# A Linear Time Transform for Probability Aware Planning

J. P. Gunderson, G. J. Ferrer
University of Virginia
Charlottesville, VA, 22903

## Abstract

We present a transform that enables traditional Shortest-Feasible-Plan planners to reason about uncertain operators and produce plans which have higher probabilities of success. This transform converts a probability-aware domain description into a STRIPS-style description, where the probability of success is expressed by plan length. Using this transformed description a plan can be generated by a traditional planner. The transform is shown to be at worst linear in the size of the input, and allows the planning system to trade-off accuracy against runtime as an anytime computation.

## 1 Introduction

Beginning with General Problem Solvers[17], and the STRIPS[8] planning system, traditional planners have defined an optimal plan as the shortest plan which achieves the goal. Such planning systems assumed that the correct operators, applied in the correct order, would always achieve the goal.

Unfortunately, in the real world, operators fail. As a result, the 'best' plan may not be the shortest, since a longer plan may have a higher probability of success, and the impacts of even low levels of uncertainty can significantly affect plan success[11]. In addition, when an operator fails, there may be many different possible results, each with its own likelihood of occurrence. Many different approaches to handling this uncertainty have been taken (see section 1.1 below), ranging from planners which require complete knowledge of all the possible outcomes and the attendant probability distributions, to traditional planners which have no way to represent, or reason about, operator failure.

Recent work has suggested that humans solve complex problems not by exhaustively modeling the probabilities of every possible result, but by applying simple rules which capture broad sets of likelihoods[9]. In this paper we evaluate a simple planner that has basic knowledge of the likelihood of success of an action, and the goal of finding the plan that is most likely to succeed. Such a **probability-aware** planner has several possible benefits: since it is only finding one plan, it avoids the computational complexity of conditional and probabilistic planners; since it has no information about the possible outcomes of operator failure, the domain knowledge needed to use the planner is reduced; and since it has knowledge of the likelihood of failure, it can make trade-offs between plan length and success. Probability aware planners are designed to be part of an interleaved planning/execution system[7], where, upon operator failure, a new planning problem will be instantiated from the resulting world state.

Our purpose is to develop a methodology to allow traditional planners to be aware of uncertainty by providing a transformation from a probability aware domain description to a STRIPS-style description. We can then leverage off the extensive work which has been done to improve the speed and efficiency of traditional planners such as SatPlan [13], Graphplan[1], and others.

To be effective, there are several constraints on any such transformation:

1. It must preserve the ability to produce the plan with the highest probability of success.
2. It must not increase the size of the input to the planner by more than a linear amount, or already computationally intense solutions may become intractable.
3. The overhead of the transformation must not overwhelm the planning process.

### 1.1 Previous Work

A large body of work exists on the subject of planning under uncertainty. Many traditional planners have been extended to handle degrees of uncertainty in the domains, including **Buridan**[14], **C-Buridan**[6], Graphplan extensions[2], as well as development of MDP based planners[4], and other planners based on decision theory[12].

Much of this work focuses on representing two features of operating in an uncertain world: 1) operators have a probability of failure, and 2) the outcome of that failure. The second of these seems to be the most problematic. Research has suggested that it may be impossible to capture an exhaustive and mutually exclusive set of failure modes for any but the simplest operators[5], and even representing a reasonable, though non-exhaustive, set can lead to intractable problem descriptions.

Theoretic work on the complexity of planning under uncertainty has been done by Bylander[3], Littman, Goldsmith and Mundhenk[10, 15], and Pearl[18]. In addition, a recent article by Nebel[16] suggests that the introduction of conditional effects into STRIPS-style description significantly increases the complexity of the planning process.

## 2 Method

We have developed a domain transformation that allows traditional planners to reason about the likelihood of plan success, by encoding the likelihood of plan success into the length of the plan. We assume that the operators can be represented as conditionally independent steps, all of which must succeed if the plan is to succeed. Thus, the probability of a plan succeeding is:

$$P(Success_{Plan}) = \prod_{allsteps} P(Success_{Step})$$

While this requires the product of the probabilities, traditional planners minimize the sum of the steps in the plan. However, we can accommodate this by using a log transform as follows

$$P(Success_{Plan}) = \exp \sum_{allsteps} \ln P(Success_{Step})$$

We transform the probability of an operator succeeding into additional plan steps. These additional steps must be invoked to apply the operator. The higher the operator's probability of success, the fewer the additional steps added. Thus, the shortest feasible plan is also the plan with the highest probability of success.

However, the current transform is insufficient, since it produces real valued results which are less than zero, and we need positive, integer values to be encoded as discrete plan steps. To achieve this final state, we use a *probability quantum* to discretize the number of additional steps to add to the operator. This probability quantum is a value in the range $(0.0, 1.0)$ which is used to determine the bin size of the probability steps, and to produce a positive number of additional plan steps. Thus the number of additional steps for any operator is:

$$Steps = \lfloor \frac{\ln(P(Operator))}{-Quantum} \rfloor$$

The quantum has the following impact on the planning process:

- The smaller the quantum, the less error introduced by the floor function,
- The smaller the quantum, the more additional steps for any given probability.
- The smaller the quantum, the longer the runtime of the planner.

The result of these impacts is that the quantum becomes a mechanism for the planning system to trade off the optimality of the plan with the run time of the planning system.

### 2.1 A quick example

To demonstrate these impacts, suppose our domain were blocks world, and we wished to stack block A on block B. The two operators needed might be `Pickup A`, and `Stack A B`. Suppose that these operators had the following probabilities of succeeding: $P(\texttt{Pickup}) = 0.8$, and $P(\texttt{Stack}) = 0.6$ (e.g., the `Stack` operator will fail 4 times out of 10).

To achieve our goal, we must successfully execute both the operations, and so, recalling our conditional independence assumption, the probability of goal satisfaction is $0.8*0.6 = 0.48$. When we apply the probability-aware planning transformation, we also need to select an appropriate Probability Quantum.

Following the steps of the transformation with a Quantum of 0.1, we see that the `Stack` operator

$$\begin{aligned} Stack_{Steps} &= 1 + \lfloor \frac{\ln(0.6)}{-0.1} \rfloor \\ &= 1 + \lfloor \frac{-0.5108}{-0.1} \rfloor \\ &= 1 + \lfloor 5.108 \rfloor \\ &= 6 \end{aligned}$$

| Quantum | Pickup | Stack | $P\,(Plan)$ |
|---|---|---|---|
| 0.5 | 1 | 2 | 0.22 |
| 0.2 | 2 | 3 | 0.36 |
| 0.1 | 3 | 6 | 0.4065 |
| 0.05 | 5 | 11 | 0.4493 |
| 0.02 | 12 | 26 | 0.4670 |
| 0.01 | 23 | 52 | 0.4723 |

Table 1: Growth of Plan Size, and increase in accuracy with changes in Probability Quantum

expands into 6 steps, where the higher probability operator `Pickup` only expanded into 3 steps. The resulting plan has a length of 9 steps, which corresponds to a probability of success of:

$$P\,(Plan) = \exp\,(9 * -0.1) = 0.4065$$

This probability of success differs from the true probability of success due to the inaccuracies introduced by the discretization process. As the magnitude of the Probability Quantum decreases, these errors should also decrease, however, this increased accuracy comes at the price of longer and longer plans (see Table 1).

## 3 Transform Methodology

The transform is implemented as a preprocessor and postprocessor, which can be used with any planner that accepts STRIPS-style input. The preprocessor reads the domain description (current world state, desired world state, and operators), and additional information which specifies the probability quantum, and operator probabilities. From this a new domain description is created.

In this description each of the original operators is divided into two operators; one has the original preconditions, and the second has the add and delete lists. These are modified to force the planner to include a chain of dummy operators which change the length of the combined operator chain to be the required number of steps (See Figure 1). The precondition portion now has an effect of setting PSTATE3 – the precondition for the @P@3 operator. The Add/Delete portion of the original operator is modified to have PSTATE0 as a precondition. Thus, if the planner needs the original operator's effects, it must backward chain through the added operators to reach the preoperator, and then satisfy the original preconditions.

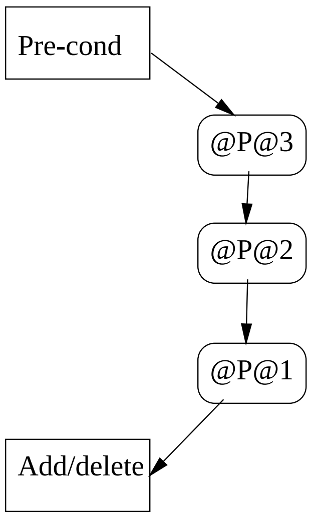

Figure 1: The transformed operator chain, with length increased to reflect operator probability

### 3.1 Change in input size

While this simple example suggests that each operator must include its own sequence of dummy steps, we also include a flag variable which allows all probabilistic operators to share a single chain. Thus the plan size increases by a constant amount which is determined by the operator with the lowest success probability and the Probability Quantum. Since this is independent of the input size, it acts as a constant.

However, each probabilistic operator is replaced with two operators, the preoperator and the postoperator. Hence the input size grows linearly with respect to the plan size. In addition, note that in the case where the probability of operator success is sufficiently high, the floor function causes no additional steps to be added. In this case there is no need to split the original operator into pre and post phases, and the overall size of the input grows sub-linearly.

## 4 Experiments

The experimental setup uses a variant of the Probabilistic Blocks World domain proposed by Blum and Langford[2]. In this domain there are the traditional `Pickup` and `Stack` operators, and a set of `faststack` operators which combine `Pickup` and `Stack` into a single step. These `faststack` operators provide the ability to add a block to towers of various heights: `faststack1` puts a block onto a single block, `faststack2` adds a new block onto a tower of

| Operator | Probability of Success |
|---|---|
| pickup | 0.88 |
| putdown | 1.0 |
| faststack3 | 0.71 |
| faststack2 | 0.81 |
| faststack1 | 0.94 |
| stack | 0.88 |
| unstack | 1.0 |

Table 2: Operator Success Probabilities

two blocks, and `faststack3` adds a new block onto a tower of more than 2 blocks. Their success probabilities are ordered:

$$P(\texttt{faststack3}) < P(\texttt{faststack2}) < P(\texttt{faststack1})$$

We ran a series of experiments to determine the performance of the Probability-Aware planning system. The experiment fell into three categories:

1. Shortest feasible plan,
2. Effect of problem size on performance, and
3. Effect of quantum value on performance.

In each case, the Blocks World domain description, augmented with operator probabilities and a Probability Quantum, was preprocessed and the output was passed to Graphplan. The probabilities used for the operators are in Table 2. The output from Graphplan was then postprocessed to result in a feasible plan.

The first set of experiments did no probability aware planning at all. Graphplan returned the shortest feasible plan. This data provides a baseline for comparing the probabilistic runs. We used six blocks world problems ranging in size from 2 to 7 blocks.

To empirically verify the effect of problem size on performance we ran a series of experiments with a fixed quantum of 0.04 to generate probabilistic plans for the same six blocks world problems.

Our third set of experiments used just the 7 blocks problem and varied the quantum values from 0.02 to 0.10.

## 5 Results

One critical issue was the overhead of performing the transformation to the domain descriptions. Typically, the planning time overshadows any preprocessing time, and we found this to be the case. Attempts to time the transform resulted in time values of 0.0 seconds, for all experiments. We concluded that our transform does not add significant overhead to the planning system.

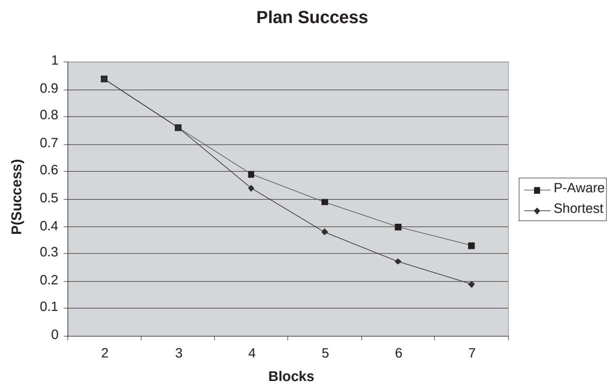

Figure 2: Probability of plan success for shortest-feasible planner and Probability-Aware planner

The probability-aware plans started out by unstacking everything, but then exhibited a curious behavior. When placing a block atop a stack too tall to use anything except `faststack3` or the `pickup`/`stack` combination, they would `faststack1` a block onto some arbitrary block, use the perfect success `unstack` operator to grab the block, and then `stack` the block atop the stack. The `faststack1`/`unstack`/`stack` combination has a higher probability than either `faststack3` or the `pickup`/`stack` combination, so this action sequence makes perfect sense, but we did not anticipate it. When no solitary blocks remained, the `pickup`/`stack` combination was used.

### 5.1 Probability of Success

As expected, the probability of success for all of the probability-aware plans exceeded that of the shortest feasible plans (See Figure 2) in all cases except the two and three block problems, where the probabilities were equal. Also as expected, the plan lengths for the probability-aware plans were greater, or equal to, the shortest feasible plans. In addition, a comparison of the number of lines of domain description needed to capture the additional information shows sub-linear growth (See Figure 3) as the number of blocks increases. This is due, in part, to the fact that some of the operators did not require expansion into pre

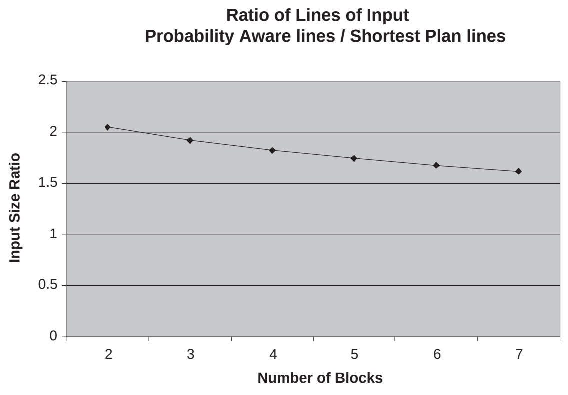

Figure 3: Ratio of lines of input for probability-aware planner and shortest path planner with quantum = 0.04

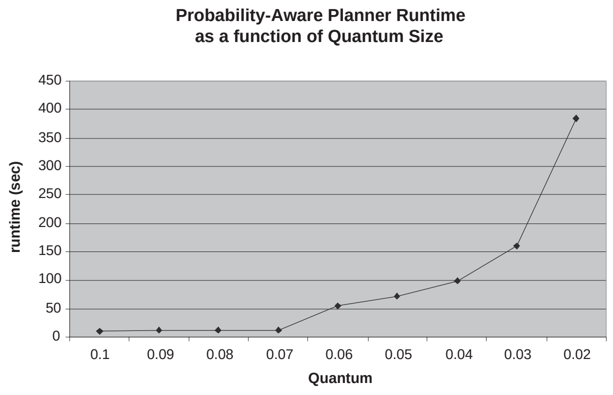

Figure 4: Exponential growth of runtime as a function of quantum size.

and post states. The rest is due to the fact that part of the size increase is cause by the need to add the chain of dummy operators, which is a constant at any selected quantum.

## 5.2 Runtime

The run times of the quantum variation experiments show a clear exponential increase in run time as the quantum value decreases (See Figure 4).

The variable-quantum experiments demonstrated the sensitivity of the Probability-Aware planner to variations in this number. Quantum values 0.02, 0.03, 0.04, 0.07, and 0.08 all resulted in the same plan, which was the highest probability plan of the set. Likewise, quantum values 0.09 and 0.10 resulted in another plan, this being the second highest probability plan. Quantum value 0.05 resulted in a plan noticeably worse than either of these, and quantum value 0.06 resulted in the lowest probability plan of all. In general the smaller the quantum, the higher the probability of plan success. However, the mapping is not monotonic, due the the approximation effects of the dicretization.

Shifting the quantum values results in the various operators and combinations of operators falling in different "bins" when the probability space gets quantized. In the case of quantum value 0.06, six total operators are required to execute a `faststack1`, five dummy operators plus the actual operator. Three total operators are required to execute each of `pickup` and `stack`, two dummy operators plus the actual operator. The result is that for quantum 0.06, `faststack1` and the `pickup`/`stack` combination are considered probability equivalent, even though `faststack1` has a probability of 0.6 and the `pickup`/`stack` combination is about 0.72. Hence, the 0.06 plan uses both in its plan when it should have stuck exclusively with the `pickup`/`stack` combination.

# 6 Summary and Conclusions

We have developed a domain transformation, which allows traditional planners to reason about the likelihood of plan success, by representing the likelihood of plan failure as the length of the plan. This transformation begins with a traditional STRIPS-style representation of the domain, and additional information about the probability of operator failure. It produces a new domain description, in the same representation. Since any traditional STRIPS-style planner can use the new domain to reason about the effects of operator uncertainty on plan success, many previously intractable planning domains are now accessible.

We have demonstrated that this transform meets three critical requirements:

1. A Shortest-Feasible-Plan planner produces plans with the highest probability of success,
2. The transform increases the input size at worst linearly, and
3. The overhead of the transform in negligible.

In addition, we have shown that traditional planners can be used for a form of anytime planning, by starting with large quantum values and planning using progressively smaller values as time permits. Since the transform allows the tradeoff of accuracy in the plan certainty for improved run-time, this transformation supports "anytime planning" in uncertain domains.

## Acknowledgments

The authors would like to acknowledge the Medical Automation Research Center at the University of Virginia, and the ARCS foundation, for providing funding for parts of this research.

## References

[1] A. L. Blum and M Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[2] A. L. Blum and J. C. Langford. Probabilistic planning in the graphplan framework. In *Proceedings of the Fifth European Conference on Planning*, Durham, United Kingdom, 1999.

[3] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69:165–204, 1994.

[4] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1-2):35–74, jul 1995.

[5] R. Dearden and C. Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89:219–283, 1997.

[6] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. In *International Conference on Artificial Intelligence Planning and Scheduling*, 1994.

[7] G. J. Ferrer, G. S. Wasson, and W. N. Martin. Constraining planning episodes in an interleaved architecture. In *Integrated Planning for Autonomous Agent Architectures.* AAAI Fall Symposium Series, 1998.

[8] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[9] G. Gigerenzer, P. M. Todd, and the ABC Research Group. *Simple Heuristics that Make Us Smart.* Oxford University Press, 1999.

[10] J. Goldsmith, M. L. Littman, and M. Mundhenk. The complexity of plan existence and evaluation in probabilistic domains. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 182–189, 1997.

[11] J. P. Gunderson. Effects of uncertainty on variable autonomy in maintenance robots. In *Workshop on Autonomy control software.* Autonomous Agents '99, 1999.

[12] P. Haddawy and S. Hanks. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the Third International Conference of Principles of Knowledge Representation and Reasoning*, pages 71–82. Morgan Kaufman, 1992.

[13] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, USA, 1996.

[14] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabalistic planning. *Artificial Intelligence*, 76(1–2):239–286, Jul 1995.

[15] M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabalistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[16] B. Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

[17] A. Newell and H. Simon. Gps, a program that simulates human thought. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought.* R. Oldenbourg KG., 1963.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, San Mateo, CA, 1988.