Strong 1-dimensional Clobber Solver

TheSolvers

Taylor Folkersen, Zahra Bashir, Fatemeh Tavakoli

Outline

- Correctness
- Abstract Code Flow
- Database Structure
- Improvements
  - Dominated Moves
  - Game Values
    - Rules
    - Statistics
- Results
- Next Step

Correctness

Bugs

- Generating illegal moves
- Database generation
- Non null terminated strings
- 0-0 instead of None

Correctness

Fixing bugs

- We also conducted a pipeline for testing our solver based on the reference solver that we have.
- We recognized the source of our few wrong results and debugged our code to fix them.
- Finally, having a correct verified solver.

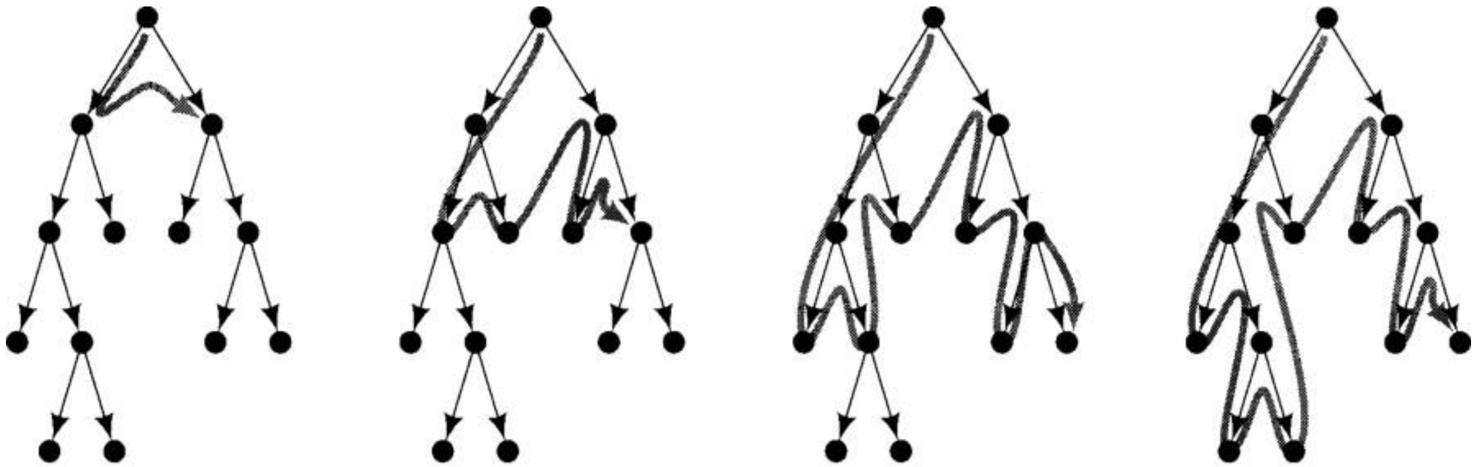
Abstract Code Flow

Simplify board()
lookup TT, if found → return the result
generating subgames()
for each subgame:
   outcome, game value, dominated moves← db.get(subgame)
if the game can be solved using just the outcomes → return the result
else if the game has a value using sum of subgames →return the result
else:
solve the game with the help of dominated moves, based on TT, and ID search.

The Search Algorithm

Iterative Deepening:

We use the iterative deepening search, with a limit on the number of searches that are allowed to reach terminal nodes (nodes at the current maximum depth, or actual terminal nodes).



Depth bound = 1 Depth bound = 2 Depth bound = 3 Depth bound = 4

Database Structure

		outcome	player1 player2 dominated moves	game value	link to the equivalent node
entry:	**WBBBBBB**	**OC_B**	**1001…1 0100..11**	**05000**	4

- Creating a database for up to 20 tiles with outcome and game values while computing dominated moves for up 12 to tiles in 30m 12.221s
- Computing a database up to 20 tiles without computing dominated moves and outcome takes 52m 47.186s.

Improvements - dominated moves

Removing dominated moves

Steps of determining if a move is dominated

- generates the list of moves for a player at a state
- compare all pairs of moves (i, j) and their respective resulting games I, J
- check if a single player wins I - J regardless of who goes first
- If this happens -> one of moves i and j are marked as dominated
- save the dominated move in the database

For now, we store dominated moves for all connected boards from sizes 1 through 12.

Improvements - Game Values

Game Values

We extended our database to be able to save some game values.

- First of all, saved some initial values (rule 3-5) in our db.
- Saved the number of ups, downs and star values.
    - (two digits for up, two for down, and one for star)

WBBB = WB^3 = $\uparrow\uparrow *$ = 2001

Improvements - Game Values

For the remaining boards, if the board value is not set:

- Find the game value for left and right
  - Play all the left and right options for one level
- Return the max/min value for L/R
  - Comparing left values (or right values) using rules in fig1 and fig2
- Set the final value based on the rules (1) using L and R values.

$$\uparrow\uparrow > \uparrow + \uparrow^{2} > \uparrow + \uparrow^{3} > \uparrow > \uparrow^{2} > \uparrow^{3} > 0 > \downarrow^{3} > \downarrow^{2} > \downarrow > \downarrow + \downarrow^{3} > \downarrow + \downarrow^{2} > \downarrow\downarrow .$$

fig 1

$$\uparrow\uparrow * \quad > \uparrow * \quad > \quad * \quad > \downarrow * \quad > \downarrow\downarrow *.$$

fig 2

Game value Rules

1

$$\boldsymbol{n} \cdot \uparrow \;= \{0 \mid (\boldsymbol{n}-1) \;\cdot \uparrow *\} \quad if\, \boldsymbol{n} \geq 1$$

$$\boldsymbol{n} \cdot \uparrow * = \begin{cases} \{0 \mid (\boldsymbol{n}-1) \cdot \uparrow\} & if\, \boldsymbol{n} > 1 \\ \{0, * \mid 0\} & if\, \boldsymbol{n} = 1 \end{cases}$$

2

$$n \cdot \downarrow \;= \{(n-1) \cdot \downarrow * \mid 0\}$$

$$n \cdot \downarrow * = \begin{cases} \{(n-1) \cdot \downarrow \mid 0\} & if\, n > 1 \\ \{0 \mid 0, *\} & if\, n = 1 \end{cases}$$

3

$$(BBW)^n \;=\; \left[\frac{n+1}{2}\right] . \uparrow$$

4

$$B^m W^n = 0\,,\; for\; m, n \geq 2$$

5

$$WB^n = (n-1). \uparrow \;+\; n.*$$

Game Value Statistics

Number of exact game values within the $\uparrow\uparrow$ and $\downarrow\downarrow$ bound

*	48	$\downarrow$	32
0	74	$\uparrow\uparrow$	2
$\uparrow$	32	$\downarrow\downarrow$	2

Examples

example of some initial patterns:

- ↑↑ = (BBW)^4, (BBW)^3
- ↑ = (BBW)^2, (BBW), WBB,
- * = BW, WB
- ↓↓ = (WWB)^4, (WWB)^3
- ↓ = (WWB)^2, (WWB), BWW,
- ↑↑* = WBBB
- ↓↓* = BWWW
- 0 = 34 (B^mW^n, W^mB^n, m,n>=2)

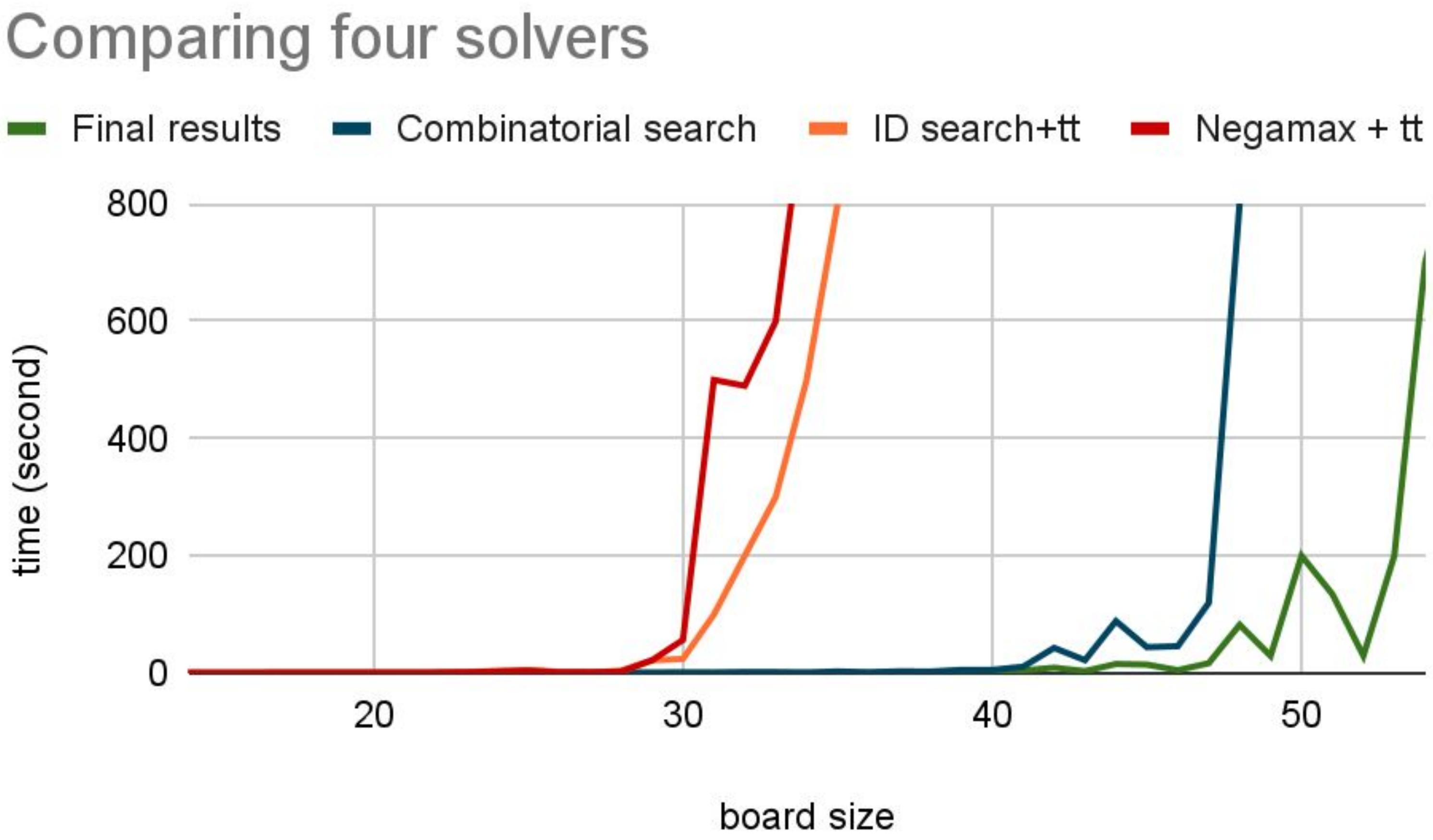
Other Improvements

We implemented the rule "One N game with several L/R subgames is the first-player win for Left/Right."

- The correctness of this improvement is also tested.
- It speeded up the solver a bit (not noticeable), and the number of nodes that were searched is decreased.

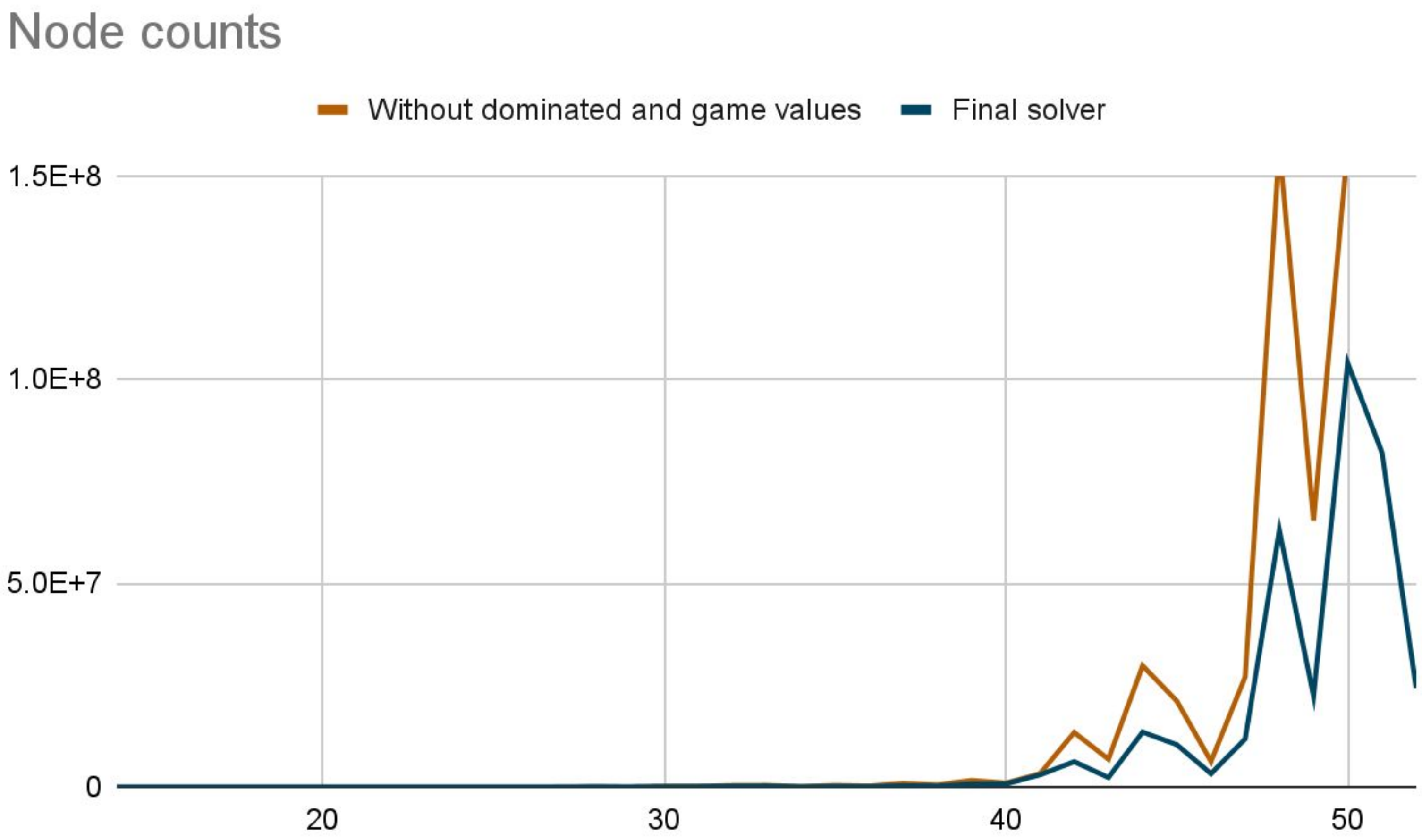
Trying alpha-beta pruning on heuristic values → did not improve our solver with our current heuristic.

Results - runtime



The runtime result of consecutive B and W stones of different length

Node counts vs board length



Result of consecutive B and W stones of different length

Overall runtime on 500 random test cases

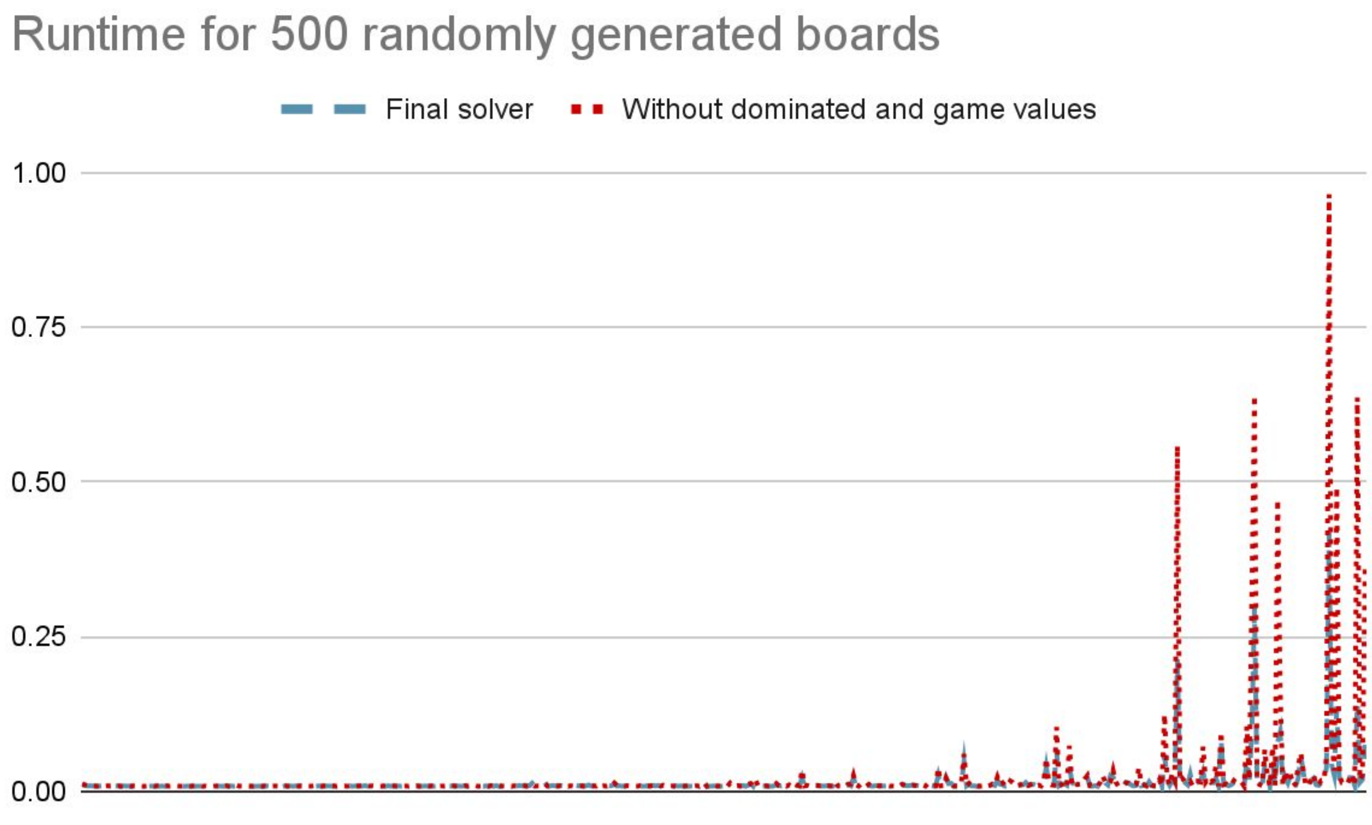
Overall runtime for solving 500 randomly generated boards:

Final results 7.18s

Previous best combinatorial solver:

15.93s

[10-52 tiles]



Next Steps

- Simplifying subgames
- Lower/upper bounds of subgames
- Improve subgame deletion heuristic

Demo

References

[1] J Fernando Hernandez, Using Left and Right Stop Information for Solving Clobber.

[2] Michael Albert et al., Introduction to Clobber.

[3] Claessen. "Combinatorial Game Theory in Clobber." Maastricht University. 2011.

[4] Kristopher De Asis, A CGT-informed clobber player.

[5] Max Blankestijn, Stops, All-Smalls & Infinitesimals, Lessons In Play.