APROPOS^2: Approximate Probabilistic Planning Out of Stochastic Satisfiability

Stephen M. Majercik
Department of Computer Science
Bowdoin College
Brunswick, ME 04011-8486
smajerci@bowdoin.edu

Abstract

The probabilistic contingent planner ZANDER (Majercik 2000) operates by converting the planning problem to a stochastic satisfiability problem and solving that problem instead. Although ZANDER can solve some simple standard test problems more efficiently than three alternative approaches to probabilistic planing, ZANDER is currently confined to small problems. We introduce APROPOS^2, a probabilistic contingent planner based on ZANDER that produces an approximate contingent plan and improves that plan as time permits. APROPOS^2 does this by considering the most probable situations facing the agent and constructing a plan, if possible, that succeeds under those circumstances. Given more time, less likely situations are considered and the plan is revised if necessary. In some cases, a plan constructed to address a relatively low percentage of possible situations will succeed for situations not explicitly considered as well, and may return an optimal or near-optimal plan. This means that APROPOS^2 can sometimes find optimal plans faster than ZANDER. And the anytime quality of APROPOS^2 means that suboptimal plans could be efficiently derived in larger time-critical domains where ZANDER might not have time to calculate the optimal plan. We describe some preliminary experimental results and suggest further work needed to bring APROPOS^2 closer to attacking real-world problems.

Introduction

Previous research (Majercik 2000) has extended the planning-as-satisfiability paradigm to support probabilistic contingent planning. This extension, called ZANDER, is based on *stochastic satisfiability* (SSAT) (Littman, Majercik, & Pitassi 2001), a type of Boolean satisfiability problem in which some of the variables have probabilities attached to them.

ZANDER solves probabilistic propositional planning problems. A subset of the state variables is declared *observable*, meaning that any action can be made contingent on any of these variables. This scheme can express various levels of observability. ZANDER, however, is not able to scale up to large, real-world problems. Majercik (2000) described a number of improvements to ZANDER that showed promise for scaling up to larger problems; e.g. more efficient SSAT encodings and encoding domain specific knowledge. We feel, however, that these types of improvements are unlikely to provide the leverage needed to scale up to large problems. Some type of approximation technique will be a necessary step on the road toward solving real-world probabilistic planning problems.

APROPOS^2, a contribution toward that goal, is a probabilistic contingent planner based on ZANDER, that produces an approximate contingent plan and improves that plan as time permits. APROPOS^2 does this by considering the most probable situations facing the agent and constructing a plan, if possible, that succeeds under those circumstances. Given more time, less likely situations are considered and the plan is revised if necessary.

Many other researchers have explored the possibility of using some type of approximation to speed the planning process. Drummond & Bresina (1990) proposed a system—"anytime synthetic projection"—in which a set of control rules establish a base plan which has a certain probability of achieving the goal. The probability of achieving the goal is incrementally increased, time permitting, by identifying failure situations that are likely to be encountered by the current plan and synthesizing additional control rules to handle these situations, thereby increasing the probability of success.

MAHINUR (Onder & Pollack 1997) is a probabilistic partial-order planner that also creates a base plan with some probability of success and then improves that plan. MAHINUR identifies those contingencies whose failure would have the greatest negative impact on the plan's success and focuses its planning efforts on generating plan branches to deal with those contingencies.

Exploring approximation techniques in Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs) has been a very active area of research in recent years. Boutilier & Dearden (1996) represent value functions using decision trees and prune these decision trees so that the leaves represent ranges of values, thereby approximating the value function produced. Koller & Parr (1999) have presented evidence that the value function of a factored MDP can often be well approximated using a factored value function; *i.e.* a linear combination of restricted basis functions, each of which refers to only a small subset of variables in the MDP. They show that this value function approximation technique can be used as a subroutine in a policy itera-

tion process to solve factored MDPs (Koller & Parr 2000). Kearns, Mansour, & Ng (1999) describe a method for choosing, with high probability, approximately optimal actions in an infinite-horizon discounted Markov decision process using truncated action sequences and random sampling. Zhang & Lin (1997) transform a POMDP into a simpler *region observable* POMDP in which it is assumed an oracle tells the agent what region its current state is in. This POMDP is easier to solve and they use its solution to construct an approximate solution for the original POMDP.

In the next section we briefly describe ZANDER. We then describe the APROPOS^2 algorithm for approximate planning and describe some preliminary experimental results. We conclude with some suggestions for further work.

ZANDER

In this section, we provide a brief overview of ZANDER. Details are available elsewhere (Majercik 2000). ZANDER works on partially observable probabilistic propositional planning domains consisting of a finite set of distinct *propositions*, any of which may be `true` or `false` at any discrete time t. A *state* is an assignment of truth values to these propositions. A possibly probabilistic *initial state* is specified by a set of decision trees, one for each proposition. *Goal states* are specified by a partial assignment to the set of propositions; any state that extends this partial assignment is a goal state. Each of a finite set of *actions* probabilistically transforms a state at time t into a state at time $t + 1$ and so induces a probability distribution over the set of all states. A subset of the set of propositions is the set of *observable propositions*. The task is to find an action for each step t as a function of the value of observable propositions for steps before t and that maximizes the probability of reaching a goal state.

ZANDER uses a propositional problem representation called the sequential-effects-tree representation (ST), which is a syntactic variant of two-time-slice Bayes nets (2TBNs) with conditional probability tables represented as trees (Majercik 2000). In the ST representation, each action a is represented by an ordered list of decision trees, the effect of a on each proposition represented as a separate decision tree. This ordering means that the tree for one proposition can refer to old *and* new values of previous propositions, thus allowing the effects of an action to be correlated. The leaves of a decision tree describe how the associated proposition changes as a function of the state and action, perhaps probabilistically.

ZANDER converts the ST representation of the problem into a *stochastic satisfiability* (SSAT) problem. An SSAT problem is a satisfiability (SAT) problem, assumed to be in conjunctive normal form, with two types of Boolean variables—termed *choice* variables and *chance* variables (Majercik 2000)—and an ordering specified for the variables. A choice variable is like a variable in a regular SAT problem; its truth value can be set by the planning agent. Each chance variable, on the other hand, has an independent probability associated with it that specifies the probability that that variable will be `true`.

These probabilities, in the general case, make it impossible to satisfy the SSAT formula with certainty, and the challenge is to select values for the choice variables that maximize the probability of satisfaction. This can be complicated by the ordering of the variables. In the simplest type of SSAT problem containing both types of variables, all of the choice variables come first in the ordering, and the problem is to select values of the choice variables such that the probability of getting a satisfying assignment is maximized with respect to the chance variables. But, in the type of SSAT problem used by ZANDER to encode contingent planning problems, groups of choice variables and chance variables alternate in the variable ordering.

Choice variables can be thought of as being existentially quantified—we must pick a single, best value for such a variable—while chance variables can be thought of as "randomly" quantified—they introduce uncontrollable random variation which, in general, makes it more difficult to find a satisfying assignment. So, for example, an SSAT formula with the ordering $\exists v R w \exists x R y \exists z$ asks for values of v, x (as a function of w), and z (as a function of w and y) that maximize the probability of satisfaction given the independent probabilities associated with w and y. This dependence of choice variable values on the earlier chance variable values in the ordering allows ZANDER to map contingent planning problems to stochastic satisfiability.

The variables in an SSAT plan encoding fall into three segments (Majercik 2000): a plan-execution history, the domain uncertainty, and the result of the plan-execution history given the domain uncertainty. The plan-execution-history segment is an alternating sequence of choice-variable blocks (one for each action choice) and chance-variable blocks (one for each set of possible observations at a time step). The domain uncertainty segment is a single block containing all the chance variables that modulate the impact of the actions on the observation and state variables. The result segment is a single block containing all the non-observation state variables. Essentially, ZANDER must find an assignment *tree* that specifies the optimal action choice-variable assignment given all possible settings of the observation variables (Majercik 2000).

The solver does a depth-first search of the tree of all possible truth assignments, constructing a solution subtree by calculating, for each variable node, the probability of a satisfying assignment given the partial assignment so far. For a choice variable, this is the maximum probability of its children. For a chance variable, the probability will be the probability weighted average of the success probabilities for that node's children. The solver finds the optimal plan by determining the subtree that yields the highest probability at the root node.

ZANDER uses unit propagation (assigning a variable in a unit clause—a clause with a single literal—its forced value) and, to a much lesser extent, pure variable assignment (assigning the appropriate truth value to a choice variable that appears only positively or only negatively) to prune subtrees in this search. Also, although the order in which variables are considered is constrained by the SSAT-imposed variable ordering, where there is block of similar (choice

or chance) variables with no imposed ordering, ZANDER considers those with the earliest time index first. This TIME_ORDERED heuristic takes advantage of the temporal structure of the clauses induced by the planning problem to produce more unit clauses. ZANDER also uses dynamically calculated success probability thresholds to prune branches of the tree. (e.g. if the best possible success probability of a particular branch is less than the current best success probability). We are currently working on incorporating learning to improve ZANDER's performance.

APROPOS^2

Before we describe APROPOS^2 it is worth looking at a previous approach to approximation in this framework (Majercik 2000). This approach illuminates some of the problems associated with formulating an approximation algorithm in this framework and explains some of the choices we made in developing APROPOS^2.

Majercik (2000) describes an algorithm called RANDEVALSSAT that uses stochastic local search in a reduced plan space. RANDEVALSSAT uses random sampling to select a subset of possible chance variable instantiations (thus limiting the size of the contingent plans considered) and stochastic local search to find the best size-bounded plan.

As Majercik (2000) point out, there are two problems with this approach. First, since chance variables are used to describe observations, a random sample of the chance variables describes an observation sequence as well as an instantiation of the uncertainty in the domain, and the observation sequence thus produced may not be *observationally consistent*, and these inconsistencies can make it impossible to find a plan, even if one exists. Second, this algorithm returns a partial policy, that specifies actions only for those situations represented by paths in the random sampling of chance variables.

APROPOS^2 addresses these two problems by:

- designating each observation variable as a special type of variable, termed a *branch* variable, rather than a chance variable, and
- evaluating the approximate plan's performance under all circumstances, not just those used to generate the plan.

The introduction of branch variables violates the pure SSAT form of the plan encoding, but is justified, we think, for the sake of conceptual clarity. We could achieve the same end in the pure SSAT form by making observation variables chance variables (as Majercik (2000) does), and not including them when the possible chance-variable assignments are enumerated. But, rather than taking this circuitous route, we have chosen to acknowledge the special role played by observation variables; these variables indicate a potential branch in a contingent plan (hence the name). As such, the value of an observation variable node in the assignment tree described above is the *sum* of the values of its children. This introduces a minor modification into the ZANDER approach and has the benefit of clarifying the role of the observation variables.

The operation of APROPOS^2 can be summarized as follows:

1. *Enumerate* some fraction of the possible assignments to the chance variables starting with the most probable and continuing in decreasing order of probability.
2. *Calculate*, for each chance-variable assignment, all the action-observation paths that are consistent with the chance variable assignment (lead to a satisfying assignment).
3. *Increase* the weight of each of the consistent action-observation paths by the probability of the chance-variable assignment.
4. *Extract* the best plan by finding the tree of action-observation paths that has the highest weight and whose observations are consistent (note that this weight is a lower bound on the true probability of success of the plan embodied in the tree).
5. *Evaluate* the full assignment tree given the extracted plan in order to determine the true success probability of the plan.

This basic framework is easily modified to produce an *anytime* algorithm. One way to think about this is that we are incrementally constructing the optimal action-observation tree by building up the probabilities as we process the possible instantiations of the chance variables. We can stop the approximation process after any number of chance variable assignments have been considered and extract and evaluate the best plan for the chance-variable assignments that have been considered so far. If the probability of success of this plan is sufficient (probability 1.0 or exceeding a user-specified threshold), the algorithm halts and return the plan and probability; otherwise, APROPOS^2 continues processing chance variable assignments. Note also that the probability of success of the just-extracted plan can now be used as a new lower threshold in subsequent plan evaluations, often allowing additional pruning to be done. The quality of the plan produced increases (if the optimal success probability has not already been attained) with the available computation time.

Because the chance variable instantiations are investigated in descending order of probability, it is likely that a plan with a relatively high percentage of the optimal success probability will be found quickly. One of the exceptions here is the situation in which the high probability situations are hopeless and the best that can be done is to construct a plan that addresses some number of lower probability situations. Even here, the basic SSAT heuristics used will allow APROPOS^2 to quickly discover that no plan is possible for the high-probability situations, and lead it to focus on the low-probability situations for which a plan is feasible. Of course, if *all* chance-variable assignments are considered, the plan extracted is the optimal plan, but, as we shall see, the optimal plan may sometimes be produced even after only a relatively small fraction of the chance-variable assignments have been considered.

This approach may seem counter-intuitive. Instead of instantiating *all* the chance variables, wouldn't it make

more sense to instantiate chance variables based on the instantiation of variables (particularly action variables) at previous times steps? For example, given the action choice made at the first time step, what is the most likely instantiation of chance variables facing the agent now? This type of approach mirrors the operation of ZANDER much more closely—searching the SSAT tree of possible assignments—except that the search is speeded up by considering only the more likely chance variable instantiations. This would seem to require a significant amount of overhead to keep track of which choices have been made and, in most cases, would entail the repeated solving of a number of SSAT subproblems with one or more chance variable settings changed.

The APROPOS^2 approach seeks to accomplish a similar goal in a different way. By enumerating complete instantiations of the chance variables in descending order of probability, APROPOS^2 is choosing the most likely outcomes of all actions at all time steps. Because it is not taking variable independencies into account, it does so somewhat inefficiently. At the same time, however, by instantiating all the chance variables at the same time, APROPOS^2 reduces the SSAT problem to a much simpler SAT problem. Although this approach will also entail the repeated solving of a number of subproblems with one or more chance variable settings changed, the conjecture is that solving a large number of SAT problems will take less time than solving a large number if SSAT problems. Obviously, this will depend on the relative number of problems involved, but we have chosen to explore the approach embodied in APROPOS^2 first.

In the current implementation of APROPOS^2, the user specifies the total number of chance-variable assignments to be considered, the interval at which the current plan should be extracted and evaluated (the default is 5% of the total number of chance-variable assignments being considered), and optional lower and upper success probability thresholds. If the algorithm finds a plan that meets or exceeds the upper success probability threshold, it halts and returns that plan.

All of the operations in APROPOS^2 can be performed as or more efficiently than the operations necessary in the ZANDER framework.

1. *Enumerate*: The chance variable instantiations can be generated in descending order in time linear in the number of instantiations using a priority queue.
2. *Calculate*: Finding all consistent action-observation paths amounts to a depth-first search of the assignment tree checking for satisfiability using pruning heuristics (the central operation of ZANDER). Note also that once an action-observation path is instantiated, checking whether it can be extended to a satisfying assignment amounts to a series of fast unit propagations.

   In fact, once the chance variables have all been set, the remaining variables are all choice variables and the search for all action-observation paths that lead to satisfying assignments can be accomplished by any efficient SAT solver that finds all satisfying assignments. In particular, this opens up the possibility for using a better splitting heuristic. We tested a number of different heuristics: choosing a variable randomly, finding the literal that satisfies the most clauses in the current formula, 2-sided Jeroslow-Wang (Hooker & Vinay 1995), positive, 2-sided Jeroslow-Wang (Hooker & Vinay 1995), choosing the variable that has "Maximum Occurrences in clauses of Minimum Size" (MOMS) and MAX_UNIT (Bayardo & Schrag 1997), which is similar to heuristics used by the successful POSIT (Freeman 1995) and TABLEAU (Crawford & Auton 1996) solvers, preferring variables that will lead to a maximal number of unit propagations. Preliminary tests indicate that the TIME_ORDERED heuristic used by ZANDER (Majercik 2000) usually performs better than any of these heuristics, probably because this ordering takes advantage of the structure provided by the temporal ordering of the clauses and induces more unit propagations (i.e. setting the variables at time step t tends to force the values of variables at time step $t+1$)
3. *Increase*: A simple addition.
4. *Extract*: A simple depth-first search of the action-observation tree suffices; this is sped up further by the fact that satisfiability does not have to be checked (having been taken care of in the *calculate* phase).
5. *Evaluate*: A depth-first search of the entire assignment tree is necessary, but heuristics speed up the search, and the resulting probability of success can be used as a lower threshold if the search continues, thus possibly speeding up subsequent computation.

Results

Preliminary results are mixed but indicate that APROPOS^2 has some potential as an approximation technique. In some cases, it outperforms ZANDER, in spite of the burden of the additional approximation machinery. And, in those cases, where its performance is poorer, there is potential for further performance improvements (see Further Work).

We tested APROPOS^2 on three domains that ZANDER was tested on (Majercik 2000). The HUNGRY-TIGER problem contains uncertain initial conditions and a noisy observation; the agent needs the entire observation history in order to act correctly. The COFFEE-ROBOT problem is a larger problem (7 actions, 2 observation variables, and 8 state propositions in each of 6 time steps) with uncertain initial conditions, but perfect causal actions and observations. Finally, the GENERAL-OPERATIONS problem has no uncertainty in the initial conditions, but requires that probabilistic actions be interleaved with perfect observations. All experiments were conducted on an 866 MHz Dell Precision 620 with 256 Mbytes of RAM, running Linux 7.1.

In the 4-step HUNGRY-TIGER problem, ZANDER found the optimal plan (0.93925 probability of success) in 0.01 CPU seconds. APROPOS^2 requires 0.42 CPU seconds to find the same plan (extracting and evaluating the current plan after every 5% of chance variable instantiations). This is, however, if we insist on forcing APROPOS^2 to look for the best possible plan (and, thus, to process all 512 chance variable instantiations), which seems somewhat out of keeping with the notion of APROPOS^2 as an approximation technique. If we run APROPOS^2 on this problem

Domain	Number of Chance Variable Instantiations	Time in CPU Seconds	Probability of Plan Success
4-step HUNGRY-TIGER	1	0.0	0.307062
	2	0.0	0.614125
	3	0.0	0.614125
	4	0.0	0.668312
	5	0.01	0.668312
	6	0.01	0.722500
	7	0.01	0.722500
	8	0.01	0.722500
	9	0.01	0.776687
	10	0.01	0.776687
	11	0.01	0.830875
	12	0.01	0.830875
	13	0.01	0.885062
	14	0.01	0.885062
	15	0.01	0.885062
	16	0.02	0.885062
	17	0.02	0.885062
	18	0.02	0.939250
6-STEP COFFEE-ROBOT	1	2.24	0.5
	2	4.98	0.5
	3	9.12	1.0
	4	15.07	1.0
7-STEP GENERAL-OPERATIONS	1	1.06	0.125
	2	1.20	0.125
	3	1.51	0.125
	4	1.74	0.125
	5	1.98	0.125
	6	2.17	0.125
	7	2.47	0.125
	8	2.67	0.125
	9	2.92	0.125
	10	3.07	0.125
	11	3.36	0.1875
	12	3.62	0.1875
	13	3.83	0.1875
	14	4.03	0.1875
	15	4.26	0.1875
	16	4.47	0.1875
	17	4.83	0.1875
	18	4.97	0.1875
	19	5.16	0.25
	20	5.44	0.25

Table 1: The probability of success increases with the number of chance variable instantiations considered.

under similar assumptions, but specify a success probability threshold of 0.90 (we will accept any plan with a success probability of 0.90 or higher), APROPOS^2 returns a plan in 0.02 CPU seconds. The plan returned is, in fact, the optimal plan, and is found after examining the first 18 chance variable instantiations.

Table 1 provides an indication of what kind of approximation would be available if less time were available than what would be necessary to compute the optimal plan. This table shows how computation time and probability of plan success increases with the number of chance variable instantiations considered until the optimal plan is reached at 18 chance-variable instantiations.

The 6-step COFFEE-ROBOT problem provides an interesting counterpoint to the HUNGRY-TIGER problem in that APROPOS^2 does better than ZANDER. ZANDER is able to find the optimal plan (success probability 1.0) in 19.34 CPU seconds, while APROPOS^2 can find the same plan in 9.12 CPU seconds. There are only 4 chance variable instantiations in the COFFEE-ROBOT problem and, since extraction and evaluation of the plan at intervals of 5% would result in intervals of less than one, the algorithm defaults to extracting and evaluating the plan after each chance variable instantiation is considered. Although one might conjecture that this constant plan extraction and evaluation is a waste of time, in this case it leads to the discovery of an optimal plan (success probability of 1.0) after processing the first 3 chance variable instantiations, and the resulting solution time of 9.12 CPU seconds (including plan extraction and evaluation time) is less than the solution time if we force APROPOS^2 to wait until all four chance variable instantiations have been considered before extracting and evaluating the best plan (15.07 CPU seconds).

This illustrates an interesting tradeoff. In the latter case, although APROPOS^2 is not "wasting" time extracting and evaluating the plan after each chance variable instantiation (doing it once instead of three times), it does an extra chance variable instantiation, and this turns out to take more time than the extra plan extraction and evaluations. This is not surprising since checking a chance variable instantiation involves solving a SAT problem to find all possible satisfying assignments, while extracting and evaluating the plan is straightforward depth-first search. This suggests that we should be biased toward more frequent plan extraction and evaluation; more work is needed to determine if some optimal frequency can be automatically determined for a given problem. Table 1 provides an indication of how computation time and probability of plan success increases with the number of chance variable instantiations considered for the COFFEE-ROBOT problem. Interestingly, although the probability mass of the chance variables is spread uniformly across the four chance variable instantiations, APROPOS^2 is still able to find the optimal plan without considering all the chance variable instantiations.

The 7-step GENERAL-OPERATIONS problem shows us that this not necessarily the case, especially when, as in the GENERAL-OPERATIONS problem, the probability mass is spread uniformly over many more ($2^{21} = 2097152$) chance variable instantiations. In this problem, ZANDER is

able to find the optimal plan (success probability 0.773437) in 2.48 CPU seconds. Because of the large number of chance variable instantiations to be processed, APROPOS^2 cannot even approach this speed. APROPOS^2 needs about 566 CPU seconds to process only 3000 (0.14%) of the total chance variable instantiations, yielding a plan with success probability of 0.648438. Table 1 provides an indication of how computation time and probability of plan success increases with the number of chance variable instantiations considered for the GENERAL-OPERATIONS problem.

As the size of the problem increases, however, to the point where ZANDER might not be able to return an optimal plan in sufficient time, APROPOS^2 may be useful if it can return *any* plan with some probability of success in less time than it would take ZANDER to find the optimal plan. We tested this conjecture on the 10-step GENERAL-OPERATIONS problem ($2^{30} = 1073741824$ chance variable instantiations). Here, ZANDER needed 405.35 CPU seconds to find the optimal plan (success probability 0.945313). APROPOS^2 was able to find a plan in somewhat less time (324.92 CPU seconds to process 20 chance variable instantiations), but this plan has a success probability of only 0.1875.

Further Work

We need to improve the efficiency of APROPOS^2 if it is to be a viable approximation technique, and there are a number of techniques we are in the process of implementing that should help us to achieve this goal. First, we are implementing an incremental approach: every time a new action-observation path is added, APROPOS^2 would incorporate that path into the current plan, checking to see if it changes that plan by checking values stored in that path from that point to the root. Whenever this process indicates that the plan has changed, the plan extraction and evaluation process will be initiated.

Second, when APROPOS^2 is processing the chance variable instantiations in descending order, in many cases the difference between two adjacent instantiations is small. We can probably take advantage of this to find the action-observation paths that satisfy the new chance variable instantiation more quickly.

Third, since we are repeatedly running a SAT solver to find action-observation paths that lead to satisfying assignments for the chance-variable assignments, and since two chance variable assignments will frequently generate the same satisfying action-observation path, it seems likely that we could speed up this process considerably by incorporating learning into APROPOS^2.

Finally, we are investigating whether plan simulation (instead of exact calculation of the plan success probability) would be a more efficient way of evaluating the current plan.

References

Bayardo, Jr., R. J., and Schrag, R. C. 1997. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 203–208. AAAI Press/The MIT Press.

Boutilier, C., and Dearden, R. 1996. Approximating value trees in structured dynamic programming. In Saitta, L., ed., *Proceedings of the Thirteenth International Conference on Machine Learning*.

Crawford, J. M., and Auton, L. D. 1996. Experimental results in the crossover point in random 3SAT. *Artificial Intelligence* 81(1-2):31—57.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 138–144. Morgan Kaufmann.

Freeman, J. W. 1995. *Improvements to propositional satisfiability search algorithms*. Ph.D. Dissertation, Department of Computer and Information Science, University of Pennsylvania.

Hooker, J. N., and Vinay, V. 1995. Branching rules for satisfiability. *Journal of Automated Reasoning* 15(3):359–383.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, 1324–1231.

Koller, D., and Parr, R. 1999. Computing factored value functions for policies in structured MDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1332–1339. The AAAI Press/The MIT Press.

Koller, D., and Parr, R. 2000. Policy iteration for factored MDPs. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 2000)*.

Littman, M. L.; Majercik, S. M.; and Pitassi, T. 2001. Stochastic Boolean satisfiability. *Journal of Automated Reasoning* 27(3):251—296.

Majercik, S. M. 2000. *Planning Under Uncertainty via Stochastic Satisfiability*. Ph.D. Dissertation, Department of Computer Science, Duke University.

Onder, N., and Pollack, M. E. 1997. Contingency selection in plan generation. In *Proceedings of the Fourth European Conference on Planning*.

Zhang, N. L., and Lin, W. 1997. A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research* 7:199–230.