Other Agents' Actions as Asynchronous Events

Kurt D. Krebsbach
Department of Computer Science
Lawrence University
Appleton, WI 54912-0599
kurt.krebsbach@lawrence.edu

Abstract

An individual planning agent does not generally have sufficient computational resources at its disposal to produce an optimal plan in a complex domain, as deliberation itself requires and consumes scarce resources. This problem is further exacerbated in a distributed planning context in which multiple, heterogeneous agents must expend a portion of their resource allotment on communication, negotiation, and shared planning activities with other cooperative agents. Because other agents can have different temporal grain sizes, planning horizons, deadlines, and access to distinct local information, the delays associated with local deliberation and, in turn, shared negotiation are asynchronous, unpredictable, and widely variable.

We address this problem using a principled, decision-theoretic approach based on recent advances in Generalized Semi-Markov Decision Processes (GSMDPs). In particular, we use GSMDPs to model agents who develop a continuous-time deliberation policy offline which can then be consulted to dynamically select both deliberation-level and domain-level actions at plan execution time. This scheme allows individual agents to model other cooperative agents' actions essentially as asynchronous events, e.g., that might or might not fulfill a request (uncertain effect) after a stochastically-determined delay (uncertain event duration). With this approach, the decision-theoretic planner for the individual agent can make near-optimal execution-time decisions that trade off the risks and opportunities associated with their own actions, other agents' actions, and asynchronous external threats.

Introduction

Agents planning in interesting, overconstrained domains must operate under the assumption of *bounded rationality*, i.e., in a manner that is as near to optimal as possible with respect to its goals and available resources. To address this observation, researchers have applied planning techniques to the planning process itself. Here, planning is conducted at a higher level of abstraction than the base domain called the *meta domain*, with the higher-level planning sometimes referred to as *metaplanning*, or *metacognition*.

In general, deliberation scheduling involves deciding what aspects of an artifact (e.g., the agent's plan) should be improved, what methods of improvement should be chosen, and how much time should be devoted to each of these activities. In particular, we define *deliberation scheduling* to be a form of metacognition in which two planners exist: one, a *base-level planner* that attempts to solve planning problems in the base domain, and two, a *meta-level planner* deciding how best to instruct the base-level planner to expend units of planning effort. Both the meta and base domains are stochastic. Actions in the meta domain consist of a set of *base domain* problem configurations from which to choose, each of which constitutes a planning problem of varying difficulty (the successful result of which is a plan of corresponding quality), which might or might not be solvable, and which takes an unknown (but probabilistic) amount of time to complete. Similarly, the base domain's events and actions can succeed or fail, and have continuously-distributed durations.

The goal of the meta-level planner is to schedule the deliberation effort available to the base-level planner to maximize the expected utility of the base domain plans. This complex problem is further complicated by the fact that base planning and execution happen concurrently, further constraining the resources being allocated by the meta-level planner.

Deliberation Scheduling

In this paper we build on previous work on deliberation scheduling for SELF-ADAPTIVE CIRCA (SA-CIRCA) (Goldman, Musliner, & Krebsbach 2001; Musliner, Goldman, & Krebsbach 2003). In this earlier work, we developed a meta-level planner (called the Adaptive Mission Planner) as a specific component of SA-CIRCA to address domains in which the autonomous control system onboard an unmanned aerial vehicle (UAV) is self-adaptive: that is, it can modify its later plans to improve its performance while executing earlier ones. In this context, adaptation may be necessary for a variety of reasons – because the mission

is changed in-flight, because some aircraft equipment fails or is damaged, because the weather does not cooperate, or perhaps because its original mission plans were formed quickly and were never optimized.

While our previous work on deliberation scheduling problems has involved discretizing time to make MDPs tractable (Goldman, et.al.), recent advances in *Generalized Semi-Markov Decision Processes* (GSMDPs) suggest a way to model time and the related probability distributions continuously to support both goal-directed (Younes, Musliner, & Simmons 2003) and decision-theoretic planning (Younes & Simmons 2004). Of particular relevance here, Younes introduces GSMDPs as a model for asynchronous stochastic decision processes, and implements a decision theoretic planner, called TEMPASTIC-DTP. In this paper, we use TEMPASTIC-DTP to construct decision-theoretic deliberation policies in a domain similar to the UAV domain reported on earlier, and will show it is a more flexible and effective approach for several reasons, including:

- It provides a model of truly asynchronous events. This property in particular will allow us to extend the approach to distributed planning domains in which negotiation between agents is fundamentally asynchronous.
- It accurately models the essentially continuous nature of time and avoids biases due to the arbitrary granule size chosen to discretize time.
- It models the duration of a actions (controllable) and events (uncontrollable) as continuous random variables, rather than as discretized approximations or constants.
- The semi-Markov process relaxes the Markov assumption, which does not hold in general for problems involving continuous quantities such as time, and continuous probability distributions governing action and event durations.

Because deliberation scheduling involves several different planners at different levels of abstraction, we make a distinction between the *base domain*–where the base-level planner plans actions for UAVs flying a multi-phase mission, and the *meta domain*–in which the meta-planner schedules units of deliberation effort for the base-level planner. Meta-level policies are constructed by TEMPASTIC-DTP in advance of the mission. Base-level (base) planning can occur both offline, before the agent begins executing in the environment, and online, during execution of earlier base phase plans. We focus on online base planning–as controlled by the policy constructed during offline metaplanning–which attempts to adapt to changing circumstances and to continually improve the quality of base plans to be executed in future mission phases.

Background

Stochastic models with asynchronous events can be rather complex, in particular if the Markov assumption does not hold, such as if event delays are not exponentially distributed for continuous-time models. Still, the Markov assumption is commonly made, and the attention in the AI planning literature, in particular, is given almost exclusively to discrete-time models, which are inappropriate for asynchronous systems. We believe, however, that the complexity of asynchronous systems is manageable.

Stochastic Discrete Event Systems

We are interested in domains that can be modeled as *stochastic discrete event systems*. This class includes any stochastic process that can be thought of as occupying a single state for a duration of time before an *event* causes an instantaneous state transition to occur. In this paper, for example, an event may constitute the UAV reaching a waypoint, causing a transition into a state in the next phase of the misson. We call this a discrete event system because the state change is discrete and is caused by the triggering of an event.

Although state change is discrete, time is modeled most appropriately as a continuous quantity. A *continuous-time, discrete-state system* with countable state space S is a mapping $\{X : [0, \infty) \rightarrow S\}$ such that:

1. Transitions occur at strictly increasing time points: $0 = \tau_0 < \tau_1 < \tau_2 < \ldots$, where τ_n denotes the time of the nth transition.
2. Transitions can continue to infinite horizon: $\lim_{n\rightarrow\infty} \tau_n = \infty$; however, if all goals are achieved or no further actions or events are possible, the system will halt.
3. The index $\imath(t)$, denotes the last transition on or before time t: For $t > 0$, $\imath(t) = \max\{k \in \mathbb{N} | \tau_k \leq t\}$
4. The DES has piecewise constant trajectories: $X(t) = X(\tau_{\imath(t)})$. This ensures that the DES maintains a constant state $s_n = X(\tau_n)$ between the times of the nth and $(n+1)$th transitions.

Decision Processes

We now describe several important processes that can be used to model various classes of DESs. For each of these processes, we can add a decision dimension by distinguishing a subset of the events as controllable (called *actions*), and add rewards for decisions that lead to preferred outcomes. The resulting process is known as a *decision process*.

Markov Processes: A stochastic discrete event system is a **Markov process** if the future behavior at any point in time depends only on the state at that point in time, and not in any way on how that state was reached. Such a process is called *time inhomogeneous* if the probability distribution over future trajectories depends on the time of observation in addition to the current state. Because the path to the state cannot matter, the continuous probability distribution function that describes the holding time in the state must depend only on the current state, implying that for continuous-time Markov processes, this holding time is exponentially distributed (i.e., memoryless).

Semi-Markov Processes: In realistic domains, many phenomena are not accurately captured by memoryless distributions. The amount of time before a UAV reaches a waypoint and proceeds to the next mission phase, for example, clearly depends on how long the UAV has been flying toward that waypoint. A **semi-Markov** process is one in which, in addition to state s_n, the *amount of time spent in* s_n (i.e., *holding* time) is also relevant in determining state s_{n+1} Note, however, that the time spent in the current state (needed by an SMP) is not the same as the time of observation (needed for a time inhomogeneous MDP). Also note that the path of previous states taken to reach the current state is still inconsequential in determining the next state.

Generalized Semi-Markov Processes: Both Markov and semi-Markov processes can be used to model a wide variety of stochastic discrete event systems, but ignore the event structure by representing only the combined effects of all events enabled in the current state. The generalized semi-Markov process (GSMP), first introduced by Matthes (1962), is an established formalism in queuing theory for modeling stochastic discrete event systems that emphasizes the system's event structure (Glynn 1989).

A GSMP consists of a set of states S and a set of events E. At any time, the process occupies some state $s \in S$ in which a subset E_s of the events are enabled. Associated with each event $e \in E$ is a positive trigger time distribution $F(t, e)$, and a next-state distribution $p_e(s, t)$. The probability density function for $F, h(s)$, can depend on the entire execution history, which distinguishes GSMPs from SMPs. This property allows a GSMDP, unlike an SMDP, to remember if an event enabled in the current state has been continuously enabled in previous states without triggering–a critical property for modeling asynchronous processes.

Specifying DESs with GSMDPs: To define a DES, for all $n \geq 0$, we define τ_{n+1} and s_{n+1} as functions of $\{\tau_k, k \leq n\}$ and $\{s_k, k \leq n\}$. The GSMP model (Glynn 1989) provides a way to *stochastically* specify these inductive functions, and thus provides a representation of stochastic DESs. This representation assumes a countable set E of *events*. For each state $s \in S$ there is an associated set of events $E(s) \subseteq E$; these are the events that will be *enabled* (or *active*) whenever the system enters state s. At any given time, the active events are categorized as *new* or *old*. Initially at time 0, all events associated with s_0 are new events. Whenever the system makes a transition from s_n to s_{n+1}, events that are associated with both s_n and s_{n+1} are called old events, while those that are associated only with s_{n+1} are called new events.

Whenever the system enters state s_n, each new active event e receives a *timer value* that is generated according to a time distribution function $F(t, e)$. The timers for the active events run down toward zero until one or more timers reach zero. This is the time τ_{n+1}. Events that have zero clock readings are called *triggering events* (whose set is denoted by E_n). The next state s_{n+1} is determined by a probability distribution $p_e(s, t)$. Any non-triggering event associated with s_{n+1} will become an old event with its timer continuing to run down. Any non-triggering event not associated with s_{n+1} will have its timer discarded and become inactive. Finally, any triggering event e that is associated with s_{n+1} will receive a fresh timer value from $F(t, e)$. The definition of the GSMP is complete with the specification of a probability distribution over the initial state s_0. Note that the probability of two or more events simultaneously triggering is zero if all timer distributions are continuous; however, there are discrete event systems in which some timer distributions have discrete components, and thus simultaneity may be unavoidable (Alur, Courcoubetis, & Dill 1993; Shedler 1993).

The Necessity for GSMPs

The fact that the timers of old events continue to run down (instead of being reset) means that the GSMP model is non-Markovian with respect to the state space S. On the other hand, it is well-known that a GSMP as described above can be formally defined in terms of an underlying Markov chain $\{(s_n, c_n)|n \geq 0\}$, where s_n is the state and c_n is the vector of clock readings just after the nth state transition(Glynn 1989). In the special case where the timer distributions $F(t, e)$ are exponential with intensity $\lambda(e)$ for each event e, the process becomes a continuous-time Markov process. While each of the aspects of stochastic decision processes listed above have been individually addressed in research on decision theoretic planning, no existing approach deals with all aspects simultaneously. Continuous-time MDPs (Howard 1960) can be used to model asynchronous systems, but are restricted to events and actions with exponential trigger time distributions. Continuous-time SMDPs (Howard 1971) lift the restriction on trigger time distributions, but cannot model asynchrony. A GSMDP, unlike an SMDP, remembers if an event enabled in the current state has been continuously enabled in previous states without triggering. **This is key in modeling asynchronous processes, which typically involve events that race to trigger first in a state, but the event that triggers first does not necessarily disable the competing events** (Younes 2005). In the UAV domain, the fact that a threat presents itself in a particular phase in no way implies that other applicable threats in that phase do not continue to count down to their trigger times as well. For this reason, an "improved" plan that handles a certain combination of threats simultaneously is better than one that only handles single concurrent threats. Of course, the meta-level planner must decide whether the extra base-level planning work is justified given the time constraints and other improvement actions available to it.

A GSMDP Model

We have posed our deliberation scheduling problem as one of choosing, at any given time, what phase of the mission plan should be the focus of computation, and what plan improvement method should be used. This decision is made based on several factors, including:

- Which phase the agent is currently in;
- How long the agent is expected (probabilistically) to remain in that phase;
- The list of threats (and their combinations) that are possible in each phase along with a distribution governing how soon each threat might trigger;
- The probabilistic measure of the quality of the current plan for each remaining phase;
- The expected duration of each applicable improvement operator;
- The probability of success of each applicable improvement operator;
- The marginal improvement of each applicable improvement operator over the current plan for that phase.

We assume a fixed distribution of potential rewards among mission phases. For example, it might be worth 2 units of reward to survive to the point in the mission in which an important reconnaissance photo can be taken by the UAV agent. In general, each phase will not have explicit reward associated with it, but survival in each that phase still implies reward because the agent must survive to accumulate any further reward in future phases. The quality of an individual phase plan is thus based on the probability of surviving it by executing a plan that effectively handles the threats (harmful events) that actually occur. The quality of the overall plan is measured by how much reward it is expected to achieve. The system improves its expected reward by reducing its likelihood of failure, which, in turn, eliminates the possibility of future reward attainment.

The overall mission is decomposed into a sequence of **phases**:

$$\mathcal{B} = b_1, b_2, ..., b_n \ . \tag{1}$$

The base-level planner, under the direction of the meta-level planner, has determined an initial **plan**, P^0 made up of individual state space plans, p_i^0, for each phase $b_i \in \mathcal{B}$:

$$P^0 = p_1^0, p_2^0, ..., p_n^0 \ . \tag{2}$$

P^0 is an element of the set of possible plans, $\mathcal{P}$. We refer to a $P^i \in \mathcal{P}$ as the overall **mission plan**. The state of the system, then, may be represented as an ordered triple of time index, the phase in which the agent is currently operating, and the current mission plan. With some abuse of notation, we refer to the components of a state using operators $t(S), b(S)$ and $P(S)$ for $S \in \mathcal{S}$.

We model the duration of mission phases as continuous random variables. In essence, an event occurs according to a continuous probability distribution that indicates that the agent has crossed a phase boundary.

The meta-level planner has access to several **plan improvement methods**,

$$\mathcal{M} = m_1, m_2, ..., m_m \ . \tag{3}$$

At any point in time, t, the meta-level planner can choose to apply a method, m_j, to a phase, b_i (written as $m_j(i)$). Application of this method *may* yield a new plan for mission phase b_i, producing a new P^{t+1} as follows: if

$$P^t = p_1^t, p_2^t, \ldots p_i^t, \ldots, p_n^t \ , \tag{4}$$

then

$$P^{t+1} = p_1^t, p_2^t, \ldots p_i^{t+1}, \ldots, p_n^t \ , \tag{5}$$

where

$$p_i^t \neq p_i^{t+1} \ . \tag{6}$$

Note that the application of this method may fail, yielding $P^{t+1} = P^t$. Application of a method never yields a new plan of lower measured quality than the original plan, since if we generate a worse phase plan, we simply discard it and keep the previous one. Finally, the amount of time it takes for the base planner to return is similarly modeled as a continuous random variable; thus, all plan improvement may actions have uncertain effects and uncertain duration, but obey the constraints imposed by a continuous probability distribution.

To complete the formulation, we must have a utility (reward) function, U applying to the states. Again, to simplify, we assess the agent a reward, $U(i)$ on the completion of mission phase i. For example, if the aircraft achieves it mid-mission objective of taking a reconnaissance photo it receives some reward, and receives the balance of the possible reward by returning home safely. In this domain, there is no notion of receiving reward by staying alive; the mission only exists to achieve a primary objective and, if possible, to recover the UAV (a secondary objective).

Different phases of the mission present different degrees of danger. We represent this by varying the average delay before certain threats are likely to occur (as threats are simply represented as delayed events). The probability of surviving a given phase is a function of both the hazards posed by that phase, and the quality of the state space plan that is actually in place when that phase is executed. This is shown in the description of the delayed event `die-phase-1-med` in Figure 1. Because a medium-competency plan is in place, the exponential delay mean has been increased to 300 time units instead of 100 time units for the initial, low-competency plan (not shown). By dynamically completing plan improvement actions, average delays on threatening actions increase the probability of surviving the phase before threats occur. The problem of achieving reward is transformed into one of surviving to complete phases that provide reward. The problem of choosing a deliberation action for each state consists of choosing which phase of the mission plan to try to improve, and how difficult an improvement operator to attempt.

Domain Description

Figure 1 contains an example of each of the major domain components written in PPDDL+ (Younes 2003; Younes & Littman 2004). States are represented as a combination of binary predicates. Because PPDDL+ restricts discrete state features to binary predicates, non-binary but discrete state features (such as the phase number) must be encoded with a set of binary variables (e.g., phv1=1 and phv2=1 encodes phase=3).[1] Survival probabilities are encoded as a two-bit value as well. For example, `(sp11)` and `(sp12)` cover the plan quality possibilities of low, medium, high, and highest for mission phase 1.

Planning Problem

As seen in the particular problem description, the base-level agent starts out alive in phase 0 with various survival probabilities pre-established by existing plans for each mission phase. In this example, we assume the agent will begin execution of the plan for phase 0 immediately, and thus has the option of selecting any improvement actions applicable to phases 1 through 3. The agent is allowed to attempt to improve the plan for the mission that it is currently executing. Such a new plan could add utility to the rest of the phase if it is constructed in time to be "swapped in" for the remainder of the phase. Finally, the mission is over when one of two absorbing states are reached: `(not alive)` or `(home)`.

Delayed Events

Phase transitions and failures are represented as delayed events. The former reflect the fact that the amount of time it will take for the UAV to avoid threats and arrive at the next waypoint is uncertain. The latter models the fact that threats to the agent can trigger asynchronously according to a probability distribution. One difference between the two is that phase transitions must happen in a predefined order: the agent will not move on to phase 2 until phase 1 has been completed; however, this is not the case with threats. Much of the rationale for using TEMPASTIC-DTP is that events–especially bad ones like transitions to failure–do not invalidate other bad events when one of them occurs. Therefore, the deliberation mechanism must be able to model a domain in which certain combinations of threats can occur in any order, and can remain enabled concurrently. Planning to handle these combinations of threats given uncertain information and an evolving world state is the primary reason for performing online deliberation scheduling in the first place. The asynchrony of the events and actions can have a dramatic effect on the optimality of the plan, and the amount of utility attainable.

[1]This restriction makes specifying action preconditions and effects cumbersome, and could be fairly easily fixed by adding a bounded integer type to PPDDL+.

```
;;; Examples from the UAV deliberation domain.
;;;
(define (domain UAV-deliberation)
  (:requirements :negative-preconditions
                 :gsmdp :rewards)
  (:predicates
     (alive) (home) (pic-taken)
     (phv1)(phv2)  ; 4 phases, binary-encoded
     (sp11)(sp12)  ; 4 quality levels/phase
     (sp21)(sp22) (sp31)(sp32)
     (ph1-help-requested-high))

;;; A mission phase transition event.
(:delayed-event phase-2-to-3
  :delay (exponential (/ 1.0 100.0))
  :condition (and (alive)
                  (phv1)  ; in ph 2 (10)
                  (not phv2))
  :effect (phv2))         ; -> ph 3 (11)

;;; A failure event.
(:delayed-event die-phase-1-med
  :delay (exponential (/ 1.0 300.0))
  :condition (and (alive) (not phv1) (phv2)
                  (not sp11) (sp12))
  :effect (not (alive)))

;;; A deliberation action (with mean varied).
(:delayed-action improve-ph2-high
  :delay (exponential (/ 1.0 (* 3.0 lambda)))
  :condition (and (alive)) ; in any context
  :effect (and (sp21) (not sp22))) ; -> high plan

;;; A domain action.
(:delayed-action take-recon-picture
  :delay (exponential 1.0)
  :condition (and (alive) (not pic-taken)
                  (phv1) (not phv2)) ; in ph2 (10)
  :effect (and (pic-taken)
               (increase (reward) 2))) ; goal

;;; An action requesting help from other agent.
(:delayed-action improve-ph1-request-help-high
   :delay (exponential (/ 1.0 20.0))
   :condition (and (alive))         ; in any context
   :effect (ph1-help-requested-high))  ; -> help-high

;;; Other agent's action modeled as a delayed event.
(:delayed-event improve-ph1-high-other-agent
  :delay (exponential (/ 1.0 100.0))
  :condition (and (alive)
                  (ph1-help-requested-high))
  :effect (and (sp11) (not sp12)))

;; The problem description.
(define (problem delib1)
  (:domain UAV-deliberation)
  (:init  (alive))    ; begin alive in phase 0
  (:goal (home))      ; goal to return safely
  (:goal-reward 1)    ; one unit for safe return
  (:metric maximize (reward)))
```

Figure 1: Partial PPDDL+ Domain Description.

Delayed Actions

Improvement actions are also delayed; i.e., when the deliberation planner begins an improvement action, the base-level planner will return an "answer" (either an improved plan, or failure) at some future, probabilistically-determined time. In the example above, this time is described by the exponential time distribution function where $\lambda = 20$ time units. This reflects the fact that while we have some information on the performance of the base-level planner, we cannot predict–based only on the planner inputs–the actual duration of the planning activity. As we will see shortly, this uncertainty should be taken into account and traded off at the meta level along with other costs, rewards, and uncertainties.

A second type of delayed action is a domain action. In this domain, there is only one such non-deliberation action: `take-recon-picture`. This action is fundamentally different from the other actions because it is a physical action in the world (not a command to commence deliberation on a specific planning problem), it has a very short duration, and it constitutes the primary goal of the plan, thus earning the most reward.

Negotiation with Other Agents

Finally, we provide an example of an action/event combination used to model negotiation and "remote" deliberation. Here, a distinguished action can be selected as an indirect plan improvement action in which the original agent requests assistance in handling a threat in a certain phase. The negotiation phase of this exchange is modeled as the fairly low-cost delayed action called `improve-ph1-request-help-high`. When this action is completed, the predicate `ph1-help-requested-high` becomes true, enabling the preconditions for the delayed event `improve-ph1-high-other-agent`. This event represents the original (requesting) agent's view of a second, cooperating agent deliberating on its behalf. An important difference with a normal deliberation action is that other actions can be chosen and executed by the original agent while the second agent deliberates. By representing negotiation as a controllable action (that takes time for the original agent), along with an uncontrollable event (an event for which we do have limited modeling information in the form of a parametric probability distribution), we split the overall exchange into distinct synchronous and asynchronous parts. We believe that eventually the negotiation portion can be best modeled as a series of asynchronous processes to avoid the lock-step, multi-stage, synchronous process that plagues many distributed planning systems.

Experiments

TEMPASTIC-DTP (T-DTP) accepts PPDDL+ as an input planning domain language. Based on this domain description, it converts the problem into a GSMDP which can then be approximated as a continuous-time MDP using phase-type distributions (Younes & Simmons 2004). If all events of a GSMDP have exponential delay distributions (as with the experimental results reported below), then the GSMDP is already a continuous-time MDP with a factored transition model. This continuous-time MDP can be solved exactly, for example by using value iteration, or uniformization can be used to obtain a discrete-time MDP if it is convenient to use an existing solver for discrete-time models.

TEMPASTIC-DTP then uses Multi-Terminal Binary Decision Diagrams (MTBDDs, aka, Algebraic Decision Diagrams) to compactly represent the transition matrix of a Markov process , similar to the approach proposed by Hoey et al. (1999). For our UAV deliberation scheduling domain, T-DTP uses expected finite-horizon total reward as the measure to maximize. Once the complete policy is constructed, T-DTP provides a simulation facility for executing policies given specific, randomly generated agent trajectories in the environment.

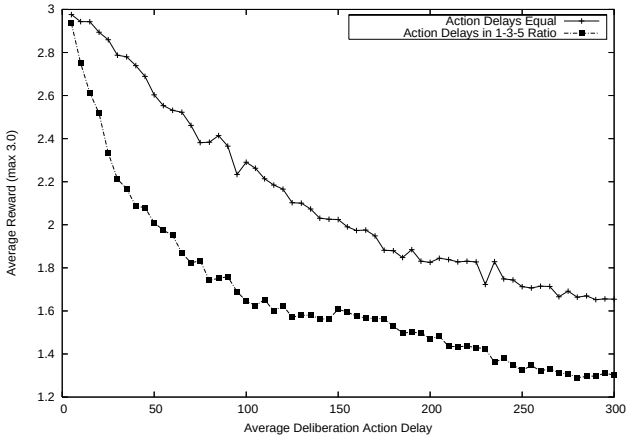

Figure 2: Average reward as a function of deliberation action duration.

Figure 2 illustrates the effect of average deliberation action duration (denoted as the variable `lambda` in Figure 1) on average reward obtained. For each average duration, TEMPASTIC-DTP constructs a policy for the planning problem partially described in Figure 1. This policy is then used to dictate the agent's behavior in 1000 simulations of the agent interacting with its environment. In each simulation, random values are chosen according to the stated (in this case, exponential) probability distributions to determine particular delays of actions and events in the domain. There are three possible results of a given simulation:

1. In the best case, the agent survives the threats encountered and returns home safely. This agent will earn two points of reward in mid-mission for achieving the goal of taking the reconnaissance picture, and one point for a safe return, for a maximum total reward of three points.
2. A partial success occurs when the agent survives long enough to take the reconnaissance picture (which can

be electronically transmitted back to base), but does not return home. Such an agent will earn two points.

3. In the worst case, the agent will not have constructed plans that allow it to survive at least one of the early threats it encounters, constituting a zero-reward mission.

Figure 2 contrasts the average reward obtained under two different assumptions. In one case, deliberation action delays obey an exponential probability distribution with the indicated action delay mean (x value) regardless of resulting plan quality. In other words, the average delay for a deliberation action resulting in a high-quality plan is the same as the average delay associated with producing a medium-quality plan. In the other case, action delay means obey a 1-3-5 ratio for medium, high, and highest quality plans respectively. For example, an average delay of 50 (shown as the x value) indicates that a deliberation action with a medium-value result will obey an exponential time distribution with a mean of 50. The two higher quality plans will take correspondingly longer on average, at 150 and 250 time units respectively. As expected, higher rewards can be obtained in the first case because producing higher quality plans is less expensive for a given value of x.

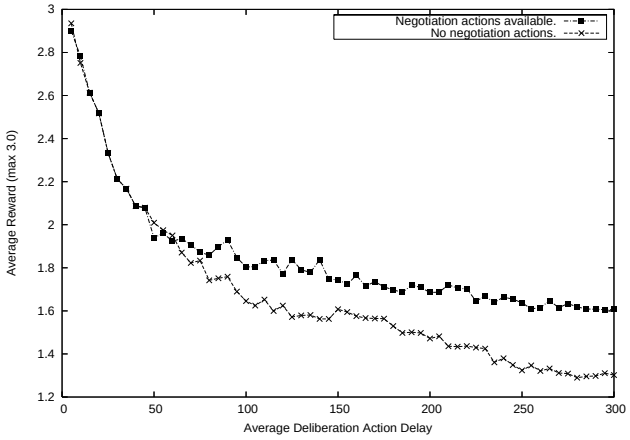

Figure 3: The effect of inter-agent negotiation options on average reward.

Figure 3 shows the average reward obtained as an agent's *own* deliberation actions take longer (as contrasted with a second, cooperating agent's actions). In one case, negotiating for threat-handling help from a cooperating agent is possible, while in the other, no negotiation is possible. While negotiation action duration and the other agent's deliberation delay is uncertain, the means of these probability distributions remain constant while the agent's own deliberation action durations increase.

Conclusion

We introduce a new approach to the problem of deliberation scheduling based on Generalized Semi-Markov Decision Processes. Due to inherent advantages unique to GSMDPs in which truly asynchronous events can be properly modeled, we suggest a new way to model negotiation with, and planning processes by, other cooperating agents. In particular, we suggest that negotiation actions, instigated by the original agent, be modeled as delayed actions and that the other agent's planning effort (action) be modeled as an asynchronous event from the original agent's perspective. In this way, the original agent can make primary planning decisions about which actions to take at any particular time by comparing the long-term utility of its own actions, negotiation actions with other agents, events involving cooperating agents' planning effort, and events viewed as threats to its own survival.

Acknowledgments

Many thanks to Haakan Younes for developing, distributing, and supporting TEMPASTIC-DTP, and for prompt, thorough answers to my many questions. Thanks also to Vu Ha and David Musliner for fruitful discussions and useful comments on earlier drafts.

References

Alur, R.; Courcoubetis, C.; and Dill, D. 1993. Model-checking in dense real-time. *Information and Computation* 104(1):2–34.

Glynn, P. 1989. A GSMP formalism for discrete event systems. *Proceedings of the IEEE* 77(1):14–23.

Goldman, R. P.; Musliner, D. J.; and Krebsbach, K. D. 2001. Managing online self-adaptation in real-time environments. In *Proc. Second International Workshop on Self Adaptive Software*.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In Laskey, K. B., and Prade, H., eds., *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 279–288. Stockholm, Sweden: Morgan Kaufmann Publishers.

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. New York: John Wiley & Sons.

Howard, R. A. 1971. *Dynamic Probabilistic Systems, Volume II*. New York, NY: John Wiley & Sons.

Matthes, K. 1962. Zur theorie der bedienungsprozesse. In *Transactions of the Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*, 513–528. Liblice, Czechoslovakia: Publishing House of the Czechoslovak Academy of Sciences.

Musliner, D. J.; Goldman, R. P.; and Krebsbach, K. D. 2003. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In *Proc. Third International Workshop on Self Adaptive Software*.

Shedler, G. S. 1993. *Regenerative Stochastic Simulation*. San Diego, CA: Academic Press.

Younes, H., and Littman, M. 2004. PPDDL 1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Younes, H. L. S., and Simmons, R. G. 2004. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 742–747. San Jose, California: American Association for Artificial Intelligence.

Younes, H.; Musliner, D.; and Simmons, R. 2003. A framework for planning in continuous-time stochastic domains. In *Proceedings of the Thirteenth Conference on Automated Planning and Scheduling*.

Younes, H. 2003. Extending PDDL to model stochastic decision processes. In *Proceedings of the ICAPS-03 Workshop on PDDL*, 95–103.

Younes, H. L. S. 2005. *Verification and Planning for Stochastic Processes with Asynchronous Events*. Ph.D. Dissertation, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.