Stochastic Deliberation Scheduling using GSMDPs

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

Abstract

We propose a new decision-theoretic approach for solving execution-time deliberation scheduling problems using recent advances in Generalized Semi-Markov Decision Processes (GSMDPs). In particular, we use GSMDPs to more accurately model domains in which planning and execution occur concurrently, plan-improvement actions have uncertain effects and duration, and events (such as threats) occur asynchronously and stochastically. We demonstrate a significant improvement in expressibility over previous discrete-time approximate models in which mission phase duration was fixed, failure events were synchronized with phase transitions, and planning time was discretized into constant-size planning quanta.

Introduction

Agents planning in overconstrained domains do not have unlimited computational resources at their disposal, and must therefore operate in a manner that is as near to optimal as possible with respect to their goals and available resources. Researchers have addressed this issue by applying planning techniques to the planning process itself; i.e., a second level of 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*. Because the problem described here involves both deciding *which* actions to take (planning) and *when* to take them (in continuous time), we refer to the problem as one of *deliberation scheduling*.

In general, deliberation scheduling involves deciding which aspects of an agent's plan to improve, what methods of improvement should be chosen, and how much time should be devoted to each of these activities. In particular, we describe a model in which two separate 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.

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

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 uncertain 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. The fact that base planning and execution occur concurrently further constrains the time 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.

A More Expressive Model of Uncertainty

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). Intuitively, 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. 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:

- 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 **uncertain durations** of both actions (controllable) and events (uncontrollable) as continuous random variables, rather than as discretized approximations or constants.
- It provides a model of **truly asynchronous events.**

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 deliberation effort for the base-level planner. Meta-level policies are constructed by TEMPASTIC-DTP in advance of the mission. While base-level planning can occur both offline and online, 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 the current or future mission phases.

Background

Stochastic models with asynchronous events can be complex 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 attention in the AI planning literature 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* (DESs). 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 agent reaching a waypoint, causing a transition into a state in the next phase of the mission. We call this a DES 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. In the interest of brevity, we describe a *continuous-time, discrete-state system* with countable state space S as a mapping $\{X : [0, \infty) \rightarrow S\}$ in which transitions occur at strictly increasing time points, can continue to infinite horizon (although absorbing events can be defined), and constant state is maintained between the transitions (piecewise constant trajectories).

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 DES is a **Markov process** if the future behavior at any time depends only on the state at that 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 (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 DESs, 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 DESs 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 function $F(t, e)$, and a next-state distribution function $p_e(s,t)$. 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 assumes a countable set E of *events*. For each state $s \in S$ there is an associated set of events $E(s) \subseteq E$ that will be *enabled* whenever the system enters state s. At any given time, the active events are categorized as *new* or *old*. 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.

When 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 (at 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 will 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 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 formally be 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). For example, in the UAV domain, the fact that a threat presents itself in no way implies that other threats do not continue to count down to their trigger times.

A GSMDP Model of Uncertainty

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;
- The expected duration of the current phase;
- The quality of the current plan for the current phase and each remaining phase;
- The expected duration of each applicable improvement operator;
- The probability of success for each applicable improvement operator;
- The expected marginal improvement if an improvement operator is successfully applied to a given phase.

The 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 base 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 current state of the system is represented by the current time, mission phase, and mission plan. We model mission phase duration as a continuous random variable in which an event occurs according to a continuous probability distribution that determines when the agent crosses a phase boundary.

Plan Improvement

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

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

At any time, t, the meta-level planner can choose to apply a method, m_j, to a phase, b_i. Applying 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

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

Of course, the application of this method may instead fail, yielding $P^{t+1} = P^t$, in which case the original phase plan is retained. Similarly, the amount of time it takes for the base planner to return a result is modeled as a continuous random variable. This uncertainty of duration implies one type of uncertainty of effect; namely, that an action will effectively fail if it returns a result too late to improve a plan for a phase that has already completed. In fact, if the currently-executing improvement action is relevant to the currently-executing phase, the utility of even its successful completion is constantly decreasing as less of the current phase will benefit from any improvement result.

We represent differing degrees of danger per phase by varying the average delay before certain threats (delayed events) are likely to occur. The probability of surviving a given phase is a function of both the threats that occur in that phase and the quality of the base plan that is actually in place when that phase is executed. By successfully completing plan improvement actions, new current or future phase plans increase the probability of handling threats should they occur.

Reward

We assume a fixed distribution of potential rewards among mission phases. In our example, it is worth two units of reward for the UAV agent to survive long enough to take an important reconnaissance photo. While each phase will not necessarily be associated with explicit reward, survival in that phase still implies reward due to reward in future phases. The quality of a phase plan is thus based on the probability of surviving it due to a plan that effectively handles the threats (harmful events) that actually occur. The quality of the overall plan is measured by how much reward the agent actually achieves given the threats that actually occur in a given simulation. The system improves its expected reward by increasing expected survival and thus, future reward potential. For example, if the UAV achieves its mid-mission objective of taking a reconnaissance photo it receives some reward (two units); it then receives the balance of the possible reward by returning home safely (one unit). There is no notion of receiving reward by just surviving; the mission only exists to achieve a primary objective and, if possible, to recover the UAV (a secondary objective).

Domain Description

Figure 1 exemplifies of each of the major domain components written in PPDDL+ (Younes 2003). States are represented as a combination of binary predicates. Because PPDDL+ restricts discrete state features to binary predicates, non-binary but discrete state features (e.g., phase number) must be encoded with a set of binary variables (e.g., phv1=1, phv2=1 encodes phase=3).[1]

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

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

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

;; 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: PPDDL+ Action and Event Descriptions.

Survival probabilities are encoded as two-bit values as well; e.g., (sp11) and (sp12) cover low, medium, high, and highest plan qualities for mission phase 1.

Planning Problem

In these experiments, the base-level agent starts out alive in phase 0 with various survival probabilities preestablished by existing plans for each mission phase. The agent then begins execution of the plan for phase 0 immediately, and has the option of selecting any improvement actions applicable to phases 1 through 3. In general, however, the agent is allowed to attempt to improve the plan for the phase that it is currently executing as well, as 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).

Uncertain Event Durations

Phase duration and threat delays are represented as delayed events. The former reflects the fact that the amount of time it will take for the UAV to navigate threats and arrive at the next waypoint is uncertain; the latter model the fact that threats to the agent can trigger asynchronously according to a probability distribution. 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 harmful events like threats leading to failure–do not invalidate other harmful events. 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 reward attainable.

Uncertain Action Durations

Improvement actions are also delayed: 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 uncertain time. As shown in Figure 1, this time is described by the exponential time distribution function where λ is varied. This reflects the fact that while we have there exists 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. Modeling this uncertainty properly allows the overall system to trade off uncertainties at the meta-level with base-level 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.

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). This continuous-time MDP can be solved exactly, for example by using value iteration, or via a discrete-time solver after a uniformization step. TEMPASTIC-DTP then uses 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 environmental trajectories.

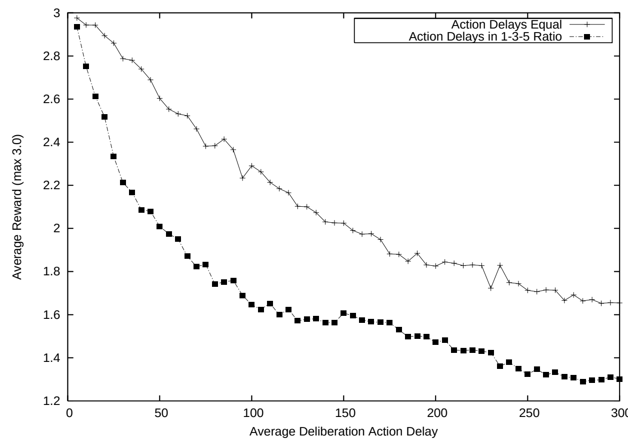

Figure 2: Average reward as a function of deliberation action duration.

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 probability distributions to determine particular delays of actions and events for a single simulation run. 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 (i.e., the maximum y value).
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 earns two points.
3. In the worst case, the agent fails to construct plans that allow it to survive some early (pre-photo) threat, resulting in a zero-reward mission.

Figure 2 contrasts the average reward obtained under two different assumptions. In one case (Action Delays Equal), *all* deliberation action delays obey an exponential probability distribution with the indicated action delay mean (x value); i.e., 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 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 planning tasks will take correspondingly longer on average, at 150 and 250 average 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.

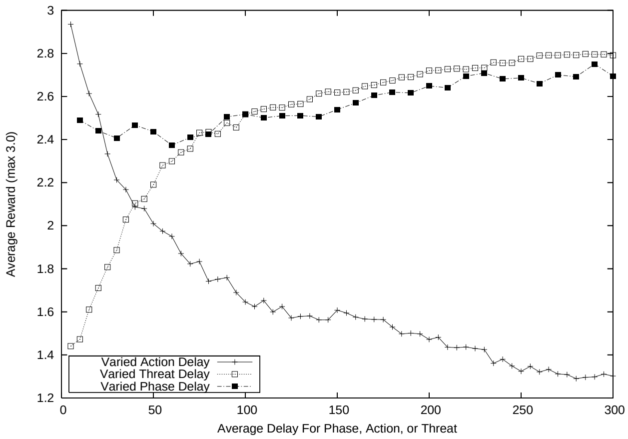

Figure 3: A comparison of average reward obtained as average delay varies for mission phases, agent actions, and environmental threats.

Figure 3 demonstrates the effect of varying the average delays for the three main sources of uncertainty: length of mission phase, duration of agent deliberation action, and delay before a triggered threat has its harmful effect. For each average delay (x value) between 5 and 300 (in increments of 5) a policy is constructed and followed for 1000 simulations. The actual rewards (of 0, 2, or 3) are then averaged and plotted as each source of uncertainty is varied while holding the other two constant; thus, each curve constitutes 75,000 simulations for a total of 225,000 simulations. As expected, agent performance declines as its own actions take longer, but improves as threats take longer to unfold. Performance also improves as phases take longer, since this affords the metaplanner more time to improve base plans before those base plans are executed; still, the effect is less pronounced as longer phases also give threats more time to occur. By modeling the problem domain more expressively as a GSMDP, the planner can take into account the interplay of these various uncertainties–at both the meta and base level–in a way that was previously not possible.

Conclusion

We introduce a new approach to more expressively modeling several sources of uncertainty inherent in overconstrained planning problems. By exploiting recent research results in the area of Generalized Semi-Markov Decision Processes, we demonstrate that previously inexpressible problems of deliberation scheduling can now be both stated and solved, allowing metaplanning agents to make better decision-theoretic tradeoffs in time-pressured, execution-time situations.

Acknowledgments

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

References

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 Conf. 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 Int'l Workshop on Self Adaptive Software.*

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.