Decision Theoretic Execution Strategies for Modularized Problem Solvers

Kris Hauser Jean-Claude Latombe

Department of Computer Science
Stanford University, Stanford, CA 94305
khauser@cs.stanford.edu, latombe@cs.stanford.edu

SUMMARY

Large-scale problem solving algorithms will prove to be critical in implementing the long-term vision of cyber-physical systems (CPS). Such algorithms will be highly modularized -- that is, they will use subroutines implemented behind abstract interfaces. A characteristic of many problem solving algorithms is that their performance varies wildly depending on which order subroutines are executed, with which parameters, and which computational resources. These choices must be made in the face of unpredictable subroutine behavior. We propose a formalism that treats the problem solving algorithm as a decision-making agent in a belief space, casting the problem of choosing a subroutine execution strategy as a Markov decision process (MDP). This formalism has already been applied successfully to many existing problems in robotics and AI, and holds promise for CPS applications and fundamental CPS research in the future.

I. INTRODUCTION

Cyber-physical systems (CPS) will inevitably pose computational problems of tremendous scope and complexity. For example, one fundamental problem is to decide how to drive the physical system to a desired state. This *planning/control* problem has been well-studied in robotics, but to achieve the grand visions of CPS, we must be able to solve problems of far higher complexity than anything that has been solved before. To adequately represent the physical world, configuration/state spaces will need thousands (perhaps even millions?) of dimensions, with highly nonlinear and unpredictable dynamics.

For practical reasons, problem solving algorithms in CPS will be highly *modularized*, by breaking problems into subproblems to be solved individually. Modularization helps utilize the highly distributed computational resources that characterize CPS. Furthermore, it can help with computational complexity by avoiding the solution of unnecessary subproblems. This gives the problem solver considerable flexibility in the order it chooses to solve subproblems, and with which subroutines and computational resources. We call these choices its *execution strategy.* For example, to check a robot trajectory for collision, one may break the trajectory into several pieces and test them separately. If one can estimate the likelihood for a certain piece to collide, a good strategy first tests the piece that is the most likely to collide (Sánchez and Latombe, 2002). Many algorithms may also be able to perform partial or approximate computation with adjustable levels of accuracy (and consequently, various running times). For example, a collision checker based on bounding volume hierarchies [ref] may only test the top-level bounding boxes for overlap, or k levels, or all levels. The sequencing of the subroutines and the choice of accuracy levels form the *execution strategy.* Different strategies may have widely different performance, both in execution time (and more generally, use of resources) and in quality of the results. On the other hand, many subroutines have unpredictable performance, either because they take noisy inputs (e.g., sensory data), have randomized components, or are highly sensitive to small variations in the input date.

We propose a *decision theoretic* framework for optimizing a problem solver's execution strategy. The problem solver is modeled as a decision-making agent in a *belief space*, facing *uncertainty* about a subroutine's output, computational cost, and reliability. This defines a partially observable Markov decision process (POMDP), where the agent attempts to minimize computation costs and maximize output quality. We have applied this framework with great success to a wide range of subproblems in robot motion planning, sometimes improving prior approaches by orders of magnitude. We stress that the approach is general, even though, so far, we have focused on motion planning. We conclude with recommendations for future work.

II. DECISION THEORETIC FRAMEWORK

The decision theoretic framework casts a problem solver as a POMDP. We currently assume the solver solves subproblems sequentially. Uncertainty lies in the output and cost of a subroutine, and is either stochastic (a result of randomized algorithms or stochastic input) or unpredictable (has high sensitivity to variations in input). In the scope of a given problem, the solver operates on hypothetical logical *statements*. To a statement S, assign a belief p in [0,1], with p=0 meaning S is invalid (certain falseness) and p=1 meaning S is valid (certain truth). 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 solver operates by choosing a sequence of *tests*, primitive operations that modify the belief on hypothetical statements. Upon observing the results of executing a test, the solver moves to a new belief state. An *exact* test determines the factuality of a statement *S* exactly, with probability of success equal to the belief on *S*. A *partial test* modifies the belief on *S* without making it into a fact. (This can be modeled as an exact test on an auxiliary statement *S'*, where *S* and *S'* are dependent.) A partial test is *incremental* if it can be controlled step-by-step, incrementally changing the belief on S. This class of tests includes the bounding-volume collision checker mentioned in the Introduction, as well as includes probabilistically complete tests like Las Vegas algorithms, used often in motion planning (e.g., Probabilistic Roadmaps, Kavraki et. al. 1996). Incremental tests can also model remote procedures with unreliable communications.

The problem is considered solved when the belief state contains certain truths. The solver may then choose to terminate, or it may decide to compute a better solution. The *utility* function consists of positive *rewards* received by the solver upon termination (a function of the belief state upon termination) and negative *costs* incurred during execution. Rewards assess the output quality. Costs measure the cost of execution, including computation time and resource usage. The designer should weight rewards and costs reflecting his or her judgment of the relative importance of these factors. For example, real-time constraints could be implemented by penalizing time limit violations. The optimal *execution strategy* (i.e. policy) terminates with the maximum expected utility.

In full generality, a belief state on statements $\mathcal{S}$ and future test results $\mathcal{T}$ forms a joint probability distribution $\Pr(\mathcal{S},\mathcal{T} \mid Z)$, were, Z represents *background knowledge* established prior to the current solver state. After executing a test T, the belief state should change to $\Pr(\mathcal{S},\mathcal{T} \mid T \text{ succeeds}, Z)$ with probability $\Pr(T \text{ succeeds} \mid Z)$, or to $\Pr(\mathcal{S},\mathcal{T} \mid T \text{ fails}, Z)$ with probability $\Pr(T \text{ fails} \mid Z)$. There are often too many logical statements to explicitly represent, so their costs and beliefs must be initialized on demand. A joint distribution is impractical to compute or represent explicitly because its size is exponential in the number of statements. The extreme alternative is to assume 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. Background knowledge may be generated using a variety of machine learning and statistical models.

Solutions to general POMDPs are notoriously hard to compute (Madani et. al., 1999), but approximation techniques are a subject of active AI research. An alternative is to analyze widely-applicable *model problems*, such as the *n*-armed bandit problem (Berry and Fristedt, 1985) or boolean formula testing (Griener et. al. 2006), as templates. Our experience in robotics has been that it is often possible to efficiently compute execution strategies that outperform heuristic strategies by making crude independence assumptions (Hauser, 2008).

III. APPLICATIONS

To illustrate, consider optimizing the length of a collision-free robot path in a (high-dimensional) configuration space. Suppose we are given one primitive action: checking if a line segment between two points on the path has a collision. Then, we can shorten the path by repeatedly performing "shortcuts": we test for a segment for collision, and the segment replaces the intermediate section of the path if it is collision-free. But which line segments should we check? The decision theoretic approach assigns a belief that a segment has a collision, a cost to check collisions, and a payout established upon termination (the overall reduction in path length). Even using a crude belief representation (probability of collision increasing with segment length), experiments show that a greedy strategy is 2 to 3 times faster at reducing path length than a frequently-used heuristic strategy of picking points at random (Hauser, 2008). Furthermore, the greedy strategy terminates natural when expected utility is negative, which corresponds to the case where the path is already fairly optimized, and further shortcuts are unlikely to provide much benefit.

The decision theoretic framework formalizes a number of existing approaches for speeding up problem solving algorithms. Efficient strategies have been developed for testing hypotheses represented as boolean formulas of uncertain statements (Griener et. al. 2006). Decision theoretic models have been applied to heuristic selection in heuristic search (Cicirello and Smith, 2005). They have also shown promise in the field of robot motion planning, with speed gains of up to orders of magnitude (Burns and Brock, 2007; Hsu et. al. 2005; Sánchez and Latombe, 2002). In our research, we have successfully applied this approach to several motion

planning subproblems (Hauser, 2008), including path optimization, collision testing, configuration sampling, and contact selection strategies for robotic systems with contact.

This decision theoretic approach has a number of potential applications in robotics and CPS that have yet to be explored. Most existing applications of the approach have focused on "mental" uncertainty (i.e., beliefs), but *physical* uncertainty (i.e., sensor noise) can be treated as well. For example, the framework may formalize distributed active learning systems that probe the world to gain information. These probes could be physical sensors, such as cameras or touch sensors, for use in robotic systems; or even questions to a human, for use in medical systems. Decision theory could help choose good branch-and-bound strategies for global optimization, for use in high real-time hybrid controllers. Distributed numerical optimization techniques could be used for sensor fusion and model building applications. Decision-theoretic strategies could improve the speed of automated planning and scheduling software in logistics systems. They could be used in planning for visual inspection tasks on several scales, from single-robot spacecraft inspection to surveillance with massive swarms of UAVs.

IV. RECOMMENDATIONS

Decision theoretic optimization offers a wealth of research opportunities for future research in the context of CPS. Theoretical contributions could be made in addressing parallelism in the decision theoretic framework: the sequential decision theoretic model outlined above should be extended to handle the highly distributed systems that are characteristic of CPS. Theoretical contributions could also be made in efficient POMDP approximation, which is already an active area of AI research, e.g., (Kurniawati, Hsu, and Lee, 2008).

New decision theoretic programming paradigms would greatly facilitate the development and optimization CPS problem solvers. Ideally, the machinery would be hidden to the programmer, much like traditional compiler optimizations. A POMDP representation of an algorithm could be automatically generated from code. This would be facilitated by a comprehensive probabilistic inference framework for discrete and continuous variables. Background knowledge could be generated by automated testing and machine learning. Additionally, as an alternative to developing general-purpose POMDP solvers, a programming environment might contain a library of frequently encountered model problems which are applied to new computer code.

REFERENCES

D.A. Berry and B. Fristedt. *Bandit Problems.* Chapman and Hall, 1985.

B. Burns and O. Brock. *Single-Query Motion Planning with Utility-Guided Random Trees*. In Proc. IEEE Int. Conf. Rob. Aut., 2007.

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.

R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. *Finding optimal satisficing strategies for and-or trees.* In Artificial Intelligence, vol 170, pp. 19-58, 2006.

K. Hauser. *Motion Planning For Legged and Humanoid Robots.* Ph.D. Thesis, Stanford University, 2008.

D. Hsu, G. Sánchez-Ante and Z. Sun. *Hybrid PRM sampling with a cost-sensitive adaptive strategy.* In Proc. IEEE Int. Conf. Rob. Aut., pp 3885-3891, 2005.

L.E. Kavraki, P. Svetska, J.-C. Latombe, and M. Overmars. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*. In IEEE Int. J. of Robotics and Automation, vol. 12(4), pp. 556-580, 1996.

H. Kurniawati, D. Hsu, and W.S. Lee. *SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces*. In Proc. Robotics: Science and Systems, 2008.

O. Madani, S. Hanks, and A. Condon. *On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems.* In AAAI, pp. 541-548, 1999.

G. Sánchez and J.-C. Latombe. *On Delaying Collision Checking in PRM Planning: Application to Multi-Robot Coordination*. In IEEE Int. J. of Robotics. Res, vol. 21(1), pp 5-26, 2002.