Parallel Monte Carlo Tree Search

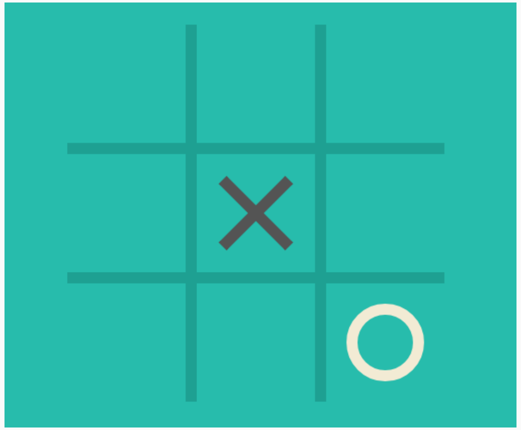
Spring 2018 CS87

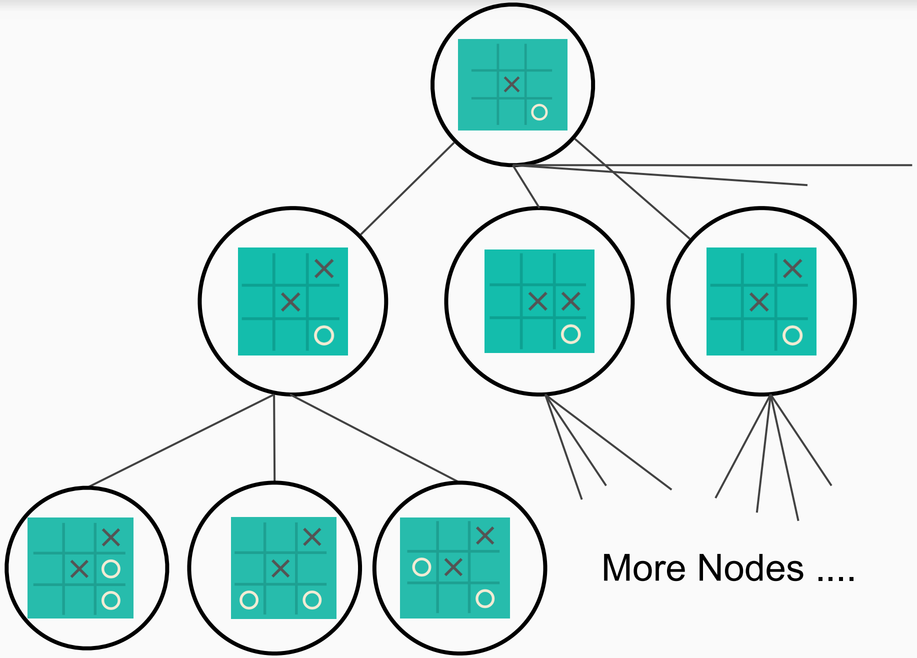
By: Michael Davinroy, Shawn Pan, Jimmy Shah

Outline

- Game Trees
- What is Monte Carlo Tree Search?
- Parallelizing Monte Carlo Tree Search
- Parallel MCTS Results
- Novel Approaches: Top-K and Smart-K
- Top-K and Smart-K Results
- Conclusions
- Future Work

Example of Game State (Tic-tac-toe)

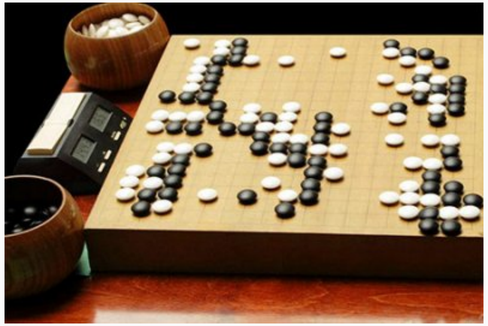




Game state can be extremely large

Estimated number of game states:

- Chess: 10^{46}
- Go: 10^{170}



What is Monte Carlo Tree Search

- Game decision making
- Precision of tree search
- Generality of random simulation

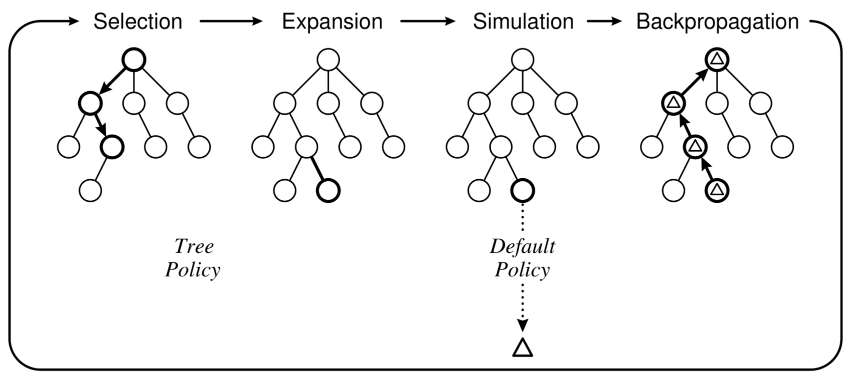
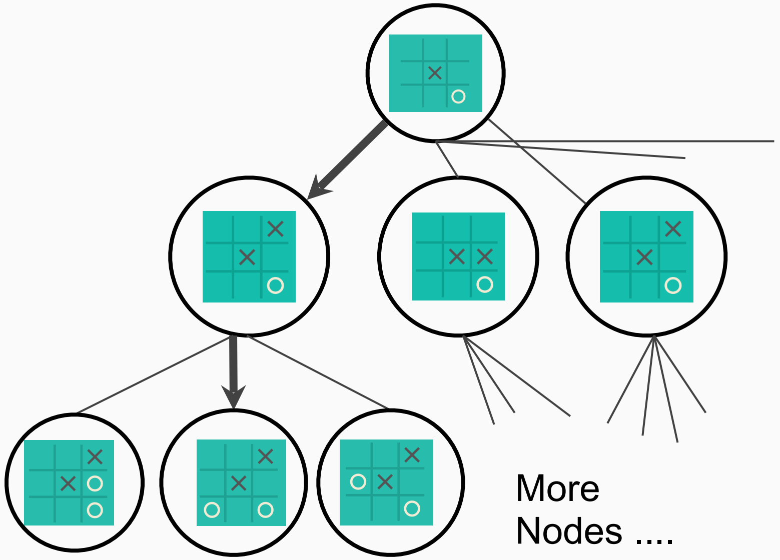


Fig. 2. One iteration of the general MCTS approach.

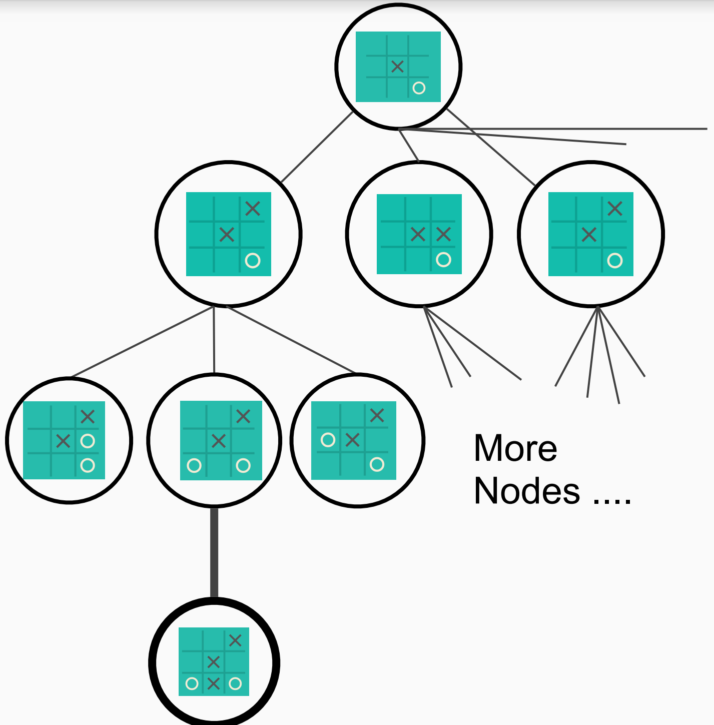
Selection Phase - MCTS

- Exploration- Exploitation trade-off
- Traverses already explored tree
- Halts after finding unexplored child



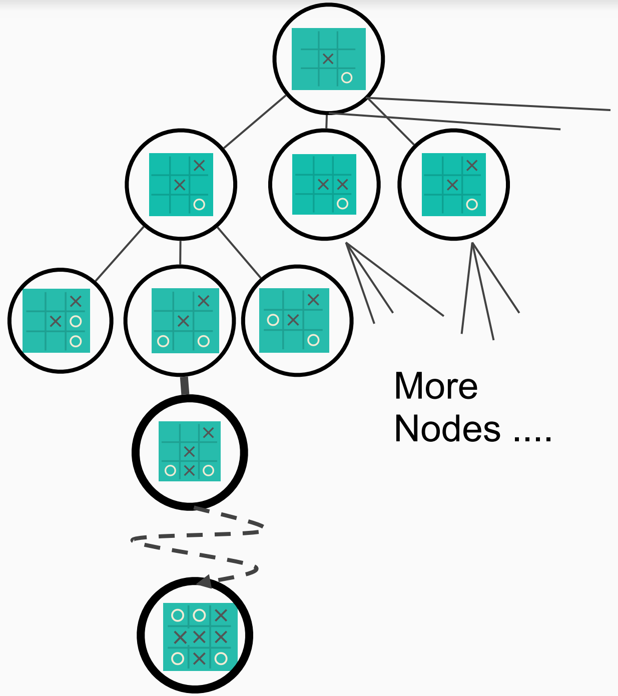
Expansion Phase - MCTS

- Finds node with unexplored children
- Uniformly expands one at random



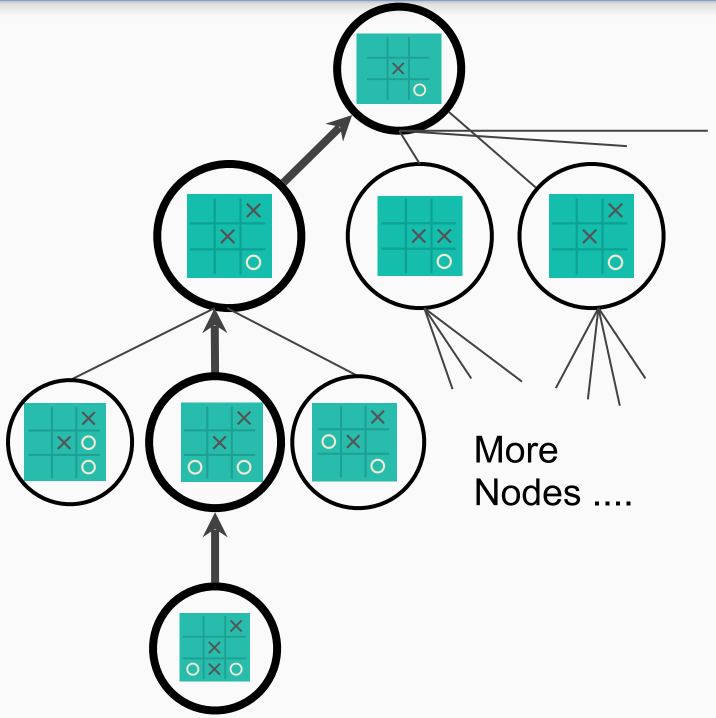
Simulation Phase - MCTS

- Obtain value by simulation
- Random until terminal state
- Win = 1
- Loss = -1
- Draw = 0



Backpropagation Phase - MCTS

- Action's reward added to average value of parent nodes



Monte Carlo Tree Search Recap!

- Brute force extremely ineffective
- Random sims and tree search
- Multiple rollouts
- Exploration-exploitation tradeoff
    - Asymmetric growth

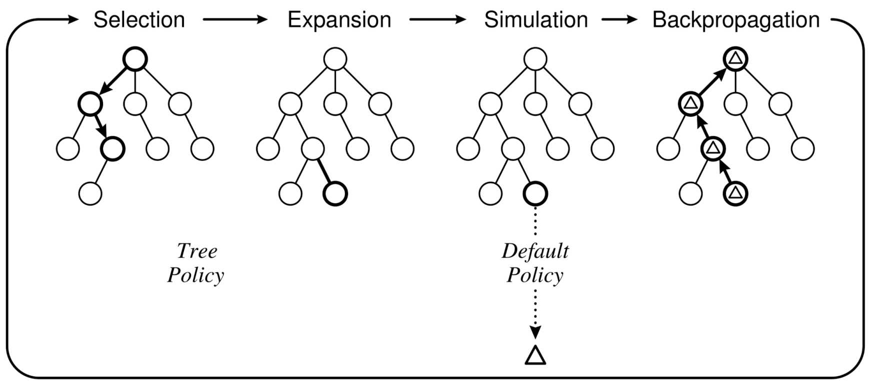
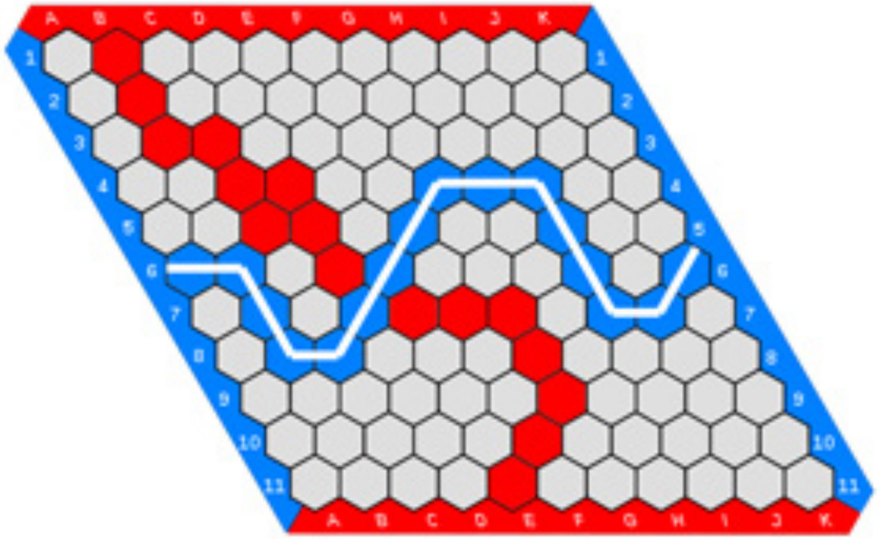


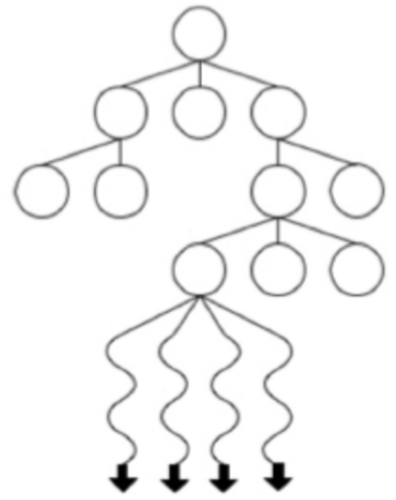
Fig. 2. One iteration of the general MCTS approach.

Hex



Leaf Parallelization

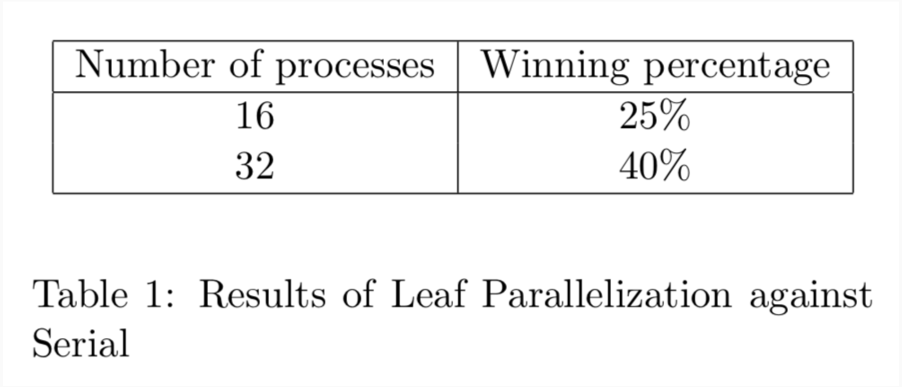
- Simulation Phase in Parallel
- Leaf nodes are reached
- Bottlenecks in this method



Leaf Parallelization

Experiments and Results - Leaf Parallelization

- Serial mostly wins
- Slow, so only 10 games
- Likely equal level players



Number of processes	Winning percentage
16	25%
32	40%

Table 1: Results of Leaf Parallelization against Serial

Root Parallelization

- Multiple MCTS trees in parallel
    - One process per tree
- Information not shared
- Combine results by voting
    - Best move = vote
    - Break ties by visits

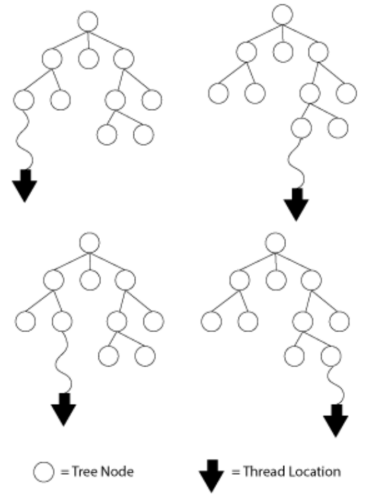
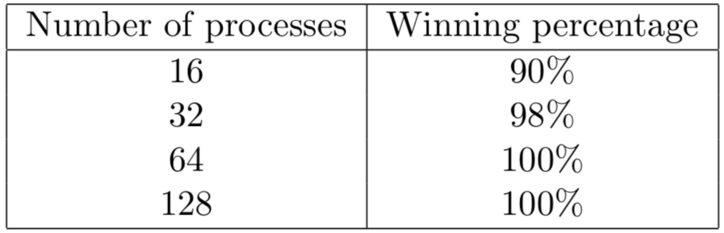


Figure 6: Root Parallelization

Experiments and Results - Root Parallelization

- Effective!
- 50 games
- Much better than Leaf
- Voting
    - Increases confidence
- Much less overhead
    - Much faster



Number of processes	Winning percentage
16	90%
32	98%
64	100%
128	100%

Table 2: Results of Root Parallelization against Serial

Top-k Parallelization

- Two rounds
- Root parallel for x rollouts
- In depth round for y rollouts
  - Top-K valued moves
  - Evenly split among children

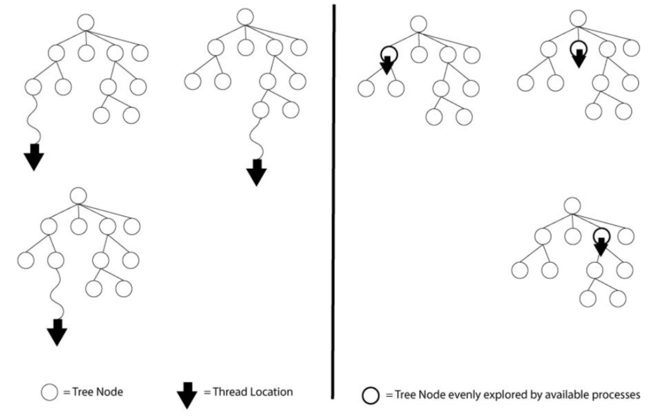
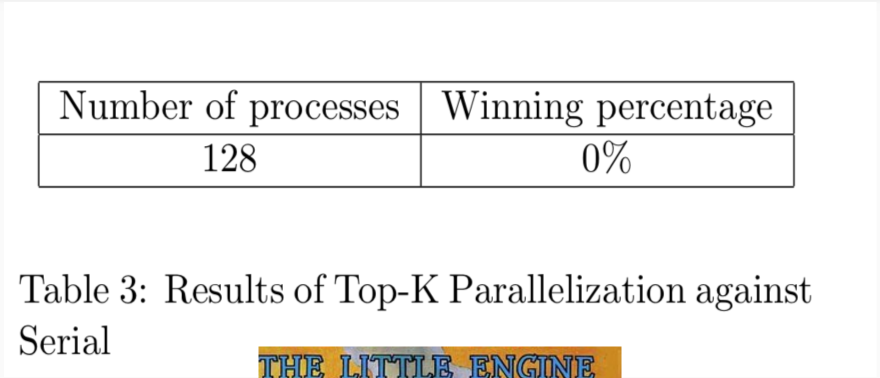


Figure 8: Top-K Parallelization

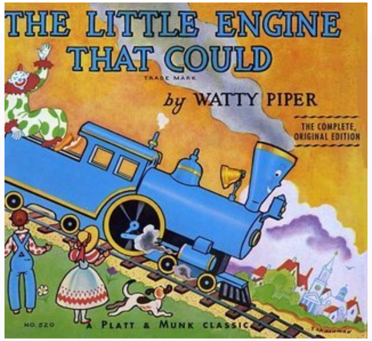
Experiments & Results - Top-k Parallelization

- 128 processes
  - But loses every time
- Refutes our hypothesis
- Maybe bad parameters
- Narrows search too early
- Could be better
  - More processes
  - More initial rollouts



Number of processes	Winning percentage
128	0%

Table 3: Results of Top-K Parallelization against Serial



Smart-k Parallelization

- N rounds for M rollouts
- One round of root parallelization
  - Return values and visits
- N-2 rounds of Smart-K
  - Distribute based on values
  - Return values and visits
- One round of root parallelization
  - Return best child move

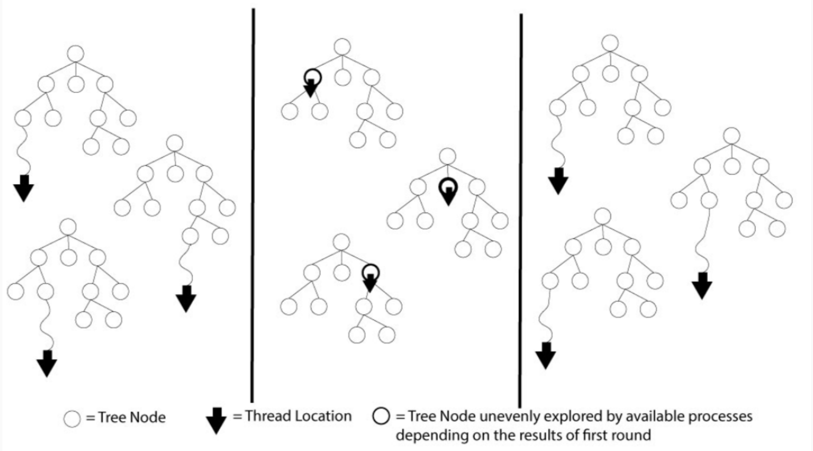
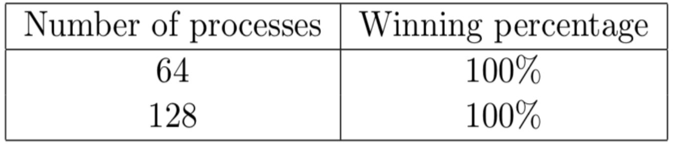

= Tree Node
= Thread Location
= Tree Node unevenly explored by available processes depending on the results of first round

Figure 9: Smart-K Parallelization

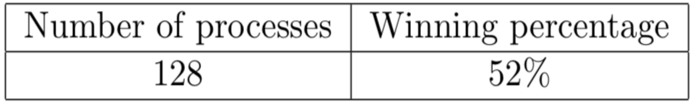
Experiments & Results - Smart-k Parallelization

- About 50/50 against Root
  - Suggests they are equal
- Still Promising
  - More rollouts
  - Larger game trees
  - Shared State



Number of processes	Winning percentage
64	100%
128	100%

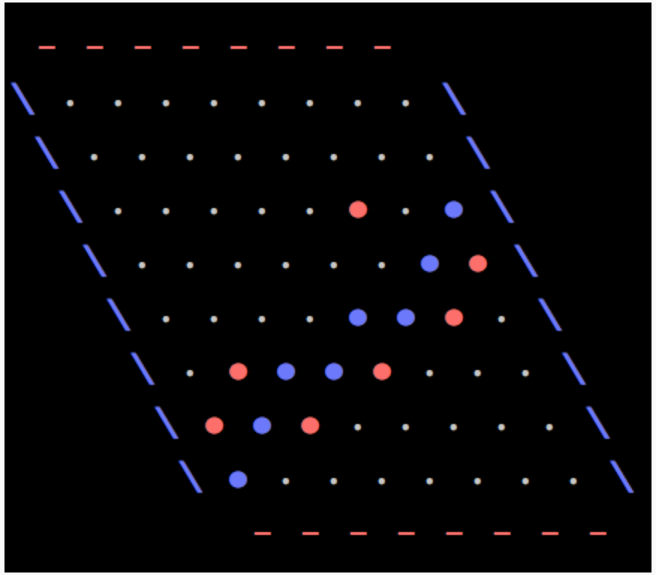
Table 4: Results of Smart-K Parallelization against Serial



Number of processes	Winning percentage
128	52%

Table 5: Results of Smart-K Parallelization against Root Parallelization

Smart-k wins against Root Parallelization!



Conclusions

- Parallelization gives improvement!
    - Voting
- Root (Embarrassingly Task Parallel)
- Top-K and Smart-K
    - Not as good as we expected
    - Still seem promising!

Future Work

- Top-K and Smart-K on larger boards
    - Go or Chess
- Port to C or C++
- Increased number of rollouts
- Share state among children
    - Promising game trees merged
- GPU

Special Thanks

- Professor Tia Newhall!
- Professor Bryce Wiedenbeck!
- Jeff Knerr!