Enhancements for Monte Carlo Tree Search in The Mario AI Framework

Emil Juul Jacobsen
ejuu@itu.dk

Rasmus Greve
ragr@itu.dk

May 22, 2013

Abstract

(Bare copy/paste-ish fra projektbasen, skal skrives om!) In this experiment we explore different implementations and enhancements of the Monte Carlo Tree Search algorithm for an AI, in order to evaluate their performance and results in the Super Mario AI Benchmark tool. We have implemented the basic MCTS algorithm in the Mario AI Framework and characterised the performance and identification of the strengths and weaknesses of the algorithm relative to the framework. We have identified a set of refinements and alterations of the algorithm and through implementation and evaluation of these individually we came up with compositions that greatly increase the performance of the AI.

1 Introduction

In this experiment we explore different implementations and enhancements of the Monte Carlo Tree Search algorithm for an AI, in order to evaluate their performance and results in the Super Mario AI Benchmark tool.

2 Background

- Om MCTS [1]
- Om UCB og UCT [1] måske også [4]
- (kort!) Om The Mario AI Framework [5]

3 Approach and Improvements

3.1 Monte Carlo Tree Search with UCT

Kilde [1]

3.2 Domain knowledge

3.2.1 Limited actions

Hvis vi er nødt til at begrænse ham til ikke at bruge ↓ til alle de andre implementationer skal det fremgå her!

3.2.2 Hole detection

Hvis vi er nødt til at bruge hulgenkendelse til alle de andre implementationer skal det fremgå her!

3.3 Softmax Backup

$$exploitation = Q * maxReward + (1 - Q) * averageReward \tag{1}$$

We use equation 1 to calculate the exploration part for the confidence of nodes

3.4 Macro actions

[3]

3.5 Heuristic Partial Tree Expansion Policy

3.6 Checkpoints

3.7 (Combination)

4 Results

Her er noget mere tekst, reference til figur 1

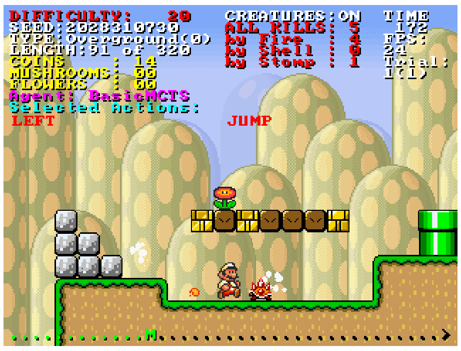
Method	**Score**
Softmax backup q = 0 (UCT)	34,162
Softmax backup q = $\frac{1}{8}$	-
Softmax backup q = $\frac{1}{4}$	34,387
Softmax backup q = $\frac{1}{2}$	34,147
Softmax backup q = 1	26,842

Table 1: Results of using Softmax backup with different q values

Method	**Avg. number of nodes**	**Score**
MCTS w/ UCT, limit = 0	-	-
MCTS w/ UCT, limit = 1	-	-
MCTS w/ UCT, limit = 2	-	-
MCTS w/ UCT, limit = 4	-	-
MCTS w/ UCT, limit = 8	-	-
MCTS w/ UCT, limit = 16	-	-
MCTS w/ UCT, limit = ∞	-	-

Table 2: Results of using UCT with a different limit for random moves

5 Conclusion



```
DIFFICULTY:    20       CREATURES:ON   TIME
SEED:20283310730        ALL KILLS: 5    172
TYPE:Overground(0)      by Fire   : 4  FPS:
LENGTH:91 of 320        by Shell  : 0  24
COINS     : 14          by Stomp  : 1  Trial:
MUSHROOMS: 00                          1(1)
FLOWERS   : 00
Agent: BasicMCTS
Selected Actions:
 LEFT                    JUMP
```

Figure 1: Mario being followed

References

[1] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of monte carlo tree search methods,” Computational Intelligence and AI in Games, IEEE Transactions on, vol. 4, no. 1, pp. 1–43, 2012.

[2] S. R. K. Branavan, D. Silver, and R. Barzilay, “Non-linear monte-carlo search in civilization ii,” in Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three, 2011, pp. 2404–2410.

[3] E. J. Powley, D. Whitehouse, and P. I. Cowling, “Monte Carlo Tree Search with macro-actions and heuristic route planning for the Physical Travelling Salesman Problem,” in Computational Intelligence and Games (CIG), 2012 IEEE Conference on, 2012, pp. 234–241.

[4] T. Pepels and M. H. Winands, “Enhancements for Monte-Carlo Tree Search in Ms Pac-Man,” in Computational Intelligence and Games (CIG), 2012 IEEE Conference on, 2012, pp. 265–272.

[5] S. Karakovskiy and J. Togelius, “The mario ai benchmark and competitions,” Computational Intelligence and AI in Games, IEEE Transactions on, vol. 4, no. 1, pp. 55–67, 2012.