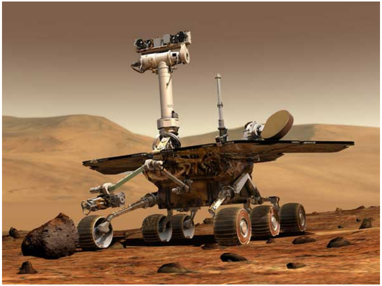
TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-Constrained Domains

Todd Hester

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

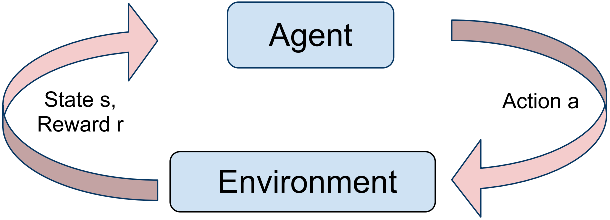
Thesis Defense
December 3, 2012

Robot Learning



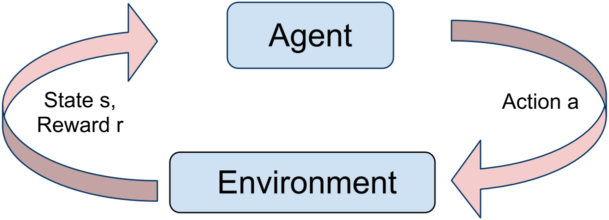
- Robots have the potential to solve many problems
- Moving from controlled to natural environments is difficult
- We need methods for them to learn and adapt to new situations

Reinforcement Learning



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

Reinforcement Learning



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

Velocity Control of an Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- 10 second episodes (at 10 Hz: 100 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 handle **continuous** state
3. Learning algorithm must handle **delayed** actions
4. Learning algorithm must take actions **continually** in real-time (while learning)

Desiderata

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

Desiderata

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

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-MAX	Brafman and Tennenholtz, 2001	Yes	**No**	**No**	**No**
Q-LEARNING	Watkins, 1989	**No**	Yes	**No**	**No**
with F.A.	Sutton and Barto, 1998	**No**	Yes	Yes	**No**
SARSA	Rummery and Niranjan, 1994	**No**	Yes	**No**	**No**
PILCO	Deisenroth and Rasmussen, 2011	Yes	**No**	Yes	**No**
NAC	Peters and Schaal 2008	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., 2009	Yes	**No**	**No**	**No**
SPITI	Degris et al., 2006	Yes	**No**	**No**	**No**
MBS	Walsh et al., 2009	Yes	**No**	**No**	Yes
U-TREE	McCallum, 1996	Yes	**No**	**No**	Yes
DYNA	Sutton, 1990	Yes	Yes	**No**	**No**
DYNA-2	Silver et al., 2008	Yes	Yes	Yes	**No**
KWIK-LR	Strehl and Littman, 2007	Yes	**No**	**Partial**	**No**
FITTED R-MAX	Jong and Stone, 2007	Yes	**No**	Yes	**No**
DRE	Nouri and Littman 2010	Yes	**No**	Yes	**No**

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-MAX	Brafman and Tennenholtz, 2001	Yes	**No**	**No**	**No**
Q-LEARNING	Watkins, 1989	**No**	Yes	**No**	**No**
with F.A.	Sutton and Barto, 1998	**No**	Yes	Yes	**No**
SARSA	Rummery and Niranjan, 1994	**No**	Yes	**No**	**No**
PILCO	Deisenroth and Rasmussen, 2011	Yes	**No**	Yes	**No**
NAC	Peters and Schaal 2008	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., 2009	Yes	**No**	**No**	**No**
SPITI	Degris et al., 2006	Yes	**No**	**No**	**No**
MBS	Walsh et al., 2009	Yes	**No**	**No**	Yes
U-TREE	McCallum, 1996	Yes	**No**	**No**	Yes
DYNA	Sutton, 1990	Yes	Yes	**No**	**No**
DYNA-2	Silver et al., 2008	Yes	Yes	Yes	**No**
KWIK-LR	Strehl and Littman, 2007	Yes	**No**	**Partial**	**No**
FITTED R-MAX	Jong and Stone, 2007	Yes	**No**	Yes	**No**
DRE	Nouri and Littman 2010	Yes	**No**	Yes	**No**
TEXPLORE	This thesis	Yes	Yes	Yes	Yes

Sample Complexity of Exploration

Definition: Number of sub-optimal actions the agent must take

- Lower bound is polynomial in N (# of states) and A (# of actions) [Kakade 2003]
- On a very large problem, NA actions is too many
- If actions are expensive, dangerous, or time-consuming, even a few thousand actions may be unacceptable
- What should we do when we do not have enough actions to guarantee convergence to an optimal policy?

Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

- Takes actions continually at specified frequency (not batch mode)
- Concerned with reward during learning (not just final policy)

Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

- Agent has a limited number of time steps
- Not enough time steps to learn optimal policy without some assumptions

Solution

Model-Based Method

- Learn transition and reward dynamics, then update value function using model
- Typically more sample-efficient than model-free approaches
- Can update action-values without taking real actions in the world

Solution

Model-Based Method

- Learn transition and reward dynamics, then update value function using model
- Typically more sample-efficient than model-free approaches
- Can update action-values without taking real actions in the world
- **But,** can take significant computation time

Solution

Model-Based Method

- Learn transition and reward dynamics, then update value function using model
- Typically more sample-efficient than model-free approaches
- Can update action-values without taking real actions in the world
- **But,** can take significant computation time

Real-Time Architecture

- Parallelize model learning, planning, and acting onto 3 parallel threads
- Utilize an **anytime** sample-based planning algorithm

The TEXPLORE Algorithm

1. Model generalization for **sample efficiency**
2. Handles **continuous** state
3. Handles actuator **delays**
4. Selects actions continually in **real-time**

Available publicly as a ROS package:

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

Time-Constrained Domains

- For many practical problems, agent cannot take thousands of actions
- Actions may be expensive, time-consuming, or dangerous
- Agent does not have enough actions to guarantee it can learn an optimal policy
- Define domains that have this property as **time-constrained domains**

Sample Complexity of Exploration

Definition: Number of sub-optimal actions the agent must take

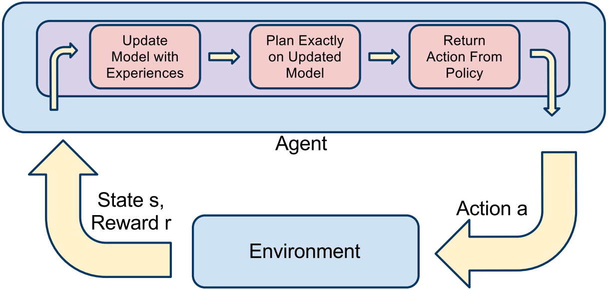
- Proven lower bound: $O(\frac{NA}{\epsilon(1-\gamma)} log(\frac{1}{\delta}))$
- For deterministic domains: $O(\frac{NA}{(1-\gamma)})$ [Kakade 2003]

Time-Constrained Domains

- Lifetime L bounds the number of actions agent can take
- Time-Constrained if $L < 2NA$
- Two orders of magnitude less than lower bound
- The agent does **not** have enough time steps to learn the optimal policy without some additional assumptions about the domain
- **Assumption:** Transition and reward are **similar** across states

Domain	No. States	No. Actions	No. State-Actions	Min Bound Deterministic	Min Bound Stochastic	Maximum L
Taxi	500	6	3,000	300,000	1,050,000	6,000
Four Rooms	100	4	400	40,000	140,000	800
Two Rooms	51	4	204	20,400	72,400	408
Fuel World	39,711	8	317,688	31,768,800	111,190,800	635,376
Mountain Car	10,000	3	30,000	300,000	10,500,000	60,000
Puddle World	400	4	1,600	160,000	560,000	3,200
Cart-Pole Balancing	160,000	2	320,000	32,000,000	11,200,000	640,000

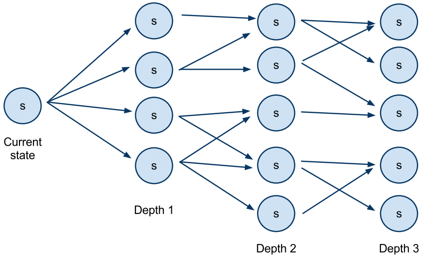
Real-Time Action Selection



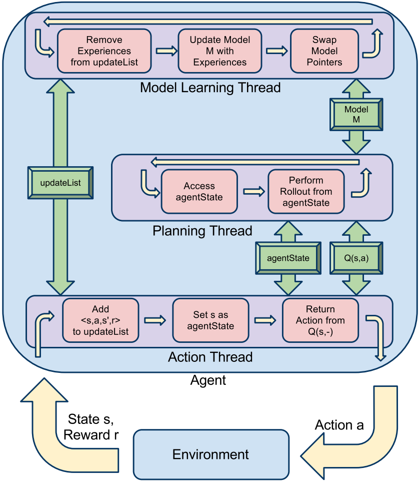
- Model update can take too long
- Planning can take too long

Monte Carlo Tree Search Planning

- **Simulate trajectory** from current state using model (rollout)
- Use upper confidence bounds to select actions (UCT [Kocsis and Szepesvári 2006])
- Focus computation on states the agent is most likely to visit
- **Anytime**—more rollouts, more accurate value estimates
- Update value function at each state in rollout

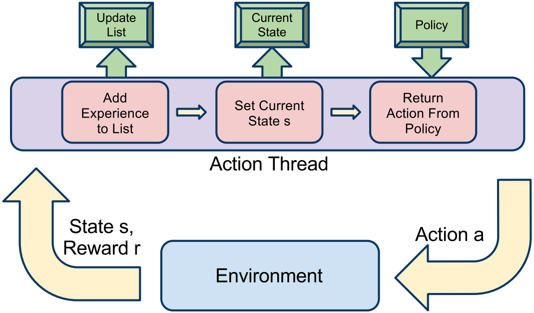


Real-Time Model Based Architecture (RTMBA)



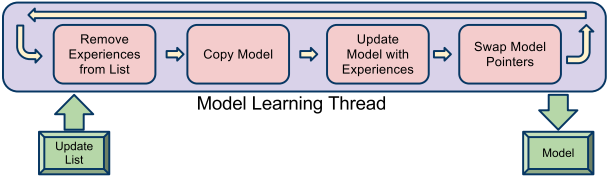
- Model learning and planning on parallel threads
- Action selection **is not restricted** by their computation time
- Use sample-based planning (anytime)
- Mutex locks on shared data

Action Thread



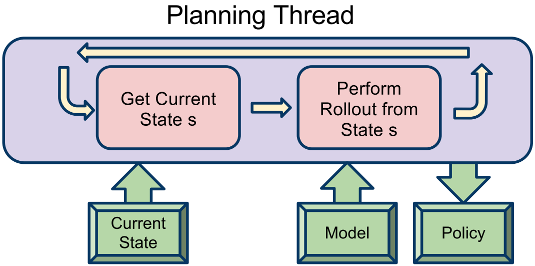
- Add experience, $\langle s, a, s', r \rangle$ to list of experiences to be added to model
- Set agent's current state for planning
- Return best action according to policy

Model Learning Thread



- Make a copy of current model
- Update model copy with new experiences from list (batch updates)
- Swap model pointers
- Repeat

Planning Thread



- Plan using a sample-based MCTS planner (i.e. UCT [Kocsis and Szepesvári 2006])
- Continually perform rollouts from agent's current state
- Rollouts from previous state can help

Sample Efficiency

Model Generalization

- Generalize that actions have similar effects across states
- Do not want to explore every state-action
- Speed model learning by making predictions about unseen state-actions

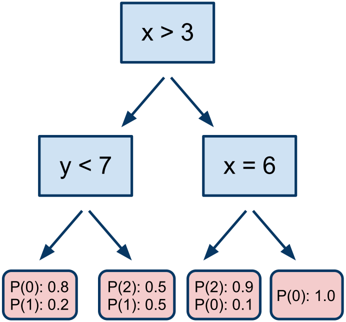
Exploration

- Model learning is dependent on acquiring useful experiences
- Balance exploration and exploitation to maximize rewards in time-constrained lifetime

Model Generalization

- Model learning is a supervised learning problem [Hester and Stone 2009]
- Input: State and Action
- Output: Distribution over next states and reward
- Factored state $s = \langle s_1, s_2, ..., s_n \rangle$
- Separate model for each state feature and reward

C4.5 Decision Trees [Quinlan 1986]

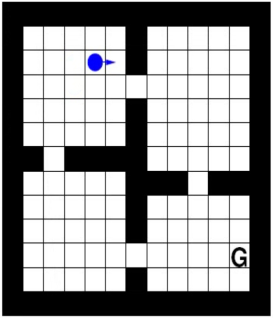


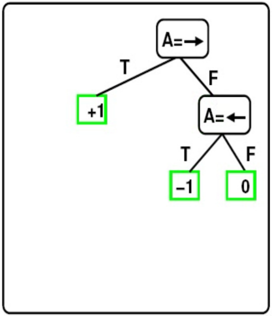
- Incremental and fast
- Generalize broadly at first, refine over time
- Split state space into regions with similar dynamics
- Good at selecting relevant state features to split on

Relative Effects

- Predict the **change in state**: $s^{rel} = s' - s$ rather than absolute next state s'
- Often actions have the **same effect across states**
- Previous work predicts relative effects [Jong and Stone 2007] [Leffler et al. 2007]

How the Decision Tree Model works

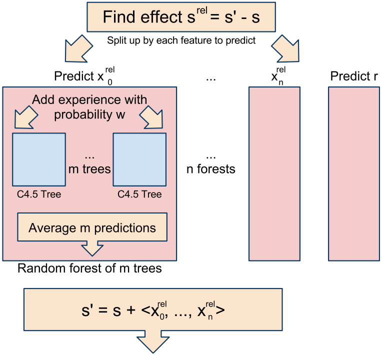




- Build one tree to predict each state feature and reward
- Combine their predictions: $P(s^{rel}|s,a) = \Pi_{i=0}^{n} P(s_i^{rel}|s,a)$
- Update trees on-line during learning

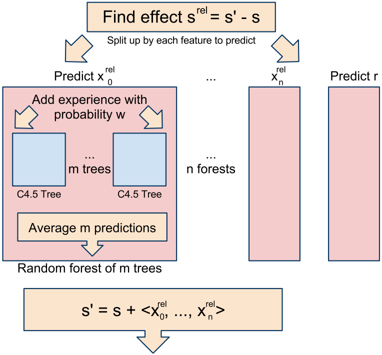
Random Forest Model [Hester and Stone 2010]

- Create a random forest of *m* different decision trees [Breiman 2001]
- Each tree is trained on a **random subset** of the agent's experiences
- Each tree represents a **hypothesis** of the true dynamics of the domain



Random Forest Model [Hester and Stone 2010]

- Create a random forest of *m* different decision trees [Breiman 2001]
- Each tree is trained on a **random subset** of the agent's experiences
- Each tree represents a **hypothesis** of the true dynamics of the domain
- How best to use these different hypotheses?



How to use these hypotheses?

Bayesian Approaches

- BOSS: Plan over most optimistic model at each action
- MBBE: Solve each model and use distribution of q-values

How to use these hypotheses?

Bayesian Approaches

- BOSS: Plan over most optimistic model at each action
- MBBE: Solve each model and use distribution of q-values

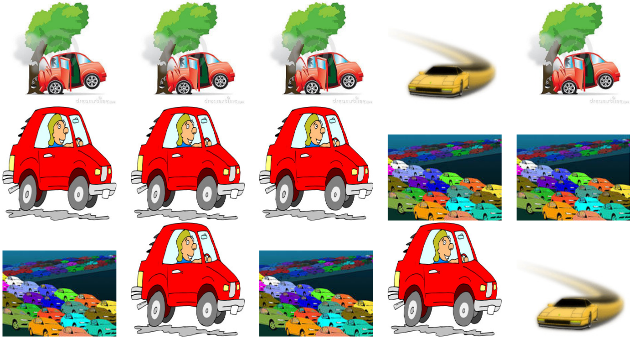
TEXPLORE

- Desiderata: Explore less, be greedier
- Plan on average of the predicted distributions
- Balance models that are optimistic with ones that are pessimistic

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

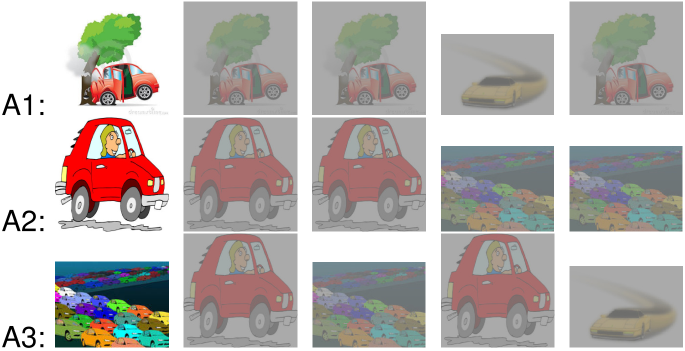
A1:
A2:
A3:



5 models

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

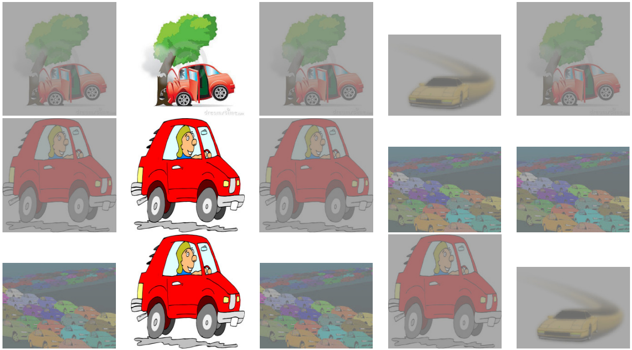


Model 1

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

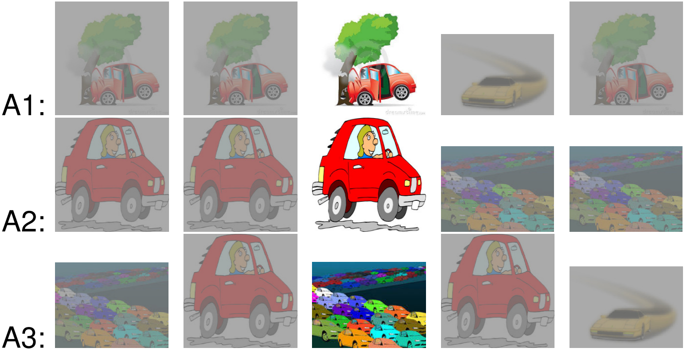
A1:
A2:
A3:



Model 2

Exploration

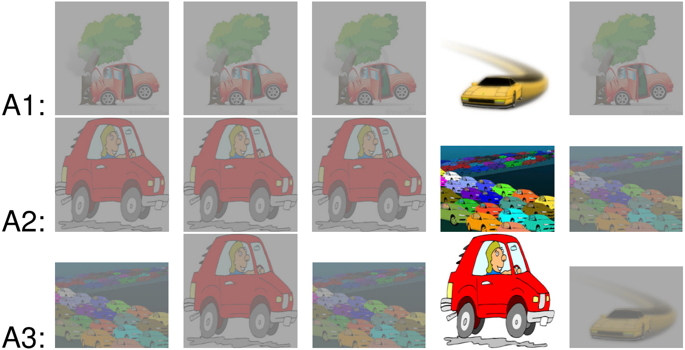
- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes



Model 3

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes



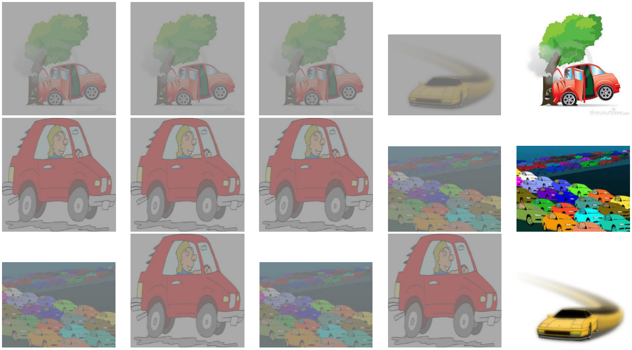
Model 4

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

A1:

A2:

A3:



Model 5

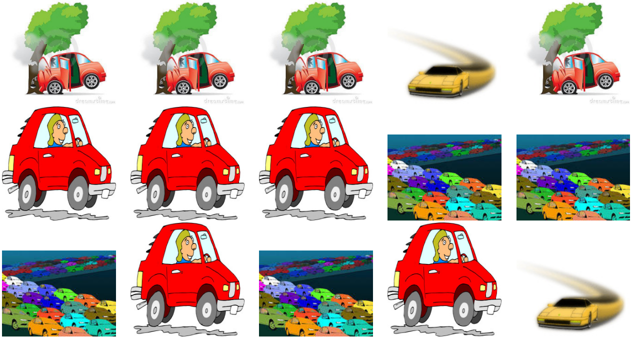
Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

A1:

A2:

A3:

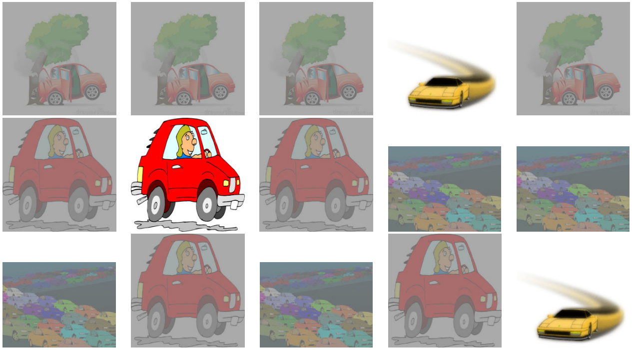


BOSS

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

A1:
A2:
A3:

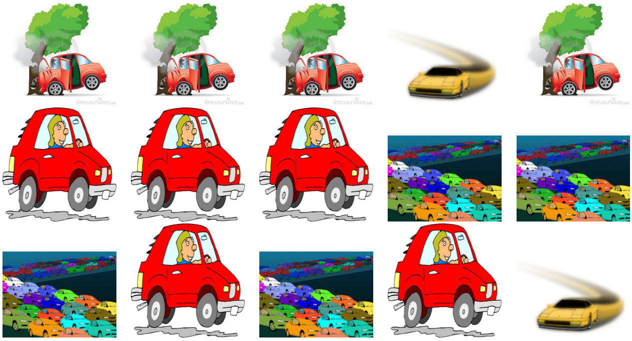


BOSS

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

A1:
A2:
A3:

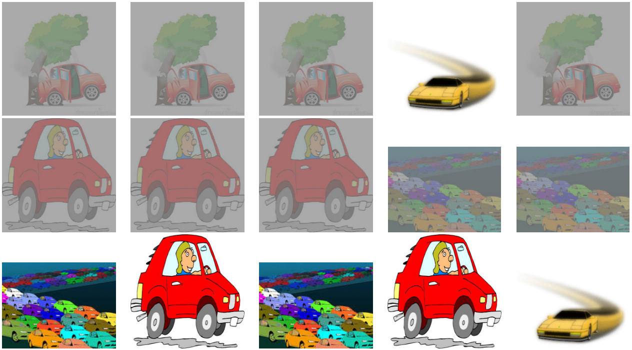


MBBE

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

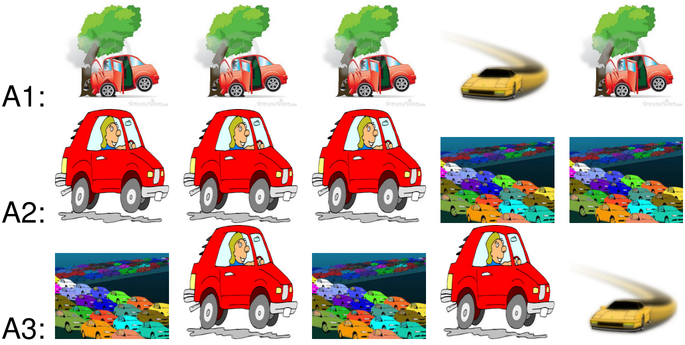
A1:
A2:
A3:



MBBE

Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes



TEXPLORE

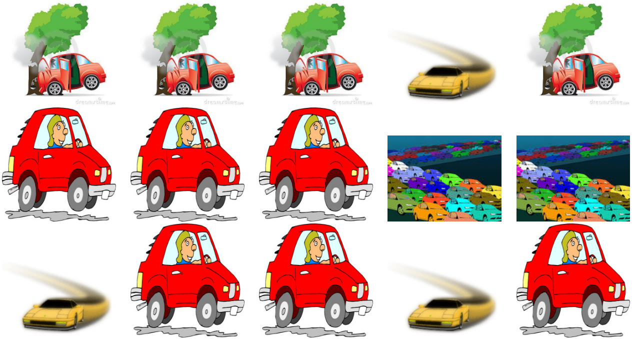
Exploration

- **Limits** exploration to state-actions that appear promising, **avoids** those which may have negative outcomes

A1:

A2:

A3:



TEXPLORE

Seed experiences

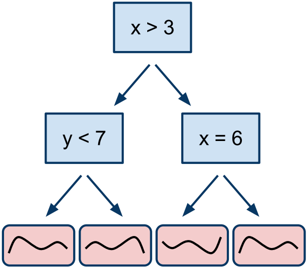
- Do not want to start from scratch learning on robots [Smart and Kaelbling 2002]
- Provide a few **example transitions** to initialize model
- Example transitions could come from human experience
- Avoid having the agent explore **every** state-action for unusual states

Continuous State

Problems

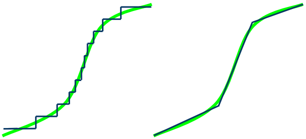
- Make continuous predictions
- Plan over continuous state space

Continuous Modeling



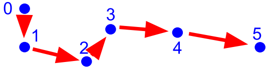
- Use M5 regression trees to model continuous state [Quinlan 1992]
- Each tree has a linear regression model at its leaves
- Piecewise linear prediction

Continuous Modeling



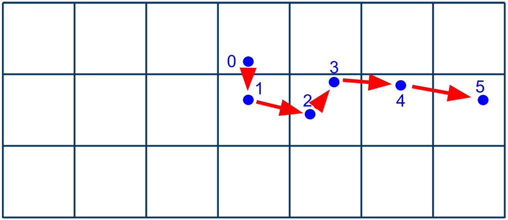
- Use M5 regression trees to model continuous state [Quinlan 1992]
- Each tree has a linear regression model at its leaves
- Piecewise linear prediction

Continuous Planning



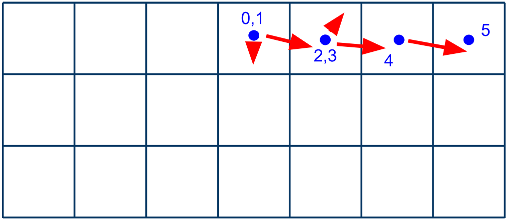
- Perform UCT rollouts over continuously valued states
- Discretize state space only for value backups

Continuous Planning



- Perform UCT rollouts over continuously valued states
- Discretize state space only for value backups

Continuous Planning

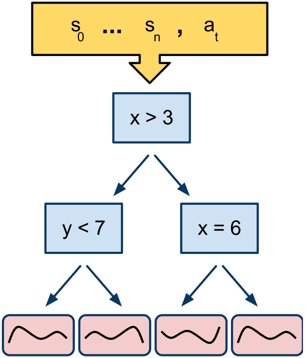


- Perform UCT rollouts over continuously valued states
- Discretize state space only for value backups
- Know where in discretized state you are

Delays

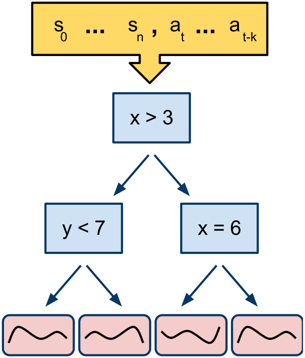
- Must know what state robot will be in when action is executed
- Delays make domain **non-Markov**, but **k-Markov**

Modeling Delay



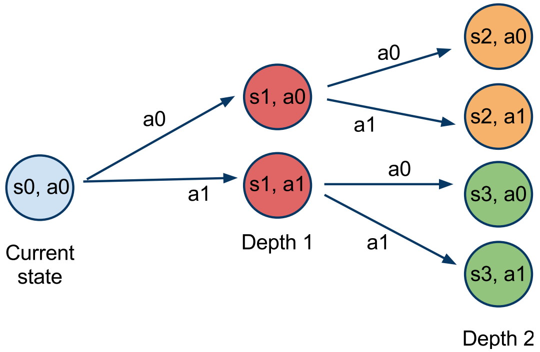
- Provide model with previous *k* actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant
- Only requires **upper bound on *k***

Modeling Delay



- Provide model with previous *k* actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant
- Only requires **upper bound on *k***

Planning with Delays



- UCT can plan over augmented state-action histories easily
- Would not be as easy with dynamic programming

The TEXPLORE Algorithm

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

The TEXPLORE Algorithm

1. Limits exploration to be **sample efficient**
2. Handles **continuous** state
3. Handles actuator **delays**
4. Selects actions continually in **real-time**
5. Models domains with **dependent feature transitions**

Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- Vehicle runs at 10 Hz.
- Agent **must** provide commands at this frequency.

Velocity Control Task

- 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)

Simulated Velocity Control Task

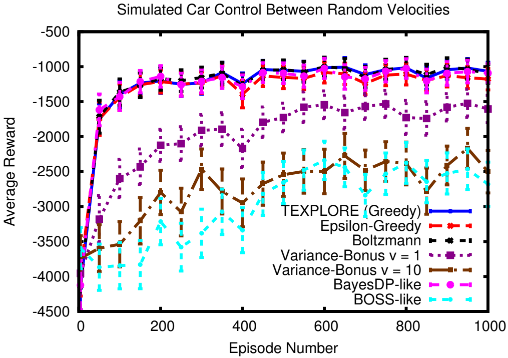
- Experiments performed **in simulation**
- 10 second episodes (100 samples)
- **Random starting and target velocity** chosen each episode
- Time-Constrained Lifetime is $436,150$ actions ($4,361$ episodes)
- **No seed experiences**
- Brake is controlled by a **PID** controller

Exploration Comparisons using TEXPLORE's model

1. TEXPLORE
2. ϵ-greedy exploration ($\epsilon = 0.1$)
3. Boltzmann exploration ($\tau = 0.2$)
4. VARIANCE-BONUS Approach $v = 1$ [Deisenroth & Rasmussen 2011]
5. VARIANCE-BONUS Approach $v = 10$
6. Bayesian DP-like Approach (use sampled model for 1 episode) [Strens 2000]
7. BOSS-like Approach (use optimistic model) [Asmuth et al. 2009]

First five approaches use **TEXPLORE's model**

Sample Efficiency Results

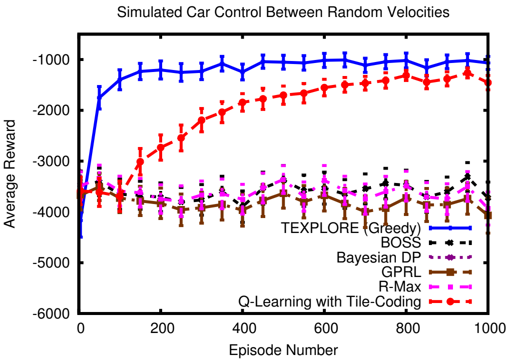


- Adding ϵ-greedy, Boltzmann, or Bayesian DP-like exploration **does not** improve performance

Comparing with other models

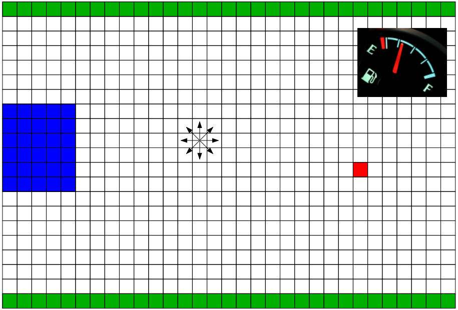
1. BOSS (Sparse Dirichlet prior) [Asmuth et al. 2009]
2. Bayesian DP (Sparse Dirichlet prior) [Strens 2000]
3. PILCO (Gaussian Process Regression model) [Deisenroth & Rasmussen 2011]
4. R-MAX (Tabular model) [Brafman & Tennenholtz 2001]
5. Q-LEARNING using tile-coding [Watkins 1989]

Sample Efficiency Results



- TEXPLORE accrues **significantly more rewards** than all the other methods after episode 24 ($p < 0.01$).

Fuel World

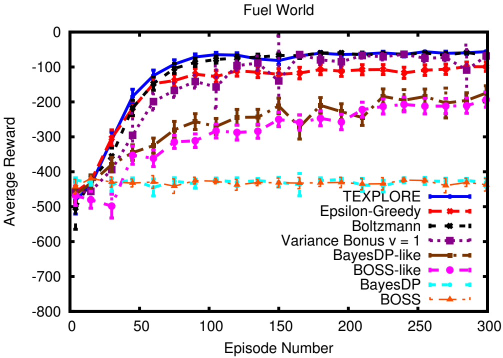


- Most of state space is very **predictable**
- But fuel stations have **varying costs**
- 317, 688 State-Actions, Time-Constrained Lifetime: 635, 376 actions
- Seed experiences of goal, fuel station, and running out of fuel

Comparison methods

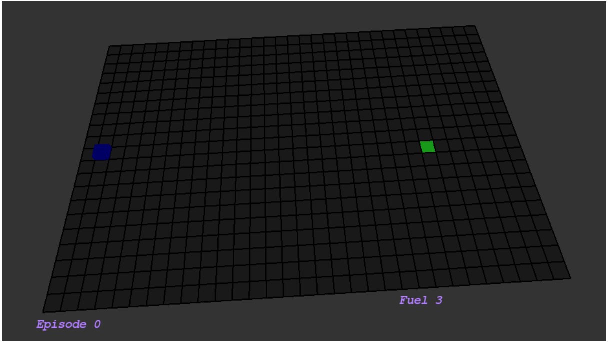
1. TEXPLORE (Greedy w.r.t. aggregate model)
2. ϵ-greedy exploration ($\epsilon = 0.1$)
3. Boltzmann exploration ($\tau = 0.2$)
4. VARIANCE-BONUS Approach $v = 10$ [Deisenroth & Rasmussen 2011]
5. Bayesian DP-like Approach (use sampled model for 1 episode) [Strens 2000]
6. BOSS-like Approach (use optimistic model) [Asmuth et al. 2009]
7. BOSS (Sparse Dirichlet prior) [Asmuth et al. 2009]
8. Bayesian DP (Sparse Dirichlet prior) [Strens 2000]

Fuel World Results



- TEXPLORE learns the **fastest** and **accrues the most cumulative reward** of any of the methods.
- TEXPLORE learns the task **within the time-constrained lifetime** of 635, 376 steps.

Fuel World Behavior

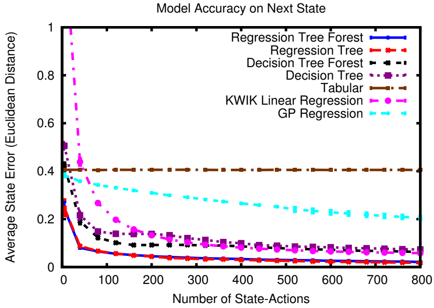


- Agent **focuses its exploration** on fuel stations near the shortest path to the goal.
- Agent **finds near-optimal policies**.

Continuous Models

1. Regression Tree Forest (TEXPLORE Default)
2. Single Regression Tree
3. Decision Tree Forest
4. Single Decision Tree
5. Tabular Model
6. KWIK Linear Regression [Strehl and Littman 2007]
7. Gaussian Process Regression (PILCO model) [Deisenroth & Rasmussen 2011]

Continuous Model Accuracy



- **Regression tree forest and single regression tree** have significantly less error than all the other models in predicting the next state ($p < 0.001$).
- For reward, regression tree is significantly better than all models but GP regression after 205 state-actions ($p < 0.001$).

Methods for Delays

Model-Based Simulated (MBS) [Walsh et al. 2009]

- Input **exact value** of delay k
- Use model to simulate forward k steps
- Use policy at that state to select action

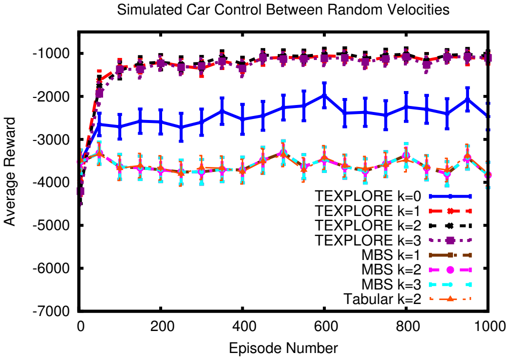
Tabular model

- Separate table entry for each state-action-history tuple

Car brake delay

- Brake pedal is physically actuated, controlled with PID
- Delay is **not** a number of discrete steps
- **Delay varies** based on how far brake is from target position

Handling Action Delays



- TEXPLORE with $k = 1, 2$, or 3 all perform **significantly better** than than using no delay ($k = 0$) ($p < 0.005$).
- These approaches are **significantly better** than using another approach to handling delay ($p < 0.005$).

Real-Time Action Selection Methods

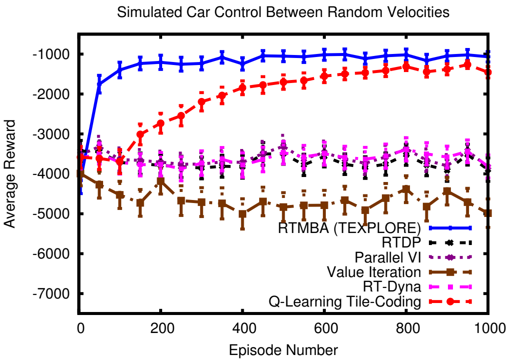
Using TEXPLORE's model

1. RTMBA (TEXPLORE)
2. Real Time Dynamic Programming (RTDP) [Barto et al. 1995]
3. Parallel Value Iteration
4. Value Iteration

Other methods

1. Dyna [Sutton 1990]
2. Q-Learning with tile-coding [Watkins 1989]

Real-Time Action Selection Results



- TEXPLORE receives **significantly more average rewards** per episode than the other methods after episode 29 ($p < 0.01$)

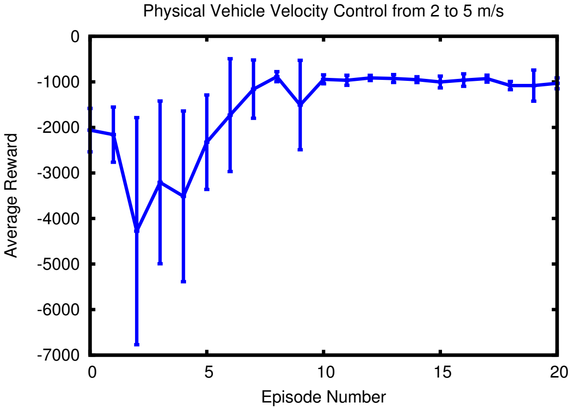
On the physical vehicle

- But, does it work on the actual vehicle?

On the physical vehicle

- 5 trials, starting at 2 m/s, target of 5 m/s.
- Time-constrained lifetime: 33, 550 steps, or 335 episodes.
- No seed experiences

On the physical vehicle



- **Yes!** It learns the task in **2 minutes** (< 11 episodes)

Characterization of Domains

Haystack Domains

- Some state-action with unusual transition or reward function image (doorway, goal, etc.)
- Best exploration: try each state-action

Informative Domains

- Some state features predict the locations of unusual states (robot with distance sensors, camera)
- Can use these features to explore more intelligently

Prior Information Domains

- Agent is given information about location of unusual states (given a map for navigating)

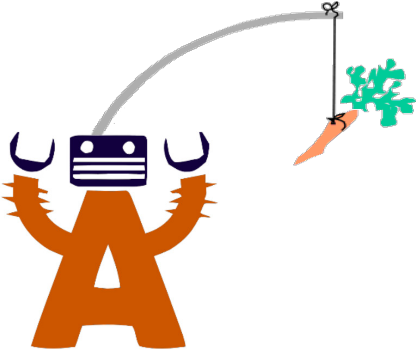
Exploration for Different Domain Types

- Haystack Domains
  - **TEXPLORE with Explicit Exploration** (TEXPLORE-EE)
- Informative Domains
  - **TEXPLORE with Variance and Novelty Intrinsic Rewards** (TEXPLORE-VANIR)
- **Unknown** Domain Type
  - **TEXPLORE with Learning Exploration Online** (TEXPLORE-LEO)

Exploration for Different Domain Types

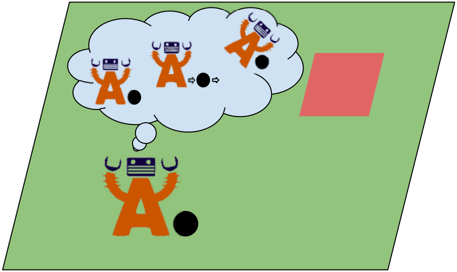
- Haystack Domains
  - **TEXPLORE with Explicit Exploration** (TEXPLORE-EE)
- Informative Domains
  - **TEXPLORE with Variance and Novelty Intrinsic Rewards** (TEXPLORE-VANIR)
- **Unknown** Domain Type
  - **TEXPLORE with Learning Exploration Online** (TEXPLORE-LEO)

TEXPLORE-VANIR for Informative Domains



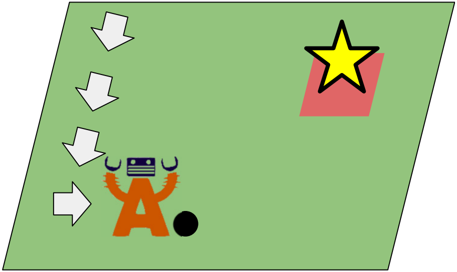
- Use **intrinsic rewards** to drive exploration
- Combine TEXPLORE model with **two** intrinsic rewards:
  1. Drives agent to where model is uncertain
  2. Drives agent to transitions different from what the model was trained on

Variance Intrinsic Reward



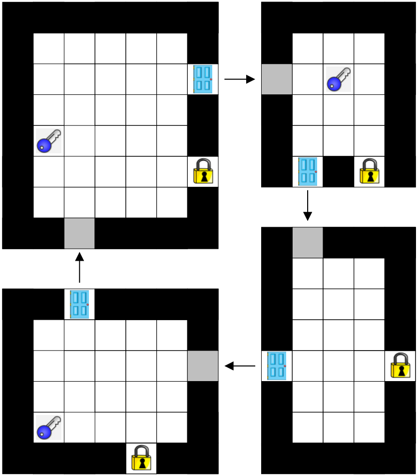
- Reward where model is **uncertain**
- Calculate a **measure of variance**:
  $D(s,a) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} D_{KL}(P_j(x_i^{rel}|s,a) || P_k(x_i^{rel}|s,a))$
- Add intrinsic reward proportional to variance measure:
  $R(s,a) = \mathbf{v} D(s,a)$

Novelty Intrinsic Reward



- Reward transitions that are **most different** from what was seen
- Calculate L_1 **distance in feature space** to nearest state where this action was taken:
  $\delta(s,a) = \min_{s_x \in X_a} ||s - s_x||_1$
- Add intrinsic reward proportional to novelty measure:
  $R(s,a) = \mathbf{n}\delta(s,a)$
- Given enough time, will drive agent to explore **all** state-actions.

LightWorld Domain [Konidaris and Barto 2007]



- Six actions: N, S, E, W, PICKUP, PRESS
- Agent must PICKUP key, use it to PRESS lock, and then can leave through door
- Keys, locks, and unlocked doors **emit** different colors of light
- 17 state features: ROOM, X, Y, KEY, LOCKED, and RED, GREEN, and BLUE light sensors in each of the four directions
- Reward: $+10$ for exiting door, 0 otherwise

Comparison Methods

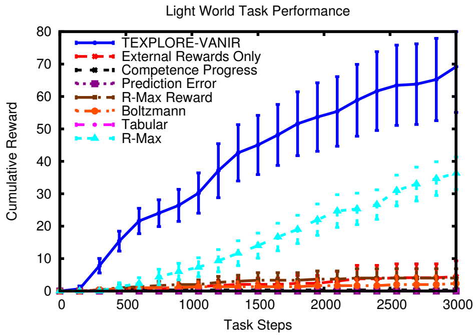
TEXPLORE Model

1. TEXPLORE-VANIR with v = 1, n = 3
2. External Rewards Only (TEXPLORE)
3. Bonus for regions with more competence progress (similar to IAC [Baranes and Oudeyer 2009])
4. Bonus for regions with higher prediction errors
5. Explore state-actions with fewer than m visits (R-MAX [Brafman and Tennenholtz 2001])

Tabular Model

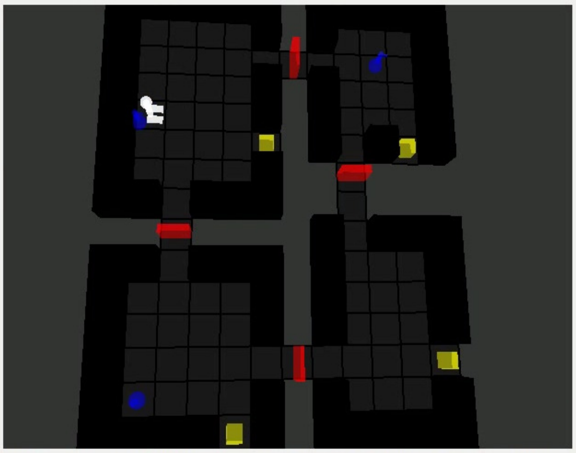
1. External Rewards Only
2. R-MAX (explore state-actions with fewer than m visits)

Task Performance



- TEXPLORE-VANIR receives **significantly more** cumulative rewards ($p < 0.001$).

TEXPLORE-VANIR Exploration



- **Novelty** rewards draw agent to objects and corners.
- **Variance** rewards make it explore using objects.

Related Work: Sample Efficient Model Learning

- **SPITI** [Degris et al. 2006]
  - Learn decision tree models for each feature
  - Used ϵ-greedy exploration
- **AMBI** [Jong and Stone 2007]
  - Instance-based model with relative effects
  - R_{max} bonus for state regions with few visits
- **PILCO** [Deisenroth and Rasmussen 2011]
  - Use Gaussian Process regression to model dynamics
  - Exploration based on variance of GP predictions
  - Batch mode, agent is provided reward model

Related Work: Bayesian RL

- Offers optimal solution to exploration problem [Duff 2003]
- Computationally intractable
- Many approximate solutions:
  - Tie model parameters together [Poupart et al. 2006]
  - Sample from model distributions [Strens 2000, Asmuth et al. 2009]
  - Learn Bayesian optimal policy over time [Kolter and Ng 2009]

Related Work: Continuous State

- **KWIK Linear Regression** [Strehl and Littman 2007]
  - Linear regression model with prediction confidence
  - Only for linearly parametrized domains
- **FITTED-R-MAX** [Jong and Stone 2007]
  - R-MAX style algorithm in continuous state
  - Use fitted value iteration [Gordon 1995]

Related Work: Sensor and Actuator Delay

- **Model Based Simulation** (MBS) [Walsh et al. 2009]
  - Provide domain's delay, k
  - Simulate k steps ahead in model, take best action for this state
- **U-TREE** [McCallum 1996]
  - Build decision trees for representing value function
  - Split on previous actions to handle delays

Related Work: Real-Time Architecture

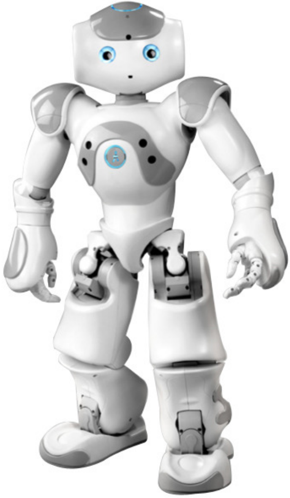
- **Dyna** Framework [Sutton 1990, 1991]
  - Do Bellman updates on random states using model when not action
  - Still uses tabular model, assumes model update takes insignificant time
- Combining sample-based planning with model-based method
  - With UCT [Silver et al. 2008], With FSSS [Walsh et al. 2010]
  - Neither places a time restriction on model update or planning
- **Real-Time Dynamic Programming** RTDP [Barto et al. 1995]
  - Similar to UCT, but full backups at each state

Related Work: Robot Learning

- **Helicopter Control** [Ng et al. 2003]
  - Learn helicopter model from experiences acquired from human expert
  - Computation performed off-line
- **PILCO** [Deisenroth and Rasmussen 2011]
  - Use Gaussian Process regression for model learning and planning
  - Very sample efficient in learning to control cart-pole
  - Takes 10 minutes of computation for every 2.5 seconds of experience
- **POWER** [Kober and Peters 2008]
  - Policy search for parameterized motor primitives
  - Only for episodic tasks

Future Work: Continuous Actions

- Many robots could utilize **continuous actions** (angles, velocities)
- Regression tree model can make predictions on the basis of continuous actions
- Could utilize work on UCT-like planning in continuous actions spaces (HOOT) using continuous bandit algorithms [Bubeck et al. 2011; Mansley et al. 2011; Weinstein and Littman 2012]



Future Work: Opponent Modeling

- Initialize each tree in forest with **possible opponent strategy**
- Could be from experience with past opponents
- Explore to determine which type of opponent you are playing

Future Work: Lifelong Robot Learning

- **Goal:** Act and learn in environment over lifetime, performing many tasks
- Handle large and complex state space: **Make algorithm more parallel**
- Generalize knowledge to new tasks: **Find best state representation**

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
- Targeted Exploration
- Real-Time Architecture
- ROS RL Interface
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
  - Limits exploration to be **sample efficient**
  - Handles **continuous** state
  - Handles actuator **delays**
  - Selects actions continually in **real-time**
  - Models domains with **dependent feature transitions**
- MDP Models that generalize action effects
- Targeted Exploration
- Real-Time Architecture
- ROS RL Interface
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
  - Random forests of regression trees
- Targeted Exploration
- Real-Time Architecture
- ROS RL Interface
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
- Targeted Exploration
  - Average predictions of each tree in random forest
  - Use intrinsic rewards for **informative** domains
- Real-Time Architecture
- ROS RL Interface
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
- Targeted Exploration
- Real-Time Architecture
  - Maintain **sample efficiency** of model-based methods
  - While acting in **real-time**
- ROS RL Interface
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
- Targeted Exploration
- Real-Time Architecture
- ROS RL Interface
  - Enables **easy integration** of RL on robots using ROS
- Empirical Evaluation

Contributions

- The TEXPLORE algorithm
- MDP Models that generalize action effects
- Targeted Exploration
- Real-Time Architecture
- ROS RL Interface
- Empirical Evaluation
  - Real-time learning while running on-board **physical robot**

Conclusion

- TEXPLORE:
  1. Learns in few **samples**
  2. Acts continually in **real-time**
  3. Learns in **continuous** domains
  4. Handles sensor and actuator **delays**
- TEXPLORE has been released as a ROS package: www.ros.org/wiki/rl-texplore-ros-pkg