Asynchronous Execution in Multiagent POMDPs: Reasoning over Partially-Observable Events

João V. Messias
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
jmessias@isr.ist.utl.pt

Matthijs T. J. Spaan
Delft University of Technology
Delft, The Netherlands
m.t.j.spaan@tudelft.nl

Pedro U. Lima
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
pal@isr.ist.utl.pt

ABSTRACT

This paper proposes a novel modeling approach to problems of multiagent decision-making under partial observability, building on the framework of Multiagent Partial Observable Markov Decision Processes (MPOMDPs). Unfortunately, the size of MPOMDP models (and their solutions) grows exponentially in the number of agents, and agents are required to act in synchrony. In the present work, we show how these problems can be mitigated through an event-driven, asynchronous formulation of the MPOMDP dynamics.

We introduce the necessary extensions to the dynamics and solution algorithms of standard MPOMDPs. In particular, we prove that the optimal value function in our Event-Driven Multiagent POMDP framework is piecewise linear and convex, allowing us to extend a standard point-based solver to the event-driven setting. Finally, we present simulation results, showing the computational savings of our modeling approach.

INTRODUCTION

Many multiagent applications take place in stochastic domains, in which the interaction of each agent with its environment carries a measure of uncertainty with respect to its outcome. In particular, the information that each agent can extract from that environment may be limited or noisy. Examples can be found in sensor networks [7], cooperative robotics [15], and distributed manufacturing systems [12].

Most existing approaches to multiagent decision making under uncertainty are grounded in the theory of Markov Decision Processes (MDPs), for example Decentralized Partially Observable MDPs (Dec-POMDPs) [2] which are intractable without communication (NEXP-Complete); and Multiagent MDPs (MMDPs) / POMDPs (MPOMDPs), in which free communication between agents is assumed [3, 11]. In the latter case, the planning problem is of equivalent complexity to that of solving a centralized single-agent problem defined over the whole team [11]. We focus on situations in which it is reasonable to assume that agents can communicate freely, but the system is partially observable, even given the observations of all agents. This is typical, for example, when dealing with teams of mobile robots, or in autonomous surveillance systems. We will therefore rely on the MPOMDP paradigm.

Modeling a multiagent problem as an MPOMDP carries a notable drawback: the complexity of solving an MPOMDP model, even approximately, is exponential in the number of agents. This undesirable property, however, follows from the implicit assumption that agents are performing actions and observing their outcomes *in synchrony*. That is to say, at each decision step, each agent is expected to simultaneously perform individual local actions, and to receive simultaneous individual observation symbols.

The present work proposes Event-Driven MPOMDPs, an alternative description of the dynamics of multiagent decision making under uncertainty, based on the operation of real-time discrete-event systems [5][1]. In our approach, agents must react to events, which are detected locally, and *asynchronously*, by each agent. Through the assumption of free communication, each local event triggers a joint observation, which is shared by the team. Since multiple events cannot occur simultaneously, this means that the total number of joint observations in this model grows *linearly* in the number of agents (instead of exponentially), allowing these methods to scale better to larger scenarios, while retaining MPOMDP functionality.

Furthermore, the processes through which events are detected are considered to be susceptible to error. An agent may signal the detection of an event which did not actually take place (a false positive), or fail to detect a real event (a false negative). It will be shown that the latter case, in particular, implies minimal modifications with respect to the dynamics of a standard POMDP. We prove that the optimal value function is piecewise linear and convex, allowing us to extend a point-based solver to the event-driven setting. Lastly, we consolidate the proposed methods through simulated results, comparing the performance of our event-driven models to that of equivalent synchronous MPOMDPs.

BACKGROUND

This section introduces the necessary background regarding MPOMDPs and their solution.

Definition 1. *A Multiagent Partially Observable Markov Decision Process (MPOMDP) is a tuple* $\langle d, \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R \rangle$, *where:*

[1]Our definition and use of "events" differs from existing work [1], and concerns different purposes. There, events model interdependencies between agent policies in Dec-MDPs. Here, events are simply state changes: the system dynamics are driven by events.

Appears in *The Eighth Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty* (*MSDM-2013*), held in conjunction with *AAMAS*, May 2013, St. Paul, Minnesota, USA.

d is the number of agents;

$\mathcal{S} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_k$ is the state space*, a discrete set of possibilities for the state s of the process. The (joint) state $s \in \mathcal{S}$ is a tuple of* state factor *assignments: $s = \langle x_1, x_2, \ldots, x_k \rangle$, where $x_i \in \mathcal{X}_i$;*

$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_d$ is a set of joint *actions. Each joint action $\mathbf{a} \in \mathcal{A}$ is a tuple of individual actions: $\mathbf{a} = \langle a_1, a_2, \ldots, a_d \rangle$, where $a_i \in \mathcal{A}_i$;*

$\mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2 \times \ldots \times \mathcal{O}_d$ is the space of joint observations $\mathbf{o} = \langle o_1, ..., o_d \rangle$, where $o_i \in \mathcal{O}_i$ are the individual observations of each agent.

$T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, such that $T(s, \mathbf{a}, s') = \Pr(s'|s, \mathbf{a})$;

$O : \mathcal{A} \times \mathcal{S} \times \mathcal{O} \to [0, 1]$ is the observation function, such that $O(\mathbf{a}, s', \mathbf{o}) = \Pr(\mathbf{o}|\, \mathbf{a}, s')$;

$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the instantaneous reward function*. The value $R(s, \mathbf{a})$ represents the "reward" that the team of agents receives for performing joint action $\mathbf{a}$ in state s;*

The state of the system cannot be observed directly in an (M)POMDP, and therefore agents only have access to probability distributions over the state space, also known as belief states, $b \in \Pi(\mathcal{S})$. The belief state at time step t, $b|_t$, can be computed using Bayes' rule given the initial belief $b|_0$ and the sequence of actions taken and observations received. The goal of the planning problem is to maximize the expected discounted reward of a system over a given number of decisions (or planning horizon) h. Specifically, we seek to obtain a sequence of decision rules $\pi = \{\pi_{h-1}, \ldots, \pi_0\}$, where $\pi_i : \Pi(\mathcal{S}) \to \mathcal{A}$, which maximizes the quantity:

$$V^{\pi}_{h-1}(b) = E\left[\sum_{t=0}^{h-1} \gamma^t \sum_{s \in \mathcal{S}} b|_t(s) R(s, \pi_{h-1-t}(b|_t)) \,\middle|\, b = b|_0 \right], \tag{1}$$

where $\gamma \in [0, 1]$ is a specified discount factor. The function $V^{\pi}_{h-1}(b)$ is also known as a *value function* for horizon h, under the policy π. We will restrict our attention to optimal value functions, $V^*_n(b)$, for $n = h-1, \ldots, 0$. It is well known that POMDP value functions are piecewise linear and convex (PWLC) for $h < \infty$, and can be arbitrarily well approximated by a PWLC function if $h = \infty$ [13]. Therefore, they admit a compact representation:

$$V^*_n(b) = \max_{\upsilon_n \in \Upsilon_n} \upsilon_n \cdot b \quad , \tag{2}$$

where Υ_n is a set of $|\mathcal{S}|$-dimensional vectors. A related concept is that of Q-value functions: $V^*_n = \max_{\mathbf{a} \in \mathcal{A}} Q^*_n(b, \mathbf{a})$. These functions represent the expected value of performing joint action $\mathbf{a}$ when there are n remaining decisions, and following the optimal policy afterwards [10].

EVENT-DRIVEN MPOMDPS

In this section we propose a novel modeling approach to multiagent decision making under partial observability. We overview the multiagent dynamics of typical MPOMDPs, how they affect the potential applications of the framework, and how these limitations are addressed through an event-driven perspective. We then formalize our proposed model.

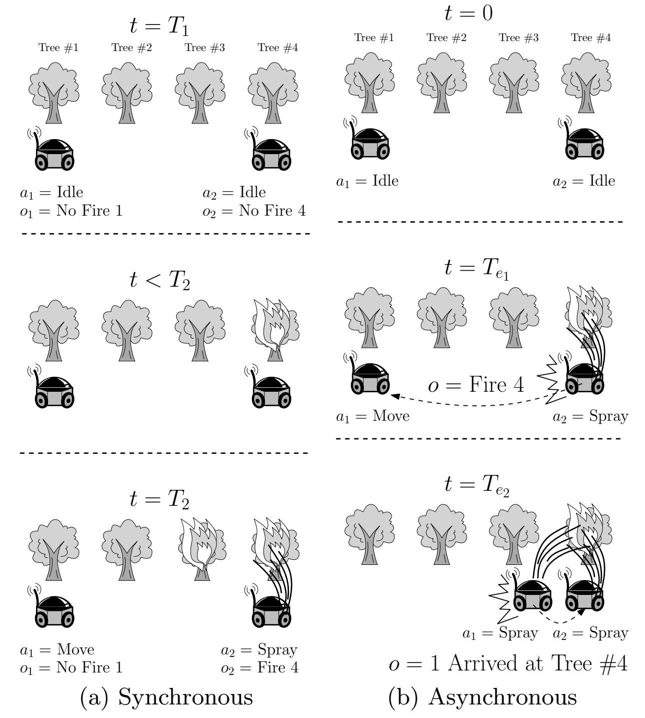


Figure 1: A graphical representation of the dynamics of a synchronous MPOMDP (a) and of an Event-Driven MPOMDP (b). Two agents must extinguish forest fires. In (a), agents cannot react instantaneously, for example, when a fire breaks out between instants $t = T_1$ and $t = T_2$; In (b), at instants $t = T_{e_1}$ and $t = T_{e_2}$, agents 1 and 2 respectively detect events, triggering new decision episodes for the team.

Synchronous vs. Asynchronous Execution

From a planning perspective, the dynamics of a multiagent POMDP are a trivial extension of those of a single-agent model: agents select an individual action based on a joint belief state, b, according to a joint policy, $\pi(b)$. In practice, however, this implies that agents must only communicate their observations at the *end of each decision episode*, so as not to exclude any potential information. Since, in most cases, no agent can individually determine if a decision episode has finished or not the team must operate *synchronously*: taking actions and sharing observations at a fixed rate, or at predefined time instants. However, this means that agents can no longer react immediately to sudden changes in the state, which, in some cases, can be vital to the outcome of their plan. Furthermore, there are exponentially many possible combinations of individual observations which can be shared amongst the team, which is an evident computational drawback to any planning algorithm which iterates over possible joint observations. The synchronous dynamics of an MPOMDP are illustrated in Figure 1a.

We propose an alternate approach, represented in Figure 1b, that is arguably more intuitive: team decisions are triggered by changes in the state of the system. These changes may be detected by any agent, in the form of a characteristic observation, and are promptly communicated to the rest of the team. The detecting agent can select an appropriate action immediately, and other agents may do so as soon as they receive that information. In addition to preserving responsiveness, this approach implies that, as a function of the number of agents, there are linearly many possible observations to be considered, corresponding to the sum of the possible detections of each of them. This forms the core of our approach.

Stochastic, Partially Observable Events

Before formally defining a model which operates according to the requirements formulated above, we need to have a rigorous definition of what constitutes an "event".

Definition 2. *A set $\mathcal{E}$ is a set of* events *of an MPOMDP if and only if there is a surjective mapping $\Phi : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{E}$ such that if $\Phi(s_1, s_1') = \Phi(s_2, s_2')$, then $\Pr(s_1' \mid s_1, \mathbf{a}) = \Pr(s_2' \mid s_2, \mathbf{a})$ and $\Pr(\mathbf{o} \mid s_1, \mathbf{a}, s_1') = \Pr(\mathbf{o} \mid s_2, \mathbf{a}, s_2')$, for every $s_1, s_1' \in \mathcal{S}, s_2 \neq s_1, s_2' \neq s_1' \in \mathcal{S}$, $\mathbf{a} \in \mathcal{A}$, $\mathbf{o} \in \mathcal{O}$.*

An event $e \in \mathcal{E}$ is said to be enabled *at $\langle s, a\rangle$ if, for s' such that $e = \Phi(s, s')$, $\Pr(s' \mid s, a) > 0$. The set of all enabled events in these conditions will be represented as $E(s, \mathbf{a})$.*

Through this definition, events are seen as abstractions to the state transitions $\langle s, s'\rangle$ which share the same stochastic properties. Their probability of occurrence can be influenced locally by the actions of each agent, or cooperatively through joint actions. In fact, since trivially $\Pr(s, s'|s, \mathbf{a}) = \Pr(s'|s, \mathbf{a})$, we can also represent the transition function as $T(s, \mathbf{a}, e)$ with $e = \Phi(s, s')$. Note that, in contrast to standard POMDP models, we include both s and s' in O as we are interested in observing characteristics of *state transitions*, as opposed to characteristics of states. As such, we have $O(s, \mathbf{a}, s', \mathbf{o}) = \Pr(\mathbf{o} \mid s, \mathbf{a}, s')$, allowing the observation model to be indexed through events, as $O(\mathbf{a}, e, \mathbf{o})$.

Noisy event detection processes are typically characterized through their susceptibility to false positive and false negative errors. However, in a system where decision episodes are driven by detected events, the latter case raises a fundamental problem – if all agents fail to detect the occurrence of an event, agents will not be able to change their actions. This has evident implications for the correctness of the expected value which is calculated during planning and, as such, will be explicitly taken into account.

A Model for Event-Driven MPOMDPs

We can now formally introduce our modeling approach:

Definition 3. *An Event-Driven Multiagent POMDP is a tuple $\langle d, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{E}, T, O, \mathcal{C}, R\rangle$ where:*

$d, \mathcal{S}, \mathcal{A}, R$ are defined as in an MPOMDP (Definition 1);

$\mathcal{O}$ is a set of observations o which can be generated by the environment upon the occurrence of a state transition. Observations are shared by agents;

$\mathcal{E}$ is a set of events (Definition 2);

$T : \mathcal{S} \times \mathcal{A} \times \mathcal{E} \rightarrow [0, 1]$ is the transition function*, such that $T(s, \mathbf{a}, e) = \Pr(e \mid s, \mathbf{a})$ for $e \in \mathcal{E}$, $s \in \mathcal{S}$, $\mathbf{a} \in \mathcal{A}$;*

$O : \mathcal{A} \times \mathcal{E} \times \mathcal{O} \rightarrow [0, 1]$ is the observation function*, such that $O(\mathbf{a}, e, o) = \Pr(o \mid \mathbf{a}, e)$ for $o \in \mathcal{O}$, $e \in \mathcal{E}$, $\mathbf{a} \in \mathcal{A}$;*

$\mathcal{C} : \mathcal{A} \times \mathcal{O} \rightarrow PS(A)\backslash\emptyset$, where $PS(A)$ is the power set of A, is a constraint-generating *function which returns, for each pair $\langle \mathbf{a}, o\rangle$, a* constrained action set *$\mathcal{C}(\mathbf{a}, o) \subseteq \mathcal{A}$. This set represents the joint actions which are available to the agents at the onset of a decision episode, given that, at the previous step, the team of agents executed $\mathbf{a}$ and observed o.*

The presence of the constraint function $\mathcal{C}$ in this definition addresses the problem of unobservable events raised before. In particular, during planning, we can model the occurrence of such events through token observations f such that $\mathcal{C}(\mathbf{a}, f) = \mathbf{a}$. These observations are never received by an agent during plan execution (hence why they are simply *tokens*), but they force planning algorithms to select the same action across different time steps, to account for the fact that agents will not be able to observe the events associated to f and change their actions accordingly.

SOLVING EVENT-DRIVEN MPOMDPS

Having defined a framework for Event-Driven MPOMDPs, we will show in this section that these models retain the necessary properties that allow them to be solved through dynamic programming approaches. We will then present an example of how to modify a typical POMDP solution algorithm to allow it to provide (optimal or approximate) policies for our proposed framework.

Dynamic Programming

We can show that a value function for an Event-Driven MPOMDP in the presence of action constraints is still PWLC, which enables the use of dynamic programming techniques to calculate (or approximate) an optimal policy.

Theorem 1. *For an Event-Driven MPOMDP, and for finite n, the optimal value function V_n^* can be written as:*

$$V_n^*(b) = \max_{\upsilon_n \in \Upsilon_n} \upsilon_n \cdot b \quad ,$$

where Υ_n is a set of $|S|$-dimensional vectors.

Proof. Given a constraint-generating function $\mathcal{C}$, any policy in an Event-Driven MPOMDP is subject to the following restrictions:

$$\begin{cases} \pi_n(b|_n) \in \mathcal{A} & \text{if } n = h \\ \pi_n(b|_n) \in \mathcal{C}(\pi_{n+1}(b|_{n+1}), o|_{n+1}) & \text{if } n < h \end{cases} ,$$

where $b|_n$ and $o|_n$ are, respectively, the belief state and the observation received when there are n steps remaining. Taking these restrictions into consideration, the Bellman backup for this model can be written as:

$$V_n^*(b) = \max_{\mathbf{a}\in\mathcal{A}} \Big\{ \sum_{s\in S} b(s)R(s, \mathbf{a}) \quad + \\ \gamma \sum_{\substack{s\in\mathcal{S}, o\in\mathcal{O} \\ e\in E(s,a),}} b(s)O(\mathbf{a}, e, o)T(s, \mathbf{a}, e) \max_{\mathbf{a}'\in\mathcal{C}(\mathbf{a},o)} Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}')\Big\}, \tag{3}$$

where $b_o^{\mathbf{a}}$ is the updated belief state according to $\langle \mathbf{a}, o\rangle$.

For $\mathbf{a} \in \mathcal{A}$, $o \in \mathcal{O}$, let $H_o^{\mathbf{a}}$ be a $|\mathcal{S}| \times |\mathcal{S}|$ matrix with

$$\begin{aligned}[H_o^{\mathbf{a}}]_{i,j} &= \Pr(s_i|s_j, \mathbf{a})\Pr(o|\, s_j, \mathbf{a}, s_i) \\ &= T(s_j, \mathbf{a}, \Phi(s_j, s_i))O(\mathbf{a}, \Phi(s_j, s_i), o) \quad .\end{aligned}$$

The Bellman backup is then, in vectorial form:

$$V_n^*(b) = \\ \max_{\mathbf{a}\in\mathcal{A}} \Big\{ r^{\mathbf{a}} \cdot b + \gamma \sum_{o\in\mathcal{O}} \mathbf{1}^T H_o^{\mathbf{a}} b \max_{\mathbf{a}'\in\mathcal{C}(\mathbf{a},o)} Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}')\Big\}, \tag{4}$$

where $r^{\mathbf{a}}$ denotes the $\mathbf{a}-$th column of $R(s, \mathbf{a})$. Also in this notation, the belief update step is:

$$b_o^{\mathbf{a}} = \frac{H_o^{\mathbf{a}} b}{\mathbf{1}^T H_o^{\mathbf{a}} b} \quad , \tag{5}$$

where $\mathbf{1}_i = 1$, $i = \{1 \ldots |\mathcal{S}|\}$.

At the final decision step, $n = 0$, we have that:

$$V_0^*(b) = \max_{\mathbf{a} \in \mathcal{A}} r^{\mathbf{a}} \cdot b, \tag{6}$$

And therefore V_0^* is clearly PWLC with $\Upsilon_0 = \{r^{\mathbf{a}} | \mathbf{a} \in \mathcal{A}\}$. Inductively, at $n-1$ steps-to-go:

$$V_{n-1}^*(b_o^{\mathbf{a}}) = \max_{\upsilon_{n-1} \in \Upsilon_{n-1}} \upsilon_{n-1} \cdot b_o^{\mathbf{a}} \quad .$$

Also, since $V_{n-1}^*(b_o^{\mathbf{a}})$ is PWLC iff $Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}')$ is PWLC:

$$Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}') = \max_{q_{n-1}^{\mathbf{a}'} \in \mathcal{K}_{n-1}^{\mathbf{a}'}} q_{n-1}^{\mathbf{a}'} \cdot b_o^{\mathbf{a}} \quad , \tag{7}$$

where $\mathcal{K}_n^{\mathbf{a}}$ is a set of $|\mathcal{S}|-$dimensional vectors. Taking this form for the Q-value, and substituting (5):

$$Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}') = \max_{q_{n-1}^{\mathbf{a}'} \in \mathcal{K}_{n-1}} \frac{(q_{n-1}^{\mathbf{a}'})^T H_o^{\mathbf{a}} b}{\mathbf{1}^T H_o^{\mathbf{a}} b} \quad .$$

Let

$$q_{n-1}^{*,\mathbf{a}'} | \, b, \mathbf{a}, o = \operatorname*{arg\,max}_{q_{n-1}^{\mathbf{a}'} \in \mathcal{K}_{n-1}} \left\{ (q_{n-1}^{\mathbf{a}'})^T H_o^{\mathbf{a}} b \right\} \quad . \tag{8}$$

Then,

$$Q_{n-1}^*(b_o^{\mathbf{a}}, \mathbf{a}') = \frac{(q_{n-1}^{*,\mathbf{a}'})^T H_o^{\mathbf{a}} b}{\mathbf{1}^T H_o^{\mathbf{a}} b} \quad .$$

Returning to (4), and reorganizing and simplifying terms:

$$\begin{aligned} &V_n^*(b) = \\ &\max_{\mathbf{a} \in \mathcal{A}} \left\{ \left(r^{\mathbf{a}} \cdot b + \gamma \sum_{o \in \mathcal{O}} \mathbf{1}^T H_o^{\mathbf{a}} b \max_{\mathbf{a}' \in \mathcal{C}(\mathbf{a}, o)} \frac{(q_{n-1}^{*,\mathbf{a}'})^T H_o^{\mathbf{a}} b}{\mathbf{1}^T H_o^{\mathbf{a}} b} \right) \right\} \\ &= \max_{\mathbf{a} \in \mathcal{A}} \left\{ \left(r^{\mathbf{a}} \cdot b + \gamma \sum_{o \in \mathcal{O}} \max_{\mathbf{a}' \in \mathcal{C}(\mathbf{a}, o)} (q_{n-1}^{*,\mathbf{a}'})^T H_o^{\mathbf{a}} b \right) \right\} \end{aligned} \tag{9}$$

Therefore, $V_n^*(b) = \max_{\upsilon_n \in \Upsilon_n} \upsilon_n \cdot b$, with

$$\upsilon_n = r^{\mathbf{a}} + \left(\gamma \sum_{o \in \mathcal{O}} \max_{\mathbf{a}' \in \mathcal{C}(\mathbf{a}, o)} (q_{n-1}^{*,\mathbf{a}'})^T H_o^{\mathbf{a}} \right)^T \quad . \tag{10}$$

It should be noted that each vector υ is associated with a particular action $\mathbf{a}$. This concludes the proof that Event-Driven MPOMDPs have PWLC optimal value functions. $\square$

Next, we will show how this result can be used by most current POMDP planning algorithms, with minor modifications, for Event-Driven MPOMDPs.

A Randomized Point-Based Algorithm

We now turn our attention to the problem of calculating approximately optimal policies for Event-Driven MPOMDPs. We here focus explicitly on *approximately* optimal policies for computational reasons. However, that does not preclude the possibility of adapting optimal algorithms as well.

A particularly efficient family of approximate POMDP solvers is that of point-based algorithms [8, 9, 14, 6]. We propose an adaptation of the Perseus randomized point-based algorithm that can handle Event-Driven MPOMDPs. The basic premise of any point-based algorithm is that, given a belief state $b \in \Pi(\mathcal{S})$, and a value function (or set of $Q-$value functions) at stage $n-1$, it is possible to obtain the stage-n maximizing vector at that point at a relatively low computational cost.

We are therefore interested in obtaining:

$$q_n^{\mathbf{a},b} = \operatorname*{arg\,max}_{q_n^{\mathbf{a}} \in \mathcal{K}_n^{\mathbf{a}}} q_n^{\mathbf{a}} \cdot b \tag{11}$$

From (10), we have that:

$$q_n^{\mathbf{a}} = r^{\mathbf{a}} + \left(\gamma \sum_{o \in \mathcal{O}} \max_{\mathbf{a}' \in \mathcal{C}(\mathbf{a}, o)} (q_{n-1}^{*,\mathbf{a}'})^T H_o^{\mathbf{a}} \right)^T \quad . \tag{12}$$

Note that this already implicitly defines an optimal $q_{n-1}^{*,\mathbf{a}'}$ at a given point b for a particular $\langle \mathbf{a}, o \rangle$ pair, see Eq. (8). If, instead of taking the maximum, we evaluate the expected future reward for each vector in $\mathcal{K}_{n-1}^{\mathbf{a}}$, and for a given $\langle \mathbf{a}, o \rangle$:

$$q_{n,o,\mathbf{a}}^{k,\mathbf{a}'} = \gamma \left((q_{n-1}^{k,\mathbf{a}'})^T H_o^{\mathbf{a}} \right)^T \quad . \tag{13}$$

Taking the action constraints into consideration, we can select from these vectors the best at b:

$$q_{n,o}^{\mathbf{a},b} = \operatorname*{arg\,max}_{q_{o,\mathbf{a}}^{k,\mathbf{a}'} \,|\, \mathbf{a}' \in \mathcal{C}(\mathbf{a}, o)} q_{o,\mathbf{a}}^{k,\mathbf{a}'} \cdot b \quad . \tag{14}$$

And finally, summing over observations and adding the immediate reward:

$$q_n^{\mathbf{a},b} = r^{\mathbf{a}} + \gamma \sum_{o \in \mathcal{O}} q_{n,o}^{\mathbf{a},b} \quad . \tag{15}$$

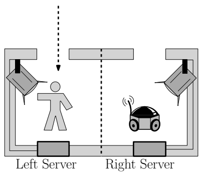
Equipped with this result, we can formulate our variant of the Perseus algorithm, which we refer to as *Constraint-Compliant* Perseus (CC-Perseus). The belief-backup (15) is applied to a subset of belief points in a sampled set $\mathcal{B}$ [14], for all possible actions. The resulting set of vectors for each action is taken as an approximation of $Q_{n+1}^{*,\mathbf{a}}$, and their union as V_{n+1}^*. Our explicit use of the $Q-$value functions stems from the fact that the sets $\mathcal{K}_n^{\mathbf{a}}$ (c.f. (7)) must never be empty. Otherwise, if an action had no previous-stage vectors associated to it, we would not have an estimate of its expected value when "forced" by $\mathcal{C}$.

A note on complexity: since this algorithm is keeping track of all $Q-$value functions, in the worst case, it has to perform $|\mathcal{A}|$ times as many evaluations over the set $\mathcal{B}$ as the standard version of Perseus. It is expected, then, that it should underperform Perseus when running over the same model. However, recall that the main advantage of our formulation is that it allows considerably smaller representations (particularly in $|\mathcal{O}|$) of the same problem.

Finally, we emphasize that the considerations made in this section could be applied to virtually any POMDP solution algorithm. The only requirements are that: $Q-$value functions must be maintained for all actions (so value functions can only be pruned action-wise); and the constraints generated by $\mathcal{C}$ should be satisfied when performing backups.

Execution-Time Belief Updates

In a standard POMDP, a belief state b can be updated by an agent during plan execution, following the execution of $\mathbf{a}$ (by the team), and observation of o, through Eq. (5). In an Event-Driven model, this update step is not always applicable. Planning algorithms, such as CC-Perseus, can



Problem	Model	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{O}\rvert$	d
Access2	E	72	6	9	3
	S	72	18	96	3
Access3	E	216	8	10	4
	S	216	54	256	4

Figure 2: Left: A layout of the *Access2* problem; Right: Size of the model components for the tested scenarios, using event-driven (E) and synchronous (S) approaches.

explicitly model the occurrence of false negative detections of events as symbolic observations. During execution, however, agents will not have access to any information indicating false negative detections. Therefore, agents must take into account the fact that the system can undergo several unobserved transitions between any two belief update steps.

THEOREM 2. *Let $f \in \mathcal{O}$ represent false negative detections of events. For an infinite-horizon agent in an Event-Driven POMDP, given that the team is executing* $\mathbf{a}$ *and observing o in belief state $\hat{b}$, the belief update step is:*

$$\hat{b}^{\mathbf{a}}_{o} = \frac{\left(H^{\mathbf{a}}_{o}(I - H^{\mathbf{a}}_{f})^{-1}\hat{b}\right)}{\mathbf{1}^{T}\left(H^{\mathbf{a}}_{o}(I - H^{\mathbf{a}}_{f})^{-1}\hat{b}\right)} \quad , \tag{16}$$

iff for all eigenvalues λ of $H^{\mathbf{a}}_{f}$, $|\lambda_i| < 1$.

PROOF. For $f \in \mathcal{O}$ indicating false negative observations, $o \in \mathcal{O}\backslash f$ and $\mathbf{a} \in \mathcal{A}$, we have that:

$$\begin{aligned}\hat{b}^{\mathbf{a}}_{o}(s') \propto & \sum_{s\in\mathcal{S}} \Pr(o|s,\mathbf{a},s')\Pr(s'|s,\mathbf{a}) \times \\ & \Big(1 + \sum_{s''\in\mathcal{S}} \Pr(f|s,\mathbf{a},s'')\Pr(s'|s'',\mathbf{a}) \\ & + \Big(\sum_{s''\in\mathcal{S}} \Pr(f|s,\mathbf{a},s'')\Pr(s'|s'',\mathbf{a})\Big)^{2} + \ldots\Big)\hat{b}(s)\end{aligned}$$

In matrix notation, as before, this is:

$$\hat{b}^{\mathbf{a}}_{o} \propto H^{\mathbf{a}}_{o}\Big(\sum_{k=0}^{\infty}(H^{\mathbf{a}}_{f})^{k}\Big)\hat{b} \tag{17}$$

If $|\lambda_i| < 1$ for all eigenvalues $|\lambda|$ of $H^{\mathbf{a}}_{f}$:

$$\hat{b}^{\mathbf{a}}_{o} = \frac{1}{\eta}H^{\mathbf{a}}_{o}(I - H^{\mathbf{a}}_{f})^{-1}\hat{b} \tag{18}$$

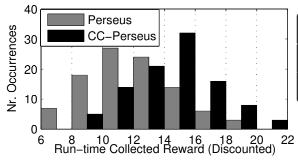
with $\eta = \mathbf{1}^{T}H^{\mathbf{a}}_{o}(I - H^{\mathbf{a}}_{f})^{-1}\hat{b}$, since $\mathbf{1}^{T}\hat{b}^{\mathbf{a}}_{o} = 1$. We note that if any $|\lambda_i| = 1$, this implies that the system has completely unobservable ($\Pr(f|\cdot) = 1$) chains, and the belief state cannot be tracked. □

The notation $\hat{b}$ here indicates run-time belief states, so as to clarify that Eqs. (5) and (16) will produce different outputs. In large systems, if the computational complexity of obtaining $(I - H^{\mathbf{a}}_{f})^{-1}$ is prohibitive, it can be approximated through a finite number of sums (see proof).

EXPERIMENTS

In this section, we present an evaluation of our proposed methodology in simulated multiagent decision-making problems. We consider an autonomous multiagent surveillance system, where static agents (e.g. sensors) and mobile agents (e.g. robots) must cooperate in order to control the access of human operators to sensitive equipment. In the *Access*2 problem, two sensors are connected to two restricted-access servers located in adjacent rooms (see Figure 2). Either sensor can mistake the validity of a user's credentials, or fail to detect that a user is there at all (0.2 probability each). A robot aids these sensors by performing additional measurements, reducing the possibility of false positives and negatives, and also acts as a failsafe in case one of the sensors malfunctions. The robot can move between the two rooms, and knows its position. The goal of the problem is to authorize as many valid users as possible.

This task was represented using both an event-driven and a standard synchronous approach, using factored models [4, 9]. The sizes of the respective model components can be seen in Figure 2. To see why there is such a pronounced difference in $|\mathcal{O}|$, consider that each sensor can observe authorized and non-authorized users, hardware failures, or nothing at all. The robot can also observe users (or nothing), along with its own position. In the synchronous approach, we must take the product of all of these possibilities. In the event-driven approach, the only events we need to model are the arrival/exit of a user, the failure of either sensor, and a change of state by the robot, plus an "unobserved" event. Additionally, we can reduce the number of joint actions by ruling out inconsistent decisions. Both the normal PERSEUS algorithm [14] and CC-PERSEUS were run on this problem, over a set of 50 sampled belief points with $\gamma = 0.95$ and $h = \infty$. As a baseline, the PERSEUS algorithm was run on our event-driven model, disregarding the effect of unobservable events. Figure 3a shows the maximum improvement (or



Algorithm	Run-Time Reward	$\bar{\delta}_V$
PERSEUS	12.94 ± 2.82	4.73
CC-PERSEUS	15.45 ± 2.75	0.60

Figure 4: Reward accumulated at run-time for *Access2*. Left: Comparative histogram for 100 runs of 25 steps. Right: respective mean/deviation, and mean error between collected reward and expected value ($\bar{\delta}_V$).

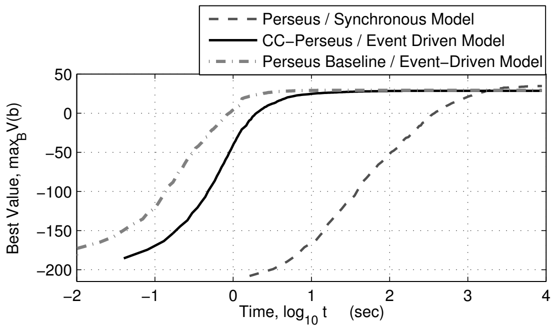
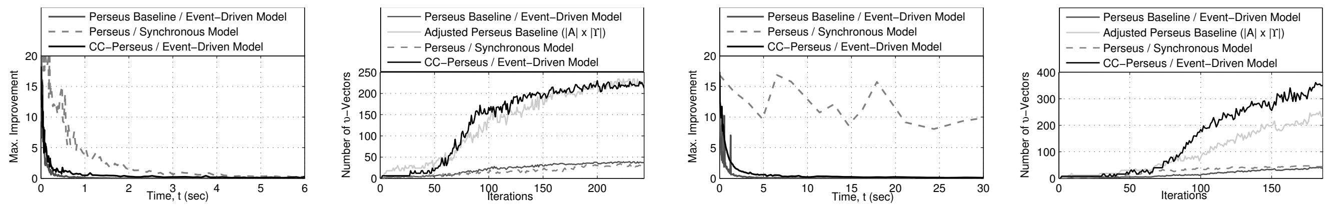


Figure 5: Evolution of $\max_{\mathcal{B}} V_n(b)$ in real time for the various models/solvers in *Access3*, showing similar final results, but faster convergence in the event-driven case.



(a) Access2, residual (b) Access2, $|\Upsilon_n|$ (c) Access3, residual (d) Access3, $|\Upsilon_n|$

Figure 3: (a), (c) Residual difference between successive value function approximations, $\max_{\mathcal{B}}\{V_n(b) - V_{n-1}(b)\}$**.(b), (d) Size of the value function,** $|\Upsilon_n|$**, as a function of** n**.**

residual) to the value function between steps using these algorithms, which is indicative of their real-time convergence. From here we can see that CC-Perseus on the event-driven model outperforms its standard version in the synchronous setting (total run-time was $22.78\,s$ vs. $29.03\,s$, respectively, until a residual of 10^{-4}). Figure 3b also shows how the total number of vectors of CC-Perseus follows $|\mathcal{A}|$ times that of the baseline, since $Q-$functions are explicitly maintained for each action. In Figure 4 we show how the baseline policy, even though faster to compute, is outperformed by CC-Perseus, since action constraints are not considered in the former case. Due to this fact, the expected value calculated by Perseus does not correspond to the reward accrued at run-time (an error of 27%, vs. 3% with CC-Perseus).

In order to showcase the scalability of these methods, a larger version of the above problem was implemented. *Access3* has 3 rooms/sensors/servers along a corridor, but the sensor at the center can only detect authorized personnel (or nothing). Therefore, there is only one more event with respect to the previous problem, but the presence of another agent causes an exponential increase in the number of observations of the synchronous model. Figure 2 shows the sizes of the models for this problem, and Figures 3c and 3d show performance results. Although synchronous and event-driven models are not strictly comparable with regard to expected reward, since their dynamics are inherently different, we show in Figure 5 an overlay of the best expected value (in the sample set) for *Access*3 using either model, to establish that they in fact converged to similar near-optimal policies. Running times to a residual of 10^{-4} were $6\,m\,52.39\,s$ for event-driven CC-Perseus and $2h\,27\,m\,24.27\,s$ for synchronous Perseus. This shows that the simple addition of an agent increased the computational advantage of the event-driven model over its alternative by more than an order of magnitude (also clearly visible in Figure 5). This establishes the scalability of our methods to large partially observable domains, with no assumptions regarding action/observation independence between agents.

CONCLUSIONS

In this work, we propose a novel modeling approach for multiagent decision-making under partial observability, based on the MPOMDP framework, which draws from the concepts of asynchrony in Discrete-Event systems to allow a more compact representation of such scenarios than what is typically possible through decision-theoretic frameworks.

We have described how such a model could be formalized and shown that it still retains the essential properties that allow it to be solved through dynamic programming. We have also shown how a common POMDP-solving algorithm could be adapted to function in an event-driven paradigm, and how agents can track belief states at run-time in the presence of false negative observations.

As future work, our event-driven models can be extended to continuous-time dynamics (such as in Semi-Markov Decision Processes). This would facilitate the application of these methods in other domains. We will also investigate specific solvers for this class of models, which could exploit their asynchrony for further computational gains.

ACKNOWLEDGMENTS

This work was funded in part by the Carnegie Mellon - Portugal Program (project CMU-PT/SIA/0023/2009), and by Fundação para a Ciência e a Tecnologia (FCT), through the PIDDAC Program funds (ISR/IST pluriannual funding, ref. PEst-OE/EEI/LA0009/2011).

J.M. was supported by a PhD Student Scholarship, from the Portuguese FCT POCTI programme, SFRH/BD/44661/2008.

M.S. is funded by the FP7 Marie Curie Actions Individual Fellowship #275217 (FP7-PEOPLE-2010-IEF).

REFERENCES

[1] R. Becker, S. Zilberstein, and V. Lesser. Decentralized markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 302–309. IEEE Computer Society, 2004.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[3] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210. Morgan Kaufmann Publishers Inc., 1996.

[4] C. Boutilier, R. Dearden, M. Goldszmidt, et al. Exploiting structure in policy construction. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1104–1113, 1995.

[5] C. Cassandras and S. Lafortune. *Introduction to discrete event systems.* Kluwer academic publishers, 1999.

[6] H. Kurniawati, D. Hsu, and W. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[7] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proc. of the National Conference on Artificial Intelligence*, 2005.

[8] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In

International Joint Conference on Artificial Intelligence, volume 18, pages 1025–1032. Citeseer, 2003.

[9] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes.* PhD thesis, University of Toronto, 2005.

[10] M. Puterman. *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons, Inc., 1994.

[11] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[12] W. Shen and D. H. Norrie. Agent-based systems for intelligent manufacturing: A state-of-the-art survey. *Knowledge and Information Systems*, 1:129–156, 1999.

[13] E. Sondik. *The optimal control of partially observable markov processes.* PhD thesis, Stanford, 1971.

[14] M. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195–220, 2005.

[15] F. Wu and X. Chen. Solving large-scale and sparse-reward DEC-POMDPs with correlation-MDPs. In U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 208–219. Springer Berlin / Heidelberg, 2008.