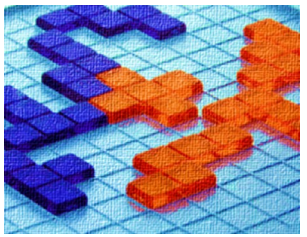


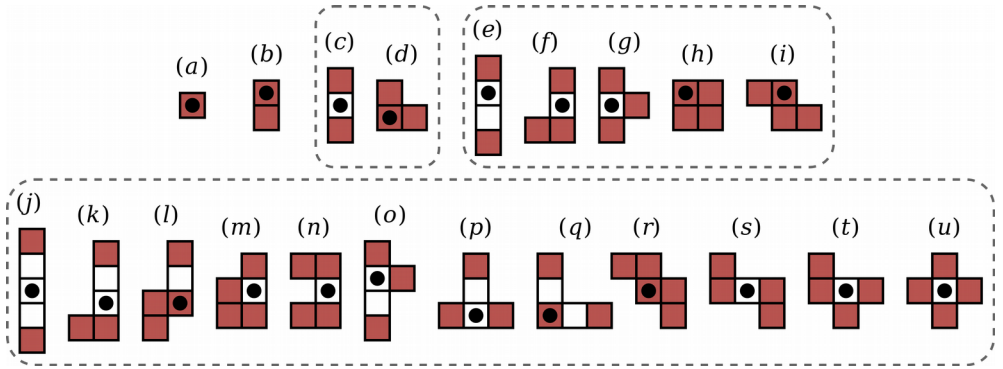
Competição
BlokusDuo
REC'2020

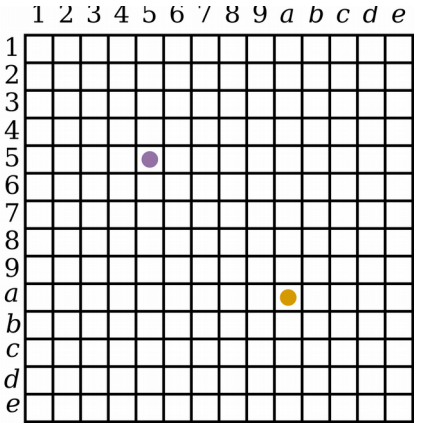
Playing BlokusDuo in a ZYNQ Device: A Quest for an Efficient Algorithm

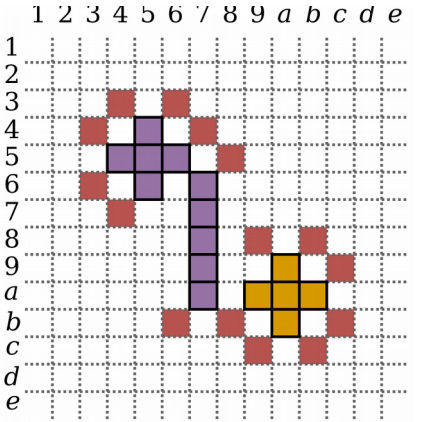
João Vieira joaomiguelvieira@tecnico.ulisboa.pt

BlokusDuo

- **14×14 game board**
- **21 game pieces with 5, 4, 3, 2 or 1 square(s) each**
- **First move** of **player 1 on *(5, 5)*** and **player 2 on *(a, a)***
- Pieces can only be laid on the board such that:
  - It makes **corner contact** with another piece of the same player
  - It makes **no edge contact** with piece of the same player
- **Goal is to lay the greatest amount of pieces**

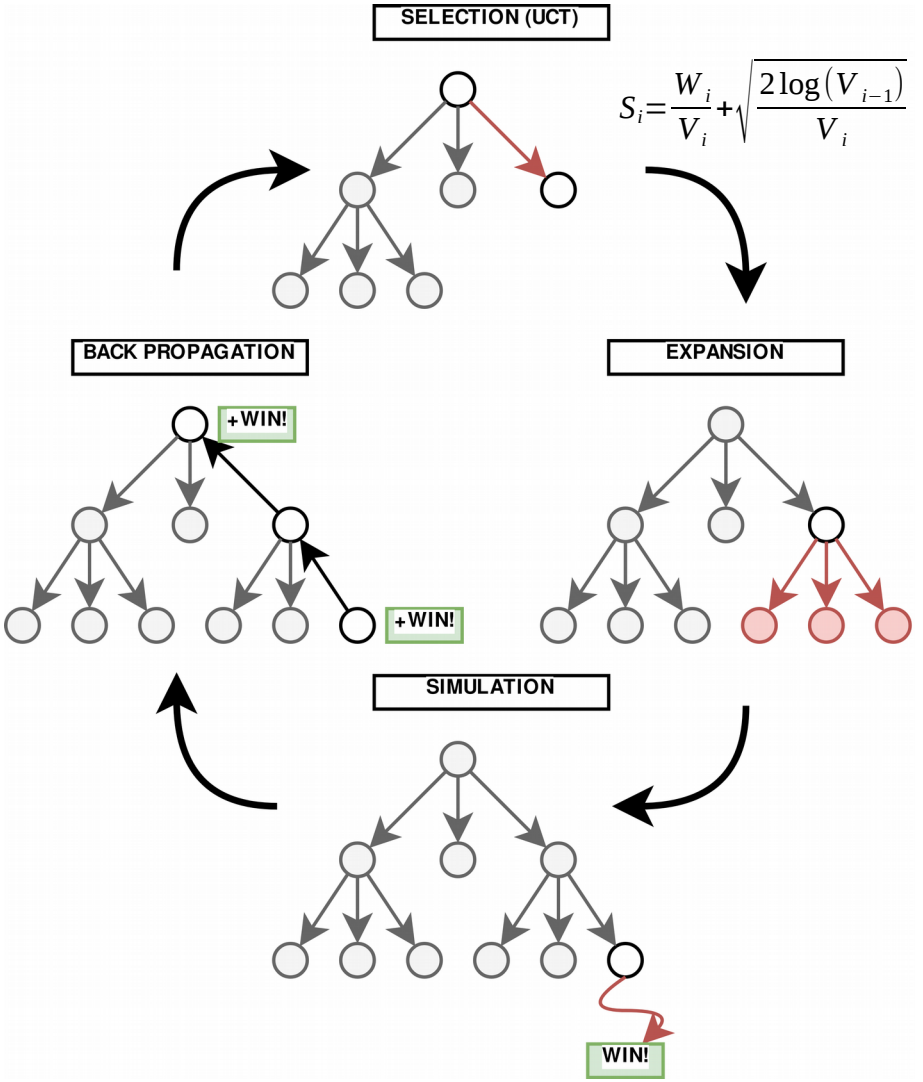






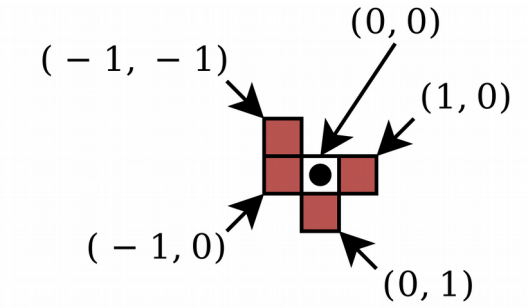
Monte Carlo Tree Search (MCTS)

- Heuristic algorithm combining **tree search** with **reinforcement learning**:
  - **Explore** possible paths
  - **Exploit** promising paths
  - **Chose best path based on past simulations**
- Balance exploration with exploitation using **Upper Confidence Bounds applied to Trees (UCT)**

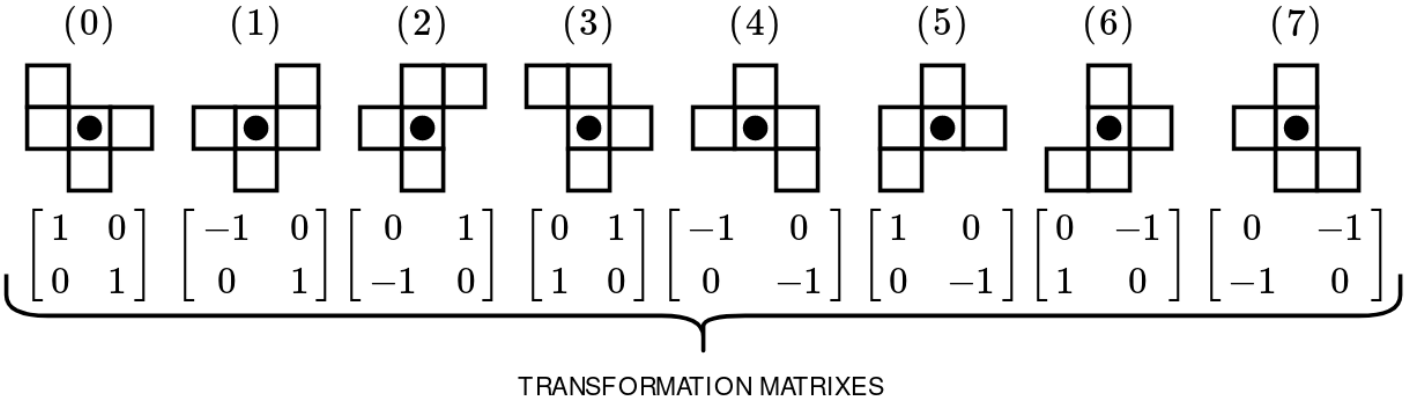


Data Structures

- Game pieces are represented as an array of coordinates



- Transformations are represented as linear transformation matrices

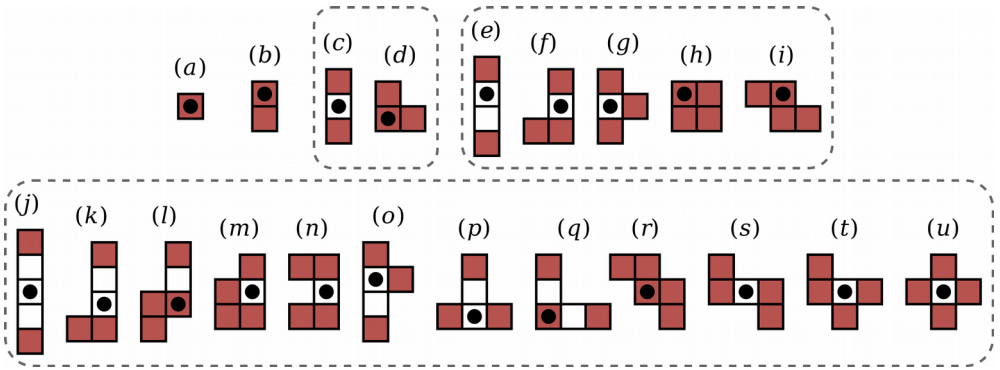


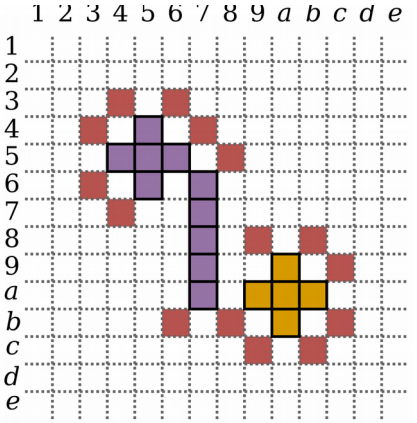
Expanding nodes as quick as possible

- **All possible moves have to be found**
- **Naive approach:**
  - **For each** square in the game board
    - **For each** game piece
      - **For each** variant
        - Try to center the variant of the game piece in the square
  - Very greedy approach leads to an unfeasible workload
- Consider only the cases where corner contact can be established

Expanding nodes as quick as possible (2)

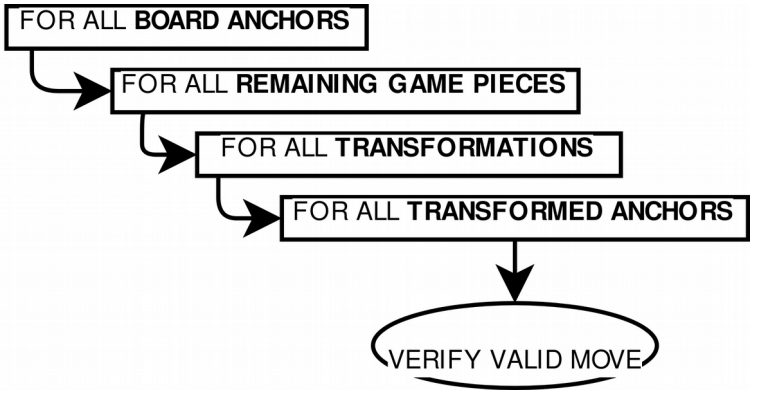
- **Definitions:**
  - **Anchor:** property of the game piece indicating indicates a square that can establish corner contact with an already-laid game piece
  - **Anchoring point:** property of the board indicating a square where an anchor can be laid establishing corner contact with an already-laid game piece

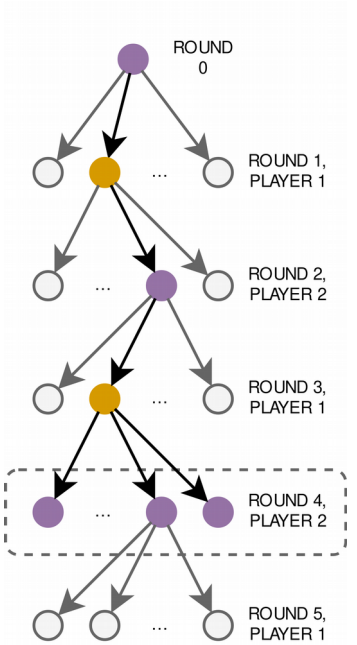




Expanding nodes as quick as possible (3)

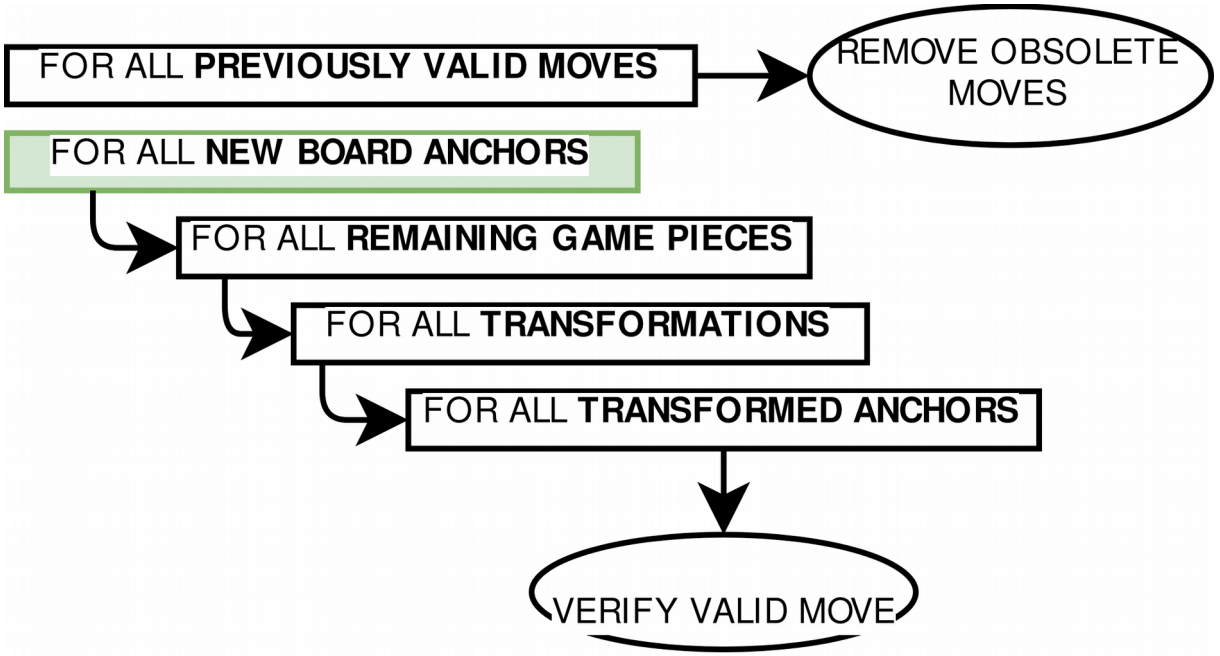
- **Finding all possible moves from anchors and anchoring points**





- Considering that **(1) players are not given new pieces** and **(2) anchoring points can only be created**:
  - There are **common moves in consecutive rounds**
  - **Finding moves can be done differentially** using only the anchoring points generated during the last round

Expanding nodes as quick as possible (4)



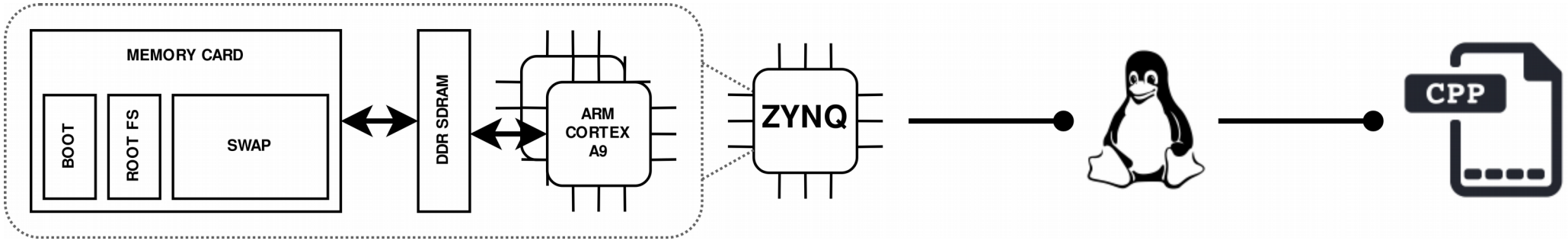
- Finding new moves differentially provides a performance boost of nearly to 2×

Modifications to the MCTS algorithm

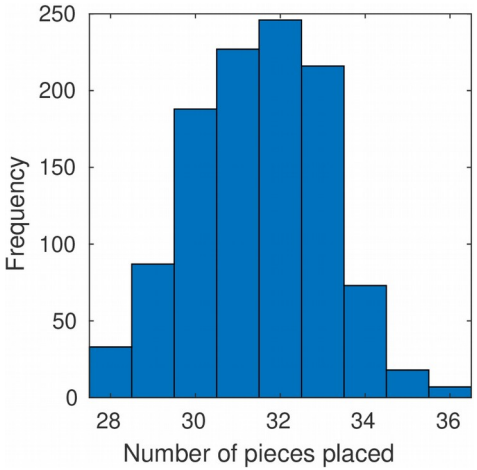
- Memory between rounds: **cumulative learning**
- Upon a limited number of simulations, **MCTS may not converge**:
  - Different heuristics have to be used:
    - **Number of squares** of the game piece
    - **Number of anchors** of the game piece
    - **Proximity to the opponent's last move**
    - **Proximity to the anti-diagonal** of the board

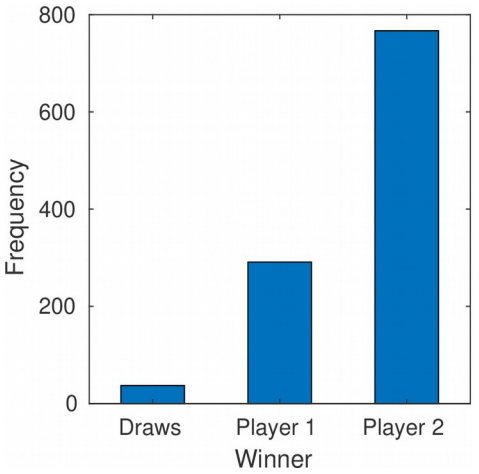
Hardware platform

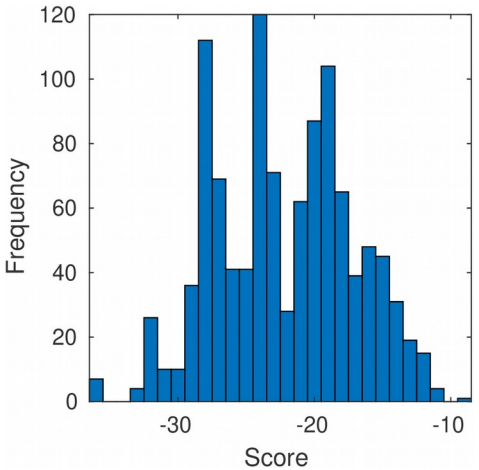
ZYNQ + Xillinux + C++

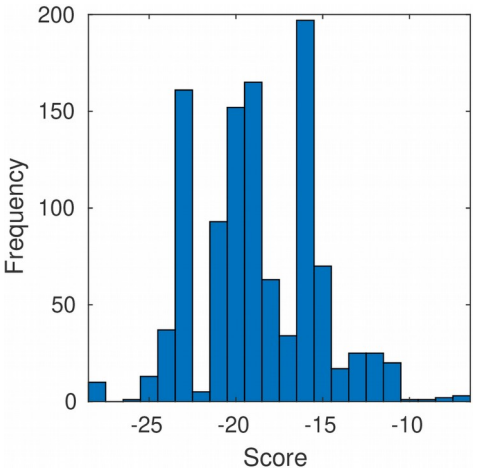


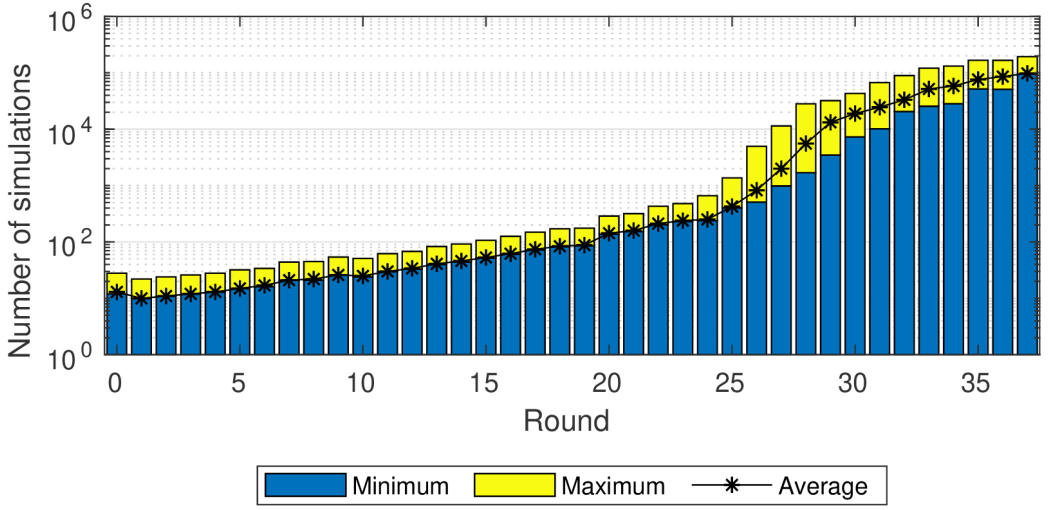
Results


Frequency
250
200
150
100
50
0
28
30
32
34
36
Number of pieces placed


Frequency
800
600
400
200
0
Draws
Player 1
Player 2
Winner


Frequency
120
100
80
60
40
20
0
-30
-20
-10
Score


Frequency
200
150
100
50
0
-25
-20
-15
-10
Score


Number of simulations
10^6
10^4
10^2
10^0
0
5
10
15
20
25
30
35
Round
Minimum
Maximum
Average

Conclusions

- **Software-only approach**
- Efficiency of **MCTS is dependent of the number of simulations**
- **Optimizing node expansion** is the key to increase the number of simulations
- Research directions could be:
  - **Further software optimization** (e.g.: optimize memory management)
  - **FPGA dedicated accelerator(s)** for finding new moves

Thank you! Questions?