Reinforcement Planning: Planners as Policies

Matt Zucker and J. Andrew Bagnell
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
mzucker@cs.cmu.edu, dbagnell@ri.cmu.edu

Introduction. State-of-the-art robotic systems [1, 2, 3] increasingly rely on search-based planning or optimal control methods to guide decision making. Similar observations can be made about computer game engines. Such methods are nearly always extremely crude approximations to the reality encountered by the robot: they consider a simplified model of the robot (as a point, or a "flying brick"), they often model the world deterministically, and they nearly always optimize a *surrogate* cost function chosen to induce the correct behavior rather than the "true" reward function corresponding to a declarative task description. Such approximations are made as they enable efficient, real-time solutions.

Despite this crudeness, optimal control methods have proven quite valuable because of the ability to transfer knowledge to new domains; given a way to map features of the world to a cost function, we can compute a plan that navigates a robot in a never-before-visited part of the world. While value-function methods and reactive policies are popular in the reinforcement learning community, it often proves remarkably difficult to transfer the ability to solve a particular problem to related ones using such methods [4]. Planning methods, by contrast, consider a sequence of decisions in the future, and rely on the principle underlying optimal control that cost functions are more parsimonious and generalizable than plans or values.

However, planners are only successful to the extent that they can transfer domain knowledge to novel situations. Most of the human effort involved in getting systems to work with planners stems from the tedious and error-prone task of adjusting surrogate cost functions, which has until recently been a black art. Imitation learning by Inverse Optimal Control, using, *e.g.* the Learning to Search approach [3], has proven to be an effective method for automating this adjustment; however, it is limited by human expertise. Our approach, Reinforcement Planning (RP), is straightforward: we demonstrate that crude planning algorithms can be learned as part of a direct policy search or value function approximator, thereby allowing a system to experiment on its own and enabling outperforming human expert demonstration.

It is important in this work to distinguish between the "true" reward function $r(x)$, true state (observation) space $\mathcal{X}$, and control space $\mathcal{A}$ defined by the problem domain and the surrogate states and costs used to generate plans, $\mathcal{S}$ and $c(s)$. We rely on implicit map that can take any x to a corresponding s; this function should be thought of as a many-to-one coarse-graining. We note that planning algorithms have been used extensively in previous RL work (*e.g.* Dyna [5]); our work contrasts with these in embracing the reality that real-world planners operate on a coarse approximation and must be trained to have a cost-function that induces the correct behavior, not the "true" cost function.

Value-Function Subgradient. To enable RP, we rely on the fact that the value of a state s has a subgradient with respect to the parameters of the planner's cost function. The state-action cost for an optimal planner is given by a function $c(s, a, \theta)$ that takes as input a state-action pair as well as a parameter vector θ. Then the optimal value $V(s, \theta)$ of a state is given by $V(s, \theta) = \min_{\xi(s)} \sum_{i \in \xi(s)} c(s_i, a_i, \theta)$ where $\xi(s)$ is a trajectory of state-action pairs. Then a sub-gradient of $V(s, \theta)$ with respect to the cost parameters θ is simply the gradient at the minimizer: $\nabla_\theta V(s, \theta) = \sum_{i \in \xi^*(s)} \nabla_\theta \, c(s_i, a_i, \theta)$. This result enables us to extend any gradient-based RL algorithms to leverage optimal control methods.

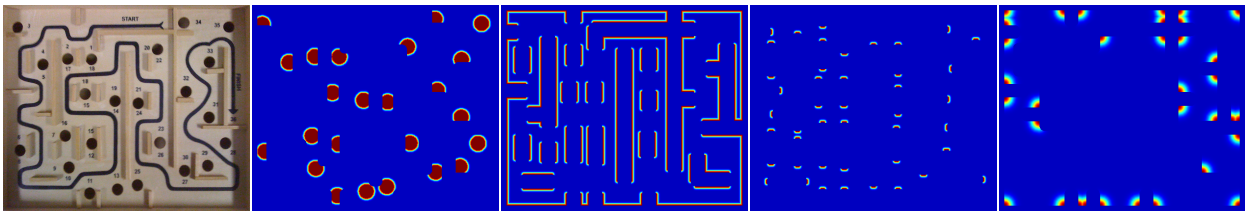

Figure 1: Marble maze and features used for planning. Left to right: holes, walls, wall ends, corners.

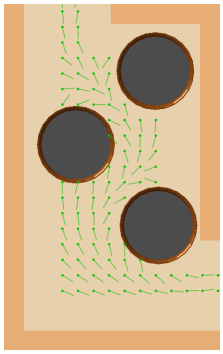

Figure 2: Local gradient direction of planner value function, used to compute $a^*(x, \theta)$.

Example: Marble Maze. The Marble Maze toy has been used as an example domain for imitation and reinforcement learning in the past [4]. However, previous approaches had limited ability to transfer due to their choice of parameterization. In our approach, we use an optimal planner to compute a 2D value function for the board. The planner uses only local features of the board, such as distances to holes, walls, and corners, as shown in figure 1. The true state of the marble maze is given by the ball position, ball velocity, and board tilt: $x = (u, v, \dot{u}, \dot{v}, t_u, t_v)$, and the action is the commanded tilt: $a = (a_u, a_v)$. The cost function for the planner is simply a linear combination of features at the 2D position of the ball $c(s, a, \theta) = \theta^T f(u, v)$. We use a Gibbs policy over controls:

$$p(a_j|x, \theta) \propto \exp\left[\left(k_a \|a_j - a^*(x, \theta)\|^2 + k_v V(s', \theta) \right)\right]$$

where $a^*(x, \theta)$ is the commanded tilt that aligns the ball's velocity with the local gradient direction of the value function (as shown in figure 2), and where $V(s', \theta)$ is the computed value for the successor state given by the 2D planner, and k_a, k_v are weighting parameters.

Results. We use REINFORCE [6] to learn the parameters θ for the cost function in a marble maze simulator. The simulator provides a noisy, high-latency (200ms) estimate of ball position and board tilt, and control bandwidth is low (10Hz), to reflect real-world operating conditions on the true marble maze. We compared the performance of the planner against a kernelized policy learned using global features. Preliminary results indicate that the planner performs comparably, but unlike the kernelized policy, generalizes to previously unseen boards. We will present our full results, including on a physical Marble Maze, at the workshop.

References

[1] J. Chestnutt. *Navigation Planning for Legged Robots*. PhD thesis, RI, Carnegie Mellon University, 2007.

[2] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *JFR*, 2008.

[3] N. Ratliff, D. Silver, and J. A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 2009.

[4] M. Stolle. *Finding and Transferring Policies Using Stored Behaviors*. PhD thesis, RI, Carnegie Mellon University.

[5] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

[6] J. Baxter and P.L. Bartlett. Infinite-horizon policy-gradient estimation. *JAIR*, 2001.

Topic: control/reinforcement learning, Preference: Oral, Presenting Author: J. A. Bagnell