

Planning in Continuous Domains with MDPs

Neil Bacon, Supervisor: Scott Sanner

August 19, 2008

1 Planning with MDPs

Markov Decision Processes are typically applied to finite discrete domains. This project focuses on combining some recently proposed techniques to apply them to AI planning problems in continuous domains. A Markov Decision Process

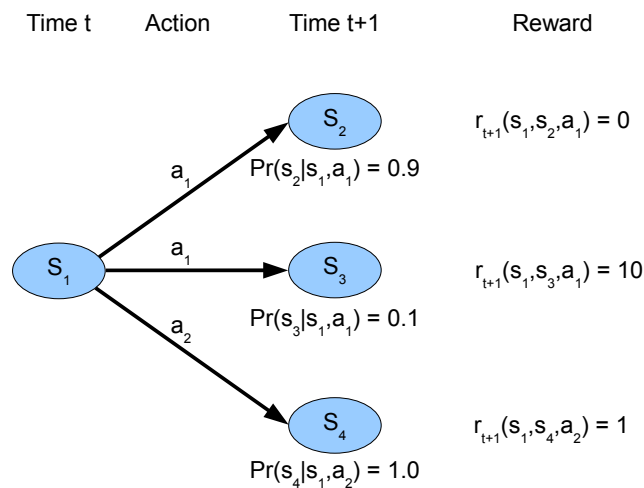


Figure 1: Markov Decision Process

(MDP) is characterised by

$$M = \langle S, A, \mathcal{P}_{s,s'}^a, \mathcal{R}_{s,s'}^a \rangle$$

where

- S is the state space,
- A is the action space,
- $\mathcal{P}_{s,s'}^a = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the stochastic Markov state transition model and
- $\mathcal{R}_{s,s'}^a = \mathbb{E}(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$ is the expected value of the next reward.

The transition probabilities satisfy the Markov property: that is that given the current state and action, the next state is conditionally independent of all earlier states.

A policy $\pi(s)$ is a function from states to actions.

An optimal policy $\pi^*(s)$ is a policy that maximises some definition of long term reward, e.g. sum of discounted future rewards.

Many AI planning problems can be formulated as a MDP [3].

The goal of a MDP planning problem is to find the optimal policy $\pi^*(s)$.

2 Project Goal

Solve MDP planning problems with continuous state and action spaces and continuous transition models.

Solve using value iteration on Extended Algebraic Decision Diagrams (ADDs).

Simplifying assumption: state variables conditionally independent given previous state.

Apply to problems in the traffic controller domain and if time permits the Mars rover domain.

3 Using Algebraic Decision Diagrams

A Binary Decision Diagram is a tree representing a function $\mathbb{B}^n \rightarrow \mathbb{B}$. See Figure 2.

An Algebraic Decision Diagram is obtained by replacing the Boolean values in the terminal nodes of a Binary Decision Diagram with real values. An Algebraic Decision Diagram is a tree representing a function $\mathbb{B}^n \rightarrow \mathbb{R}$ [1]. See Figure 3.

We extend Algebraic Decision Diagrams by adding two new types of nodes.

- non-terminal nodes that perform inequality tests on Real valued parameters; and
- terminal nodes that are polynomial functions of Real valued parameters.

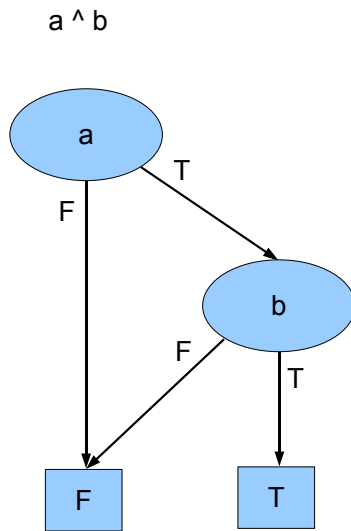


Figure 2: Binary Decision Diagram

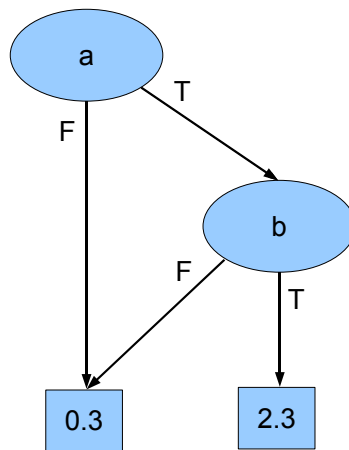


Figure 3: Algebraic Decision Diagram

Extended Algebraic Decision Diagrams represent a useful special case of piece-wise polynomial functions, where the pieces are restricted to hyper-rectangles. They provide a compact representation that is amenable to computation. An Extended Algebraic Decision Diagram is a tree representing a function $\langle \mathbb{B}^n, \mathbb{R}^m \rangle \rightarrow \mathbb{R}$. See Figure 4.

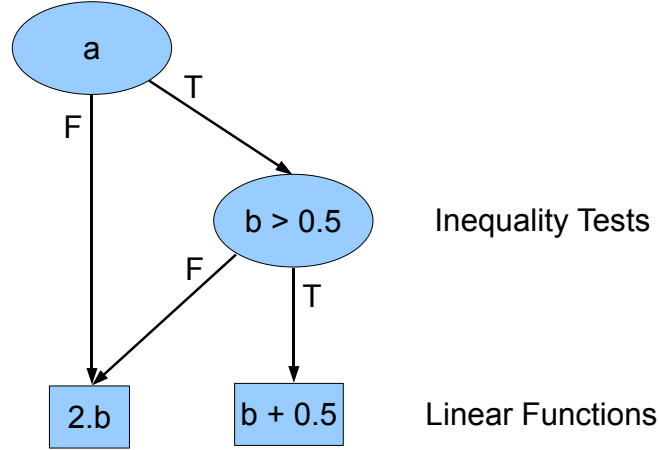


Figure 4: Extended Algebraic Decision Diagram

We perform value iteration on Extended Algebraic Decision Diagrams (EADD) to find the optimal value function. At each iteration:

- the next value function and policy are computed exactly from current EADDs;
- computation involves a convolution - result has higher polynomial order [2];
- limit the polynomial order by approximating the result with a piece-wise polynomial function of lower order (e.g. piece-wise linear) on a hyper-rectangular partition, represented by an EADD.

4 Traffic Control Domain

The initial application domain is a traffic controller for a single intersection.

The intersection model will be kept as simple as possible. The state model $\langle s, \vec{q} \rangle$ consists of

- the cycle state $s \in \{s_1, \dots, s_m\}$ determines which directions get a green light and which get a red,
- the number of cars in each of n lane queues $\vec{q} = \langle q_1, \dots, q_n \rangle$ where each $q_i \in [0, \infty]$ (assume fully observable).

Time permitting, the model can be extended to include:

- multiple intersections (including simple models of traffic as they travel between intersections),
- pedestrian push buttons,
- skipping a cycle if the queue is short in order to clear a longer queue on another road (as long as fairness is ensured).

5 Control Model

We assume the only control action at any stage in the cycle is how long to remain in that cycle stage. Thus, we model the action as a continuous waiting time t on the interval $t \in [MinCycleTime, MaxCycleTime]$.

6 Traffic Model

The traffic model updates the queue lengths at each stage as the difference between an *arrival model* and a *clearance model*.

The proposed arrival model is Poisson, with the probability of k cars arriving in the interval t given by:

$$\Pr(k|t) = \frac{(\lambda \frac{t}{l})^k e^{-\lambda \frac{t}{l}}}{k!}$$

where λ is the mean number of arrivals in the interval l .

The proposed clearance model for the queue whose light is green is represented as the following piecewise linear function of the action duration:

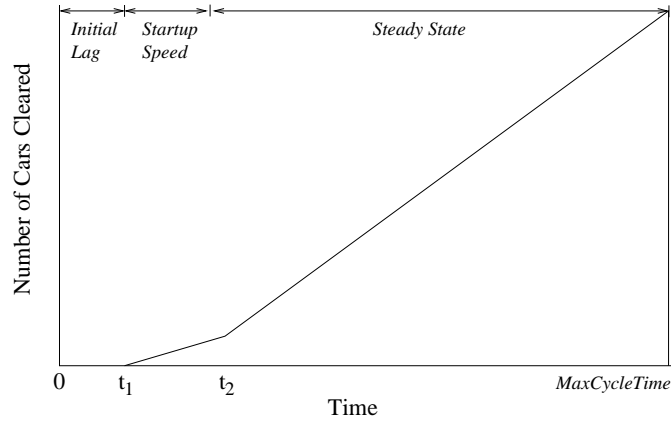


Figure 5: Clearance Model

7 Reward and Objective

The immediate reward is negative sum of the queue lengths:

$$r(s, \vec{q}) = \sum_i -q_i$$

and the discounted long term reward that we seek to maximise, is defined at time step t as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where $0 < \gamma < 1$ is the discount rate.

8 Schedule

Duration	Due	Description
2w	12/Aug	Reading, task definition, initial presentation
1w	19/Aug	Coding Extended Algebraic Decision Diagrams
5w	26/Sep	Reading and coding: <ul style="list-style-type: none">- EADD operators- problem instance parser (extended SPUDD format)- value iteration
2w	10/Oct	Perform experiments for Report
1w	17/Oct	Report
1w	24/Oct	Presentation

References

- [1] Jesse Hoey, Robert St-aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic Planning Using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279-288. Morgan Kaufmann, 1999.
- [2] Lihong Li and Michael L. Littman. Lazy approximation for solving continuous finite-horizon MDPs. In *National Conference on Artificial Intelligence*, 2005.
- [3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.