Establishing Multiscale Anticipatory Behavior by Hierarchical Reinforcement Learning

Matthias Rungger, Hao Ding, and Olaf Stursberg

Institute of Automatic Control Engineering
Technische Universität München
Theresienstr. 90, D-80290 Munich, Germany
{matthias.rungger;hao.ding;stursberg}@tum.de

Abstract. In order to establish autonomous behavior for technical systems, the well known trade-off between reactive control and deliberative planning has to be considered. Within this paper, a two level hierarchical reinforcement learning scheme is proposed to orchestrate behaviors represented by hybrid dynamic systems, which combine continuous and discrete behavior. Reinforcement learning (RL) with model-based value function is used on the higher level based on a MDP abstracted from the hybrid system. On the lower level, policy gradient-based RL considers the hybrid dynamics.

1 Introduction

Several approaches for anticipatory behavior of learning systems have been developed in recent years and are described e.g. in [3, 4]. Reinforcement learning (RL) is one of the main approaches to establish anticipatory behavior in adaptive learning systems. RL uses an estimate on the future outcome of the systems actions, to guide its behavior. Depending on the model on which RL is used, it might result in reactive control or deliberative planning. Reactive control is often referred as bottom-up approach, which quickly reacts to changing environment, in contrast, deliberative planning, the top-down approach, plans sequences of actions on an slowly changing structured environment. Inspired by animal or human behavior, which surely uses both approaches, the coupling between these two approaches to behavior control for autonomous systems has to be found. In [2] and [7] hybrid automata are proposed as trade-off solution, bridging the gap between reactive and deliberative control. Hybrid systems consist of collections of continuous dynamics combined with discrete events. This allows to switch among basic behaviors, modelled by continuous dynamics, through discrete controls and thus to build complex behavior in order to achieve a certain goal autonomously.

Based on HA, a two level hierarchical RL (HRL [6]) scheme is proposed in this paper to calculate the controls. The scheme uses model-based value iteration to obtain state anticipations on the higher level, i.e. the model is used to learn the value function and the reward is adapted from experiences. Policy gradient

based RL (PGRL) as payoff anticipations on the lower level is used such that the problem of discretization of the continuous spaces of HA can be avoided [1]. Using this combination, the benefit of continuous PGRL, as it was demonstrated in several impressive examples [8], is here extended to hybrid systems.

2 Hybrid Automaton and its Abstraction

As introduced earlier, the use of hybrid dynamic models is appropriate to model qualitatively different behaviors. The system is modeled by a hybrid automaton $HA = (X, U, Z, inv, \Theta, g, r, f)$ according to [9]:

- $X \subseteq \mathbb{R}^{n_x}$ specifies the continuous state space, on which the *state vector* x is defined;
- $U \subseteq \mathbb{R}^{n_u}$ is the continuous input space of dimension n_u;
- the finite set of discrete states is denoted by $Z = \{z_1, \cdots, z_{n_z}\}$;
- a mapping *inv*: $Z \to 2^X$ assigns an invariant for x to each location $z_j \in Z$;
- the set of *transitions* is given by $\Theta \subseteq Z \times Z$;
- a mapping g: $\Theta \to 2^X$ associates a transition *guard* $g((z_1, z_2)) \subseteq X$ with each transition $(z_1, z_2) \in \Theta$;
- a reset function $j : \Theta \times X \to X$ assigns an updated state $x' \in X$ to each $(z_1, z_2) \in \Theta$, $x \in g((z_1, z_2))$;
- $f_z : Z \times X \times U \to \mathbb{R}^{n_x}$ defines a flow function of the form: $\dot{x} = f_z(x, u)$ for each location $z \in Z$, x and u denote the state and input respectively.

Such a dynamic allows, e.g., to model the continuous motion of a robot in different modes (transport of objects, gripping of objects, etc.) characterized by different discrete states z.

Abstraction: The HA can be abstracted into a Markov Decision Process (MDP) with the form: $M = (S, A, P, r)$, where S is a finite set of states corresponding to the set of discrete locations of the hybrid automaton. Two states $s_0 \in S$ and $s_g \in S$ specify an initial and goal state respectively. For every transition in Θ, an action $a \in A$ is defined, which triggers the transition from state s to s' – it thus represents the set of all trajectories $x(t)$ within the location z (represented by s), which end up in the guard set g that lead to the location z' (corresponding to s') by the transition $(z, z') \in \Theta$. The transition function of M is given by $P : S \times A \times S \to \{0, 1\}$ and $r : S \times A \times S \to \mathbb{R}$ is the reward function associated to a transition of M. A MDP is capable to model uncertainties in the outcome of the actions, but here only deterministic transitions are regarded and transition probabilities will be subject of future work.

3 Hierarchical Reinforcement Learning

RL is learning from interaction, i.e. the system learns a strategy (as a state-to-action mapping) based on the reward obtained for a past execution. At time t, the system observes the state $s \in S$, and for a policy $a = \pi(s)$, an action

$a \in A$ is chosen. The system moves from state s to s' according to the transition function, and it receives the reward r. The goal is to learn a policy π, which maximizes the expected cumulative future reward $R = E\left[\sum_{i=0}^{N-1} r(s_i, a, s_{i+1})\right]$ (N: number of steps to reach the goal state). The anticipation is based on the current immediate reward and the value function from previous experience – this principle is underlying a large number of RL approaches in different variations.

The proposed method within this work uses a hierarchical approach to solve the problem on two different levels. On the higher level, the abstracted MDP M from the hybrid automata is used to apply a value iteration where the value function for each state is updated according to:

$$V(s) = \max_{a \in A} \sum_{s'} P(s'|s,a)\left[r(s,a,s') + V(s')\right]$$

after each iteration, where the reward is:

$$r(s,a,s') = \begin{cases} 1 & \text{if } s' = s_g \\ 0 & \text{otherwise.} \end{cases}$$

After the convergence of the algorithm for computing $V(s)$, the goal leading abstract sequence of discrete states is determined by following a greedy policy. For each discrete transition (z, z') within this sequence, a continuous parameterized value function $V_z(x(t), z', w)$ is learned, which guides the system from an arbitrary continuous state $x_0 \in inv(z)$ to the corresponding guard set $g((z, z'))$ of the consecutive discrete state. To learn the parameters w of the continuous value function, the RL algorithm from [5] is applied for a maximum number of N trials. The number n of reaching the desired guard set is used to inform the higher level about the performance of the RL on the continuous level. The reward $r(s, a, s')$ is reduced after the completion of the N trials by

$$r(s,a,s') = r(s,a,s') - \frac{c}{n+1},$$

and thus the model of the environment is updated. The parameter $c > 0$ is used tune the information propagation from the lower level to the higher. By using this reward update method an exploratory behavior is realized, which favors 'easy-to-learn' locations within the HA. A location where the desired guard set is reached many times during learning, is considered as 'easy-to-learn' here. The pseudo code of the describe scheme is listed in Algorithm 1.

4 Discussion

This approach focuses on the importance of an appropriate model within the different anticipatory behaviors. Even in model-free anticipatory behavior a sound model of the system to be controlled (agent) is essential. In [8], off-line learning, based on system models is used in order to initialize the value functions for the real systems. HA are proposed as a general and adequate model for describing

Algorithm 1 Hierarchical Reinforcement Learning (HRL) Algorithm.

initialize $V(s) = 0$ and $V_z(x, s', w)$
while 1 **do**
  do value iteration
  while $s \notin s_g$ **do**
    $s' \leftarrow \arg\max_{a \in A} [r(s, a, s') + \gamma P(s'|s, a)V(s')]$
    for N *trials* **do**
      do continuous RL within location $z = s$
    end for
    $r(s, a, s') \leftarrow r(s, a, s') - \frac{c}{n+1}$
    $s \leftarrow s'$
  end while
end while

complex systems. A two level hierarchical RL algorithm is introduced, which guides the system by an exploratory behavior through the state space. Omitting the continuous dynamics, a MDP is abstracted from the HA, in order to facilitate a long-term anticipation of the future reward. The computed sequence of abstracted states from this long-term anticipation is used to guide the system on the lower level. Due to space limitations the introduction of the simulation of a simple manufacturing system, consisting of an conveyor-belt and a scara robot is omitted here and will be presented along with performance measures in the post-proceedings.

References

1. A. J. Blanchard and L. Caamero. Anticipating Rewards in Continuous Time and Space: A Case Study in Developmental Robotics. In *Anticipatory Behavior in Adaptive Learning Systems*, LNCS 4520, pages 267–284. Springer-Verlag, 2007.
2. M.S. Branicky. Behavioral programming. In *Working Notes AAAI Spring Symp. on Hybrid Systems and AI*, 1999.
3. M.V. Butz, O. Sigaud, and P. Gerard. Anticipatory Behavior: Exploiting Knowledge about the Future to Improve Current Behavior. In *Anticipatory Behavior in Adaptive Learning Systems*, LNCS 2684, pages 69–86. Springer-Verlag, 2003.
4. M.V. Butz, O. Sigaud, and P. Gerard. Internal Models and Anticipations in Adaptive Learning Systems. In M.V. Butz, O. Sigaud, and P. Gerard, editors, *Anticipatory Behavior in Adaptive Learning Systems*, LNCS 2684, pages 253–273. Springer-Verlag, 2003.
5. K. Doya. Reinforcement learning in continuous time and space. *Neural Comput.*, 12(1):219–245, 2000.
6. M. Ghavamzadeh. *Hierarchical Reinforcement Learning in Continuous State and Multi-Agent Environments*. PhD thesis, University of Massachusetts Amherst, 2005.
7. Tejas R. Mehta and Magnus Egerstedt. Multi-modal control using adaptive motion description languages. *Automatica, In Press*, 2008.
8. J. Morimoto and K. Doya. Acquisition of stand-up behavior by a real robot using hierarchical RL. *Robotics and Autonomous Systems*, 36(1):37–51, 2001.
9. O. Stursberg. Supervisory control of hybrid systems based on model abstraction and refinement. *Journal on Nonlinear Analysis - Hybrid Systems and Applications*, 65(6):1168–1187, 2006.