Monte Carlo method applied in Board Game AI

Gaojie He
2009.10.27

Outline

- Introduction of Board Game AI
- Why use Monte Carlo method
- What is Monte Carlo method
- Monte Carlo Tree Search (MCTS)
- Parallelization of MCTS

Introduction of Board Game AI

Board Game AI overview

- Board Game (Chess, Go, Gomoku, kriegspiel, etc.)
  - Perfect information
  - Imperfect information
- Board Game AI can be simply seemed as game tree search
- 2 most important elements of Board Game AI
  - Evaluation function
  - Tree search algorithm (minimax algorithm, etc.)

Game tree

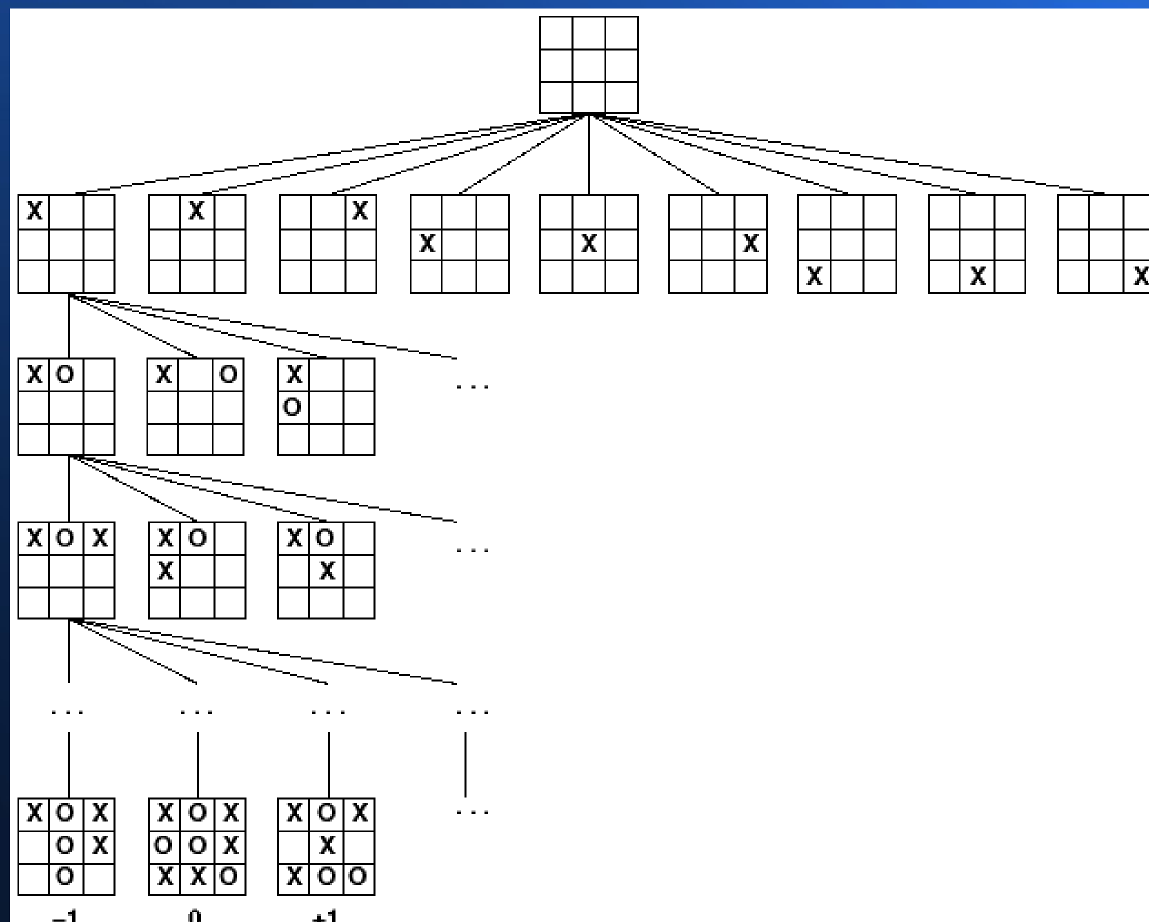

Figure 1. Game tree of Tic-Tac-Toe

Minimax game tree

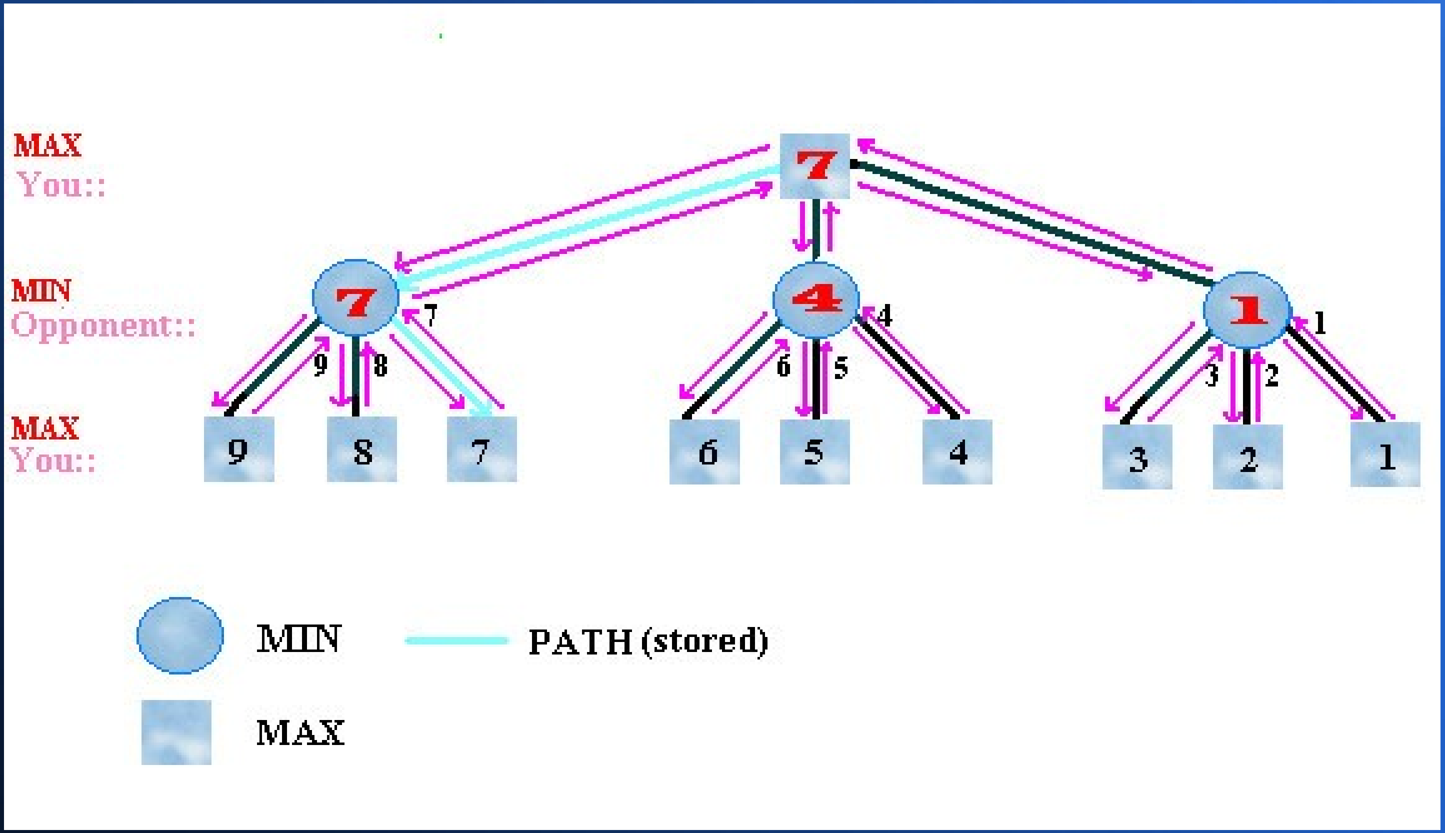

Figure 2. Minimax tree

Evaluation function

- Go (number of remained pieses & eyes, pattern)
- Chess (number of remained pieses, or pattern)
- Gomoku (pattern: connected 2, 3, 4 even 5)
- Etc.

Tree search algorithms

- Classical approaches (more game-dependent heuristic knowledge)
  - Alpha-beta pruning
  - Negascout
  - MTD(f)
  - SSS*
  - Others algorithms and enhancement techniques, e.g. Transposition table, etc.
- Monte Carlo methods
  - Monte carlo tree search

Why use Monte Carlo method

Drawbacks of classical approaches

- Branching factor
  - Average children number of each node
  - Huge search space if branching factor is big
    - Chess, ≈ 35
    - Go, >100 (100^5 with search depth of 5)
- High requirement of evaluation function
  - End game position evaluation
  - Non-end position evaluation (due to the limited search depth)
  - More game-dependent heuristic knowledge

Pros and cons of Monte Carlo method

- Pros
  - Less requirement of game-dependent knowledge, even none (Evaluation function)
  - Relatively easy to parallelized
- Cons
  - Finite length, e.g. Go, Gomoku, Chess is not suitable using Monte Carlo method
  - The random simulation still need to be improved
  - The number of simulated games

What is Monte Carlo method

Monte Carlo method introduction

- A class of computational algorithms that rely on repeated random sampling to compute their results
- Often used when simulating physical and mathematical systems
  - Simulated annealing
  - Pi estimation
  - Traveling salesman problem

Monte Carlo method used in Board Game

- From a single random game, it is quite less to be learnt, but with multitude random games, it becomes meaningful
- Essence: fast self-play game (random game simulation)

Monte Carlo method used in Board Game (cont.)

- Abramson by 1990 (ethello)
  - Evaluation function
- Bruegmann by 1993 (Go)
  - Simulated annealing
  - Just find the best move (lack of accuracy)
- Many other researchers enhanced the Monte Carlo method used in Board Game (esp. Go)
  - MCTS

Monte Carlo Tree Search (MCTS)

MCTS overview

- Best-first search method
- Not classical tree search followed by Monte Carlo evaluation
  - Classical tree search is basically depth-first
- Dynamic growing game tree
  - Classical tree is pre-generated completely
- A lot of simulated play-outs
- The state of art computer Go

MCTS principle

- 4-step procedure
  - Selection
  - Expansion
  - Simulation
  - Back-propagation

1^{st} Selection

- Traverse the tree from root to leaf node (L) using some selection strategy
  - Each node is a board position, stores v_i & n_i
  - Root is current position
  - Leaf node (L) is not end game position
  - Selection strategies

Selection strategy

- UCT algorithm (Upper Confidence bound applied to Trees)
  - Select the move that leads to the best results (exploitation)
  - The least promising moves still have to be explored due to the uncertainty of the evaluation (exploration)
  - Essence: choosing the move that maximizes formula below

$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

2^{nd} Expansion

- Store one child of leaf node (L) in the tree
- Expand one node per simulation (simplest rule)
- The expanded node corresponds to the first position encountered that was not stored yet
- Dynamic growing tree

3rd Simulation

- So called “play-out”
- Self-play with random sequence moves until the end of the game
- Simulation strategy
  - Plain random
  - Pseudo-random
    - Simulated annealing
    - Involve patterns, capture consideration, etc.

4^{th} Back-propagation

- Compute v_i & n_i for each node that is traversed in the one simulation (play-out)
  - N_i += R (R = 1, 0, -1, according to the win/loss)
  - V_i += average score
- Finally the move played by AI player is the child of root with the highest N_i or V_i

MCTS principle scheme

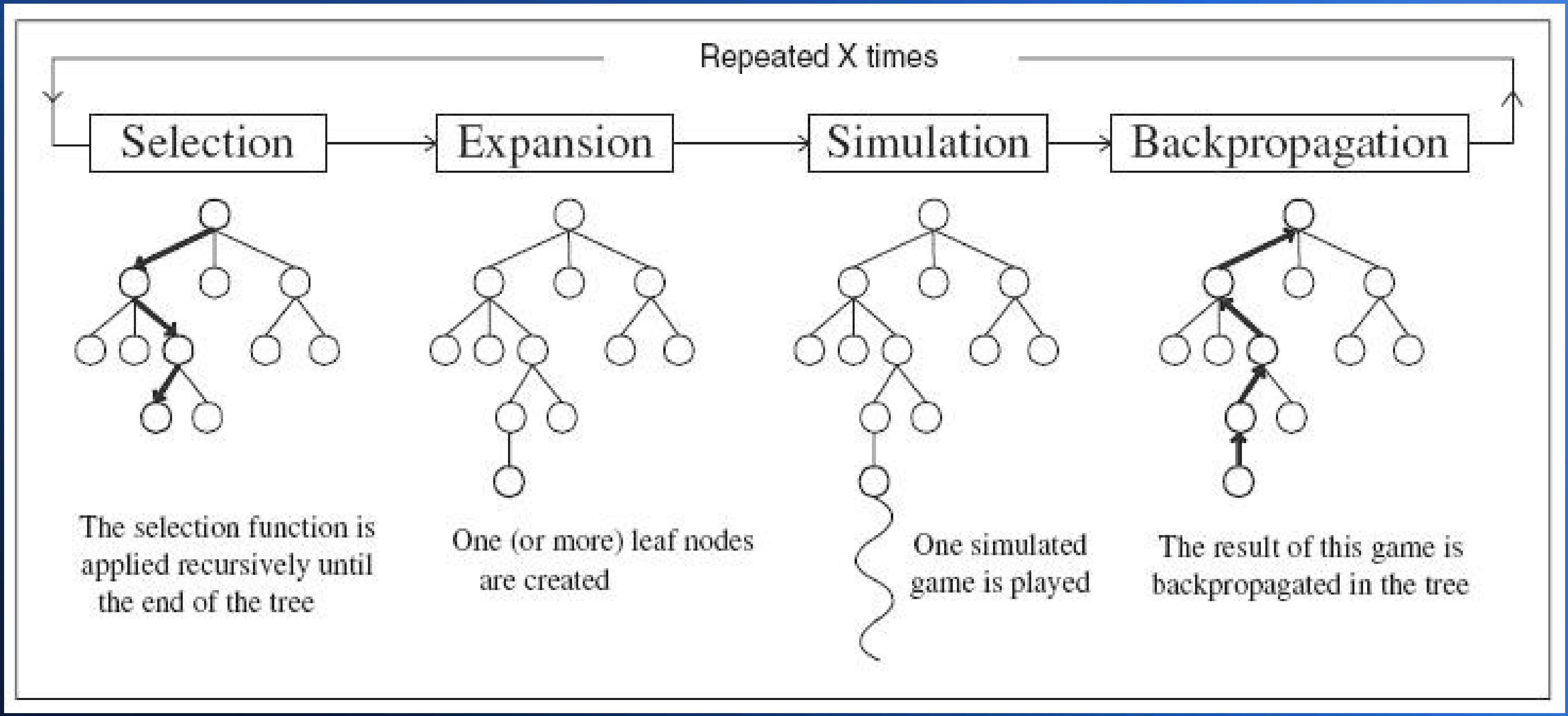

Figure 3. Scheme of Monte Carlo Tree Search

Parallelization of MCTS

Overview

- Independent simulated games imply the parallelization
- 3 different types of parallelization depending on different MCTS steps
  - Leaf parallelism
  - Root parallelism
  - Tree parallelism

Leaf parallelization

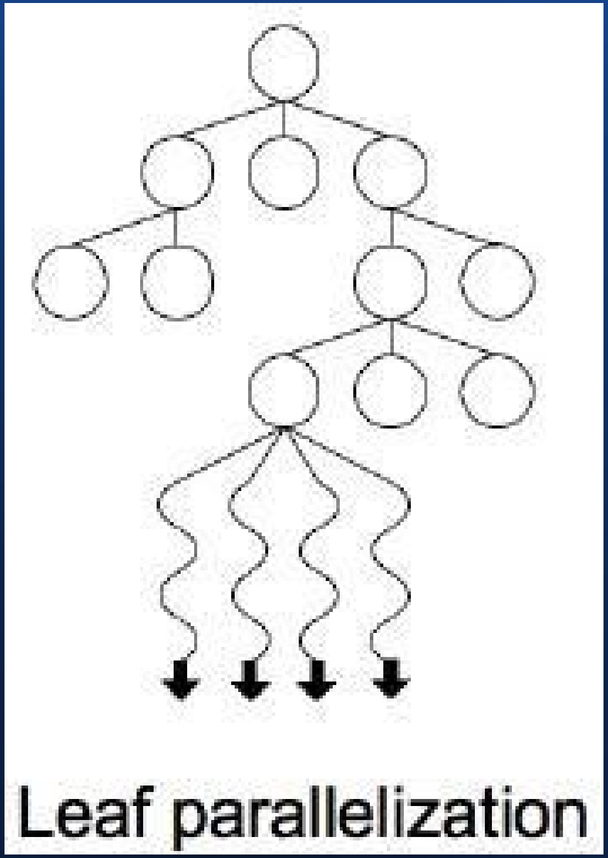

Figure 4. Leaf parallelization

- Simulation step
- Multiple threads simulate game independently, while one threads perform 3 other steps
- 2 problems
  - Waiting time for some threads
  - Information is not shared

Root parallelization

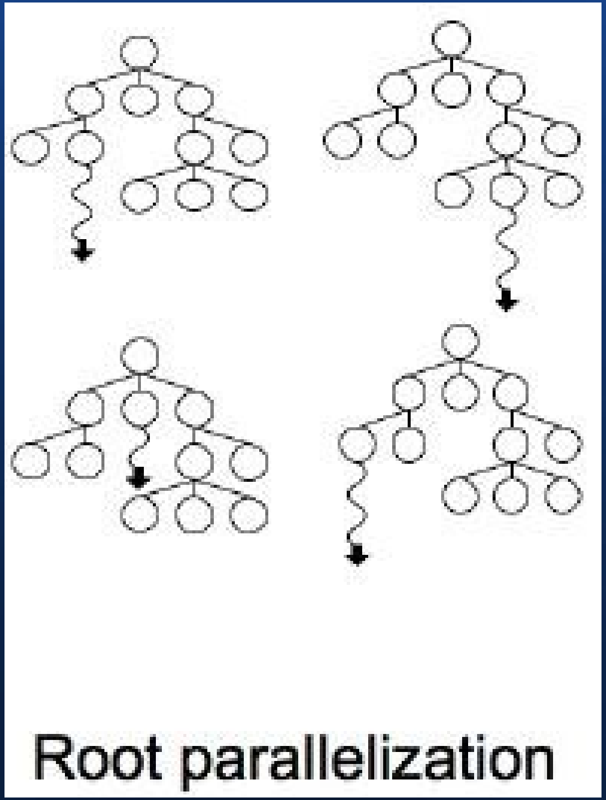

Figure 5. Root parallelization

- The whole MCTS procedure
- Building multiple MCTS tree in parallel
- The final score should be collected from all MCTS trees to decide the best move

Tree parallelization

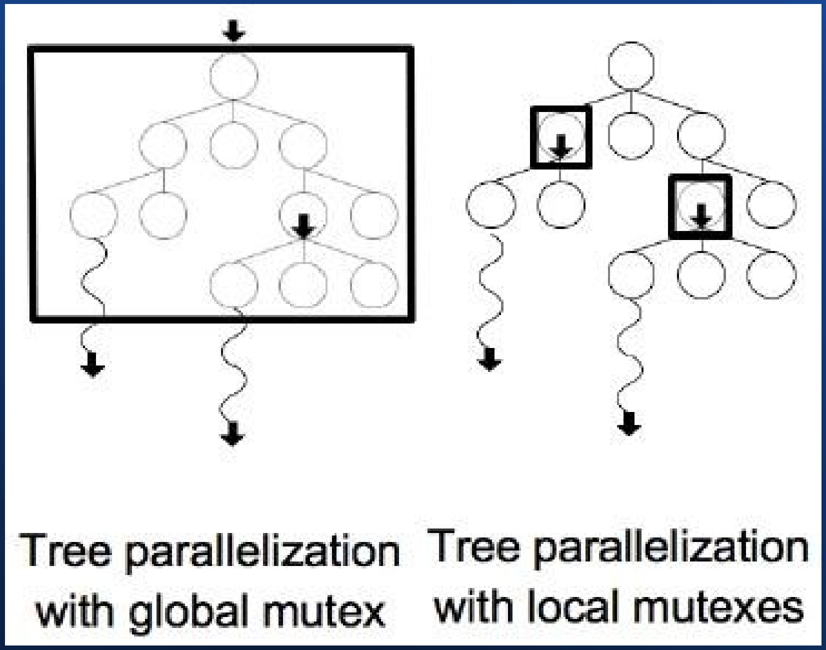

Figure 6. Tree parallelization

- Global mutex
  - Can't access the same MCTS in parallel (step 1, 2, 4)
  - Multiple threads can play simulated games from different leaf node in parallel (step 3)
- Local mutexes
  - Access the same MCTS in parallel

Reference

- Chaslot, G.M.J.-B., Saito, J.-T., Bouzy, B., Uiterwijk, J.W.H.M., van den Herik, H.J.: Monte-Carlo Strategies for Computer Go. In: Schobbens, P.-Y., Vanhoof, W., Schwanen, G. (eds.) Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, pp. 83–90 (2006)
- Chaslot, G.M.J.-B., Winands, M.H.M., Uiterwijk, J.W.H.M., van den Herik, H.J., Bouzy, B.: Progressive strategies for Monte-Carlo Tree Search. New Mathematics and Natural Computation 4(3), 343–357 (2008)
- Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: van den Herik, H.J., Ciancarini, P., Donkers, H.H.L.M(J.) (eds.) CG 2006. LNCS, vol. 4630, pp. 72–83. Springer, Heidelberg (2007)
- Bernd Br□ugmann. Monte Carlo Go. White paper, 1993.