Comprehensive Analysis of Life and Death in the Game of Go

Tyson Williams
Honors Advisor: Degang Chen  Project Advisor: Jack Lutz  Project Supervisors: Ramon Mercado, Kyle Blocher

Abstract
The goal of this project was to create a program to analyze life and death problems in Go subject to a user-defined scope. The program determines the state of a group of stones; possible states are alive, dead, or ko. While this program may not be the first to discern among these states, it is the first to evaluate the position with respect to several other metrics, including points, ko threats, and who starts the ko. The program was tested against positions with known solutions, and in two cases, the program found better solutions than what was previously published.

Introduction
Go is an intractable game for computers to play at a human level, but they have their advantage. Humans are skilled at determining scope, while computers can execute an exhaustive search flawlessly. To enhance our ability to play go, some researchers are using computer programs to exhaustively search sub-problems of Go and comparing the results against the most trusted human conclusions and go theory (1). One area that deserves more attention is that of life and death.

Methods
To combine the strengths of humans and computers, the user must define the scope of the problem by specifying where each player is allowed to play. Then, the computer will exhaustively search each possible line of play. The search is terminated when the defending player is unconditionally alive as determined by Benson’s algorithm (2) or when the defending player is dead because there is no way to make life. The other state that the program can detect is ko, which happens when the board is exactly the same as it was two moves ago. In this case, the program assumes that the player that moved first will win the ko and then searches for the best subsequent outcome.

The best move in a local situation often depends on the context of the whole board. In order to return a set of outcomes which contains the best outcome for all possible cases, the program compares each outcome using several metrics. Depending on the state, the possible metrics are the number of points, number of ko threats, and who starts the ko. When comparing two outcomes of different state, life is better than ko and ko is better than death. When comparing two outcomes with the same state, one outcome is greater than the other when all the metrics of the first outcome are better than all the metrics of the second outcome. Otherwise, they are considered equal. The possible outcomes are returned according to the Minimax algorithm and Zobrist hashing is used to check for repeated board states.

Results
The program was tested with over 50 life and death problems from two trusted sources, Sensei’s Library (3) and GoProblems.com (4), and better solutions were found for problems from each site. Sensei’s Library discusses a shape known as Rectangular Six on the Side (Figure 1). Two points, A and B, are presented as miai to threaten the life of the white group using two ko threats each according to Figures 2 and 3. However, my program found a different line of play, Figure 4, which contains three ko threats. This is clearly better for the attacking player.

On GoProblems.com, problem 9121 relates to a group called Rectangular Six in the Corner. After playing (one of) the correct moves, shown in Figure 5, A and B are claimed to be miai. However, my program disagrees. My program showed that Black at B allows white to play at A for certain life and a total of one ko threat (Figure 6). But, as shown in Figure 7, Black at A forces white to play twice for a total of two ko threats.

Future Work
Extending this work can happen in two primary ways. From a Go perspective, I would like to be able to detect seki and allow for pass as a move. From an algorithms perspective, I would like to implement more artificial intelligence techniques, such as alpha-beta pruning.

References
(3) Sensei’s Library: Front Page <http://senseis.xmp.net/>
(4) Goproblems.com <http://goproblems.com/>