# Applying Context-Tree Weighting To Learning in Various Environments

**John Aslanides, Ryk Budzynski, Jarryd Martin, Nrupenda Rao, Yadunandan Sannappa, Cheng Yu**

COMP4620/COMP8620
The Australian National University

October 2015
Draft Copy – 10 October 2015

# Abstract

Put your abstract here.

# Contents

# List of Figures

# List of Tables

# Introduction

Hutter has proven that AIXI model produces better result than any other existing learning algorithm. The Bayesian Mixture model will converge to the unknown true model. However, Kolmogorov complexity is incomputable. Using Context Tree Weighting and Monte Carlo sampling as approximation, we can build a universal reinforcement learning agent based on approximation of AIXI.

## 1.1 Main Finding

Group 3 has implemented the reinforcement agent that can learn from various encoded environments.

## 1.2 Introduction

This assignment is closely related to Veness's paper. We are building a reinforcement learning agent using UCT algorithm, MC sampling, AIXI, and universal Bayesian mixture. The agent then learns to play Pacman, Tic-Tac-Toe, Biased Rock-Paper-Scissor, Extended Tiger, and Cheese Maze.

## 1.3 Thesis Outline

Chapter **??** mainly deals with the theoretical findings of Vanness' paper, and briefly touches on Monte Carlo sampling, UTC algorithm, context tree weighting [V+11], and how our agent combines these, as required by the assignment. Chapter 2 Chapter 3, Chapter 4, and Chapter 5.

Chapter 2

# Design and Implementation

Our team divides the assignment into three parts: execution, learning, environment, and report.

## 2.1 Roles

Execution is for the agent to choosing the most optimal action using $\rho UTC$ algorithm using the updated probability from the root node of the context tree. This part is done by Ryk and John.

Learning is about using the algorithm mentioned in Venness's paper. Using the observation from the environment to update the KT estimator for each node in the context tree. This part is handled by Yadu and Jarryd.

Environment part of the project is about encoding different games into structured format to dole out reward and provides observation towards the agent. This part is handled by Rao.

Report summaries what the group understands and has implemented for the assignment. This part is done by Cheng.

## 2.2 Smart Design

## 2.3 Summary

Same as the last chapter, summary what you discussed in this chapter and be the bridge to next chapter.

Chapter 3

# Experimental Methodology

## 3.1 Software platform

We use C++ standard libraries. Due to time and library constraints, we decide not to visualise the cheese maze and pacman.

See Appendix for implementation documentation including class diagram.

Table 3.1 shows how to include tables and Figure 3.1 shows how to include codes.

| Architecture | Pentium 4 | Atom D510 | i7-2600 |
|---|---|---|---|
| Model | P4D 820 | Atom D510 | Core i7-2600 |
| Technology | 90nm | 45nm | 32nm |
| Clock | 2.8GHz | 1.66GHz | 3.4GHz |
| Cores × SMT | 2 × 2 | 2 × 2 | 4 × 2 |
| L2 Cache | 1MB × 2 | 512KB × 2 | 256KB × 4 |
| L3 Cache | none | none | 8MB |
| Memory | 1GB DDR2-400 | 2GB DDR2-800 | 4GB DDR3-1066 |

Table 3.1: Processors used in our evaluation.

```c
int main(void)
{
  printf("Hello World\n");
  return 0;
}
```

(a)

```java
void main(String[] args)
{
  System.out.println("Hello World");
}
```

(b)

Figure 3.1: Hello world in Java and C.

Chapter 4

# Results

## 4.1 Pacman

### 4.1.1 Agent Performance

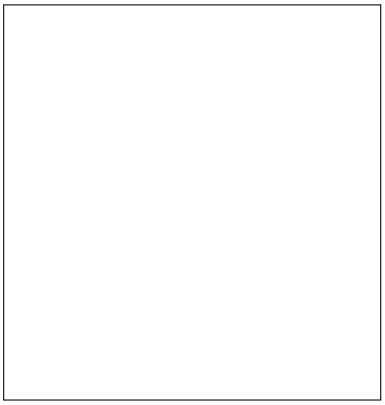


Figure 4.1: This is a placeholder.

### 4.1.2 Analysis

## 4.2 Tic-Tac-Toe

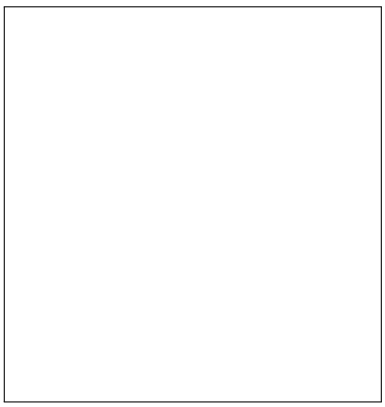


Figure 4.2: This is a placeholder.

### 4.2.1 Agent Performance

### 4.2.2 Analysis

## 4.3 Biased Rock-Paper-Scissor

### 4.3.1 Agent Performance

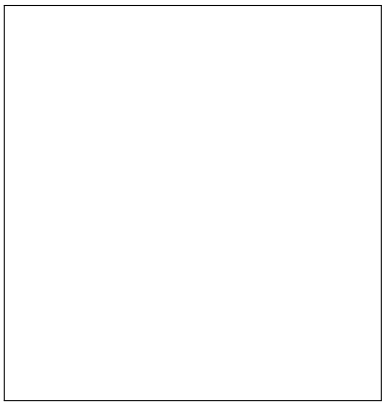


Figure 4.3: This is a placeholder.

### 4.3.2 Analysis

## 4.4 Extended Tiger

### 4.4.1 Agent Performance

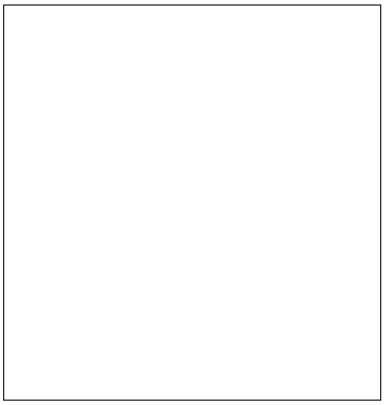


Figure 4.4: This is a placeholder.

### 4.4.2 Analysis

## 4.5 Cheese Maze

### 4.5.1 Agent Performance

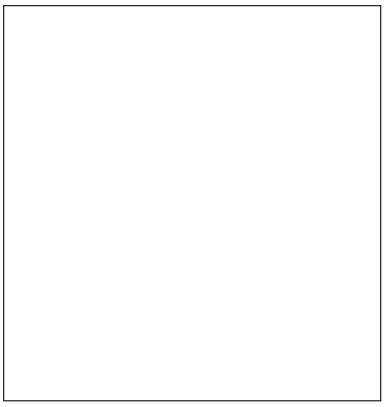


Figure 4.5: This is a placeholder.

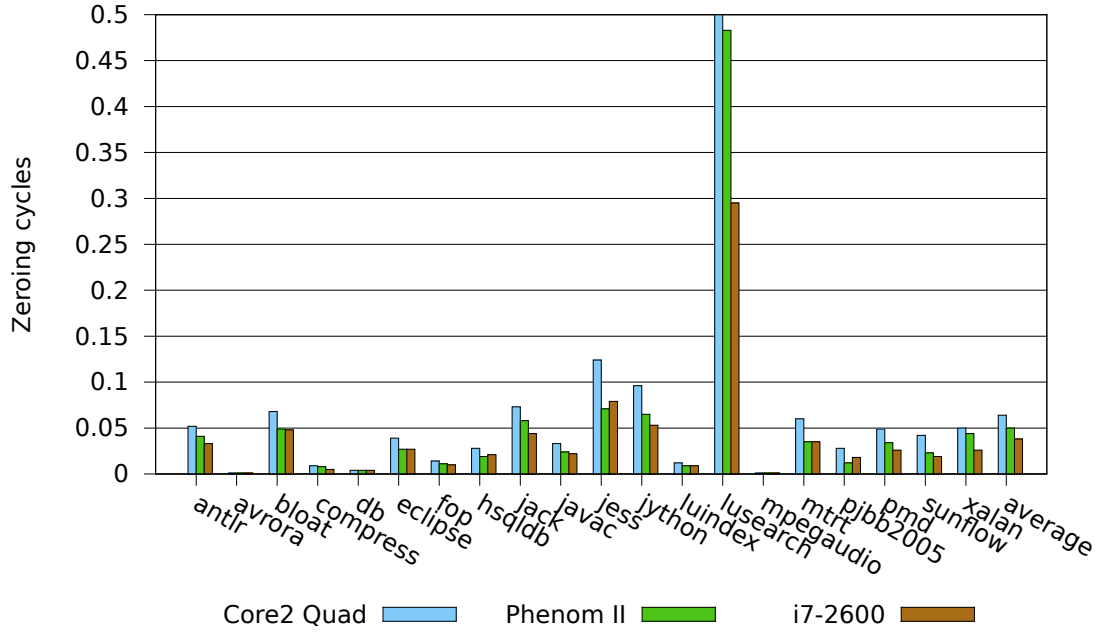
### 4.5.2 Analysis

## 4.6 Additional Questions

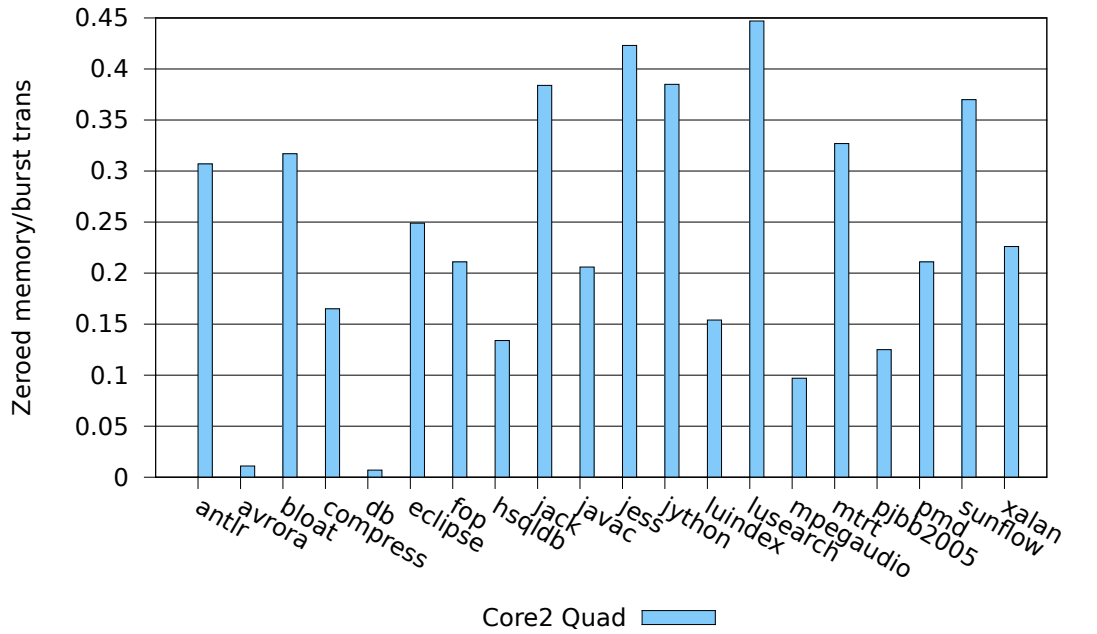
1. Train AIXI on d1 and then continue with d2 (without resetting the CTW and UCT). Is AIXI performing better or worse in learning d2, after having been biased towards d1, compared to training on d2 from scratch?
2. Now go back to d1 and train AIXI on d1 again (without resetting the CTW and UCT). Does AIXI remember d1 and how fast does it recover it?

Here is the example to show how to include a figure. Figure 4.3 includes two subfigures (Figure 4.3(a), and Figure 4.3(b));

## 4.7 Summary




(a) Fraction of cycles spent on zeroing




(b) BytesZeroed / BytesBurstTransactionsTransferred

Figure 4.6: The cost of zero initialization

Chapter 5

# Conclusion

Summary your thesis and discuss what you are going to do in the future in Section 5.1.

## 5.1 Future Work

Good luck.