Monte Carlo Tree Search

Simon M. Lucas

Outline

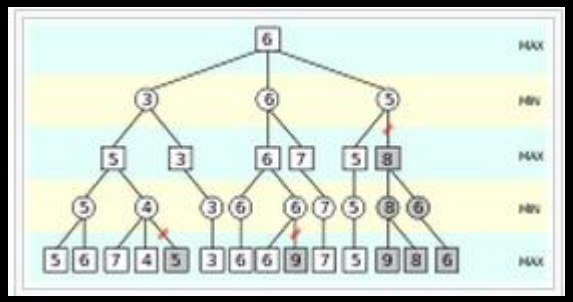
- MCTS: The Excitement!
- A tutorial: how it works
- Important heuristics: RAVE / AMAF
- Applications to video games and real-time control

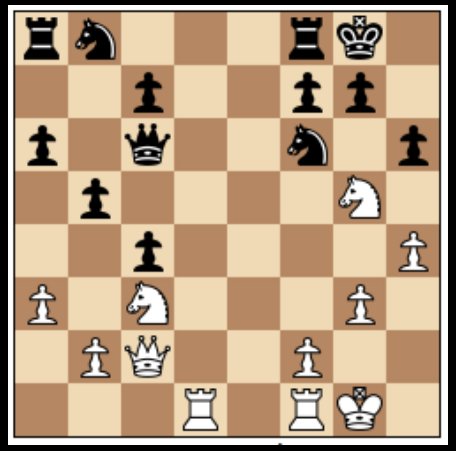
The Excitement…

- Game playing before MCTS
- MCTS and GO
- MCTS and General Game Playing

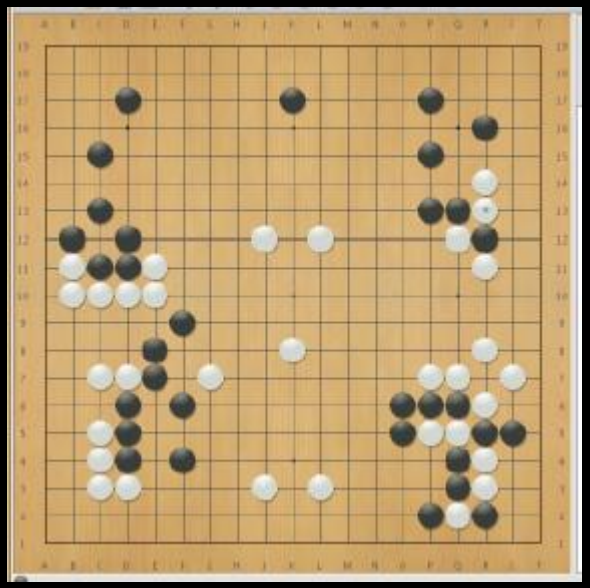
Conventional Game Tree Search

- Minimax with alpha-beta pruning, transposition tables
- Works well when:
  - A good heuristic value function is known
  - The branching factor is modest
- E.g. Chess, Deep Blue, Rybka etc.





Go



- Much tougher for computers
- High branching factor
- No good heuristic value function

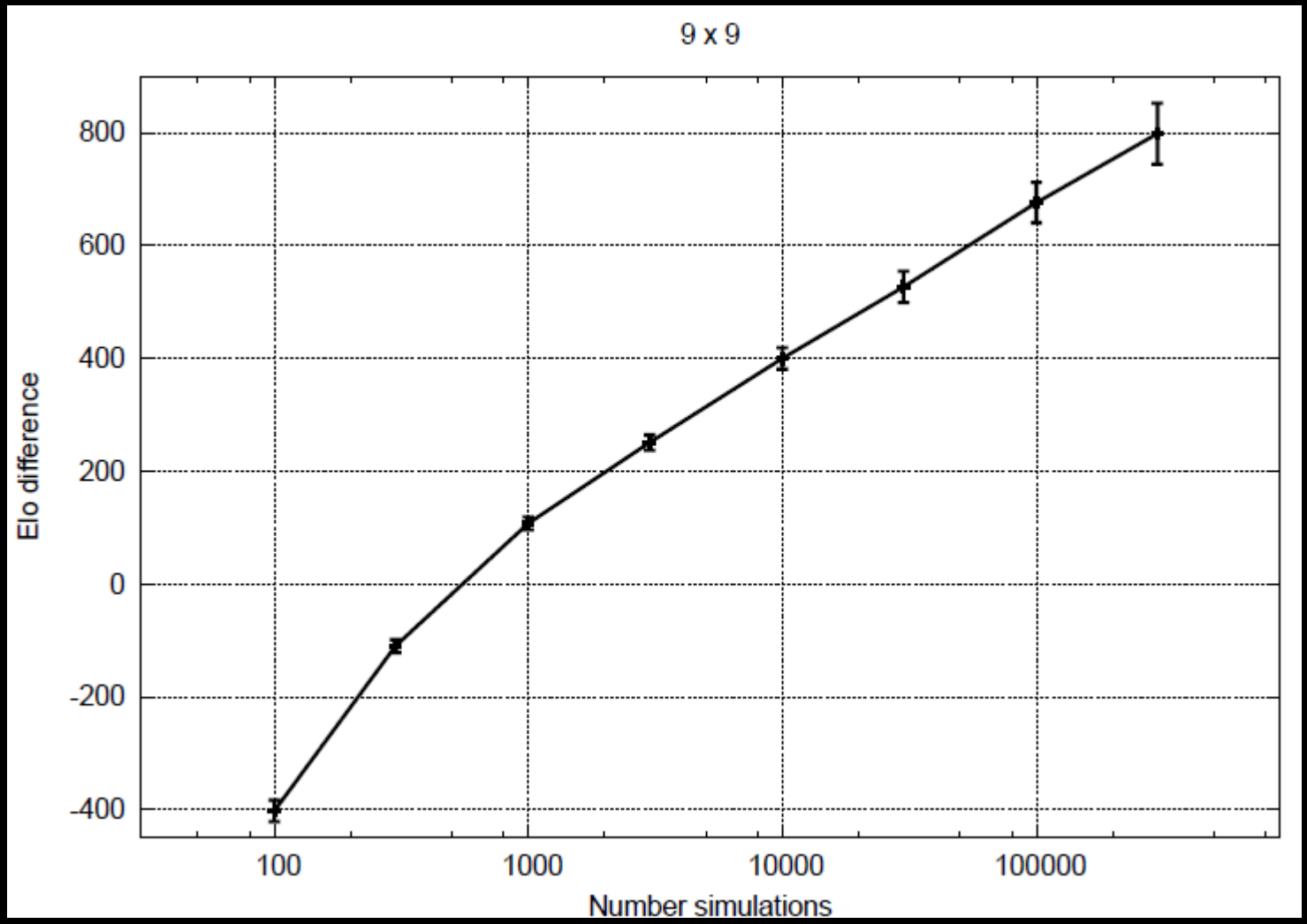
"Although progress has been steady, it will take many decades of research and development before world-championship–calibre go programs exist". Jonathan Schaeffer, 2001

Monte Carlo Tree Search (MCTS)

- Revolutionised the world of computer go
- Best GGP players (2008, 2009) use MCTS
- More CPU cycles leads to smarter play
  - Typically lin / log: each doubling of CPU time adds a constant to playing strength
- Uses statistics of deep look-ahead from randomised roll-outs
- Anytime algorithm

Fuego versus GnuGo

(from Fuego paper, IEEE T-CIAIG vol2 # 4)



General Game Playing (GGP) and Artificial General Intelligence (AGI)

- Original goal of AI was to develop general purpose machine intelligence
- Being good at a specific game is not a good test of this – it's narrow AI
- But being able to play *any* game seems like a good test of AGI
- Hence general game playing (GGP)

GGP: How it works

- Games specified in predicate logic
- Two phases:
  - GGP agents are given time to teach themselves how to play the game
  - Then play commences on a time-limited basis
- Wonderful stuff!
- Great challenge for machine learning,
  - But interesting to see which methods work best...
- Current best players all use MCTS

MCTS Tutorial

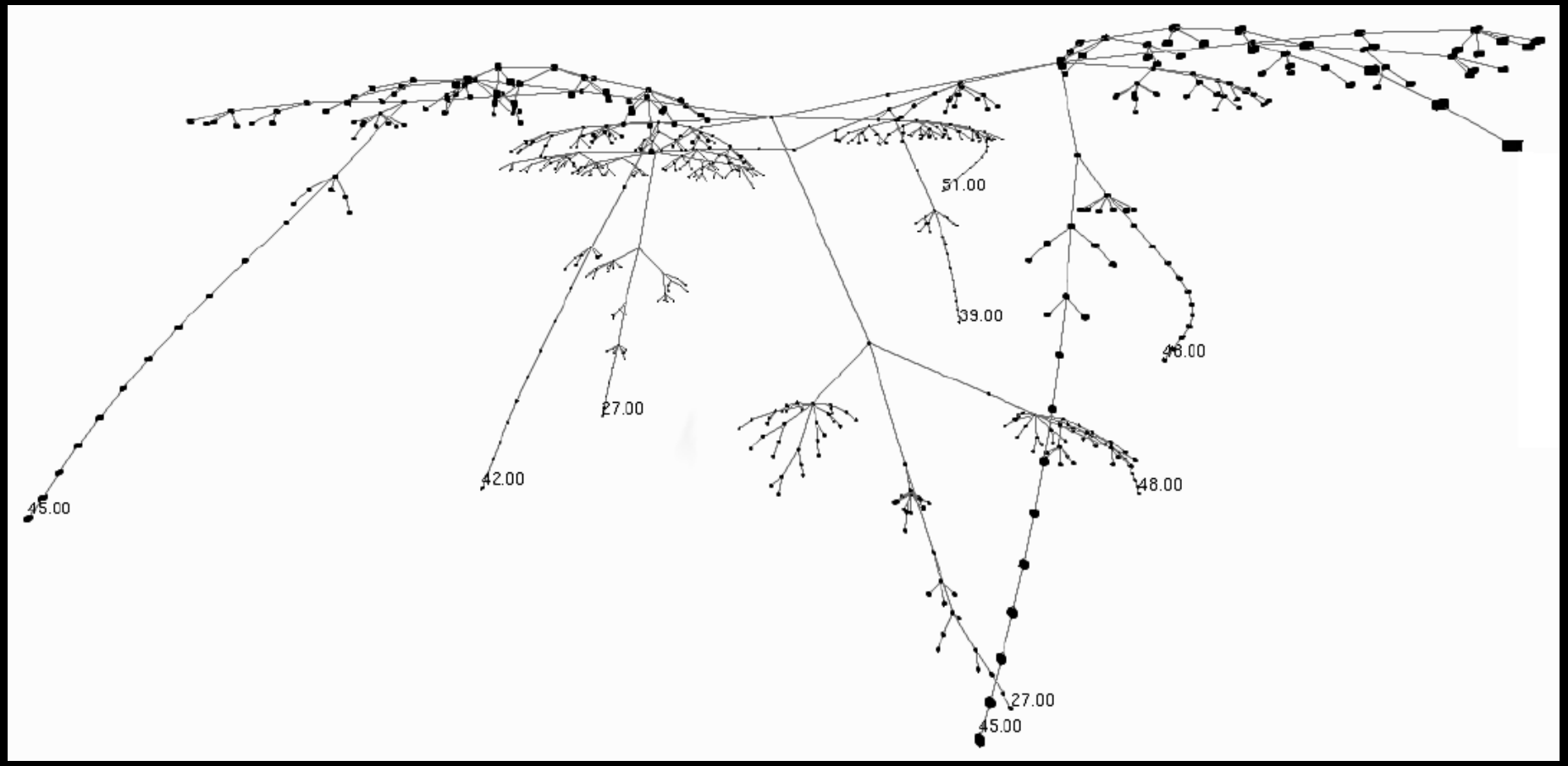
- How it works: MCTS general concepts
- Algorithm
- UCT formula
- Alternatives to UCT
- RAVE / AMAF Heuristics

MCTS

- Builds and searches an asymmetric game tree to make each move
- Phases are:
  - Tree search: select node to expand using tree policy
  - Perform random roll-out to end of game when true value is known
  - Back the value up the tree

Sample MCTS Tree

(fig from CadiaPlayer,
Bjornsson and Finsson, IEEE T-CIAIG)



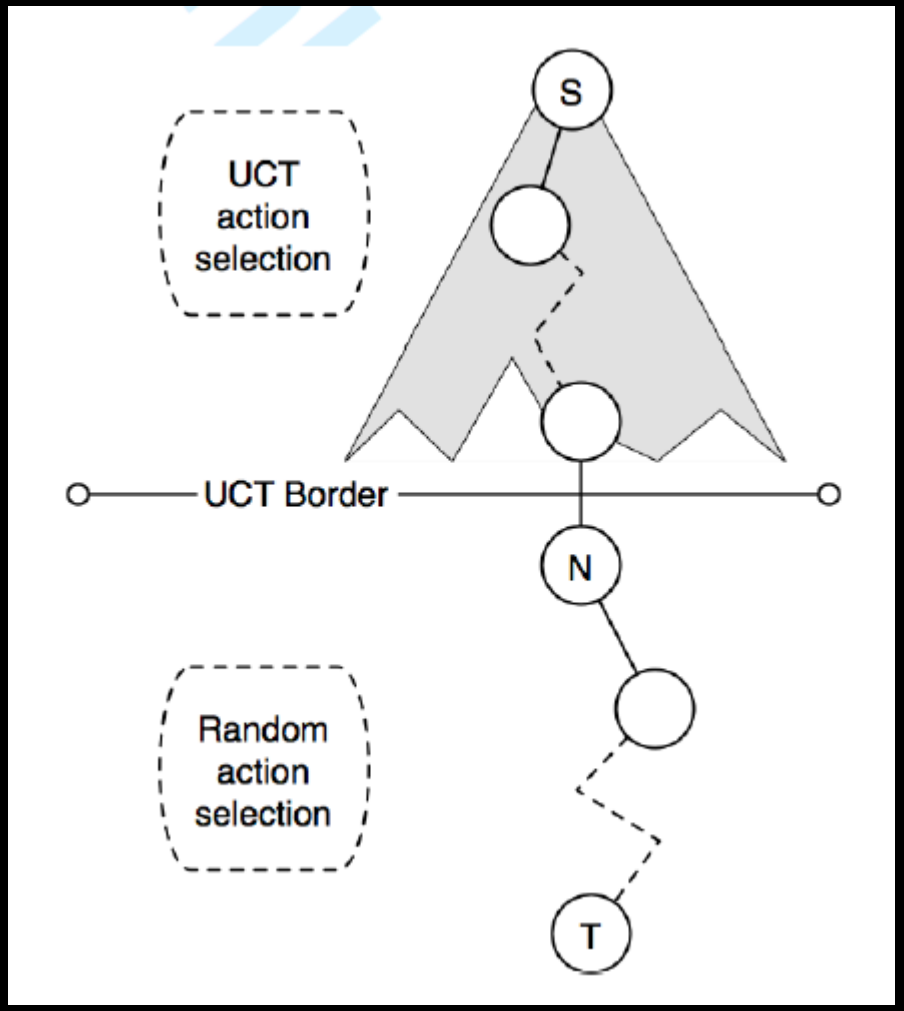
MCTS Algorithm for Action Selection

```
repeat N times {  // N might be between 100 and 1,000,000
    // set up data structure to record line of play
    visited = new List<Node>()
    // select node to expand
    node = root
    visited.add(node)
    while (node is not a leaf) {
      node = select(node, node.children)  // e.g. UCT selection
      visited.add(node)
    }
    // add a new child to the tree
    newChild = expand(node)
    visited.add(newChild)
    value = rollOut(newChild)
    for (node : visited)
        // update the statistics of tree nodes traversed
        node.updateStats(value);
    }
}
return action that leads from root node to most valued child
```

MCTS Operation

(fig from CadiaPlayer,
Bjornsson and Finsson, IEEE T-CIAIG)

- Each iteration starts at the root
- Follows tree policy to reach a leaf node
- Then perform a random roll-out from there
- Node ‘N’ is then added to tree
- Value of ‘T’ back-propagated up tree



Upper Confidence Bounds on Trees (UCT) Node Selection Policy

- From Kocsis and Szepesvari (2006)
- Converges to optimal policy given infinite number of roll-outs
- Often not used in practice!

$$Select\ i_{next} = \arg \max_{i \in \text{children nodes}} \hat{\mu}_i + \sqrt{\frac{\log N}{n_i}}$$

Tree Construction Example

- See Olivier Teytaud's slides from AIGamesNetwork.org summer 2010 MCTS workshop

AMAF / RAVE Heuristic

- Strictly speaking: each iteration should only update the value of a single child of the root node
- The child of the root node is the first move to be played
- AMAF (All Moves as First Move) is a type of RAVE heuristic (Rapid Action Value Estimate) – the terms are often synonymous

How AMAF works

- Player A is player to move
- During an iteration (tree search + rollout)
  - update the values in the AMAF table of all moves made by player A
- Add an AMAF term to the node selection policy
  - Can also apply this to moves of opponent player?

Should AMAF work?

- Yes: a move might be good irrespective of when it is player (e.g. playing in the corner in Othello is ALWAYS a good move)

- No: the value of a move can depend very much on when it is player
  - E.g. playing next to a corner in Othelo is usually bad, but might sometimes be very good
- Fact: works very well in some games (Go, Hex)
- Challenge: how to adapt similar principles for other games (Pac-Man)?

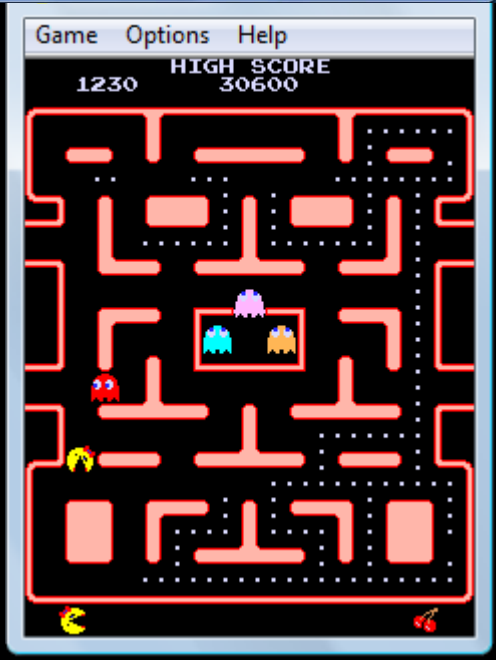
Improving MCTS

- Default roll-out policy is to make uniform random moves
- Can potentially improve on this by biasing move selections:
  - Toward moves that players are more likely to make
- Can either program the heuristic – a knowledge-based approach
- Or learn it (Temporal Difference Learning)
  - Some promising work already done on this

MCTS for Video Games and Real-Time Control

- Requirements:
  - Need a fast and accurate forward model
  - i.e. taking action a in state s leads to state s' (or a known probability distribution over a set of states)
- If no such model exists, then could maybe learn it?
- How accurate does the model need to be?
- For games, such a model always exists
  - But may need to simplify it

Sample Games



MCTS Real-Time Approaches

- State space abstraction:
  - Quantise state space – mix of MCTS and Dynamic Programming – search graph rather than tree
- Temporal Abstraction
  - Don’t need to make different actions 60 times per second!
  - Instead, current action is usually the same (or predictable from) the previous one
- Action abstraction
  - Consider higher-level action space

Initial Results on Video Games

- Tron (Google AI challenge)
  - MCTS worked ok
- Ms Pac-Man
  - Works brilliantly when given good ghost models
  - Still works better than other techniques we've tried when the ghost models are unknown

MCTS and Learning

- Some work already on this (Silver and Sutton, ICML 2008)
- Important step towards AGI (Artificial General Intelligence)
- MCTS that never learns anything is clearly missing some tricks
- Can be integrated very neatly with TD Learning

Multi-objective MCTS

- Currently the value of a node is expressed as a scalar quantity
- Can MCTS be improved by making this multi-dimensional
- E.g. for a line of play, balance effectiveness with variability / fun

Some Remarks

- MCTS: you have to get your hands dirty!
  - The theory is not there yet (personal opinion)
- To work, roll-outs must be informative
  - i.e. they must return information
- How NOT to use MCTS
  - A planning domain where a long string of random actions is unlikely to reach goal
  - Would need to bias roll-outs in some way to overcome this

Some More Remarks

- MCTS: a crazy idea that works surprisingly well!
- How well does it work?
  - If there is a more applicable alternative (e.g. standard game tree search on a fully enumerated tree), MCTS may be terrible by comparison
- Best for tough problems for which other methods don't work