Risk-Sensitive Planning with Probabilistic Decision Graphs

Sven Koenig
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Reid G. Simmons
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Abstract

Probabilistic AI planning methods that minimize expected execution cost have a neutral attitude towards risk. We demonstrate how one can transform planning problems for risk-sensitive agents into equivalent ones for risk-neutral agents provided that exponential utility functions are used. The transformed planning problems can then be solved with these existing AI planning methods. To demonstrate our ideas, we use a probabilistic planning framework ("probabilistic decision graphs") that can easily be mapped into Markov decision problems. It allows one to describe probabilistic effects of actions, actions with different costs (resource consumption), and goal states with different rewards. We show the use of probabilistic decision graphs for finding optimal plans for risk-sensitive agents in a stochastic blocks-world domain.

1 Introduction

In recent years, numerous planning methods have been developed that are able to deal with stochastic domains.[1] Consider the stochastic domains for which it is easy to construct plans that always reach a given goal for sure (at least, in the limit). Then, one needs a criterion for choosing among these plans. Such a metric is for example the execution cost of the plans: One quantifies the "resource consumption" (for example, time, energy, or money) of an action with a single real number that depends only on the action, the state it is executed in, and the resulting state. Then, the execution cost of a plan is defined to be the sum of the resource consumption costs of all actions executed from the time at which the agent begins until it stops in a goal state.

Since the execution cost of a probabilistic plan can vary from plan execution to plan execution, almost all probabilistic planning methods that take execution cost into account use the *expected* execution cost as ranking criterion: Out of all plans that guarantee to achieve the given goal, they choose the one that minimizes the expected execution cost (when optimizing) or one whose expected execution cost is smaller than a given number (when satisficing). Since they do not take the variance of the execution cost into account, they assume that the agent that executes the plan has a risk-neutral attitude.

However, people are usually not risk-neutral. A risk-seeking agent ("gambler") is willing to accept a plan with a larger expected execution cost if the uncertainty is increased and vice versa for a risk-averse agent ("insurance holder"): If a plan is executed only once (or a small number of times), then — among all plans with the same expected execution cost — the larger the variance of the execution cost, the larger the chance to do much better than average. Of course, the chance to do much worse rises as well.

Imagine, for example, that your task is to design a robot for the annual AAAI robot competition, where it has to complete a given task (for example, "find the coffee pot") in as short a time as possible. You want the robot to win the competition, but — in case it loses — do not care whether it makes second or last place. You know that your robot is not much faster than your competitors' robots, maybe even a bit slower, but cannot assess the capabilities of the other robots in enough detail to use them for determining the utilities of the various task completion times of your robot. In this case, you probably want your robot to take chances, and thus a risk-seeking attitude should be built into the robot's planning mechanism.

It is possible to achieve a risk-sensitive attitude by ranking plans not according to their expected execu-

[1] Examples of probabilistic planning methods include [Smith, 1988], [Bresina and Drummond, 1990], [Christiansen and Goldberg, 1990], [Hansson *et al.*, 1990], [Koenig, 1991], [Dean *et al.*, 1993], [Kushmerick *et al.*, 1993], and many others.

tion costs, but according to their expected execution costs plus or minus a fraction of the variances [Filar *et al.*, 1989] [Karakoulas, 1993], or by searching for plans whose execution costs are optimal in the best or worst case ("nature acts like a friend or enemy") [Heger and Karsten, 1992] [Heger, 1994], see also [Moore and Atkeson, 1993]. However, utility theory [von Neumann and Morgenstern, 1947] — a subfield of decision theory — provides a normative framework for making decisions according to a given risk attitude, provided that the agent accepts a few simple axioms and has unlimited planning resources available. The key result of utility theory is that, for every attitude towards risk, there exists a utility function that transforms costs c into real values $u(c)$ ("utilities") such that it is rational to maximize expected utility. Its application to planning problems has been studied by [Etzioni, 1991], [Russell and Wefald, 1991], [Haddawy and Hanks, 1992], [Wellman and Doyle, 1992], [Goodwin and Simmons, 1992], and others. Therefore, we would like to stay within this framework.

In this paper, we describe a planning framework ("probabilistic decision graphs") that can easily be mapped into Markov decision problems and of which cost-annotated decision trees (the kind used in utility theory) are a special case. It allows one to describe probabilistic effects of actions, actions with different costs (resource consumption), and goal states with different rewards (goodness). We show that replacing all costs and rewards with their respective utilities, but leaving the planning mechanism unchanged, usually leads to erroneous results. Furthermore, the best action to execute in a state can depend on the total cost that the agent has already accumulated when deciding on the action.

For utility functions of a certain class, however, planning problems for risk-sensitive agents can be transformed into equivalent planning problems for risk-neutral agents which can then be solved with dynamic programming methods or probabilistic AI planning methods that minimize (or satisfice) expected execution cost. The transformation has the property that the better a plan is for the transformed, risk-neutral planning problem, the better it is for the original, risk-sensitive planning problem as well. Our approach builds on previous work by [Howard and Matheson, 1972] in the context of Markov decision theory. A blocks-world example is used to illustrate our ideas and show how the optimal plan depends on the degree of risk-sensitivity of the agent.

2 The Planning Framework

The following representation of probabilistic planning problems was used in [Koenig, 1991]. A similar framework has recently been used by [Dean *et al.*, 1993] and is commonly used for table-based reinforcement learning.

Planning is done in a state space. S is its finite set of states, $s_0 \in S$ the start state, and $G \subseteq S$ a set of goal states. A plan determines at every point in time which action the agent has to execute in its current state. In a goal state s, the agent receives a (positive or negative) one-time goal reward[2] $r[s]$ and then has to terminate. The goal rewards reflect that different goal states can be of different value to the agent. However, to keep the following discussion simple, we will use only planning examples for which all goal rewards are zero. In a non-goal state s, the agent can choose an action a from a finite set of actions $A(s)$. Nature then determines the outcome of the action with a coin flip: with transition probability $p^a[s, s']$, the agent incurs an action cost $c^a[s, s'] < 0$ and is in successor state s'. Thus, we assume that the outcomes of all action executions are mutually independent given the current state of the agent (Markov property). The action costs reflect the resources consumed, for example, time needed or effort spent. We assume that the values of all parameters are completely known and do not change over time. We do not assume, however, that the planner uses a planning approach that operates in the state space (instead of, say, the space of partial plans).

For a given plan, we define the probability of goal achievement of state s as the probability with which the agent eventually reaches a goal state if it is started in s and obeys the plan. If this probability equals one, we say that the plan solves s. A plan that solves the start state is called admissible. In the risk-neutral case, a plan is evaluated according to the expected total reward of the start state. The expected total reward $v[s]$ of state s for a given plan is the expected sum of the reward of the goal state and the total cost of the actions that are executed from the time at which the agent is started in s until it stops in a goal state (given that it obeys the plan). Similarly, the expected total utility $u[s]$ of state s is the expected utility of the sum of the goal reward and the total cost of the executed actions.

The planning framework described above is very general. For example, one can easily represent goal states in which the agent does not have to stop (that is, goal states that the agent can leave in order to reach a different goal state that has a larger goal reward). This is necessary if one wants unsolvable states to have an expected total reward that is finite instead of minus infinity. One could, for example, allow the agent to stop in *any* state, but penalize it for stopping in a non-goal state. (In this case, all non-goal states must be converted to goal states that have a very small goal reward and can be left again.)

[2] From here on, we use the terms "rewards" and "costs" as follows: Rewards can be positive or negative values, but costs are *always* negative values.

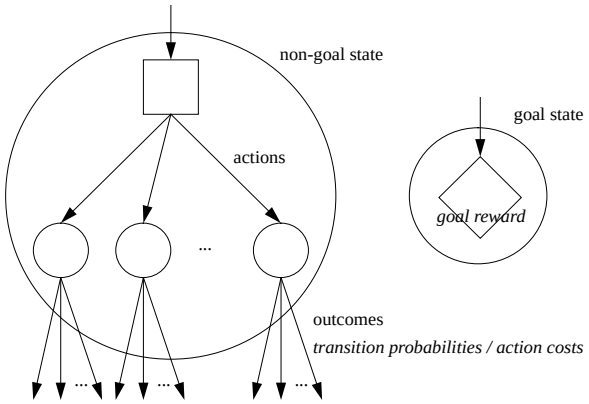


Figure 1: Building Blocks

For risk-neutral agents, the planning framework is isomorphic to Markov decision problems [Mine and Osaki, 1970]. A state-action mapping ("stationary, deterministic policy") specifies for every state the action that the agent has to execute when it is in that state. For Markov decision problems, one can restrict plans to state-action mappings without losing optimality.

We use an easier-to-depict representation for probabilistic planning problems here, which we call "probabilistic decision graphs", that resembles the kind of decision trees that are used in utility theory. Its building blocks are shown in Figure 1. Every state corresponds to a (large) circle. The large circle of a non-goal state s contains a decision tree that consists of a decision node (square) followed by chance nodes (small circles), one for every $a \in A(s)$. Transition probabilities and action costs are specified for every outcome of the actions. The circle of a goal state contains a value node (diamond) for the goal reward. To represent a planning problem, these building blocks have to be connected so that there are no dangling outcome arrows. In addition, the start state is marked with an incoming arrow that has no source state and is labeled "start."

Note that probabilistic decision graphs can have cycles: cycles do not imply that a decision depends on itself, but that a decision depends on the same decision made at an earlier point in time. In the following, we will distinguish two simplifications of this planning framework, namely *acyclic* probabilistic decision graphs and the even simpler acyclic probabilistic decision graphs *without shared subtrees* ("cost-annotated decision trees"). The last two varieties are commonly used in utility theory.

3 The Problem

We suspect that researchers have largely ignored the question of how to incorporate risk-sensitive attitudes into their planning mechanisms because they assume that by replacing all costs and rewards with their respective utilities (for an appropriate utility function) one can achieve risk-sensitive attitudes without changing the planning mechanisms. In the following, we use acyclic probabilistic decision graphs to demonstrate that this is not necessarily the case after reviewing how to use dynamic programming techniques to determine optimal plans for risk-neutral agents.

3.1 Planning for Risk-Neutral Agents

A risk-neutral agent has to solve planning task PT1: given a complete specification of the planning problem, find a plan for which the start state has the largest expected total reward.

An optimal state-action mapping for planning task PT1 can be determined in polynomial time with dynamic programming techniques. To solve an *acyclic* probabilistic decision graph, we could transform it as follows: First, we propagate the action costs to the value nodes. This amounts to duplicating shared subtrees, since every path from the start state to a goal state needs to have its own value node. The resulting decision tree can then be solved in time linear in its size: the expected total reward of a value node is the sum of its goal reward and the accumulated costs, the expected total reward of a chance node is the average over the expected total rewards of its successor nodes weighted with the transition probabilities, and the expected total reward of a decision node is the maximum of the expected total rewards of its successor nodes. The action that achieves the maximum is the optimal decision for the decision node. The expected total reward $v[s]$ of a (non-goal) state s is equal to the expected total reward of the decision node that it contains, and the optimal action $a[s]$ for the state is the same one that is optimal for its decision node.

The transformation outlined above can be done in linear time if no subtrees are shared. However, if subtrees are shared, it is expensive, since the number of paths — and therefore the size of the transformed decision tree — can be exponential in the number of states of the original tree. Fortunately, it is well known that the following dynamic programming technique ("[averaging-out-and-]folding-back") solves acyclic probabilistic decision graphs for risk-neutral agents on the original tree in linear time, that is, without duplicating shared subtrees.

$$v[s] \quad := \quad \begin{cases} r[s] & \text{for } s \in G \\ \max\limits_{a \in A(s)} \sum\limits_{s' \in S} p^a[s,s'](c^a[s,s'] + v[s']) & \text{otherwise} \end{cases}$$

$$a[s] \quad := \quad \arg\max_{a \in A(s)} \sum_{s' \in S} p^a[s,s'](c^a[s,s'] + v[s'])$$
$$\text{for } s \in S \setminus G$$

Thus, one evaluates every subtree only once and the run-time of the algorithm is linear in the size of the original decision tree. Dynamic programming algorithms, such as this one, can be used to solve planning task PT1, because the Markov property holds for all states: the expected total reward $v[s]$ of every state (and thus the optimal action $a[s]$ for the state) is independent of how the agent reached the state.

3.2 Planning for Risk-Sensitive Agents

A risk-sensitive agent has to solve planning task PT2: given a utility function and a complete specification of the planning problem, find a plan for which the start state has the largest expected total utility.

Planning task PT2 can be solved for probabilistic decision graphs without action costs by first replacing all goal rewards with their respective utilities and then using any planning method for risk-neutral agents. In reality, however, the probabilistic decision graphs of planning task PT2 do have action costs. Similarly to how we proceeded earlier for risk-neutral agents, we could first propagate the action costs to the value nodes (which involves duplicating shared subtrees if they exist). Next, all rewards at the value nodes are replaced with their respective utilities. Finally, folding-back is used to determine an optimal plan. Remember that this method has an exponential run-time in the worst case (and is not directly applicable to cyclic probabilistic decision graphs). As an example, consider the partly specified planning problem shown in Figure 2. This planning problem contains a shared subtree that represents the choice between a deterministic lottery[3] A (reward -0.48 for sure) and a nondeterministic lottery B (rewards -0.10 and -1.00 with equal probability). The application of folding-back on this planning problem for a risk-seeking agent with utility function $u(c) = -\sqrt{-c}$ (for $c \leq 0$) is shown in Figure 3. (Actions that are sub-optimal are "crossed out" with two horizontal lines.)

It is not optimal to simply replace all costs and rewards with their respective utilities and then use folding-back on the resulting tree, because in general $u(c_1 + c_2) \neq u(c_1) + u(c_2)$ for two rewards c_1 and c_2 (that is, the value function is no longer time-additive). In fact, dynamic programming methods can no longer be used *in any way* without considering the total action cost that the agent has already accumulated when deciding on the actions, because the Markov property does not

[3] "Lottery" is a term from utility theory. A lottery is recursively defined to be either a reward that is received for sure (that is, with probability one) or a probability distribution over lotteries.

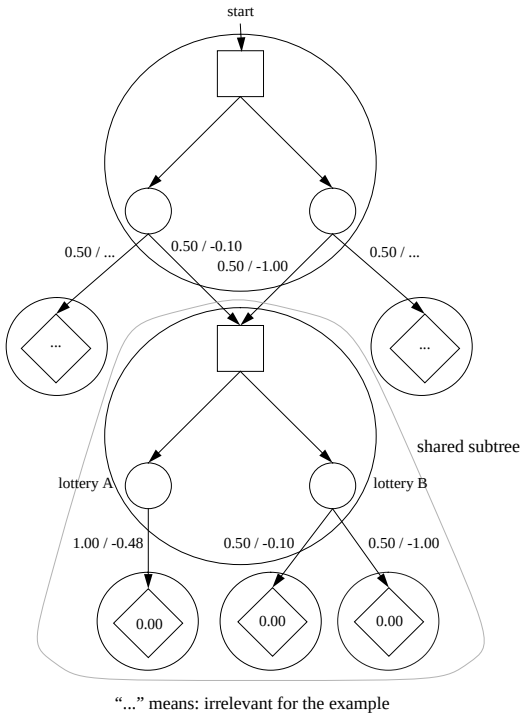


Figure 2: A Planning Problem with a Shared Subtree

necessarily hold for risk-sensitive agents [Raiffa, 1968].

Consider again the planning problem from Figure 2. As demonstrated in Figure 3 for $u(c) = -\sqrt{-c}$, the agent should choose lottery B if it has already accumulated action costs of -0.10 when deciding between the two lotteries. However, if the accumulated action costs are -1.00, it should prefer lottery A. This can be explained as follows: The agent is risk-seeking, since its utility function $-\sqrt{-c}$ is convex, but the convexity decreases the more negative c gets. The action costs that the agent has already accumulated have to be added to all rewards of a lottery. For example, if the accumulated action costs are -0.10, then lottery B becomes "rewards -0.20 and -1.10 with equal probability." If the agent has already accumulated cost -1.0, then lottery B becomes "rewards -1.10 and -2.00 with equal probability." Thus, the more action costs the agent accumulates, the more negative the total rewards become and the less risk-seeking the agent is. Since the agent accumulates more and more action costs over time, it becomes less and less risk-seeking.

This problem makes planning very inefficient. One can circumvent it with planning methods that have a limited look-ahead. The planner of [Kanazawa and Dean, 1989], for example, determines the plan that generates the largest expected total utility in the first n steps, executes the first action of the plan, and repeats. Such

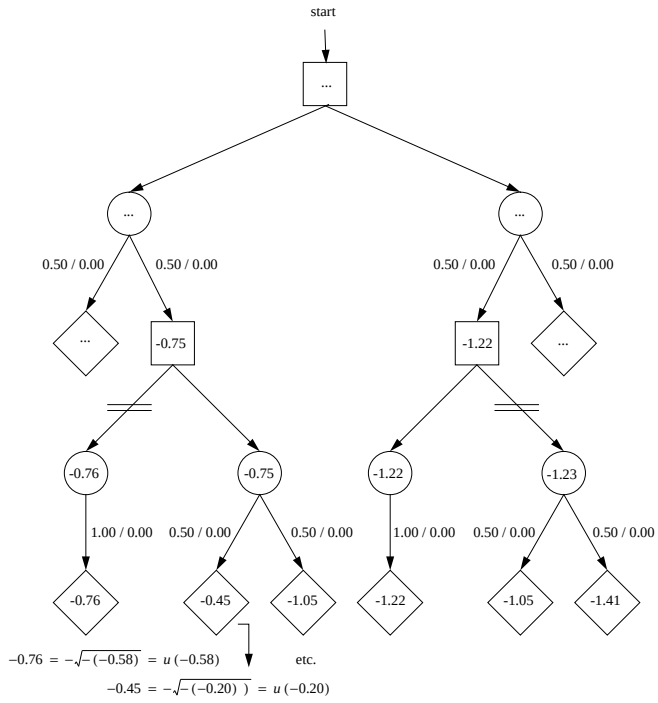


Figure 3: Solution for a Risk-Sensitive Agent

planning methods still duplicate shared subtrees (since they "unroll" the underlying Markov decision problem), but one can now control the amount of work required for one iteration by varying the look-ahead n. These, and related heuristic planning methods, suffer from the limited horizon problem and their success depends critically on the structure of the planning task.

4 A Solution

Our proposed method for incorporating risk-sensitive attitudes involves transforming planning task PT2 into a planning task PT3, which can then be solved with any standard (that is, risk-neutral) planning method. The resulting, optimal plan for planning task PT3 is optimal for the risk-sensitive planning task PT2 as well. The key to accomplishing this task is to utilize utility functions that maintain the Markov property.

Consider utility functions with the following property (called "constant local risk aversion" [Pratt, 1964] or "delta property" [Howard and Matheson, 1972]): if all rewards of an arbitrary lottery are increased by an arbitrary amount, then the certainty equivalent of the lottery is increased by this amount as well. (If the expected utility of a lottery is x, then $u^{-1}(x)$ is called its certainty equivalent.) The only utility functions with this property are the identity function, convex exponential functions $u(c) = \gamma^c$ for $\gamma > 1$, concave exponential functions $u(c) = -\gamma^c$ for $0 < \gamma < 1$, and their strictly positively linear transformations [Watson and Buede, 1987]. Since these utility functions are parameterized with a parameter γ, one can express a whole spectrum of risk-sensitivity, ranging from being strongly risk-averse to being strongly risk-seeking. The larger γ, the more risk-seeking the agent is, and vice versa. For γ approaching infinity, for example, the agent is "extremely risk-seeking": it assumes (wrongly) that nature does not flip coins to determine the outcomes of its actions, but makes the ones happen from which the agent benefits most [Koenig and Simmons, 1993]. Similarly, for γ approaching zero – the other extreme case – the agent thinks that nature hurts it as much as it can. Such "extremely risk-averse agents" believe in Murphy's law: If anything can go wrong, it will. They have recently been studied in the AI literature by [Moore and Atkeson, 1993] and [Heger, 1994].

These utility functions have the advantage that they maintain the Markov property [Howard and Matheson, 1972]: if the agent executes an action in its current state and behaves optimally afterwards, then it faces a lottery. There is one lottery for every action that the agent can execute in its current state. The lottery with the largest expected utility or, equivalently, the largest certainty equivalent identifies the optimal action. Before determining the certainty equivalents, however, one has to add the action costs that the agent has already accumulated to all (goal) rewards of every lottery. This increases the certainty equivalent of every lottery by the same amount (namely, the accumulated action costs), since the utility function has the delta property. Thus, when comparing lotteries, one can ignore the accumulated action costs.

[Howard and Matheson, 1972] apply utility functions with the delta property to Markov decision problems with finite and infinite time horizons. In the later case, they assume a non-goal oriented task, and every state-action mapping has to determine an irreducible (that is, strongly connected) Markov chain. As shown in [Koenig and Simmons, 1994], their analysis can be applied to non-goal oriented planning and reinforcement learning tasks if the agent is risk-sensitive towards variations of the reward that it receives per action execution. Unfortunately, our goal-oriented planning task PT2 does not possess the properties required by Howard and Matheson, and thus we cannot use their methods and proofs unchanged.

4.1 Planning for Risk-Seeking Agents

In the following, we will temporarily restrict our attention to risk-seeking agents with utility function $u(c) = \gamma^c$ (or any strictly positively linear transformation thereof) for risk parameter $\gamma > 1$. For these utility functions, we show how to calculate the expected total utility of a given plan. Then, we will transform

the planning problem into one for a risk-neutral agent and show how to solve it.

4.1.1 Calculating the Expected Total Utility of a Plan

Assume that, for some planning problem, a plan (that is, a state-action mapping) is given that assigns action $a[s]$ to non-goal state s. The expected total utility of this plan, that is, the expected total utility $u[s_0]$ of its start state s_0, can recursively be calculated as follows.

The (expected) total utility of a goal state s is $u[s] = u(r[s]) = \gamma^{r[s]}$. After the agent has executed action $a[s]$ in a non-goal state s, it incurs action cost $c^{a[s]}[s,s']$ and is in successor state s' with probability $p^{a[s]}[s,s']$. In state s', it faces a lottery again. This lottery has expected total utility $u[s']$ and certainty equivalent $u^{-1}(u[s']) = \log_\gamma u[s']$. According to the axioms of utility theory, the lottery can be replaced with its certainty equivalent. Then, the agent incurs a total reward of $c^{a[s]}[s,s'] + u^{-1}(u[s'])$ with probability $p^{a[s]}[s,s']$. Thus, the expected total utility of s can be calculated as follows:[4]

$$\begin{aligned}
u[s] &= \sum_{s' \in S} p^{a[s]}[s,s'] u(c^{a[s]}[s,s'] + u^{-1}(u[s'])) \\
&= \sum_{s' \in S} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s'] + u^{-1}(u[s'])} \\
&= \sum_{s' \in S} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} \gamma^{u^{-1}(u[s'])} \\
&= \sum_{s' \in S} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} u[s'] \\
&= \sum_{s' \in S \setminus G} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} u[s'] \\
&\quad + \sum_{s' \in G} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} \gamma^{r[s']}
\end{aligned}$$

This system of linear equations is always uniquely solvable.

4.1.2 Transforming the Planning Problem

To show how every planning task PT2 for a risk-seeking agent can be transformed into an equivalent planning task PT3 for a risk-neutral agent, we assume again that a state-action mapping is given. We use the same symbols for planning task PT3 that we used for PT2, but overline them.

Since (without loss of generality) a risk-neutral agent has utility function $\bar{u}(c) = c$, it holds that $\bar{u}[s] = \bar{v}[s]$.

[4] This corresponds to the policy-evaluation step in [Howard and Matheson, 1972] with the "certain equivalent gain" $\bar{g} = 0$.

A goal state s has (expected) total utility $\bar{u}[s] = \bar{u}(\bar{r}[s]) = \bar{r}[s]$. The expected total utility of a non-goal state s is

$$\begin{aligned}
\bar{u}[s] &= \sum_{s' \in S} \bar{p}^{a[s]}[s,s'] \bar{u}(\bar{c}^{a[s]}[s,s'] + \bar{u}^{-1}(\bar{u}[s'])) \\
&= \sum_{s' \in S} \bar{p}^{a[s]}[s,s'] (\bar{c}^{a[s]}[s,s'] + \bar{u}[s'])
\end{aligned}$$

Comparing these results with the ones in the previous section shows that $\bar{u}[s] = u[s]$ for all states $s \in S$ and all planning problems if and only if three equalities hold: $\bar{r}[s] = \gamma^{r[s]}$ for $s \in G$. Furthermore, $\bar{p}^{a[s]}[s,s'] = p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']}$ and $\bar{c}^{a[s]}[s,s'] = 0$ for $s \in S \setminus G$ and $s' \in S$.

Thus, planning task PT2 for a risk-seeking agent with utility function $u(c) = \gamma^c$ is equivalent to the following planning task PT3 for a risk-neutral agent:

> Introduce one additional goal state $\bar{s}$ with goal reward zero. Otherwise, the state space, action space, start state, and goal states remain unchanged. The goal reward of any goal state $s \neq \bar{s}$ is $\gamma^{r[s]}$. When the agent executes action a in a non-goal state s, it incurs an action cost of zero and is in successor state s' with transition probability $p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']}$. These probabilities do not sum up to one. With the complementary transition probability $1 - \sum_{s' \in S} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']}$, the agent incurs an action cost of zero and is in successor state $\bar{s}$.

Thus, given γ, one transforms planning task PT2 into the above planning task, for which one then determines the plan with the largest expected total reward. The transformation is trivial and can be done in linear time, since both representations are of the same size.

The only reason for introducing state $\bar{s}$ is to make the probabilities sum up to one. Since its expected total reward is zero, it will not show up in the calculations. The specification of PT3 for the risk-seeking planning problem from Figure 2 is shown in Figure 4. Note that, although they can both be expressed with probabilistic decision graphs of the same topology, the specification of the planning problem for PT3 differs fundamentally from the one of PT1. For example, an obvious difference is that all actions of planning task PT3 have action cost zero. Therefore, action costs can be ignored for risk-sensitive planning.

4.1.3 Finding Optimal Plans

Planning task PT3 can be solved with probabilistic AI planning methods or, alternatively, with dynamic

It turns out that the Markov decision problems for planning task PT3 have a simpler structure than the ones for PT1 (namely, all state-action mappings determine *absorbing* Markov chains). This simplifies the optimization algorithms.

In order to determine an optimal plan for planning task PT3, one can for example use value-iteration [Bellman, 1957], policy-iteration [Howard, 1964], Q-learning [Watkins, 1989], or linear programming. As an example of such a dynamic programming technique consider a simplified version of Howard's single-chain policy-iteration algorithm [Howard, 1964] [Howard and Matheson, 1972]. One can either use the algorithm on the transformed planning task PT3 or, as we have done here, adapt the algorithm so that it works on the original planning task PT2:

1. Choose an arbitrary state-action mapping $a[s] \in A(s)$ for all $s \in S \setminus G$.
2. (value-determination operation) Solve the system of linear equations

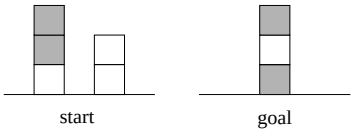
$$u[s] = \sum_{s' \in S \setminus G} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} u[s'] + \sum_{s' \in G} p^{a[s]}[s,s'] \gamma^{c^{a[s]}[s,s']} \gamma^{r[s']}$$

   for $s \in S \setminus G$.
3. If no $u[s]$ for any $s \in S \setminus G$ has changed in the previous step (from the value that it had in the previous iteration), then stop. An optimal state-action mapping is to select action $a[s]$ in state $s \in S \setminus G$.
4. (policy-improvement routine) Set for every $s \in S \setminus G$

$$a[s] := \arg\max_{a \in A(s)} \Big(\sum_{s' \in S \setminus G} p^{a}[s,s'] \gamma^{c^{a}[s,s']} u[s'] + \sum_{s' \in G} p^{a}[s,s'] \gamma^{c^{a}[s,s']} \gamma^{r[s']} \Big)$$

5. Go to 2.

This algorithm is an anytime algorithm. The term "anytime algorithm" was coined by [Boddy and Dean, 1989]), and [Bresina and Drummond, 1990] first developed an anytime planner. The policy-iteration algorithm is an anytime algorithm in the sense that the expected total utility of no state can decrease from one iteration to the next, but the expected total utility of at least one state strictly increases, until the optimal state-action mapping is found in finite time [Howard, 1964]. Thus, the expected total utility of the currently best plan cannot decrease from iteration to iteration. A solved state remains solved in the following iterations and an admissible plan stays admissible. Anytime planning methods can be used to determine — according to decision-theoretic criteria — when to stop planning and start executing the plan, because the possible future increase in plan quality does not justify the effort of planning any further.



There are seven goal states, all of which are equally preferable.

In every blocks-world state, one can move a block that has a clear top onto either the table or a different block that has a clear top, or paint a block white or black.

Moving a block takes only one minute to execute, but is very unreliable. With probability 0.10, the moved block ends up at its intended destination. With probability 0.90, however, the gripper loses the block and it ends up directly on the table. (Thus, moving a block to the table always succeeds.)

Painting a block takes three minutes and is always successful.

Figure 6: A Blocks-World Problem

However, dynamic programming algorithms are brute-force search algorithms and thus are often impractical, since they do not utilize available domain knowledge such as how different actions interact with each other. AI planning methods, on the other hand, are knowledge-based. Although AI planning methods, such as the ones of [Smith, 1988] or [Dean *et al.*, 1993], are usually not able to guarantee the optimality of their plans, they can be used for risk-seeking planning instead of Markov decision algorithms. The larger the expected total reward of the plan that they determine for planning task PT3, the larger is the expected total utility of the same plan for the corresponding planning task PT2.

4.1.4 Example: A Stochastic Blocks-World

We use the blocks-world problem that is stated in Figure 6 to illustrate this planning method. Figure 7 shows four of the state-action mappings that solve it, and Figure 8 illustrates how the certainty equivalents $u^{-1}(u[s_0]) = \log_\gamma u[s_0]$ of the four plans vary with the natural logarithm of the risk parameter γ.

Plan A, which involves no risk and can be executed in six minutes (that is, has total reward -6.00), has the largest expected total reward of all plans (not just the four plans shown) and will therefore be chosen by a risk-neutral agent. However, plan A is not necessarily optimal for a risk-seeking agent. When executing plan D, for example, the agent can reach a goal state in

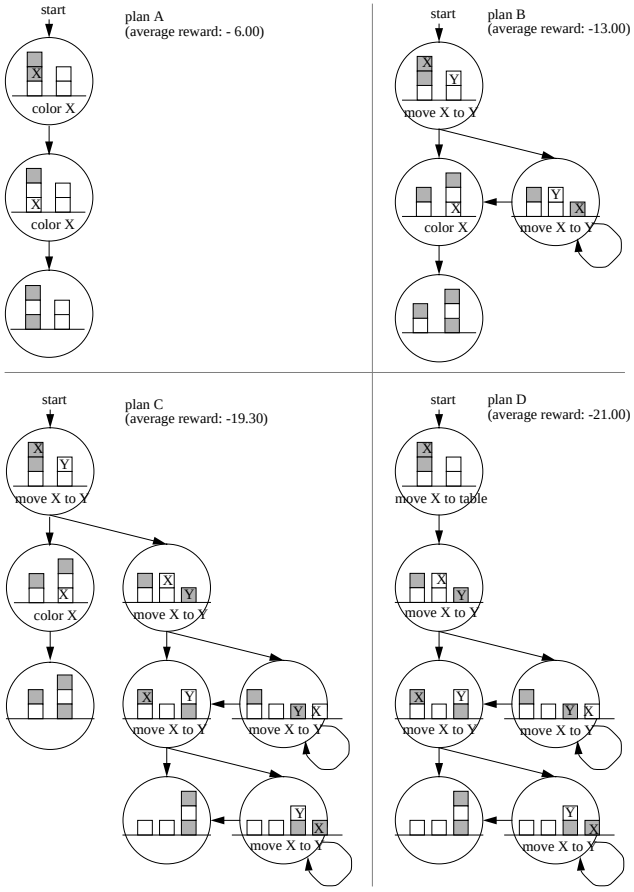


Figure 7: Some Plans for the Blocks-World Problem

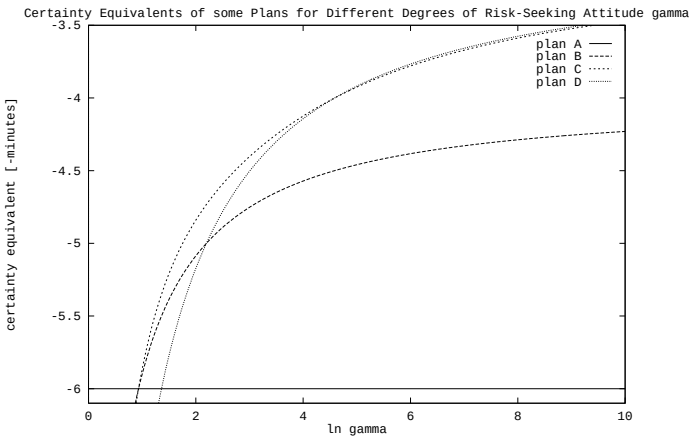


Figure 8: Certainty Equivalents (Risk-Seeking Case)

only three minutes if it is lucky.

The optimal plan for a risk-seeking agent is the one with the largest expected total utility or, equivalently, certainty equivalent. Since Plan A is deterministic, its certainty equivalent equals the (expected) total reward of its start state, no matter what the risk-attitude of the agent is. The other three plans are non-deterministic. Thus, their certainty equivalents increase, the more risk-seeking the agent becomes, and different plans can be optimal for different degrees of risk-seeking attitude. Figure 8 shows that plan A is optimal in the interval $\ln\gamma \in (0.00, 0.93]$. For $\ln\gamma \in [0.94, 4.58]$, plan C is optimal, and plan D should be chosen for $\ln\gamma \in [4.59, \infty)$. (These statements hold for all plans, not just the four plans shown in the figure.)

In order to be able to apply probabilistic planning methods other than Markov decision algorithms, we explicitly transform the planning problem into one for a risk-neutral agent. The original planning problem can for example be expressed with augmented STRIPS-rules [Koenig, 1991], three for the move actions ("move block X from the top of block Y on top of block Z," "stack block X on top of block Y," "unstack block X from block Y") and one for the paint action ("paint block X with color C"). The first move action can be expressed as follows:

```
move(X,Y,Z)
 precond:  on(X,Y), clear(X), clear(Z),
           block(X), block(Y), block(Z),
           unequal(X,Z)
  outcome:
   /* the primary outcome */
   prob:   0.1
   reward: -1.0
   delete: on(X,Y), clear(Z)
   add:    on(X,Z), clear(Y)
  outcome:
   /* failure: block X falls down */
   prob:   0.9
   reward: -1.0
   delete: on(X,Y)
   add:    clear(Y),on(X,table)
```

The transformation changes the transition probabilities, action costs, and goal rewards. In particular, the STRIPS-rules are transformed as shown in Section 4.1.2. For example, for $\gamma = 2$, the above STRIPS-rule is transformed into the following one:

```
move(X,Y,Z)
 precond:  on(X,Y), clear(X), clear(Z),
           block(X), block(Y), block(Z),
           unequal(X,Z)
  outcome:
```

```
  /* the primary outcome */
  prob:   0.05
  reward: 0.0
  delete: on(X,Y), clear(Z)
  add:    on(X,Z), clear(Y)
outcome:
  /* failure: block X falls down */
  prob:   0.45
  reward: 0.0
  delete: on(X,Y)
  add:    clear(Y),on(X,table)
```

With the complementary probability (0.5), the action execution results in the new goal state $\bar{s}$, that has goal reward zero, but is not modeled explicitly. All other goal states (i.e. the goal states of the original, risk-seeking planning problem) get assigned a goal reward of one. Now, one can use any planning method that maximizes expected total reward on the transformed STRIPS-rules to determine an optimal plan for the risk-seeking agent.

4.2 Planning for Risk-Averse Agents

For risk-averse agents, one can proceed as outlined for risk-seeking agents in the previous section. In this case, one has to use a function from the family $u(c) = -\gamma^c$ (or any strictly positively linear transformation thereof) for $0 < \gamma < 1$. Although the values $p^{a[s]}[s, s']\gamma^{c^{a[s]}[s,s']}$ can no longer be interpreted as probabilities (since $\sum_{s' \in S} p^{a[s]}[s, s']\gamma^{c^{a[s]}[s,s']} > 1$), one can use the same methods as in the risk-seeking case if one takes care of one complication: The solution $u[s_0]$ of the system of linear equations from Section 4.1.1 can now be finite even for plans that have expected total utility minus infinity. The planning methods can then erroneously return such plans as optimal solutions. Fortunately, these plans are easy to characterize ("plans that have at least one cycle with 'probability' greater than one"), and one can remedy the problem by either initializing the dynamic programming algorithms more restrictedly or extending them slightly. Details are given in [Koenig and Simmons, 1994].

If there are cycles in probabilistic decision graphs, then — unfortunately — the expected total utilities of admissible plans (and thus their certainty equivalents) can be minus infinity. Imagine for example an extremely risk-averse agent. Thus, given a plan, the agent assumes that nature will try to keep it away from a goal state. The agent assigns a plan an expected total utility of minus infinity if a vicious nature could indeed prevent it from reaching a goal state. In this case, utility theory might no longer be able to distinguish admissible plans from inadmissible ones. Table 1 shows that this problem can not arise for risk-neutral or risk-seeking agents.

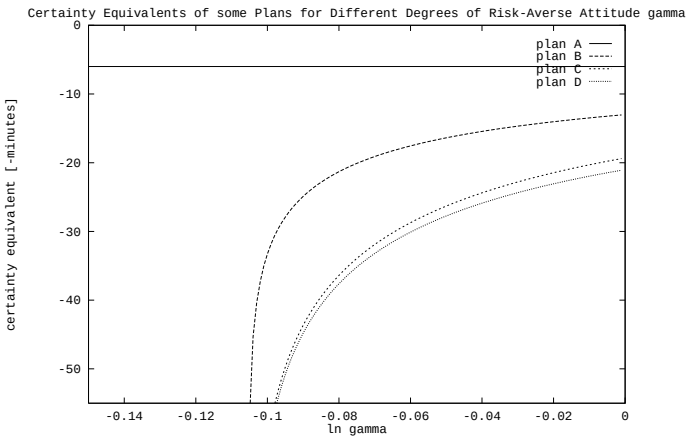


Figure 9: Certainty Equivalents (Risk-Averse Case)

As an example, consider again the blocks-world domain from Section 4.1.4. Figure 9 shows how the certainty equivalents of the four plans for the blocks-world problem vary with the natural logarithm of the risk parameter γ if the agent is risk-averse. The optimal plan for such an agent is always plan A, independent of γ. Although the certainty equivalent of plan A is defined for all values of $\ln \gamma$, the certainty equivalents of plans B, C, and D are finite only for $-0.11 \leq \ln \gamma < 0$ (that is, $0.9 < \gamma < 1$). They are minus infinity for smaller values of $\ln \gamma$.

5 Conclusion

This paper focuses on probabilistic planning for *risk-sensitive* agents, since there are many situations where it is not optimal to determine plans that minimize expected execution cost. We use acyclic and cyclic probabilistic decision graphs as the planning framework and use utility functions that possess the delta property. These utility functions cover a whole spectrum of risk-sensitive attitudes from being strongly risk-averse to being strongly risk-seeking, and fill a gap between approaches previously studied in the AI literature.

We have shown that one can use standard probabilistic planning methods to solve risk-sensitive planning problems. However, it is not enough to replace all costs and rewards with their respective utilities. Instead, one can transform the acyclic probabilistic decision graph into a different probabilistic decision graph of equal size, that one can then optimize for a risk-neutral agent in linear time with dynamic programming methods. Cyclic probabilistic decision graphs can be solved in a similar way in polynomial time with Markov decision algorithms.

This approach to risk-sensitive planning allows one to

augment risk-neutral probabilistic AI planning algorithms, since one can use *any* planning method on the transformed planning problem that minimizes (or satisfices) expected execution cost to determine an optimal (or satisficing) plan for a risk-seeking agent. The better a plan is for the transformed planning problem, the better it is for the original planning problem as well. Although the derivation of the transformation requires some knowledge of utility theory and Markov decision theory, the transformation itself is very simple and can be applied without any understanding of the formalisms involved.

We believe that much of the work in operations research or decision theory can be utilized for AI research in a similar way. These disciplines have a different approach to decision making than AI and, consequently, most of their methods might not be interesting from an AI point of view. However, they also offer results that are useful for other problem solving approaches. These results can (and should) be utilized by AI researchers.

Acknowledgements

Thanks (in alphabetical order) to Justin Boyan, Lonnie Chrisman, Matthias Heger, Andrew Moore, Joseph O'Sullivan, Stuart Russell, Jiří Sgall, and Michael Wellman for helpful discussions or comments. This research was supported in part by NASA under contract NAGW-1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

References

(Bellman, 1957) Bellman, R. 1957. *Dynamic Programming.* Princeton University Press, Princeton, NJ.

(Boddy and Dean, 1989) Boddy, M. and Dean, T. 1989. Solving time-dependent planning problems. In *Proceedings of the IJCAI.* 979–984.

(Bresina and Drummond, 1990) Bresina, J. and Drummond, M. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the AAAI.* 138–144.

(Christiansen and Goldberg, 1990) Christiansen, A.D. and Goldberg, K.Y. 1990. Robotic manipulation planning with stochastic actions. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control.*

(Dean *et al.*, 1993) Dean, T.; Kaelbling, L.P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings of the AAAI.* 574–579.

(Etzioni, 1991) Etzioni, O. 1991. Embedding decision-analytic control in a learning architecture. *Artificial Intelligence* (1-3):129–159.

(Filar *et al.*, 1989) Filar, J.A.; Kallenberg, L.C.M.; and Lee, H.-M. 1989. Variance-penalized Markov decision processes. *Mathematics of Operations Research* 14(1):147–161.

(Goodwin and Simmons, 1992) Goodwin, R. and Simmons, R.G. 1992. Rational handling of multiple goals for mobile robots. In *Proceedings of the First International Conference on AI Planning Systems.* 70–77.

(Haddawy and Hanks, 1992) Haddawy, P. and Hanks, S. 1992. Representation for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning.*

(Hansson *et al.*, 1990) Hansson, O.; Mayer, A.; and Russell, S. 1990. Decision-theoretic planning in BPS. In *Proceedings of the AAAI Spring Symposium on Planning.*

(Heger and Karsten, 1992) Heger, M. and Karsten, B. 1992. Risikoloses Reinforcement-Lernen. *KI* 4:26–32.

(Heger, 1994) Heger, M. 1994. Risk and reinforcement learning. Technical report, Computer Science Department, University of Bremen, Bremen, Germany. (a shorter version has been accepted for publication in the Proceedings of the Eleventh International Machine Learning Conference, 1994).

(Howard and Matheson, 1972) Howard, R.A. and Matheson, J.E. 1972. Risk-sensitive Markov decision processes. *Management Science* 18(7):356–369.

(Howard, 1964) Howard, R.A. 1964. *Dynamic Programming and Markov Processes.* The MIT Press, Cambridge, MA, third edition.

(Kanazawa and Dean, 1989) Kanazawa, K. and Dean, T. 1989. A model for projection and action. In *Proceedings of the IJCAI.* 985–990.

(Karakoulas, 1993) Karakoulas, G.J. 1993. A machine learning approach to planning for economic systems. In *Proceedings of the Third International Workshop on Artificial Intelligence in Economics and Management.*

(Koenig and Simmons, 1993) Koenig, S. and Simmons, R.G. 1993. Utility-based planning. Technical Report CMU–CS–93–222, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

(Koenig and Simmons, 1994) Koenig, S. and Simmons, R.G. 1994. Risk-sensitive game-playing, anytime planning, and reinforcement learning. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. (forthcoming).

(Koenig, 1991) Koenig, S. 1991. Optimal probabilistic and decision-theoretic planning using Markovian

decision theory. Master's thesis, Computer Science Department, University of California at Berkeley, Berkeley, CA. (available as Technical Report UCB/CSD 92/685).

(Kushmerick *et al.*, 1993) Kushmerick, N.; Hanks, S.; and Weld, D. 1993. An algorithm for probabilistic planning. Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

(Mine and Osaki, 1970) Mine, Hisashi and Osaki, Shunji 1970. *Markovian Decision Processes.* American Elsevier, New York, NY.

(Moore and Atkeson, 1993) Moore, A.W. and Atkeson, C.G. 1993. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Proceedings of the NIPS.*

(Pratt, 1964) Pratt, J.W. 1964. Risk aversion in the small and in the large. *Econometrica* 32(1-2):122–136.

(Raiffa, 1968) Raiffa, H. 1968. *Decision Analysis: Introductory Lectures on Choices under Uncertainty.* Addison Wesley, Menlo Park, CA.

(Russell and Wefald, 1991) Russell, S. and Wefald, E. 1991. *Do the Right Thing – Studies in Limited Rationality.* The MIT Press, Cambridge, MA.

(Smith, 1988) Smith, D.E. 1988. A decision-theoretic approach to the control of planning search. Technical Report LOGIC-87-11, Department of Computer Science, Stanford University, Palo Alto, CA.

(von Neumann and Morgenstern, 1947) von Neumann, J. and Morgenstern, O. 1947. *Theory of games and economic behavior.* Princeton University Press, Princeton, NJ, second edition.

(Watkins, 1989) Watkins, C.J. 1989. *Learning from Delayed Rewards.* Ph.D. Dissertation, King's College, Cambridge University, Cambridge (Great Britain).

(Watson and Buede, 1987) Watson, S.R. and Buede, D.M. 1987. *Decision Synthesis.* Cambridge University Press, Cambridge (Great Britain).

(Wellman and Doyle, 1992) Wellman, M. and Doyle, J. 1992. Modular utility representation for decision theoretic planning. In *Proceedings of the First International Conference on AI Planning Systems.* 236–242.