A Decision-Theoretic Formalism for Belief-Optimal Reasoning

Kris Hauser
School of Informatics and Computing, Indiana University
Lindley Hall, Room 215
150 S. Woodlawn Ave.
Bloomington, Indiana
hauserk@cs.indiana.edu

ABSTRACT

Intelligent systems must often reason with partial or corrupted information, due to noisy sensors, limited representation capabilities, and inherent problem complexity. Gathering new information and reasoning with existing information comes at a computational or physical cost. This paper presents a formalism to model systems that solve logical reasoning problems in the presence of uncertainty and priced information. The system is modeled a decision-making agent that moves in a probabilistic belief space, where each information-gathering or computation step changes the belief state. This forms a Markov decision process (MDP), and the belief-optimal system operates according to the belief-space policy that optimizes the MDP. This formalism makes the strong assertion that belief-optimal systems solve the reasoning problem at minimal expected cost, given the background knowledge, sensing capabilities, and computational resources available to the system. Furthermore, this paper argues that belief-optimal systems are more likely to avoid overfitting to benchmarks than benchmark-optimized systems. These concepts are illustrated on a variety of toy problems as well as a path optimization problem encountered in motion planning.

1. INTRODUCTION

To complement the standard tool of benchmarking, computer scientists have theoretical tools to express algorithm performance, such as big-O notation for worst-case asymptotic complexity. The power of these tools is that they express rigorous, theoretical performance bounds over all inputs. But for complex intelligent systems that interact and reason about the world, benchmarks are often the most reliable way to measure performance. Indeed it is not clear that worst-case complexity is meaningful in the physical world. For example, the problem of navigation among movable obstacles is PSPACE-complete [20], but humans routinely navigate among movable obstacles without much difficulty. The worst-case problems that are truly hard are puzzles specifically designed to be hard. But benchmarking is not without its flaws; systems can perform well on benchmarks and poorly in the real world, and it can be difficult to design enough benchmarks to fully capture real-world performance.

This paper presents a new formalism to define rigorous, but realistic, theoretical bounds on the performance of systems that reason about and interact with an uncertain world. Specifically, given a scalar valued performance metric, this paper derives a bound on the best expected-case performance over a (typically infinite) space of problem instances Ω. The system is treated as a *decision-making agent* that operates in an "environment" drawn from Ω. The agent does not know which instance was picked, and must learn about the environment by probing it. This perspective has often been taken for systems that operate with physical uncertainty; the novelty in this work is to also treat "mental" uncertainty in the same unified framework. For example, it may know that $f(x) = 0$ but has not yet computed the value of x. Though x is mathematically defined, it is considered a random variable *unknown to the system* until the system computes it.

The formalism casts the operation of the system as a Markov decision process (MDP) (see [2] for a survey). At every point in time the system is identified with a belief state, and it has a set of available actions, such as gathering sensor readings or performing computation. Executing an action moves the belief state in a probabilistic way. To encode the performance metric, a utility function sums costs and rewards encountered by the system before it terminates. The belief-optimal system behaves according to the policy that optimizes expected utility. This paper shows that if the belief state encodes hypotheses about Ω accurately (through appropriate definitions of background knowledge), then on average a belief-optimal system performs better than any algorithm on Ω.

Furthermore, this work highlights the danger of overfitting on benchmarks when they are used to optimize program performance. A program optimized on benchmarks $\Omega' \subset \Omega$ will perform as least as well as any other program on Ω', but may perform poorly on Ω. I argue that systems optimized with decision-theoretic principles are often more robust than those optimized on benchmarks, because any set of benchmarks that represents a complex, high-dimensional problem space is impractically large. The decision-theoretic formalism approximates the problem space with statistical models, which are more likely to generalize to new problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PerMIS'09 September 21-23, 2009, Gaithersburg, MD, USA
Copyright 2009 ACM 978-1-60558-747-9/09/09 ...$10.00.

2. RELATED WORK

Decision theory is a well-known tool for modeling and designing systems that interact with the real world, but it also has powerful applications to *logical problem solving*. Decision-theoretic approaches have been explored for reasoning problems in a diverse range of fields, such as numerical analysis, artificial intelligence, and robotics, where solutions are deterministically defined, and no *inherent* uncertainty exists except in the "mind" of the system. They have also been used to model bounded rationality in humans [17].

Examples of applications include analysis of Monte Carlo and quasi-Monte Carlo numerical integration and function approximation techniques [19], and in sampling strategies for stochastic numerical optimization [5, 21]. In artificial intelligence, efficient strategies have been developed for testing hypotheses represented as boolean formulas of uncertain statements [8]. Decision theoretic models have been applied to heuristic selection in heuristic search [4]. They have also shown promise in the field of robot motion planning, with speed gains of up to orders of magnitude [3, 11, 18]. My own research has applied decision-theoretic approaches to several motion planning subproblems [9], including path optimization, collision testing, configuration sampling, and contact selection strategies for robotic systems with contact.

It has also been argued that certain heuristics for randomized algorithms can be understood in a decision-theoretic sense, even if the heuristics are not themselves derived from the same principles. For example, stochastic optimization heuristics like simulated annealing, genetic algorithms, and ant colony optimization can be interpreted has having implicit probabilistic models of the function space [21]. The probabilistic roadmap (PRM) technique for robot motion planning, which connects a network of randomly sampled points in the robot's free space, can be interpreted as having implicit hypotheses about the shape of the free space [10]. This suggests that the performance of these algorithms may be improved by using better initial hypotheses, or better exploiting the information gathered during computation.

3. FORMAL MODELING OF BELIEF SPACE REASONING

This section describes how to model an intelligent reasoning system as a decision-making agent, and describes the associated POMDP formalism.

3.1 Assumptions

Consider a computer program that is given some background knowledge as input, and produces an output after executing a sequence of tests, which may involve either computational reasoning or gathering sensor readings. Assume that the sequencing of tests and termination conditions are structured by the problem logic such that the output is always correct. The outcome and/or cost of each test behaves with uncertainty, due either to stochasticity (a result of randomization or physical noise) or unpredictability (must be treated as a black box with internal workings that are too complex to be explicitly represented).

3.2 Modeling as a Belief-Space POMDP

The agent operates on a number of hypothetical logical *statements*. To a statement S, assign a belief p in $[0, 1]$, with $p = 0$ meaning S is certainly false and $p = 1$ meaning S is certainly true. An assignment of beliefs to all statements in the scope of a problem defines a *belief state*, and the set of all possible belief states is the *belief space*.

The program executes a sequence of *tests*, which are atomic operations that modify the belief on hypothetical statements. Upon observing the results of executing a test, the program moves to a new belief state. Without loss of generality, we define a test such that it determines the factuality of a statement S exactly.

The problem is considered solved when the belief state contains certain factual statements. The program may choose to terminate, which produces an output. The performance of the system is measured by a utility function that sums the negative *costs* incurred during execution, and positive *rewards* that assess output quality. Costs may include execution time and resource usage; for example, real-time constraints could be implemented by penalizing time limit violations. Rewards and costs should be weighted by the system designer so that the utility function measures overall system performance.

In full generality, a belief state on statements $\mathcal{S}$ forms a joint probability distribution $Pr(\mathcal{S}|Z)$. Here, Z represents *background knowledge* established prior to the current state. The importance of background knowledge will be discussed in Section 4.1. After executing a test T, the belief state should change to $Pr(\mathcal{S}|T \text{ succeeds}, Z)$ with probability $Pr(T \text{ succeeds}|Z)$, and to $Pr(\mathcal{S}|T \text{ fails}, Z)$ with probability $Pr(T \text{ fails}|Z)$. These changes are called the *transition dynamics*.

With these definitions, the problem solver has been cast as a belief-space Markov decision process. Because the variables that define the belief state are not directly observable, the MDP is known as a partially-observable Markov decision process (POMDP).

3.3 Illustration on Matrix Inversion

To illustrate the approach on a small example, suppose we are designing a system to compute the inverse or pseudoinverse of a matrix, where the properties of the matrix are unknown beforehand. Suppose the system has access two three methods of computing inverses: Cholesky decomposition, LU decomposition, and singular value decomposition (SVD). The challenge in this particular system is that Cholesky decomposition is much faster than LU decomposition when the matrix is symmetric positive definite (s.p.d.), but this property is unknown beforehand (in fact, attempting a Cholesky decomposition is frequently used to test if an unknown matrix is s.p.d.). Furthermore, LU decomposition is much faster than SVD when a matrix is invertible, but again, invertibility is unknown beforehand. SVD will compute a pseudoinverse or inverse for a general matrix.

Here, the belief space consists of four unknown properties of the input matrix: Square, Symmetric, Invertible, and SPD (symmetric positive-definite). The tests available to the system consist of two "probes", Is-Square and Is-Symmetric, and the three methods of computing inverses, labeled Cholesky, LU, SVD. The operations listed in the order of increasing cost are Is-Square, Is-Symmetric, Cholesky, LU, and SVD.

The optimal policy depends on the distribution of the kinds of matrices in Ω. For example, if very few matrices are invertible, then the policy of always using SVD would be

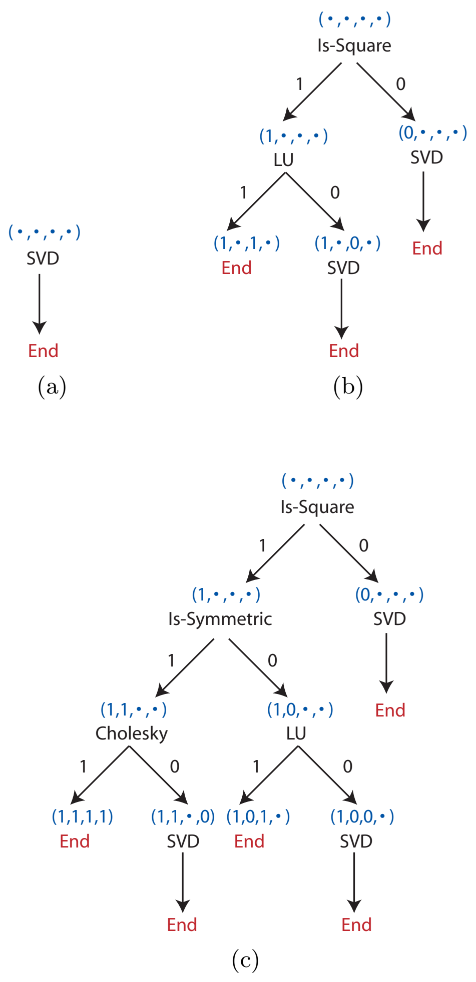

Figure 1: Three policies for the matrix inversion example which may be optimal, depending on the distribution of input matrices. Tuples $(\cdot,\cdot,\cdot,\cdot)$ indicate belief states, where elements respectively indicate that a matrix is square, symmetric, invertible, and symmetric positive-definite.

optimal (Figure 1a). If many matrices are invertible but few are symmetric positive-definite, then the policy in Figure 1b may be optimal. If all kinds of matrices are well-represented, then the optimal policy may gather as much information as possible, as illustrated in Figure 1c.

The joint distribution of Ω over the belief space (which can be represented by 16 real values), and the distribution of costs associated with each test, is sufficient to compute the optimal policy using the POMDP framework. This is not difficult for a problem of this size, but can be tremendously challenging for problems that are not much larger.

3.4 Illustration on a Household Mobile Robot

Consider another example of a mobile robot that is asked to retrieve an item in a house. It has a map of the house, and knows of N locations $L_1,\ldots,L_N$ where the item may be, some of which are inaccessible (out of reach or blocked by obstacles). The true location of the item is given by the random variable I. The robot does not know a priori which locations are inaccessible, and instead calls a planning sub-

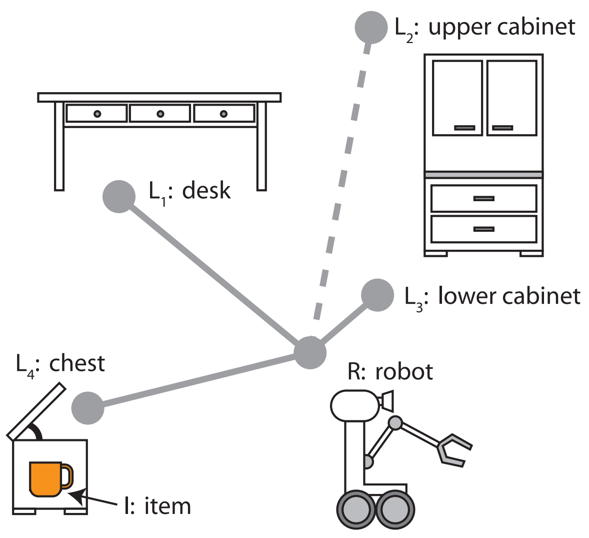

Figure 2: A household mobile robot is asked to search for a cup. The robot does not know, a priori, that the upper cabinet location is inaccessible. The robot determines accessibility by running a planning subroutine. Physical uncertainty is present in the location of the cup, and "mental" uncertainty is present in the accessibility of locations.

routine Plan(L_i) that either returns a path from its current location R to the location L_i, or reports that it is inaccessible. If a path is available, Travel(L_i) moves the robot to the location and looks for the item. This example combines both physical uncertainty (where the item is located), as well as "mental" uncertainty (which locations are accessible).

If we denote the accessibility of location L_i with the variable A_i, the belief state is defined over the statements $(I, R, A_1, \ldots, A_N)$. The program terminates when the item is found, or the robot knows the item is in an inaccessible location. The background knowledge will define the probability distribution on I and $A_1,\ldots,A_N$, as well as the expected costs of Plan() and Travel().

4. BELIEF-OPTIMAL POLICIES

The decision-theoretic formalism almost immediately defines the concept of a *belief-optimal policy*, and the performance bound associated with it. More precisely, let $U(\pi, Z)$ denote the expected utility of executing a policy π given background knowledge Z. Then the belief-optimal policy is

$$\pi^\star = \operatorname*{argmax}_\pi U(\pi, Z).$$

The next section shows that $U(\pi^\star, Z)$ is an upper bound to the average performance for problem instances drawn from Ω, so long as Z is properly defined.

4.1 The Role of Background Knowledge

Typically, Ω is enormous and unknown, so beliefs about Ω must be represented in the background knowledge Z. Suppose that background knowledge is defined such that, given any observed test results, the distribution of future test results can be inferred *accurately*. Then, the policy that optimizes the POMDP is the policy that achieves the lowest expected cost over Ω.

Accuracy is defined as follows. Let $\mu(X)$ denote the probability that a problem instance is drawn from a subset $X \subseteq \Omega$. Let $\Omega_{\mathcal{T}}$ be the set of instances that are consistent with a history of observed tests $\mathcal{T}$. Then the background knowledge Z is accurate if

$$Pr(T|\mathcal{T}, Z) = \frac{\mu(\Omega_{\mathcal{T}\cup\{T\}})}{\mu(\Omega_{\mathcal{T}})} \tag{1}$$

holds for any history of tests $\mathcal{T}$ and any future test T.

Appendix A shows that if background knowledge is accurate, then $U(\pi, Z)$ is equal to the average performance of π on the problem instances in Ω, and by consequence, $U(\pi^\star, Z)$ is an upper bound on performance. So in theory, the belief-optimal policy and performance bound are rigorously defined, and can be computed for small problems. But in practice, one can only hope for approximate solutions.

4.2 Representing and Computing Belief States

It is usually extremely difficult to define a problem distribution Ω and μ, much less compute an accurate representation in background knowledge. So for algorithm designers the perspective is reversed: background knowledge is chosen explicitly, and the problem distribution is defined implicitly to be consistent with the chosen background knowledge. Background knowledge can be encoded using a variety of machine learning and statistical models, e.g., logistic models, Bayesian networks, decision trees, neural networks, etc. Such models can encode prior beliefs as well as incorporate training on real problem instances. This in turn enables belief-optimal policies to generalize better to unseen instances, as argued in Section 6.

It is also impractical to compute or explicitly represent the joint distribution of the belief state because its size is exponential in the number of statements. The belief state and dynamics can be approximated by assuming statement independence, in which case the joint distribution is simply the product of distributions of individual statements in $\mathcal{S}$. More refined strategies might use assumptions of conditional independence, for example, representing variables in a sparse Bayesian network or other graphical models.

4.3 Optimizing POMDP Policies

Solving POMDPs in the general case is extremely computationally hard [15, 16]. In general there are two primary approaches to solving large POMDPs (which are not mutually exclusive): exploit problem structure, or approximate. Some problems have a structure for which exact solutions can be computed efficiently, such as several variants of the n-armed bandit problem [1, 4], and hypothesis testing of two-level AND/OR boolean-formulas [8]. POMDP approximation in large belief spaces is an area of active research, and perhaps the most promising results have come from sparse sampling and depth-limited search techniques [12, 13].

The advent of POMDP optimization techniques for large problems, on the order of thousands or millions of logical statements, will greatly accelerate the rate of development of complex intelligent systems. Intelligent systems will inevitably need to tackle computationally hard problems more frequently in the future, and belief-based optimization is an attractive, systematic approach to achieving good practical performance in the face of discouraging worst-case complexity theories.

5. A PATH SMOOTHING EXAMPLE

Here we illustrate a practical application of the decision-theoretic formalism to a path smoothing problem encountered in motion planning. Probabilistic roadmap motion planners tend to produce jerky, unnatural-looking paths due to their random exploration. A simple shortcutting method [6, 7] smoothes these paths by picking two random points A and B on a path, and testing the line segment $\overline{AB}$ for feasibility. If $\overline{AB}$ lies in free space, it replaces the portion of the path between A and B. After a handful of iterations, the largest unnecessary jerks are likely to be eliminated. However, it can take a huge number of iterations before the process converges to a smooth path. How many iterations are enough?

In a decision-theoretic formulation, the agent chooses shortcuts as tests and receives rewards that are accumulated over time. Each potential shortcut incurs a negative cost, and produces a reward only if it is successful. Although it is difficult to define background knowledge accurately, we approximate it by making some independence assumptions. The belief-optimal policy, given these assumptions ,is a greedy strategy. Experiments demonstrate that the greedy strategy converges much faster than picking random points, and has a natural termination criterion: halt when no shortcut has positive expected utility.

5.1 Problem Statement

Suppose the path $y(u)$ is parameterized with u in [0,1]. Denote the length of the path between parameters u and u' as $l(u, u')$, and denote the distance between two points q and q' in C-space as $d(q, q')$. Testing the line segment between u and u' incurs cost $c(u, u')$. If successful, $y(u)$ is replaced with the new path, and the planner receives reward $l(u, u') - d(y(u), y(u'))$ (the amount that path length is reduced). Let $p(u, u')$ denote the estimated probability of success. We assume $c(u, u') = c_s d(y(u), y(u'))$, where c_s is a constant reflecting the amount of computation time one is willing to spend for a unit decrease in path length.

This is essentially a "one-pull" variant of the well-studied n-armed bandit problem [1]. In the n-armed bandit problem, the decision maker chooses from n actions, $A_1, \ldots, A_n$. A stochastic payout z_k is awarded after choosing A_k, and payouts accumulate over time. If the distributions of each z_k are known and independent, the optimal strategy is greedy and picks the choice with the highest expected reward. This is also the case for "one-pull" bandits.

In other words, the greedy choice of u and u' maximizes $p(u, u')(l(u, u') - d(q, q')) - c(u, u')$. Dependencies between two candidate shortcuts do occur when the range of the shortcuts overlap, so the greedy strategy is not necessarily optimal, but it does perform quite well in practice.

5.2 Belief Estimation

The performance of the greedy strategy depends on the quality of the estimate $p(u, u')$. From first principles we know that the probability that a random line segment is feasible decreases with distance between its endpoints. Also, a line segment is likely to be invalid if nearby line segments are invalid. We keep a history of infeasible configurations $\mathcal{I}$ (which may be initialized with samples from PRM planning) and updated as shortcuts fail. We assign a belief to each shortcut as a function of segment length and the distance between the segment and the closest configuration in $\mathcal{I}$.

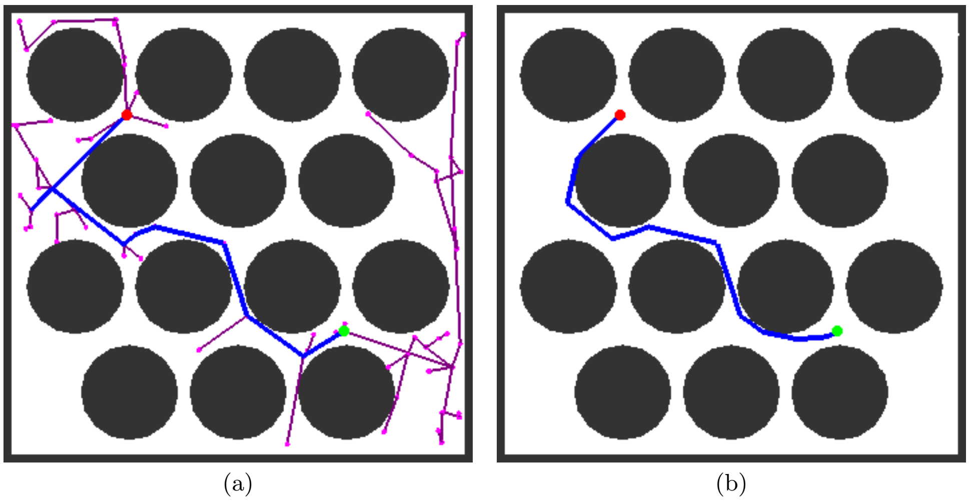

Figure 3: (a) PRM planners produce jerky paths. (b) Shortcutting produces a shorter path.

In the following experiments, we use a crude method to estimate $p(u, u')$ given $\mathcal{I}$. We tested the feasibility of 100,000 randomly sampled line segments, and built a histogram $p_0(d)$ of success rate indexed by distance d. We then weight the baseline probability $p_0(d(q, q'))$ by a function of the distance d' to the closest infeasible configuration. The experiments below used $f(d') = 1 - e^{-\alpha d'}$, where α was chosen by a small amount of tuning. A better method might estimate the joint success rate as a function of $d(q, q')$ and d'.

5.3 Experimental results

We demonstrate this for a C-space with regularly spaced circular obstacles in a unit square (Figure 3). The shortest path between two points may contain curves on the C-space obstacle boundaries (this is also true in C-spaces for robotic mechanisms with revolute joints). A good piecewise linear approximation to the shortest path requires a huge number of line segments, so shortcutting converges slowly.

To compare performance across varying start and goal configurations, we normalize reward by dividing by the straight-line distance between the starting points. We set the cost proportionality constant c_s to 0.01. Figure 4 compares the randomized strategy with a greedy, adaptive strategy. 200 random start and goal locations were sampled and connected using an RRT planner [14]. Starting at the output path, random shortcuts were performed for 1000 iterations. For the same starting path, adaptive shortcuts were performed until the strategy chose to terminate. The set of infeasible configurations $\mathcal{I}$ is initially empty.

Initially, the adaptive strategy reduces the path length quickly. Throughout execution, it achieves a given reduction in path length about two or three times faster than the randomized strategy. As more iterations are taken, shortening the path becomes increasingly harder. The adaptive strategy terminates naturally when the cost of making a shortcut exceeds the expected reward. All adaptive runs terminated by iteration 200.

6. AN ALTERNATIVE TO BENCHMARKS

6.1 An Argument Against Benchmarks

Benchmarks are a useful tool for performance optimization, but it is difficult to pick a representative benchmarking suite, and caution must be taken to avoid overfitting. In general, there will exist a sub-optimal algorithm whose benchmark performance is at least as good as the optimal algorithm. Existing techniques for avoiding overfitting include using large benchmarking sets, regularization, and cross-validation. Outside of the machine learning community, researchers rarely employ such techniques (I myself am guilty of this), and even if they are employed, the size of the benchmarking suite is rarely adequate to draw statistically significant conclusions.

One reason why it is so difficult to choose a benchmarking suite that accurately reflects real-world performance is that any set of problem instances that is statistically identical to Ω, as far as the algorithm is concerned, must span the joint distribution of Ω in belief space. Such a set is practical only if the belief space is tiny, or Ω happens to span a low dimensional subspace of belief space.

I argue that performance optimization using the decision-theoretic formalism can be more robust than benchmarks. The system's background knowledge Z can be defined to capture the belief space distribution of Ω more accurately and more broadly than a set of benchmarks.

A skeptic may raise two objections. First, background knowledge should be trained on a set of problem instances, which is essentially a set of benchmarks. Second, in practice, background knowledge can only represent the distribution of Ω approximately, and these modeling errors may reduce performance. To the first objection, I argue that background knowledge can better generalize small datasets by taking advantage of statistical and machine learning tools that are based on sound and well-developed theory (some of which were mentioned in Section 4.2). These tools will generalize far better than ad-hoc techniques that try to encode generalization into a complex algorithm. To the second objection, the empirical success of the decision-theoretic approach suggests that their behavior is relatively robust to approximation errors. An example will be shown in the following section.

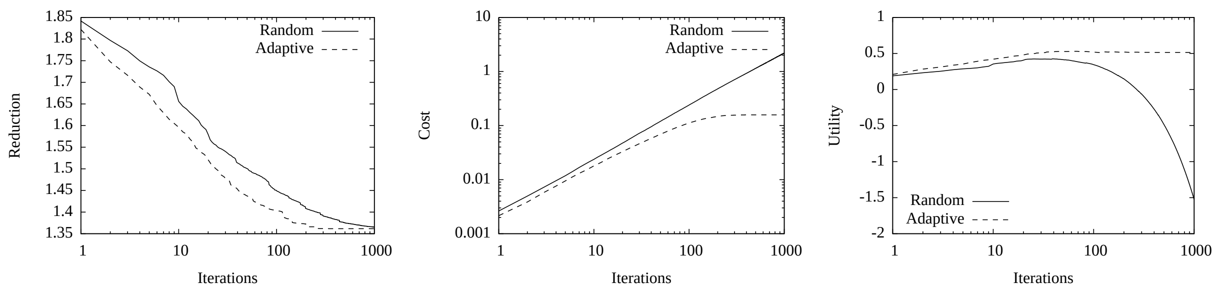

Figure 4: Results of shortcutting experiments, reporting relative path length, accumulated cost, and utility. Averaged over 200 plans with random start and goal configurations. Iteration numbers are plotted on a log scale.

6.2 Illustration on an Urn Problem

Consider a system that searches for a black ball placed in one of two urns. In each problem instance, the black ball is placed into an urn and both urns are filled with additional white balls. The system has two types of tests: A_k asks the number of balls in urn k, and U_k picks urn k and searches through it. A search continues until it finds the black ball or empties the urn completely (that is, it is not allowed to stop midway through). A problem instance can be specified as a 3-tuple (n_1, n_2, K), where n_1 and n_2 are respectively the number of balls in the 1st and 2nd urns, and K is the index of the urn containing the black ball. The cost of A_k is 1, and the expected cost of U_k is $n_k/2$ if $K = k$, and is n_k otherwise.

Suppose the benchmark problem instances are $(30, 1, 1)$, $(20, 15, 1)$, $(40, 30, 2)$, and $(35, 60, 2)$. A benchmark-optimal strategy is to test A_2, and if $n_2 \geq 30$, then to pick U_2 first. Otherwise, pick U_1 first. Call this strategy π_1. On the benchmarks, π_1 has average cost 18.5.

Suppose now that our *approximate* background knowledge consists of a minimum and maximum estimate of the number of balls in each urn, and the fraction of black balls found in urn 1. Also, let the system assume that the number of balls in an urn has a uniform distribution between the minimum and maximum values, and is independent of the number in the other urn. Given these approximations, the POMDP expected utility of π_1 is 35. Now consider a strategy π_2 that tests both A_1 and A_2, and then picks U_1 first if $n_1 < n_2$ and U_2 otherwise. The expected value of this strategy is approximately 28.3. The worst-case cost of π_2 is also better than π_1: 72 versus 81.

7. CONCLUSIONS

This paper presented a decision-theoretic formalism for optimizing and bounding the expected performance of an intelligent system that interacts and reasons about the physical world. The formalism is based on casting the system as a decision-making agent that faces uncertainty due to sensor noise as well as uncertainty due to bounded rationality. This forms a partially-overvable Markov decision process (POMDP) that can be optimized. I showed that if the beliefs of the system accurately capture the problem space characteristics, then the strategy that optimizes the POMDP also optimizes the average case performance over the problem space. This paper also argued that the decision-theoretic approach is an attractive alternative to benchmarking for performance optimization that avoids the problem of overfitting to benchmarks. These principles are illustrated on a variety of example problems. Future work should investigate the tractability and accuracy of approximation techniques for solving the large POMDPs that result from this formalism, and also to apply the formalism to systems that solve new and challenging problems.

8. ACKNOWLEDGEMENT

Many thanks to Jean-Claude Latombe for numerous helpful discussions during the early stages of this work.

APPENDIX

This appendix derives the result that if background knowledge Z is defined such that (1) holds, then $V(\pi, \Omega) = U(\pi, Z)$. It is largely a technical matter, but is included here for completeness.

To define U and V more precisely we first need some preliminary definitions. Here we will represent a state s as a history of tests $\{T_1, \ldots, T_n\}$, because the belief state and dynamics are fully determined by the background knowledge (which stays constant) and the test history. Let $\pi(s)$ denote the action taken by policy π at state s, and in a slight abuse of notation, also let it denote the result of the test $\pi(s)$. Let $R(s, a)$ be the reward minus the cost of executing action a in state s. Assume without loss of generality that Ω is defined such that given a problem instance ω, transitions are deterministic (i.e., ω is the "true" value of the world state). Also, assume terminal states are absorbing, and the performance metric is bounded.

Define the utility function $U_\pi(s)$ as the unique solution to the following system of equations over s:

$$\begin{aligned} U_\pi(s) = R(s, \pi(s)) &+ Pr(\pi(s)=0|s, Z) U_\pi(s \cup \{\pi(s)=0\}) \\ &+ Pr(\pi(s)=1|s, Z) U_\pi(s \cup \{\pi(s)=1\}). \end{aligned} \tag{2}$$

This is a slight abuse of notation; our original definition of the utility function is $U(\pi, Z) \equiv U_\pi(\{\})$.

Let the trace of running policy π on instance ω be defined as $(s_0, s_1, \ldots)$. Define the return v_0 using the recursive formula

$$v_i(\pi, \omega) = R(s_i, \pi(s_i)) + v_{i+1}(\pi, \omega).$$

Then the average performance of π is

$$V(\pi, \Omega) = \int_{\omega \in \Omega} v_0(\pi, \omega) d\mu(\omega).$$

Let's define

$$Y_\pi(s) = \int_{\omega \in \Omega_s} v_{|s|}(\omega, \pi) d\mu(\omega)$$

so that $Y_\pi(\{\}) \equiv V(\pi, \Omega)$. Then, denote $s' = s \cup \{\pi(s)=1\}$ and $s'' = s \cup \{\pi(s)=0\}$, and split Ω_s into subsets $\Omega_{s'}$ and $\Omega_{s''}$ depending on the results of $\pi(s)$. We get

$$\begin{aligned} Y_\pi(s) &= \int_{\omega \in \Omega_s} R(s, \pi(s)) + v_{|s|+1}(\omega, \pi) d\mu(\omega) \\ &= \mu(\Omega_s) R(s, \pi(s)) + \int_{\omega \in \Omega_{s'}} v_{|s|+1}(\omega, \pi) d\mu(\omega) \\ &\quad + \int_{\omega \in \Omega_{s''}} v_{|s|+1}(\omega, \pi) d\mu(\omega) \\ &= \mu(\Omega_s) R(s, \pi(s)) + Y_\pi(s') + Y_\pi(s'') \end{aligned} \tag{3}$$

Now if we let $X_\pi(s) = Y_\pi(s)/\mu(\Omega_s)$, we have

$$X_\pi(s) = R(s, \pi(s)) + \frac{\mu(\Omega_{s'})}{\mu(\Omega_s)} X_\pi(s') + \frac{\mu(\Omega_{s''})}{\mu(\Omega_s)} X_\pi(s'') \tag{4}$$

Since background knowledge is accurate, then by (1) we have

$$\begin{aligned} X_\pi(s) = R(s, \pi(s)) &+ Pr(\pi(s)=0|s, Z) X_\pi(s \cup \{\pi(s)=0\}) \\ &+ Pr(\pi(s)=1|s, Z) X_\pi(s \cup \{\pi(s)=1\}). \end{aligned} \tag{5}$$

Since X_π solves for (2), then it must be identical to U_π. Then $V(\pi, \Omega) \equiv Y_\pi(\{\}) = U_\pi(\{\}) \equiv U(\pi, Z)$ as desired.

A. REFERENCES

[1] D. Berry and B. Fristedt. *Bandit Problems.* Chapman and Hall, 1985.

[2] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[3] B. Burns and O. Brock. Single-query motion planning with utility-guided random trees. In *IEEE Int. Conf. Rob. Aut.*, 2007.

[4] V. A. Cicirello and S. F. Smith. The max k-armed bandit: A new model of exploration applied to search heuristic selection. In *AAAI*, 2005.

[5] D. Fouskakis and D. Draper. Comparing stochastic optimization methods for variable selection in binary outcome prediction, with application to health policy. *Journal of the American Statistical Association*, 03(484):1367–1381, 2008.

[6] R. Geraerts and M. Overmars. Clearance based path optimization for motion planning. In *IEEE Int. Conf. Rob. Aut.*, New Orleans, LA, 2004.

[7] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *Intl. J. of Rob. Res.*, 26(8):845–863, 2007.

[8] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170:19–58, 2006.

[9] K. Hauser. *Motion Planning for Legged and Humanoid Robots.* PhD thesis, Stanford University, 2008.

[10] D. Hsu, J. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Rob. Res.*, 25(7):627–643, 2006.

[11] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *IEEE Int. Conf. Rob. Aut.*, pages 3885–3891, Barcelona, Spain, 2005.

[12] M. Kearns, Y. Mansour, and A. Y. Ng. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[13] H. Kurniawati, D. Hsu, , and W. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

[14] S. M. LaValle and J. J. Kuffner, Jr. Rapidly-exploring random trees: progress and prospects. In *WAFR*, 2000.

[15] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.

[16] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov chain decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[17] A. Rubinstein. *Modeling Bounded Rationality.* MIT Press, 1998.

[18] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *Int. J. of Rob. Res.*, 21(1):5–26, 2002.

[19] J. F. Traub, H. W. G. W., and Wasilkowski. *Information-Based Complexity.* Academic Press, New York, 1988.

[20] G. Wilfong. Motion planning in the presence of movable obstacles. In *Fourth Annual Symposium on Computational Geometry*, pages 279 – 288, Urbana-Champaign, IL, 1988.

[21] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1–4):373–395, October 2004.