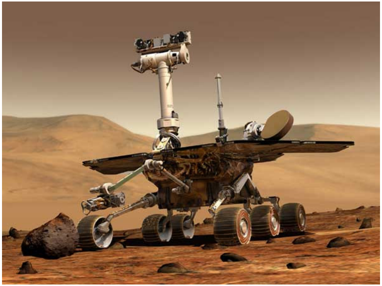
TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots

Todd Hester and Peter Stone

Learning Agents Research Group
Department of Computer Science
The University of Texas at Austin

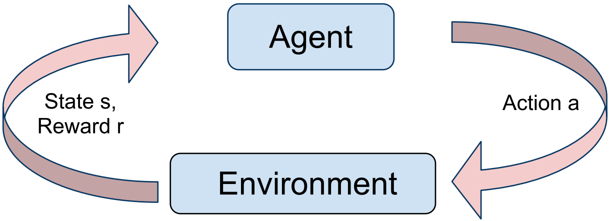
Journal Track: appeared in **Machine Learning**, 2013

Robot Learning



- Robots have the potential to solve many problems
- We need methods for them to learn and adapt to new situations

Reinforcement Learning



- Value function RL has string of positive theoretical results [Watkins 1989, Brafman and Tennenholtz 2001]
- Could be used for learning and adaptation on robots

Reinforcement Learning

Model-free Methods

- Learn a value function directly from interaction with environment
- Can run in real-time, **but** not very sample efficient

Model-based Methods

- Learn model of transition and reward dynamics
- Update value function using model (planning)
- Can update action-values without taking real actions in the world

Velocity Control of an Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- 10 second episodes (at 20 Hz: 200 samples / episode)

Velocity Control

- State:
  - Current Velocity
  - Desired Velocity
  - Accelerator Pedal Position
  - Brake Pedal Position
- Actions:
  - Do nothing
  - Increase/decrease brake position by 0.1
  - Increase/decrease accelerator position by 0.1
- Reward: -10.0 * velocity error (m/s)

Desiderata

1. Learning algorithm must learn in very few actions (be **sample efficient**)
2. Learning algorithm must take actions **continually** in real-time (while learning)
3. Learning algorithm must handle **continuous** state
4. Learning algorithm must handle **delayed** actions

Desiderata

1. Learning algorithm must learn in very few actions (be **sample efficient**)
2. Learning algorithm must take actions **continually** in real-time (while learning)
3. Learning algorithm must handle **continuous** state
4. Learning algorithm must handle **delayed** actions

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-Max	Brafman 2001	Yes	No	No	No
Q-Learning	Watkins 1989	No	Yes	No	No
with F.A.	Sutton & Barto 1998	No	Yes	Yes	No
SARSA	Rummery & Niranjan 1994	No	Yes	No	No
GPRL	Deisenroth & Rasmussen 2011	Yes	No	Yes	No
BOSS	Asmuth et al 2009	Yes	No	No	No
Bayesian DP	Strens 2000	Yes	No	No	No
MBBE	Dearden et al 1999	Yes	No	No	No
MBS	Walsh et al 2009	Yes	No	No	Yes
Dyna	Sutton 1990	No	Yes	No	No

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-Max	Brafman 2001	Yes	No	No	No
Q-Learning	Watkins 1989	No	Yes	No	No
with F.A.	Sutton & Barto 1998	No	Yes	Yes	No
SARSA	Rummery & Niranjan 1994	No	Yes	No	No
GPRL	Deisenroth & Rasmussen 2011	Yes	No	Yes	No
BOSS	Asmuth et al 2009	Yes	No	No	No
Bayesian DP	Strens 2000	Yes	No	No	No
MBBE	Dearden et al 1999	Yes	No	No	No
MBS	Walsh et al 2009	Yes	No	No	Yes
Dyna	Sutton 1990	No	Yes	No	No

The TEXPLORE Algorithm

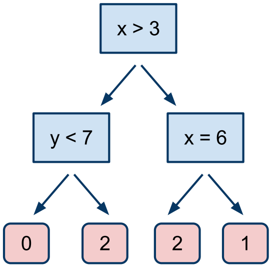
1. Limits exploration to be **sample efficient**
2. Selects actions continually in **real-time**
3. Handles **continuous** state
4. Handles actuator **delays**

Available publicly as a ROS package:

www.ros.org/wiki/rl-texplore-ros-pkg

Challenge 1: Sample Efficiency

- Treat model learning as a supervised learning problem
  - **Input:** State and Action
  - **Output:** Distribution over next states and reward
- **Factored** model: Learn a separate model to predict each next state feature and reward
- **Decision Trees**: Split state space into regions with similar dynamics



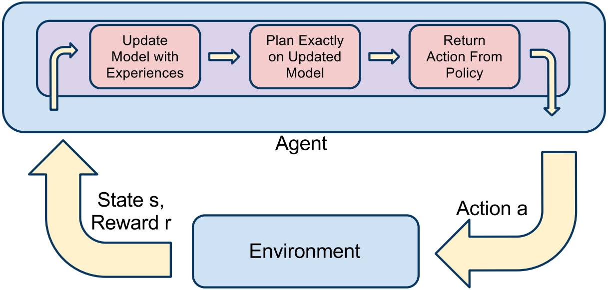
Random Forest Model [ICDL 2010]

- Average predictions of *m* different decision trees
- Each tree represents a **hypothesis** of the true dynamics of the domain
- Acting greedily w.r.t. the average model balances predictions of optimistic and pessimistic models
- **Limits** the agent's exploration to state-actions that appear promising, while avoiding those which may have negative outcomes

Random Forest Model [ICDL 2010]

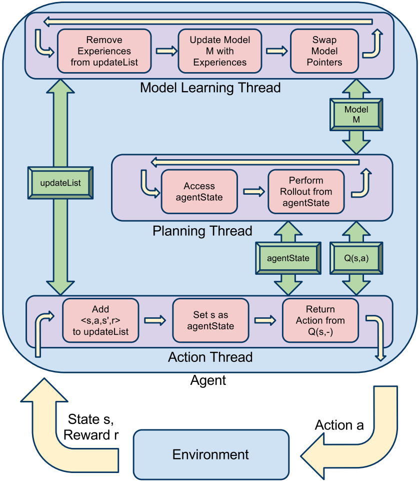
- Average predictions of *m* different decision trees
- Each tree represents a **hypothesis** of the true dynamics of the domain
- Acting greedily w.r.t. the average model balances predictions of optimistic and pessimistic models
- **Limits** the agent's exploration to state-actions that appear promising, while avoiding those which may have negative outcomes

Challenge 2: Real-Time Action Selection



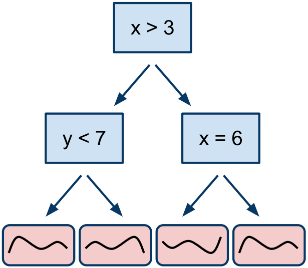
- Model update can take too long
- Planning can take too long

Real-Time Model Based Architecture (RTMBA)



- Model learning and planning on parallel threads
- Action selection **is not restricted** by their computation time
- Use sample-based planning (anytime)
- Mutex locks on shared data

Challenge 3: Continuous State



- Use regression trees to model continuous state
- Each tree has a linear regression model at its leaves
- Discretize state space for value updates from UCT, but still plan over continuously valued states

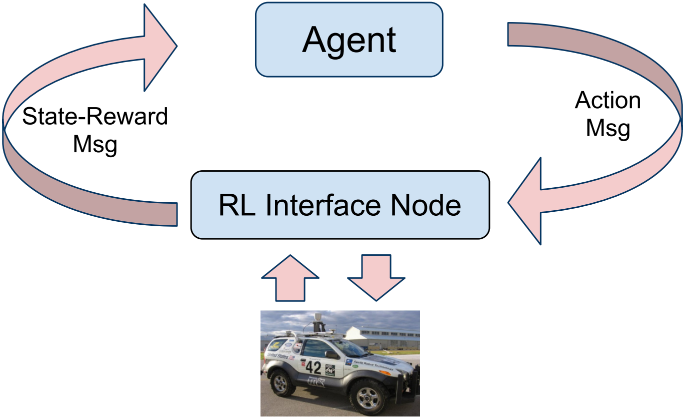
Challenge 4: Actuator Delays

- Delays make domain non-Markov, but k-Markov
- Provide model with previous *k* actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant
- UCT can plan over augmented state-action histories easily
- Would not be as easy with tabular models or dynamic programming

Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- Vehicle runs at 20 Hz.
- Agent **must** provide commands at this frequency.

Uses ROS [Quigley et al 2009]



- http://www.ros.org/wiki/rl_msgs

Simulation Experiments

Exploration Approaches

- Epsilon-Greedy
- Boltzmann Exploration
- Use merged BOSS-like model
- Use random model each episode

Sample Efficient Methods

- BOSS [Asmuth et al 2009]
- Bayesian DP [Strens 2000]
- Gaussian Process RL [Deisenroth & Rasmussen 2011]

Simulation Experiments

Continuous Models

- Tabular Models
- Gaussian Process RL [Deisenroth & Rasmussen 2011]
- KWIK linear regression [Strehl & Littman 2007]

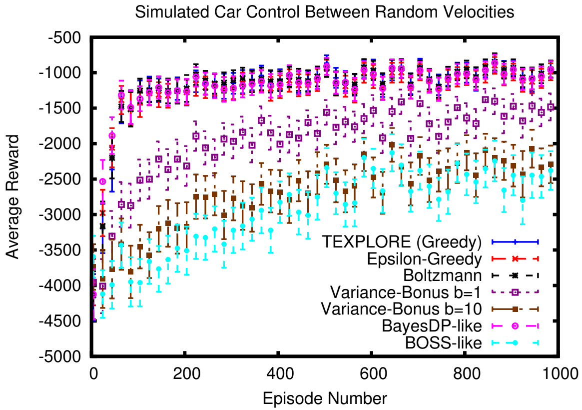
Real-Time Architectures

- Real Time Dynamic Programming [Barto et al 1995]
- Dyna [Sutton 1990]
- Parallel Value Iteration

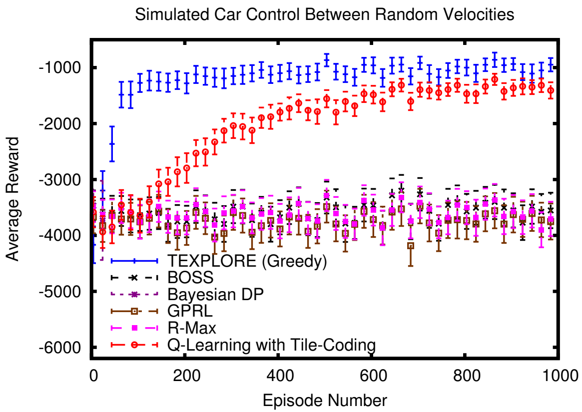
Actuator Delays

- Model Based Simulation [Walsh et al 2009]

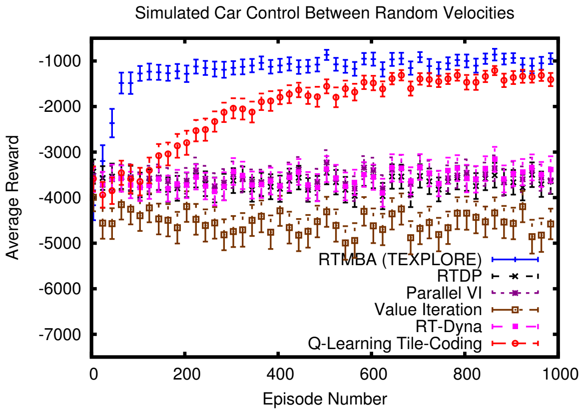
Challenge 1: Sample Efficiency



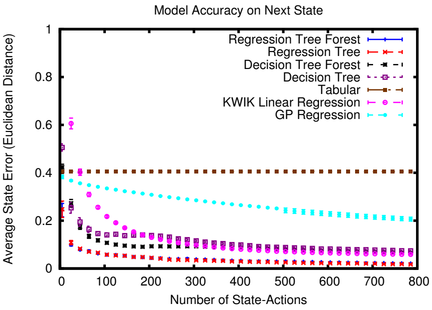
Challenge 1: Sample Efficiency



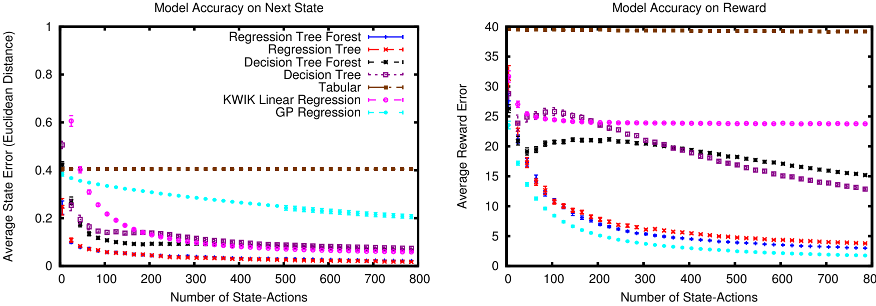
Challenge 2: Real-Time Action Selection



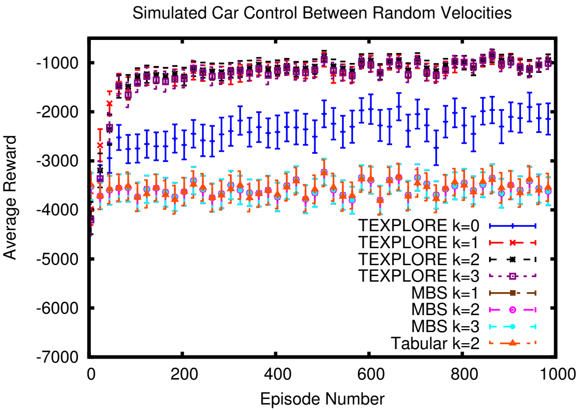
Challenge 3: Modeling Continuous Domains



Challenge 3: Modeling Continuous Domains



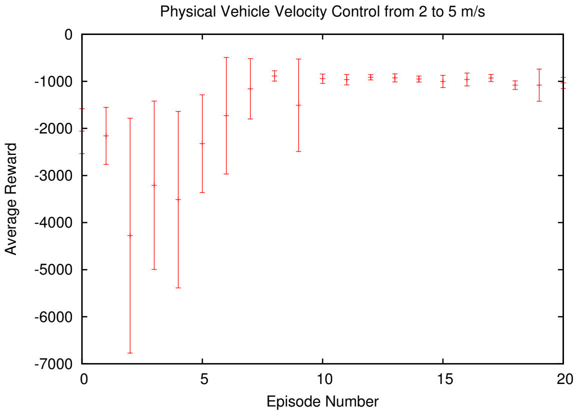
Challenge 4: Handling Delayed Actions



On the physical vehicle

- But, does it work on the actual vehicle?

On the physical vehicle



- Yes! It learns the task within 2 minutes of driving time

Conclusion

- TEXPLORE can:
  1. Learn in few **samples**
  2. Act continually in **real-time**
  3. Learn in **continuous** domains
  4. Handle actuator **delays**
- TEXPLORE code has been released as a ROS package: www.ros.org/wiki/rl-texplore-ros-pkg