Sensor Planning with Non-Linear Utility Functions

Sven Koenig and Yaxin Liu

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA
{skoenig,yxliu}@cc.gatech.edu

Abstract. Sensor planning is concerned with when to sense and what to sense. We study sensor planning in the context of planning objectives that trade-off between minimizing the worst-case, expected, and best-case plan-execution costs. Sensor planning with these planning objectives is interesting because they are realistic and the frequency of sensing changes with the planning objective: more pessimistic decision makers tend to sense more frequently. We perform sensor planning by combining one of our techniques for planning with non-linear utility functions with an existing sensor-planning method. The resulting sensor-planning method is not only as easy to implement as the sensor-planning method that it extends but also (almost) as efficient. We demonstrate empirically how sensor plans change as the planning objective changes, using a common testbed for sensor planning.

1 Introduction

Sensor planning [4, 1, 11, 12, 19, 10] is concerned with when to sense and what to sense. Sensor planning problems are interesting because sensing provides information that can reduce the plan-execution costs but comes at a cost itself. Although sensor planning can be used in conjunction with different planning objectives, sensor planners traditionally minimize the expected plan-execution costs. However, decision makers often trade-off between minimizing the worst-case, expected, and best-case plan-execution costs. Some decision makers are even so pessimistic (conservative) that they prefer plans with minimal worst-case plan-execution costs, a planning objective often used in robotics [13]. Other decision makers are more optimistic. This is often necessary even for pessimistic decision makers because often the worst-case plan-execution costs are infinite for all plans. Then, pessimistic decision makers need to adopt a more optimistic planning objective than minimizing the worst-case plan-execution costs but still prefer a more pessimistic planning objective than minimizing the expected plan-execution costs. Planners should be able to adopt this planning objective and similar planning objectives. Studying sensor-planning in the context of these planning objectives is interesting not only because they are realistic but also because the frequency of sensing changes with the planning objective. Sensing often provides additional information that can be used to avoid catastrophes, which is important for pessimistic decision makers. Consequently, more pessimistic decision makers tend to sense more frequently.

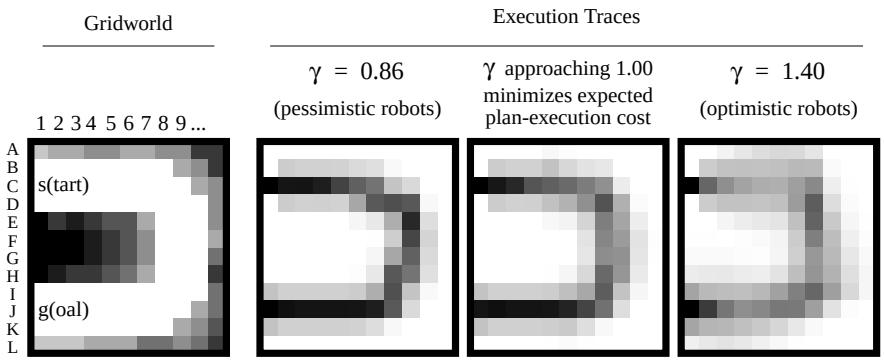


Fig. 1. Gridworld and Execution Traces

In this paper, we describe how one can generalize an existing sensor-planning method by Hansen [5] that minimizes the expected plan-execution costs to these more realistic planning objectives, thereby extending its applicability while maintaining its efficiency. We achieve this by combining one of our techniques for planning with nonlinear utility functions [8] with this efficient sensor-planning method, demonstrating the ease with which our technique can be combined with existing planning methods that minimize the expected plan-execution costs. The resulting sensor-planning method is not only as easy to implement as the sensor-planning method that it extends but is also almost as efficient. To gain insight into how sensor plans change as the planning objective changes, we use a simple robot-navigation domain that is easy to visualize. In particular, we study robot-navigation problems with actuator uncertainty. The robot can always decide to sense its current location, which is costly but prevents it from deviating from the nominal path into muddy terrain that is costly to traverse. Problems of this kind have been studied before [5], traditionally with the planning objective of minimizing the expected plan-execution costs, and can therefore be considered good test problems for new sensor-planning methods. We demonstrate experimentally how the sensor plans change for these problems as the planning objective changes.

2 The Sensor-Planning Problem

We consider robot-navigation problems with the following properties [5]. The robot has to navigate from a given start location to a given goal location in a known environment. Since motion is noisy, the robot can deviate from the nominal path but it can always opt to sense its current location. Sensing provides certainty about its current location but is costly (for example, consumes energy). We assume that there is a finite set of locations L. The robot knows that it starts at location $l_{start} \in L$ and its task is to navigate to location $l_{goal} \in L$ and be sure that it stops at exactly that location. There is a finite set M of movement actions, all of which can be executed at all locations. Motion uncertainty is modeled with conditional probability distributions. Executing movement action $m \in M$ results with cost $c(l, m) > 0$ and

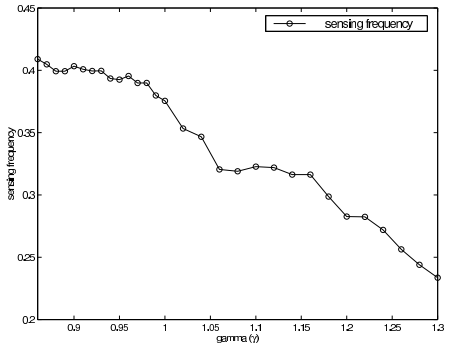
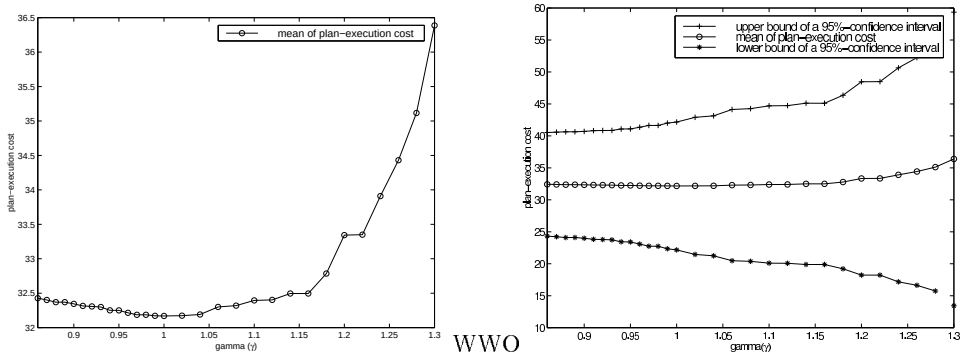


Fig. 2. Sensing Frequency



WWO

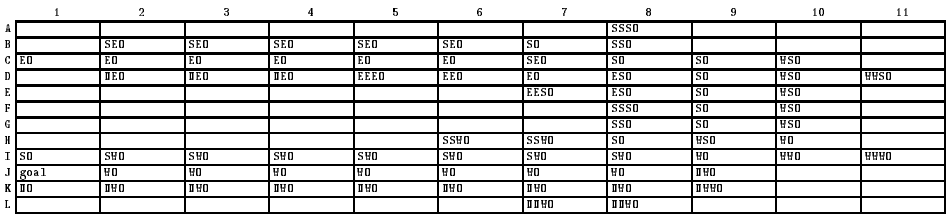
Fig. 3. Mean of Plan-Execution Costs (left) and Mean with Confidence Interval (right)

probability $p(l'|l, m)$ in location l'. The robot receives no feedback as to what its new location is (which makes the simplifying assumption that the costs of the executed actions cannot be observed directly) but there is one sensing action o that can be executed at all locations. Executing it incurs cost $c(l, o) > 0$ and reports the current location of the robot with certainty. We assume that it is possible to reach every location from every other location.

3 Example

We illustrate the sensor-planning problem using simple gridworlds, similar to those used on real robots [18]. Figure 1 (left) shows the gridworld that we use in our experiments, where the locations are squares. The start location is C1 and the goal location is J1. The robot can always sense its current location (O) or move north (N), east (E), south (S), or west (W) to an adjacent square. If the robot attempts to move in a certain direction (say, move east in square C1), then it either moves as intended (C2, with probability 0.6) or strays off by one square to the left (B2, with probability 0.2) or right (D2, with probability 0.2) due to actuator noise and not facing precisely in the right direction. The robot does not move when it bumps

$\gamma = 0.86$ (pessimistic robots)



$\gamma = 1.40$ (optimistic robots)

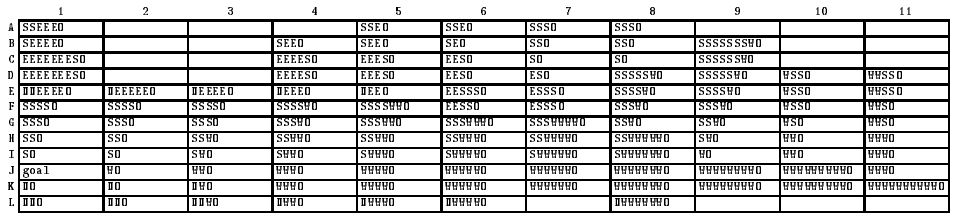


Fig. 4. Optimal Sensor Plans

into the border of the gridworld. The movement costs range from 1.0 to 10.0. They are low for roads (white) and high for muddy terrain (darker colors). The sensing costs are always 0.2. We express the trade-off between minimizing the worst-case, expected, and best-case plan-execution costs using a parameter γ with $0 < \gamma < \infty$. As γ approaches infinity, the robots become more optimistic and thus more interested in plans with small best-case plan-executions costs. As γ approaches one, the robots become more interested in plans with small expected plan-execution costs. Finally, as γ approaches zero, the robots become more pessimistic and thus more interested in plans with small worst-case plan-execution costs. Notice that pessimistic robots have to trade off between minimizing the worst-case and expected plan-execution costs in our example domain. It is not possible to minimize the worst-case plan-execution costs directly because all plans cycle with some probability and thus have worst-case plan-execution costs that are infinite. Figure 2 shows that the sensing frequency (that is, the percentage of sensing actions among all executed actions) increases as the robots become more pessimistic and γ decreases. This is also illustrated in Figure 4, that shows optimal sensor plans for two different values of γ.[1] The sensor plans are

[1] There are some exceptions to this trend, for example, in the vicinity of the goal. This can be explained as follows: Optimistic robots assume that short action sequences that have a chance of reaching the goal location will indeed reach it. Thus, they execute these action sequences followed by a sensing action to confirm that they have reached the goal location. For example, for $\gamma = 1.40$, the action sequence of location I2 is SO. The robot hopes that it will drift to the goal location J1 as it moves south, although this is less likely than moving to location J2. More pessimistic robots are more cautious and execute longer action sequences. For example, for $\gamma = 1.86$, the action sequence of location I2 is SWO, which reaches the goal location with higher probability than the action sequence SO. This phenomenon and similar phenomena contribute to the small local minima in the graph of Figure 2.

depicted as gridworlds, each location of which is annotated with an action sequence. These action sequences are used as follows: After the robots have executed a sensing action, they look up the action sequence that corresponds to the sensed location, execute it, and repeat the process, until they sense that they are at the goal location. For example, for $\gamma = 0.86$, the action sequence of location B2 is SEO. Consequently, after the robot has sensed that it is at location B2, it first moves south (S), then moves east (E), and finally senses again (O). Locations whose action sequences are not used for getting from the start location (C1) to the goal location (J1) are left blank. Figure 1 (right) shows how often the robots visit each grid square during two million runs for three different values of γ. Darker colors indicate a larger number of visits. Thus, more pessimistic robots are more likely to stay on the road and close to the nominal path, which is possible due to the increased sensing frequency. That more pessimistic robots are more likely to stay on the road can be explained as follows: By staying on the road, the robots are likely able to avoid the large costs necessary for getting out of the mud, which pessimistic robots consider to be important. On the other hand, smaller sensing frequencies and attempts to cut the corners decrease the probability that the robots stay on the road but also decrease the plan-execution costs in the best case, which optimistic robots consider to be important. (In fact, the action sequences of the start location get longer and longer as the robots become more optimistic until the action sequences are able to move the robots to the goal location if they are lucky.) This explanation suggests that there is a mean-variance trade-off in our example domain. Mean-variance trade-offs are often used as crude but easy-to-understand explanations for trade-offs between minimizing the worst-case, expected, and best-case plan-execution costs. For example, the graph in Figure 3 (left) shows the mean of the plan-execution costs, and the difference of the upper and lower graphs in Figure 3 (right) corresponds to four times the standard deviation of the plan-execution costs. The mean of the plan-execution costs increases but the variance decreases as the robots become more pessimistic and γ decreases from one to zero. The variance decreases because more pessimistic robots stay on the road and close to the nominal path. The mean-variance trade-off can be explained as follows: More pessimistic robots are willing to accept a larger mean of the plan-execution costs for a decrease in variance because they expect to be unlucky and fear for the worst case. A small variance avoids plan-execution costs that are much larger than average. Figure 3 (right) illustrates this using the upper bound of a 95-percent-confidence interval (that is, mean plus twice the standard deviation) as an approximation of the worst-case plan-execution costs. The upper bound indeed decreases as the robots become more pessimistic since the decrease of the variance outweighs the increase of the mean. On the other hand, more optimistic robots are willing to accept a larger mean for an increase in variance. They expect to be lucky and hope for the best case since a larger variance promises a chance to realize plan-execution costs that are much smaller than the expected plan-execution costs. Figure 3 (right) illustrates this using the lower bound of a 95-percent-confidence interval (that is, mean minus twice the standard deviation) as an approximation of the best-case plan-execution costs. The lower bound indeed decreases as the robots become more optimistic since the increase of the variance outweighs the increase of the mean. The optimal sensor plans can be calculated efficiently with our sensor-planning

method that we discuss in the following. For $\gamma = 0.86$, our sensor-planning method expands 2,071 nodes and needs an average of 4.6 milliseconds per node expansion on a Sun Ultra 1 running Solaris 7. The original sensor-planning method by Hansen, that our sensor-planning method extends, corresponds to the case where γ approaches one. It needs 2,815 node expansions and 2.0 milliseconds per node expansion. For $\gamma = 1.40$, our sensor-planning method needs 5,808 node expansions and 5.2 milliseconds per node expansion. The number of node expansions depends on the sensing frequency. It increases as the sensing frequency of the optimal plans decreases. Our sensor-planning method has a slight run-time disadvantage per node expansion compared to the original sensor-planning method by Hansen because it has to calculate exponentials and logarithms, and its heuristic search method cannot calculate the heuristic values quite as efficiently as the original sensor-planning method.

4 Formalizing the Sensor-Planning Problem

The robot-navigation problems are special cases of partially observable Markov decision process models that can be translated into goal-directed Markov decision process models. Goal-directed Markov decision process models are often used for probabilistic planning in artificial intelligence [2]. They are totally observable Markov decision process models and consist of a finite set of states, including a start state and a goal state in which execution stops. Each non-goal state has a finite set of actions associated with it that can be executed in that state. Executing an action incurs a given finite cost and results in a successor state that is determined by a given probability distribution that depends only on the action and the state that it is executed in. A goal-directed Markov decision process is a stream of $\langle$state, action, cost$\rangle$ triples. The process is always in exactly one state and makes state transitions at discrete time steps. It starts in the start state. A policy determines which actions to execute. A (stationary, deterministic) policy is a mapping from non-goal states to actions, and the goal-directed Markov decision process always executes the action that the policy assigns to its current state. Execution stops only in goal states, in which case the goal-directed Markov decision process does not incur any further costs. In case of our robot-navigation problems, the states of the goal-directed Markov decision process model correspond to the locations. The start state is the start location, and the goal state is the goal location. The actions of the goal-directed Markov decision process correspond to sequences of one or more movement actions followed by the sensing action. (This assumes that the robot always needs to execute the sensing action before stopping to ensure that it is at the goal location.) In the remainder of the paper, we first explain a special case that is easy to understand but makes several simplifying assumptions. First, we assume that $\gamma > 1$. Second, we assume that the robot has to execute the sensing action after at most $b \geq 1$ movement actions, which makes the number of action sequences finite. We use M^b to denote all sequences of movement actions whose lengths are between one and b. Third, we assume that the costs of the movement and sensing actions do not depend on the location. Executing movement actions $m \in M$ incurs costs $c(m) > 0$ and executing sensing actions o incurs costs $c(o) > 0$. A policy δ then assigns action sequences to locations. Executing action sequence $m_1 \ldots m_j o$ at location l results with cost $\sum_{i=1}^{j} c(m_i) + c(o)$ and probabil-

ity $P(l'|lm_1 \ldots m_j)$ in location l', where the value $P(l'|lm_1 \ldots m_j)$ can be calculated recursively as follows:

$$P(l'|l) = \begin{cases} 1 \text{ if } l = l' \\ 0 \text{ otherwise} \end{cases}$$

$$P(l'|lm_1 \ldots m_i) = \sum_{l''} p(l'|l''m_i) P(l''|lm_1 \ldots m_{i-1}) \qquad \text{for all } 1 \leq i \leq j.$$

We will relax all of these assumptions in Section 6. The example in Section 3 did not include them.

5 Planning Objectives

The plan-execution costs of plans are the sum of the costs of all executed actions. If a plan has plan-execution costs c_i with probabilities p_i, then its expected plan-execution cost is $\sum_i p_i c_i$. Minimizing the expected plan-execution costs is a traditional planning objective. An optimal policy for this planning objective satisfies the following system of $|L|$ Bellman equations, where the $|L|$ unknowns $V(l)$ are the expected plan-execution costs if the start location is known to be l.

$$V(l) = 0 \qquad \text{for all } l \in L \text{ with } l = l_{goal}$$

$$V(l) = \min_{m_1 \ldots m_j o \in M^b o} \sum_{l'} P(l'|lm_1 \ldots m_j)(\sum_{i=1}^{j} c(m_i) + c(o) + V(l')) \text{ for all } l \in L \text{ with } l \neq l_{goal}.$$

A policy that minimizes the expected plan-execution costs can be found with dynamic programming methods from operations research, including policy iteration [6], as well as combinations of these methods and heuristic search methods from artificial intelligence [5]. We modify these methods for a different planning objective, namely to maximize the expected utilities of the plan-execution costs for given exponential utility functions, which allows one to trade-off between minimizing the worst-case, expected, and best-case plan-execution costs. Markov decision process models with this planning objective are studied in operations research in the context of risk-sensitive Markov decision process models [14]. A utility function u maps plan-execution costs c into the corresponding real-valued utilities $u(c)$. If a plan has plan-execution costs c_i with probabilities p_i, then its expected utility is $\sum_i p_i u(c_i)$ and its certainty equivalent is $u^{-1}(\sum_i p_i u(c_i))$ for the given utility function u. Our approach uses exponential utility functions, both convex exponential utility functions: $u(c) = \gamma^{-c}$ for parameter $\gamma > 1$, and concave exponential utility functions: $u(c) = -\gamma^{-c}$ for parameter γ with $0 < \gamma < 1$. Exponential utility functions are perhaps the most often used utility functions in utility theory [20]. Their parameter γ can be used to trade-off between minimizing the best-case, expected, and worst-case plan-execution costs [8]. For example, the plan with maximal expected utility minimizes the best-case plan-execution costs as gamma approaches infinity (under appropriate assumptions). The plan with maximal expected utility minimizes the expected plan-execution costs as gamma approaches one. Finally, the plan with maximal expected utility minimizes the worst-case plan-execution costs as gamma approaches zero. As mentioned earlier,

we assume for now that the exponential utility functions are of the form $u(c) = \gamma^{-c}$ for parameter $\gamma > 1$. An optimal policy for this planning objective satisfies the following system of $|L|$ Bellman equations, where the $|L|$ unknowns $V(l)$ are the expected utilities if the start location is known to be l.

$$V(l) = u(0) = \gamma^{-0} = 1 \qquad \text{for all } l \in L \text{ with } l = l_{goal} \quad (1a)$$

$$\begin{aligned} V(l) &= \max_{m_1 \ldots m_j o \in M^b_o} \sum_{l'} P(l'|lm_1 \ldots m_j) u\Big(\sum_{i=1}^{j} c(m_i) + c(o) + u^{-1}(V(l'))\Big) \qquad (1b) \\ &= \max_{m_1 \ldots m_j o \in M^b_o} \sum_{l'} P(l'|lm_1 \ldots m_j) \gamma^{-\left(\sum_{i=1}^{j} c(m_i) + c(o) - \log_\gamma(V(l'))\right)} \\ &= \max_{m_1 \ldots m_j o \in M^b_o} \gamma^{-\sum_{i=1}^{j} c(m_i) - c(o)} \sum_{l'} P(l'|lm_1 \ldots m_j) V(l') \qquad \text{for all } l \in L \text{ with } l \neq l_{goal}. \end{aligned}$$

That this set of equations has a unique solution follows directly from [15], and that the optimal policy reaches the goal with probability one under the assumption that it is possible to reach every location from every other location follows from [9].

Explanation of the Formulae: If the current location l of the robot is known to be the goal location then plan execution stops right away with plan-execution cost zero and thus expected utility one, which explains equation (1a). To derive the other equation we utilize a property of exponential utility functions called "delta property" [7] or, equivalently, "constant local risk aversion" [17], namely that the certainty equivalent of a plan increases by c if we add the same costs c to all of its plan-execution costs. To see this, assume that a plan has plan-execution costs c_i with probabilities p_i. Then,

$$\begin{aligned} u^{-1}\Big(\sum_i p_i u(c + c_i)\Big) &= -\log_\gamma\Big(\sum_i p_i \gamma^{-(c+c_i)}\Big) = -\log_\gamma\Big(\gamma^{-c} \sum_i p_i \gamma^{-c_i}\Big) \\ &= c - \log_\gamma\Big(\sum_i p_i \gamma^{-c_i}\Big) = c + u^{-1}\Big(\sum_i p_i u(c_i)\Big). \end{aligned}$$

If the current location l of the robot is known not to be the goal location then the execution of action sequence $m_1 \ldots m_j o$ at location l results with cost $\sum_{i=1}^{j} c(m_i) + c(o)$ and probability $P(l'|lm_1 \ldots m_j)$ in location l', where the robot proceeds to execute a plan with expected utility $V(l')$ and thus certainty equivalent $u^{-1}(V(l'))$. The previously executed actions increase this certainty equivalent by their costs (due to the delta property), resulting in a certainty equivalent of $\sum_{i=1}^{j} c(m_i) + c(o) + u^{-1}(V(l'))$ and thus an expected utility of $u(\sum_{i=1}^{j} c(m_i) + c(o) + u^{-1}(V(l')))$. Since this utility occurs with probability $P(l'|lm_1 \ldots m_j)$, the expected utility is $\sum_{l'} P(l'|lm_1 \ldots m_j) u(\sum_{i=1}^{j} c(m_i) + c(o) + u^{-1}(V(l')))$, which explains equation (1b).

A policy that satisfies equations (1a) and (1b) and thus maximizes the expected utility for the given exponential utility function can be found with dynamic programming methods from operations research. For example, Figure 5 shows how to apply policy iteration to this problem. Step 3 corresponds to a search problem that is essentially deterministic. Thus, heuristic search methods from artificial intelligence can be

1. Start with an arbitrary initial policy δ.
2. Compute the value function V^δ for policy δ by solving the following system of $|L|$ equations, where the $|L|$ unknowns $V^\delta(l)$ are the expected utilities if policy δ is followed and the start location is known to be l. Assume that $\delta(l) = m_1 \dots m_j o$. (To keep the notation simple, we do not index the action sequence with $\delta(l)$.). Then,

$$V^\delta(l) = 1 \qquad \text{for all } l \in L \text{ with } l = l_{goal}$$
$$V^\delta(l) = \gamma^{-\sum_{i=1}^{j} c(m_i) - c(o)} \sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l') \quad \text{for all } l \in L \text{ with } l \neq l_{goal}.$$

3. For each location $l \in L$ with $l \neq l_{goal}$, attempt to find an action sequence $m_1 \dots m_j o \in M^b o$ such that

$$V^\delta(l) < \gamma^{-\sum_{i=1}^{j} c(m_i) - c(o)} \sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l').$$

   If such an action sequence is found, set $\delta(l) = m_1 \dots m_j o$ otherwise do not change $\delta(l)$. The search procedure has to find such an action sequence for at least one location if such action sequences exist.
4. If no $\delta(l)$ has changed in the previous step, then stop. Otherwise go to step 2.

Fig. 5. Policy Iteration

used to implement it efficiently. For example, Figure 6 shows how to use A* [16] to implement it for a given location $l \in L$, adapting a method developed by Hansen in the context of minimizing the expected plan-execution costs [5]. The states of the search problem are of the form $lm_1 \dots m_j \in l \cup lM^b$ or $lm_1 \dots m_j o \in lM^b o$.[2] The start state is l. The goal states are of the form $lm_1 \dots m_j o$, that is, the states ending in the sensing action. The actions of the search problem are the movement actions and the observation action. The movement actions $m \in M$ apply to all states of the form $lm_1 \dots m_j$ with $j < b$. We assume that they result with costs $c(m)$ in the state $lm_1 \dots m_j m$. The observation actions o apply to all states of the form $lm_1 \dots m_j$ with $j \leq b$. We assume that they result with costs $c(o) - \log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l'))$ in the state $lm_1 \dots m_j o$. The h-value (heuristic estimate of the goal distance) of state $lm_1 \dots m_j$ is $-\log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l'))$, resulting in an f-value (heuristic estimate of the smallest cost of a path from the start state to a goal state via state $lm_1 \dots m_j$) of $\sum_{i=1}^{j} c(m_i) - \log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l'))$. The h-value of state $lm_1 \dots m_j o$ is zero, resulting in an f-value of $\sum_{i=1}^{j} c(m_i) + c(o) - \log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j) V^\delta(l'))$. Although the h-values are not admissible, A* finds an action sequence $m_1 \dots m_j o$ for at least one location l such that

[2] Instead of using $lm_1 \dots m_j$ as states, we could also use the corresponding probability distributions $P(\cdot|lm_1 \dots m_j)$. This can result in smaller state spaces at the expense of more extensive bookkeeping.

1. Start with the empty OPEN list.
2. Create a search node lm for each movement action $m \in M$ and add them to the OPEN list.
3. Remove the search node with the smallest f-value. If the search node is of the form $lm_1 \dots m_j o \in lM^b o$ (that is, ends in the sensing action) then go to step 5.
4. *(The search node is of the form $lm_1 \dots m_j \in lM^b$:)* Create a search node $lm_1 \dots m_j o$ and add it to the OPEN list. If $j < b$ then also create search nodes $lm_1 \dots m_j m$ for each movement action $m \in M$ and add them to the OPEN list. Go to step 3.
5. *(The search node is of the form $lm_1 \dots m_j o \in lM^b o$:)* If the f-value of $lm_1 \dots m_j o$ is smaller than $-\log_\gamma V^\delta(l)$, that is, if

$$\sum_{i=1}^{j} c(m_i) + c(o) - \log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j)V^\delta(l')) < -\log_\gamma V^\delta(l),$$

then set $\delta(l) = m_1 \dots m_j o$. Return.

Fig. 6. A* Implementation of Step 3 of Policy Iteration for a Given Location l

$$V^\delta(l) < \gamma^{-\sum_{i=1}^{j} c(m_i)-c(o)} \sum_{l'} P(l'|lm_1 \dots m_j)V^\delta(l')$$

$$-\log_\gamma V^\delta(l) > -\log_\gamma(\gamma^{-\sum_{i=1}^{j} c(m_i)-c(o)} \sum_{l'} P(l'|lm_1 \dots m_j)V^\delta(l'))$$

$$-\log_\gamma V^\delta(l) > \sum_{i=1}^{j} c(m_i) + c(o) - \log_\gamma(\sum_{l'} P(l'|lm_1 \dots m_j)V^\delta(l')) = f(m_1 \dots m_j o) \quad (1)$$

if such action sequences exist, as required by policy iteration.

Sketch of the Proof: The proof is similar to Hansen's proof for the planning objective of minimizing the expected plan-execution costs [5]. It is by contradiction. A* terminates because the search space is finite. Assume that there exists an action sequence for some location with property (1) but that A* does not find any such action sequence for any location and consequently does not improve policy δ. Let the action sequence $m_1 \dots m_j o$ at location l be a shortest action sequence with property (1). When A* terminates for location l there is some prefix of $lm_1 \dots m_j o$ on the OPEN list, since initially lm_1 is put on the OPEN list. This prefix cannot be $lm_1 \dots m_j$ or $lm_1 \dots m_j o$ because $f(m_1 \dots m_j) < f(m_1 \dots m_j o) < -\log_\gamma V^\delta(l)$ and thus A* would have been able to improve policy δ. Thus, the prefix must be of the form $lm_1 \dots m_k$ for some $1 \leq k < j$. This state has an f-value that is no smaller than the f-value of the state that terminated the search, otherwise A* would have chosen $lm_1 \dots m_k$ instead. The f-value of the state that terminated the search, in turn, is no smaller than $-\log_\gamma V^\delta(l)$, otherwise A* would have been able to improve policy δ. Finally, we assumed that $-\log_\gamma V^\delta(l)$ was strictly larger than the f-value of $lm_1 \dots m_j o$. Put together,

$$f(m_1 \dots m_k) > f(m_1 \dots m_j o)$$

$$\sum_{i=1}^{k} c(m_i) - \log_\gamma(\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l')) > \sum_{i=1}^{j} c(m_i) + c(o) - \log_\gamma(\sum_{l''} P(l''|lm_1 \ldots m_j)V^\delta(l''))$$

$$-\log_\gamma(\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l')) > \sum_{i=k+1}^{j} c(m_i) + c(o) - \log_\gamma(\sum_{l''} P(l''|lm_1 \ldots m_j)V^\delta(l''))$$

$$-\log_\gamma(\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l')) > -\log_\gamma(\sum_{l''} P(l''|lm_1 \ldots m_j)\gamma^{-\sum_{i=k+1}^{j} c(m_i)-c(o)}V^\delta(l''))$$

$$\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l') < \sum_{l''} P(l''|lm_1 \ldots m_j)\gamma^{-\sum_{i=k+1}^{j} c(m_i)-c(o)}V^\delta(l'')$$

$$\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l') < \sum_{l''}(\sum_{l'} P(l'|lm_1 \ldots m_k)P(l''|l'm_{k+1} \ldots m_j))\gamma^{-\sum_{i=k+1}^{j} c(m_i)-c(o)}V^\delta(l'')$$

$$\sum_{l'} P(l'|lm_1 \ldots m_k)V^\delta(l') < \sum_{l'} P(l'|lm_1 \ldots m_k)(\sum_{l''} P(l''|l'm_{k+1} \ldots m_j)\gamma^{-\sum_{i=k+1}^{j} c(m_i)-c(o)}V^\delta(l'')).$$

Thus, there exists a location l' such that

$$V^\delta(l') < \sum_{l''} P(l''|l'm_{k+1} \ldots m_j)\gamma^{-\sum_{i=k+1}^{j} c(m_i)-c(o)}V^\delta(l'')$$

$$-\log_\gamma V^\delta(l') > \sum_{i=k+1}^{j} c(m_i) + c(o) - \log_\gamma \sum_{l''} P(l''|l'm_{k+1} \ldots m_j)V^\delta(l'')$$

$$-\log_\gamma V^\delta(l') > f(l'm_{k+1} \ldots m_j o).$$

But this is in contradiction to our assumption that the action sequence $m_1 \ldots m_j o$ at location l was the shortest action sequence with property (1) since the action sequence $m_{k+1} \ldots m_j o$ at location l' also has property (1) but is strictly smaller than the action sequence $m_1 \ldots m_j o$. Consequently, A* implements step 3 of policy iteration correctly.

This combination of policy iteration and A* finds a policy that maximizes the expected utility for the given exponential utility function under the assumption that the robot always executes the sensing action after at most b movement actions.

6 Extensions

In the following, we briefly explain (without proofs) how to relax the assumptions that we made earlier for didactic reasons. First, we assumed that $\gamma > 1$. If this is not the case and $0 < \gamma < 1$, then the robots are pessimistic. Our sensor-planning method still applies, with the following changes: The value of the goal location is now minus one instead of one, and the initial policy now has to guarantee that the robot reaches a goal location with probability one from every start location. Such a policy can easily be found by assuming that the robot senses after every movement action. Second, we assumed that the robot had to execute the sensing action after at most b movement actions. If this is not the case, our sensor-planning method can increase

b iteratively as follows, as long as it is optimal to sense in finite intervals (which is always the case for pessimistic robots but might not be the case for optimistic robots): If the length bound b is not used to prune any search path in step 4 of Figure 6 during the last iteration of policy iteration before termination, the policy to which it converges maximizes the expected utility even if no length bound exists. If the length bound is used to prune some search path during the last iteration of policy iteration, then one can increase the length bound and continue to execute policy iteration, until eventually the policy does not change from one iteration to the next and the length bound is not used to prune any search path during the last iteration of policy iteration. Third, we assumed that the costs of the movement and sensing actions did not depend on the location. If this is not the case, then the f-values have to be calculated as follows during the heuristic search, making their calculation slightly less efficient:

$$f(lm_1 \ldots m_j) = \sum_{l'} \bar{P}(l'|lm_1 \cdots m_j) V(l')$$

$$f(lm_1 \ldots m_j o) = \sum_{l'} \bar{P}(l'|lm_1 \cdots m_j) \gamma^{-c(l',o)} V(l'),$$

where

$$\bar{P}(l'|l) = \begin{cases} 1 \text{ if } l = l' \\ 0 \text{ otherwise} \end{cases}$$

$$\bar{P}(l'|lm_1 \ldots m_i) = \sum_{l''} p(l'|l''m_i) \gamma^{-c(l'',m_i)} \bar{P}(l''|lm_1 \ldots m_{i-1}) \qquad \text{for all } 1 \leq i \leq j,$$

which can be calculated when a node is expanded (at the cost of an additional amount of bookkeeping).

7 Conclusions

We described a sensor-planning method that seamlessly trades off between minimizing the best-case, expected, and worst-case plan-execution costs, and thus generalizes the planning objectives of traditional sensor planners that often either minimize the expected plan-execution costs or the worst-case plan-execution costs. Sensor planning with our more realistic planning objectives is interesting because the frequency of sensing depends on the trade-off between minimizing the best-case, expected, and worst-case plan-execution costs: more pessimistic decision makers tend to sense more frequently. We derived our sensor-planning method by combining one of our techniques for planning with non-linear utility functions with an existing sensor-planning method. The resulting sensor-planning method is not only as easy to implement as the sensor-planning method that it extends but also as almost efficient. It assumes that sensing provides perfect information about the state of the world. We are currently working on more general sensor-planning methods that do not make this assumption, using results from the theory of risk-sensitive partially observable Markov decision process models [3].

8 Acknowledgments

The Intelligent Decision-Making Group is partly supported by an NSF Career Award to Sven Koenig under contract IIS-9984827. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.

References

1. B. Abramson. A decision-theoretic framework for integrating sensors in AI plans. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:366–373, 1993.
2. C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1999.
3. E. Fernandez-Gaucherand and S. Marcus. Risk-sensitive optimal control of hidden Markov models: Structural results. *IEEE Transactions on Automatic Control*, 1997.
4. G. Hager. *Task-Directed Sensor Fusion and Planning: A Computational Approach.* Kluwer Academic Publishers, 1990.
5. E. Hansen. Markov decision processes with observation costs. Technical Report CMP-SCI 97–01, Department of Computer Science, University of Massachusetts, Amherst (Massachusetts), 1997.
6. R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, third edition, 1964.
7. R. Howard and J. Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.
8. S. Koenig and R.G. Simmons. How to make reactive planners risk-sensitive. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 293–298, 1994.
9. S. Koenig and R.G. Simmons. Risk-sensitive planning with probabilistic decision graphs. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 2301–2308, 1994.
10. S. Kristensen. Sensor planning with Bayesian decision theory. In L. Dorst, M. van Lambalgen, and R. Voorbraak, editors, *Reasoning with Uncertainty in Robotics*, volume 1093 of *Lecture Notes in Artificial Intelligence*, pages 353–367. Springer, 1996.
11. P. Langley, W. Iba, and J. Shrager. Reactive and automatic behavior in plan execution. In *Proceedings of the International Conference on Planning Systems*, pages 299–304, 1994.
12. S. Lee and X. Zhao. Sensor planning with hierarchically distributed perception net. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 591–598, 1994.
13. T. Lozano-Perez, M. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.
14. S. Marcus, E. Fernàndez-Gaucherand, D. Hernàndez-Hernàndez, S. Colaruppi, and P. Fard. Risk-sensitive Markov decision processes. In C. Byrnes et. al., editor, *Systems and Control in the Twenty-First Century*, pages 263–279. Birkhauser, 1997.
15. H. Mine and S. Osaki. *Markovian Decision Processes.* Elsevier, 1970.
16. N. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.
17. J. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32(1-2):122–136, 1964.

18. A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145, 1995.
19. K. Tarabanis, P. Allen, and R. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, 1995.
20. S. Watson and D. Buede. *Decision Synthesis*. Cambridge University Press, 1987.