Approximate Universal Artificial Intelligence

A Monte-Carlo AIXI Approximation

Joel Veness   Kee Siong Ng   Marcus Hutter   Dave Silver

UNSW / NICTA / ANU / UoA

September 8, 2010

General Reinforcement Learning Problem

Worst case scenario. Environment is unknown. Observations may be noisy. Effects of actions may be stochastic. No explicit notion of state. Perceptual aliasing. Rewards may be sparsely distributed.

Notation:

- Agent interacts with an unknown environment μ by making actions $a \in \mathcal{A}$.
- Environment responds with observations $o \in \mathcal{O}$ and rewards $r \in \mathcal{R}$. For convenience, we often use $x \in \mathcal{O} \times \mathcal{R}$.
- The goal of the agent is to maximize its expected total future reward.

Optimality Notions and Reinforcement Learning

- What does it mean for an agent to “solve” the general reinforcement learning problem?
- The AIXI agent provides one such *notion of optimality*. It is a purely mathematical notion and ignores computational concerns.
- In what sense can we approximate AIXI computationally? Can we do this efficiently? Will it lead to interesting and practical agent algorithms? These questions are the focus of my research.

The AIXI agent in one simple equation...

$$a_t^{AIXI} = \arg\max_{a_t} \sum_{o_t r_t} \ldots \max_{a_{t+m}} \sum_{o_{t+m} r_{t+m}} [r_t + \cdots + r_{t+m}] \sum_{q : U(q, a_1 \ldots a_{t+m}) = o_1 r_1 \ldots o_{t+m} r_{t+m}} 2^{-\ell(q)}$$

Where:

- $U(q, a_{1:n})$ is a Universal Turing Machine, running program q with the agent action sequence $a_{1:n}$ as input
- q is some "environment program"
- $l(q)$ gives the length of program q

Caveat: Incomputable

Where we are headed...

- AIXI provides *practical guidance* to constructing real-world RL algorithms.
- Will introduce the MC-AIXI-CTW agent, a real world, efficient, model based RL agent that can be viewed as a scaled down AIXI agent.
- MC-AIXI-CTW is the extension and synthesis of two *powerful* algorithms: CTW and UCT.
- Even if AIXI theory doesn’t interest you, the empirical results might.

AIXI as a principle

An alternative characterization of the AIXI agent:

$$a_t^{AIXI} = \arg\max_{a_t} \sum_{x_t} \ldots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{\rho \in \mathcal{M}} 2^{-K(\rho)} \rho(x_{1:t+m} \,|\, a_{1:t+m}),$$

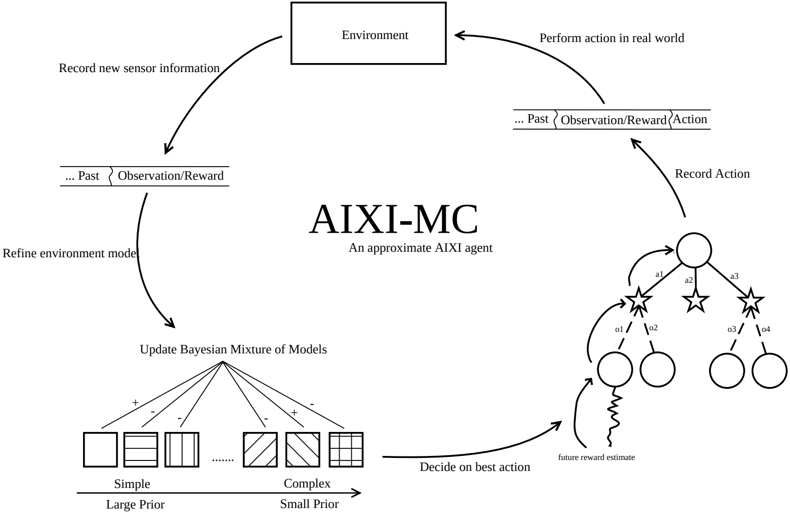
Note:

- Uses a Bayesian Sequence Prediction method
- Kolmogorov Complexity used as an Ockham prior
- Immediately suggestive of a direct approximation!

What does MC-AIXI-CTW inherit from AIXI theory

- Uses same problem setup: i.e. general reinforcement learning problem.
- Uses same definition of *environment model*, a probabilistic predictor describing what an agent will experience next, given everything that it has already experienced.
- Uses a computationally efficient combination of approximate Expectimax planning with Bayesian Sequence Prediction.
- No exploration/exploitation issue.
  Value of information implicitly captured just like in AIXI!
- Inherits strong theoretical results from Bayesian Sequence Prediction theory.

Overview of proposed agent architecture



Environment

Record new sensor information

... Past Observation/Reward

Refine environment model

Update Bayesian Mixture of Models

Simple

Complex

Large Prior

Small Prior

Decide on best action

AIXI-MC

An approximate AIXI agent

future reward estimate

Record Action

... Past Observation/Reward Action

Perform action in real world

Background

Two main areas:

- Learning - online sequence prediction / model building
- Planning/Control - search / sequential decision theory

The hard parts:

- Large model class required for Bayesian mixture predictor to have *general* prediction capabilities.
- Fortunately, an efficient and general class exists: all learning Prediction Suffix Trees of maximum finite depth D. Class contains over $2^{2^{D-1}}$ models!
- If details hard to follow, just think of the model class as all D-Markov models, with a prior that favours “less complex” models.

Sequence Prediction with Bayes Mixtures

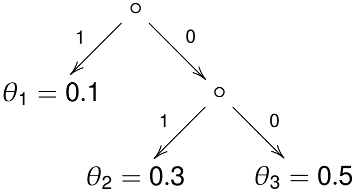
Consider a class of models $\mathcal{M}$ and some (binary) data $x_{1:n}$ generated by an unknown model $\mu \in \mathcal{M}$. Consider the mixture $\xi(x_{n+1} \,|\, x_{1:n}) := \sum_{\nu \in \mathcal{M}} \nu(x_{n+1} \,|\, x_{1:n}) w_\nu^n$, where $\forall n > 0$, posterior w_ν^n is updated using Bayes Rule and w_ν^0 is the prior on ν.

- The predictions made by ξ *rapidly* converge to μ.
- Pareto Optimality of ξ. No predictor can peform at least as well as ξ for all $\nu \in \mathcal{M}$ and strictly better for a least one $\nu' \in \mathcal{M}$.
- If $\mu \notin \mathcal{M}$, ξ rapidly converges to $\hat{\mu} \in \mathcal{M}$, where $\hat{\mu}$ is the best (w.r.t KL-divergence) predictor in $\mathcal{M}$.

Prediction Suffix Trees

A prediction suffix tree is a simple, tree based variable length Markov model. For example, using the PST below, having initially been given data 01:

$$
\begin{aligned}
\Pr(010|01) &= \Pr(0|01) \times \Pr(1|010) \times \Pr(0|0101) \\
&= (1-\theta_1)\theta_2(1-\theta_1) \\
&= 0.9 * 0.3 * 0.9 \\
&= 0.243
\end{aligned}
$$



Context Tree Weighting

Context Tree Weighting is an online prediction method that was originally developed for data compression. It uses a mixture of prediction suffix trees to make predictions. Smaller prediction suffix trees are given initial higher weight, which helps to avoid overfitting when data is limited. If we let C^D denote the class of all prediction suffix trees of maximum depth D, then CTW computes:

$$\Pr(x_{1:t}) = \sum_{M \in C_D} 2^{-\Gamma_D(M)} \Pr(x_{1:t} \,|\, M) \tag{1}$$

in time $O(D)$. $\Gamma_D(\cdot)$ is a description length based prior. This is truly amazing, as computing the sum naively would take time double-exponential in D!

Connection to Bayesian Sequence Prediction

With a bit of simple algebra, one can show that the previous mixture equation implies:

$$\Pr(x_t \,|\, x_{1:t-1}) = \sum_{M \in C_D} \Pr(x_t \,|\, M, x_{1:t-1}) \Pr(M \,|\, x_{1:t-1}) \qquad (2)$$

which makes the connection to Bayesian sequence prediction explicit. i.e. $\Pr(M \,|\, x_{1:t-1})$ is the posterior weight of model M given the data.

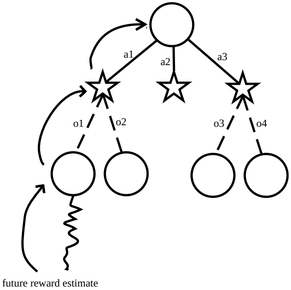
Expectimax - An optimality notion for planning

$$a^* := \arg\max_{a \in \mathcal{A}} \sum_{(o_1, r_1) \in \mathcal{O} \times \mathcal{R}} \Pr(o_1, r_1) \ldots \max_{a \in \mathcal{A}} \sum_{(o_m, r_m) \in \mathcal{O} \times \mathcal{R}} \Pr(o_m, r_m) \left[\sum_{i=1}^{m} r_i \right]$$

- A natural optimality notion given we know the true environment.
- Intuitively: the expectation, with respect to all possible futures, if we picked the best action at each (possible) future time point up to a fixed horizon *m*.
- Yields a straightforward, brute force, decision theoretic algorithm.
- However, branching factor enormous, and search horizon may be large. Need a smarter approximation!

UCT - Bandit Based Monte Carlo Tree Search

- Online planning algorithm for finite horizon MDPs.
- Requires a generative model of the environment.
- Converges to the expectimax value defined previously.



What has been done...

- Generalise CTW to the agent setting (Action-conditional CTW)
- Generalise UCT from MDPs to our history based setting (μUCT)
- Shown that μUCT converges to the expectimax value
- Extended CTW with a ”revert” operation, required for the MCTS

Finally, the two most important steps:

- μUCT $+$ Action-conditional CTW $=$ MC-AIXI-CTW
- Implement it. :-)

Relationship to AIXI

One can show that MC-AIXI-CTW, given enough thinking time, chooses:

$$a_t = \arg\max_{a_t} \sum_{x_t} \cdots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{M \in \mathcal{C}_D} 2^{-\Gamma_D(M)} \Pr(x_{1:t+m} \,|\, M, a_{1:t+m})$$

In contrast, AIXI chooses:

$$a_t = \arg\max_{a_t} \sum_{x_t} \ldots \max_{a_{t+m}} \sum_{x_{t+m}} \left[\sum_{i=t}^{t+m} r_i \right] \sum_{\rho \in \mathcal{M}} 2^{-K(\rho)} \Pr(x_{1:t+m} \,|\, a_{1:t+m}, \rho),$$

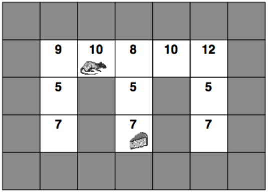
Algorithmic considerations

- Restricted the model class to gain the desirable computational properties of CTW
- Approximated the finite horizon expectimax operation with a MCTS procedure
- $O(Dm \log(|\mathcal{O}||\mathcal{R}|))$ operations needed to generate m observation/reward pairs (for a single simulation)
- $O(tD \log(|\mathcal{O}||\mathcal{R}|))$ space overhead for storing the context tree.
- Anytime search algorithm
- Search is embarassingly parallel
- $O(D)$ to update the context tree online

Experimental Results

- MC-AIXI-CTW agent applied to a number of toy problems.
- Agent needs to *learn from scratch* a model of the environment dynamics.
- Model is then used by μUCT to choose best action.
- Some domains have noisy observations, perceptual aliasing, etc.
- Most domains come from the POMDP literature. However, *learning and solving* a POMDP is *much* more difficult than solving a POMDP given an explicit POMDP model.

Domain description - Cheese Maze

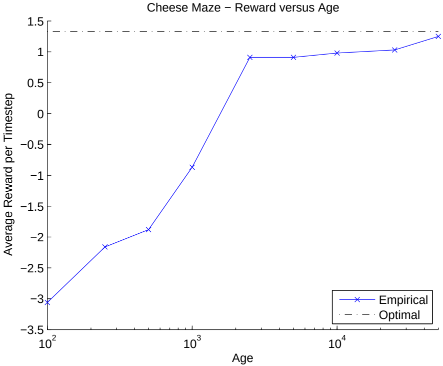


$\mathcal{A}$ = { north, south, east, west }

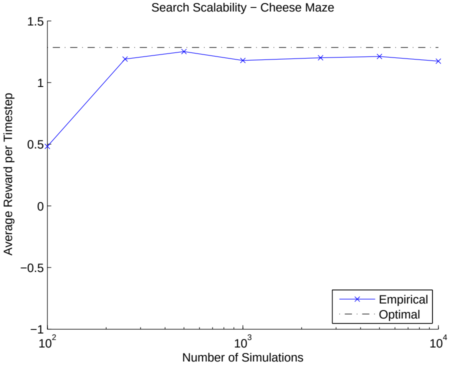
$\mathcal{O}$ = { wall-north, wall-south, wall-east, wall-west }

$\mathcal{R}$ = { -1, 10, -10 }

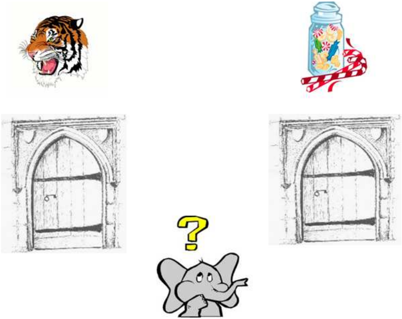
Performance versus age



Performance versus time



Domain description - Tiger

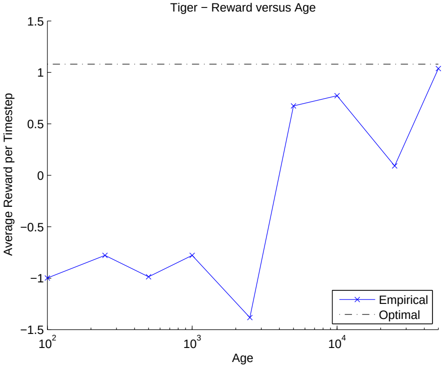


$\mathcal{A}$ = { listen, open-left, open-right }

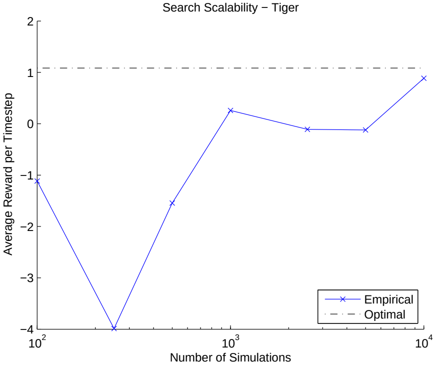
$\mathcal{O}$ = { none, tiger-left, tiger-right }

$\mathcal{R}$ = { -1, 10, -100 }

Performance versus age



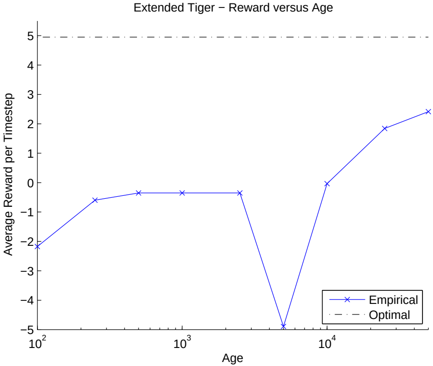
Performance versus time



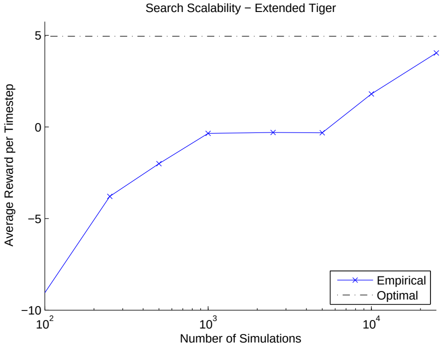
Domain description - Extended Tiger

- Similar to Tiger domain, except agent now starts sitting on a chair
- Agent has an additional action: *stand*
- Listening whilst sitting provides information, listening while standing doesn't.
- Opening the correct door now gives a reward of 30.

Performance versus age



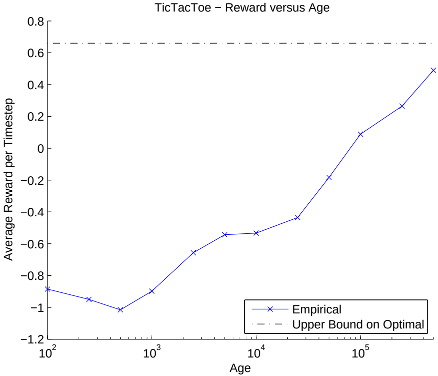
Performance versus time



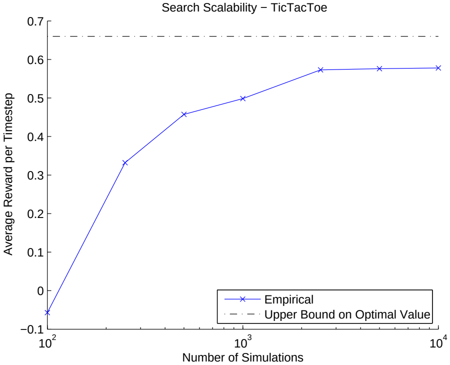
Domain description - TicTacToe

- TicTacToe
- Observations: 18 bit encoding of the board state
- Reward: +2 for a win, +1 draw, 0 game continuing, -2 loss, -3 illegal move
- Actions: Mark a cross in one of 9 cells.

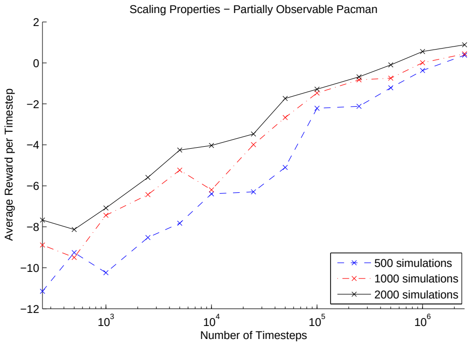
Performance versus age



Performance versus time



Performance on a challenging domain - Pocman



- Show video after 2.5 million steps of interaction.

Summary

- Similar results on other well-known toy POMDPs: 4x4 grid, Windy grid world, 1d Maze, HeavenHell etc.
- Approximately optimal performance on toy problems, given enough thinking time.
- Although problems are modest, these results represent *state of the art performance* in this difficult setting.
- A validation of AIXI? I think so, but you don't have to.
- Agent has been designed with scalability in mind. How will it go on more challenging problem domains?

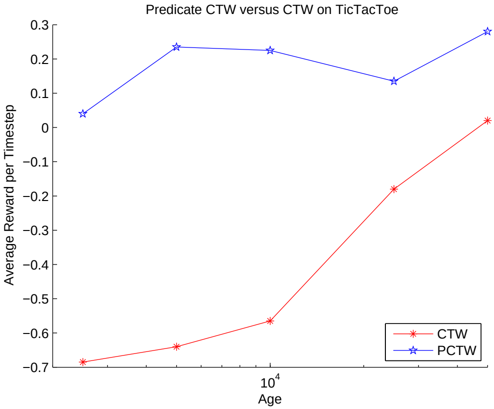
Limitations

- Prediction Suffix Trees are obviously simplistic models - a long way from the AIXI ideal.
- If the environment is not *n*-Markov, one cannot expect good performance
- The playout policy used for the MCTS doesn't incorporate or learn any domain knowledge.
- However, a general agent framework now is in place. How can we scale it up?

Predicate CTW

- Natural generalisation of the notion of ”context”.
- Context is a vector of predicate values.
- Predicates are arbitrary boolean functions on the agent’s history.
- In principle, a way to enrich the model class.
- Also a way to incorporate domain knowledge.
- For example, one could add a “is the last move legal” predicate to TicTacToe...

Some results on TicTacToe



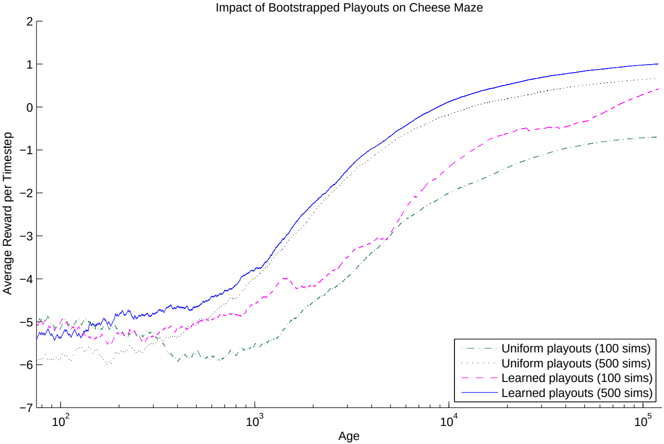
Future work

- Easy to extend the model class to improve general prediction ability.
- However, not so easy to do this in a *computationally efficient* manner.
- My current research involve looking at *efficient* ways of using more expressive mixtures; i.e. I want to move well beyond *n*-Markov predictors.
- Need to understand that MC-AIXI-CTW is pushing the limits of what can be done on present day desktop machines. Motivation to finish PhD: access to more resources!

Learning heuristic playout policies

- Agent builds an internal model of its own *search-enhanced* behaviour.
- The action model attempts to predict the actions recommended by the MCTS.
- This action model can then be used as the heuristic playout policy by μUCT.
- Intuitive idea, does it work? Promising initial results.

Results on Cheese Maze



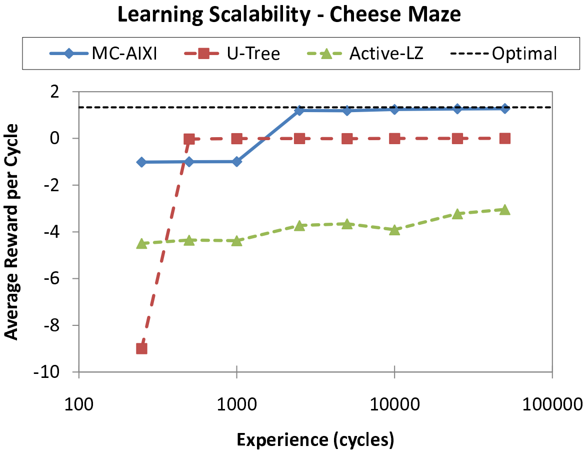
Comments

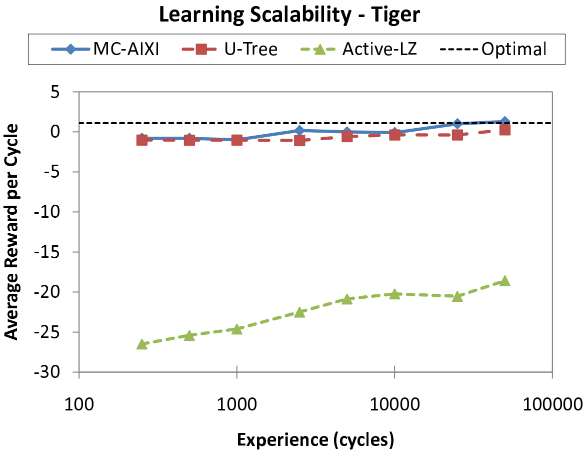
- Obviously a long way to go before we can contruct truly powerful, general agents.
- If Moore's Law continues to hold, AI has an interesting future!
- AI is an arms race. Pointless working on it without access to state of the art hardware.
- Hopefully it will be become more culturally acceptable to do serious research on general agent architectures... if not, well, there are plenty of people who want automated trading agents...

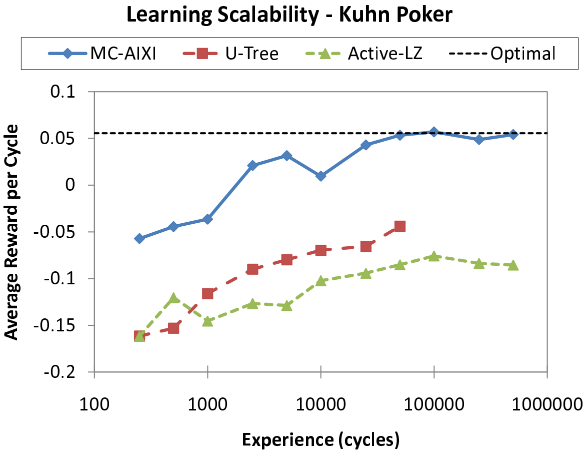
Questions?

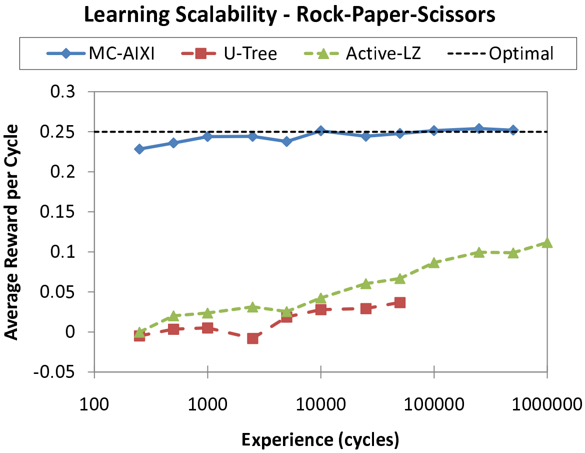
- Thanks for coming, I hope you enjoyed my talk.
- For more information, see:

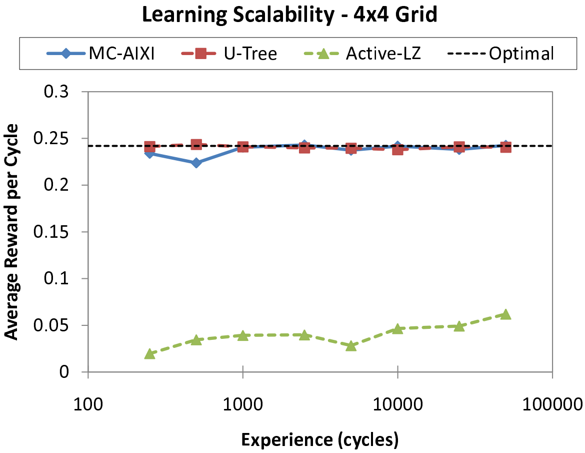
  A Monte Carlo AIXI Approximation,
  Joel Veness, Kee Siong Ng, Marcus Hutter, David Silver
  http://arxiv.org/abs/0909.0801

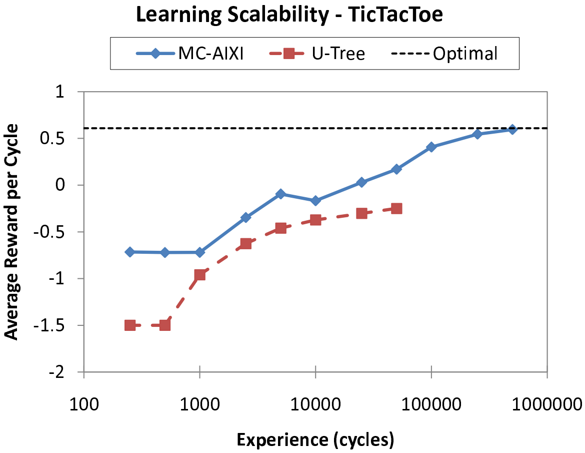

Learning Scalability - Cheese Maze
MC-AIXI
U-Tree
Active-LZ
Optimal
Average Reward per Cycle
2
0
-2
-4
-6
-8
-10
100
1000
10000
100000
Experience (cycles)


Learning Scalability - Tiger
MC-AIXI
U-Tree
Active-LZ
Optimal
Average Reward per Cycle
5
0
-5
-10
-15
-20
-25
-30
100
1000
10000
100000
Experience (cycles)


Learning Scalability - Kuhn Poker
MC-AIXI
U-Tree
Active-LZ
Optimal
Average Reward per Cycle
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
100
1000
10000
100000
1000000
Experience (cycles)


Learning Scalability - Rock-Paper-Scissors
MC-AIXI
U-Tree
Active-LZ
Optimal
Average Reward per Cycle
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
100
1000
10000
100000
1000000
Experience (cycles)


Learning Scalability - 4x4 Grid
MC-AIXI
U-Tree
Active-LZ
Optimal
Average Reward per Cycle
0.3
0.25
0.2
0.15
0.1
0.05
0
100
1000
10000
100000
Experience (cycles)


Learning Scalability - TicTacToe
MC-AIXI
U-Tree
Optimal
Average Reward per Cycle
1
0.5
0
-0.5
-1
-1.5
-2
100
1000
10000
100000
1000000
Experience (cycles)

Questions?

- Thanks for coming, I hope you enjoyed my talk.