Australian National University

A MC-AIXI-CTW Implementation Group Project

Johannes Kirschner
Kerry Olesen
Jesse Wu

October 26, 2013

1 Introduction

The AIXI model [Hut05] is an attempt to solve the general AI problem. The AIXI agent interacts with an environment in cycles. Denote by $\mathcal{A}, \mathcal{O}$ and $\mathcal{R}$ an action, observation and reward space respectively. In each cycle, AIXI takes an action $a \in \mathcal{A}$ and receives an observation $o \in \mathcal{O}$ and a reward $r \in \mathcal{R}$. The goal of the agent is to maximize the total future reward. AIXI does not require any previous knowledge of an environment, actions are chosen based on past perceptions, which are used to build a model of the environment. Let $\mathcal{M}$ be the model class of all chronological semi-computable semi-measures and $K(\rho)$ the Kolmogorov Complexity of ρ. Then AIXI chooses in cycle k an action

$$a_k = \arg\max_{a_k} \sum_{o_k r_k} \dots \max_{a_m} \sum_{o_m r_m} (r_k + \dots + r_m) \sum_{\rho \in \mathcal{M}} 2^{-K(\rho)} \rho(o_1 r_1 \dots o_m r_m | a_1 \dots a_m, \rho)$$

Unfortunately the AIXI model is incomputable. For all practical applications, the agent must be approximated. One approach in approximating AIXI is the MC-AIXI-CTW [VNHS09] model. Here the expectimax search is solved by an Monte-Carlo approach. The UTC [KS06] algorithm is used to balance exploration and exploitation. The class of environment models used in the implementation is a mixture of d-th order Markov Decision Process. Notably, Context Tree Weighting allows efficient linear time computation of this rather general class of models [WST95].

In comparison to AIXI, at cycle k MC-AIXI-CTW selects an action

$$a_k = \arg\max_{a_k} \sum_{o_k r_k} \ldots \max_{a_m} \sum_{o_m r_m} (r_k + \cdots + r_m) \sum_{M \in \mathcal{C}_D} 2^{-\Gamma_D(M)} \rho(o_1 r_1 \ldots o_m r_m | a_1 \ldots a_m, M)$$

Here $\mathcal{C}_D$ is the class of all prediction suffix trees of maximum depth D, and $\Gamma_D(M)$ is the description length of a context tree M.

In the following we present our implementation of the MC-AIXI-CTW model. In Section 2 we explain how to use the program and specify different options. Section 3 describes the results of the model on several experimental environments.

2 User Manual

Our approximation of aixi is written in C++ and requires g++ for compilation.

2.1 Setup

Compile:

cd aixi
make

Run:

./aixi file.conf [--option1=value1 --option2=value2 ...]

Include trained ctw data?? I think this is a good idea Johannes

2.2 Configuration Options

Options can be either specified in the configuration file or passed directly as --option=value to the program. Several configuration files are available, each specifies a particular environment and a set of default options.

Available Options

--environment=env Specifies the environment. Available environments are

- biased_rock_paper_scissor
- coinflip
- kuhn_poker
- pacman
- tiger

--mc-timelimit=N The number N of MC simulations per cycle.

--WRITE-CT=FILE Write CTW to file before agent termination.

--LOAD-CT=FILE Specifies a (trained) CTW for the agent to load at initialisation.

--LOG=FILE

--TERMINATE-AGE=N The number N of agent/environment interaction cycles.

--EXPLORATION=P Probability $0 \leq P \leq 1$ that a action is chosen randomly.

--EXPLORE-DECAY=D Decay $0 \leq D \leq 1$ of exploration constant. P is multiplied by D in each cycle.

--INTERMEDIATE-CT=[1|0] If set to 0 , no intermediate context tree at time $t = 2^k$ is written. Default is 1. (TODO: I actually would prefer default 0 but that's not so important :) Johannes)

3 Experimental Results

3.1 Experimental Setup

TODO: reword the following paragraphs
We evaluate the performance of our agent on five sample environments. For each environment the agent was allowed 100000 cycles to learn a model. During the learning process an exploratory constant was used. The performance of the model after various amounts of experience was then evaluated by running the agent without exploration for 5000 cycles, and the average reward per cycle reported.
The parameters used for each environment are shown in Figure 3.1. The experiments themselves were performed using a 3.47GHz CPU with 4GB of RAM.

Domain	CTW depth	m	ϵ	γ	ρUCT Simulations
Biased Rock-Paper-Scissor	32	4	0.999	0.99999	500
Coinflip	x	x	x	x	x00
Kuhn-Poker	42	2	0.99	0.9999	500
Partial Observable Pacman	96	4	0.9999	0.99999	500
Tiger	96	5	0.99	0.9999	500

Figure 3.1: MC-AIXI-CTW model learning configuration

3.2 Results

Present Results/Graphs of each environment. Maybe mention optimal result and/or scale results accordingly
All environments except for Pacman have a known optimal policy and reward.

3.3 Further Experiments

0.2 Coin vs 0.8 Coin We run the coinflip environment with the following settings:

Domain	CTW depth	m	ϵ	ρUCT Simulations	bias p
Coinflip A	16	2	0.999	100	0.8
Coinflip B	16	2	0.999	100	0.2

The values are chosen to give reasonable speed and accuracy. We run the following experiments:

1. run agent on A with exploration=0.2 and save context tree
2. run agent on A with exploration=0
3. load context tree generated in 1. and run agent on A with exploration=0
4. run agent on B, exploration = 0.2 (expected: same result as in 1.)
5. run agent on B, exploration = 0 (expected: same result as in 2.)
6. load tree 1, run agent on B with exploration = 0
7. load tree 1, run agent on B with exploration = 0.2

TODO: Add plot with results for 1-7

Biased rps vs tiger

Domain	CTW depth	m	ϵ	γ	ρ UCT Simulations
Tiger	32	5	0.999	0.99	500
rps	32	5	0.999	0.99	500

1. run agent in Tiger environment and save context tree
2. load context tree and run Tiger environment with exploration=0
3. run rps environment
4. run rps environment with exploration=0
5. load tiger context tree and run rps environment
6. load tiger context tree and run rps environment with exploration=0

TODO: Add plot with results for 1-6

3.4 Discussion

How well did it do.
Include statistics about cycles required for optimal performance, time per cycle as in the VNHS paper [VNHS09]. Also note the number of simulations required at each cycle for near optimal performance.

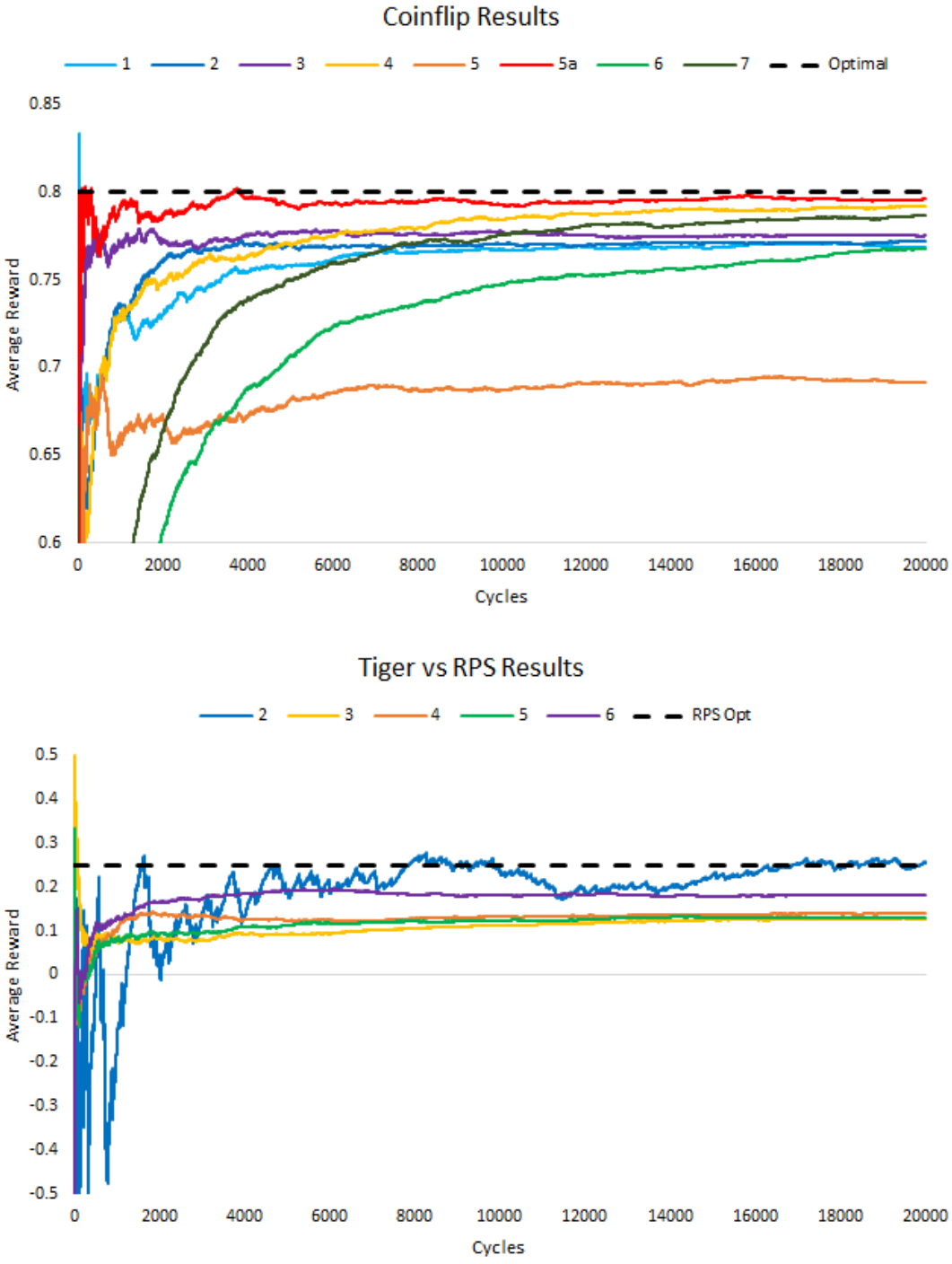

Figure 3.2: Further experiments

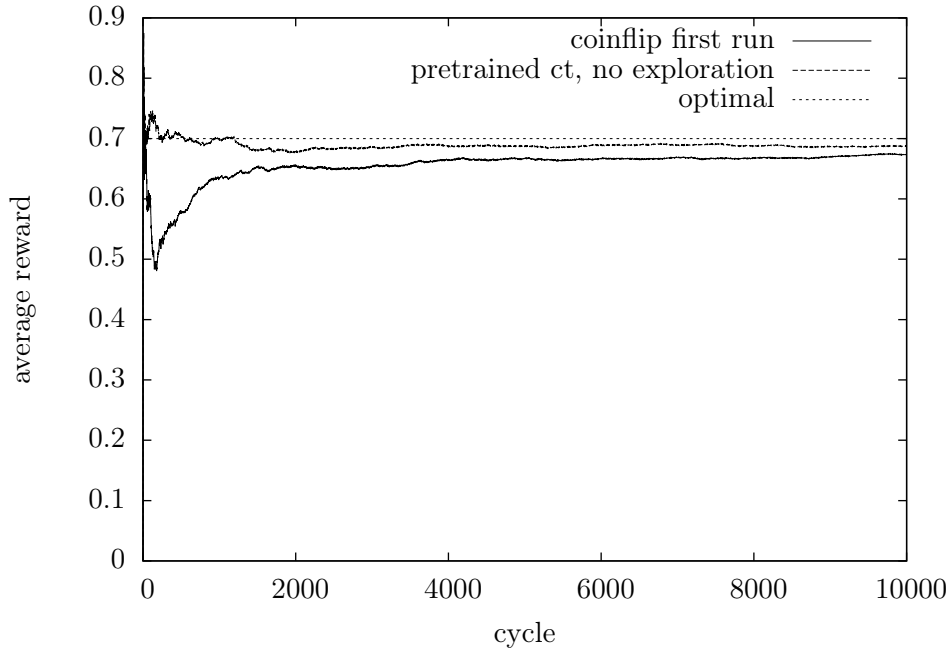

Figure 3.3: Results on the coinflip environment

References

[Hut05] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2005.

[KS06] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

[VNHS09] Joel Veness, Kee Siong Ng, Marcus Hutter, and David Silver. A monte carlo aixi approximation. *CoRR*, abs/0909.0801, 2009.

[WST95] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.

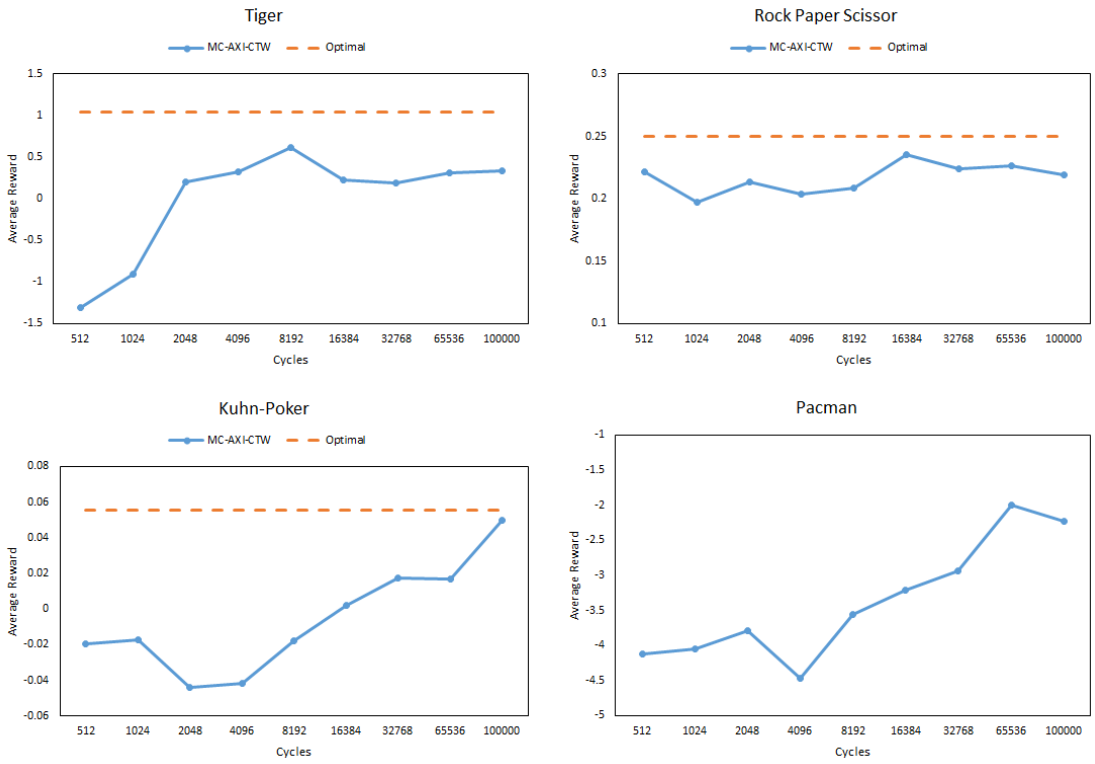

Figure 3.4: Average Reward per Cycle vs Experience