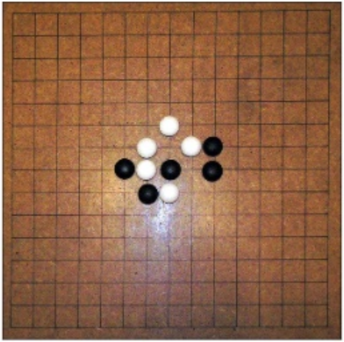
Parallel Game Tree Search: Gomoku

Yuya Kawakami & Haoran Wang

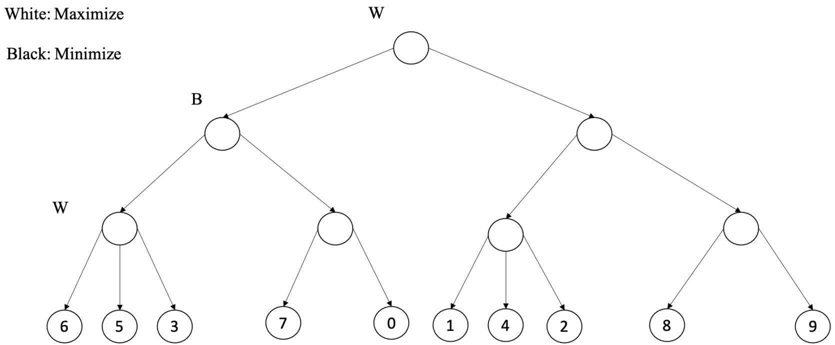
Dec 3, 2020

Introduction
Gomoku

- Also called Five in a Row.
- More complex and difficult than Tic Tac Toe.

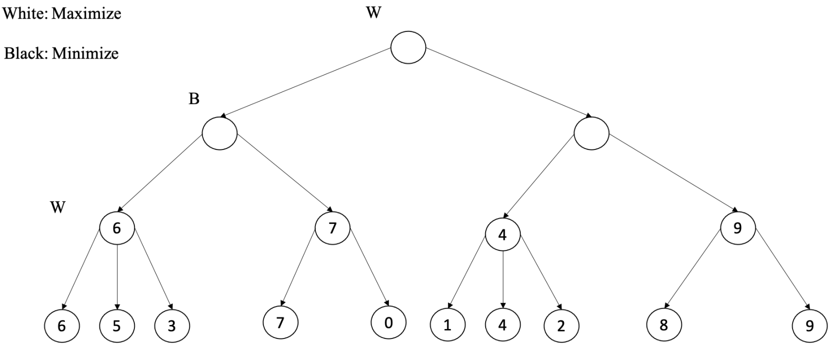


- Maximizer tries to get the highest score.
- Minimizer tries to get the lowest score.



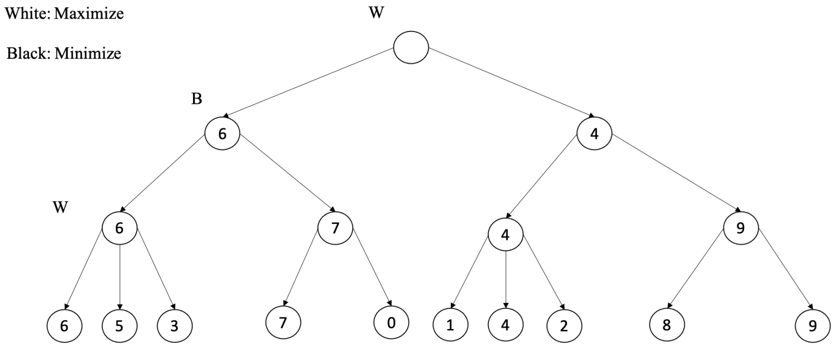
Introduction

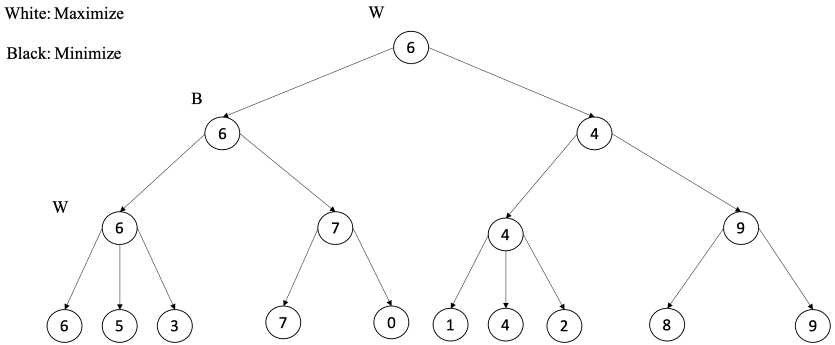
Minimax Game Tree for Zero-Sum Game



Introduction

Minimax Game Tree for Zero-Sum Game

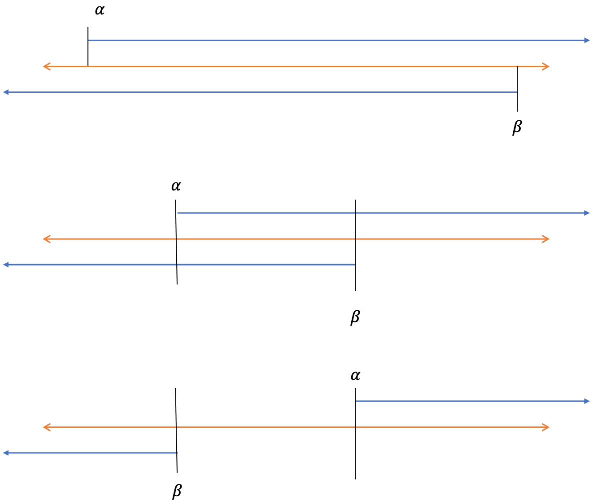




White: Maximize
Black: Minimize
W
6
B
6
4
W
6
7
4
9
6
5
3
7
0
1
4
2
8
9

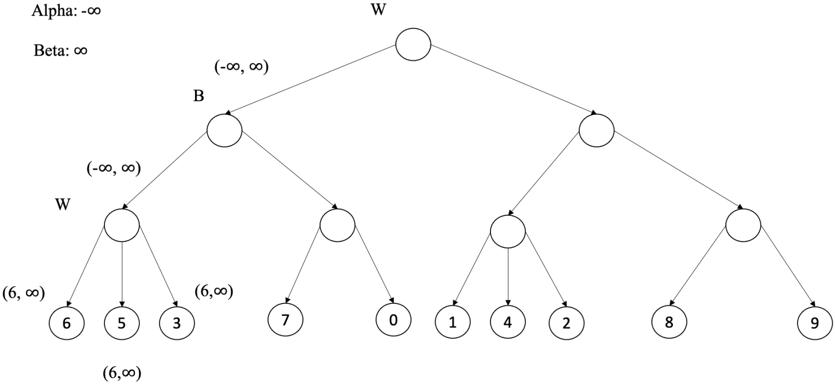
- Alpha: It is the lower bound of possible solutions for maximizer.
- Beta: It is the upper bound of possible solutions for minimizer.

- If beta <= alpha is true, we can prune.



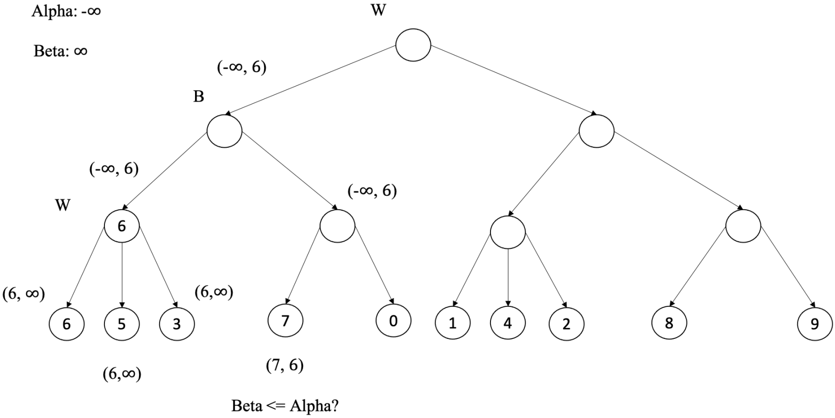
Alpha: -∞

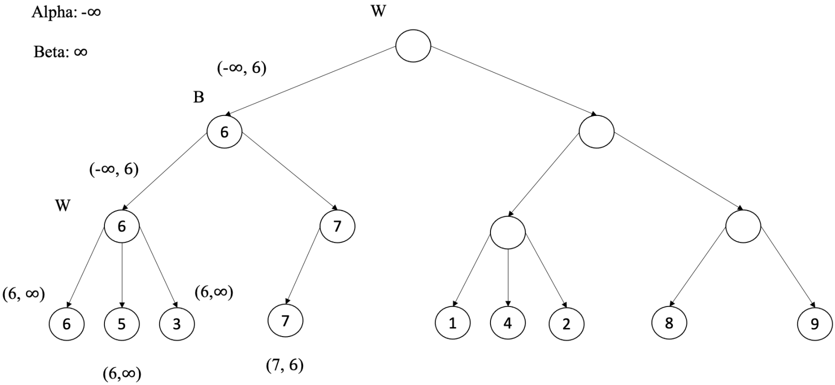
Beta: ∞



Beta <= Alpha?

Introduction
Alpha-Beta Pruning





Alpha: -∞
Beta: ∞
W
(-∞, 6)
B
6
(-∞, 6)
W
6
7
(6, ∞)
6
5
3
(6,∞)
7
(6,∞)
(7, 6)
1
4
2
8
9

- Five in a Row
- Live Four
- Dead Four
- Live Three
- Dead Three
- Live Two
- Dead Two

Introduction

Heuristic: Stone Shapes

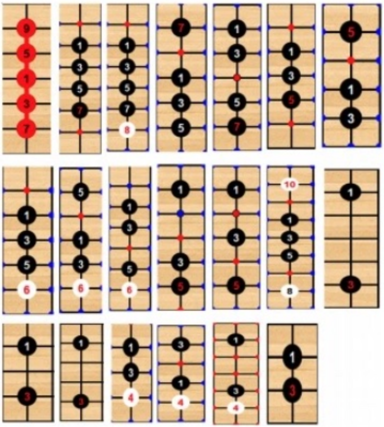
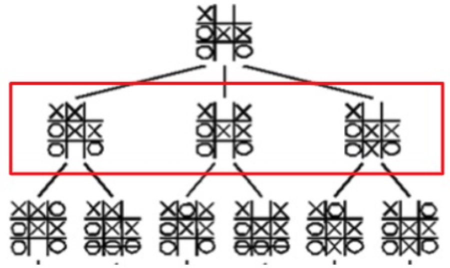


Figure 2.3: Stone shapes

- Since the tree search is called recursively, OpenMP is trivial.
- Even with OpenMP, CPU has limited capability of searching at a higher depth on a larger board.

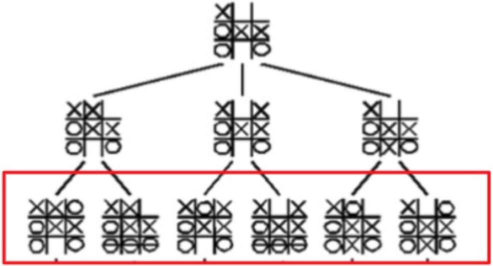
Solutions
Naive CUDA

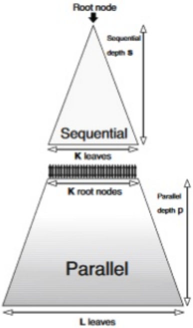
- Launch a GPU thread for each possible move from the root node.
- Introduces thread divergence -> very poor performance.
- A lot of work for little number of threads.



Solutions
Sequential-Parallel CUDA implementation

- Idea: Launch as many light weight GPU threads.
- Process some depth s on the CPU and get k leaf nodes.
- Calculate these k leaf nodes in parallel.





- The result from earlier branches are used to determine whether later branches should be examined or not.
- If we search multiple branches in parallel, those branches do not have the bounds from each other to work with.
- This will result in searching branches that would have been pruned in the serialized version.
- In theory, a thread searching one branch could update the other threads with its bounds when the thread finishes.
- However, it is difficult to implement on GPU since it would require blocks to be able to break other blocks out of recursive function calls.

- Idea: Search down the leftmost branch of the tree on CPU, and then search the remaining nodes in parallel on GPU.
- This allows the GPU branches to use the bounds from the first CPU-searched branch.

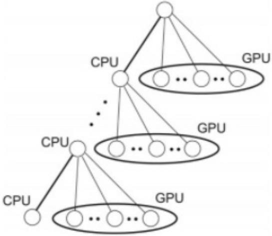
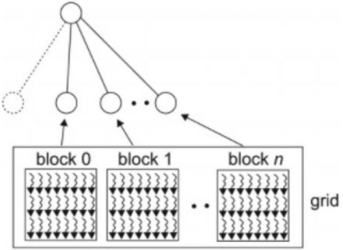


Figure 1. GPU-based PV-split



- One block per child node.
- Use that block to search down the subtree rooted at the child node.
- As depth increasing, GPU threads need to search increasingly larger subtrees.
- Even though there are many threads searching in parallel, each thread has to search a subtree serially.
- Therefore, we can further parallelize subtree searching.

Solutions

Dynamic Parallelism

- Launch a new kernel for each node to searched, instead of using one block to fully search a subtree.
- The overall structure remains the same. CPU will search one branch of the tree, and using the bounds obtained for GPU to search the remaining nodes.
- One possible trade off is the overhead of launching so many kernels.

Experiments

Performance to Calculate One Step

```
$ ./gomoku_cuda
0: 1 0 0 0 0 0 0 0 0
1: 0 0 0 0 0 0 0 0 0
2: 0 0 0 0 0 0 0 0 0
3: 0 0 0 0 0 0 0 0 0
4: 0 0 0 0 0 0 0 0 0
5: 0 0 0 0 0 0 0 0 0
6: 0 0 0 0 0 0 0 0 0
7: 0 0 0 0 0 0 0 0 0
8: 0 0 0 0 0 0 0 0 0

Size of options: 80
GPU one turn done
elapsed time = 14245.6 ms
0: 1 0 0 0 0 0 0 0 0
1: 0 0 2 0 0 0 0 0 0
2: 0 0 0 0 0 0 0 0 0
3: 0 0 0 0 0 0 0 0 0
4: 0 0 0 0 0 0 0 0 0
5: 0 0 0 0 0 0 0 0 0
6: 0 0 0 0 0 0 0 0 0
7: 0 0 0 0 0 0 0 0 0
8: 0 0 0 0 0 0 0 0 0
```

Results

Depth = 3

Time measured in seconds.

Type	9*9	12*12	15*15	18*18	21*21	24*24
Serial	0.506	5.56	34.72	158.13	571.07	1718.01
OpenMP (12 threads)	0.116 (4.36x)	1.126 (4.93x)	6.717 (5.17x)	30.75 (5.14x)	109.87 (5.19x)	358.98 (4.78x)
Seq-Par GPU (2-3)	0.324 (1.56x)	1.75 (3.17x)	10.94 (3.17x)	13.21 (11.97x)	37.79 (15.11x)	91.49 (18.78x)
PVS	1.328 (0.38x)	5.467 (1.02x)	155.25 (0.22x)	----	----	----
Dynamic	----	----	----	----	----	----

Observation

Depth $= 3$

- OpenMP achieved steady speedup when the board size grows.
- Seq-Par achieved very good speedup when the board size grows.
- PVS suffered when the board size grows.
- Dynamic took too long to finish at this depth.

Results

Depth $= 4$

Time measured in seconds.

Type	Board 6*6; Depth=4	Board 9*9; Depth=4	Board 12*12; Depth=4
Serial	0.317	12.669	271.388
OpenMP (12 threads)	0.0757 (4.19x)	2.712 (4.67x)	56.947 (4.76x)
Naive GPU	45.344 (0.007x)	1420.64 (0.009)	----
Seq-Par GPU (2-4)	1.649 (0.19x)	87.972 (0.14x)	1999.45 (0.13x)
Seq-Par GPU (3-4)	0.719 (0.43x)	11.91 (1.06x)	141.18 (1.92x)
PVS	0.203 (1.56x)	0.368 (34.43x)	2.75 (98.69x)
Dynamic	----	----	----

Observation

Depth $= 4$

- OpenMP achieved steady speedup when the board size grows.
- Naive GPU suffered when the board size grows.
- Seq-Par achieved very little speedup when the board size grows.
- PVS achieved impressive speedup when the board size grows.
- Dynamic took too long to finish at this depth.

Experiments

Performance to Calculate a Whole Game

```
Starting GPU game...
 0: - - - - - - - - 0 0 - X X X X - - - - -
 1: - - X 0 X - - - - - - - 0 - - - - - - -
 2: - - - - - 0 - 0 0 X - 0 - 0 - - 0 - - -
 3: - 0 - - - X 0 X - X - - - - X 0 - - - -
 4: - - - - - X - 0 - - X - - - - - - 0 - -
 5: - 0 0 0 - X 0 X - X X 0 - X - X - X - -
 6: - X 0 - - X - - - 0 - 0 - - - - 0 X X -
 7: 0 X - - - 0 X X - - - 0 - - X - - - X -
 8: 0 X - 0 - - 0 - X X - X 0 X - 0 0 - X -
 9: - X - - X - - X - - - - X X X - 0 X - 0
10: - X 0 X - X 0 X - - 0 0 - - - 0 - - - X
11: - X 0 X X - - - 0 - 0 - - 0 - - 0 - - X
12: - 0 0 - 0 0 0 - - X - - - - X - - 0 X -
13: - 0 - X X 0 - - 0 0 X - X X X 0 0 - - -
14: - 0 - X 0 X X X 0 0 0 - - 0 - - X - 0 -
15: - - - X X 0 - 0 X X - 0 0 X - X - 0 - -
16: - - - 0 - 0 0 - - - X X - - X - X - - -
17: X X - 0 0 X X 0 0 0 - 0 0 0 - - X - - -
18: 0 - - - - - X - - X - - - - 0 - X - - -
19: 0 - - - - - - X X - - - - - - - - - - 0

GPU PVS Game completed.
 elapsedTime = 61.2855 s
X won! Board Size: 20 * 20 Depth : 2
```

Results

Type	Board 8*8; Depth=2	Board 12*12 Depth=2	Board 16*16 Depth=2	Board 20*20 Depth=2
PVS	0.376183	1.60381	11.9597	61.2855

Our PVS and Dynamic implementation was fixed at the last minute. We will include the experiment results in final report.

Citation

- D. Strnad and N. Guid, "Parallel alpha-beta algorithm on the GPU," CIT, vol. 19, no. 4,pp. 269-274, 2011.
- K. Rocki and R. Suda, "Parallel Minimax Tree Searching on GPU", JST CREST, 2009.
- Li, L., Liu, H., Wang, H., Liu, T., Li, W.: A parallel algorithm for game tree search using gpgpu (2014)
- H. Liao, "New Heuristic algorithm to improve the Minimax for Gomoku artificial intelligence.", 2019.
- Marsland, T.A., Campbell, M.: Parallel search of strongly ordered game trees. ACM Computing Surveys (CSUR) 14(4), 533–551 (1982)
- Strnad, D., Guid, N.: Parallel alpha-beta algorithm on the gpu. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on. pp. 571–576. IEEE (2011)

Sequential-Parallel CUDA

- Needs memory optimizations for it run at higher depths and larger boards.
- Redundant calculations.

Questions?

Thank you!