Improving Trust Estimates in Planning Domains with Rare Failure Events

Colin M. Potts and Kurt D. Krebsbach
Dept. of Mathematics and Computer Science
Lawrence University
Appleton, Wisconsin 54911 USA
{colin.m.potts, kurt.krebsbach}@lawrence.edu

Jordan T. Thayer and David J. Musliner
SIFT, LLC
211 N. First Street
Minneapolis, MN 55401 USA
{jthayer, musliner}@sift.net

Abstract

In many planning domains, it is impossible to construct plans that are guaranteed to keep the system completely safe. A common approach is to build probabilistic plans that are guaranteed to maintain system with a sufficiently high probability. For many such domains, bounds on system safety cannot be computed analytically, but instead rely on execution sampling coupled with a plan verification techniques. While probabilistic planning with verification can work well, it is not adequate in situations in which some modes of failure are very rare, simply because too many execution traces must be sampled (e.g., 10^{12}) to ensure that the rare events of interest will occur even once.

The P-CIRCA planner seeks to solve planning problems while probabilistically guaranteeing safety. Our domains frequently involve verifying that the probability of failure is below a low threshold (< 0.01). Because the events we sample have such low probabilities, we use *Importance sampling* (IS) (Hammersley and Handscomb 1964; Clarke and Zuliani 2011) to reduce the number of samples required. However, since we deal with an abstracted model, we cannot bias all paths individually. This prevents IS from achieving a correct bias. To compensate for this drawback we present a concept of *DAGification* to partially expand our representation and achieve a better bias.

Introduction

Our approach (Younes and Musliner 2002; Younes, Musliner, and Simmons 2003) instead relies on an execution sampling-based verifier as the validation procedure used to establish whether the degree of trust specified by the human for a particular generated plan is sufficient, or whether further planning is required to probabilistically guarantee the desired threshold. While this approach to probabilistic planning with empirical verification works well in many complex domains, it is inadequate for domains with extremely rare events because too many execution traces must be sampled (e.g., 10^{12}) to ensure that the rare events of interest will be sampled at all.

Importance sampling is a statistical technique that relies on artificially biasing probability distributions of selected random variables prior to sampling, and then correspondingly unbiasing the results afterward. Due to its ability to increase the prevalence of rare events in the sample space, IS suggests a promising method for significantly reducing the number of samples needed to guarantee a desired safety level for a verification-based planner operating in these types of domains. We briefly describe how we use IS techniques with the probabilistic planner built into P-CIRCA (Younes, Musliner, and Simmons 2003), and describe the particular challenge of choosing good initial biases for appropriate random variables in the planner's state space. Finally, we describe a graph transformation algorithm that improves the setting of the initial IS biases, and analytically characterize the computational tradeoff between further graph transformation and the reduction in required verifier traces.

P-CIRCA

CIRCA is an architecture for real-time intelligent control. The original planner in CIRCA (Musliner, Durfee, and Shin 1993) builds reactive control plans that achieve system goals and maintain system safety subject to strict time bounds and models of the dynamic external environment. While the original CIRCA model includes nondeterminism in the outcome of actions and uncertainty about the timing and occurrence of externally-caused transitions, it does not have *quantified* uncertainty information. The extended model, called P-CIRCA, includes quantified uncertainty in the form of probability distributions on both the timing of different transitions and the outcomes themselves. This allows CIRCA to build plans that are not completely guaranteed to prevent failure; rather, plans may allow for the possibility of failure as long as the failure probability is below some specified threshold.

The Planner

The flowchart on the left of Figure 1 demonstrates how the planner and verifier interact. As shown, the planner accepts as input a model of transitions between states that includes both probability distributions over nondeterministic outcomes, and timing information. Given this model, the planner generates a plan to achieve the given objectives while preserving safety in a dynamic real-time environment. Because P-CIRCA plans actions to preempt transitions that lead to failure, the *plan* is a directed, possibly *cyclic* graph of the states reachable from the initial state given the transitions possible from each state, but not including states that have

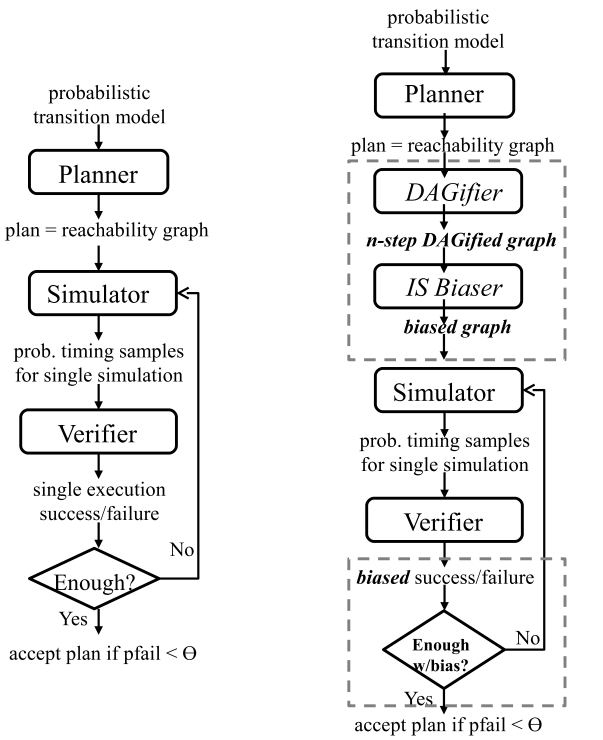
probabilistic transition model
Planner
plan = reachability graph
Simulator
prob. timing samples for single simulation
Verifier
single execution success/failure
Enough?
No
Yes
accept plan if pfail < Θ
probabilistic transition model
Planner
plan = reachability graph
DAGifier
n-step DAGified graph
IS Biaser
biased graph
Simulator
prob. timing samples for single simulation
Verifier
biased success/failure
Enough w/bias?
No
Yes
accept plan if pfail < Θ

Figure 1: P-CIRCA planning and verification both without (left) and with (right) importance sampling.

been made unreachable by planned preemptive actions. As noted, cycles are common in the graph for a variety of reasons: the world can undo things we want to achieve; actions can probabilistically fail to achieve postconditions and cause a self-loop. In such cases, the agent can essential remain or return to the "same state" but with some time having run off of the state's "dwell time" clock. It is for this reason that the world model of the probabilistic extension corresponds to a generalized semi-Markov process (i.e., the dwell time in a state can influence which transitions are possible out of a state). In this paper we will introduce technique for reducing the impact of these cycles on the verification step, a process we call DAGification. We will see shortly how eliminating at least some of these same states with different clocks through DAGification results in a better graph in which to employ importance sampling techniques.

The Verifier

In earlier work, Younes and Musliner (2002) presented a procedure for probabilistic plan verification to ensure that heuristically-generated plans achieve the desired level of safety. With this approach, we attempt to minimize verification effort while guaranteeing that at most a specified proportion of good plans are rejected and bad plans accepted. As shown on the left of Figure 1, the Monte Carlo simulator generates a set of random variables which constitute the dynamics of the environment for a single execution path. The plan is then executed in this environment to determine

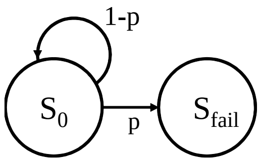

Figure 2: The original model.

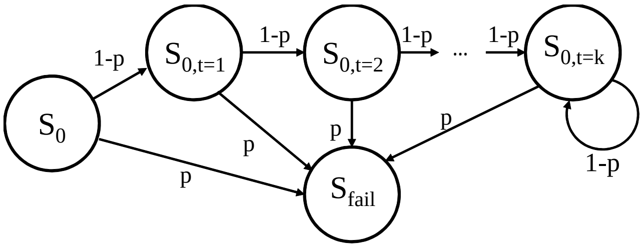

Figure 3: The original model with paths expanded out to length k.

a single success or failure outcome. Given enough sample paths, the verifier can guarantee that at most a specified proportion of good plans are rejected and bad plans accepted. Without importance sampling, the sequential probability ratio test (SPRT) (Wald 1945) is used after each simulation to determine whether enough samples have been run to halt simulation and return an overall result. This result will be to accept the plan if the simulations have determined that the probability of failure of the plan is less than the specified safety threshold ($p_{fail} < \theta$).

Importance Sampling

Statistical verification techniques work much more efficiently when sampling events with high probabilities. Importance sampling was developed to take a domain with very low-probability events and bias the probabilities of those events so that the low probability events can be sampled at a high probability. Through principled unbiasing of the results obtained during simulation, it is possible to derive the desired probabilities of hypothesis relevant to the original model. The biasing works by selecting some set of the probabilistic transitions in a model and assigning them new probabilities. (Note that the size of this selected set of transitions is critical to leveraging importance sampling to significantly reduce the number of samples, as we will discuss shortly.) Then when a sample is drawn we compute the ratio of the true probability of the trace $f(X)$ over the biased probability $f_*(X)$. We then take the mean of these ratios which gives us an estimate of the probability of the event (Hammersley and Handscomb 1964). Because we are sampling rare events much more often, we often require far fewer samples to prove or disprove the original hypothesis.

DAGification

As in most search problems, the state space is described implicitly via action and event descriptions. In the course

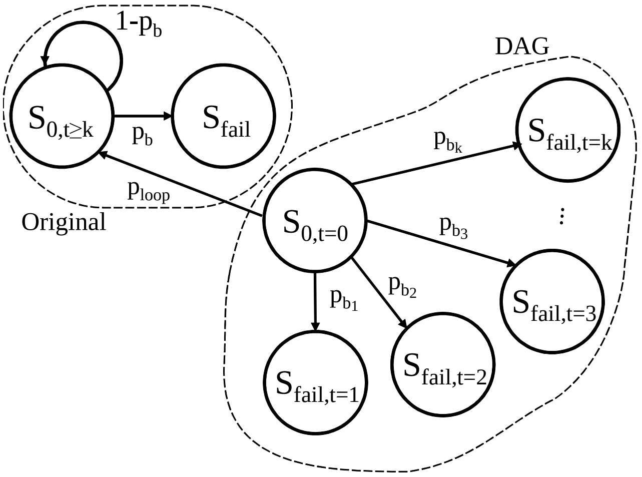

Figure 4: The partially-DAGified model.

of planning appropriate actions, P-CIRCA will expand portions of the reachable state space; however P-CIRCA will not explicitly "DAGify" cycles for which it can adequately plan without doing so. Such a situation is shown in Figure 2. P-CIRCA will attempt to plan the best action it can from state S_0 which may only probabilistically preempt the transition to S_{fail}; however, the planner itself might not need to expand states forward from S_0 to determine that best action.

Although planning might not always require it, DAGification of cycles provides more transitions to individually bias, and consequently provides more opportunities to improve the initial biases upon which more efficient importance sampling depends. If there is a cycle, we might be unable to express the optimal bias for IS and obtain a zero-variance estimator. This is not true for directed acyclic graphs (DAGs), in which case we can always express a zero-variance estimator. While it can sometimes be prohibitively expensive to fully-DAGify the cycle, partial DAGification can, as we will show, get us significantly closer to zero-variance.

Beginning with an implicit description of the state-space model we seek to expand this model to enable greater control of importance sampling. Ideally we would like to bias the probabilities of individual *traces* (as opposed to individual transitions), which would theoretically allow us to achieve an optimal bias, and obtain maximum benefit from importance sampling. We now describe a two-step approach that makes this possible, at least out to length k, at which point the computation may no longer be justified for the expected benefit.

The first step of DAGification involves taking the state-space model as shown in Figure 2 and expanding all paths from the initial state out to length k, as shown in Figure 3. We then introduce a new initial state, $S_{0,t=0}$. Let Y equal the set of all paths who either reach a sink state in less than k steps, or any path of length k from $S_{0,t=0}$. The process works by taking each $y \in Y$, and adding a single transition from $S_{0,t=0}$ to $S_{end(Y)}$. Here we use $S_{end(Y)}$ to denote the state where path y terminates. In the example we use, these paths all end in S_{fail} after a certain number of steps (i steps), so we have $S_{end(Y)} = S_{0,t=i}$ for $i = 1..k$, and one additional path that ends at $S_{0,t\geq k}$. The transition model from this point remains the same and corresponds to the original, unDAGified portion of the model, except with timing characteristics of $S_{0,t\geq k}$ adjusted to account for the dwell time of the states in the DAGified portion. We now analyze the relationship between this extra DAGification computation and the computation saved by requiring fewer samples due to improved importance sampling.

DAGification vs. Sampling

We would like the expected decrease in samples to be asymptotically greater than the work required by the DAGification process. Suppose our model – DAGified to depth k – has an average branching factor of b. Then DAGification is simply a breadth first search to depth k, and thus will expand b^k nodes. In terms of memory, we must then create a new model with a new state/transition pair for all fringe nodes of the search. This leads to the observation that ideally k is less than log_b of the expected decrease in the number of samples.

Analytical Results

We now describe two predictive functions for this simple domain that estimate the expected probability of failure. The function $MaxI$ gives us the largest length of a path that will be sampled if we take n samples in the given plan. To determine this number, we say that no path whose probability is less than x will be sampled. Φ is a function that depends on the same variables as $MaxI$ (i.e. $\Phi(p_*, n, x), \Phi(p_l, p_f, k, n, x)$). This dependency is notationally suppressed below for brevity. Equation 2 describes the expected failure rate for plain IS while Equation 3 describes the version with the additional DAGification step. We compute the analytical probability of failure after T time steps in Equation 1.

$$p_{fail} = 1 - (1-p)^T \tag{1}$$

$$p_{IS} = \frac{1}{\Phi}\left(1 - (1-p)^{MaxI(p_*,n,x)}\right) \tag{2}$$

$$p_{DAG} = \frac{1}{\Phi}\left(1 - (1-p)^{MaxI(p_l,p_f,k,n,x)}\right) \tag{3}$$

Given these probabilities, we now define functions to express the expected error. Each error is divided by p_{fail} to convert it to a percentage.

$$Err[p_{IS}] = \frac{p_{fail} - p_{IS}}{p_{fail}} \tag{4}$$

$$Err[p_{DAG}] = \frac{p_{fail} - p_{DAG}}{p_{fail}} \tag{5}$$

The graph in Figure 5 shows the corresponding error rates for values of 0.1, 0.5, and 0.9 for each of p_* (plain IS) and p_f (DAGified IS). The number of samples, n, varies along the x-axis from 50 to 500. We plot the error rate without DAGification ($Err[p_{IS}]$, from Equation 4 – as shown in the top three curves – against the error rate of IS with DAGification ($Err[p_{DAG}]$, Equation 5). The biased probabilities of

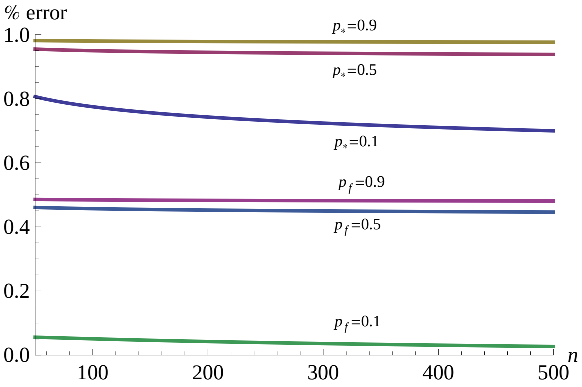

Figure 5: Expected error rates using importance sampling both with and without DAGification

transition to failure (from the original model) are denoted p_* for plain IS and p_f for DAGified IS. In both cases, the values $x = .1$, $p = .0001$, and $T = 200$ remain constant, while p_* and p_f each take on the same three values as shown. The additional parameters pertinent to IS with DAGification are $p_{loop} = .5$ (as shown in Figure 4), and $k = 100$ (the maximum depth of allowable graph expansions).

We note that IS with DAGification results in significantly lower error rates in all three cases. Since (3) depends on the sum of the probabilities of the paths expanded during DAGification $(1 - p_{loop})$, the lower error rates are not guaranteed to hold for every such choice of p_{loop}; however, we expect that for cases of interest to us the advantage would generally be gained. For instance, with the Cross-Entropy method as described in (Rubinstein and Davidson 1999; Clarke and Zuliani 2011), we would expect to produce a good bias and therefore would see results similar to these, even if depending on our initial bias the results would be indeterminate. DAGification thus allows such a method to achieve better results by allowing more control over the biasing.

DAGs vs. Digraphs

In general, the problem is one of the structure of directed acyclic graphs vs. generic directed graphs. So long as we can apply the right bias to a given path, importance sampling will produce the correct solution (with low variance). However, in P-CIRCA we have restricted ourselves to biasing actions rather than whole paths. This makes sense as we have to restrict cost of defining the bias, and it allows us to specify the perfect bias in many places.

Specifically, when specifying the bias over actions, we can perfectly bias directed acyclic graphs, however we cannot bias generic directed graphs. The proof follows this line of reasoning. In a DAG, we have a system of n equations and n unknowns. Such a system must have at least one solution, which shows that we can compute the optimal bias for the given problem. When we have at least one cycle in the problem, we get an infinitely large set of equation and n unknowns, which is not solvable.

Let the following terms be defined. $B^*(path)$ is the optimal bias of a path. $P(path)$ is the probability of a path. $paths$ refers to all paths in the DAG or digraph. $K^*(path) = \frac{P(path)}{B^*(path)}$ is the optimal computation of a sample. α multiplicative error in the bias. γ is the error induced by breaking samples. Let a_i where i ranges from 0 to the length of the path be the i$^{\text{th}}$ action along a path. $P(a_i)$ is the unbiased probability of that action, and $B^*(a_i)$ is the optimal bias of that action. It should be clear that:

$$P(path) = \Pi_{i=0}^{|path|} P(a_i), \tag{6}$$

$$B^*(path) = \Pi_{i=0}^{|path|} B^*(a_i), \tag{7}$$

$$K^*(path) = \Pi_{i=0}^{|path|} \frac{P(a_i)}{B^*(a_i)}. \tag{8}$$

Thus, if there exists an optimal per-action bias $B^*(a_i)$ for all actions a_i in the graph, then we can compute an optimal bias for all paths in the graph. If B^* was known, we could construct a system of equations and unknowns so that we could attempt to solve for each $B^*(a_i)$. In an acyclic graph, there cannot be more paths than there are actions. Thus, the system is under constrained and there exists one or more solutions. However, as soon as a cycle exists, there are infinitely many equations, a fixed number of unknowns, and there is no solution to the problem.

Optimal DAG Bias The DAG case covers two special kinds of problems that we might want to use importance sampling on. The first is, of course, directed acyclic graphs. The second is any graph for which we care about events within a finite horizon. So long as the problem has a finite horizon, we can compile any arbitrary graph down into a DAG by unrolling all of the paths up to the length of the horizon and rewriting that as a new DAG.

For ease of the proof, we assume that the graphs in question has special structure. Namely, they have a single initial state and a single sink state representing the event of interest. Without loss of generality, any DAG can have this structure. To do this we simply add a new initial state and create actions to each original initial state (with action probabilities based on the probabilities of the initial states themselves), and create an action from each state of interest to a single new sink state (these will be deterministic).

Theorem. *An optimal bias can be described when performing importance sampling on a DAG with probabilities and biases ascribed to single edges in the DAG.*

Lemma 1. *Each path in a DAG has at least one edge which only exists in that path.*

Lemma 2. *A DAG contains at most $|E|$ unique paths through it, where E is the set of all edges in the DAG.*

Lemma 3. *DAGs of interest for importance sampling also contain at least one path through them.*

Proof. Let D be a DAG consisting of the vertexes V and edges E. We will say that the edges have probability a_i, where i represents the i$^{\text{th}}$ member of E. The biases for importance sampling will be described similarly, having bias

b_i for the i$^{\text{th}}$ edge. We will find it useful to discuss the paths through the DAG to the event of interest, and we will refer to them as p being members of P. Let K be the actual probability of the event of interest.

The DAG represents multiple possible events in sequence, some of which we are interested in. In fact, without loss of generality we can remove all edges from the DAG which cannot lead to an event of interest. As a result, the sum of the weights on all edges in the DAG is equal to the probability of the event in question:

$$\Sigma_{p \in P} \Pi_{a_i inp} a_i = K \tag{9}$$

The optimal bias for the DAG can be computed as the solution to a set of linear equations. In fact, each $p \in P$ adds an equation to the system:

$$\Pi_{i \in p} \frac{a_i}{b_i} = K \tag{10}$$

where $i \in p$ represents the inclusion of edge i in the path p. Essentially, the product of all probabilities divided by the product of all biases must give the actual event probability K. This is exactly the definition of the optimal bias.

Additionally, we must add the following constraints:

$$\Sigma_{p \in P} \Pi_{i \in p} b_i = 1 \tag{11}$$

which simply says that in the biased DAG, all probability mass leads to the event of interest, another requirement of the optimal bias.

We have now described a system of $|P| + 1$ equations and $|E|$ unknowns (recall that a_i is given by the DAG). By lemma 1 we know that each path contains at least one unique edge. Because the edges exist on a path, they are naturally ordered. As a result of the ordering and their existence, there must be a first unique edge along a path, which we will refer to as u_p, and the collection of all such edges will be referred to as U.

We can set the bias of ever edge not in U to be exactly the weight of the edge:

$$\forall_{b_i \in E-U} b_i = a_i \tag{12}$$

This reduces Equations 10 to just finding the bias for the first unique edge in each path, as the rest of the product will be equal to 1 by Equation 12. In fact, the bias associated with each first unique can be computed as:

$$\frac{u_p}{b_p} = K \tag{13}$$

$$\frac{1}{b_p} = \frac{K}{u_p} \tag{14}$$

$$b_p = \frac{u_p}{K} \tag{15}$$

Such an assignment guarantees that any given trace through the DAG will produce a ratio that is exactly the probability of the event. Effectively, we have reduced our system of $|P| + 1$ equations and $|E|$ unknowns to a system of $|P| + 1$ equations and $|P|$ unknowns. A solution, should one exist, gives us a zero variance estimator, satisfying the requirements of the optimal bias. It remains to be shown that this over-constrained system is solvable. We will show this by showing that the constraint expressed in Equation 11 is simply a linear combination of the Equations from 10:

Equation 10 be rewritten as:

$$\forall_{p \in P} \Pi_{b_i \in p} b_i = \frac{\Pi_{a_i \in p} a_i}{k} \tag{16}$$

Starting with a linear combination of these $|P|$ equations, we will now derive Equation 11, thereby showing that this equation is redundant and that the system is not over constrained:

$$\begin{aligned}
\Sigma_{p \in P} \Pi_{b_i \in p} b_i &= \Sigma_{p \in P} \frac{\Pi_{a_i \in p} a_i}{K} && (17)\\
&= \frac{1}{K} \cdot \Sigma_{p \in P} \Pi_{a_i \in p} a_i && (18)\\
&= \frac{1}{K} \cdot K && (19)\\
&= 1 && (20)
\end{aligned}$$

17 is a linear combination of equations. 18 follows since K is not dependent on p. 19 comes from equation 9, and 20 is algebra.

So for any given DAG from which we might sample rare events, there exists a set of equations which, when solved, will provide an optimal importance sampling bias to the problem. This set of equations is not over constrained, so it must contain a solution. □

Current Work

We are currently working on several improvements to this work. First, we are in the process of conducting experiments using P-CIRCA directly to empirically verify the analytical results reported here. Secondly, we are generalizing the results to state spaces with cycles of arbitrary length, while simultaneously developing a heuristic graph expansion algorithm for doing more intelligent expansion of those portions of the state space. We are are also proving formal properties of this heuristic DAGification algorithm to analytically derive relationships between computation required for further graph expansion, and a corresponding computation savings in verification sampling.

Conclusion

We are interested in the issue of trust from the perspective of a planning agent making probabilistic safety guarantees over a finite time span when exogenous events can lead to failure. Due to complex timing interactions, a sampling-based verifier is used to test the safety claims made; however, if failure events are rare, the number of samples required to verify a given hypothesis can be prohibitive. We describe an approach based on importance sampling (IS) to greatly reduce the number of samples required, and describe a graph expansion algorithm that improves the initial IS biases upon which successful IS depends with relatively modest computation. Finally, we provide an analysis of the improvement gained, and briefly describe current work to test this analysis empirically and to extend this work to more complex probabilistic models.

Acknowledgments

This work was supported by both a Mellon Grant for student research at Lawrence University and by the Office of Naval Research grant N00014-10-1-0188 via subaward from Carnegie Mellon University. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Lawrence University, ONR, the U.S. Government, or CMU.

References

Clarke, E. M., and Zuliani, P. 2011. Statistical model checking for cyber-physical systems. In *Proceedings of the 9th international conference on Automated technology for verification and analysis*, ATVA'11, 1–12. Berlin, Heidelberg: Springer-Verlag.

Hammersley, J. M., and Handscomb, D. C. 1964. *Monte Carlo Methods*. London & New York: Chapman and Hall.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6):1561–1574.

Rubinstein, R., and Davidson, W. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability* 2:127–190.

Wald, A. 1945. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics* 16(2):pp. 117–186.

Younes, H. L. S., and Musliner, D. J. 2002. Probabilistic plan verification through acceptance sampling. In *In Proc. AIPS 2002 Workshop on Planning via Model Checking*, 0–7695. AAAI Press.

Younes, H. L. S.; Musliner, D. J.; and Simmons, R. G. 2003. A framework for planning in continuous-time stochastic domains. In *Proc. Thirteenth International Conference On Automated Planning And Scheduling*, 195–204. AAAI Press.