Utility-Based Planning

Sven Koenig and Reid G. Simmons

November 1993

CMU-CS-93-222

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Parts of this Technical Report will appear in the *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, 1994

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.

Abstract

Probabilistic planning methods can have various objectives: usually they either maximize the probability of goal achievement or minimize the expected execution cost of the plan, and therefore assume that the agent that executes the plan has a risk-neutral attitude. Although there are many situations where risk-sensitive behavior is more appropriate, researchers have largely ignored the question how to incorporate risk-sensitive attitudes into their planning mechanisms. Utility theory shows that it is rational to maximize expected utility, given that the agent accepts a few simple axioms and has unlimited planning resources available. Thus, researchers might believe that one could allow for risk-sensitive attitudes by replacing all costs with their respective utilities (for an appropriate utility function). We show that this is usually not the case and, moreover, that — in general — the best action in a state can depend on the total cost that the agent has already accumulated (that is, the Markov property does not need to hold). However, 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 probabilistic AI planning methods or, alternatively, dynamic programming methods. If the agent is risk-seeking, then one can use *any* planning method on the transformed planning problem that either minimizes (or satisfices) the expected execution cost or, alternatively, maximizes (or satisfices) the probability of goal achievement. The better a plan is for the transformed planning problem, the better it is for the original planning problem as well. Thus, one can extend the functionality of these planners to risk-sensitive planning. — To demonstrate our algorithms, 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 (goodness). We then apply probabilistic decision graphs to finding optimal plans for risk-sensitive agents in a stochastic blocks-world domain and a stochastic path-planning domain.

1 Introduction

In recent years, numerous planning methods have been developed that are able to deal with stochastic domains.[1] Probabilistic planning methods are important, because actions in the real world usually do not have deterministic effects. This uncertainty can be an inherent part of the domain, but is often due to limitations of the planner or the agent that executes the plan: the planner might have an insufficient model of the domain, or the agent might not be able to execute its actions with the required precision.

Not all probabilistic planning methods have the same planning objective: different planners consider different plans to be optimal for the same planning problem. In this paper, we concentrate on three possible planning objectives: to maximize the probability of goal achievement, to minimize the expected execution cost, and to maximize the expected utility of plan execution.

In some domains, it is impossible to determine probabilistic plans that are guaranteed to achieve the given goal. When planning in such domains, one is usually satisfied with finding plans that succeed with a given probability, see [Bresina and Drummond, 1990] and [Kushmerick *et al.*, 1993]. Even if a plan exists that achieves a goal with probability one, time might not permit to find it. Then, planners often use an any-time approach: they increase the probability of goal achievement over time by incrementally extending the plan envelope.

In other domains, however, it is easy to construct plans that always succeed for sure (at least, in the limit). Then, one needs a criterion for choosing among these plans. Such a metric is for example the total execution cost of the plans[2]: 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 total execution cost is defined to be the sum of the resource consumption costs of all actions executed from the time at which the agent is started in the start state until it stops in a goal state (given that it obeys the plan).

Since the total execution cost of probabilistic plans can vary from plan execution to plan execution, almost all planning methods reported in the literature use the *expected* total execution cost as ranking criterion: Out of all plans that guarantee to achieve the given goal, they choose the one for execution that minimizes the expected total execution cost (when optimizing) or, at least, one whose expected total execution cost is smaller than a given number (when satisficing). Thus, they assume that the agent that executes the plan has a risk-neutral attitude.

[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.

[2]An example is travel planning. When planning how to get to a given conference, one can easily devise plans that are always successful (at least under normal circumstances). Possible evaluation metrics for these plans include travel cost or travel time.

However, people are usually not risk-neutral for non-repetitive planning problems. Whereas a risk-neutral agent does not care about the variance, a risk-seeking agent ("gambler") is willing to accept a plan with a larger expected total 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 total execution cost — the larger the variance of the total 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.

On the other hand, there are also many situations where people want to avoid risk. The plans produced by planning methods should reflect the risk-attitudes of the people that depend on them. However, researchers have largely ignored the question of how to incorporate risk-sensitive attitudes into their planning mechanisms. In fact, the term "decision-theoretic planning" is often incorrectly equated with optimal (meta-)planning for a *risk-neutral* agent.

It is possible to achieve a risk-sensitive attitude by ranking plans not according to their expected total execution costs, but according to their expected total execution costs plus or minus a fraction of the variances [Filar *et al.*, 1989] [Karakoulas, 1993], or by searching for plans whose total execution costs are optimal in the best or worst case ("nature acts like a friend or enemy") [Witsenhausen, 1966] [Heger, 1992], see also [Moore and Atkeson, 1993a]. 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. Its application to planning problems has been studied by [Etzioni, 1991], [Russell and Wefald, 1991], [Haddawy and Hanks, 1992], [Wellman and Doyle, 1992], [Goodwin, 1992], and others. Therefore, we would like to stay within this framework.

The utility-theoretic approach encompasses the other two approaches: Maximizing the probability of goal achievement is rational according to utility theory if the agent does not incur action costs, but receives total reward r_{goal} for achieving a goal state and total reward $r_{non-goal}$ otherwise, where $r_{non-goal} < r_{goal}$. Minimizing the expected total execution cost is rational according to utility theory if the agent is risk-neutral.

In the following, we will first review the basic concepts of utility theory and then 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 that are 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). Next, we will show that replacing all costs and rewards with their respective utilities, but leaving the planning mechanism unchanged, usually leads to erroneous results and, moreover, that — in general — 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. However, for utility functions with a certain property, we will demonstrate how planning problems for risk-sensitive agents can be transformed into equivalent planning problems for risk-neutral agents, that can then be solved with dynamic programming methods or, alternatively, probabilistic AI planning methods. 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. We will show that this approach can, at least for risk-seeking agents, be implemented with *any* AI planning method that minimizes (or satisfices) the expected total execution cost. The same statement holds, perhaps surprisingly, for *all* planners that maximize (or satisfice) the probability of goal achievement. Finally, we will use both a blocks-world and a path-planning example to illustrate our ideas and show how the optimal plan depends on the degree of risk-sensitivity of the agent.

The purpose of this paper is to motivate the importance of risk-sensitive planning and introduce our approach to it that builds on previous work by [Howard and Matheson, 1972] in the context of Markov decision theory. It contains a summary of our results, but describes the ideas only informally. Mathematical derivations, more precise statements, and additional qualifications are given in [Koenig and Simmons, 1994]. Although our discussion can easily be applied to meta-planning problems if planning resources are limited, that is, the theory of "limited rationality" [Horvitz, 1988] [Russell and Wefald, 1991] (for example, making decisions when to stop refining a plan and start to execute it), we assume throughout this paper for simplicity that the agent has unlimited planning resources available.

2 Utility Theory

In this section, we will briefly review some of the concepts of utility theory — a theory that explains why people do not always minimize expected costs or maximize expected rewards. Daniel Bernoulli studied around 1778 the following scenario, known as the "St. Petersburg paradox":[3]

> You can participate in a lottery at most once. The lottery involves the successive tossing of a fair coin until heads appears for the first time. If heads appears on the ith toss, you win 2^i dollars. What is the dollar

[3]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.

amount c_{max} that you are willing to pay at most for a ticket that allows you to participate in this lottery?

If people who own w dollars initially ("their wealth"), choose not to participate in the lottery, then they maintain their assets. On the other hand, if they buy the ticket for c dollars and then play the lottery, they own $2^i + w - c$ dollars afterwards with probability $1/2^i$. Thus, the expected reward of the lottery is $\sum_{i=1}^{\infty} \frac{2^i+w-c}{2^i} = \infty$ dollars afterwards. No matter how large the price of the ticket, the expected reward of participating in the lottery is larger than the expected reward of abstaining. Since we assume in this paper that people prefer a larger reward over a smaller one, they should be willing to wager all their assets if they were indeed maximizing expected rewards. In reality, however, most people offer only around ten dollars for the ticket.

Bernoulli suggested that people do not average over rewards, but over the pleasure ("utility") that the rewards provide. For every risk-attitude, there exists a strictly monotonically increasing utility function that transforms rewards c into real-valued utilities $u(c)$ such that it is rational to maximize expected utility, given that the decision maker accepts a few simple axioms and has unlimited planning resources available [von Neumann and Morgenstern, 1947].

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. If L is a lottery, then we use $u[L]$ to denote the expected utility of the lottery (or, in later sections, the expected utility of a state).[4] $u^{-1}(u[L])$ is called the certainty (monetary) equivalent of lottery L. Maximizing expected utility is therefore equivalent to maximizing certainty equivalents.

The utility of not participating in the St. Petersburg lottery is $u(w)$, where u is the utility function of the decision maker. Participating in the lottery results in the expected utility $\sum_{i=1}^{\infty} \frac{u(2^i+w-c)}{2^i}$. Thus, people should wager at most c_{max} dollars, where $u(w) = \sum_{i=1}^{\infty} \frac{u(2^i+w-c_{max})}{2^i}$. The following table shows c_{max} depending on w for people with utility function $u(c) = \log_2 c$ (for $c > 0$):

w	c_{max}
1.00	2.82
10.00	4.97
100.00	7.79
1000.00	10.95
10000.00	14.24
100000.00	17.55

These numbers appear to be in the right range, although Bernoulli did not claim that people have this particular utility function.

[4]Note the subtle difference: $u()$ maps costs into utilities, and $u[]$ transforms lotteries (or states) into expected utilities. If lottery L contains reward c_i with probability p_i, then $u[L] = \sum_i p_i u(c_i)$.

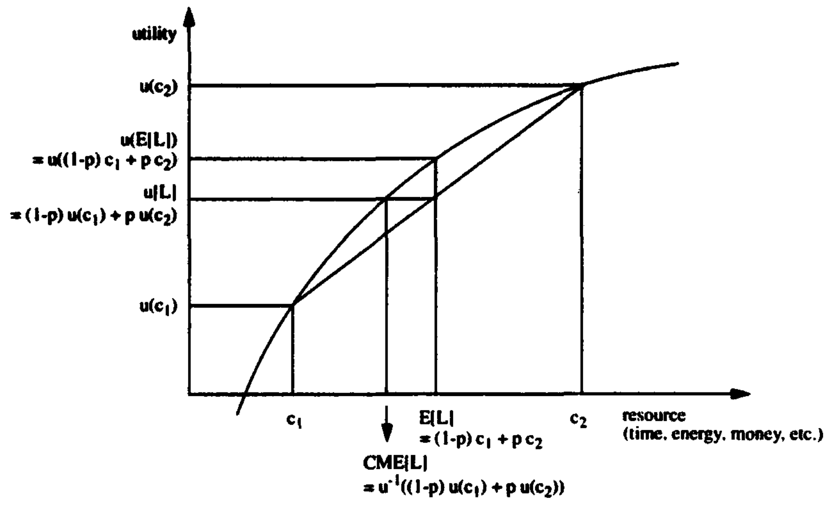


Figure 1: A Concave Utility Function

People are not willing to pay more for the ticket, because they are risk-averse. To them, losing parts of their stake has more weight than winning a fortune. More formally, people are called risk-averse (risk-seeking, risk-neutral) if and only if their expected utility of every non-deterministic lottery L is smaller than (larger than, equal to) their utility of the expected reward $E[L]$ of the same lottery. (People who are not risk-neutral are called risk-sensitive.) If risk-averse people have to decide between participating in an arbitrary non-deterministic lottery L and receiving its expected reward $E[L]$ for sure, they would prefer the latter option. In order for them to be indifferent between the two alternatives, the expected reward of the first lottery must be larger than the value of the reward of the second lottery. In other words, for risk-averse people, the certainty equivalent $CME[L]$ of every non-deterministic lottery L is smaller than its expected reward $E[L]$.

People are risk-averse (risk-seeking, risk-neutral) if and only if their utility functions are concave (convex, linear). Figure 1 illustrates, for example, why risk-averse people have concave utility functions. It shows for a lottery L with two rewards c_1 (won with probability $1 - p$) and c_2 (won with probability p) that indeed $u[L] < u(E[L])$ and $CME[L] < E[L]$.

Since utility functions encode the individual risk-preferences of people, they must be elicited from the decision maker on an individual basis. Since utility functions are determined only up to arbitrary strictly positively linear transformations, one can establish an arbitrary scale by fixing the utilities of two rewards, say the utility of the smallest possible reward c_1 and the one of the largest possible reward c_2. Assume $u(c_1) = 0$ and $u(c_2) = 1$. To determine the utility of any reward c, one asks the decision maker for the probability p that makes him/her indifferent between the following two lotteries: Either s/he receives reward $c - w$ for sure, or s/he receives reward $c_1 - w$ with probability $1-p$ and reward $c_2 - w$ with probability p, where w is the initial wealth of the person. Then, $u(c) = u(c - w + w) = (1-p)u(c_1 - w + w) + pu(c_2 - w + w) = p$.

3 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. Although we use this particular planning framework here, our method for integrating risk-sensitivity into planning applies to all planning frameworks that associate "action costs" with action-outcome pairs such that the execution of a plan can be evaluated according to the sum of the "action costs" of the executed actions and their outcomes ("time-additive value functions"). It can, for example, be used in conjunction with Smith's planner [Smith, 1988].

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 (artificial) agent has to execute in its current state. In a goal state s, the agent receives a (positive or negative) one-time reward ("goal reward") $r[s]$ 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 a negative reward[5] ("action cost") $c^a[s,s'] < 0$ and is in successor state s'. Thus, we assume in this paper that the outcomes of all action executions are mutually independent given the current state of the agent (Markov property). The action costs reflect the prices of the consumed resources, 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 $v[s]$ ("net profit") of state s for a given plan is the expected sum of the reward of the goal state (measured in some unit) and the total cost of the actions (measured in the same unit) 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

[5]The assumption $c^a[s,s'] < 0$ can be weakened. We adopt it throughout this paper for two reasons: First, it is consistent with goal oriented planning problems, since one has to put *effort* into achieving a goal. The goal itself is the *reward*. Second, it is a simple and sufficient condition for being able to guarantee that an agent that maximizes expected total reward should indeed try to reach a goal state.

$u[s]$ 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, 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 unsolved states to have an expected total execution cost 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.)

For risk-neutral agents, the planning framework is isomorphic to Markov decision problems [Mine and Osaki, 1970]: Assume that a planning problem is given. We distinguish the parameters of the corresponding Markov decision problem by adding a hat. For example, we use $\hat{S}$ for the set of states of the Markov decision problem. Then, the planning problem corresponds to the following Markov decision problem, where $\hat{s}$ and $\hat{a}$ are two arbitrary symbols with $\hat{s} \notin S$ and $\hat{a} \notin \bigcup_{s \in S \setminus G} A(s)$. Every state s of the Markov decision problem that corresponds to a goal state of the planning problem has exactly one applicable action, namely $\hat{a}$. The execution of this action causes the Markov process to receive reward $r[s]$ and change state to state $\hat{s}$. Action $\hat{a}$ is the only action applicable in state $\hat{s}$. It has action cost zero and leads deterministically back to $\hat{s}$. Once the Markov decision process has reached state $\hat{s}$, it can no longer leave the state and its expected total future reward is zero. To summarize, when the agent reaches a goal state s of the planning problem, it receives total future reward $r[s]$. This corresponds to the Markov process reaching state s of the Markov decision problem, in which case its total future reward is $r[s]$ as well.

$$\begin{aligned}
\hat{S} &:= S \cup \{\hat{s}\} \\
\hat{A}(s) &:= \begin{cases} A(s) & \text{if } s \in S \setminus G \\ \{\hat{a}\} & \text{if } s \in G \cup \{\hat{s}\} \end{cases} \quad \text{for all } s \in \hat{S} \\
\hat{p}^a[s,s'] &:= \begin{cases} p^a[s,s'] & \text{if } s \in S \setminus G \text{ and } s' \in S \\ 0 & \text{if } s \in S \setminus G \text{ and } s' = \hat{s} \\ 0 & \text{if } s \in G \cup \{\hat{s}\} \text{ and } s' \in S \\ 1 & \text{if } s \in G \cup \{\hat{s}\} \text{ and } s' = \hat{s} \end{cases} \quad \text{for all } s, s' \in \hat{S} \text{ and } a \in \hat{A}(s) \\
\hat{c}^a[s,s'] &:= \begin{cases} c^a[s,s'] & \text{if } s \in S \setminus G \text{ and } s' \in S \\ r[s] & \text{if } s \in G \text{ and } s' = \hat{s} \\ 0 & \text{if } s = \hat{s} \text{ and } s' = \hat{s} \end{cases} \quad \text{for all } s, s' \in \hat{S} \text{ and } a \in \hat{A}(s)
\end{aligned}$$

Although the planning framework is isomorphic to Markov decision problems and can therefore be described with a STRIPS-like notation [Koenig, 1991][6], we use an easier-to-depict representation here ("probabilistic decision graphs") that resembles the kind of

[6]For the original STRIPS-notation, that applies only to deterministic domains, see [Fikes and Nilsson, 1971].

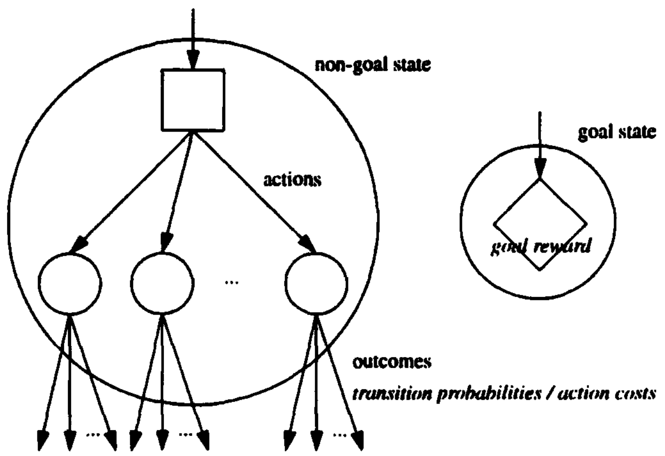


Figure 2: Building Blocks of Probabilistic Decision Graphs

decision trees that are used in utility theory. Its building blocks are shown in Figure 2. 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 $a \in A(s)$. For every outcome of an action, one specifies its transition probability and action cost. 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." As an illustration, consider a simple planning problem in a blocks-world with three indistinguishable blocks. In every blocks-world state, one can move a block that has a clear top onto either the table or a different block with a clear top. Moving a block takes one minute. With probability 0.1, the moved block ends up at its intended destination. With probability 0.9, however, the gripper loses the block and it ends up directly on the table. (Thus, moving a block to the table always succeeds.) In such a probabilistic blocks-world, it is easy to determine at least one admissible plan for every solvable planning problem: First, unstack every block. (Remember that moving a block to the table always succeeds.) Then, stack blocks as needed for the goal configuration. (If a stacking action fails, repeat it until it finally succeeds.) Figure 3 shows the representation of the planning problem to unstack fully a stack of three blocks.[7] Figure 4 contains one particular state-action mapping ("plan") that solves the planning problem. 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. Figure 4 also

[7]The symbols "X" and "Y" are introduced only to enable us to express the actions. They do *not* mark specific blocks and can therefore be used to denote different blocks in different states.

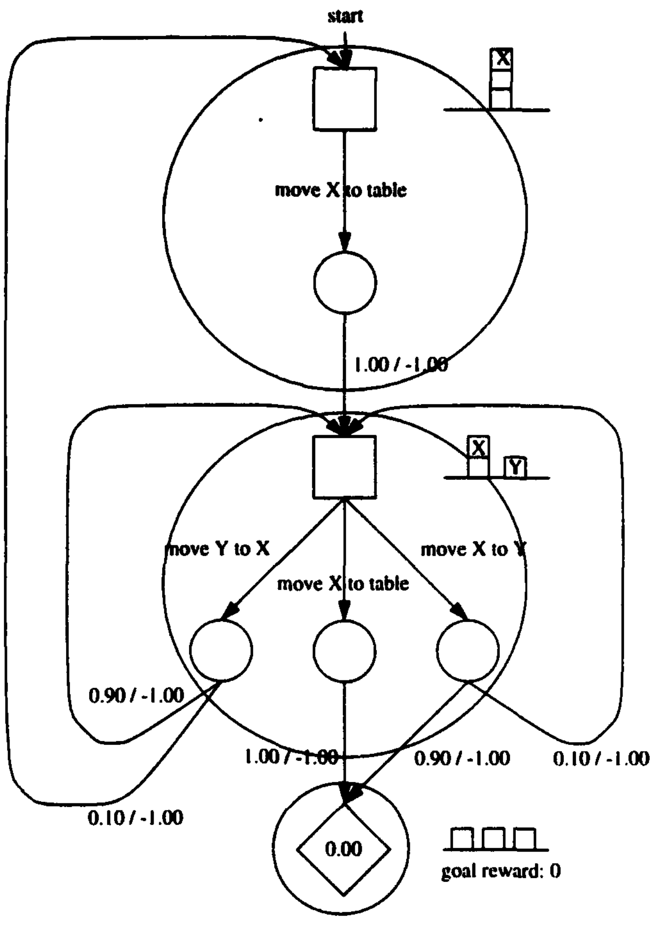


Figure 3: A Probabilistic Decision Graph ("Planning Problem")

introduces the shorthand notation that we will use in this paper to represent plans.

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 decision theory. For example, Figure 5 represents the planning problem to decide between "buying a ticket for the St. Petersburg lottery for c dollars" or "abstaining from the lottery" as an acyclic probabilistic decision graph without shared subtrees and without action costs (that is, all action costs are zero).[8]

[8]Note that the decision tree has — in contrast to the planning framework outlined above — an action with an infinite number of outcomes.

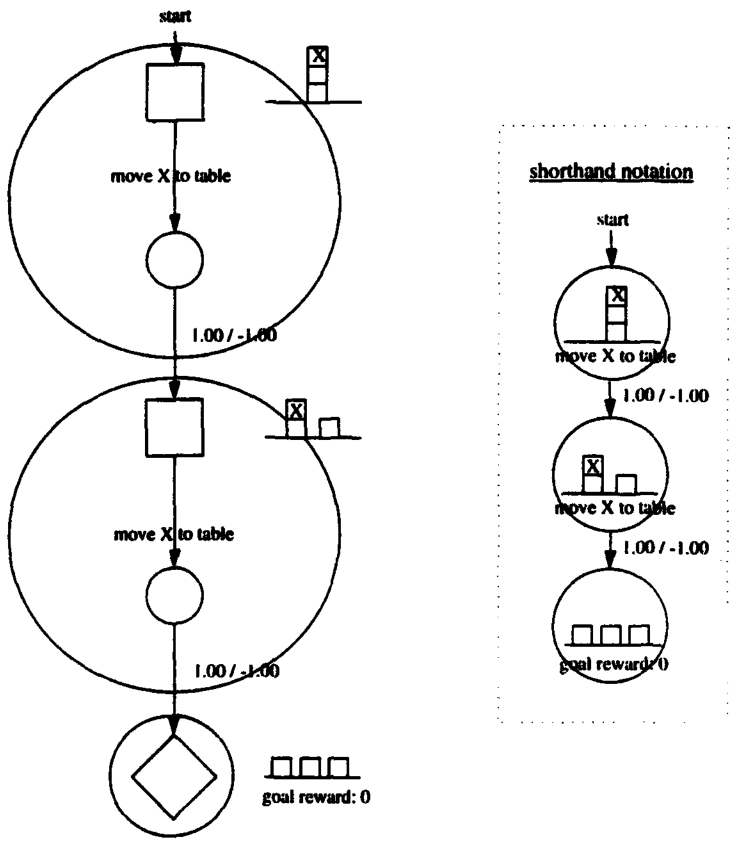


Figure 4: A State-Action Mapping ("Plan")

4 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 their planning mechanisms. In the following, we will use acyclic probabilistic decision graphs to demonstrate that this is not necessarily the case, but first we will review how to use dynamic programming techniques to determine optimal plans for risk-neutral agents.

4.1 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.

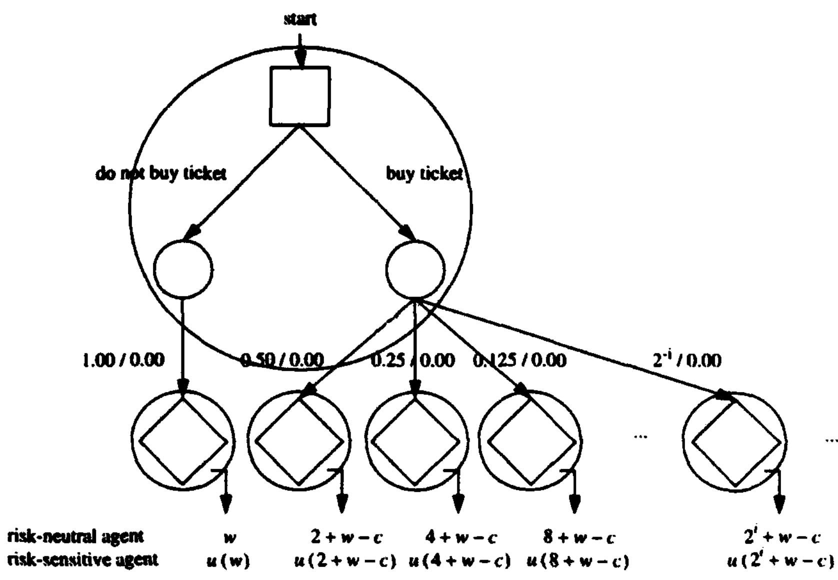


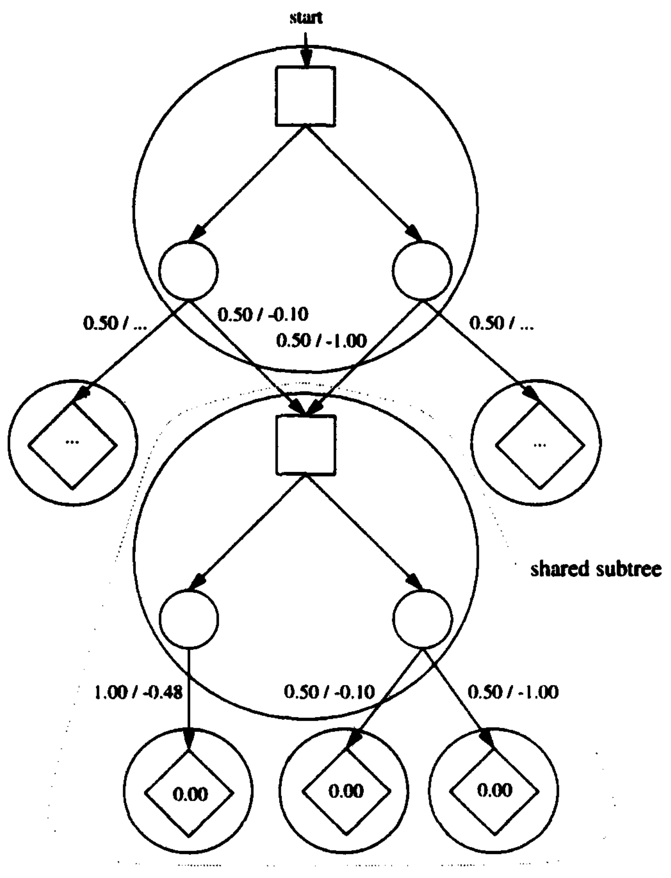
Figure 5: St. Petersburg Paradox

First, we would like to point out that every planning task PT1 can be transformed into an equivalent planning task that has only one goal state, which has goal reward zero. Planning task PT1 is equivalent to the following planning task PT1' for a risk-neutral agent:

> Introduce a new goal state with goal reward zero, and replace every (old) goal state $s \in G$ with a non-goal state in which only one action can be executed, that leads with probability one and action cost $r[s] - \max_{s' \in G} r[s'] - 1 < 0$ to the new goal state. Otherwise, the planning problem remains unchanged.

Planning task PT1' satisfies all requirements of our planning framework. It is equivalent to planning task PT1 in the following sense: the better a plan is for planning task PT1, the better it is for planning task PT1' as well, and vice versa. This is the case, because the expected total reward of every admissible plan for planning task PT1' is $\max_{s' \in G} r[s'] + 1$ lower than the expected total reward of the same plan for planning task PT1. An inadmissible plan has expected total reward minus infinity for both planning tasks. (Note that a plan is either admissible for both planning tasks or inadmissible for both of them.)

An optimal state-action mapping for planning task PT1 can be determined in polynomial time with dynamic programming techniques ("Markov(ian) decision algorithms"). As an example for how to solve a given planning task PT1 for *acyclic* probabilistic decision graphs consider the (only partly specified) example shown in Figure 6. To solve



"..." means: irrelevant for the example

Figure 6: A Planning Problem with a Shared Subtree

this cost-annotated decision tree, we could transform it. 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, as shown in Figure 7: the expected total reward of a value node is the goal reward, 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 equals the expected total reward of the decision node that it contains, and the optimal action $a[s]$ for the state is the action that is optimal of its decision node. Actions that are sub-optimal are "crossed out" with two horizontal lines in the figures.

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 there-

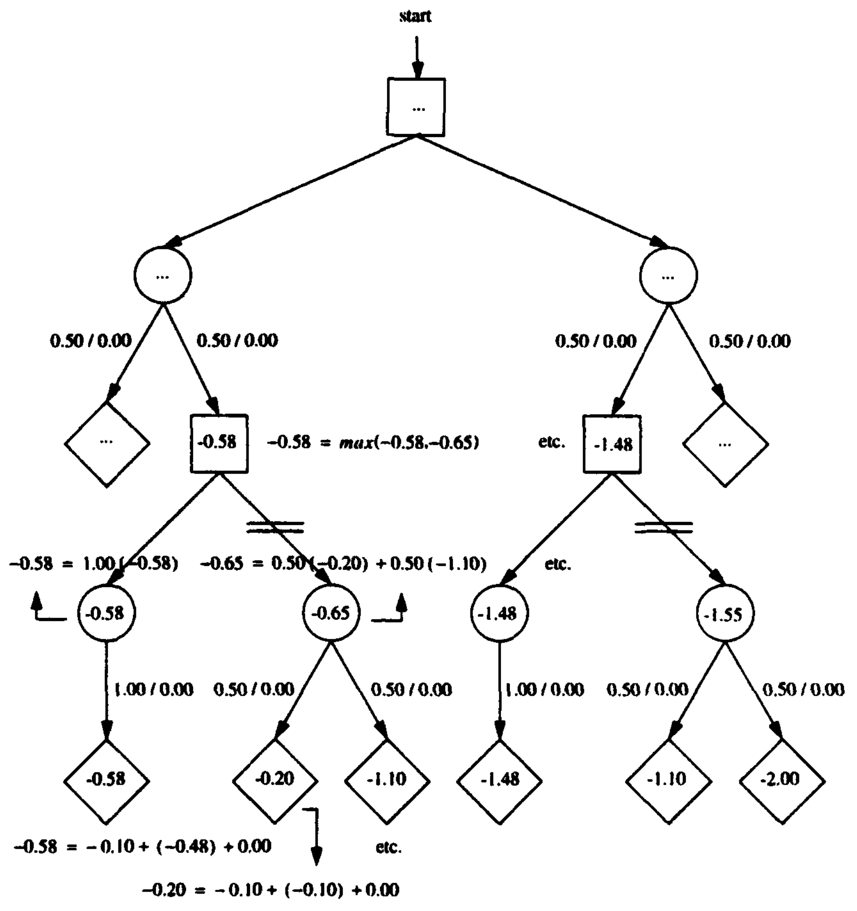


Figure 7: Solution for a Risk-Neutral Agent I

fore 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, as shown in Figure 8. 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.

$$\begin{aligned} v[s] &:= \begin{cases} r[s] & \text{for } s \in G \\ \max_{a \in A(s)} \sum_{s' \in S} p^a[s,s'](c^a[s,s'] + v[s']) & \text{otherwise} \end{cases} \\ a[s] &:= \operatorname{argmax}_{a \in A(s)} \sum_{s' \in S} p^a[s,s'](c^a[s,s'] + v[s']) \quad \text{for } s \in S \setminus G \end{aligned}$$

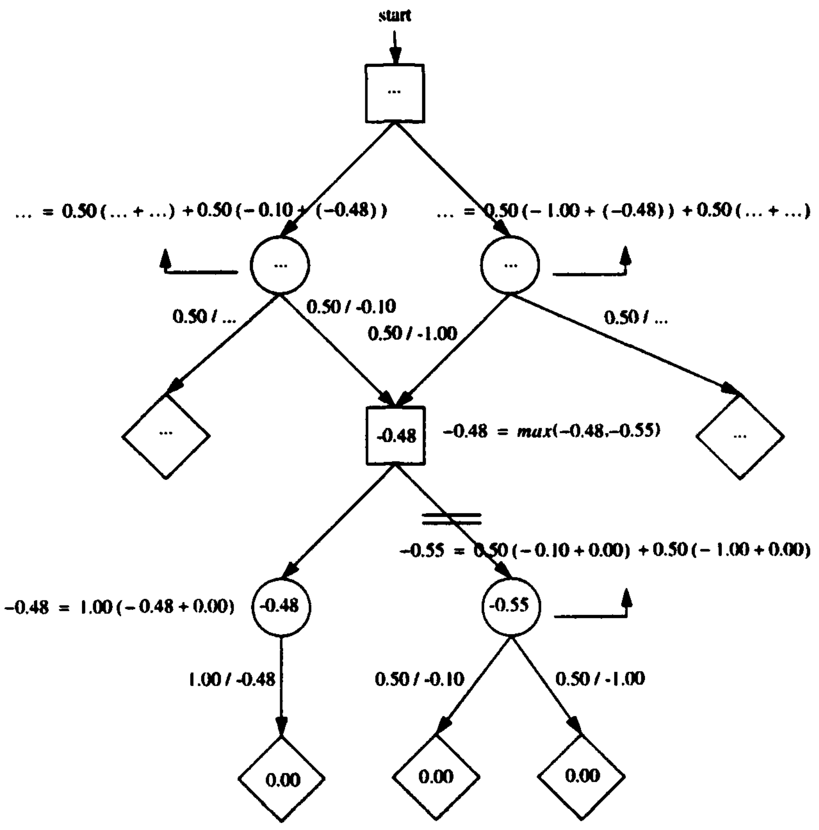


Figure 8: Solution for a Risk-Neutral Agent II

Thus, one evaluates every subtree only once and the run-time of the algorithm is linear in the size of the original decision tree.

4.2 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.[9] As outlined earlier, 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. This is illustrated for the St. Petersburg paradox in Figure 5.

In reality, however, the probabilistic decision graphs of planning task PT2 do have

[9]The expected total utility $u[s]$ of a state s for a given plan is the expected utility of the total reward (that is, goal reward plus action costs) received when starting in s and executing the plan. We have to use $u[s]$ instead of $u(s)$, since starting in s corresponds to participating in a lottery.

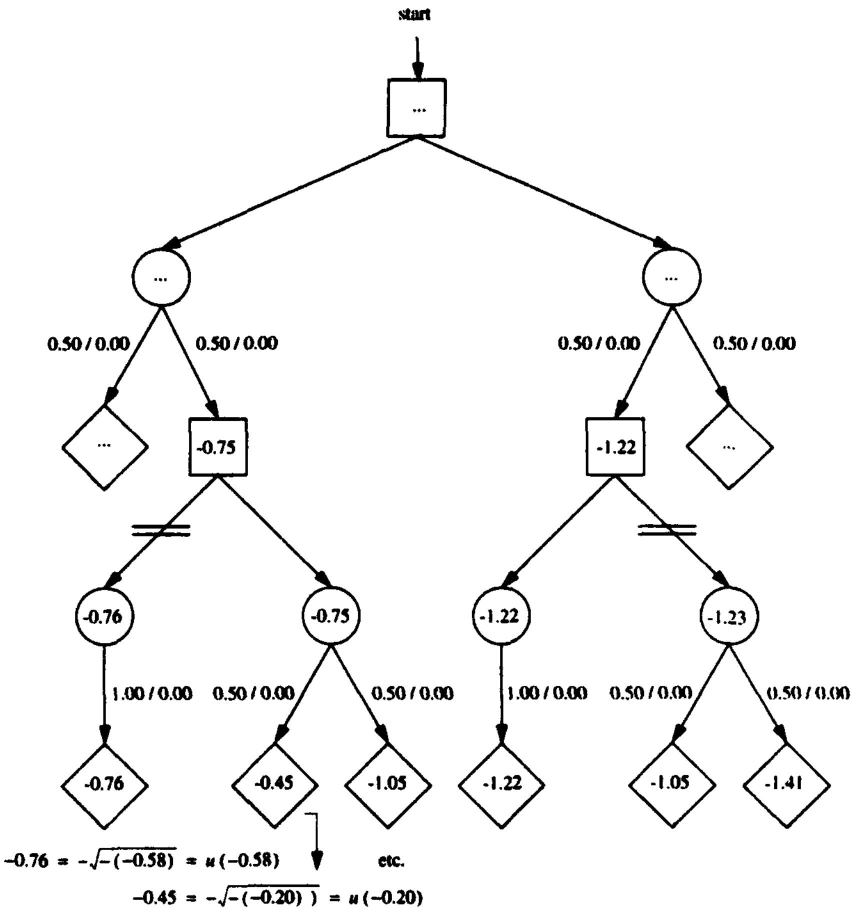


Figure 9: Solution for a Risk-Sensitive Agent

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).[10] Consider for example the planning problem for a risk-neutral agent again that was shown in Figure 7. The corresponding planning task for a risk-seeking agent with utility function $u(c) = -\sqrt{-c}$ (for $c \leq 0$) is shown in Figure 9.

It is not optimal to simply replace all costs and rewards with their respective utilities,

[10]For acyclic probabilistic decision graphs and some utility functions one can try to do much better by remembering for every state the functional dependency between the utility of the state and the wealth of the agent. However, it is important to keep in mind that a probabilistic planning method must be able to cope with *cyclic* probabilistic decision graphs as well, since — for many planning problems — either the agent cannot avoid to repeat some of the states or it is not optimal to do so.

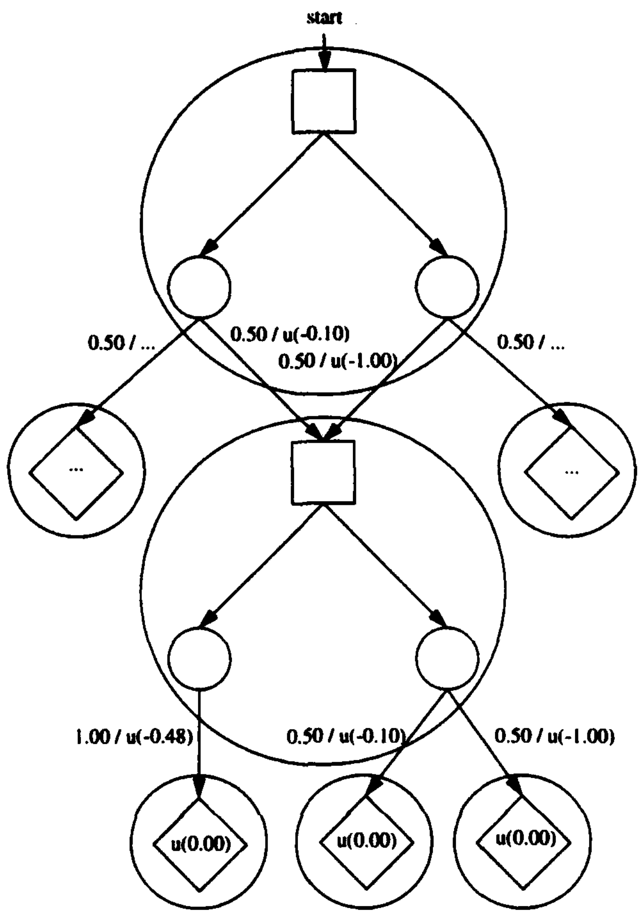


Figure 10: Transformation I (does not work)

as shown in Figure 10, 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 cannot be used any longer *in any way* without considering the wealth of the agent, because the Markov property does not necessarily hold for risk-sensitive agents [Raiffa, 1968].

Consider again the planning problem that was stated in Figure 6. The shared subtree represents the choice between a deterministic lottery A (reward -0.48 for sure) and a non-deterministic lottery B (rewards -0.10 and -1.00 with equal probability). As demonstrated in Figure 9 for $u(c) = -\sqrt{-c}$, the agent should choose lottery B if it has wealth -0.10 when deciding between the two lotteries. However, if its wealth is -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.[11] Remember that the wealth of the agent has to be added to all rewards of a

[11]For a quantification of this statement, see for example [Pratt, 1964].

lottery. For example, if the agent has acquired wealth -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 negative the wealth of an agent is, 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.

Since the optimal action for a state depends on the wealth of the agent, the Markov property does not hold and dynamic programming methods cannot be used on the original probabilistic decision graph without taking the "wealth" of the agent into account, which makes planning very inefficient. This problem exists only for risk-sensitive planning, but not for risk-neutral planning.

One can circumvent this problem with planning methods that have a limited look-ahead, say n. 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 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. Since these and related heuristic planning methods rely on hill-climbing, they suffer from the limited horizon problem and their success depends critically on the structure of the planning task.

5 A Solution

In the following, we will outline one possible way for incorporating risk-sensitive attitudes into arbitrary planning methods. This is done by 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" [12] [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. 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 over being risk-neutral to being strongly risk-seeking. The larger γ, the more risk-seeking the agent is, and vice versa.

[12] "Constant local risk aversion" means that the quantity $\frac{u''(c)}{u'(c)}$, which is called "local risk aversion," does not depend on the reward c.

These utility functions are exactly the ones that 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 wealth of the agent to all (goal) rewards of every lottery. This increases the certainty equivalent of every lottery by the wealth of the agent, since the utility function has the delta property. Thus, when comparing lotteries, one can ignore the wealth of the agent.

Howard and Matheson [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 ("behaving in the world") if the agent is risk-sensitive towards variations of the reward that it receives per action execution.[13] Unfortunately, our goal-oriented planning task PT2 does not possess the properties required by Howard and Matheson, and thus we either cannot proceed in exactly the same way or have to confirm that their theory is still applicable.

5.1 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 $\gamma > 1$.[14]

First, we would like to point out that every planning task PT2 can be transformed into an equivalent planning task that has only one goal state, which has goal reward zero. The argument, however, is different from the one for the risk-neutral case. Planning task PT2 is equivalent to the following planning task PT2' for a risk-seeking agent:

> Introduce a new goal state with goal reward zero, and replace every (old) goal state $s \in G$ with a non-goal state in which only one action

[13]Assume that a lottery with expected reward c is repeated n times and let c_i denote the reward received on the ith trial. The law of large numbers shows that, as n increases, $\sum_{i=1}^{n} c_i$ approaches nc and the variance of $\sum_{i=1}^{n} c_i$ approaches zero. Thus, if an agent can participate in a lottery a large number of times, it can evaluate the lottery according to c no matter what its attitude towards risk is *provided that it is only interested in its final wealth*. This might not be the case, however. As an example, consider an agent that can choose every month one lottery from a given set of lotteries that supply it with food for the month. The agent cannot stock unused food, since it spoils within thirty days. Such an agent is certainly interested in the expected amount of food provided by a lottery. However, instead of paying attention to the variance of $\sum_{i=1}^{n} c_i$ it will probably be more interested in the variance of c_i, which does not vary with n.

[14]We will show the derivations for $u(c) = \gamma^c$. They can easily be extended to utility functions of the form $u(c) = m\gamma^c + n$ (for $m > 0$), although these functions do not increase the power of the planning mechanism, since utility functions are defined only up to strictly positively linear transformations.

can be executed, that leads with probability one and action cost $r[s] - \max_{s' \in G} r[s'] - 1 < 0$ to the new goal state. Otherwise, the planning problem remains unchanged.

Planning task PT2' satisfies all requirements of our planning framework. It is equivalent to planning task PT2 in the following sense: the better a plan is for planning task PT2, the better it is for planning task PT2' as well, and vice versa. This is the case, because the certainty equivalent of every plan for planning task PT2' is $\max_{s' \in G} r[s'] + 1$ lower than the one of the same plan for planning task PT2. Note that this argument depends on the utility function having the delta property — it does *not* hold for arbitrary utility functions.

In the following sections, we will 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. Finally, we will demonstrate our ideas on a small blocks-world example and a path-planning example.

5.1.1 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:[15]

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

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

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

In matrix notation, we get the closed-form solution

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

where id is the identity matrix of size $|S \setminus G|^2$. (The matrix inverse in the expression always exists.)

For γ approaching one, the certainty equivalent of any state for any plan approaches the expected total reward of that state. Therefore, the optimal plan for a risk-neutral agent is the one that is best for a risk-seeking agent if γ approaches one.

Proof: Assume that the execution of the plan leads with probability p_i to total reward c_i if the agent is started in state s. Then, the expected total reward of s equals $\sum_i p_i c_i$ and its certainty equivalent is $u^{-1}(\sum_i p_i u(c_i))$. Thus, $\lim_{\gamma \to 1} u^{-1}(\sum_i p_i u(c_i)) = \lim_{\gamma \to 1} \log_\gamma(\sum_i p_i \gamma^{c_i}) = \lim_{\gamma \to 1} \frac{\ln(\sum_i p_i \gamma^{c_i})}{\ln \gamma} \overset{L'H\hat{o}pital}{=} \lim_{\gamma \to 1} \frac{(\sum_i p_i c_i \gamma^{c_i - 1}) / \sum_i p_i \gamma^{c_i}}{1/\gamma} = \lim_{\gamma \to 1} \frac{\sum_i p_i c_i \gamma^{c_i}}{\sum_i p_i \gamma^{c_i}} = \frac{\lim_{\gamma \to 1} \sum_i p_i c_i \gamma^{c_i}}{\lim_{\gamma \to 1} \sum_i p_i \gamma^{c_i}} = \frac{\sum_i p_i c_i}{1} = \sum_i p_i c_i$.

In contrast, for γ approaching infinity, the certainty equivalent of any state for any plan approaches the total reward of that state if nature acts like a friend (that is, chooses the action outcomes not with a coin flip, but deliberately so that it is best for the agent). Of course, in reality, nature does flip coins. We call an agent that assumes (wrongly) that nature helps it as much as it can and calculates it total utilities accordingly "extremely risk-seeking." Thus, max-max-ing ("both the agent and nature maximize the total reward for the agent"), which calculates the total utility of a non-goal state s for a given plan as $u[s] = \max_{s' \in S}(c^{a[s]}[s,s'] + u[s'])$, determines the plan that is best for a risk-seeking agent if γ approaches infinity.

Proof: Assume again that the execution of the plan leads with probability p_i to total reward c_i if the agent is started in state s. Then, $\max_i c_i = \log_\gamma \gamma^{\max_i c_i} = \log_\gamma \sum_i p_i \gamma^{\max_i c_i} \geq \log_\gamma \sum_i p_i \gamma^{c_i} \geq \max_i \log_\gamma(p_i \gamma^{c_i}) = \max_i(\log_\gamma p_i + c_i)$. It follows that $\max_i c_i = \lim_{\gamma \to \infty} \max_i c_i \geq \lim_{\gamma \to \infty} \log_\gamma \sum_i p_i \gamma^{c_i} \geq \lim_{\gamma \to \infty} \max_i(\log_\gamma p_i + c_i) = \max_i c_i$, and thus $\lim_{\gamma \to \infty} u^{-1}(\sum_i p_i u(c_i)) = \lim_{\gamma \to \infty} \log_\gamma \sum_i p_i \gamma^{c_i} = \max_i c_i$.

5.1.2 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 $\bar{u}[s] = \bar{v}[s]$. 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 state $\bar{s}$ with expected total reward zero. (State $\bar{s}$ could for example be modeled as a "sink": a non-goal state in which only one action can be executed. This action has action cost zero and leads back to state $\bar{s}$ with probability one. State $\bar{s}$ could also be modeled as a goal state 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 $s \neq \bar{s}$, 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 6 is shown in Figure 11. (State $\bar{s}$ is modeled as a non-goal state.) Note that, although they can both

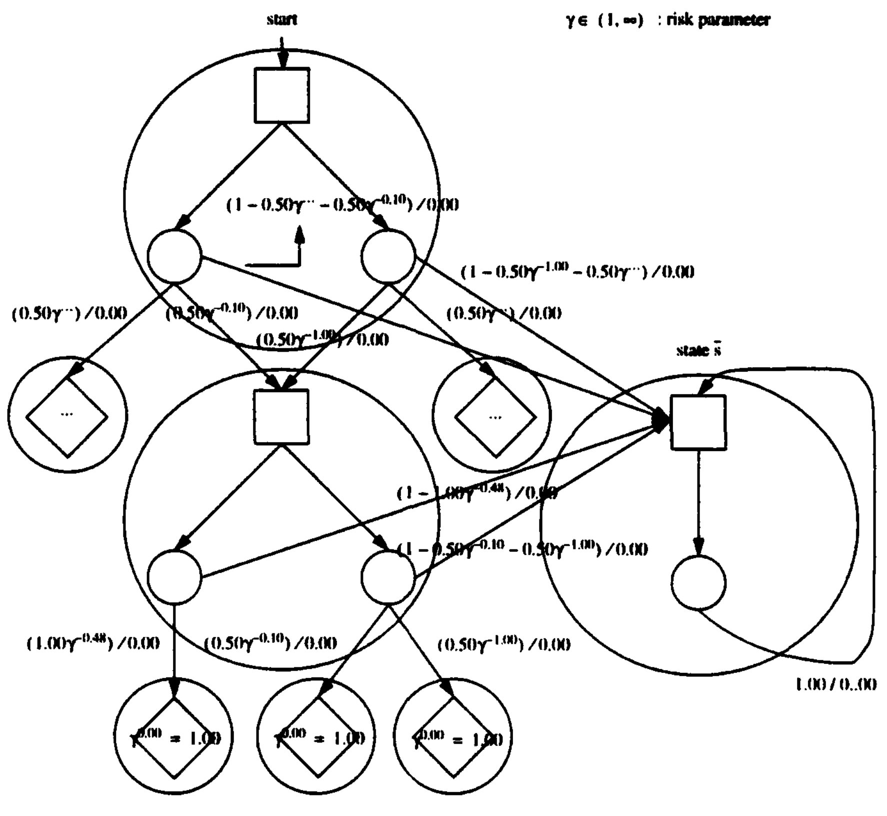


Figure 11: Transformation II (does work)

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.

5.1.3 Finding Optimal Plans

Planning task PT3 can be solved with probabilistic AI planning methods or, alternatively, with dynamic programming methods, called Markov decision algorithms.

It is interesting to note that the plan with the largest expected total utility is not necessarily admissible even if an admissible plan exists, as shown in Figure 12 for a risk-seeking agent with utility function $u(c) = 2^c$. If the agent chooses action A, then the expected total utility of the plan is $0.50u(-\infty) + 0.50u(-1) = 0.25$, but the plan is not admissible. If the agent chooses action B, then it achieves a total (expected) utility of $1.00u(-3) = 0.125$ and reaches the goal state for sure. Thus, the inadmissible plan results in a larger expected total utility. This cannot happen for risk-neutral agents, since the optimal risk-neutral plan is always admissible (if an admissible plan exists).

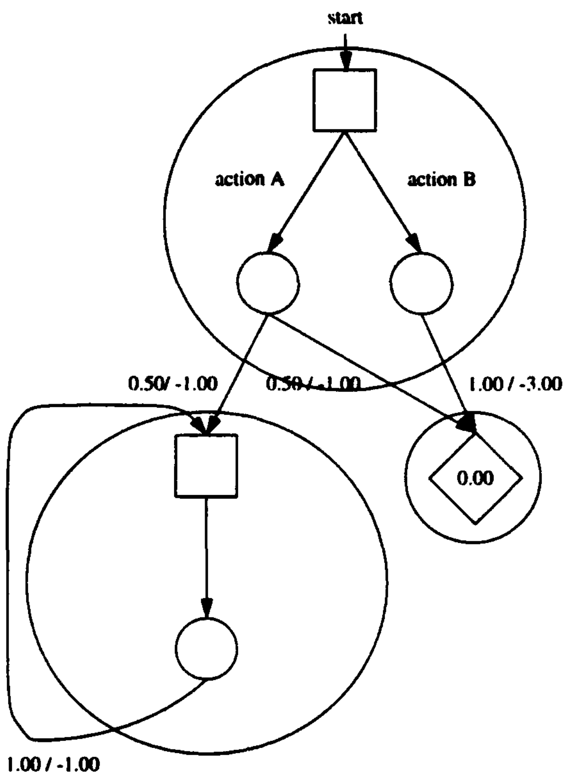


Figure 12: A Planning Problem with an Inadmissible Optimal Plan

The reason why the plan with the largest expected total utility is no longer guaranteed to be admissible for risk-seeking agents is shown in Figure 13: While for risk-neutral agents all inadmissible plans have certainty equivalent minus infinity, this is no longer true for risk-seeking agents. The table also shows that this "problem" cannot arise for risk-averse agents.

It should be pointed out that, from the standpoint of utility theory, there is absolutely no problem with optimal plans that are inadmissible. In the AAAI robot competition example, for instance, one might risk that the robot will not be able to finish the competition with a certain probability. However, if one insists on using utility theory only to choose the best plan ***among all admissible plans***, one can utilize that the optimal plan for a risk-seeking agent is guaranteed to be admissible if *every* state is solvable.[16] Thus, if some states are unsolvable, one can easily remove the unsolvable states from the planning problem before solving it [Koenig, 1991]. The optimal plan of the resulting planning problem is then guaranteed to be admissible.

The optimal plan for the transformed planning task PT3 can be determined with

[16]Properties such as this one can easily be proved using the any-time property of Markov decision algorithms such as the policy-iteration algorithm.

dynamic programming algorithms. If the transformed probabilistic decision graph is acyclic, it can be solved in linear time with folding-back. For cyclic probabilistic decision graphs, it can be formulated as Markov decision problem, that can then be solved in polynomial time with Markov decision algorithms. The representation as Markov decision problem proves that one can indeed restrict plans to state-action mappings without losing optimality and, furthermore, that there exists at least one state-action mapping that is optimal for all possible start states.[17] 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) Set

$$\begin{aligned}[u[s]]_{s \in S\setminus G} \;:=\; & (id - [p^{a[s]}[s,s']\gamma^{c^{a[s]}[s,s']}]_{s,s' \in S\setminus G})^{-1} \\ & \cdot [p^{a[s]}[s,s']\gamma^{c^{a[s]}[s,s']}]_{s \in S\setminus G, s' \in G}[\gamma^{r[s]}]_{s \in G}\end{aligned}$$

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 $s \in S \setminus G$

$$\begin{aligned}a[s] \;:=\; & \text{argmax}_{a \in A(s)}(\sum_{s' \in S\setminus G} p^a[s,s']\gamma^{c^a[s,s']}u[s'] \\ & + \sum_{s' \in G} p^a[s,s']\gamma^{c^a[s,s']+r[s']})\end{aligned}$$

5. Go to 2.

This algorithm is an any-time algorithm. The term "any-time algorithm" was coined by [Boddy and Dean, 1989]), and [Bresina and Drummond, 1990] first developed an any-time planner. Any-time 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

[17]See [Bertsekas, 1987] for a good review of Markov decision theory.

any further. The policy-iteration algorithm is an any-time 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.

When implementing a risk-sensitive planning method, however, one will usually only want to approximate optimal plans and therefore use more incremental (and faster) dynamic programming techniques that restrict their attention to certain states without having to look at the whole state space, determine for which states accurate expected total utilities are not that important, and utilize initial heuristic knowledge (see [Dean *et al.*, 1993] and [Moore and Atkeson, 1993b] for approaches in this direction). Dynamic programming methods for risk-sensitive planning can also be extended to cases where the action models are not given in advance, but have to be learned from reinforcement that is provided externally in response to action executions ("experiments") of the agent (so called "reinforcement learning").

However, dynamic programming algorithms are brute-force search algorithms, 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 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.

Instead of planning methods that maximize or satisfice expected total reward, we can also use AI planning methods that maximize or satisfice the probability of goal achievement, but do not take cost considerations into account. First, we have to transform planning task PT2 into the equivalent planning task PT2'. Remember that PT2' is a risk-seeking planning task that has only one goal state, which has goal reward zero. Now consider the risk-neutral planning task PT3 that corresponds to planning task PT2' and assume that state $\bar{s}$ has been modeled as a non-goal state. Since the total utility of the goal state of PT2' is $\gamma^0 = 1$, it has goal reward one for planning task PT3. Because all actions of planning task PT3 have action cost zero, the probability of reaching the goal state of PT3 for any plan equals its expected total reward for the same planning task, which in turn equals its expected total utility for planning task PT2. Thus, planning methods that determine plans that maximize the probability goal achievement (or whose probability of goal achievement exceeds a given probability) can be used to maximize (or satisfice) expected total utility (or, in the limit, expected total reward). Although one cannot apply these planning methods directly, one can first transform the planning problem and then apply the planning methods to the transformed planning problem. Thus, perhaps surprisingly, they *can* be used to take cost considerations into account provided that they are able to deal with planning tasks for which not all states are solvable. In fact, every non-goal state of

agent is ...	certainty equivalent $u^{-1}(u[s_0])$ of ...	
	an admissible plan	an inadmissible plan
risk-neutral	$u^{-1}(u[s_0]) > (-\infty)$	$u^{-1}(u[s_0]) = -\infty$
risk-seeking	$u^{-1}(u[s_0]) > -\infty$	$u^{-1}(u[s_0]) \geq -\infty$
risk-averse	$u^{-1}(u[s_0]) \geq -\infty$	$u^{-1}(u[s_0]) = -\infty$

(utility function has delta property)

Figure 13: Possible Certainty Equivalents of Plans

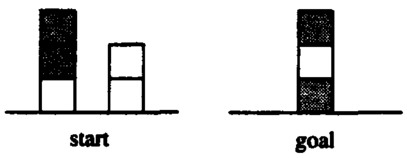
PT3 is unsolvable (even if every non-goal state of PT1 is solvable), since the unsolvable state $\bar{s}$ can be reached from every non-goal state with positive probability. Therefore, finding optimal plans is not solely a matter of extending the number of states that are covered by the state-action mapping (the so-called "plan envelope"), since the agent cannot recover from state $\bar{s}$ and thus has to minimize the probability of reaching this state if it wants to determine an optimal plan.

To summarize, planning task PT3 can be solved with AI planning methods that either maximize (or satisfice) the expected total reward or, equivalently, the probability of goal achievement. Whether state $\bar{s}$ should be modeled as a goal state or non-goal state depends on which one of the two kinds of planning methods one wants to use. If $\bar{s}$ is modeled as a non-goal state, then — as mentioned in the previous paragraph - no plan is admissible for PT3 if the start state is a non-goal state. In this case, it is sensible to rank plans according to their probability of goal achievement. If $\bar{s}$ is modeled as a goal state, then all plans are admissible for PT3 and it makes sense to choose among them according to their expected total reward.

5.1.4 Example 1: A Stochastic Blocks-World

We use the blocks-world problem that is stated in Figure 14 to illustrate our ideas. Its state space contains 162 states. Figure 15 shows four of the state-action mappings that solve it, and Figure 16 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 (calculated for example with the policy-iteration algorithm stated above, which — depending on the initial policy — typically converges after around five iterations).

Plan A, that 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 using plan D, for example, the agent reaches a goal state in only three minutes if it is lucky.



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 14: A Blocks-World Problem

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 16 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.)

For $\ln \gamma$ approaching zero (that is, γ approaching one), the certainty equivalent of any plan approaches its expected total reward. For plan D, for example, the certainty equivalent approaches -21.00. (See Figure 21 to verify this.) Since plan A is optimal for $\ln \gamma$ approaching zero, it is optimal for a risk-neutral agent. In contrast, for $\ln \gamma$ approaching infinity (which is equivalent to γ approaching infinity), the certainty equivalent of any plan approaches its total reward for the case that nature acts as a friend. When executing plan D, for example, the agent can reach a goal state in only three minutes if it is lucky. Thus, the certainty equivalent approaches -3.00, and plan D is optimal for an extremely risk-seeking agent. (The certainty equivalent of plan C also approaches -3.00. Thus, it is also optimal for an extremely risk-seeking agent. However, for $\ln \gamma \geq 4.59$ plan D dominates plan C for a risk-seeking agent in that it always has a larger certainty equivalent.)

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

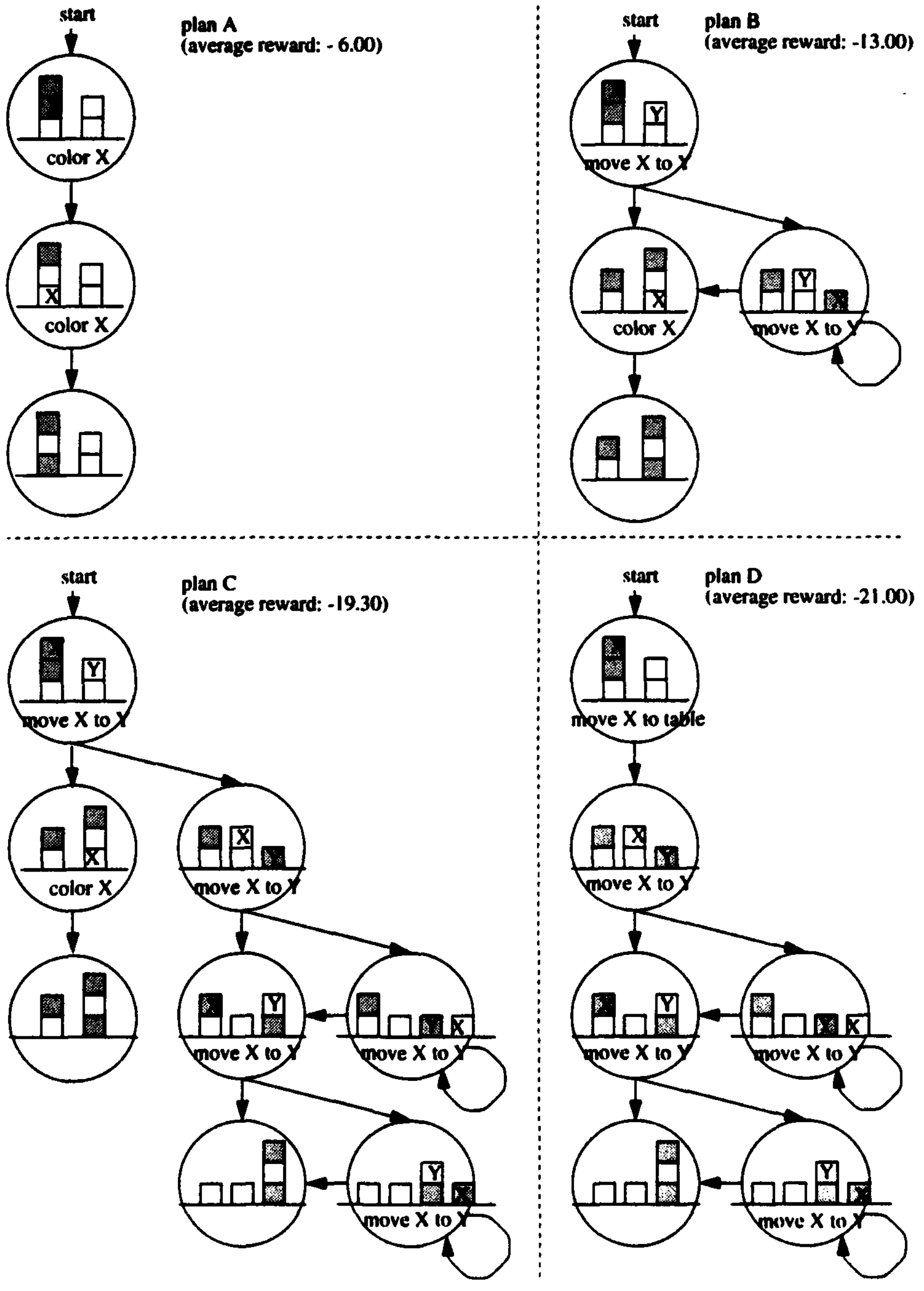


Figure 15: Some Plans for the Blocks-World Problem

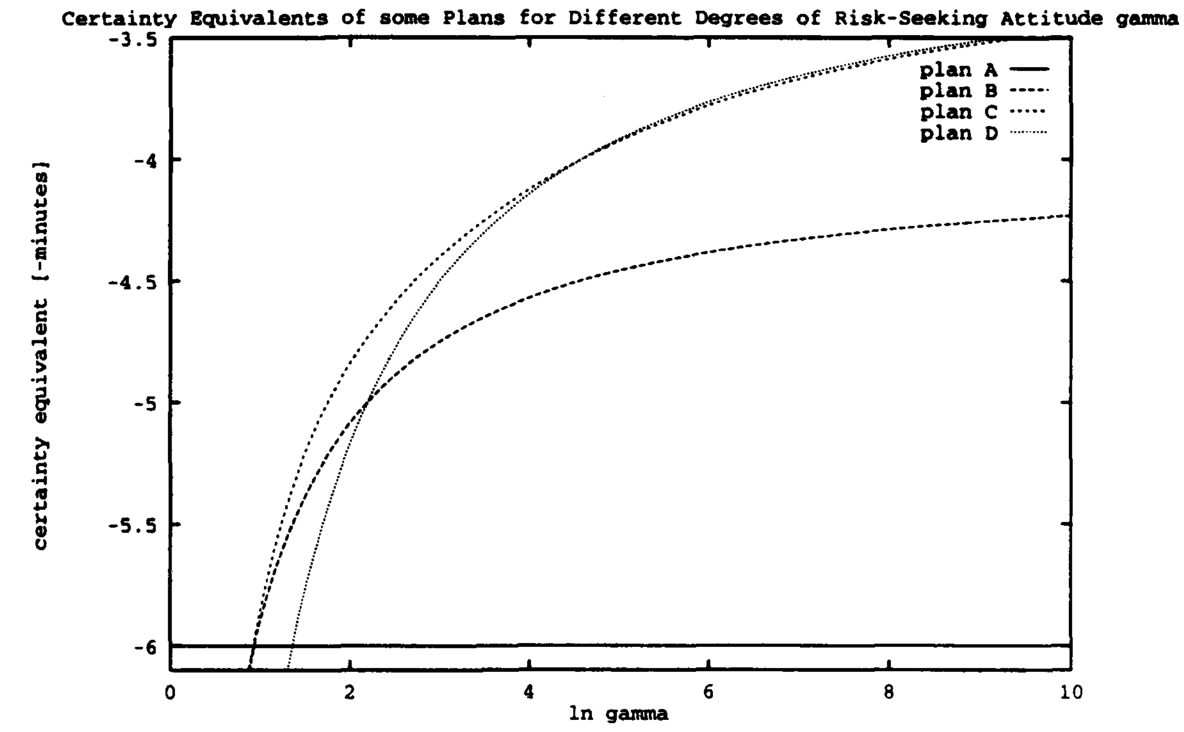


Figure 16: Certainty Equivalents of the Four Blocks-World Plans (Risk-Seeking Agent)

STRIPS-rules[18] [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
    delete: on(X,Y), clear(Z)
    add:    on(X,Z), clear(Y)
  outcome:  /* failure: block X falls onto the table */
    prob:   0.9
    reward: -1
    delete: on(X,Y)
    add:    clear(Y),on(X,table)
```

[18]The original STRIPS-notation [Fikes and Nilsson, 1971] applies to deterministic domains only.

The transformation changes the transition probabilities, action costs, and goal rewards. In accordance with our earlier comments, we do not model the transition into state $\bar{s}$. The goal states remain goal states, but now get assigned a goal reward of one. The STRIPS-rules are transformed as shown in Section 5.1.2. 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
    delete: on(X,Y), clear(Z)
    add:    on(X,Z), clear(Y)
  outcome:  /* failure: block X falls onto the table */
    prob:   0.45
    reward: 0
    delete: on(X,Y)
    add:    clear(Y),on(X,table)
```

With the complementary probability (0.5), the action execution results in the new state $\bar{s}$, that has a total reward of zero. Now, one can use any planning method that maximizes expected total reward or, in this case equivalently, the probability of goal achievement on the transformed STRIPS-rules to determine an optimal plan for the risk-seeking agent. (If the probabilistic planning method maximizes expected total reward, it might be advantageous to model state $\bar{s}$ as a goal state with goal reward zero. If the planner maximizes the probability of goal achievement, state $\bar{s}$ has to be modeled as a non-goal state from which no goal state can ever be reached.)

5.1.5 Example 2: Stochastic Path-Planning

As a second example, consider the simple path-planning domain shown in Figure 17. The states of this grid-world correspond to locations. In every state, the agent has at most four actions available, namely to go up, right, down, or left. All actions take the agent one minute to execute, but they are not necessarily deterministic. They succeed with probability $\frac{1+x}{3-x}$, but their outcome deviates ninety degrees to the left or right of the intended direction with probability $\frac{1-x}{3-x}$ each. Thus, $x \in [0, 1]$ is a parameter for the certainty of the actions: the larger the value of x is, the more certain are their outcomes. Actions have deterministic outcomes if $x = 1$; their intended outcome and its two deviations are equally likely for the other extreme, $x = 0$.

In every state, the agent can execute all of the actions whose intended direction is not immediately blocked by a wall. Besides standard walls, the grid-world also contains "one-way walls," that can be traversed from left to right, but not in the opposite

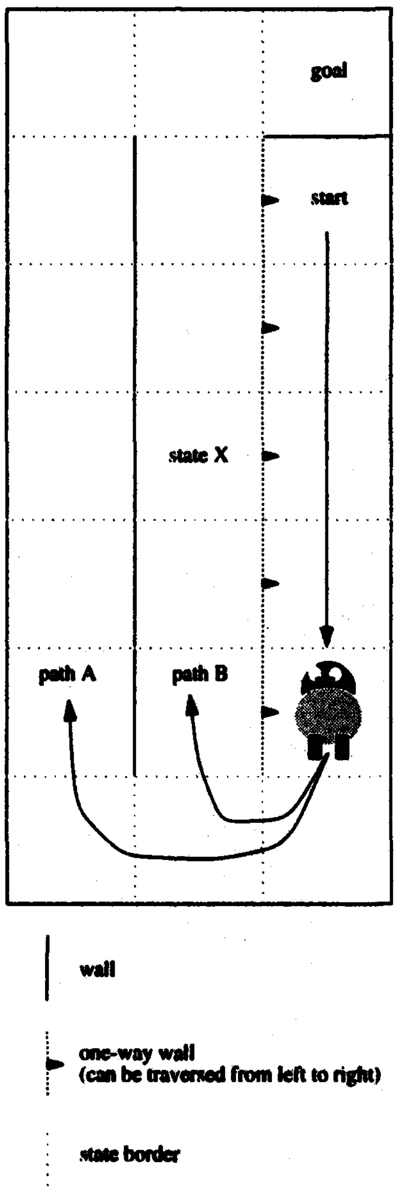


Figure 17: A Path-Planning Problem

direction. (They might for example be steep slopes that the agent can slide down, but not climb up.) If the agent executes an action and it has the intended outcome, the agent cannot bump into a wall. However, running into a wall is possible for unintended outcomes, in which case the agent does not change its location. As an example, consider state X in figure 17 and assume that $x = 0.5$. In this state, the agent can go up, right, or down. If it tries to go up, it will succeed with probability 0.6, unintentionally go right with probability 0.2, and unintentionally remain in state X with the same probability.

The agent can reach the goal state from the start state on two different paths. If the actions have deterministic effects (that is, if $x = 1$), the traversal of path B takes 13 minutes and the one of path A 15 minutes. Thus, the agent prefers path B over path A, independently of its risk-attitude. If the actions do not have deterministic effects, however, the agent risks to traverse a one-way wall unintentionally when following path B, in which case it has to retrace parts of its path. Since path B is better than path A in the best case, we expect path A to become less attractive when the agent becomes more risk-seeking.

Figure 18 shows how the value of the action certainty parameter x that makes the agent indifferent between the two paths varies with the risk-attitude of the agent (expressed as the natural logarithm of risk parameter γ). The figure actually comprises risk-seeking, risk-neutral, and risk-averse behavior. Here, we are only interested in the risk-seeking part: $\ln \gamma > 0$. If, for a given risk-attitude, the actual value of x is below the graph, then the agent chooses path A, otherwise it chooses path B. As expected, the value of x that makes the agent indifferent between the two paths decreases the more risk-seeking the agent becomes. For $\ln \gamma$ approaching infinity, it approaches zero: an extremely risk-seeking agent prefers path B over path A, since path B can be traversed in 13 minutes in the best case, whereas path A cannot be traversed in less than 15 minutes.

5.2 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 5.1.1 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.[19]

[19]See [Koenig and Simmons, 1994] for details. For plans that have cycles with "probability" greater than or equal to one, the dynamic programming equation is no longer a contraction mapping. The case of cycles with "probability" larger than one can be approached similarly to the case of cycles with

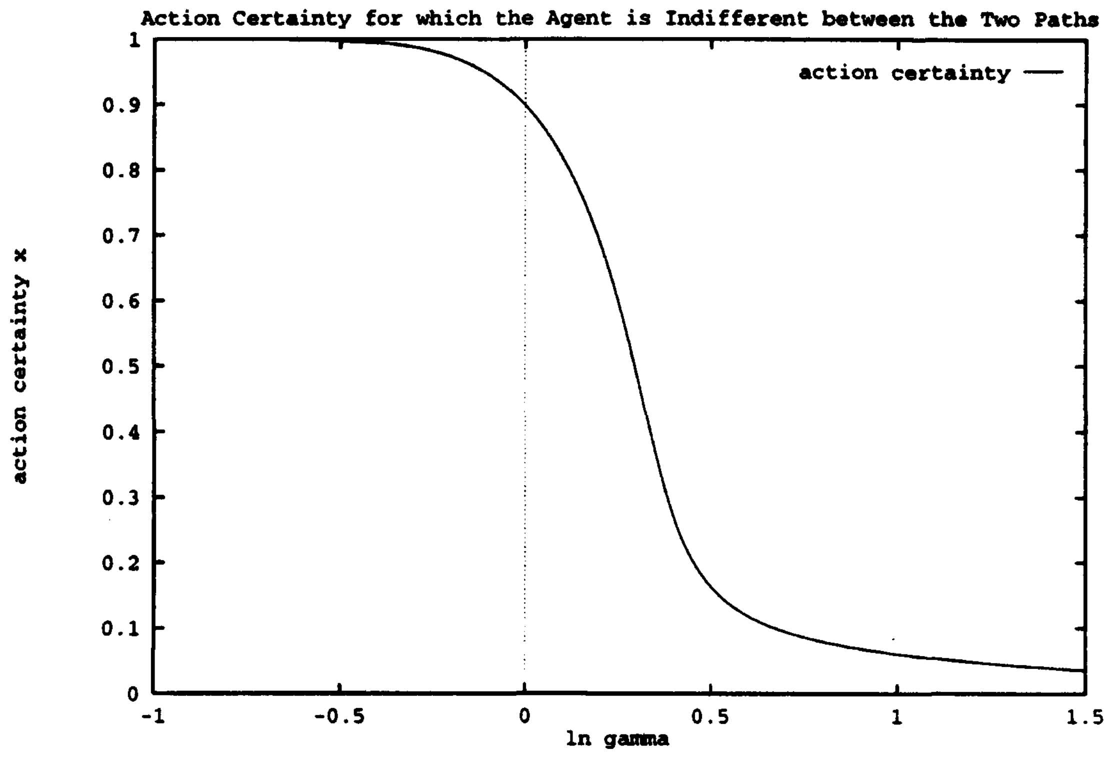


Figure 18: Action Certainty for which the Agent is Indifferent between the Two Paths

For γ approaching zero, the certainty equivalent of any state for any plan approaches the expected total reward of that state if nature acts as an enemy (that is, chooses action outcomes not with a coin flip, but deliberately so that it is worst for the agent). We call an agent that assumes (wrongly) that nature hurts it as much as it can and calculates its total utilities accordingly "extremely risk-averse." (Such an agent believes in Murphy's law: If anything can go wrong, it will.) They have recently been studied in the AI literature by [Heger, 1992] and [Moore and Atkeson, 1993a]. Thus, max-min-ing ("the agent maximizes and nature minimizes the total reward of the agent," in game-theoretical contexts also called min-max-ing), which calculates the total utility of a non-goal state s for a given plan as $u[s] = \min_{s' \in S}(c^{a[s]}[s, s'] + u[s'])$, determines the plan that is best for a risk-averse agent if γ approaches zero. (For γ approaching one, the certainty equivalent of any state for any plan approaches — as in the risk-seeking case — the expected total reward of that state.)

If there are cycles in probabilistic decision graphs, then — unfortunately — the ex-

probability one, which — as is well known in the field of Markov decision theory — can also be solved by either initializing the dynamic programming algorithms more restrictedly [Koenig and Simmons, 1994] or extending them slightly, see for example the multiple-chain policy-iteration algorithm [Howard, 1964].

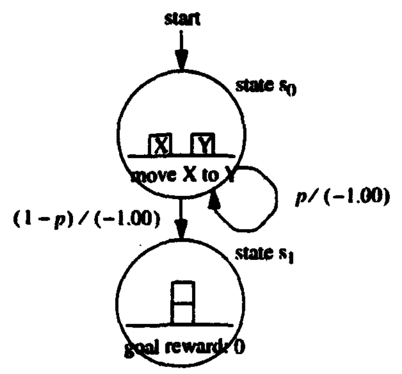


Figure 19: A Plan for Stacking Two Blocks

pected total utilities of admissible plans (and thus their certainty equivalents) can be minus infinity, see also Figure 13. 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 ("it is scared") 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. Figure 14 shows that this problem can not arise for risk-neutral or risk-seeking agents.

Consider for example the plan shown in Figure 19 that solves the problem of stacking two blocks with a move action that fails with probability p. The total reward of state s_0 is $-i-1$ with probability $p^i(1-p)$ (for all integers $i \geq 0$). Thus

$$\begin{aligned} u[s_0] &= \sum_{i=0}^{\infty} p^i(1-p)u(-i-1) \\ &= \sum_{i=0}^{\infty} p^i(1-p)(-\gamma^{-i-1}) \\ &= \frac{p-1}{\gamma}\sum_{i=0}^{\infty}\left(\frac{p}{\gamma}\right)^i \\ &= \begin{cases} \frac{1-p}{p-\gamma} & \text{for } p < \gamma \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

How the certainty equivalent of the plan depends on p is shown in Figure 20. The expected total utility of the plan is minus infinity for $p \geq \gamma$. If the agent can choose between a move action with action failure probability p_1 and a different move action with action failure probability p_2 where $p_1 > p_2 > \gamma$, it cannot decide which one to prefer although it should clearly choose the later move action over the former one.

The example also demonstrates that plans can have fixed points that do not correspond

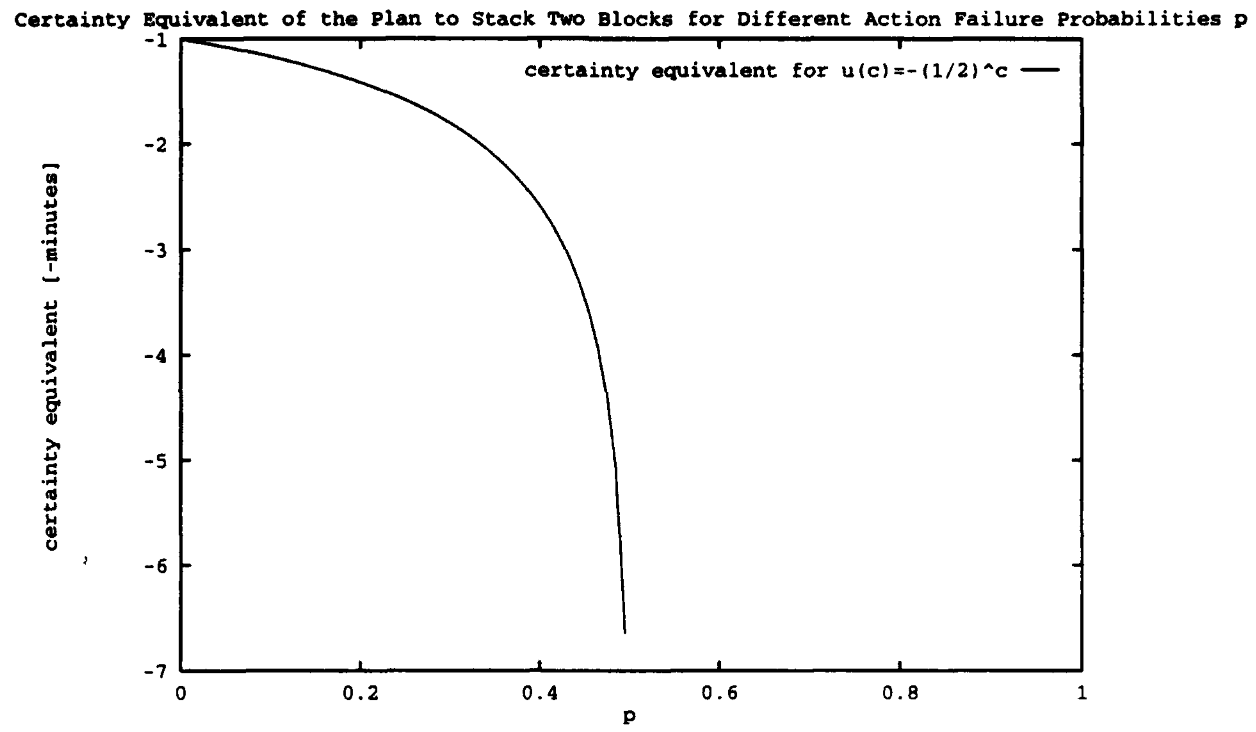


Figure 20: Certainty Equivalent of the Plan for Stacking Two Blocks

to their expected total utility. The following system of linear equations is the one used in Section 5.1.1 to calculate the expected total utilities of the states:

$$\begin{aligned} u[s_0] &= p\gamma^{-1}u[s_0] + (1-p)\gamma^{-1}u[s_1] \\ u[s_1] &= -\gamma^0 \end{aligned}$$

Its solutions are

$$\begin{aligned} u[s_0] &= \frac{1-p}{p-\gamma} \\ u[s_1] &= -1 \end{aligned}$$

This shows that, fot $p > \gamma$, the expected total utility of the plan differs from the value of $u[s_0]$.

Finally, consider again the example domains from Sections 5.1.4 and 5.1.5. Figure 21 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.

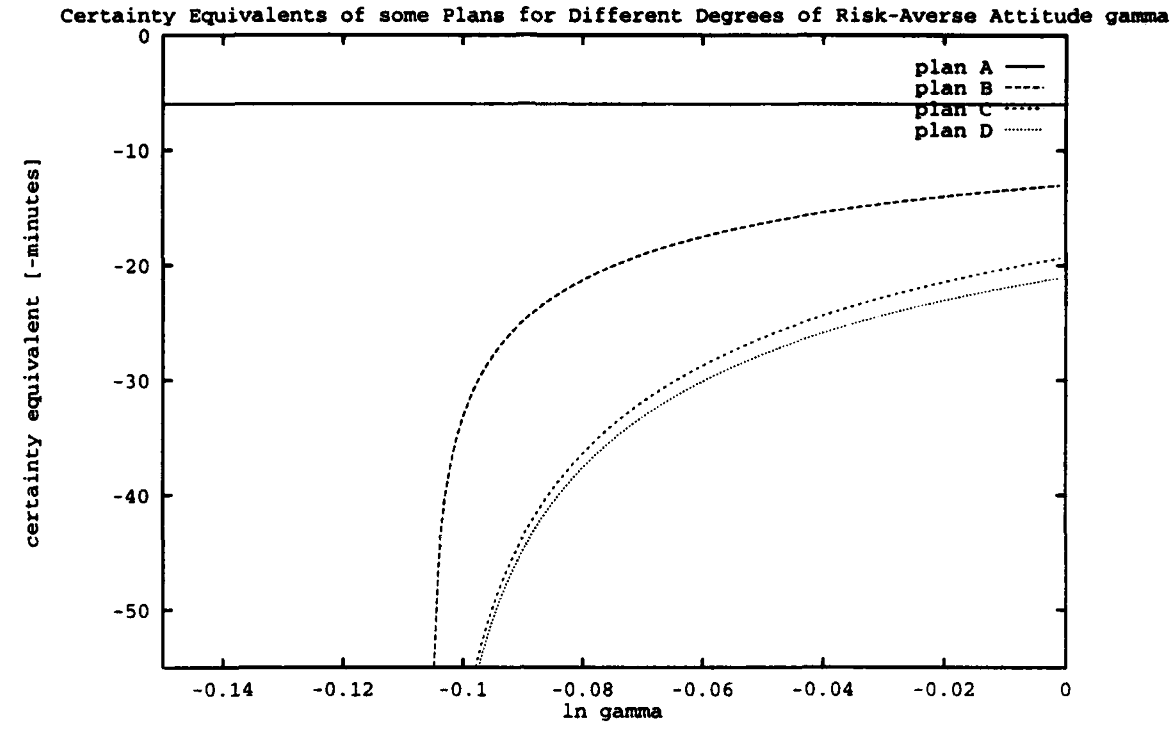
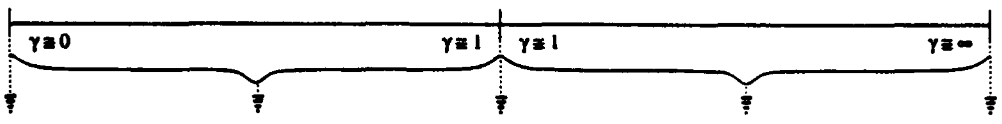


Figure 21: Certainty Equivalents of the Four Blocks-World Plans (Risk-Averse Agent)

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 $\ln \gamma$. For the path-planning domain, Figure 18 already contains the results for negative attitudes towards risk, that is, for $\ln \gamma < 0$.

6 Conclusion

In this paper, we were concerned with probabilistic planning for *risk-sensitive* agents, since there are many situations where it is not optimal to determine plans that minimize expected total execution cost or maximize the probability of goal achievement. We used acyclic and cyclic probabilistic decision graphs as STRIPS-like planning framework and utilized utility functions that possess the delta property. These utility functions cover a whole spectrum of risk-sensitive attitudes from being strongly risk-averse over being risk-neutral to being strongly risk-seeking. They fill a gap between approaches previously studied in the AI literature, namely the approach to minimize expected total execution cost (risk-neutral attitude) and the approach to assume that nature acts like



risk attitude	extremely risk-averse	risk-averse	risk-neutral	risk-seeking	extremely risk-seeking
formalization	max-min-ing	utility theory	utility theory	utility theory	max-max-ing
utility of lottery	$u(L) = \min_i u(c_i)$	$u(L) = \sum_i p_i \cdot u(c_i)$	$u(L) = \sum_i p_i \cdot u(c_i)$	$u(L) = \sum_i p_i \cdot u(c_i)$	$u(L) = \max_i u(c_i)$
utility function		$u(c) = -m \cdot \gamma^c + n$ $(m > 0, 0 < \gamma < 1)$	$u(c) = m \cdot c + n$ $(m > 0)$	$u(c) = m \cdot \gamma^c + n$ $(m > 0, \gamma > 1)$	

(L is a lottery: prize c_i is won with probability p_i for all i)

Figure 22: Continuum of Risk-Sensitive Behavior

a friend (extremely risk-seeking attitude) or enemy (extremely risk-averse attitude), all of which they can asymptotically approximate as shown in Figure 22. We obtained the following results:

While it is indeed true that — in order to incorporate risk-seeking attitudes into planning methods — one only has to change the planning problem and can leave the planning method untouched, it is not enough to replace all costs and rewards with their respective utilities.

For acyclic probabilistic decision graphs, one could first propagate all action costs to the value nodes, replace all accumulated costs with their respective utilities, and then use folding-back to determine the optimal decisions for the decision nodes. This works for all utility functions, although one might be forced to duplicate all shared subtrees, which makes the run-time in the worst case exponential in the size of the probabilistic decision graph. However, if the utility functions possess the delta property, one can easily 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. (For very risk-averse agents and some cyclic probabilistic decision graphs, however, the agent can become so "scared" of the risk that not reaching a goal state becomes as preferable as reaching a goal state even if the planning problem is solvable.)

Figure 23 summarizes the worst case run-times of the algorithms that we have discussed in this paper. Although we primarily discussed dynamic programming algorithms, our approach to risk-sensitive planning can be used to augment other risk-neutral probabilistic planning algorithms as well. After the planning problem has been transformed, one can use probabilistic AI planning methods or, alternatively, dynamic programming

generality			the agent is ...		
		utility function	risk-neutral	risk-seeking	risk-averse
↓	cost-annotated decision trees	in general	(has delta property)	linear	linear
		delta property	linear	linear	linear
	acyclic decision graphs	in general	(has delta property)	exponential	exponential
		delta property	linear	linear	linear
	decision graphs	in general	(has delta property)	(not discussed)	(not discussed)
		delta property	polynomial	polynomial	polynomial (but not always applicable)

Figure 23: Worst Case Run-Times

methods on the transformed planning problem to determine optimal (or good) plans for the original, risk-sensitive planning problem. For risk-seeking agents, one can use *any* planning method on the transformed planning problem that maximizes (or satisfices) expected total reward (for example, the planner of [Smith, 1988]) or, equivalently, the probability of goal achievement (for example, the planner of [Kushmerick *et al.*, 1993]) to determine an optimal (or satisficing) plan. The better a plan is for the transformed planning problem, the better it is for the original planning problem as well. Since our approach allows one to use existing probabilistic planners unchanged for risk-seeking planning and — depending on the planner — maybe for risk-averse planning as well, it extends their functionality. 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.

7 Acknowledgements

Thanks to (in alphabetical order) Justin Boyan, Lonnie Chrisman, Matthias Heger, Andrew Moore, Joseph O'Sullivan, Stuart Russell, Jiří Sgall, and Michael Wellman for helpful discussions or comments. Matthias Heger also gave me a chance to present some of my work at the University of Bremen (Germany).

References

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

(Bertsekas, 1987) Bertsekas, D.P. 1987. *Dynamic Programming, Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs (New Jersey).

(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.

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

(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, 1992) Goodwin, R. 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, 1992) Heger, M. 1992. Risikoloses Reinforcement-Lernen. *KI* 4:26–32.

(Horvitz, 1988) Horvitz, E.J. 1988. Reasoning under varying and uncertain resource constraints. In *Proceedings of the AAAI.* 111–116.

(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 (Massachusetts), 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, 1994) Koenig, S. and Simmons, R.G. 1994. Risk-sensitive game-playing, any-time planning, and reinforcement learning. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania). (tas).

(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 (California). (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 (Washington).

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

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

(Moore and Atkeson, 1993b) Moore, A.W. and Atkeson, C.G. 1993b. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning* 13(1):103–130.

(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 (California).

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

(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 (California).

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

(Witsenhausen, 1966) Witsenhausen, H.S. 1966. *Minimax Control of Uncertain Systems.* Ph.D. Dissertation, Department of Electrical Engineering, MIT, Cambridge (Massachusetts).