

This chapter presents a brief summary of the TEXPLORE algorithm before fully describing and presenting the real time RL architecture. First, I present a typical example of a sequential model-based RL architecture. Then I present details on using Monte Carlo Tree Search for planning, including a description of the modified version of the UCT algorithm (Kocsis and Szepesvári, 2006) that we use for planning. In Section 3.3, I present the parallel architecture for real time action, which puts model learning, planning, and acting on three parallel threads, such that actions can be taken as fast as required without being constrained by how long model updates or planning takes. Finally, I summarize the chapter in Section 3.3.

In this book, I introduce TEXPLORE, a sample-efficient model-based real time RL algorithm. When learning on robots, agents typically have very few samples to learn since the samples may be expensive, dangerous, or time-consuming. Therefore, learning algorithms for robots must be greedier than typical methods to exploit their knowledge in the limited time they are given. Since these algorithms must perform limited exploration, their exploration must be efficient and target state-actions that may be promising for the final policy. TEXPLORE achieves high sample efficiency by 1) utilizing the generalization properties of decision trees in building its model of the MDP, and 2) using random forests of those tree models to limit exploration to states that are promising for learning a good (but not necessarily optimal) policy quickly, instead of exploring more exhaustively to guarantee optimality. These two components constitute the key insights of the algorithm, and are explained in Chapter 4. Modifications to the basic decision tree model enable TEXPLORE to operate in domains with continuous state spaces as well as domains with action or observation delays.

The other key feature of the algorithm is that it can act in real time, at the frequencies required by robots (typically 5 - 20 Hz). For example, an RL agent controlling an autonomous vehicle must provide control signals to the gas and brake pedals immediately when a car in front of it slams on its brakes; it cannot stop to “think” about what to do. An alternative approach for acting in real time would be to learn off-line and then follow the learned policy in real time after the fact. However, it is desirable for the agent to be capable of learning on-line in-situ for the lifetime of the robot, adapting to new states and situations without pauses for computation. TEXPLORE combines a multi-threaded architecture with Monte
In this chapter, I introduce TEXPLORER’s parallel architecture, enabling it to return actions in real time, addressing Challenge 4 of the RL for Robotics Challenges. Most current model-based RL methods use a sequential architecture such as the one shown in Figure 2.3 in Chapter 2. Pseudo-code for the sequential architecture is shown in Algorithm 3.1. In this sequential architecture, the agent receives a new state and reward; updates its model with the new transition \(\langle s, a, s', r \rangle\) (i.e. by updating a tabular model or adding a new training example to a supervised learner); plans exactly on the updated model (i.e. by computing the optimal policy with a method such as value iteration (Sutton and Barto, 1998) or prioritized sweeping (Moore and Atkeson, 1993)); and returns an action from its policy. Since both the model learning and planning can take significant time, this algorithm is not real time. Alternatively, the agent may operate in batch mode (updating its model and planning on batches of experiences at a time), but this approach requires long pauses for the batch updates to be performed. Making the algorithm real time requires two modifications to the standard sequential architecture: 1) utilizing sample-based approximate planning (presented in Section 3.1) and 2) developing a novel parallel architecture (presented in Section 3.2). I later evaluate this planning method and parallel architecture in comparison with other approaches in Section 5.4.

### 3.1 Monte Carlo Tree Search (MCTS) Planning

The first component for providing actions in real time is to use an anytime algorithm for approximate planning, rather than performing exact planning using a method such as value iteration or prioritized sweeping. This section describes TEXPLORER’s use of UCT for approximate planning as well as the modifications we have made to the algorithm. The standard UCT algorithm was presented in Section 2.2.4 but here we have modified UCT to use \(\lambda\)-returns, generalize...