5: MONTE CARLO TREE SEARCH

COMP702: CLASSICAL ARTIFICIAL INTELLIGENCE

Monte Carlo methods

Expected value

- Let X be a **random variable**
- Let $p(x)$ be the **probability** that X has value x
- Then the **expected value** of X is

$$\sum_{x} x\, p(x)$$

Expected value – example

- Suppose that a slot machine pays out
  £1 with probability 0.05,
  £5 with probability 0.03,
  £10 with probability 0.02,
  nothing with probability 0.9.
- The expected payout is
  $$1 \times 0.05 + 5 \times 0.03 + 10 \times 0.02 + 0 \times 0.9 = 0.4$$
- On average, if you play the machine N times, you will win $N \times £0.40$

“Randomness” in computing

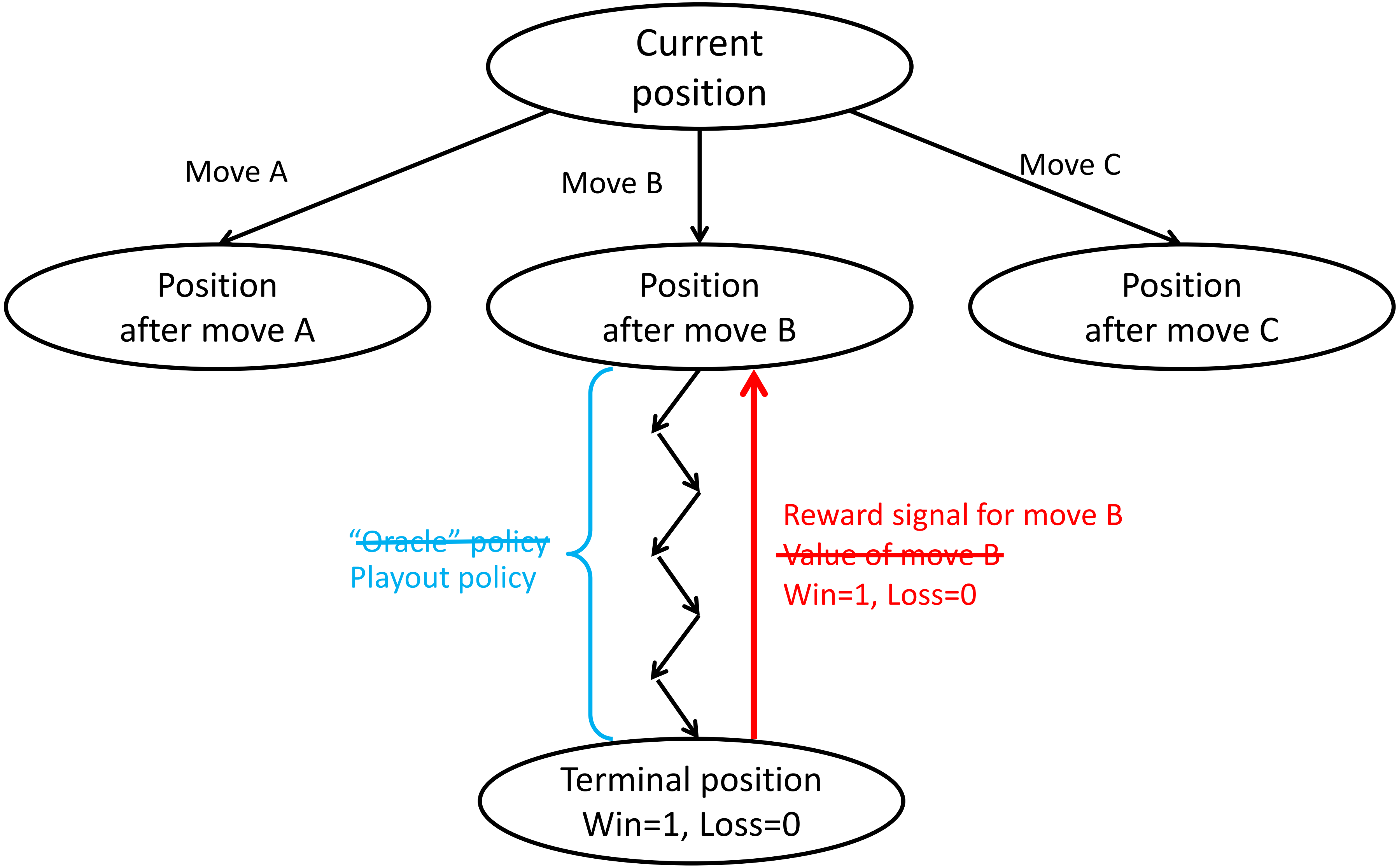
- Digital computers are **deterministic**, so there's no such thing as true randomness
  - Cryptographically secure systems use an external source of randomness e.g. atmospheric noise, radioactive decay
- What we actually have are **pseudo-random number generators (PRNGs)**
- A PRNG is an algorithm which gives an **unpredictable** sequence of numbers based on a **seed**
- Sequence is **uniformly distributed**, i.e. all numbers have equal probability
- Seed is generally based on some source of **entropy**, e.g. system clock, mouse input, electronic noise

Monte Carlo methods

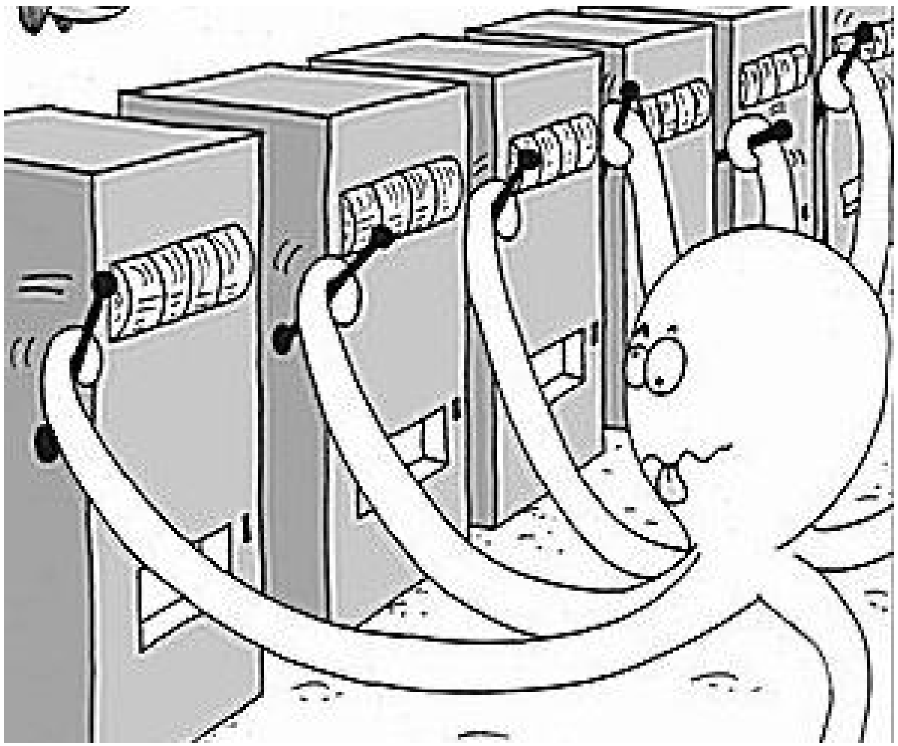
- In computing, a **Monte Carlo method** is an algorithm based on **averaging over random samples**
- The **average** over a large number of samples is a good approximation of the **expected value**
- Used for **quickly approximating** quantities over **large domains**
- Generally designed to **converge in the limit**
  - An **infinite** number of samples would give an **exact** answer
  - As the **number of samples** increases, the **accuracy** of the answer improves
- Applications in physics, engineering, finance, weather forecasting, graphics, ...

Monte Carlo Tree Search

Game decisions revisited



The multi-armed bandit problem



At each step pull one arm

Noisy/random reward signal

In order to:
- Minimise regret
- Maximise expected return
- (Find the best arm)

Upper Confidence Bound (UCB1)

- Balance **exploitation** with **exploration**
- Select the arm that maximises

Total reward from this arm so far

Total number of trials so far

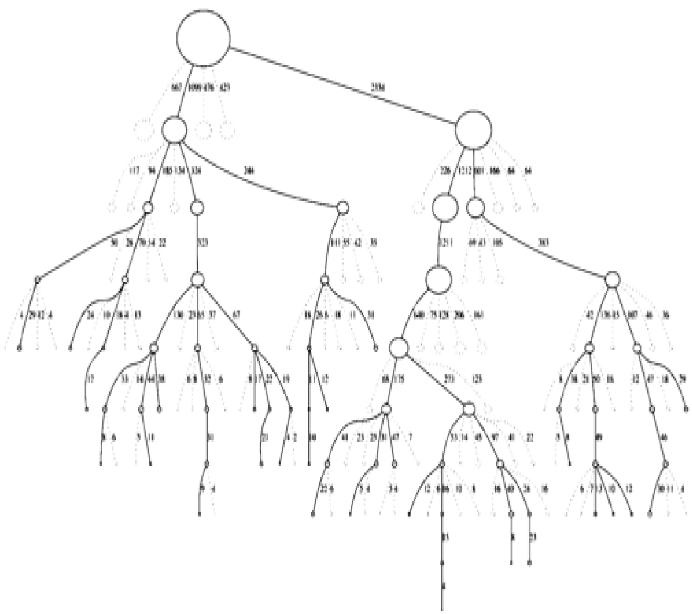
$$\underbrace{\frac{V_a}{n_a}}_{\text{Exploitation}} + k \underbrace{\sqrt{\frac{\log n}{n_a}}}_{\text{Exploration}}$$

Number of times this arm has been selected so far

Tuning constant

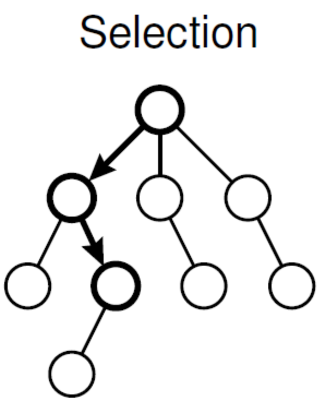
P Auer, N Cesa-Bianchi, P Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*. 47(2):235-256, 2002.

Monte Carlo Tree Search (MCTS)



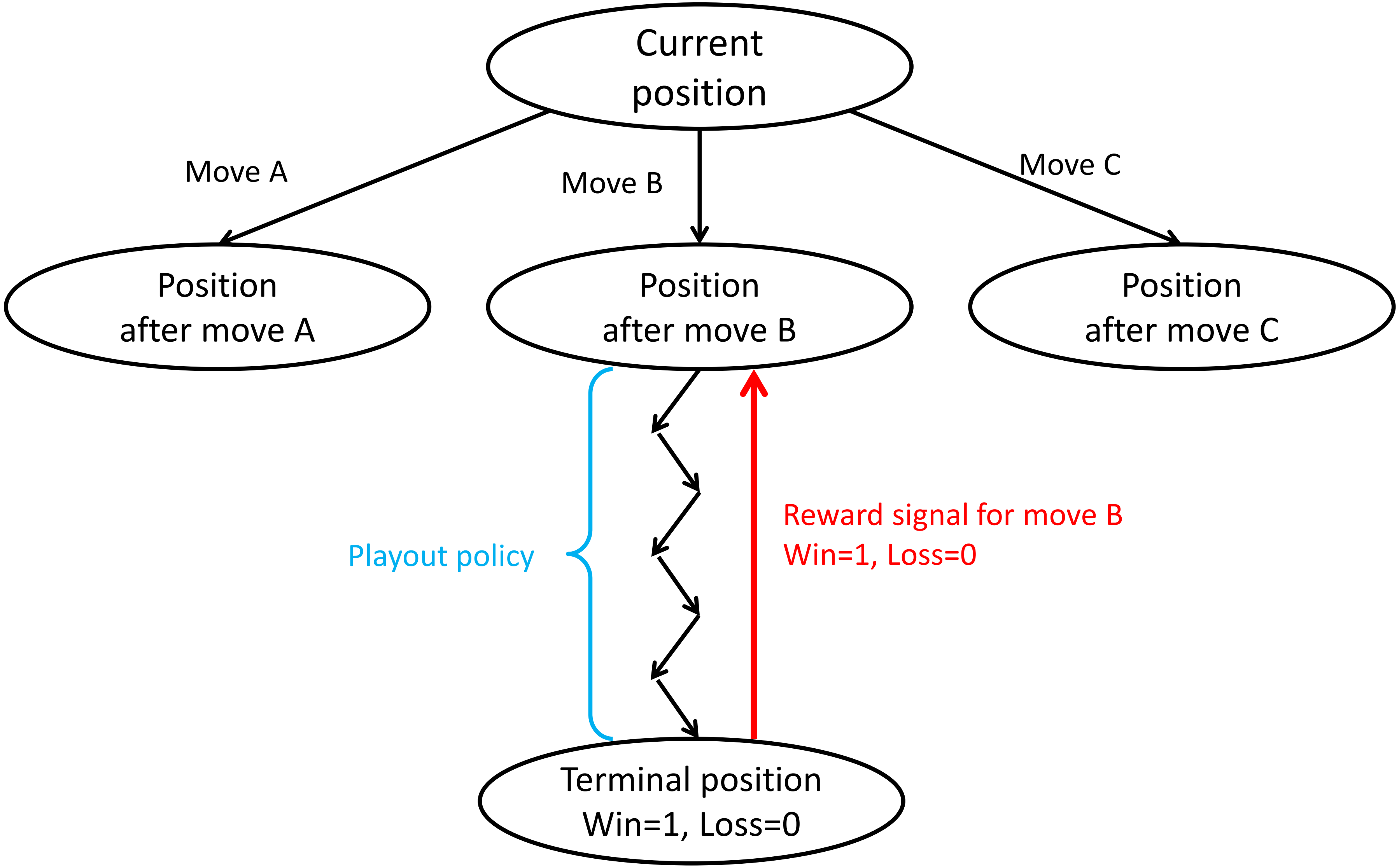
- Iteratively build a partial search tree, one node per iteration
- Monte Carlo evaluation (random playouts) for nonterminal states
- **Asymmetric**
  - balance exploitation with exploration
- **Anytime**
  - can do as many (or as few) iterations as we want
- **Aheuristic**
  - Monte Carlo evaluation requires only a forward model

Monte Carlo Tree Search (MCTS)

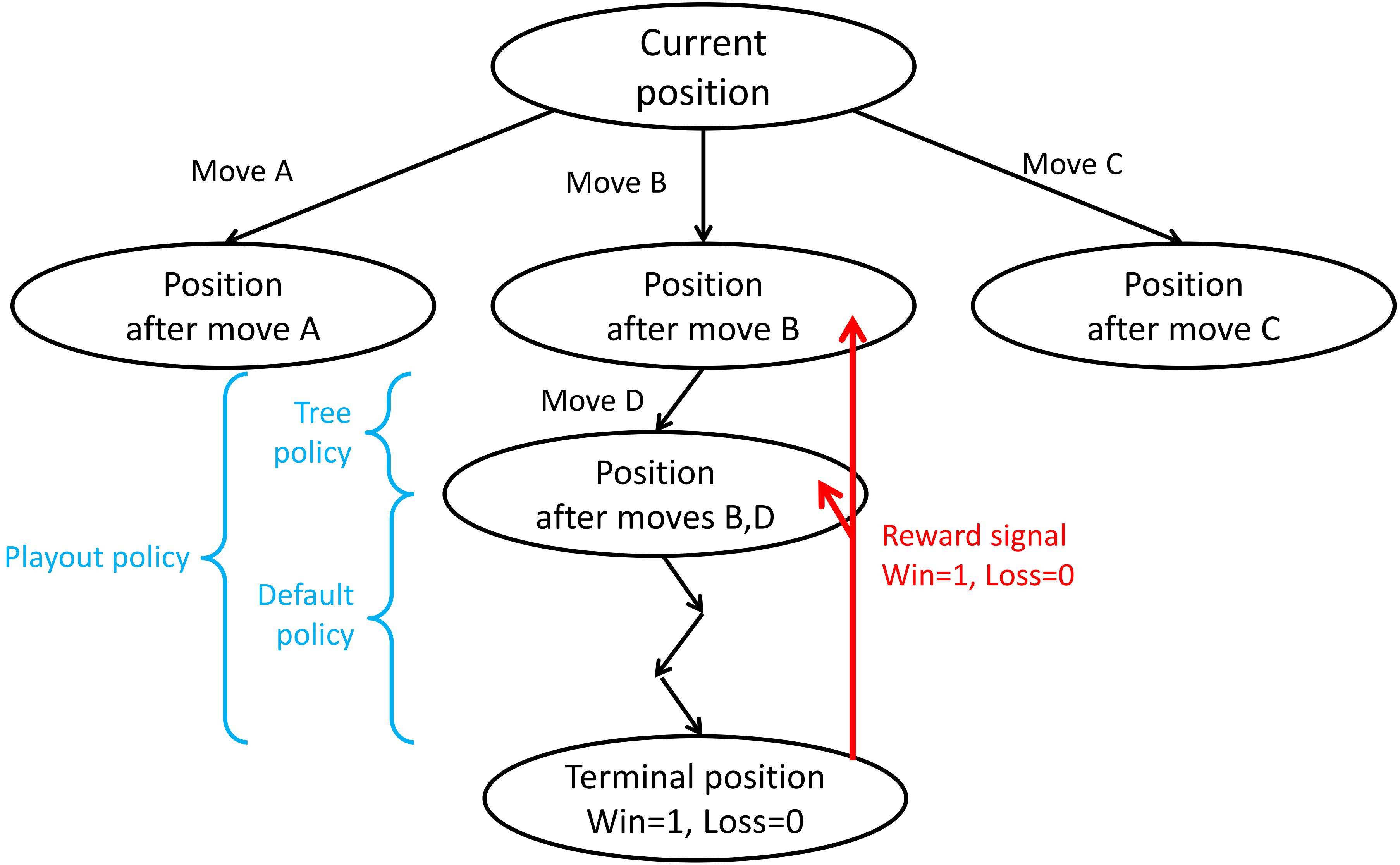


Choose a node with unexpanded children

Game Decisions



Monte Carlo Tree Search (MCTS)



Upper Confidence bound for Trees (UCT)

- Tree policy: UCB1 $\frac{V_a}{n_a} + k\sqrt{\frac{\log n}{n_a}}$

- Default policy: uniform random

L Kocsis, C Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, vol 4212 of *Lecture Notes in Artificial Intelligence,* 282-293, 2006.

Demo


MB
GAMES
Connect 4
The vertical strategy game!
7-Adult
2 players
Drop your counters into the slots. Try to outwit your opponent and be the first to get a row of 4. Not as easy as it looks!

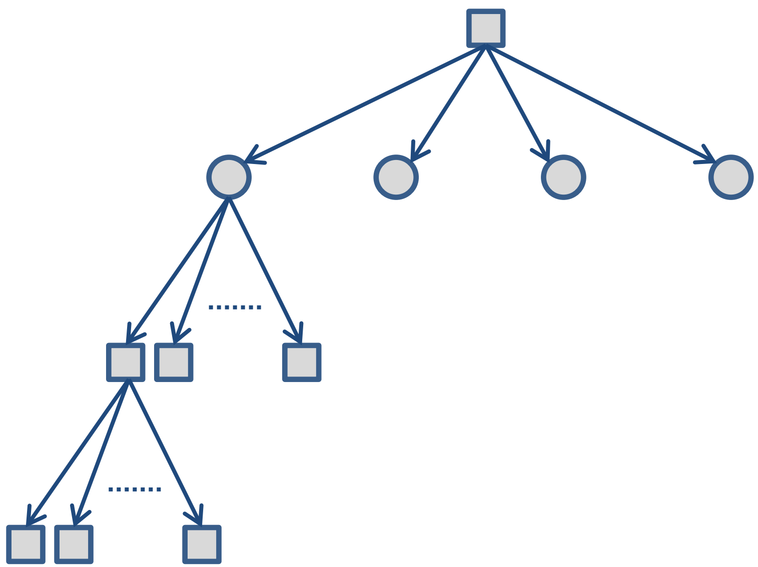
MCTS for games of imperfect information

Peter I. Cowling, Edward J. Powley and Daniel Whitehouse.
Information Set Monte Carlo Tree Search.
IEEE Transactions on Computational Intelligence and AI in Games, 4(2):120–143, 2012.

Imperfect information

- Stochasticity
- Information asymmetry
- Partial observability

An explosion in branching factor



4 possible plays by me

50 possible random card draws

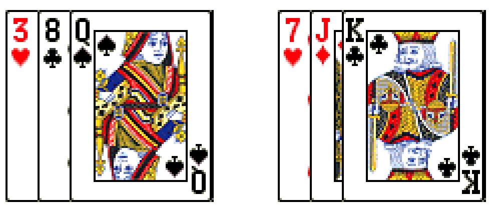
${}_{40}C_3$ = 9880 different opponent plays

...

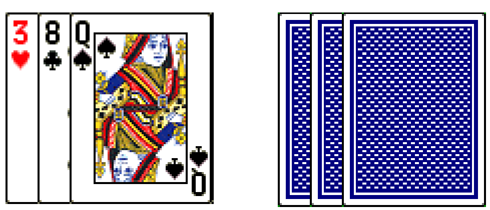
Information sets

- Observation gives a **set of possible states**, one of which is the actual state of the game

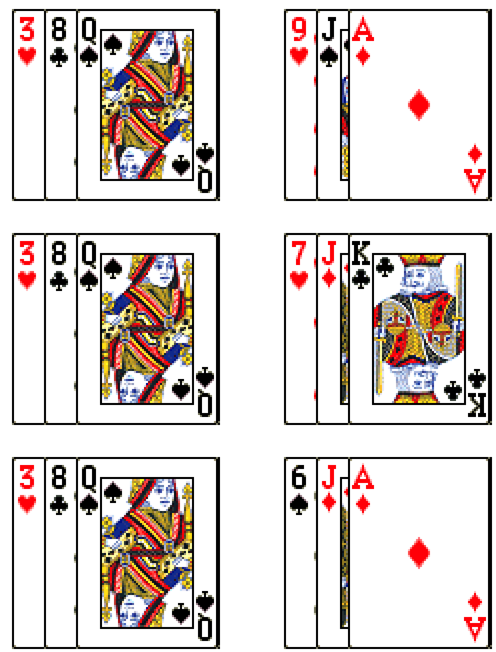
Actual state:



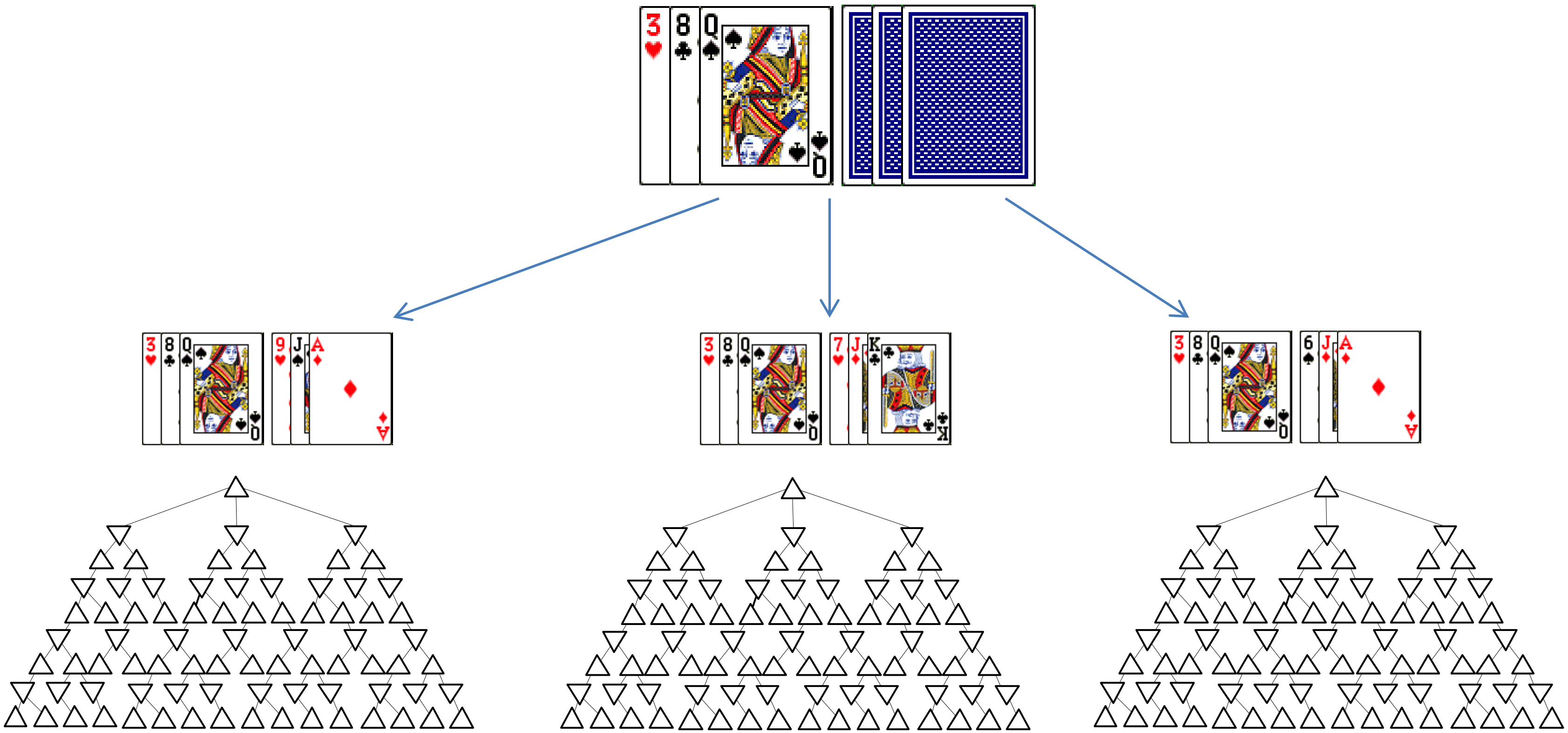
Observation:



Information set:

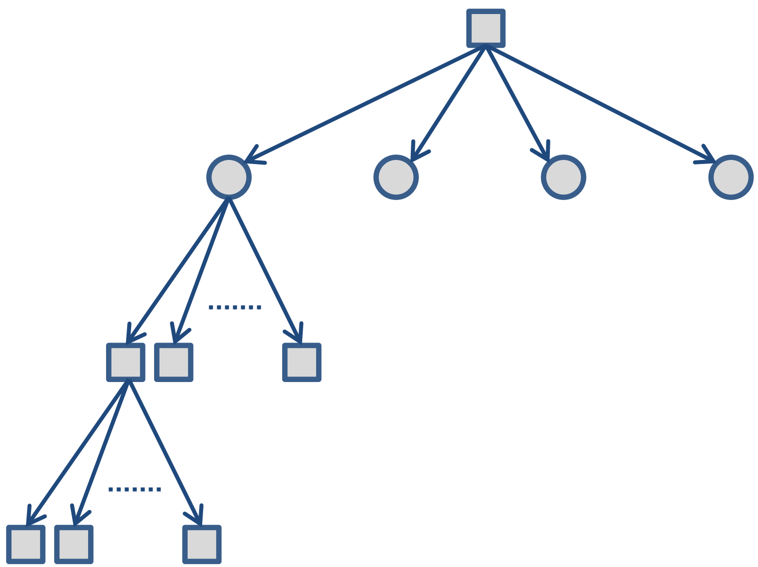
{  , , , …}

Determinization



"**Averaging over clairvoyance**" [Russell and Norvig 2009]

An explosion in branching factor



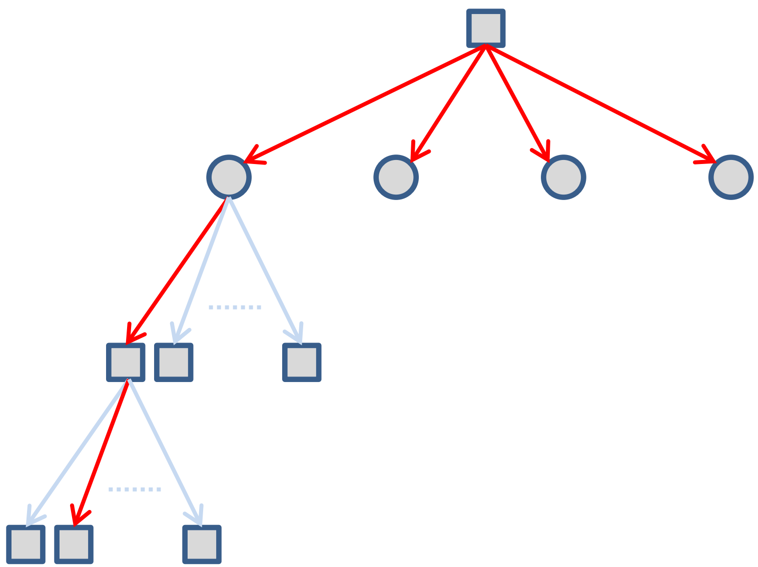
4 possible
plays by me

50 possible random
card draws

$_{40}C_3$ = 9880 different
opponent plays

...

A reduction in branching factor



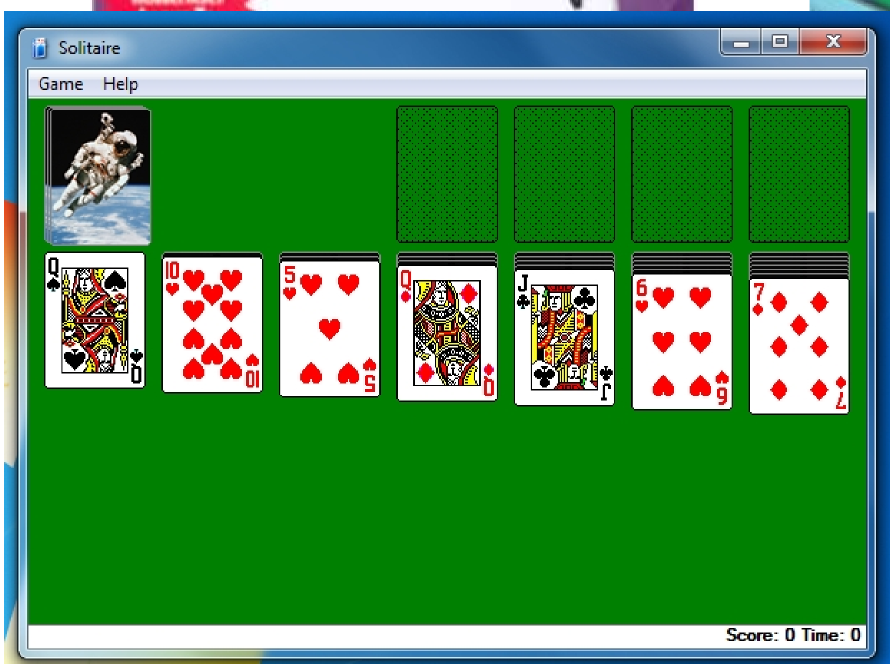
4 possible plays by me

1 guessed random card draw (determinization)

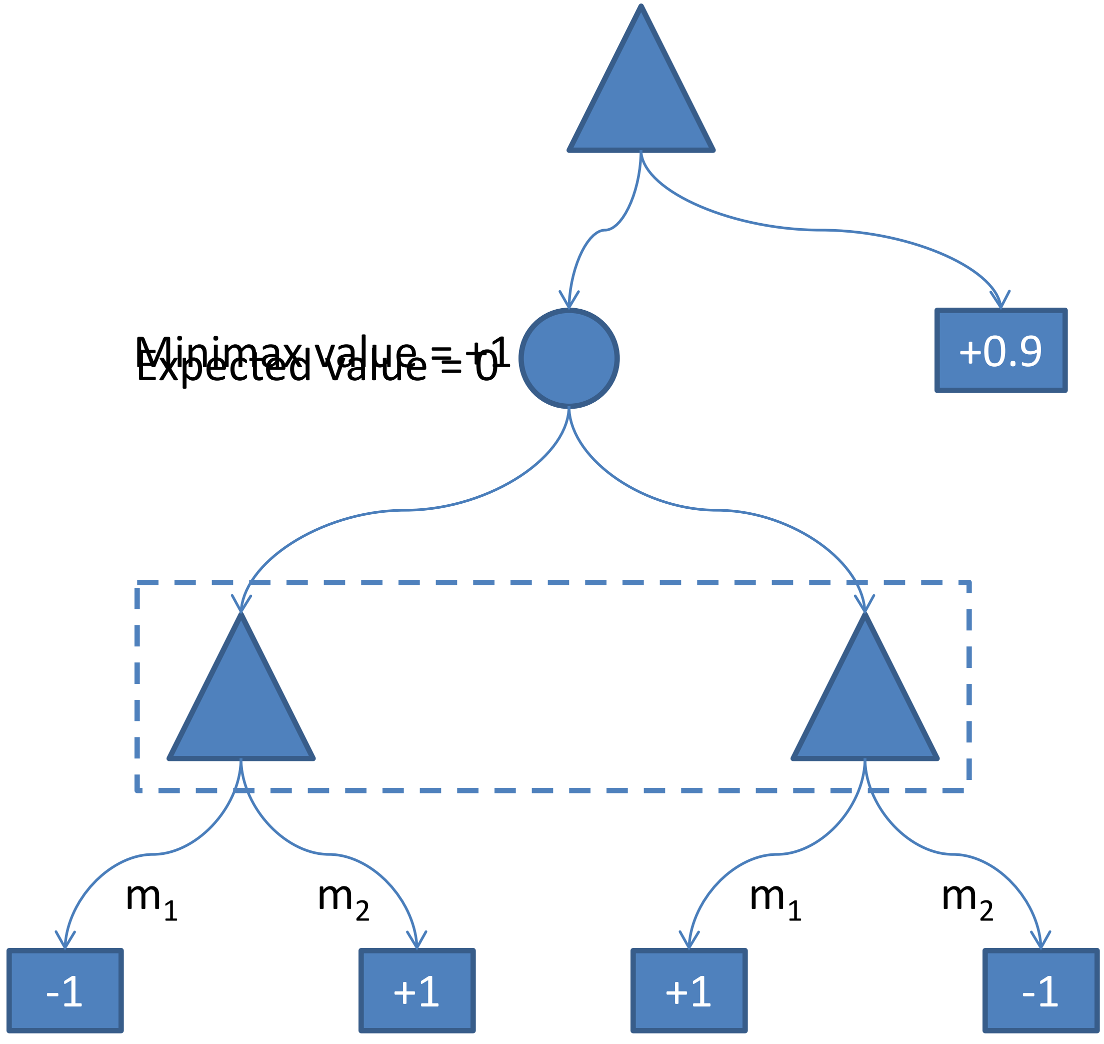
${}_4C_3$ = 4 different opponent plays

...

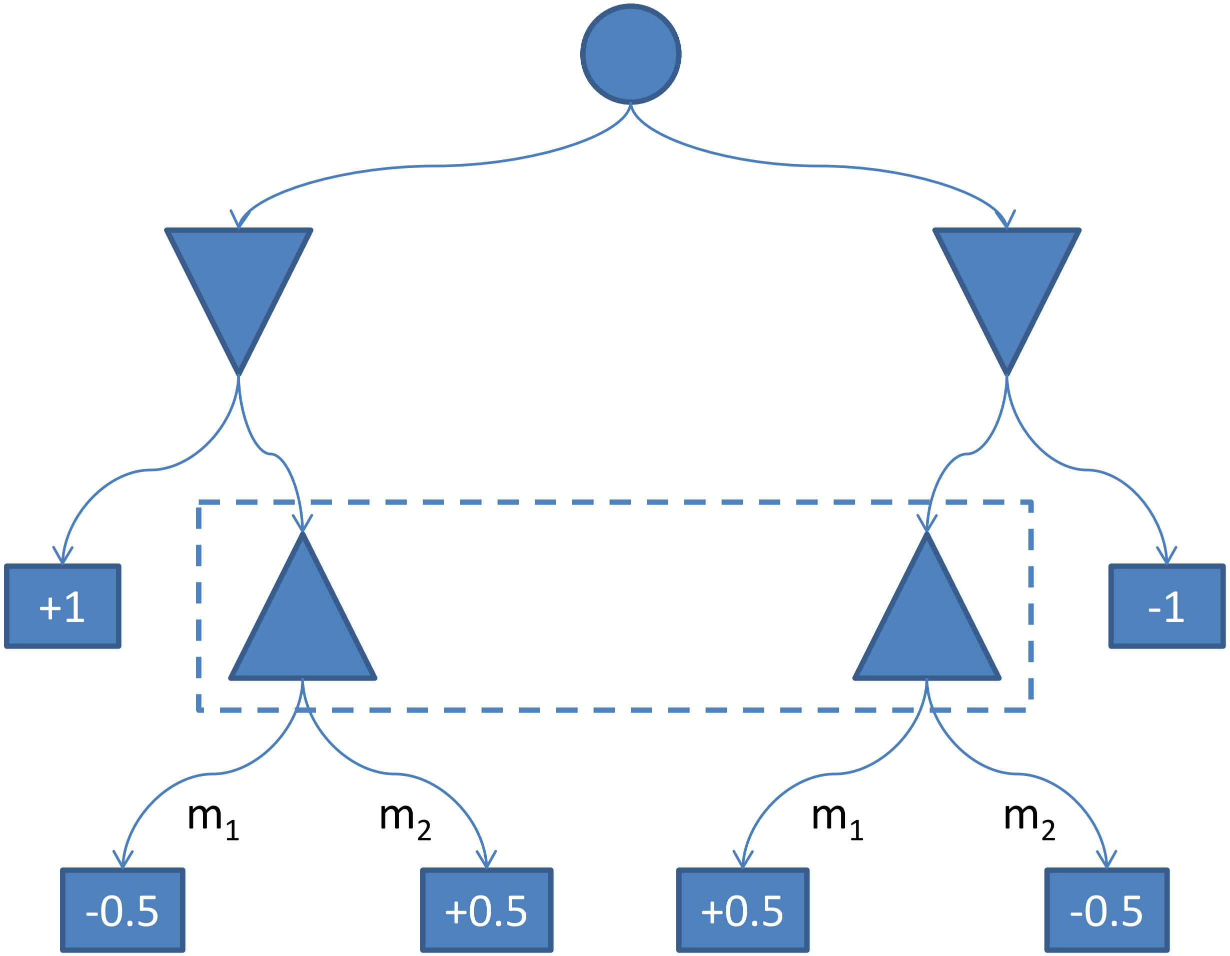
Successes for determinization



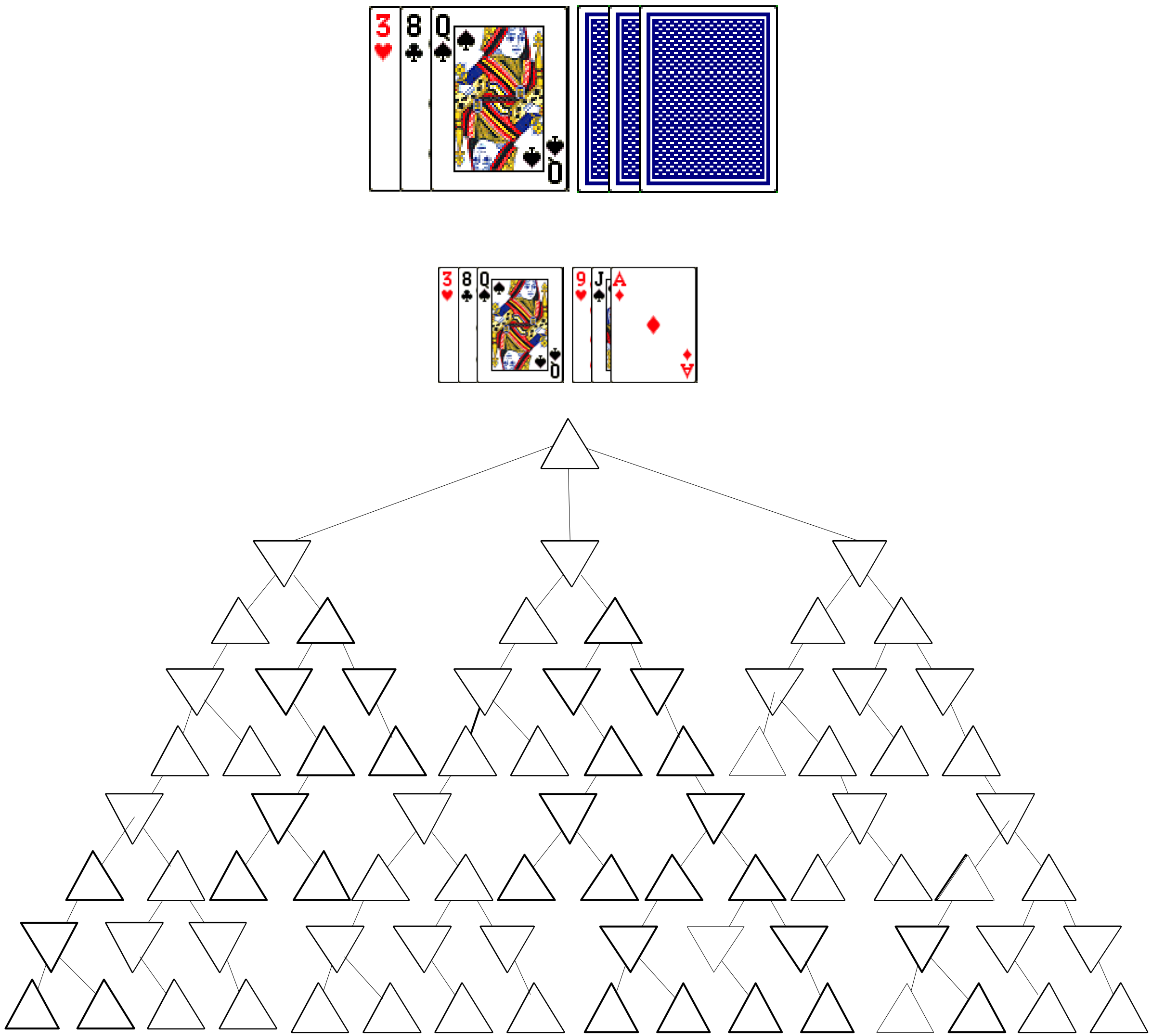
Strategy fusion



Non-locality

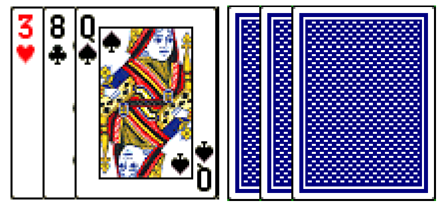


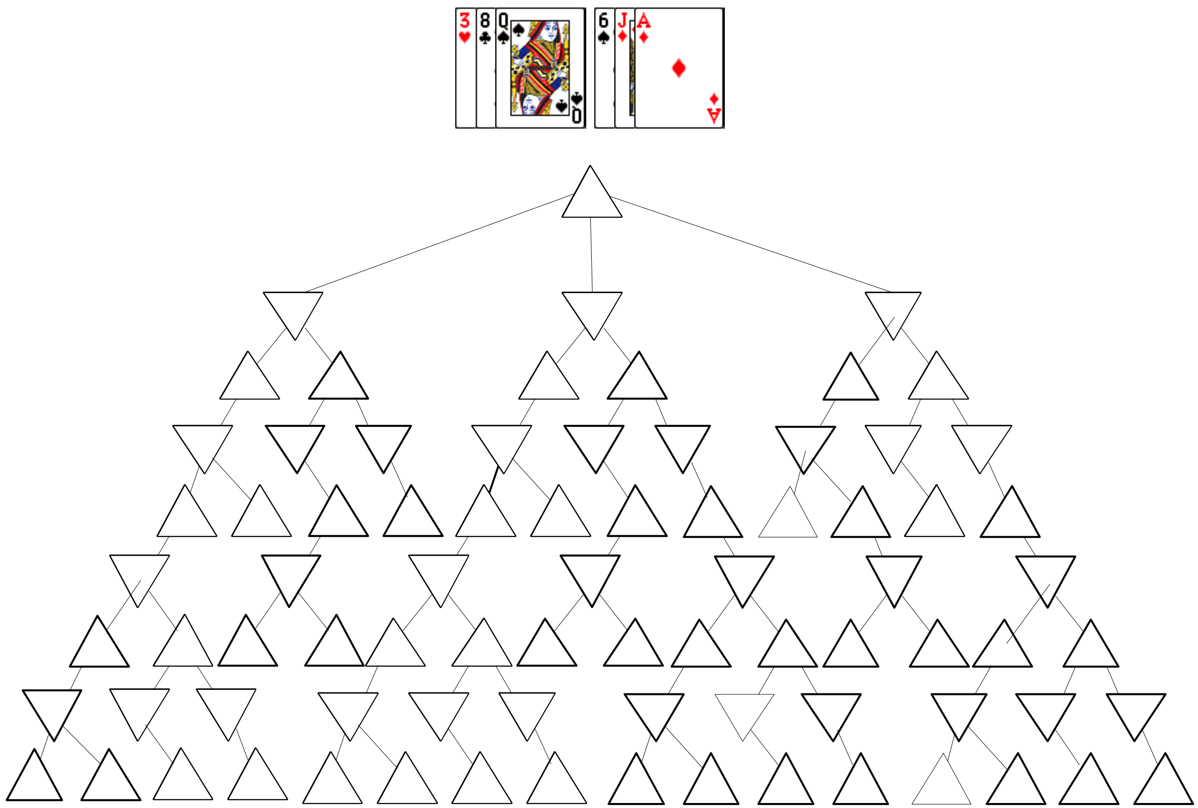
Information Set MCTS (ISMCTS)



P. I. Cowling, E. J. Powley, D. Whitehouse. *Information Set Monte Carlo Tree Search*. IEEE Transactions on Computational Intelligence and AI in Games, 4(2):120-143, 2012.

Information Set MCTS (ISMCTS)

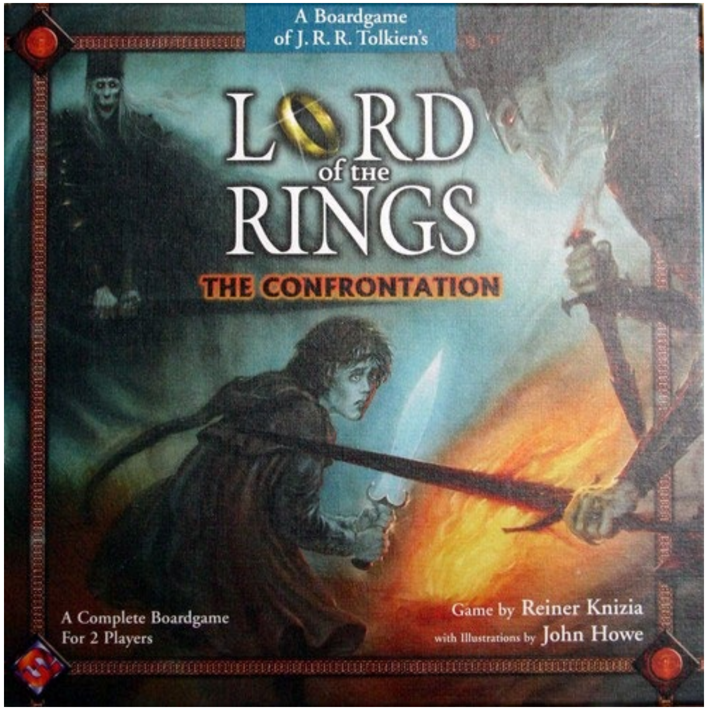


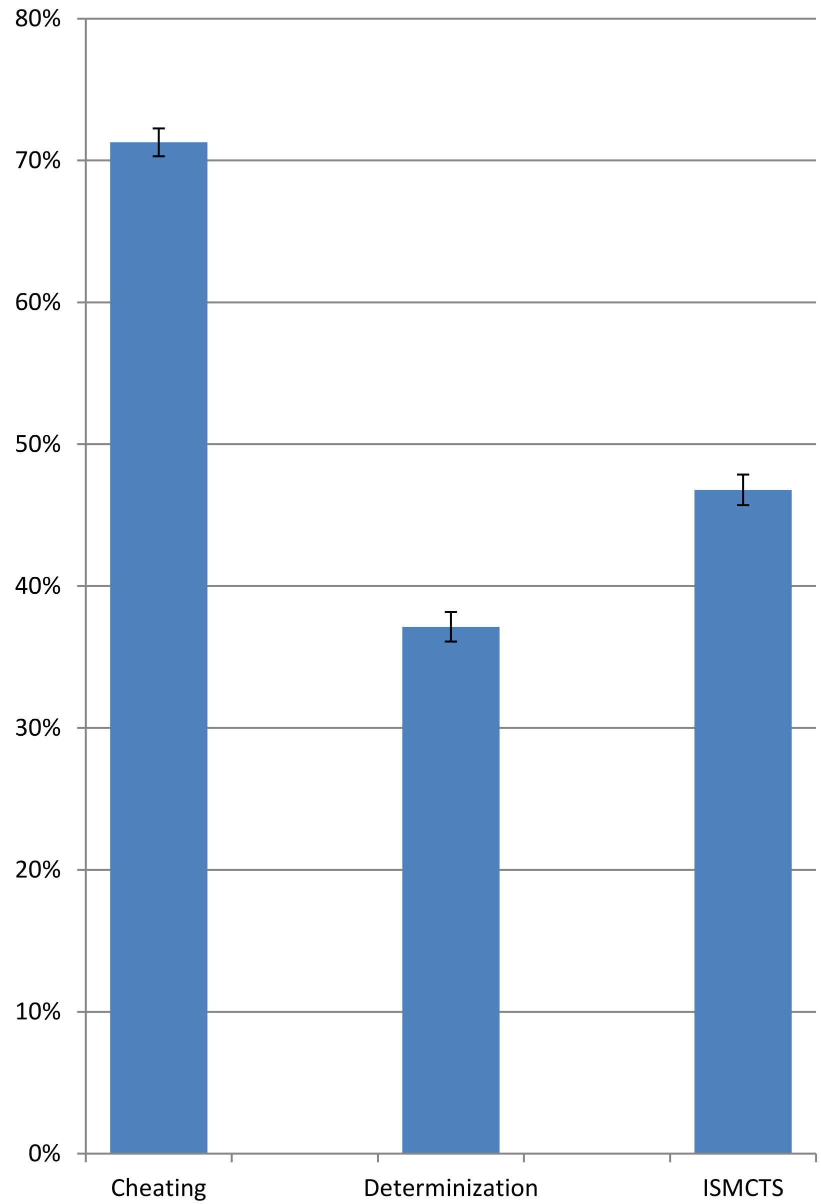


P. I. Cowling, E. J. Powley, D. Whitehouse. *Information Set Monte Carlo Tree Search*. IEEE Transactions on Computational Intelligence and AI in Games, 4(2):120-143, 2012.

Information Set MCTS (ISMCTS)

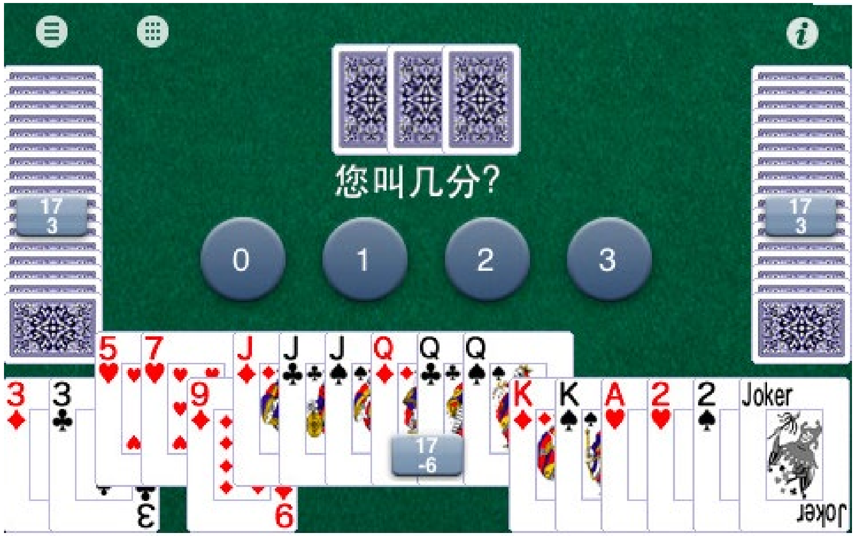
- Does not suffer from strategy fusion
- Beats determinization in
  - LOTR: The Confrontation
  - Phantom m,n,k games (noughts and crosses with hidden moves)
  - Hearts


A Boardgame
of J. R. R. Tolkien's
LORD of the RINGS
THE CONFRONTATION
A Complete Boardgame
For 2 Players
Game by Reiner Knizia
with Illustrations by John Howe


80%
70%
60%
50%
40%
30%
20%
10%
0%
Cheating
Determinization
ISMCTS

Dou Di Zhu

- Popular in China and online
- ISMCTS not a significant improvement on determinization
  - Hidden information is not as important as one might think…
  - In situations where hidden info *is* important, ISMCTS performs well



ISMCTS for Spades

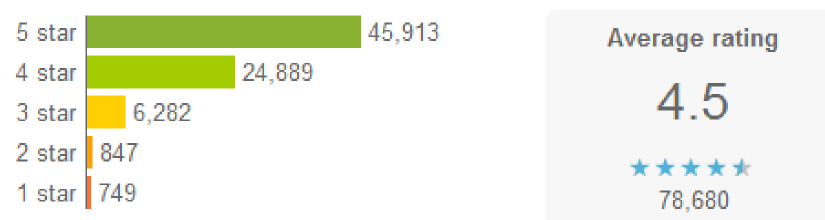
Daniel Whitehouse, Peter I. Cowling, Edward J. Powley and Jeff Rollason.

Integrating Monte Carlo Tree Search with knowledge-based methods to create engaging play in a commercial mobile game.

Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2013.

Spades by AI Factory

- Android version of the popular card game
- (Human + AI) vs (AI + AI)
- **7.5 million +** downloads



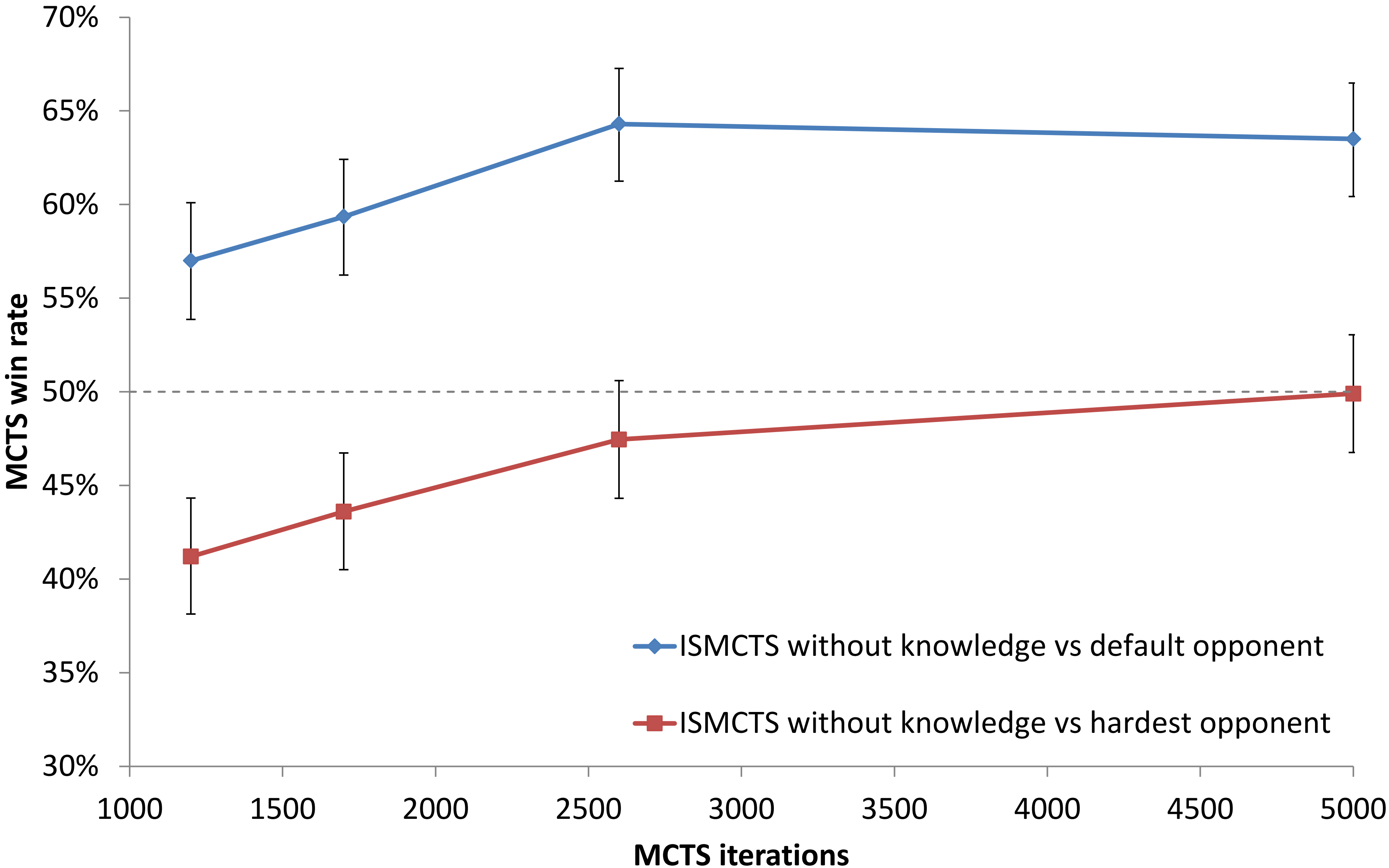
Rule-based AI for Spades

- Developed over 10 years
- Uses **determinization** and **Monte Carlo evaluation** (no trees)
- Heavily reliant on **expert knowledge based rules**
- Generally well reviewed…
- … but some deficiencies in play at highest difficulty levels

ISMCTS for Spades

- Objectively strong…

Performance of ISMCTS for Spades



ISMCTS for Spades

- Objectively strong…
- … but makes choices which **appear bad** to a human player
  - Here **plausibility** is more important than **win rate**
- Use the **rule-based AI’s knowledge** to **bias** ISMCTS towards plausible moves
  - No measurable effect on win rate…
  - … but fixes many instances of perceived bad play

I've bid nil. My partner over trumps opponent's three with Queen and then leads a ten. He should have used the ten and led the Queen.

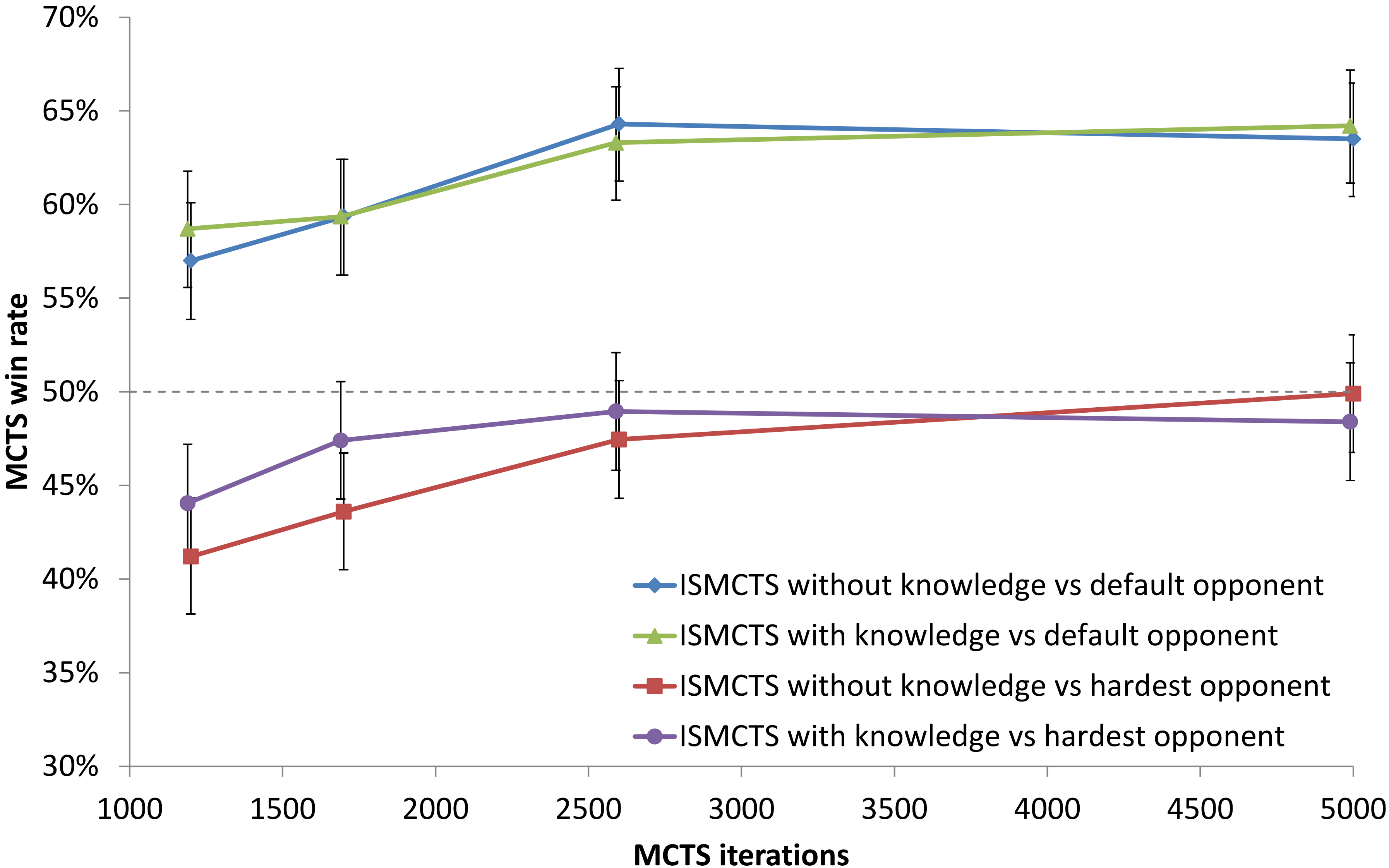
```
Version = 0F9
Seed = 1755444836
Play Levels (S,W,N,E): N/A, 28, 28, 28
Play Styles (S,W,N,E): N/A, 128, 0, 0
AI Type (S,W,N,E): N/A, 129, 129, 129

...

E Play: 7C
S Play: 4C
W Play: 3S
N Play: QS

N Play: 10S
E Play: 6S
```

Performance of ISMCTS for Spades



MCTS for a real-time game

Edward J. Powley, Daniel Whitehouse and Peter I. Cowling. Monte Carlo Tree Search with macro-actions and heuristic route planning for the Physical Travelling Salesman Problem. Proceedings of IEEE Conference on Computational Intelligence in Games (CIG), 234–241, 2012.

Diego Perez, Edward J. Powley, Daniel Whitehouse, Philipp Rohlfshagen, Spyridon Samothrakis, Peter I. Cowling and Simon M. Lucas. Solving the Physical Travelling Salesman Problem: tree search and macro-actions. IEEE Transactions on Computational Intelligence and AI in Games, 6(1):31–45, 2014.

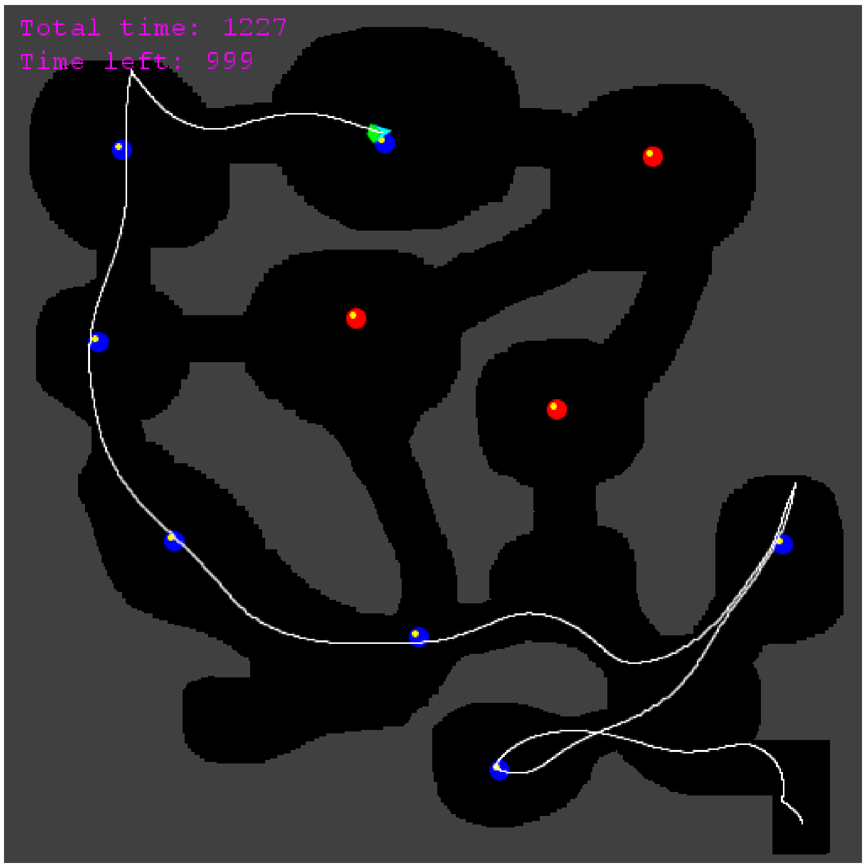
Travelling Salesman Problem (TSP)

- Classic problem in Computer Science
- We have a **graph**
- From starting node S, find the shortest possible path that visits every node exactly once and returns to S
- Many real-world applications
  - Transport and logistics
  - Manufacturing
  - Playing Pac-Man
  - Pub crawls (http://www.math.uwaterloo.ca/tsp/pubs/)

Solving TSP

- TSP is **NP-complete**
- Assuming $P \neq NP$, there is no polynomial time algorithm for solving TSP perfectly
  - I.e. all algorithms scale horribly as the graph gets large
- However there are many good **heuristics** and **approximate algorithms**
- In this work we used **multiple fragment** and **3-opt**

Physical Travelling Salesman Problem (PTSP)



- Steer a spaceship to collect all **waypoints**
  - *Asteroids*-like controls
  - Newtonian physics
- Map is unknown in advance
- Controller has **a few seconds** of initialisation time, and then must make an input to the ship every **40ms**

Demo

Our PTSP controller in action

Challenges for tree search in real-time domains

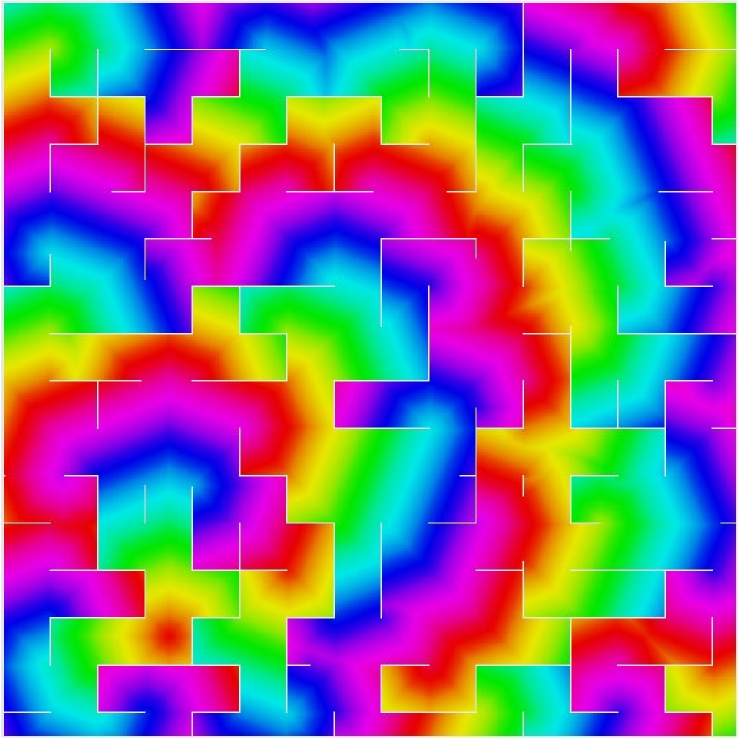
- Many more **decisions per game** than most turn-based games
- Hence state space is enormous even if branching factor is small
  - PTSP has of the order $\mathbf{10^{1556}}$ states
  - 19×19 Go has of the order $\mathbf{10^{171}}$ states
- **Time budget** is restricted (milliseconds per decision)

Key features of our PTSP controller

- **Hierarchical structure**
  - Higher-level **route planner** chooses the waypoint order
    - TSP solver with heuristics for avoiding sharp turns
  - Lower-level **steering controller** executes the route
    - MCTS with macro-actions and heuristic evaluation
- **Depth limiting and heuristic evaluation**
  - Guides the steering controller along the route
- **Macro-actions**
  - Vastly reduce the state-action space for steering

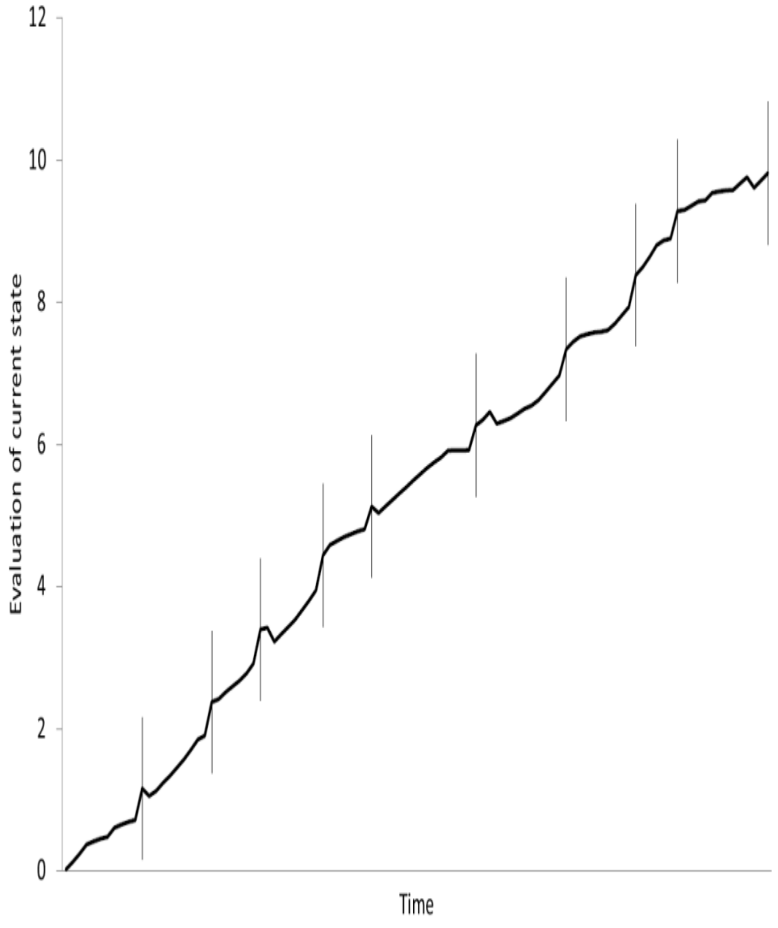
Heuristic evaluation

- Based on **“A* path” distance** to next waypoint
- Precomputed using a flood-fill algorithm



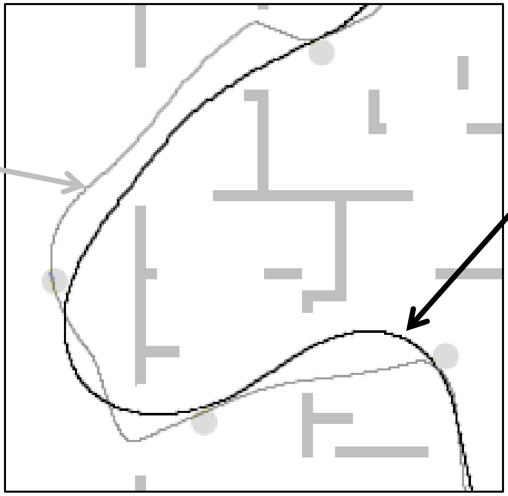
Heuristic evaluation

- Guides the ship towards the **current waypoint**
- **Sharp increase in score** for collecting the waypoint…
- … At which point the distance to the **next waypoint** takes over



Macro-actions

- Choosing a different action every time step is an **unnecessary level of granularity**
- Instead, choose an action to be executed **for the next T time steps**



Benefits of macro-actions

- **Vastly** reduce size of state space
  - Original state space size: **$\approx 10^{1556}$**
  - Macro-action state space size (T=15): **$\approx 10^{103}$**
  - A reduction of **≈1453 orders of magnitude**
  - Size of macro-action space is comparable to 9×9 Go
- See **T times further into the future** for the same tree depth
- Have **T times longer** to make each decision
  - MCTS is not only anytime but also interruptible

Conclusion

- MCTS is a powerful **general-purpose** AI technique
  - Asymmetric, Anytime, Aheuristic
- MCTS has proven successful in several **challenging classes of games**
  - Games of imperfect information
  - Commercial mobile games
  - Real-time games
- It shows promise in **many other games and non-game applications**

Further reading

IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 4, NO. 1, MARCH 2012 1

A Survey of Monte Carlo Tree Search Methods

Cameron B. Browne, *Member, IEEE*, Edward Powley, *Member, IEEE*, Daniel Whitehouse, *Graduate Student Member, IEEE*, Simon M. Lucas, *Senior Member, IEEE*, Peter I. Cowling, *Member, IEEE*, Philipp Rohlfshagen, *Member, IEEE*, Stephen Tavener, Diego Perez, Spyridon Samothrakis, *Graduate Student Member, IEEE*, and Simon Colton

***Abstract*—Monte Carlo tree search (MCTS) is a recently proposed search method that combines the precision of tree search with the generality of random sampling. It has received considerable interest due to its spectacular success in the difficult problem of computer *Go*, but has also proved beneficial in a range of other domains. This paper is a survey of the literature to date, intended to provide a snapshot of the state of the art after the first five years of MCTS research. We outline the core algorithm's derivation, impart some structure on the many variations and enhancements that have been proposed, and summarize the results from the key game and nongame domains to which MCTS methods have been applied. A number of open research questions indicate that the field is ripe for future work.**

***Index Terms*—Artificial intelligence (AI), bandit-based methods, computer *Go*, game search, Monte Carlo tree search (MCTS), upper confidence bounds (UCB), upper confidence bounds for trees (UCT).**

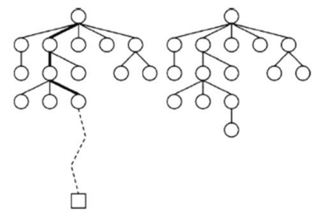


Fig. 1. The basic MCTS process [17].

I. Introduction

MONTE CARLO TREE SEARCH (MCTS) is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results. It has already had a profound impact on artificial intelligence (AI) approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems.

In the five years since MCTS was first described, it has become the focus of much AI research. Spurred on by some prolific achievements in the challenging task of computer *Go*, researchers are now in the process of attaining a better understanding of when and why MCTS succeeds and fails, and of extending and refining the basic algorithm. These developments are greatly increasing the range of games and other decision applications for which MCTS is a tool of choice, and pushing its performance to ever higher levels. MCTS has many attractions: it is a statistical anytime algorithm for which more computing power generally leads to better performance. It can be used with little or no domain knowledge, and has succeeded on difficult problems where other techniques have failed. Here we survey the range of published work on MCTS, to provide the reader with the tools to solve new problems using MCTS and to investigate this powerful approach to searching trees and directed graphs.

A. Overview

The basic MCTS process is conceptually very simple, as shown in Fig. 1 (from [17]). A tree[1] is built in an incremental and asymmetric manner. For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising). A simulation[2] is then run from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node, and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy, which in the simplest case is to make uniform random moves. A great benefit of MCTS is that the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which greatly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.

Manuscript received October 22, 2011; revised January 12, 2012; accepted January 30, 2012. Date of publication February 03, 2012; date of current version March 13, 2012. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grants EP/I001964/1, EP/H048588/1, and EP/H049061/1, as part of the collaborative research project *UCT for Games and Beyond* being undertaken by Imperial College London, the University of Essex, and the University of Bradford.

C. Browne, S. Tavener, and S. Colton are with the Department of Computing, Imperial College London, London SW7 2RH, U.K. (e-mail: camb@doc.ic.ac.uk; sct110@doc.ic.ac.uk; sgc@doc.ic.ac.uk).

S. M. Lucas, P. Rohlfshagen, D. Perez, and S. Samothrakis are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, Essex CO4 3SQ, U.K. (e-mail: sml@essex.ac.uk; prohlf@essex.ac.uk; dperez@essex.ac.uk; ssamot@essex.ac.uk).

E. Powley, D. Whitehouse, and P. I. Cowling are with the School of Computing, Informatics and Media, University of Bradford, Bradford, West Yorkshire BD7 1DP, U.K. (e-mail: e.powley@bradford.ac.uk; d.whitehouse1@bradford.ac.uk; p.i.cowling@bradford.ac.uk).

Digital Object Identifier 10.1109/TCIAIG.2012.2186810

[1]Typically a game tree.

[2]A random or statistically biased sequence of actions applied to the given state until a terminal condition is reached.

C Browne et al.

***A Survey of Monte Carlo Tree Search Methods*.**

IEEE Transactions on Computational Intelligence and AI in Games, 4(1):1-43, 2012.