Exploring Positional Linear Go

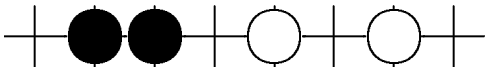
Noah Weninger Ryan Hayward

Department of Computing Science
University of Alberta
Canada

Advances in Computer Games, 2017

Linear Go

Go on an $Nx1$ board.



Motivation

Solving Go is hard. Maybe $Nx1$ Go is easier and can lead to intuition about NxM Go?

Not necessarily. Some unique challenges arise.

Motivation

- Little previous work on Nx1 Go.

Motivation

- Little previous work on Nx1 Go.
- No known unconditionally alive local patterns which are not dependent on ko rules.

Motivation

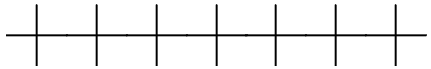
- Little previous work on Nx1 Go.
- No known unconditionally alive local patterns which are not dependent on ko rules.
- Cycles appear much more often on a linear board.

Motivation

- Little previous work on Nx1 Go.
- No known unconditionally alive local patterns which are not dependent on ko rules.
- Cycles appear much more often on a linear board.
- Predicting the outcome of long and complex cycles is essential to playing well.

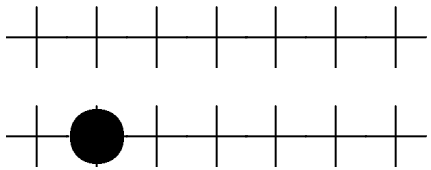
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



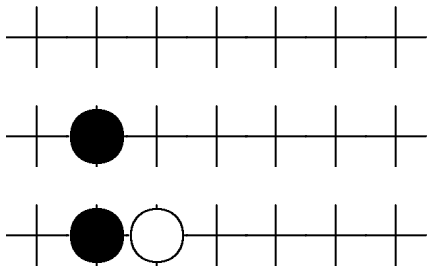
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



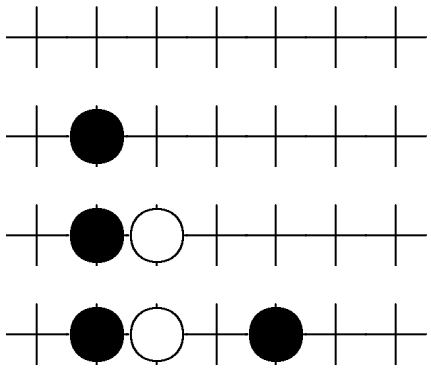
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



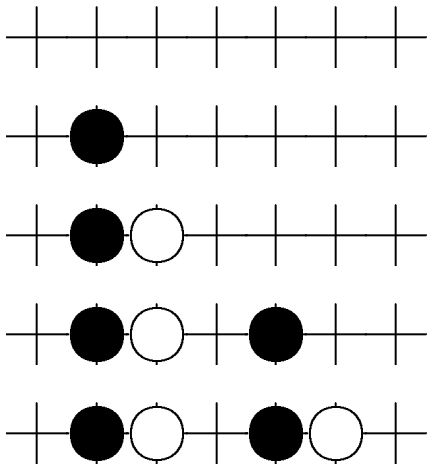
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



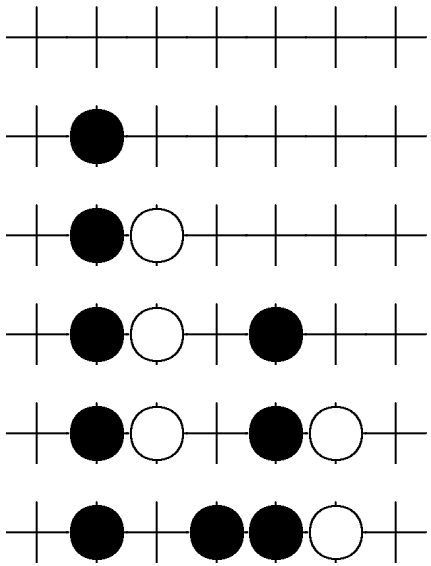
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



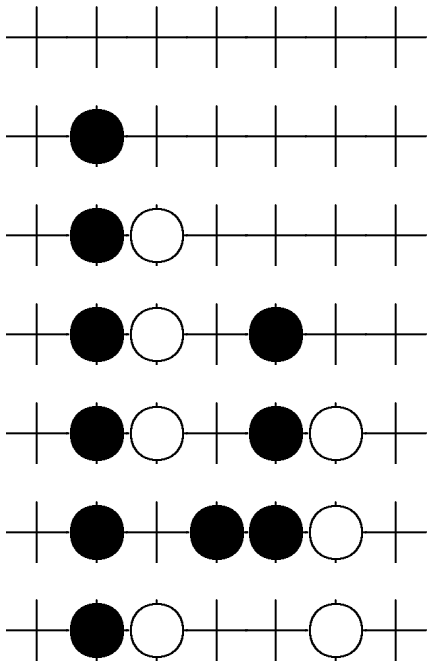
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



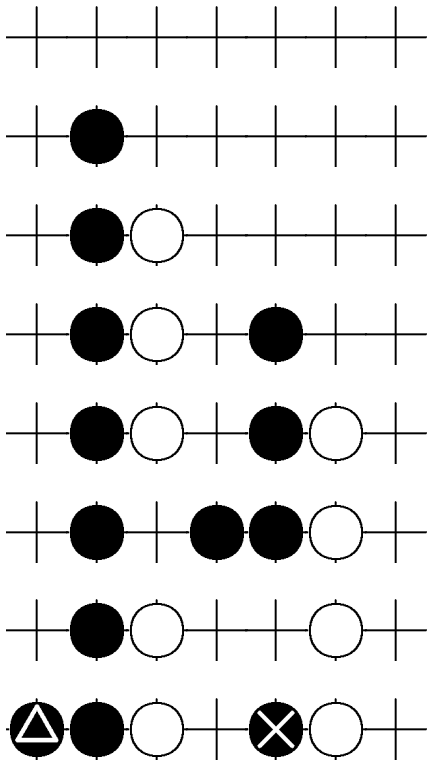
Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



Positional Linear Go

We consider Tromp-Taylor rules aka the Logical rules with no suicide and positional superko. Game ends after 2 consecutive passes.



On rules

Previous work by Erik van der Werf et al. titled *Solving Go for Rectangular Boards*, describes a solver *Migos* which uses different rules and solves Nx1 Go for N up to 12.

Our Goal

- Develop theory about Linear Go.

Our Goal

- Develop theory about Linear Go.
- Develop understanding about how to play Linear Go.

Our Goal

- Develop theory about Linear Go.
- Develop understanding about how to play Linear Go.
- Determine exact minimax values for many board sizes and opening moves.

Theory of Positional Linear Go

Our initial attempt at creating a solver for Positional Linear Go was ineffective, solving only up to size 1x6. To improve this, we sought to develop theory which we could use to enhance the search process.

Definitions

- **Position** A board configuration.
- **State** A position, along with a set of positions that have been previously visited, and a flag to indicate whether the last move was a pass and the current player to move.

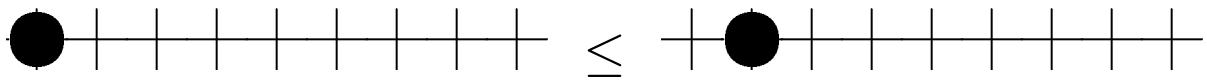
Preliminaries: Empty board pass

- **Theorem 3.** If the first player passes from the empty board, their minimax score is negated.
- Proof: use strategy stealing.

Preliminaries: Empty board score

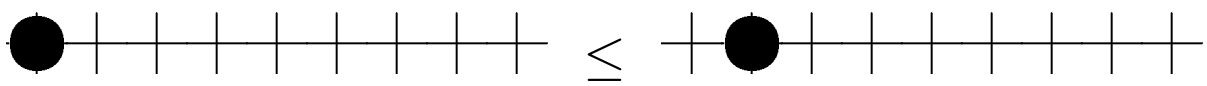
- **Theorem 1.** The first-player empty-board minimax score is non-negative.
- Proof: Assume score is negative. Take the first move to be pass. If the opponent passes, the game ends with a score of 0. If the opponent does not pass, then their minimax score is negative by our assumption. So they prefer to pass, giving a score of 0. The score is then non-negative. Contradiction.

Cell 2



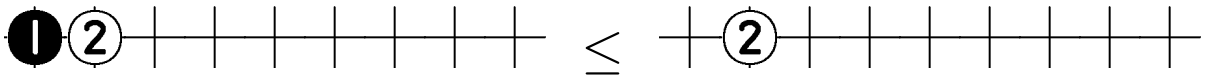
- We say a stone k away from an end of the board is in *cell* k.
- **Conjecture 1.** If cells 1 and 2 have never been played in then playing in cell 2 is at least as good as playing in cell 1.

Cell 2



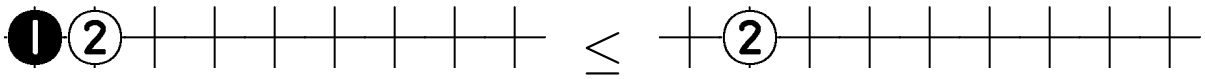
- We say a stone k away from an end of the board is in *cell* k.
- **Conjecture 1.** If cells 1 and 2 have never been played in then playing in cell 2 is at least as good as playing in cell 1.
- This has been verified for $N \leq 7$ by performing a search from the empty board which forks to test both decisions.
- It does not appear trivial to prove in general.

A weaker cell 2



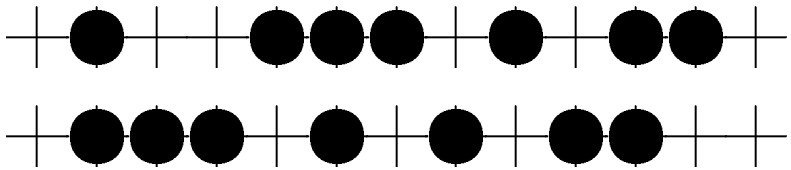
- **Theorem 2.** If cells 1 and 2 have never been played in then passing is at least as good as playing in cell 1 and being immediately captured.

A weaker cell 2



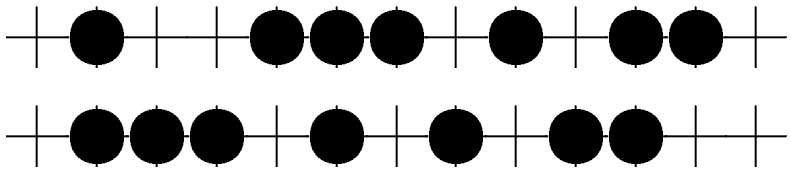
- **Theorem 2.** If cells 1 and 2 have never been played in then passing is at least as good as playing in cell 1 and being immediately captured.
- Proof idea: Any optimal strategy for the variant where black is captured can be used for the variant where black passes.

Total states



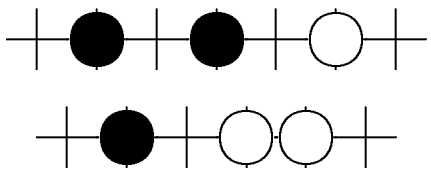
- In some cases where the board is filled with stones of a single color except for a few gaps, we call this position *total*.
- If the score of the current position is equal to the minimax value of the state, we call this state *stable*.

Total states



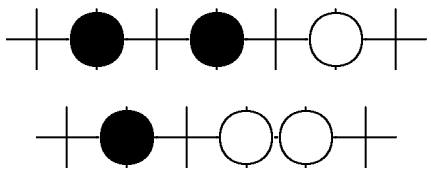
- In some cases where the board is filled with stones of a single color except for a few gaps, we call this position *total*.
- If the score of the current position is equal to the minimax value of the state, we call this state *stable*.
- **Theorem 4.** All total positions are stable.
- Proof idea: All moves by the opponent are either single-cell suicide or immediately put into atari.
- Pruning total states prevents the solver from wasting time by filling it's own eyes.

Loosely packed states

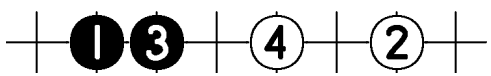


- A similar argument to theorem 4 holds even when both players have stones on the board. We call these states *loosely packed*.
- **Theorem 5.** For $N \leq 7$, all loosely packed states are stable.

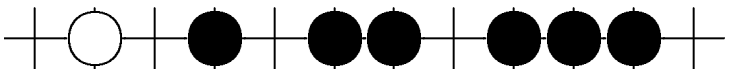
Loosely packed states



- A similar argument to theorem 4 holds even when both players have stones on the board. We call these states *loosely packed*.
- **Theorem 5.** For $N \leq 7$, all loosely packed states are stable.
- This does **not** extend to $N > 7$. For example consider this position where the score of the position is -1 but the minimax $+1$:



Telomeres



- When the ends of the board meet certain criteria, we can prune some moves or provide bounds.
- Here, white need not play into cell 1 or cell 3, because that will only further decrease it's score. So we find a lower bound of $N - 5$ for black.

Telomeres: Theorem 7

complement	prune	lower bound on μ_x
(--]	(**]	n
(o-]	(o*]	n
(---]	(*-*]	$n-5$
(o--]	(o**]	n
(-o-]	(*o*]	$n-5$
(--o]	(oo*]	n
(----]	(*--*]	$n-7$
(o---]	(o***]	n
(-o--]	(*o-*]	$n-7$
(--o-]	(*-o*]	$n-7$
(---o]	(oo-*]	$n-5$
(o-o-]	(o-o*]	$n-5$
(-oo-]	(-oo-]	?
(ooo-]	(ooo*]	$n-5$
(-----]	(*----]	?
(o----]	(o****]	n

Figure: Pruning and bounds for Theorem 7. * cells pruned. ? no bound.

Telomeres: Theorems 6 and 8

In the cases of `(-o-]` and `(-o-o-]`, we can often reach a stronger conclusion:

The minimax score of the state is at least the score of the position.

Solver implementation

We implemented a solver using alpha-beta search with

1. Iterative deepening
2. MTD(f)
3. Transposition tables
4. Enhanced move ordering
5. Knowledge based pruning

Iterative deepening

- We perform many searches from the root while gradually increasing the depth cutoff.
- Since we want to solve the game exactly, care must be taken to correctly propagate exactness information up the tree: a node is exact iff the beta cutoff can be hit using some exact child, all children are exact, or it is a endgame leaf.

MTD(f)

- We perform a series of null-window alpha-beta searches until converging on the minimax value.
- f parameter is a guess which can improve convergence time.
- Iterative deepening gives us good guesses early on, so MTD(f) can converge quickly: usually with less than two null-window searches.

Transposition tables

We use two separate transposition tables:
A traditional *state* $\rightarrow$ *bounds* table and a *position* $\rightarrow$ *move ordering* table.

state → *bounds* table

- Table needs to be indexed by a complete state to be correct.
- Difficult to implement efficiently and accurately because a state can be very large and of variable size: we store history using either a bitset or hash table, selected by board size.
- In our experience it is only useful when an entry represents a significant amount of work — measured by subtree size — and an exact node.
- Prevents iterative deepening and MTD(f) from repeating work.

position → *move ordering* table

- Searching *good* children first can speed up alpha-beta search.
- Good children are those that have:
  - Exact values: likely to be in the transposition table.
  - High scores: likely to cause a beta cutoff.
  - Small subtree size: estimated from previous iteration of iterative deepening.
- These characteristics are chosen to minimize the time it takes to hit a beta cutoff.
- It is faster to index this table by the position instead of the state, but we need some policy to correct a possibly invalid move ordering because collisions can happen.

Knowledge based pruning

- We use the theorems presented to prune nodes from the search.
- Unfortunately, we found that some pruning theorems increase the search time in certain cases because they cause easily proven bounds to be skipped.
- Useful pruning theorems were those regarding cell 2, total states and loosely packed states.

Results

Empty board scores line up previous work by Van der Werf et al.
We found scores for all opening moves on sizes up to 1x9.

n	minimax value by 1st-move location								
2	-2	-							
3	-3	3	-						
4	-4	4	-	-					
5	-5	0	0	-	-				
6	-6	1	-1	-	-	-			
7	-7	2	-2	2	-	-	-		
8	-3	3	-1	1	-	-	-	-	
9	-4	0	-1	0	0	-	-	-	-

Figure: Missing entries follow by left-right symmetry.

Q&A