Not Like Other Games - Why Tree Search in Go is Different

Martin Müller
ETL, Tsukuba, Japan
mueller@etl.go.jp

Abstract

Large-scale minimax search has been used with great success in many games, but not in Go. We investigate the reasons for the difficulty of applying minimax search to Go, using late stage endgames as a test case.

1 Minimax Tree Search in Go

Deep minimax search is the engine powering most computer programs for two-player games with perfect information. It has led to overwhelming success in many popular games, such as chess, checkers, shogi, Othello, awari, Chinese chess, gomoku, and Nine Men's Morris. However, the same approach does not work in Go, because of the large number of possible moves in each position, the length of a game, and the difficulty of developing an accurate evaluation function.

This paper contains two contributions leading towards a deeper understanding of minimax search in Go: First, Section 2 identifies three factors that complicate minimax search in Go: detecting terminal positions in the search, the unavoidability of generating pass moves, and local position repetition or *ko*. Second, Section 3 studies the problems of minimax tree search in Go on hand of an example from a late stage endgame, which is known to be efficiently solvable by *Decomposition Search* [4].

1.1 Types of Minimax Search in Go

In computer Go, two different types of minimax search are commonly used: selective global search and goal-oriented search.

The objective of global search is to maximize the full-board score. Because of the complex evaluation and high branching factor of Go, full-board minimax search is typically highly selective and shallow [2], searching only tens or hundreds of positions per move decision, in contrast to the hundred of thousands or millions of nodes typically searched in other games.

Local minimax searches focus on specific tactical goals, such as trying to capture or save a specific block of stones, life and death problems, or testing the safety of territory. In goal-directed search, evaluation consists only of a simple test, which is much faster than full evaluation. Another advantage is that the number of moves is typically much smaller than in full-board search.

2 Specific Problems of Minimax Search in Go

The following three problems are specific to minimax search in Go:

Recognizing terminal positions In Go, it can be very difficult to judge whether a position is terminal, or whether valuable moves remain.

Pass moves In Go, there is no *zugzwang*: in positions where there is no good move, a player is allowed to pass.

Local position repetition or *ko* Full board position repetition is illegal in Go. However, local position repetition occurs frequently.

2.1 Recognizing Terminal Positions

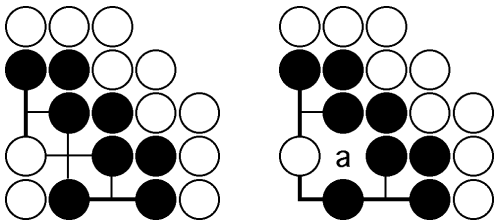

Figure 1: Recognizing terminal positions

In many games, detecting the end of the game is simple, for example if a specific piece such as the king is captured or a player runs out of moves.

In Go, a position is terminal if no more points are contested, and all points can be classified as black, white or neutral. Such classification can be hard, as shown by the two similar-looking examples in Figure 1: in both cases Black has completely surrounded a corner area, including empty points and some white stones. The example on the left is a terminal position: Black's area is safe. However, in the example on the right Black needs another move to prevent a White move at 'a' leading to coexistence in *seki*.

2.2 Pass Moves

In most board games, passing is illegal. If a player cannot move, the game is over and the outcome is determined by the rules, for example a stalemate in chess.

In contrast, a pass move must always be generated during minimax search in Go, unless it can be proven that at least one other good move exists. In positions where there is no good move, players must be allowed to pass, instead of being forced to damage their own position. Figure 2 shows such a position where all moves are bad. Adding pass moves can substantially increase the size of the search space.

2.3 Ko: Local Position Repetition

Local repetition or ko is made possible by interposing a forcing sequence outside the local scope. Ignoring the possibility of such ko during local search gives misleading results. In the same local position, a crucial move that is currently illegal due to position repetition might become legal later, if a move elsewhere on the board has changed the position.

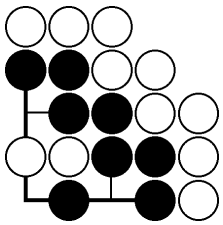

Figure 2: Seki: a position without good moves

If the outcome of a local search depends on a player winning a ko fight, that player usually has to pay a price by ignoring some opponent moves on the rest of the board. In complex ko fights, many different outcomes and tradeoffs are possible, making exact analysis very difficult within the framework of goal-oriented minimax search.

2.4 Interactions Between the Three Go-specific Problems

The interaction of the three Go-specific problems identified above leads to further complications:

Pass and recognition of terminal positions The number of consecutive pass moves cannot be indefinite. Therefore, after two or three consecutive passes the resulting position must be statically evaluated, even if the position is unsettled.

Ko and terminal positions Some ko fights are very favorable for one player, so in practice the other player wins them by default. However, it is hard to formulate general rules for handling all such cases.

Pass and ko Pass moves in conjunction with ko fights also lead to unresolved positions.

3 Minimax Search in Go: An Endgame Case Study

This section studies minimax search in Go on hand of an example from the late endgame. As a search problem, this example is simple because there are already many safe stones and territories, there are no ko fights, and it is easy to detect terminal positions.

3.1 Problem C.11 Revisited

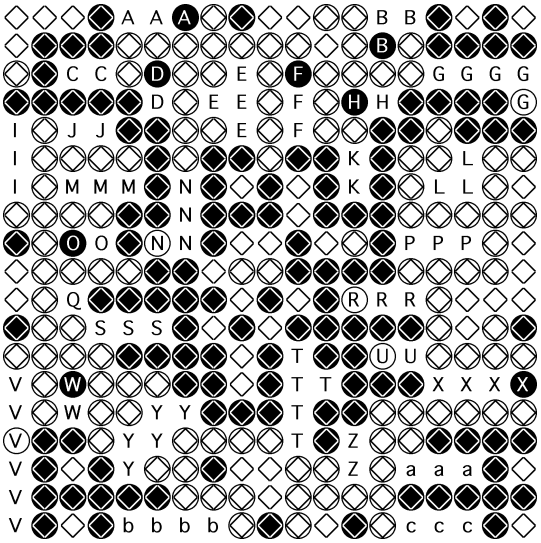

Figure 3: Endgame problem C.11 [1]

Figure 3 shows an endgame problem equivalent to problem C.11 in [1]. This problem is decomposable and can therefore be solved efficiently by the method of *Decomposition Search* [4], which uses *local combinatorial game search (LCGS)* to analyze each small local area independently.

Figure 4 shows the effect of a series of enhancements to alphabeta minimax search on sets of subproblems of this late stage Go endgame. The number of nodes searched by LCGS is given for comparison. The horizontal axis shows the subset of endgame areas that was searched. The vertical axis shows the number of nodes searched on a logarithmic scale. The following search enhancements were tested:

Move sorting (sort) sorts moves according to the size of the local area.

Global best move pruning (global) prunes all except one move candidate in positions where a globally best move exists.

Local best move pruning (local) prunes other local move candidates if a locally best move is found.

Partial order move pruning (POprune) uses pruning with a partial ordering [5] of move classes *better-than-dame*, *dame* (not generated), *pass*, and *other*.

3.2 Evaluation of Test Results

All four tested enhancements lead to substantial reductions in the number of nodes searched. The one outstanding improvement is the introduction of local move pruning, which greatly reduces the branching factor of the search. Global and partial order move pruning are most effective in near terminal positions. Earlier in the game they don't help, since no pruning move can be found by global pruning, and almost all moves are in the *other* category of partial order move pruning. Move sorting works well in every case.

4 Summary and Future Work

The game of Go has a number of properties that make minimax search difficult even for small-size problems. On the other hand, adding game-specific knowledge can greatly enhance performance and reduce the size of search trees by many orders of magnitude. Similar experiences have been reported for the single-agent search problem of Sokoban [3].

While global minimax search cannot compete with the local search method used in Decomposition Search, adding game-specific knowledge for move pruning and using *locally informed* global search greatly improves the efficiency of minimax searches. This result points the way towards developing a common framework that combines the power of local search methods with the generality of global minimax search. Such future work includes:

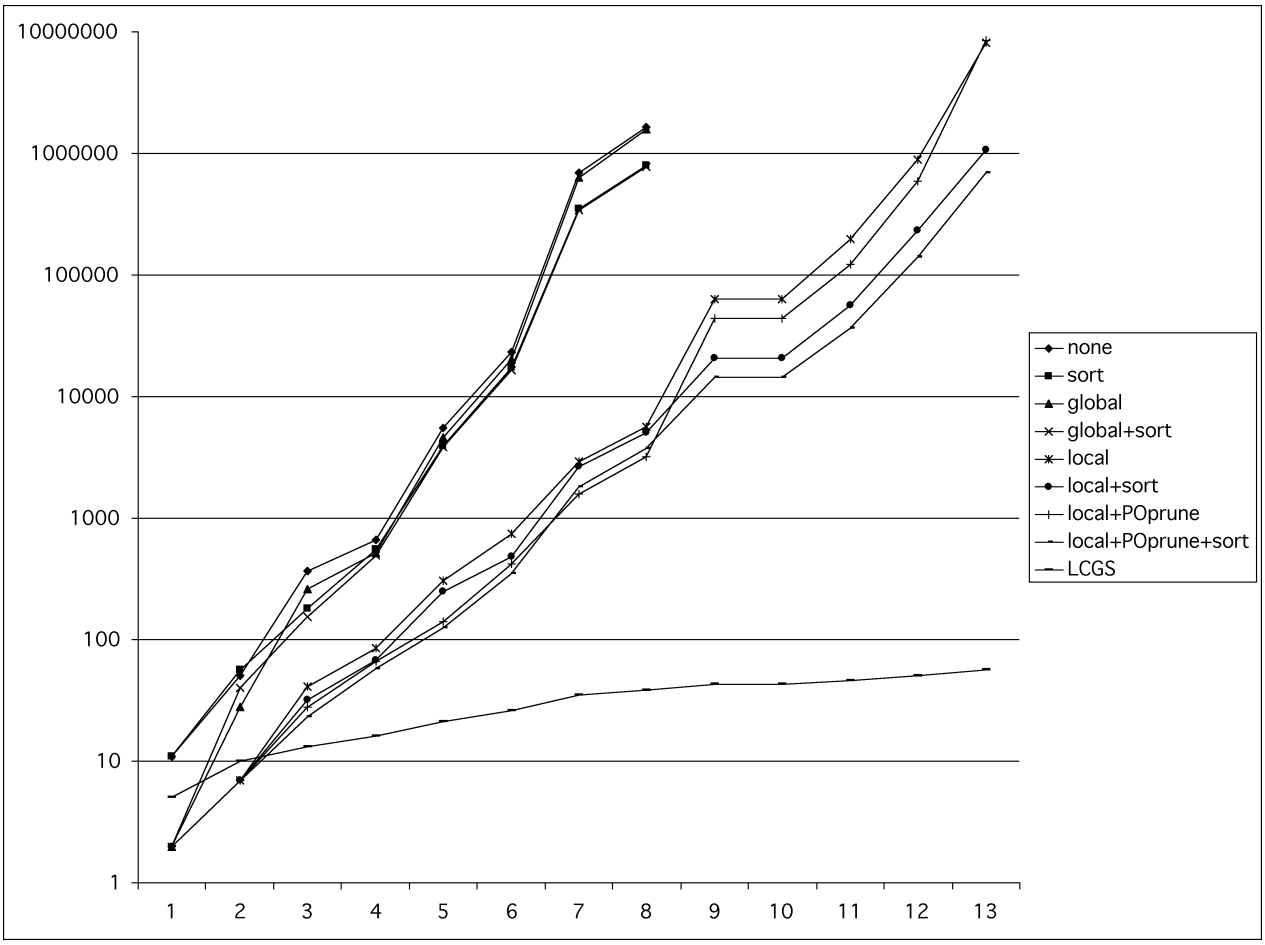

Figure 4: Minimax search enhancements in Go endgames

- Use global search in the case when there are dependencies between local games.
- Integrate more local game ideas into a global search framework, such as *miai*: using search, identify sets of two or more local games that cancel each other, and can be removed from the search process.
- Compute bounds on the value of nonterminal positions by static analysis, and use these bounds during search.

References

[1] E. Berlekamp and D. Wolfe. *Mathematical Go: Chilling Gets the Last Point.* A K Peters, Wellesley, 1994.

[2] K. Chen. The move decision process of Go Intellect. *Computer Go*, 14:9–17, 1990.

[3] A. Junghanns and J. Schaeffer. Domain-dependent single-agent search enhancements. In *IJCAI-99*, pages 570–575, 1999.

[4] M. Müller. Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames. In *IJCAI-99*, pages 578–583, 1999.

[5] M. Müller. Partial order bounding: Using partial order evaluation in game tree search. Technical Report TR-99-12, Electrotechnical Laboratory, Tsukuba, Japan, 1999.