "line of action" of the agent (i.e., the agent's choice of actions) has a purpose. In fact, every time the agent takes some action $a$ in some state $s$ it will receive a *reward*, denoted by $R(s, a)$[4].

[3] In the remainder of this report, we will refer to the pair agent-environment as *the system*.

[4] In the most generic formulation, $R(s, a)$ can be considered as being stochastic. We will work under the simplifying assumption that $R(s, a)$ is deterministic, emphasizing, however, that all remains valid if $R(s, a)$ is random.

The set of all possible rewards $R(s,a)$, is called the *reward structure* of the MDP and it somehow expresses, in terms of the parameters of the MDP, the "physical" goal of the agent.

From the point of view of the agent, the ultimate goal is usually the maximization of the *expected discounted cumulative reward*[5], defined as the expected value of the sum of all rewards over time,

$$\mathrm{E}\left[\sum_{t=0}^{\infty}\gamma^t R_t\right], \tag{4}$$

where $R_t$ is the reward received at time $t$ and $\gamma$ is a discount factor $0 \leq \gamma < 1$[6].

From all stated, a MDP can be defined as a 4-tuple $(S, A, T, R)$, where $S$ is the state-space, $A$ is the action space, $T$ represents the transition probabilities and $R$ represents the reward structure.

## 3.2 MDP Solutions

Defining a "line of action" for an agent in an MDP implies defining which action to chose in each state.

A *deterministic policy* is a function $\delta : S \longrightarrow A$ which deterministically maps the state space $S$ into the action space $A$.

Suppose that, at time $t$, the agent is at state $s_t$. What should the expected cumulative reward be, in this situation, if the agent is following some policy $\delta$? From (4), define the function $V^\delta : S \longrightarrow \mathbb{R}$ as

$$\begin{aligned} V^\delta(s_t) &= \mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i} \mid s_t\right] = \\ &= \mathrm{E}\left[R_t + \gamma\sum_{i=0}^{\infty}\gamma^i R_{t+i+1} \mid s_t\right] = \\ &= R(s_t,\delta(s_t)) + \gamma\mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i+1} \mid s_t\right], \end{aligned} \tag{5}$$

where $V^\delta(s_t)$ is the value of being in state $s_t$ when following policy $\delta$. In other words, if, at some time instant $t$, the agent is in state $s_t$ and follows the policy $\delta$ from that time on, $V^\delta(s_t)$ is the expected discounted cumulative reward of the agent.

Using the properties of the expected value operator in (5), $V^\delta(s_t)$ may be written as

$$V^\delta(s_t) = R(s_t,\delta(s_t)) + \gamma\sum_{s_{t+1}\in S} T(s_t,\delta(s_t),s_{t+1})\mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i+1} \mid s_t, s_{t+1}\right]$$

which, due to the Markov Property, becomes

$$V^\delta(s_t) = R(s_t,\delta(s_t)) + \gamma\sum_{s_{t+1}\in S} T(s_t,\delta(s_t),s_{t+1})\mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i+1} \mid s_t\right],$$

or, alternatively,

$$V^\delta(s_t) = R(s_t,\delta(s_t)) + \gamma\sum_{s_{t+1}\in S} T(s_t,\delta(s_t),s_{t+1})V^\delta(s_{t+1}). \tag{6}$$

The use of (6) in an iterative algorithm to compute $V^\delta$ for some policy $\delta$ is immediate. Such algorithm is generally called *Value Iteration* (VI). Once the values of $V^\delta$ are computed (using, for example, the VI algorithm), it is possible to improve the policy $\delta$ and define a new policy, $\delta'$, by choosing, in each state $s$, an action $a$ as

[5] In the literature it is possible to find different optimality criteria (see [SB98]).

[6] The use of the discount factor $\gamma$ can be motivated in several ways, the simpler of which is in order to make the summation in (4) converge.

$$\delta'(s) = \arg\max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\delta}(s') \right\}. \tag{7}$$

As described so far, the values of $V^{\delta}$ used in (7) are the ones obtained after the VI algorithm (using (6)) has converged. This iterative process of improving the policy $\delta$ is called *policy iteration*.

However, the policy iteration step may not occur, necessarily, after the convergence of the VI. In fact, the maximization step described by (7) may use intermediate values for $V^{\delta}$. This procedure is called *generalized policy iteration* and is defined by the following recursion in $k$.

$$V_{k+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_k(s') \right\}. \tag{8}$$

This iterative process will converge, when $k \longrightarrow \infty$ to the optimal value function $V^*$, which verifies

$$V^*(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right\}. \tag{9}$$

Equation (9) is the *Bellman optimality equation*.

Knowing the optimal value function, $V^*$, allows the assignment of a value to a *state-action* pair. In fact, if state $s$ has a value $V^*(s)$ when following the optimal policy, it is possible to define a value for the state-action pair $(s,a)$ as

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s'), \tag{10}$$

where $Q^*(s,a)$ is the expected discounted cumulative reward of being in state $s$, taking action $a$ and following the optimal policy afterwards. From the previous, it is clear that,

$$V^*(s) = \max_{a \in A} Q^*(s,a).$$

By knowing either $V^*$ or $Q^*$, it is possible to compute the optimal policy $\delta^*$ as

$$\delta^*(s) = \arg\max_{a \in A} Q^*(s,a) = \arg\max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right\}. \tag{11}$$

The computation of either the optimal policy or the optimal value function in an MDP can be programmed very efficiently using any dynamic programming package. Furthermore, it has been shown that this can be done in polynomial time (for more details on the complexity of MDP solution techniques, see [LDK95]).

For a detailed study of Markov Decision Processes and algorithmic solutions, see [Mur02, SB98, KLM96].

# 4 Partially Observable MDPs

A Partially Observable Markov Decision Process is a MDP where the agent has no direct access to the current state $s$ of the environment, but, instead, has access to an indirect observation of it. This limitation introduces significant difficulties in the process of defining what is an optimal policy in this situation, and, mainly, in its computation. These issues will be explained in greater detail in this section.

## 4.1 The Belief State

Consider a MDP defined by the 4-tuple $(S, A, T, R)$ but suppose that, at each time instant $t$, the agent has no access to the state $s$ of the environment (it does not know, with certainty, its current state). Instead, it makes an observation $x \in X$ that depends on the state $s$ and on the last action $a$ according to some probability $P[x(t) = x \mid s(t) = s, a(t-1) = a] = P(x|s,a)$. The set $X$ is the set of all possible observations and the probabilities $P(x|s,a)$ are the *observation probabilities*, also defined as a function

$$M : X \times S \times A \longrightarrow [0,1]$$
$$M(x,s,a) = P(x|s,a).$$

Therefore, a POMDP is a 6-tuple $(S, A, X, T, R, M)$, where $S, A, T$ and $R$ are the MDP parameters, $X$ is the observation space and $M$ defines the observation probabilities.

The loss of observability in a POMDP generally implies the loss of the Markov Property, since the current observation can not be related with the previous one without considering the underlying state of the environment. This is a major drawback, since it implies that a control simply based in the current observation may lead to an arbitrarily poor performance (for some examples, see [SJJ94]).

To overcome this problem, the concept of belief state is introduced. A belief state $\pi$ is a probability function over the set of all states and indicates, at each time instant, the probability of being in each of the states in $S$. This belief state, as will soon become apparent, captures all the relevant aspects of the entire previous history of the process.

Suppose that at some time instant $t$ the agent is at state $s_t$ and knows it. If, after taking action $a_t$, the agent observes $x_{t+1}$ at time $t+1$, the probability that it had moved to state $s'_{t+1}$ is given by

$$P(s'_{t+1}|s_t, a_t, x_{t+1}) = \frac{P(s'_{t+1}, x_{t+1}|s_t, a_t)}{P(x_{t+1}|s_t, a_t)} = \frac{P(x_{t+1}|s'_{t+1}, s_t, a_t)P(s'_{t+1}|s_t, a_t)}{P(x_{t+1}|s_t, a_t)},$$

which, by using the Markov Property, becomes

$$P(s'_{t+1}|s_t, a_t, x_{t+1}) = \frac{P(x_{t+1}|s'_{t+1}, a_t)P(s'_{t+1}|s_t, a_t)}{P(x_{t+1}|s_t, a_t)}. \tag{12}$$

By using the law of total probabilities in (12), this probability can be written as

$$\begin{aligned} P(s'_{t+1}|s_t, a_t, x_{t+1}) &= \frac{P(x_{t+1}|s'_{t+1}, a_t)P(s'_{t+1}|s_t, a_t)}{\sum_{s''\in S} P(x_{t+1}|s''_{t+1}, s_t, a_t)P(s''_{t+1}|s_t, a_t)} = \\ &= \frac{P(x_{t+1}|s'_{t+1}, a_t)P(s'_{t+1}|s_t, a_t)}{\sum_{s''\in S} P(x_{t+1}|s''_{t+1}, a_t)P(s''_{t+1}|s_t, a_t)}. \end{aligned} \tag{13}$$

Finally, replacing the POMDP parameters in (13) leads to

$$P(s'_{t+1}|s_t, a_t, x_{t+1}) = \frac{M(x_{t+1}, s'_{t+1}, a_t)T(s'_{t+1}, a_t, s_t)}{\sum_{s''\in S} M(x_{t+1}, s''_{t+1}, a_t)T(s''_{t+1}, a_t, s_t)}. \tag{14}$$

Notice however that, generally, the agent will not know its state $s_t$ but, instead, it knows the probability of being in $s_t$, $\pi_t(s)$. The inclusion of this probability in (14) leads to the update equation for the belief state $\pi_{t+1}$ from the value of the belief state $\pi_t$, for action $a_t$ and observation $x_{t+1}$, i.e.,

$$\pi_{t+1}(s')|_{a_t, x_{t+1}} = \frac{M(x_{t+1}, s', a_t)\sum_s \pi_t(s)T(s, a_t, s')}{\sum_{s,s''} \pi_t(s)T(s, a_t, s'')M(x_{t+1}, s'', a_t)}. \tag{15}$$

### 4.2 POMDP Solutions

As previously referred, finding the exact optimal policy for a POMDP is still a computationally untractable problem except for cases with a state space of not too large dimension.

However, it is possible to assign a value to belief states and, as in MDPs, determine a recursion similar to that of (6). Define a policy $\delta(\pi)$ as a mapping from $\Pi(S)$ to $A$. It is possible to assign a value to a belief state $\pi_t$, according to policy $\delta$ as

$$V^{\delta}(\pi_t) = \mathrm{E}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | \pi_t\right]. \tag{16}$$

As in MDPs, where $V^{\delta}(s)$ was the expected discounted cumulative reward of starting at state $s$ and following the policy $\delta$, $V^{\delta}(\pi_t)$ is the expected discounted cumulative reward of starting in belief state $\pi_t$ and following the policy $\delta$ afterwards. Notice, however, that a policy (as a function of the belief-state $\pi$) hasn't still been properly defined.

Equality (16) can be expanded by repeating the procedure used in Section 3.2, yielding the following recursion

$$V^{\delta}(\pi_t) = \sum_{s} \pi_t(s) R(s, \delta(\pi_t)) + \gamma \sum_{x,s,s'} \pi_t(s) T(s, \delta(\pi_t), s') M(x, s', \delta(\pi_t)) V^{\delta}(\pi_{t+1}). \tag{17}$$

There are two main difficulties in using (17) as a recursion, both coming from the fact that the belief space, $\Pi(S)$, is a continuous set. The first arises from the fact that the policy $\delta$ is now a function from set $\Pi(S)$, which is continuous, into $A$. This implies that its representation is much more complex that in the MDP case. The second difficulty arises from the fact that a continuous belief state implies that the computational burden necessary to iterate expression (17) is prohibitive.

In the literature, the algorithms referred to as *exact* compute the exact optimal policy for the POMDP. In order to achieve this, some of these algorithms (e.g., Incremental Pruning, see [CLZ97, Cas98]) iterate (17) by subdividing this computation in the computation of simpler, intermediate functions. Furthermore, all the exact algorithms make use of the properties of this value function, in particular its convexity (see [Cas98, Cas94]).

In spite of all the work in POMDPs, the algorithmic solutions for exactly solving POMDPs still suffer from severe limitations mainly due to the complex computation involved in any iterative process using (17).

In Section 5, a new heuristic solution method is proposed which, although still implying some computation, manages to be more efficient than known exact POMDP algorithms.

## 5 Heuristic Policies

In this section, we approach the problem of relating a POMDP solution with the solution of the underlying MDP. This will motivate the TE$Q$-MDP algorithm, which seeks to overcome some limitations of the known $Q$-MDP algorithm.

In order to do this, a simple and intuitive example is presented, which will provide a clearer understanding of the idea behind the $Q$-MDP algorithm. The main problems regarding this algorithm are then explored, and the alternative method, the TE$Q$-MDP algorithm, is presented.

### 5.1 Hitch-hiking

Consider the following hypothetical setting:

- Bob ($B$) is a hitch-hiker who is lost, somewhere in Europe. He intends to go to Portugal. He knows reasonably well the geography of Europe and the different landscapes, but he cannot read.
- Adam ($A$) is a driver going from Finland to Portugal (see Figure 2). He knows how to read.

They both want to minimize the travelling costs, by arriving in Portugal as fast as they can[7].

The problem, from Bob's point of view, can be modeled as a POMDP, since he cannot know exactly the country where he is (he cannot read the signs), but can only guess it from the appearance of the landscape, i.e., he can guess in which country he is from observing the landscape.


Figure 2: Map of a possible journey for the travelers.

From Adam's point of view, the problem can be modeled as a MDP, since he can read the signs and, hence, knows exactly in which state he is.

If, for some unnatural event, Bob comes to know that, somewhere ahead, Adam will give him a ride, what should Bob's line of action be?

Since both Bob and Adam are seeking to minimize the traveling costs, from the moment Bob rides with Adam, they will both take the best possible path (which is the same for both). This path is, obviously, the one that Adam will take.

As such, if before catching his ride, Bob believes to be in a given country (say Poland), the best he can do is move towards the country in which it would be best for Adam to give him a ride. However, when deciding, *Bob should take into account the possibility that he may not be in Poland.*

## 5.2 Optimistic Guessing

Consider that the role of Bob is played by some agent $B$, while Adam is implemented through agent $A$. Consider agent $B$ at some time instant, $T$. The belief state, at time $T$, is $\pi_T$. Agent $B$ wants to choose which action to take at time $T$ such that the expected discounted cumulative reward is maximal. This is the typical POMDP formulation.

Suppose, however, that, at time $T+1$, agent $B$ will catch a ride from agent $A$, implying that, for $t > T$, its framework turns to a typical MDP. This, in turn, means that if $A$ follows the optimal MDP policy, the value of $A$ being in state $s$ at time $T+1$ is $V^*(s)$, as seen in Section 3.

[7] Although in this formulation both travellers seek the *minimization* of a *cost*, instead of the maximization of a reward, the two formulations are dual, i.e., you can solve any of them by solving the other.

If, at the same time instant $T+1$, agent $B$ believes to be at state $s$ with probability $\pi_{T+1}(s)$, the value of the belief state will be, at time instant $T+1$,

$$V(\pi_{T+1}) = \sum_{s \in S} \pi_{T+1}(s) V^*(s). \tag{18}$$

From the point of view of agent $B$, the value of taking action $a$ at the current time $T$, denoted as $V(a|\pi_T)$, can now be computed from (18) and (10) by using the expression

$$V(a|\pi_T) = \sum_{s \in S} \pi_T(s) R(s,a) + \gamma \sum_{s,s' \in S} \sum_{x \in X} \pi_T(s) T(s,a,s') M(x,s',a) V(\pi_{T+1}), \tag{19}$$

where $\pi_{T+1}$ is computed from $\pi_T$ (using (15) with respect to action $a$ and observation $x$). It should be pointed out that (19) cannot be used as a recursion on $V$, since the value function thus obtained would be that of equation (18) which, as explained ahead and proved in Appendix A, only verifies equation (17) for an informed policy[8].

By maximizing (19), $B$ is able to take the best possible action at time $T$,

$$a_T = \arg\max_{a \in A} \left\{ \sum_{s \in S} \pi_T(s) R(s,a) + \gamma \sum_{s,s' \in S} \sum_{x \in X} \pi_T(s) T(s,a,s') M(x,s',a) V(\pi_{T+1}) \right\}. \tag{20}$$

This sort of action is based in some sort of *optimistic guessing*, since agent $B$ guesses, in a somewhat optimistic way, that the system will be completely observable from time $T$ on. In a general situation, however, this is not the case. It would be of interest to determine the performance of $B$ when deciding action $a_T$ from (20), even if the system will not become fully observable. This analysis will be conducted next.

## 5.3 $Q$-MDP and Optimistic Guessing

Consider a Markov Chain in which an agent is assigned a predefined policy $\delta$. This is not a Markov Decision Process, since the agent knows the correct "line of action" from the start. Given the desired policy $\delta$, it is straightforward to append a reward structure to a Markov Chain in order to make it an MDP with optimal policy $\delta$ (it suffices to make $R(s,a) = 1$ if $a = \delta(s)$ and 0 otherwise).

Suppose, now, that an agent is assigned a policy to execute in a Markov Chain with partial observability. By assigning the proper reward structure to the Markov Chain, is it possible to ensure that the agent will behave as desired, in the resulting POMDP?

This question, as well as the hitch-hikers example, leads to an interesting problem, regarding the relation between the optimal policy in a POMDP and the optimal policy in the underlying MDP.

A quick glance over the algorithmic solutions proposed in the literature suffices to confirm that in most of these solutions the computation of the POMDP policy makes no use of the MDP solution. Intuitively, it would make sense if the two policies were closely related, since one would expect that, in a POMDP, the agent would try to follow, as closely as possible, the optimal MDP policy. In fact, notice that if the observations allow to determine the current state of the Markov Chain unambiguously, the problem resumes to a typical MDP and, hence, the agent should follow the optimal MDP policy.

In Chapter 6 of [Cas98], some heuristic methods are described which make use of the optimal MDP value function. These methods, as argued in [Cas98], do not yield an optimal solution, since all of them disregard uncertainty in different ways. Amongst the most elaborate of these methods, the $Q$-MDP, makes use of the optimal $Q$ function (as defined in (10)) and computes a policy $\delta(\pi)$ by maximizing

$$\delta(\pi) = \arg\max_{a \in A} \left\{ \sum_{s} \pi(s) R(s,a) + \gamma \sum_{s,s'} \pi(s) T(s,a,s') V^*(s') \right\}. \tag{21}$$

[8] An informed policy (like the MDP policy), is a policy which maps the actual state $s$ of the system into the action set.

By comparing (17), (20) and (21), it is noticeable that all three equations have a similar structure. By looking carefully into the equations, it is perceivable that such similarities go beyond the structure of the equations.

In fact, some manipulation over the summations leads to the following conclusion:

**Proposition 5.1.** *Consider a POMDP $(S, A, X, T, R, M)$ and let $V^*(s)$ be the optimal value-function for the underlying MDP given by $(S, A, T, R)$. Then, the Q-MDP policy and the Optimistic Guessing policy correspond to the same policy, i.e.,*

$$\arg\max_{a\in A}\left\{\sum_s \pi(s)R(s,a) + \gamma\sum_{s,s'}\pi(s)T(s,a,s')V^*(s')\right\} =$$

$$= \arg\max_{a\in A}\left\{\sum_s \pi_T(s)R(s,a) + \gamma\sum_{s,s',x}\pi_T(s)T(s,a,s')M(x,s',a)V(\pi_{T+1})\right\}, \qquad (22)$$

*where $V(\pi_{T+1}) = \sum_s \pi_{T+1}(s)V^*(s)$.*

*Furthermore, supposing that the agent would follow the optimal MDP policy for all time instants $t > T$, the expected value of the belief state $\pi$ at time $T$ is $\sum_s \pi_T(s)V^*(s)$.*

**Proof :** See Appendix A. ■

## 5.4 Entropy

A deeper analysis of the $Q$-MDP algorithm will reveal that, in general, it will not choose an action $a$ just to collect information, as argued in [LCK95a]. If, in an MDP, it happens that the actions "look around" and "do nothing" yield the same reward and leave the state unchanged, they would have the same value under the optimal policy. However, in a POMDP, the "look around" would, in many situations, be probably preferred to the action "do nothing". In fact, the action "look around" could allow the agent to disambiguate the actual state of the system, which is preferable to "do nothing".

In order to cope with such a problem in the $Q$-MDP approach, some alternatives have been proposed which use the information that an action may, potentially, bring to the agent (see [Cas98] for a brief review).

In order to measure the difference between two actions in terms of collected information, the concept of *entropy* was introduced (see [Cas98]), which seeks to quantify the uncertainty of a given belief state $\pi$.

**Definition 5.1** (Entropy)**.** *Given a belief state $\pi$, the entropy of $\pi$, $H(\pi)$, is defined as*

$$H(\pi) = -\sum_{s\in S}\pi(s)\log(\pi(s)). \qquad (23)$$

The entropy of a belief state takes values between 0 (for a situation in which there is a state $s$ with $\pi(s) = 1$) to some maximum, when all the states are equally probable. In order to normalize the maximum value to 1, the concept of normalized entropy is derived.

**Definition 5.2** (Normalized Entropy)**.** *Given a belief state $\pi$, the normalized entropy of $\pi$, $\overline{H}(\pi)$, is defined as*

$$\overline{H}(\pi) = \frac{H(\pi)}{\log(|S|)}, \qquad (24)$$

*where $|S|$ is the number of elements of $S$.*

Entropy-based methods will yield, in some cases, results which are superior to $Q$-MDP. However, in other situations, their performance is poor when compared with the $Q$-MDP algorithm. This is because entropy-weighting methods and dual-mode algorithms assume that the point of maximum entropy corresponds to the point of minimum value, as argued in [Abe03a, Abe03b]. However, even

though the value function is, indeed, convex, the point of minimum value need not correspond to the point of maximum entropy.

We will now develop a methodology which overcomes the main problems of $Q$-MDP. In order to do so, we will use a weighted entropy method which will be described in Subsection 5.5. However, in order to overcome the inconveniences arising from blind minimum-entropy search, a new entropy measurement is needed which quantifies the entropy associated with each transition in the POMDP. The expected entropy and the expected reward arising from each transition can then be combined so that an "informed" entropy minimization criterium can be defined.

First of all, it is necessary to define the entropy associated with a single transition. This *transition entropy* must provide a measure on the entropy gain/loss associated with a particular transition. A transition will be represented by a triplet $(s, a, x)$, where $s$ is the state at some time $t$, $a$ is the corresponding action and $x$ is the observation at time $t + 1$, after the transition has occurred.

A first possibility would be to consider the agent to be at state $s$ at some time instant $t$ with probability 1. Then, by taking action $a$ and observing $x$, the belief state $\pi_{t+1}$ could easily be computed using (15) and we could define the transition entropy associated with the transition $(s, a, x)$ as the normalized entropy of the obtained belief-state, $\pi_{t+1}$. However, this measure may lead to values of entropy which do not convey the intended information.

In fact, suppose that an agent is at some state $s$ at time instant $t$ from which all actions lead to a state $s'$ and some observation $x$ with probability 1. The normalized entropy of the associated belief-state $\pi_{t+1}$ would be 0. However, if the agent does not know in which state it is (and this is the situation in which entropy information may be of use), the belief state resulting from that same transition may yield an entropy of 1! The transition entropy, as it was defined, does not distinguish between the two situations.

Suppose then that the belief state at some time instant $t$ corresponds to a uniform probability function over all states, $\pi_t(s) = \frac{1}{|S|}$, for all $s \in S$. Then, $\overline{H}(\pi_t) = 1$. If the agent takes action $a$ and observes $x$, the belief state at time $t + 1$ will be

$$\pi_{t+1}(s') = \frac{\sum_{s \in S} T(s, a, s') M(x, s', a)}{\sum_{s, s'' \in S} T(s, a, s'') M(x, s'', a)}. \tag{25}$$

Let $\overline{H}(a, x)$ be the normalized entropy of the new belief state, i.e.,

$$\begin{aligned} \overline{H}(a, x) &= \overline{H}(\pi_{t+1}) = \\ &= -\frac{1}{\log(|S|)} \frac{\sum_{s, s' \in S} T(s, a, s') M(x, s', a)}{\sum_{s, s'' \in S} T(s, a, s'') M(x, s'', a)} \log \left( \frac{\sum_{s \in S} T(s, a, s') M(x, s', a)}{\sum_{s, s'' \in S} T(s, a, s'') M(x, s'', a)} \right). \end{aligned} \tag{26}$$

The value of $\overline{H}(s, a)$ measures the mean entropy associated with a transition under action $a$ in which $x$ is observed. This happens, for each $s \in \S$, with probability $P(x \mid s, a)$ as given by

$$P(x|s, a) = \sum_{s' \in S} T(s, a, s') M(x, s', a). \tag{27}$$

It is now possible to introduce the new concept of *transition entropy.*

**Definition 5.3** (Transition Entropy)**.** *Given an action $a$ and an observation $x$, the transition entropy of action $a$ and observation $x$ in state $s$ is defined as*

$$\overline{TH}(s, a, x) = \begin{cases} 1, & \text{if } P(x|s, a) = 0, \\ \overline{H}(a, x) P(x|s, a), & \text{otherwise.} \end{cases} \tag{28}$$

Equation (28) defines an entropy measure associated with a transition represented by a state-action-observation triplet $(s, a, x)$.

One important remark is now in order. Notice that a special distinction is made for the case where $P(x \mid s, a) = 0$. In fact, if such distinction were not made, triplets $(s, a, x)$ such that

$P(x \mid s,a) = 0$ would yield minimum entropy and would, hence, be considered valuable transitions in terms of information gathering. However, $P(x \mid s,a) = 0$ implies that the triplet $(s,a,x)$ corresponds to a transition that *never occurs* and, as such, should not be taken into account in terms of information gathering.

The concept of transition entropy just introduced will play a central role in the algorithms introduced in Section 5.5, by assigning a "value" (in terms of information) to each action in each state.

## 5.5 The TE$Q$-MDP algorithm

In Definition 5.3, the concept of Transition Entropy was introduced. Transition Entropy $\overline{TH}(s,a,x)$ provides a measure of the uncertainty inherent to a transition under action $a$ followed by observation $x$, when originally in state $s$. This measure will now be introduced in the reward structure of a Markov Decision Process and the corresponding optimal policy will be used to compute the POMDP policy.

This algorithm, denoted TE$Q$-MDP, is the main contribution of this report, and corresponds to an adaptation of the $Q$-MDP algorithm.

Recall from the previous subsection that transition entropy measures the information/uncertainty resulting from a transition $(s,a,x)$. Weighted entropy methods and dual-control methods also consider the entropy of the belief-state when choosing which action to take. However, as seen in Subsection 5.4, these methods admit that the maximum-entropy and the minimum value are attained at the same point of the belief space and, as such, alternate between "blind" entropy minimization and value maximization.

In order to combine transition entropy and reward information, define the transition reward $R(s,a,x)$ as

$$R(s,a,x) = max_{a'} \sum_{s'} T(s,a,s')M(x,s',a)R(s',a'), \tag{29}$$

$R(s,a,x)$ is a myopic expected reward arising from a single transition $(s,a,x)$. This transition reward is now combined with the transition entropy $\overline{TH}(s,a,x)$ to yield

$$R_N(s,a) = \frac{1}{|X|} \sum_{x} R(s,a,x)(1 - \overline{TH}(s,a,x)). \tag{30}$$

$R_N$ combines the mean expected transition reward and the mean expected transition entropy in a new reward function $R_N$ which, by defining a new MDP $(S_N, A_N, T_N, R_N)$, where the state-space, action-space and transition function remain unchanged, i.e., $S_N = S$, $A_N = A$ and $T_N = T$, yields an optimal $Q$-function which will be denoted $Q^*_N(s,a)$.

Notice that if $\overline{TH}(s,a,x)$ measures the uncertainty in the transition $(s,a,x)$, the value of $(1-\overline{TH}(s,a,x))$ measures the information (in terms of belief-state) "gathered" in the corresponding transition. By weighting the information collected in each transition (given by $(1 - \overline{TH}(s,a,x))$) with the corresponding average transition reward (from (30)), we define a value for the information associated with each transition. By averaging this value over all measurements, we obtain the value (in terms of information) associated with action $a$ in state $s$, defined by (30).

The optimal $Q$-function for the new MDP, $Q^*_N$ is given by

$$Q^*_N(s,a) = R_N(s,a) + \gamma \sum_{s'} T(s,a,s')V^*_N(s'), \tag{31}$$

where $V^*_N(s')$ is the optimal value function for the new MDP.

Suppose, now, that at time instant $t$, the belief-state is $\pi_t$. Clearly, if the agent knows exactly in which state it is, i.e., if $\pi_t(s) = 1$ for some $s$, the agent should act as in the original MDP, since it does not need to collect information. On the other hand, if the uncertainty in the belief-state is large (suppose, for example, that $\pi(s) = 1/|S|$, for all $s$), then the agent should take an action in order to gain some information.

This leads to the action-selection procedure in the TE$Q$-MDP algorithm. Given the optimal action-value function of the original MDP, $Q^*$ and of the modified MDP, $Q_N^*$, the agent will chose action $a_H$ according to the heuristic

$$a_H = \arg\max_{a \in A} \left\{ \sum_s \pi(s) \left[ \overline{H}(\pi) Q_N^*(s,a) + (1 - \overline{H}(\pi)) Q^*(s,a) \right] \right\}. \tag{32}$$

The TE$Q$-MDP algorithm is shown in Table 1, making explicit which part of the algorithm is computed off-line and which part is computed on-line.

Table 1: TE$Q$-MDP algorithm.

| |
|---|
| `OFF-LINE` |
| `1.` Compute $Q^*(s,a)$ for $(S,A,T,R)$; |
| `2.` Compute $R(s,a,x)$ using (29) for all $s,a$ and $x$; |
| `3.` Compute $R_N(s,a)$ using (30) for all $s$ and $a$; |
| `4.` Compute $Q_N^*(s,a)$ for $(S,A,T,R_N)$. |
| |
| `ON-LINE` |
| `a.` Given $\pi_t$, compute $\overline{H}(\pi_t)$; |
| `b.` Compute $Q(s,a) = \overline{H}(\pi) Q_N^*(s,a) + (1 - \overline{H}(pi)) Q^*(s,a)$; |
| `c.` Choose action $a_H = \arg\max_a \{\sum_s \pi(s) Q(s,a)\}$; |
| `d.` Compute $\pi_{t+1}$ and return to a. |

The proposed method, which will be called *Transition Entropy Q-MDP* (TE$Q$-MDP) seeks to overcome some limitations of the $Q$-MDP algorithm. In particular, it is known that the $Q$-MDP algorithm will, in general, disregard information-collecting actions, which can lead to arbitrarily poor performance. The TE$Q$-MDP seeks to overcome that limitation by including state entropy information in the MDP reward structure.

It is evident that this comes at some cost, since the $Q$-MDP uses the solution of a $|S|$-state MDP, while TE$Q$-MDP uses the solution of two $|S|$-state MDPs. This will generally lead to a small increase in the computational effort required. However, in run-time, the TE$Q$-MDP only needs the MDP solutions to choose the POMDP action. This makes it suitable for run-time implementation, like the $Q$-MDP algorithm, since, at each time instant, it only requires the belief-state update, the computation of the corresponding entropy and its product by matrices $Q^*$ and $Q_N^*$, which are all simple operations.

In Section 6 we present the results of several experiments using examples from the POMDP literature (see [Cas98, PR95, LCK95a]).

# 6 Results

In this section we present results obtained by the TE$Q$-MDP algorithm, and compare them with those obtained with the $Q$-MDP and with the optimal behavior in the underlying MDP.

In Subsection 6.1 we briefly describe the test-environments. The results and the corresponding analysis of performance are discussed in Subsection 6.2.

## 6.1 Test environments

In this subsection we present a brief description of the several examples used in the tests.

We will not get into details such as transition probabilities, observation probabilities or rewards, except for the guessing game situation and for the large state-space map. For those technical details, see [LCK95a].

### 6.1.1 The Shuttle problem

Consider the following problem. Somewhere in space, there are two stations, which in Figure 3 are denoted as Red Station and Blue Station.

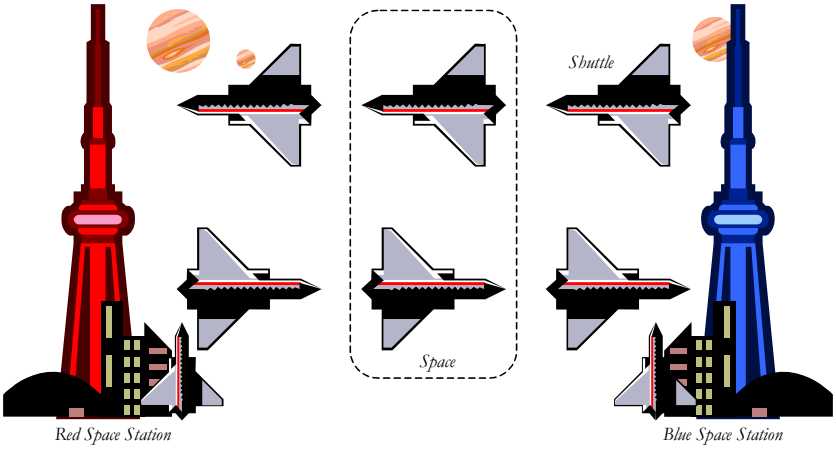



Figure 3: Representation of the Shuttle example.

A shuttle will be traveling between the two stations, supplying both of them with goods of need. Goods for the Red Station can be purchased in the Blue Station and vice-versa.

Since the shuttle is supposed to re-supply in each station before moving to the next station, the stations will be called Least Recently Visited Station (LRV) and Most Recently Visited Station (MRV). Whenever the shuttle docks at a station, it becomes the MRV Station and the other becomes the LRV station.

The goal of the shuttle is to always visit the LRV station. In order to do so, it has 3 available actions. Move forward, turn around and back-up. And, as such, there are 8 possible states (seen in Figure 3):

- Docked in MRV;
- Next to MRV, facing MRV;
- Space, facing MRV;
- Next to LRV, facing MRV;
- Next to MRV, facing LRV;
- Space, facing LRV;
- Next to LRV, facing LRV;
- Docked in LRV.

The agent will receive a reward for each successful docking in the LRV station, and a penalty for bumping into any of the stations.

### 6.1.2 The Tiger problem

This problem was first introduced in [Cas94], and it is an interesting problem, since the underlying MDP has a trivial solution. However, in the POMDP setting where it is formulated, the problem presents several interesting features, since the optimal solution will require information gathering actions.

Consider a contest where the player is facing two doors (Figure 4).

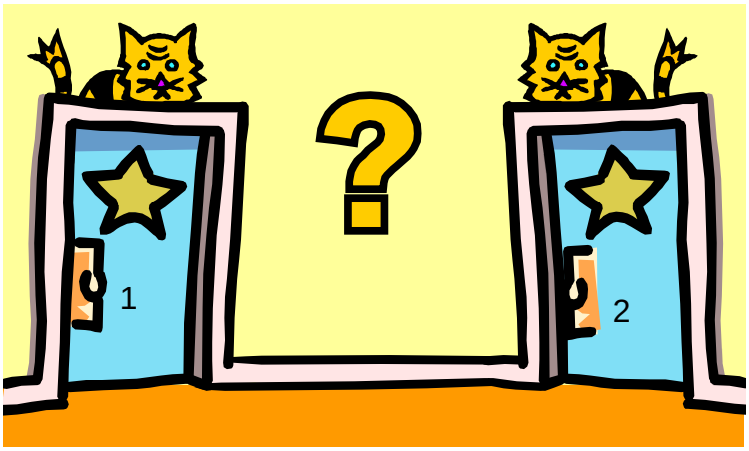



Figure 4: Representation of the Tiger example.

Behind one of these doors there is a monetary reward of 10′000 euros and behind the other there is a starving tiger, waiting to jump upon the first careless individual who dares to open such door. The player estimates that opening the door with the tiger behind will cost him 100′000 euros in hospital bills, besides all the pain and suffering.

The player is to choose between one of the two doors, knowing that he can try to listen the noise that tigers usually do. This, however, will cost him 1′000 euros each times he decides to listen (and he can do this as often as he wants). Listening behind doors is subject to errors, as the player may listen to the tiger behind the wrong door.

### 6.1.3 The Part Painting problem

Consider the supervisor in a production queue of a finishing facility, where some sort of parts are painted and shipped (see Figure 5).

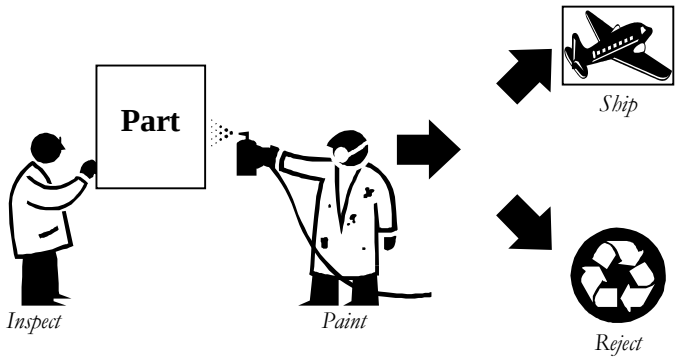



Figure 5: Representation of the Part Painting example.

Sometimes, these parts arrive from the production facility with flaws, which are often visible because of some sort of blemish appearing in the flawed parts. The parts which are flawed should be rejected, and the good ones should be painted and shipped.

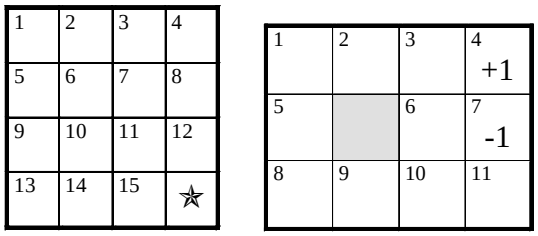
There are some nuances in this process: the company loses money if a flawed or not-painted part is shipped or if a good and painted part is rejected. However, there is the possibility that the painting process is unsuccessful or, if it is successful, the blemishes in the flawed parts will disappear.

The supervisor has, then, four possible actions: inspect the part, paint it, ship it or reject it.

Inspection will allow to determine with some degree of accuracy whether a part is good or flawed, as long as the part is not painted.

### 6.1.4 The $4 \times 4$ and $4 \times 3$ Grid problems

In these problems, represented in Figure 6, the purpose is similar, although they present slight differences in some details. The agent moves around in the environments depicted in Figure 6, trying to achieve the maximum reward.




a) b)

Figure 6: Representation of the $4 \times 4$ and $4 \times 3$ Grid examples.

The numbered cells represent the possible states. In both problems the agent has available 4 actions corresponding to the 4 directions: North, South, East and West.

In the $4 \times 4$ grid problem, the transitions corresponding to each of these actions are deterministic. The agent is then supposed to get to the goal state, marked with a star. Except for that state, no other observation is available: either the agent observes nothing or the goal. The goal rewards the agent with $+1$.

In the $4 \times 3$ grid problem, there are two marked states, one of which rewards the agent with $+1$ and the other penalizes the agent with $-1$. However, the actions are not deterministic, as in the $4 \times 4$ situation. When the agent takes an action (say North, for example), there is the possibility that it will move in an orthogonal direction (East and West, in this case). There is a gray block in the grid, which can be interpreted as an obstacle. Finally, the agent is able to detect walls on the left side and/or on the right side of each cell (for example, when in cell 5, the agent will detect a wall on the left and a wall on the right).

### 6.1.5 The Cheese Maze problem

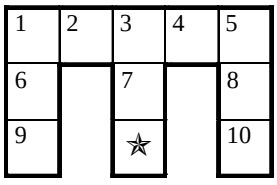



Figure 7: Representation of Cheese Maze example.

In this problem the agent is in the maze represented in Figure 7 and has available 4 different actions corresponding to movements in the 4 compass directions. The purpose of the agent is to get to the goal state, marked with a star (the cheese). In each state, there are 7 possible observations, which depend on the immediate adjacent cells (for example, in state 5, the agent will observe a cell on the west and a cell on the south).

As such, states 6, 7 and 8 all appear identical, and the same happens with states 9 and 10 or states 2 and 4. States 1, 3 and 5 are unique. Finally, there is one last observation ("Goal"), corresponding to the agent being in the goal state, i.e., the agent is able to observe the state marked with a star.

### 6.1.6 The Guessing Game problem

In this very simple problem, the agent is faced with an opponent which has one of two cards: an ace of clubs and an ace of diamonds (see Figure 8).

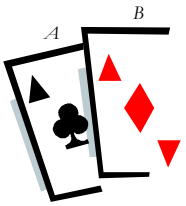



Figure 8: Representation of the Guessing Game Example.

Since the agent is blindfolded, it is not possible for the agent to know which of the two cards the opponent has. The right answer grants the agent with a $+1$ reward, and the wrong answer costs him a penalty of $-1$.

The agent has 4 actions available: guess Clubs (Cl), guess Diamonds (Dm), Think (basically, this action corresponds to doing nothing), and Peek. None of the actions Think or Peek costs or rewards the agent with anything. However, if peeking, the agent will be able to observe the chosen card.

Table 2: Transition Probabilities for Guessing Game example.

| Guess Cl or Dm | $s' = \text{Cl}$ | $s' = \text{Dm}$ |
|---|---|---|
| $s = \text{Cl}$ | 0.5 | 0.5 |
| $s = \text{Dm}$ | 0.5 | 0.5 |
| **Peek or Think** | $s' = \text{Cl}$ | $s' = \text{Dm}$ |
| $s = \text{Cl}$ | 1.0 | 0.0 |
| $s = \text{Dm}$ | 0.0 | 1.0 |

The transition probabilities for this problem can be found in Table 2 and the observation probabilities are in Table 3. In both tables, $s$ corresponds to the current state and $s'$ corresponds to the state in the following instant.

Table 3: Transition Probabilities for Guessing Game example.

| Guess Cl, Dm or Think | $x = \text{Cl}$ | $x = \text{Dm}$ | $x = \text{None}$ |
|---|---|---|---|
| $s = \text{Cl}$ | 0.0 | 0.0 | 1.0 |
| $s = \text{Dm}$ | 0.0 | 0.0 | 1.0 |
| **Peek** | $x = \text{Cl}$ | $x = \text{Dm}$ | $x = \text{None}$ |
| $s = \text{Cl}$ | 1.0 | 0.0 | 0.0 |
| $s = \text{Dm}$ | 0.0 | 1.0 | 0.0 |

The reward for every correct guess is $+1$ and for every wrong guess is $-1$. Peeking and Thinking cost 0. This problem, though trivial, will allow to illustrate a difference between the two algorithms, $Q$-MDP and TE$Q$-MDP, since, in the optimal MDP policy, both actions "Think" and "Peek" have the same value.

### 6.1.7 The 89-State Map problem

This problem was included to compare the performance of the algorithms $Q$-MDP and TE$Q$-MDP for a system with a relatively large number of states.

In this problem, the agent moves around in an office-like environment as the one in Figure 9, where the locations were discretized for the problem to have a finite number of states.

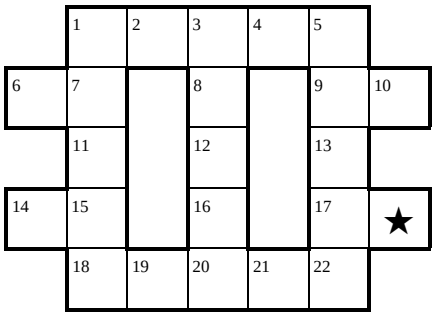



Figure 9: Representation of the 89-State Map example.

At each time instant, the agent has a limited number of available actions, namely "Move Forward", "Turn Left", "Turn Right", "Turn Back" and "Stand Still". These actions, however, only succeed sometimes. Figures 10 and 11 represent the transition probabilities for the several actions. Figure 10 represents the transition probabilities of starting in the specified location and ending in each of the depicted locations (for example, the probability of starting in the marked position and finishing in the southern cell with the opposite orientation is 0.025). Evidently, if a certain movement is not possible, the corresponding probability is added to the probability of the state remaining unchanged.

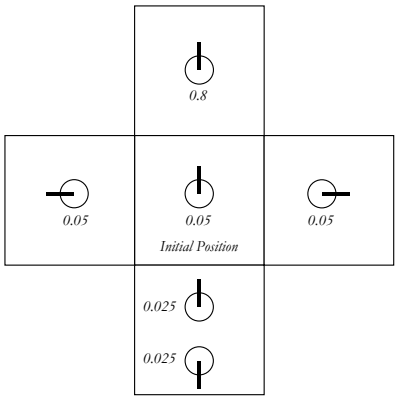



Figure 10: Transition probabilities for the "Forward" action.

The agent has 4 available sensors, allowing it to detect the presence of a wall in one of the four directions: ahead, behind, on the left and on the right. These sensors provide one of two possible observations: "wall" or "no wall".

This leads to 16 possible observations, arising from the combinations of the 4 sensors possible results ($4^2 = 16$). However, the 4 sensors are not completely reliable, since, in the presence of a wall, the sensor will determine it correctly only with probability 0.9. Furthermore, when there is no wall, the sensor will erroneously determine that there is a wall with probability 0.05. The different observation probabilities are computed from the product of the observation probabilities for each sensor, since they are considered independent.

The goal of the agent is the location marked with the symbol "★". As such, it is admitted that the agent is able to detect when it reaches the goal state. The "Goal" observation occurs with probability 1 in the goal state and probability 0 in the remaining states.

The 16 possible sensor observations and the "Goal" observation makes a total of 17 possible observations.

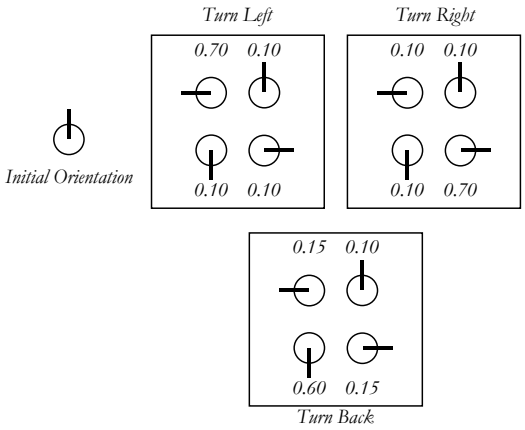



Figure 11: Transition probabilities for actions "Turn Left", "Turn Right" and "Turn Back".

### 6.1.8 The Non-Linear Value Function problem

This problem was first introduced in [PR95] and the corresponding transition diagram can be found in Figure 12.

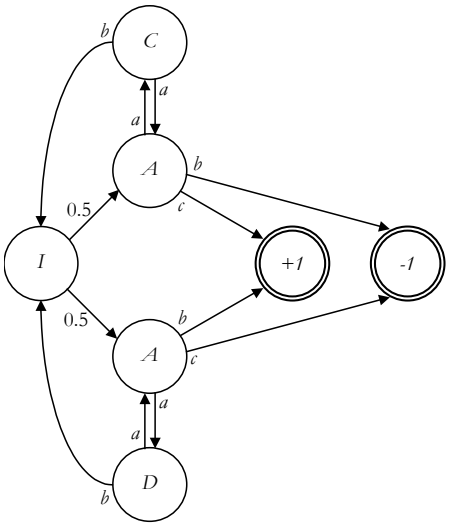



Figure 12: Representation of the Non-Linear Value Function example.

States labeled $A$ are indistinguishable. The particularity in this example arises from the fact that, although there is no specific action to collect information, action $a$ in the states labeled $A$ allows the agent to disambiguate the state and proceed with the optimal decision.

## 6.2 Experimental Results

In Subsection 6.1 a few selected test-environments from the literature were briefly described. These environments, herein named *Shuttle, Tiger, Part Painting,* $4 \times 4$ *Grid, Cheese Maze,* $4 \times 3$ *Grid, Guessing Game, 89-State Map* and *Non-Linear VF* (Value Function), will allow a comparison on the performance of the three algorithms (MDP, TE$Q$-MDP and $Q$-MDP), when applied to each of these test-environments. The dimensions of the corresponding MDPs are listed in Table 4.

In Table 5, the computation time is presented, in seconds.

Clearly, the $Q$-MDP policy uses the MDP optimal solutions and, as such, takes the same

Table 4: MDP Dimensions.

| **Environment** | $\|S\|$ | $\|X\|$ | $\|A\|$ |
|---|---|---|---|
| `Shuttle` | 8 | 5 | 3 |
| `Tiger` | 2 | 2 | 3 |
| `Part Painting` | 4 | 2 | 4 |
| $4 \times 4$ `Grid` | 16 | 2 | 4 |
| `Cheese Maze` | 11 | 7 | 4 |
| $4 \times 3$ `Grid` | 11 | 6 | 4 |
| `Guessing Game` | 2 | 3 | 4 |
| `89-State Map` | 89 | 17 | 5 |
| `Non-Linear VF` | 7 | 6 | 3 |

Table 5: Computational time for $\gamma = 0.995$. The computations to which these times refer were performed off-line.

| **Environment** | **MDP** | **TE$Q$-MDP** | **$Q$-MDP** |
|---|---|---|---|
| `Shuttle` | 0.484 s | 0.719 s | 0.484 s |
| `Tiger` | 0.281 s | 0.438 s | 0.281 s |
| `Part Painting` | 0.250 s | 0.328 s | 0.250 s |
| $4 \times 4$ `Grid` | 0.735 s | 0.922 s | 0.735 s |
| `Cheese Maze` | 0.516 s | 0.890 s | 0.516 s |
| $4 \times 3$ `Grid` | 0.562 s | 1.094 s | 0.562 s |
| `Guessing Game` | 0.172 s | 0.312 s | 0.172 s |
| `89-State Map` | 7.110 s | 20.390 s | 7.110 s |
| `Non-Linear VF` | 0.406 s | 0.609 s | 0.406 s |

computation time. The TE$Q$-MDP is, in general, computationally heavier, since it determines the intermediate rewards for the modified MDP, apart from the extra MDP solution. This leads to an increase in the computational time of, approximately, a factor of two. It should however be referred that the times in Table 5 refer to computations performed off-line.

In Table 6 we present the discounted cumulative reward (DCR) obtained by the different algorithms in a 100 time-steps trial with the results averaged over 1000 Monte-Carlo runs. In the experiments of Table 6, a value of 0.95 was used for the parameter $\gamma$. In Table 7 we present the results for the same experiments with $\gamma = 0.995$. The bold-marked lines will be commented in detail further ahead.

Table 6: DCR for $\gamma = 0.95$. The values correspond to the average of 1000 Monte-Carlo runs of 100 time instants.

| **Environment** | **MDP** | **TE$Q$-MDP** | **$Q$-MDP** |
|---|---|---|---|
| `Shuttle` | $46.710 \pm 6.708$ | $46.773 \pm 6.608$ | $46.441 \pm 6.399$ |
| `Tiger` | $198.816 \pm 0.0$ | $20.878 \pm 27.480$ | $19.035 \pm 30.577$ |
| `Part Painting` | $13.026 \pm 0.990$ | $3.453 \pm 1.813$ | $3.423 \pm 1.800$ |
| $4 \times 4$ `Grid` | $3.815 \pm 0.383$ | $3.184 \pm 0.314$ | $3.192 \pm 0.320$ |
| `Cheese Maze` | $3.670 \pm 0.389$ | $3.249 \pm 0.257$ | $3.257 \pm 0.261$ |
| $4 \times 3$ `Grid` | $3.063 \pm 0.726$ | $2.085 \pm 0.556$ | $2.160 \pm 0.574$ |
| `Guessing Game` | $\mathbf{19.882 \pm 0.0}$ | $\mathbf{9.686 \pm 0.0}$ | $\mathbf{1.090 \pm 3.069}$ |
| `89-State Map` | $\mathbf{8.669 \pm 1.801}$ | $\mathbf{4.352 \pm 3.776}$ | $\mathbf{0.184 \pm 1.013}$ |
| `Non-Linear VF` | $\mathbf{13.239 \pm 0.0}$ | $\mathbf{7.535 \pm 0.0}$ | $\mathbf{6.687 \pm 1.810}$ |

Several interesting aspects should be referred at this stage. First of all, in the first 6 environments, the TE$Q$-MDP has a behaviour which is similar to the $Q$-MDP. In fact, although some slight variations may be observed, they are not significant and are due to the random aspects

Table 7: DCR for $\gamma = 0.995$. The values correspond to the average of 1000 Monte-Carlo runs of 100 time instants.

| **Environment** | **MDP** | **TE$Q$-MDP** | **$Q$-MDP** |
|---|---|---|---|
| `Shuttle` | $201.956 \pm 19.197$ | $201.609 \pm 19.449$ | $201.585 \pm 19.508$ |
| `Tiger` | $788.459 \pm 0.0$ | $85.713 \pm 73.590$ | $81.352 \pm 81.002$ |
| `Part Painting` | $50.996 \pm 2.530$ | $11.851 \pm 2.726$ | $13.407 \pm 4.622$ |
| $4 \times 4$ `Grid` | $17.769 \pm 1.274$ | $14.689 \pm 1.044$ | $14.595 \pm 1.081$ |
| `Cheese Maze` | $15.778 \pm 1.115$ | $14.381 \pm 0.803$ | $14.357 \pm 0.807$ |
| $4 \times 3$ `Grid` | $12.611 \pm 2.108$ | $9.678 \pm 1.733$ | $9.744 \pm 1.850$ |
| `Guessing Game` | $\mathbf{78.846 \pm 0.0}$ | $\mathbf{39.324 \pm 0.0}$ | $\mathbf{1.130 \pm 7.809}$ |
| `89-State Map` | $\mathbf{62.832 \pm 4.157}$ | $\mathbf{35.747 \pm 28.853}$ | $\mathbf{0.884 \pm 6.970}$ |
| `Non-Linear VF` | $\mathbf{52.158 \pm 0.0}$ | $\mathbf{31.380 \pm 0.0}$ | $\mathbf{26.134 \pm 4.672}$ |

involved in the experiments.

In [Cas98, LCK95a], the performance of the $Q$-MDP algorithm has been proved to be near optimal for these first six environments, which leads to the conclusion that, in such situations, the TE$Q$-MDP is a suitable alternative.

Also the comparison of the results achieved by the policies obtained using the TE$Q$-MDP and the $Q$-MDP algorithms with the results obtained by the optimal MDP policy shows that, in some of the previous problems, partial observability does not pose serious difficulties in terms of control, since the observations available to the agent allow him to follow a quasi-optimal policy. This conclusion comes from the fact that both the $Q$-MDP and the TE$Q$-MDP present rewards which are not too different from the reward obtained by the optimal MDP agent. This is clear in the Shuttle problem, the Cheese Maze problem and in both Grid problems.

In the Shuttle and Maze problems this is easily understandable, since the observations allow to disambiguate many of the states of the underlying MDP (see Section 6.1 for details on the problems). On the other hand, in both Grid problems, although the observations are much less informative, the optimal policies are very simple and easy to follow even in the absence of state information.

On the other hand, by comparing the results obtained by the POMDP agents with the ones obtained by the MDP agent in the Tiger and Part Painting examples, the difference in the rewards is quite noticeable. However, this makes sense, since the POMDP policies use actions like "Listen" in the Tiger example and "Inspect" in the Part Painting example, in order to disambiguate the state of the system and choose the action in a more informed way. However, this delays the POMDP agents and hence the differences in reward between the POMDP agents and the MDP agent.

In Figures 13, 14 and 15, the average policies of the different algorithms are presented.

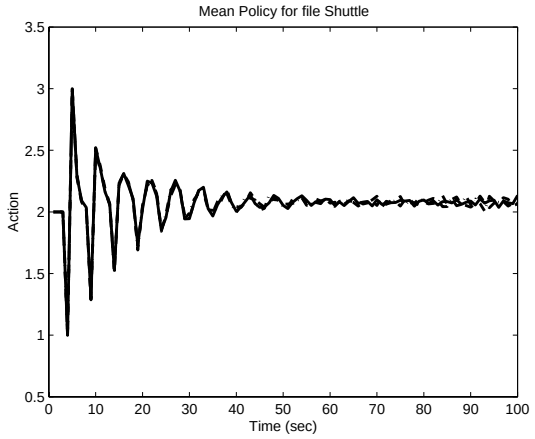
The average policies depicted in Figures 13 and 14 confirm that, in fact, in the situations where the POMDP methods present near-optimal performance (such as the Shuttle, Maze and both Grid problems), the average policies of the POMDP agents follow closely the average MDP policy (see Figure 13.a, 14.a, 14.b and 14.c).

Moreover, in the Tiger and Part Painting problems (see Figures 13.b and 13.c), the POMDP policies present clear differences from the optimal MDP policy. This fact has already been explained, and such differences are justified by the use of the actions "Listen" and "Inspect" in the POMDP setting and not in the MDP setting.

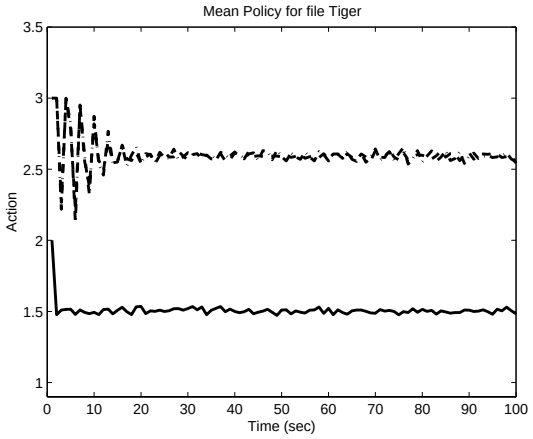
However, how can we explain the fact that, in these two situations, although actions such as "Listen" and "Inspect" are of no use in the MDP setting, they are still used by the $Q$-MDP agent?

To understand this, notice that both the Painting and the Tiger problems present actions with the specific purpose of disambiguating the state. Although for the MDP agent these actions are not necessary, in these problems, the loss for the POMDP agents making a wrong decision is somewhat larger than the gain for making a good decision. It is expected, then, that when facing an ambiguous belief-state, both the TE$Q$-MDP and the $Q$-MDP will choose a more neutral action (an action with 0-gain).

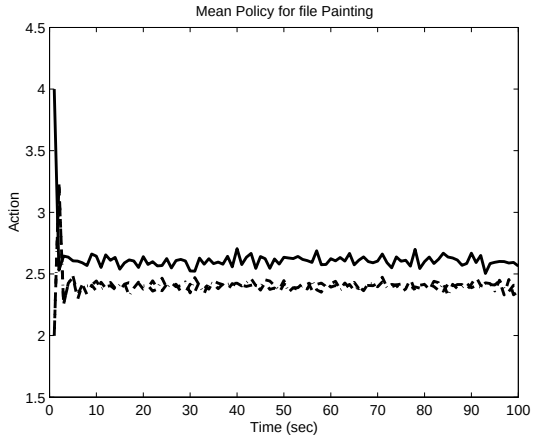
If in some other problem the right and the wrong actions lead to equal gain and loss, the $Q$-




a) Shuttle example.




b) Tiger example.




c) Part Painting example.

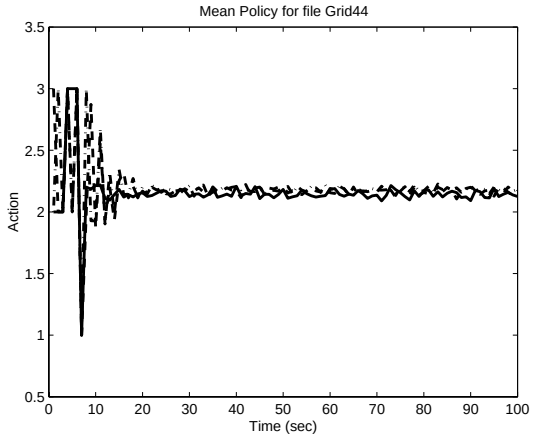
Figure 13: Average Policies for three of the test-environments with $\gamma = 0.95$. The solid line corresponds to the MDP optimal policy, dotted line corresponds to the TE$Q$-MDP optimal policy and the dashed line represents the $Q$-MDP optimal policy.

MDP algorithm will choose between the two possibilities randomly, instead of choosing a neutral, information-gathering action. This is extremely evident in the Guessing Game problem. In fact, in Figure 15.a, one can observe that the $Q$-MDP agent chose action 1 every single time instant in the run! This leads to an arbitrarily poor performance, as can be easily checked in Tables 6 and 7.

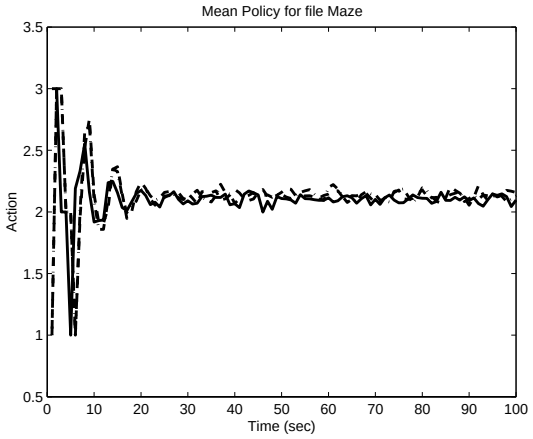
However, this is not the single situation where the $Q$-MDP fails. In the Non-Linear VF problem, although no action has the precise purpose of disambiguating the state (as happens in the Tiger, Part Painting and Game examples), there is a choice of actions which allows the agent to disambiguate its state.

By observing the average policy for the $Q$-MDP algorithm in Figure 15.c, it becomes evident that, when facing observation $A$, the agent will choose action $b$, which will yield an expected reward of 0 ($+1$ and $-1$ with probability 0.5 each). It will not choose to take action $a$ which would allow it to disambiguate the state. As a consequence, its reward will be, approximately, half of the reward of the MDP agent (as can easily be checked in Tables 6 and 7). The TE$Q$-MDP agent, however, chooses action $a$ to disambiguate the state and, as such, is able to achieve a higher reward.

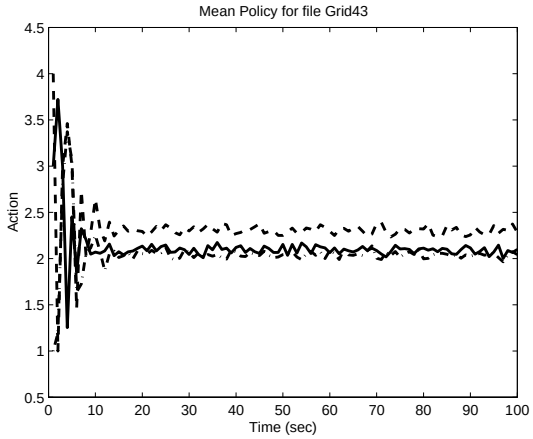
The 89-State Map problem is aimed at testing the performance of the algorithms in a larger state-space POMDP. Since, in this problem, the purpose of the agent is reaching the goal, a different measurement of performance will be used, that accounts for the number of runs in which the agent was able to reach the goal. These results will provide, in this particular case, a more intuitive understanding of the performance of the algorithms. The percentage of success (goal-achievement)




a) $4 \times 4$ Grid example.




b) Cheese Maze example.




c) $4 \times 3$ Grid example.

Figure 14: Average Policies for three other test-environments with $\gamma = 0.95$. The solid line corresponds to the MDP optimal policy, dotted line corresponds to the TE$Q$-MDP optimal policy and the dashed line represents the $Q$-MDP optimal policy.

in this problem are summarized in table 8.

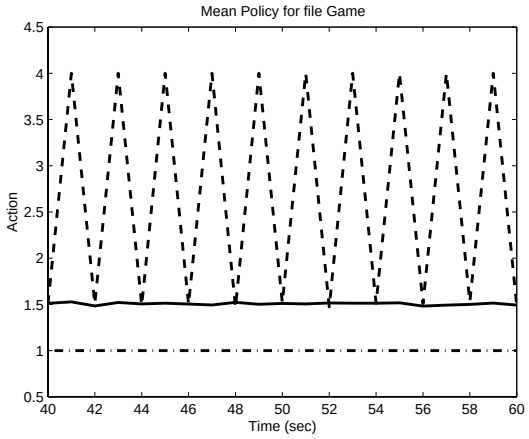
Table 8: Goal achievement percentage for the 89-State Map.

| **Method** | $\gamma = 0.95$ | $\gamma = 0.995$ |
|---|---|---|
| MDP | 100% | 100% |
| TE$Q$-MDP | 63.7% | 61.4% |
| $Q$-MDP | 3.9% | 1.6% |

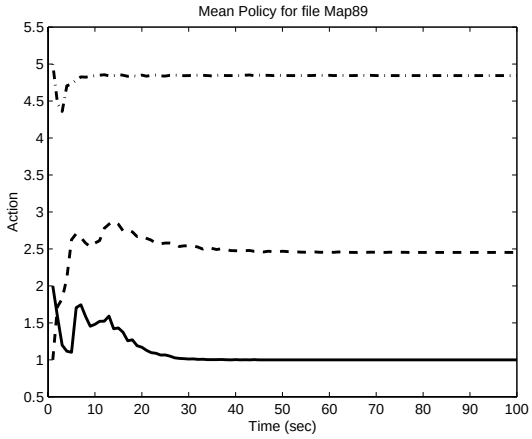
By looking at Table 8, the superiority of the TE$Q$-MDP algorithm is overwhelming. In fact, in this problem there is a large number of states which will generally yield similar observations. This will often lead to a situation where the agent will be completelly lost. The $Q$-MDP agent won't be able to choose an action in order to progress in the maze and, as such, will choose to do nothing. This becomes evident in Figure 15.b, where the $Q$-MDP agent will quickly be "stuck" repeating action "do nothing" (action 5) in the lack of anything better to do.

The results show that such problem doesn't occur with the TE$Q$-MDP agent, which clearly outperforms the $Q$-MDP agent.

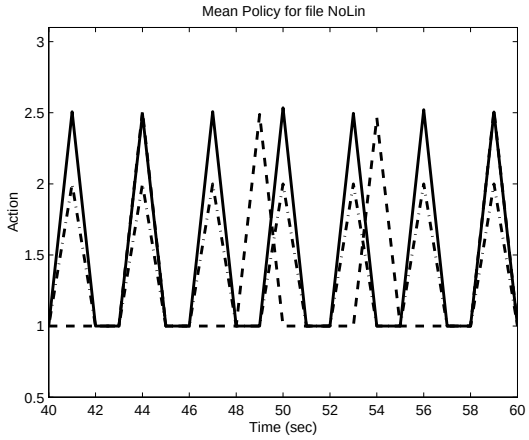
One last remark should be made regarding the behaviour of the TE$Q$-MDP agent. Notice that the MDP agent chooses, mainly, the "Move Forward" action (action 1). The TE$Q$-MDP agent, in




a) Guessing Game example.




b) 89-State Map example.




c) Non-Linear VF example.

Figure 15: Average Policies for the last three test-environments with $\gamma = 0.95$. The solid line corresponds to the MDP optimal policy, dotted line corresponds to the TE$Q$-MDP optimal policy and the dashed line represents the $Q$-MDP optimal policy.

order to be able to disambiguate its current state, chooses mainly "Turn" actions (actions 2, 3 and 4), which allow it to determine, more accuratelly, its current state. This could be interpreted as the TE$Q$-MDP agent being "looking around".

# 7 Conclusions and Future Work

In this report we described a new POMDP algorithm, named as Transition Entropy $Q$-MDP (TE$Q$-MDP). This algorithm computes the optimal policy of a modified MDP and uses the obtained optimal solution to compute the action for the POMDP, as a function of the belief-state. This modified MDP incorporates state entropy information in its reward structure, making use of the concept of *transition entropy*, also introduced in this report.

Although the modified MDP has a larger state-space, which requires additional effort to the computation of the policies, it still proves to be suitable for real-time implementation, since the main computational burden (arising from the computation of the optimal MDP solution) can be done off-line.

The algorithm was tested in several examples from the literature and its performance compared with the original $Q$-MDP algorithm. In situations where the $Q$-MDP algorithm is known to presente near-optimal performance, the performance of the TE$Q$-MDP was not worse. Further-

more, TE$Q$-MDP proved to find the optimal solution in particular situations where the $Q$-MDP algorithm clearly failed (such as the Guessing Game, the 89-State Map and the Non-Linear Value Function problems).

Future work includes the clarification of the relation between the optimal POMDP solution and the solution determined by the TE$Q$-MDP. Adittionally, there is the need for some effort in order to test the TE$Q$-MDP algorithm in even larger problems, to understand its exact range of applicability and perceive if it actually constitutes a universal alternative to the computationally untractable exact solution methods.

## Acknowledgements

The authors acknowledge Prof. Pedro Lima for valuable discussions and suggestions.

This work was partially supported by Programa Operacional Sociedade de Informação (POSI) in the frame of QCA III. The first author acknowledges the PhD grant SFRH/BD/3074/2000.

## References

[Abe03a] Douglas A. Aberdeen. A (Revised) Survey of Approximate Methods for Solving Partially Observable Markov Decision Processes. Technical report, National ICT Australia, Canberra, Australia, 2003.

[Abe03b] Douglas A. Aberdeen. *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Australian National University, April 2003.

[Cas94] Anthony R. Cassandra. Optimal Policies for Partially Observable Markov Decision Processes. Technical Report CS-94-14, Department of Computer Sciences, Brown University, August 1994.

[Cas98] Anthony R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, 1998.

[CKK87] Anthony R. Cassandra, Leslie Kaelbling, and James A. Kurien. Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[CKL94] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting Optimally in Partially Observable Stochastic Domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[CL99] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[CLZ97] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental Prunning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 54–61, Providence, Rhode Island, 1997. Morgan Kaufmann Publishers.

[GKP01] Carlos Guestrin, Daphne Koller, and Ronald Parr. Solving Factored POMDPs with Linear Value Functions. *IJCAI-01 Workshop on Planning under Uncertainty and Incomplete Information, Seattle, Washington*, pages 67–75, August 2001.

[Has02] Samuel W. Hasinoff. Reinforcement Learning for Problems with Hidden State. Technical report, University of Toronto, Department of Computer Science, 2002.

[JSJ95] Tommi Jaakkola, Satinder Singh, and Michael Jordan. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. *NIPS*, 7, 1995.

[KLM96] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[LCK95a] Michael Littman, Anthony Cassandra, and Leslie Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Armand Prieditis and Stuart Russell, editors.

[LCK95b] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Efficient Dynamic-Programming Updates in Partially Observable Markov Decision Processes. Technical Report CS-95-19, Department of Computer Sciences, Brown University, 1995.

[LDK95] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the Complexity of Solving Markov Decision Problems. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 394–402, New York, NY, 1995. Elsevier Science Publishing Company, Inc.

[Lit94] Michael L. Littman. The Witness Algorithm: Solving Partially Observable Markov Decision Processes. Technical Report CS-94-40, Department of Computer Sciences, Brown University, December 1994.

[MS99] David McAllester and Satinder Singh. Approximate Planning for Factored POMDPs using Belief State Simplification. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 409–416, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[Mur00] Kevin P. Murphy. A Survey of POMDP Solution Techniques. Technical report, Department of Computer Science, University of California, 2000.

[Mur02] Kevin P. Murphy. Learning Markov Processes. In *Encyclopedia of Cognitive Science*. L. Nadel et al. (eds), Nature Macmillan, 2002.

[Per02] T. J. Perkins. Reinforcement Learning for POMDPs based on Action Values and Stochastic Optimization. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 199–204, Menlo Park, CA, 2002. AAAI Press/MIT Press.

[PR95] Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1088–1094, 1995.

[RGT05] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding Approximate POMDP Solutions Through Belief Compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

[RT99] Nicholas Roy and Sebastian Thrun. Coastal Navigation with Mobile Robot. In *Proceedings of 1999 Conference on Neural Information Processing Systems (NIPS'99)*, pages 1043–1049, 1999.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, third edition, 1998.

[SJJ94] Satinder Singh, Tommi Jaakkola, and Michael Jordan. Learning Without State-Estimation in Partially Observable Markovian Decision Processes. *Machine Learning: Proceedings of the Eleventh International Conference*, pages 284–292, 1994.

[ST01] Jamieson Schulte and Sebastian Thrun. A Heuristic Search Algorithm for Acting Optimally in Markov Decision Processes with Deterministic Hidden State. 2001.

# A $Q$-MDP and Optimistic Guessing Equivalence

In this appendix, we present the proof of Proposition 5.1 presented in Section 5.3 which, because of its length, was left aside for the sake of clarity.

**Proposition** (5.1)**.** *Consider a POMDP $(S, A, X, T, R, M)$ and let $V^*(s)$ be the optimal value-function for the underlying MDP given by $(S, A, T, R)$. Then, the Q-MDP policy and the Optimistic Guessing policy correspond to the same policy, i.e.,*

$$\arg\max_{a\in A}\left\{\sum_s \pi(s)R(s,a) + \gamma\sum_{s,s'}\pi(s)T(s,a,s')V^*(s')\right\} =$$

$$= \arg\max_{a\in A}\left\{\sum_s \pi_T(s)R(s,a) + \gamma\sum_{s,s',x}\pi_T(s)T(s,a,s')M(x,s',a)V(\pi_{T+1})\right\}, \qquad (33)$$

*where $V(\pi_{T+1}) = \sum_s \pi_{T+1}(s)V^*(s)$.*

*Furthermore, supposing that the agent would follow the optimal MDP policy for all time instants $t > T$, the expected value of the belief state $\pi$ at time $T$ is $\sum_s \pi_T(s)V^*(s)$.*

**Proof :** As seen from Section 5, an Optimistic Guessing agent would choose action $a^*$ so as to maximize

$$\sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s,s',x}\pi_t(s)T(s,a,s')M(x,s',a)V(\pi_{t+1}), \qquad (34)$$

as seen from (20). On the other hand, an agent following a $Q$-MDP policy would choose an action $a^*$ so as to maximize

$$\sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s,s'}\pi_t(s)T(s,a,s')V^*(s'), \qquad (35)$$

where $V^*(s')$ is the optimal MDP value function, as seen from (21).

Recall that, in (34), $V$ is defined as $V(\pi_{t+1}) = \sum_s' \pi_{t+1}(s)V^*(s)$ which, replaced in (34) yields

$$\sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s,s',s'',x}\pi_t(s)T(s,a,s')M(x,s',a)\pi_{t+1}(s'')V^*(s''). \qquad (36)$$

In Section 4, the belief state update equation was introduced in (15), which is repeated in (37).

$$\pi_{t+1}(s')|_{a,x} = \frac{M(x,s',a)\sum_s \pi_t(s)T(s,a,s')}{\sum_{s,s''}\pi_t(s)T(s,a,s'')M(x,s'',a)}. \qquad (37)$$

By replacing (37) in (36), we finally obtain

$$\begin{aligned}
&\sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s,s',s_2,x}\pi_t(s)T(s,a,s')M(x,s',a)\pi_{t+1}(s_2)V^*(s_2) = \\
&= \sum_{s\in S}\pi_t(s)R(s,a) + \\
&\quad +\gamma\frac{\sum_{s,s',x}\pi_t(s)T(s,a,s')M(x,s',a)}{\sum_{s_1,s_2}\pi_t(s_1)T(s_1,a,s_2)M(x,s_2,a)}\sum_{s_1,s_2}\pi_t(s_1)T(s_1,a,s_2)M(x,s_2,a)V^*(s_2) = \\
&= \sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s_1,s_2,x}\pi_t(s_1)T(s_1,a,s_2)M(x,s_2,a)V^*(s_2) = \\
&= \sum_{s\in S}\pi_t(s)R(s,a) + \gamma\sum_{s_1,s_2}\pi_t(s_1)T(s_1,a,s_2)V^*(s_2),
\end{aligned}$$

which demonstrates that Q-MDP and Optimistic Guessing maximize the same action and, as such, are equivalent. Furthermore, if the action that maximizes (34) (and (35)) happens to be the

optimal MDP action,

$$\begin{aligned}
&\sum_{s\in S}\pi_t(s)R(s,a)+\gamma\sum_{s,s'}\pi_t(s)T(s,a,s')V^*(s')= \\
&=\sum_{s\in S}\pi_t(s)\left\{R(s,a)+\gamma\sum_{s'}T(s,a,s')V^*(s')\right\}= \\
&=\sum_{s\in S}\pi_t(s)V^*(s). \qquad (38)
\end{aligned}$$

■