Adaptive Sampling Applied to Planning with Hindsight Optimization

Alex Wallace
Department of CSE
Arizona State University
Tempe, AZ 85281
alex.wallace@asu.edu

Abstract

Generating plans in stochastic planning enviironments is a difficult and time consuming process even with state of the art planners such as FF-Hindsight. This research applies a series of adaptive sampling methods around FF-Hindsight to improve the quality of plans returned by the planner as well as allowing the planner to modify the time it takes to make a decision on an action and base it on the diffculty of the problem. By creating a pool of samples and distrbuting these based on a variety of selection methods, the planner can be more capable of solving a variety of planning problems.

Introduction

With the exploration of probabilistically interesting domains after the unexpected success of FF-Replan in IPPC-04, it has become obvious that the addition of probability to deterministic problems generally yields uninteresting domains [2,6]. Indeed, FF-Replan's success varies across these new and improved domains. In the last bi-annual International Probabilistic Planning Competition, IPPC-08, domains were derived that tried to implement two basic ideas that can generate interesting problems for planners: causing simple paths to the goal highly likely to lead to dead ends and adding optional steps on top of simple paths to make them more rewarding. To overcome these more challenging problems FF-Hindsight reevaluated FF-Replan's approach and sought to improve it with a form of dynamic determinization. FF-Hindsight randomely generates a set of non-stationary determinized problems, unlike FF-Replan's single stationary determinization, and solves each one, combining their solutions [8]. This paper builds on FF-Hindsight by allowing the planner to apply samples to specific actions and determine how many samples are necessary until execution can continue. This approach is important because it will allow the planner to change its operation based on the difficulty of each stage of a problem and can improve the speed and quality of the planner. Other selective and directed sampling methods in other domains have been tested showing improvement over other alternative methods [5, 3].

The Problem

A planning problem is modelled as a Markov decision process, a four-tuple, (S, A, $P_a(s, s')$, $R_a(s, s')$) where:

1. S is a finite set of states
2. A is a finite set of actions
3. $P_a(s, s')$ is the transition function representing the probability that an action *a* at time *t* will lead from state *s* to *s'*, $Pr(s_t+1 = s' \mid s_t+1 = s, a_t = a)$
4. $R_a(s, s')$ is the expected immediate reward received after transitioning from s to s' via action a

The work in this paper represents the transition function $P_a(s, s')$ using the probabilistic planning domain description language (PPDDL) [6]. This allows an action *a* in A to be represented using a set preconditions, a set of effects and a discrete distribution, D_a, over these effects. An action a can be executed when its preconditions are met. After execution of the action *a*, the next state is determined by drawing a set of the effects of a based on D_a and then applying these effects to the current state.

The general objetive of a planning problem is to maximize some expectation of reward in the future based on a set of actions over time, the plan. This measure of reward can be cumulative or discounted depending on the domain. In this paper the objective is to maximize the probability of reaching a goal state(s) after executing a series of actions from some initial state. A secondary objective is to minimize the plan length required to achieve a goal state. Minimizing the plan length has practical application when each action has no cost of execution or all actions have the same execution cost. In this case, minimizing the plan length represents an improvement in efficiency as it reduces the cost of plan execution.

Online Planning

Before describing the planners the work this paper builds on it is important to describe online and offline planning. Online planning is a method that simply tries to determine which action among the set of actions available to execute at some state, *s,* that will maximize the expected future reward. When one is found it executes this action. By contrast, offline planning determines an entire plan or policy that will maximize the expected future reward and subsequently executes the entire plan with no further planning necessary. Both of the planners referenced in this paper, FF-Replan and FF-Hindsight, are online planners and the work described in this paper pertains to online planning. This is not to say that this work is not applicable to offline planning, however, these applications will not be described here.

Replanning and Hindsight Optimization

FF-Replan, despite its simplistic approach to planning, was the winner of the 2004 IPPC and a top performer in IPPC-06 domains. Its method of planning involves constructing a determinized version of the probabilistic domain, solving this problem using Fast Forward (FF) and executing the plan returned by FF until some unexpected outcome occurs. When this happens, as implied by the name, FF-Replan calls FF from the current state and repeats the process creating a cycle of replanning.

FF-Hindsight operates in a similar manner but instead of creating one specific static and determinized version of the problem space, FF-Hindsight generates multiple non-stationary versions. These determinized domains can be seen as potential outcomes that may occur during execution, thus, they are aptly named 'futures'. The futures are independently solved using FF and then combined to form a heuristic that informs the planner on which action to execute next. The integration of plans returned by FF can occur in many ways. FF-Hindsight chooses the action that leads to the highest probability of reaching the goal by selecting the action whose futures return the most complete plans.

In this paper, we seek to improve on FF-Hindsight's ability to identify promising actions at each step and potentially reduce the time required to identify these actions as well through adaptive sampling.

Adaptive Sampling

Adaptive sampling, in this application, is deliberately varying the number of futures that are generated for each action in response to changing heuristic values after each sample. In doing so there is the potential to require less samples per step to determine which action will maximize the expected future reward relative to even sampling for each possible action. There is also the potential to require significantly more samples for actions that are highly variable in their results, however, this will hopefully be balanced by the identification of actions with simpler futures, actions that do not largely vary in their reward output in the futures generated.

This technique can also have the ability to identify better solutions to problems and domains. When highly variable reward values are encountered after samples are taken then it is important to get a better understanding of the action's many potential futures before making a decision. Thus, more futures will be generated for the potential action and the deterministic planner would return a larger amount of solutions or failures. The more solutions that are found for futures based on an action, the more certain the planner can be of the heurisitc values of that action. Since the heuristic's quality is most likely to increase with more sampling, the planner can make more informed decisions and achieve higher quality plans on average by increasing the number of samples.

Action Selection

A variety of selection processes can be designed and finely tuned to achieve different goals using this adaptive sampling approach. One of the sampling selection methods we implemented and tested selects the action with the largest percentage of goal achievements relative to the number of samples taken; similar to the method for selecting an action for execution. Another method selects the action that had the lowest average plan length returned by the samples. These selection methods are very simplistic and would not be very effective without a fair approximation of the futures of each action already. This follows simply because these methods are entirely exploitive and require an identifiable structure in the data to exploit. The approximations to the action's true heuristic values constitute the structure, however, these approximations will not be usable without sufficient sampling as they would be inaccurate.

The key to increasing the action's heuristic value is to increase the number of samples allocated to the action. Initially, with few samples, the heuristic value is highly variable and most likely not close to the true heuristic value. Since the previously stated methods allocate samples to the most promising actions without ever sampling less promising actions, they will never explore less attractive options. This is a problem when actions' values are poorly estimated and 'good' actions' values are below 'poor' actions' values. This causes the good actions to never be discovered because of their poor initial estimates. Thus, these two methods may be beneficial in some situations where exploration is unnecessary but other more sophisticated methods will probably perform better with less samples; results in later sections confirm this.

Moving on from simple average value checking, the next step is integrating variance and confidence intervals to the selection process to promote exploration of promising actions rather than only the most promising action. Selecting for the largest upper confidence bound is one method that succeeds in solving the exploration and exploitation tradeoff as shown in the algorithm UCB1 [5, 4]. The idea when applied to planning is to sample the action that has shown the potential to produce the largest reward. In doing so, the accuracy of the values for this action will hopefully reduce or move the confidence interval until another action becomes selected or the action dominates all other actions. Given two actions, a_1 and a_2, construct a confidence interval around the success rates for each action 3 standard deviations in radius. One action, a_1, dominates the other, a_2 if the lower bound on the confidence interval is larger than the upper bound of the other, that is $Lower(a_1) > Upper(a_2)$. 3 standard deviations were used to statistically guarantee 89% of the values are within this range assuming the samples are normally distributed. This is definitely not the case for all situations. For instance, this would not be appropriate for a domain that creates two very likely plans that stem from one action yielding different plan lengths. Despite this the assumption is sufficient for our needs.

Unfortunately convergence to the optimal values using this method can take a significant amount of time and is not guaranteed if a base level of relatively accurate sampling is not established, just as in the previous case. The number of samples required to get accurate values for action heuristics could become insurmountable because of time constraints. The results of this method of selection are shown below and give variable results depending on the difficulty of the problem. The method does solve some problems with relative ease but it is insufficient for more difficult domains.

Another step above the last method of sampling selection is based off of the UCT algorithm for sampling selection in Monte-Carlo planning. This algorithm was derived from multi-armed bandit-based games: multi-armed slot-machines that offer various rewards and differing probabilities of achieving those rewards. When faced with multi-armed bandit-based problems the objective is to minimize the regret, the difference between the expected reward of a user playing the optimal strategy and the strategy that is actually played. The UCT algorithm takes this idea and applies it to planning by finding a model of regret for a planning problem and using a heuristic for sampling selection that is dependent on both the reward as well as the number of times an action has been sampled. The exact method is detailed in [5] but the important aspect for this work is the bias sequence defined as: $c_{t,s}$ = sqrt(2 ln t / s), where *t* is the total number of samples taken and *s* is the number of samples allocated to that arm. In the context of FF-Hindsight, the bias sequence is combined with another value such as the success rate or plan length. This function is used as the selection heuristic. Because the function depends on the amount of sampling done as well as the promise of reward, this selection method promotes both exploration and exploitation. The objective of the UCT algorithm was to create a method that has a small error probability and converges to the optimal action after enough time. These two properties are very beneficial for the selection process used in FF-Hindsight because we want to have a high probability of accurate estimates as well as continual improvement of the quality of the estimates as the sample size is increased.

The last group of sampling selection methods involve the variances of the success rates and the plan length calculated after each sample and have shown to be the most effective. The objective of these sampling methods is to provide some measure of the accuracy of the action statistics, that is the success rate and the average plan length. Since these values are what the planner uses to select actions for execution after the sampling time, it is important to get these values to stabilize around their true values. The planner can measure this stabilization using the variance and the rate of change of the variance levels.

Variation in Plan Length

The variance level of the success rate is less interesting than the variance of plan length since the values that represent achieving a goal are binary, either yes or no, 1 or 0. Because of this the variance's range is simply [0, 0.25] and does not describe anything more informative than the average. However, the variance in plan lengths returned by FF can be very informative at establishing classes of futures and we exploit this to measure stabilization.

If the variance in plan length is large then FF is returning a high variety of plan lengths many of which can be put into two classes. If the domain allows cycles then the draw of the futures can be returning plans that have sections repeated or sections that, after being taken off a largely similar plan are returned to it and completed. The other class consists of plans that are highly variable in the sequence of actions by taking alternate routes to achieve the same goal. This indicates that chance is a large determinant of the future and that plans are likely to change throughout execution. The former can simply be visualized in the probabilistic version of blocksworld where after attempting to pick up and place one block on another, the block falls back to the table and the claw must pick up the block and try again. Even though more actions are required to fulfill the goal, the overall plan is largely the same across all futures. The latter possibility is better described by a domain that has a variety of methods of completing the goal, such as tireworld, and the probability of failing on any of those paths is relatively high such that futures cause FF to find different routes to the same goal.

If the variance in plan length is small then there are also two classes that many sets of plans can fit into. Potentially, FF is returning the same plan or very similar plans for each of the futures. Otherwise we have the latter case as before with a small change: FF is returning a multitude of different plans but the plans achieve the goal in a similar path length. Clearly the most desirable option is the former when the plan variance is small as this implies that chance has little influence over the plan's actions and if this action is selected then the plan will stay mostly constant as more actions are chosen.

Selecting for the largest variance value initially seems like a possible method for ensuring accurate metrics when attempting to improve the accuracy of the statistcs that are gathered on plans returned from FF. However, this is not an effective method because some states will require a large variance to describe the possible plans in the futures and will not reduce with further sampling. These acion futures will dominate the sampling, forcing other actions of the state to remain potentially poor estimates. Two ways to identify when the statistical values for each action are stabilizing is to measure the rate of change in the average value of interest and to measure the rate of the change of the variance. In this paper we chose to analyze the effects of using the rate of change of the variance, however, using the rate of change of the average is a potential area for future research.

The rate of change of the variance was used as a selection method over the rate of change of the average because the variance in plan length has the ability to show the planner in a rapid way what kind of plans are being returned by FF. With a finite set of futures there is a finite set of plans that should be returned and in many cases these plans should be similar if the effects of actions are not too dependent on chance. The planner should make a decision on which action to execute when the statistics, the mean success rate and plan length, have generally incorporated the different types of plans returned and the success rate across the futures. The rate of change of the variance after each sample better shows this than the rate of change of the mean. For instance, if two plans, p1 and p2, are returned in two different futures, f1 and f2, with plan lengths 10 and 100 respectively and another sample was taken returning a plan of length 55. The change in mean plan length would be 0 but the change in variance would be large. Since the plans returned by FF are changing in length then the planner should continue sampling because it is likely that chance is having a large effect on the plans that are returned, however the rate of change of the mean does not show this.

Implementation

To analyze the effects of the different sample selection methods, each one was built into the original code for FF-Hindsight. The new planner, FF-AdHOp for adaptive hindsight optimizaiton, was run and tested with a variety of sample selection methods and parameters. Two major functions were established that were not present before. First an even sampling phase samples each action with the same number of futures. Immediately after the adaptive sampling phase is run incorporating one of the selection methods for sampling. Each sampling method was integrated in a module like fashion such that any one could be selected for during run time. The latest version of FF-AdHOp was written in Scheme and compiled to C with Scheme-to-C. The version of FF used was written in C and called from the planner.

The general algorithm employed by FF-AdHOp is detailed in Figure 1.

```
1. Observe current state and determine possible actions
2. Generate a future and run FF on this determinized problem
3. Repeat 2 for each possible action n times
4. Select an action based on a selection method and action statistics
5. Generate a future and run FF on this new determinized problem
6. Repeat 4,5 until sample pool is exhausted, one action dominates all others, or all
   actions' statistics have stabilized
7. Select and execute an action based on the statistics computed from sampling
8. Repeat 1 through 7 until the end of the round or a goal state is reached
```

Figure 1: The psuedo-code for FF-AdHOp's algorithm

Steps 1 through 7 describe a 'turn'. The implementation of determinization, future generation and execution of FF was largely unchanged except to improve the speed of execution by integrating calls to scripts as natural code in the project. This means that many of the implementation details in FF-Hindsight are also present. Correlated futures is a detail from FF-Hindsight which can have effects on the success of the planner overall. This is not a problem for this paper, however, since we are comparing the same planner with modular modifications and the comparisons only detail the effects of adaptive sampling.

There are several aspects of this general algorithm that are important for execution and testing. First, the creation of a sample pool that maintains a maximum number of samples per turn. This pool is distributed among the actions to create the adaptive sampling effect. The separation of base sampling (steps 2,3) and adaptive sampling (steps 4,5,6) proved invaluable as a variety of tests could be run to show the effect of adaptive sampling and its selection methods. A continuum from vanilla FF-Hindsight to purely adaptive can be generated by increasing or decreasing the number of samples allocated to the base sampling phase or the adaptive sampling phase. Actions selected for execution was modified to allow the user to choose a method of selection during run time. This allows the planner to choose between mean success rate, mean plan length or any of the statistical values collected during the sampling phase to determine which action to execute. The original selection method chose the action with the largest success rate; if two actions had the same success rate then the one with the smallest plan length was selected. This method of action selection for execution was kept as it is the most obvious for maximizing the success rate, the goal of the IPPC-04 benchmarks. The plan length method of selection chose actions for the shortest plan length and then for the highest mean success rate if the plan lengths for two actions were equal.

Selection methods for the adaptive sampling phase are summarized in the Table 1 below. A few of the selection methods were implemented for the sake of completeness including selecting for the variance in success rate for each action.

A brief trial adding a 'fuzzy' type of selection that tried to reduce the impact of small sample sizes on action selection for execution was created as well. The method added bounds around the mean success values of actions incorporating the number of samples and the variance. The planner choose the acton that had the highest success rate and then, using the bounds determined for this action, selected the action with the smallest plan length inside the bounds. This was only briefly tried in an attempt to balance very similar actions and overcome the incompleteness of FF's algorithm, however, this is a potential area for further research. The answer may lie in using a more complete deterministic planner instead of varying the selection method.

Adaptive Selection Method	**Description**
'Mean / 'PlanLen	(Highest success rate / Shortest plan length)
'Mean-Var / 'PlanLen-Var	Highest variance in (success rate / plan length)
'MeanUCB	Highest upper confidence bound on success rate
'MeanLCB	Lowest lower confidence bound on success rate
'MeanUCT	Largest F(s,a)*
'PlanUCT	Smallest G(s, a)**
'Mean-VarDiff / 'Plan-VarDiff	Largest change in (success rate / plan length) variance

Table 1: Adaptive sampling selection methods used in Step 4 of Figure 1.
**F(s,a) = success rate + sqrt(2*ln(total samples) / samples of a)*
***G(s,a) = plan length - sqrt(2*ln(total samples) / samples of a)*

Results

The results detailed here are using a Pentium 2.4 GHz Quad Core computer with 2.5 GB of RAM running Linux. The results shown for FF-Hindsight were taken using the slow version of FF-Hindsight that was not used in [8] but is the original concept of hindsight optimization that was found to be too slow for competition. Despite this, it was much simpler to implement the modifications to the sampling process with this project as the faster version did not implement a sampling method compatibile with the adaptive sampling approach. Since we could run both the original slow version against the adaptive sampling versions there is a direct comparison with no extraneous variables between vanilla FF-Hindsight and FF-AdHOp.

	Adaptive Selection Method		
Domains	FF-HOP	'Mean-VarDiff	'Plan-VarDiff
Bw-c-pc-8	1	1	1
bw-nc-pc-11	0		
bw-nc-pc-nr-8	1		
explodingbw-pre	0		
towers-of-hanoise-pre	0		
zeno-pc	0	0	0

*Table 2: Results of adaptive selection methods using the success rate as criteria for action execution. The results shown use the best results from varying sample sizes of 5 rounds. *The selection methods not shown did not score at all*

Due to the poor results using the success rate as the criteria for selecting the action to execute at the end of a turn, the plan length was used instead which yielded much better results shown in Table 3.

The results in Table 2, when compared to other planners such as FF-Hindsight's faster implementation and other probabilistic planners competing in IPPC, are not promising but there is a likely cause for this. FF is not complete as a deterministic planner in that it does not always return a plan for a problem even if there exists one. This was a major issue that was noted during testing; FF would consistently fail to return plans when a plan to reach the goal did exist. This often caused the best action to be ignored for another, seemingly better, action that did not lead the agent in the correct direction. The binary nature of the success rate, either reaching the goal or not, also elevates the problem as the averages at low sample sizes could differ greatly even when their true values were close. Another potential problem is the horizon for each run was set to 100 steps. While all of the domains should be solvable within 100 steps on average it is possible that in some futures this is not the case. Some failures by FF could be the result of this but the likelihood of this occurring often is small.

	Adaptive Selection Method							
Domains	FF-HOP	'Mean	'PlanLen	'Mean UCB	'Mean-VarDiff	'Plan-VarDiff	'Mean UCT	'Plan UCT
Bw-c-pc-8	5	1	3	1	4	5	1	2
bw-nc-pc-11	0	0	0	0	0	0	0	1
bw-nc-pc-nr-8	5	5	5	2	5	4	5	5
explodingbw-pre								
towers-of-hanoise-pre								
zeno-pc	0	0	0	0	0	0	0	0

*Table 3: Results of adaptive selection methods using the plan length as criteria for action execution. The results shown use the best results from varying sample sizes of 5 rounds. *The selection methods not shown did not score at all*

After changing the selection method for action execution was changed to plan length, most of the selection methods fared better in trials; even FF-Hop. The two most effective methods of selection for the adaptive phase were 'Mean-VarDiff and 'Plan-VarDiff. These two are the most applicable methods of selection because they measure how stabile the action estimates have become and manage the number of samples that are needed. FF-Hindsight, however, also solved all of the problems it faced but took significantly longer to do so. The time difference was the result of FF-AdHOp identifying when to stop sampling because values have stabilized within a cutoff percentage whereas FF-Hindsight simply sampled evenly across all actions.

The results in Table 4 show that 'Plan-VarDiff as a selection method for adaptive sampling yields approximately the same results in the quality of plans but decreases the overall time required by approximately 20%.

	Plan Length Average			**Time / Round Average (ms)**	
Samples / Turn	FF-HOP	'Plan-VarDiff		FF-HOP	'Plan-VarDiff
~ 25		70.63			412
~ 50	66.67	69.13		1010	768
~ 100	60.2	52.2		1503	1233
~ 150	41.8			1857	
~ 250	33.4			2204	

Table 4: The effects of increasing the sample size taken per turn for FF-Hindsight on the average plan length after solving the domain (bw-c-pc-8) and the time per round. Cutoff = 1%

	Adaptive Selection Method	
Cutoff Percentage	'Mean-VarDiff	'Plan-VarDiff
1.00%	1 (70.5)	2 (36.5)
5.00%	2 (55.5)	2 (62.5)
10.00%	0	1 (78)
20.00%	0	2 (87)
40.00%	1 (80)	0

Table 5: Two adaptive selection methods varying with their cutoff percentages with unlimited samples. Two trials each on bw-c-pc-8; value in () is the average plan length.

The cutoff percentage determines the value at which FF-AdHOp stops sampling an action and is based on the rate of change of the variance value versus the variance. If the cutoff percentage is 5% then if the rate of change of the variance of the success rate has fallen below 5% of the variance of the success rate the action requires no more sampling. Table 5 shows the effect changing the cutoff rate has. By increasing the cutoff rate the ability to achieve the goal diminishes because less samples are being used but as a benefit, less time is being used. The quality of plans returned is also directly affected. As the cutoff percentage is decreased, plan quality increases. FF-AdHOp can be tuned to balance these two properties because of the advantages of adaptive sampling.

Related Work and Improvements

Other areas of interest that were not fully implemented or tested that could be of use to a sampling technique such as this would be independent futures. This would remove the correlated futures which may have an effect on the variance levels of the plans generally returned by increasing the variety of plans. Since the most effective method for sampling selection involved letting the action's statistics stabilize this could affect the rate of stabilization leading to better or worse estimates and more or less time per action.

Since FF is not guaranteed to find a plan in all domains even if there exists a possible plan, implementing a method to counteract these inaccuracies could prove extremely beneficial, especially for the simpler selection methods. Blocksworld, for instance, should almost always return a successful plan less than 100 steps except in very improbable instances. However, FF would consistently fail to return a plan more often than expected. This is most likely a cause of the poor performance exhibited by selection methods when used in conjuction with selecting the action to execute based on mean success rate. A simple method that was not fully implemented in FF-AdHOp involved a metric that incorporated the confidence in the selection value based on the value's variance and the number of samples that have been allocated to the action. This would relax the effects that a couple of failures by FF would have on the actions statistics. This may also have a negative effect, allowing the planner to choose worse actions but this tradeoff could be worth looking into.

A variety of other selection methods can also be used for sampling including selecting for the rate of change of the mean. Using the probability that a future will occur to weight the value of a sample could also have large effects on identifying useful samples and eliminating highly improbable samples similar to weighted sampling used on bayesian networks. This could also lead to a large number of useless samples that have little effect on the heuristics. The probability of a future occuring could also be used as the value of a plan's success instead of 1 which may end up being a measure of how much the future space has been searched.

Conclusion

The advancement of probabilistic domain difficulty through the identification of more interesting problems has led to vast improvement over FF-Replan's original design. FF-Hindsight's use of dynamic determinization has introduced an important question: Are there ways to direct future generation and samples to achieve better accuracy with less sample time? In this paper we suggest a variety of methods to try to improve the accuracy of action value estimates while reducing the time and energy required by the planner to do so. Many of the simple sampling methods did not yield beneficial results, however, more sophisticated methods such as directing the sampling effort to stabilize estimation values offered generous improvement over naïve sampling. This shows that sacrificing some time to direct sampling efforts can payoff in this planning situation.

We hope to continue improvement and testing on different planning domains to identify more challenging domains and their effects on the performance of FF-AdHOp and FF-Hindsight. Improving the speed and efficiency of the planner itself is an important step as well. This will allow FF-AdHOp to be compared with state of the art planners. Despite the slow speeds exhibited by FF-AdHOp it is important to note that this technique is not limited to hindsight optimization but extends to a variety of sampling based methods. Adaptive sampling could also be applied to planners such as iHop [9] which have far superior performace than the planner used in this paper with the addition of helpful actions. Other areas of interest include improving the 'fuzzy' selection method as it showed some promise in testing and allowed selection for success rate over plan length.

Acknowledgements

This work was funded in part by the National Science Foundation (NSF) and the Fulton Undergraduate Research Initiate (FURI) at Arizona State University. We thank you for your support.

References

[1] P. Auer, N. Cesa-Bianchi and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning,* 47(2-3):235-256, 2002.

[2] D. Bryce and O. Buffet. International Planning Competition Uncertainty Part: Benchmarks and Results. At http://ippc-2008.loria.fr/wiki/images/0/03/Results.pdf.

[3] C. G. Butcher. Adaptive Sampling – an iterative fast Monte Carlo procedure. *Structural Safety*, Vol 5, Issue 2: 119-126, 1988.

[4] D. E. Knuth. *The art of computer programming, volume 2: seminumerical algorithms*, 3rd edn. p. 232. Boston. Addison-wesley.

[5] L. Kocsis and C. Szepesvari, Bandit based Monte-Carlo Planning. *ECML '06*, 2006.

[6] Little and Thiebaux. Proceedings of the ICAPS'07 Workshop on the International Planning Competition: Past, Present and Future, 2007.

[7] B. P. Welford. "Note on a method for calculating corrected sums of squares and products," *Technometrics* 4(3): 419-420.

[8] S. Yoon, A. Fern, R. Givan, S. Kambhampati. Probabilistic Planning via Determinization in Hindsight. In Proceedings of the 23rd national conference on Artificial intelligence - Volume 2, 2008.

[9] S. Yoon, W. Ruml, J. Benton and M. B. Do, Improving Determinization in Hindsight for On-line Probabilistic Planning. International Conference on Automated Planning and Scheduling (ICAPS), 2010.

[10] H. Younes. Extending pddl to model stochastic decision processes. In Proceedings of the ICAPS-03 Workshop on PDDL, 2003.