Chapter 1

Artificial Intelligence techniques used in Go

Abstract

This paper surveys the most used artificial intelligence techniques used in Computer Go. Even though Computer Chess reached grandmaster level over a decade ago, Computer Go is still nowhere near the level of professional Go players. This article explains what it means for an AI technique to be successful and what AI techniques are used by the several Computer Go programs available.

1.1 Introduction

Board games like chess, checkers, reversi and go have been the subject of many studies during the last 40 years [Millington and Funge, 2009]. Especially for chess many AI techniques have been developed, and chess computers reached grandmaster level in 1994, when Kasparov, the highest rated chess player in the world, was beaten by a computer chess program in a timed tournament [Burmeister and Wiles, 1995]. The game Go, played by two players using black and white stones that can be placed on the 19x19 grid, has a much more complex strategy even though the rules are very simple. It has been estimated that the number of different game states is larger than the number of atoms in the universe. This makes it impossible to investigate every situation so smart AI techniques have to be used. This paper will give an overview of the techniques that are currently used for Go.

1.2 Research question

This paper investigates the successful AI techniques most used in Go, with the following research question:

- What successful AI techniques are used in computer Go?
- What successful AI techniques are most used in computer Go?

To answer this research question, the following subquestion needs to be answered:

- What AI techniques are most used in computer Go? How successful are Go programs? What defines the success of a Go program?

1.2.1 Structure of this paper

In order to answer the research question we will first take a closer look at the game Go. Section 3 will be all about the game itself; readers who are already familiar with the game may skip this section. Section 4 describes the success of Computer Go so far; there are many tournaments and prizes and games of professional players vs. computers that give a good view on Computer Go’s current position. Section 5 is all about the AI techniques used in Go. This paper ends with a conclusion in section 6 where the research question will be answered.
1.3 The game of Go

Go is an ancient oriental board game for two players with very simple rules, yet strategies for it can be very complex. This section first explains the rules of the game, then addresses the different ways of counting score, and finishes with some very basic techniques that are often used by beginning amateurs. The rules that are described here are the official rules used by Go associations [AGA, 2010] [BGA, 2010].

1.3.1 Basic rules

The game is played on a grid. The size of the grid may differ; usually it is 19x19 lines but sometimes smaller grids are used, especially for children or players who are new to the game. In this paper we will assume a grid is 19x19 unless it is stated otherwise. In the beginning, the grid is empty. The two players alternately place white and black stones the intersections of the grid. Stones can no longer move once they are placed on the board, unless they are captured by the opponent’s stones. The goal of the game is to control an area larger than your opponent’s. The scoring rules will be explained in subsection 3.2, but we will first mention some terms commonly used in Go.

Group Vertically and horizontally adjacent stones of the same color are called a group (or sometimes a chain or String).

Liberty A liberty is a free position horizontally or vertically adjacent to one of the stones in a group. It is a property of the stone as well as of the group.

Capturing If a player places a stone on the last liberty of the group of the opposite player, the group is said to be captured and all stones are removed.

Suicide It is usually not allowed to place a stone in such a way that it leaves your own group without liberties. Doing so is called suicide. This is not true when you can capture your opponent’s group by doing so, since their group will first be removed, leaving your group with a liberty after placing the stone. Whether suicide is allowed or not does not matter a lot for artificial intelligence techniques, since suicide is rarely beneficial.

Ko rule Sometimes it is possible that certain moves keep repeating, making the game go on indefinitely. The Ko rule prevents this: it prohibits a player to make a move that makes the games return to the state before the opponent’s last move. This is a very important rule for AI techniques, because it ensures all games have a finite number of moves and therefore prevents infinite loops.

Passing A player may pass instead of placing a stone. This usually occurs when they believe they cannot increase their score anymore. When both players have passed consecutively the game ends and the score is counted.

Living, dead or unsettled A group is said to be living, dead or unsettled. A living group cannot be captured even if it is the opponent’s turn. In practice this means the group must have at least two liberties. A dead group cannot avoid to be captured, even if the owner can move first. An unsettled group is one that can be captured if the opponent moves first but can avoid being captured if its owner moves first.

Handicap It is possible to give an advantage to one of the players by allowing him or her to place a number of stones on the grid before the game is actually started. This is called a handicap for the opponent. For example, if one player gets to place five stones before the game starts, the other player has a handicap of five stones.

1.3.2 Scoring

Even though there are some minor differences between rules from different countries (especially China and Japan/Korea), these do usually not affect the game play. However, we still explain both Chinese and Japanese scoring rules since they are the most common and differ significantly.
Area scoring  The score is the number of stones on the board plus the number of empty stones surrounded by their stones. This is the Chinese rule for scores.

Territory scoring  Japan and Korea use a different way of counting score. They count all the stones that each player captures during the game as well as the empty points surrounded by a player’s stone. Sometimes there can be disagreement on if a group is dead or not. In that case the players usually continue the game until both players pass consecutively again. Under area scoring, the score will then be based on the position of the game after the second time they passed. Under territory scoring they will return to the position after the first passes and remove the dead groups.

1.4 Successfulness of Go programs so far

Most AI techniques that are suggested for Go are implemented in some Go program in order to test them. Those Go programs are then ranked by letting them play to human Go players, since human Go players are ranked according to their skill. The rank system consists of kyus and dans, starting with 30 kyus going up to 1 kyu and then 1 dan going up to 9 dan. Usually only the first game a computer plays against a human is considered, since the human may learn from the program’s mistakes but the program is often not able to learn from the human’s mistakes (see also the section on machine learning). The following subsections describe the current Go programs and their success so far. Most AI techniques that are suggested for Go are implemented in some Go program in order to test them. Those Go programs are then ranked by letting them play to human Go players, since human Go players are ranked according to their skill. The rank system consists of kyus and dans, starting with 30 kyus going up to 1 kyu and then 1 dan going up to 9 dan, shown in the table below. Usually only the first game a computer plays against a human is considered, since the human may learn from the program’s mistakes but the program is often not able to learn from the human’s mistakes (see also the section on machine learning). The following subsections describe the current Go programs and their success so far.

<table>
<thead>
<tr>
<th>Rank Type</th>
<th>Range</th>
<th>Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-digit kyu</td>
<td>30-20k</td>
<td>Beginner</td>
</tr>
<tr>
<td>Double-digit kyu</td>
<td>19-10k</td>
<td>Casual player</td>
</tr>
<tr>
<td>Amateur dan</td>
<td>1-7d</td>
<td>Advanced amateur</td>
</tr>
<tr>
<td>Professional dan</td>
<td>1-9p</td>
<td>Professional Player</td>
</tr>
</tbody>
</table>

1.4.1 Go programs

Go programming started in the late 1960s and got a big boost in the mid 1980s, when PCs were more readily available and Go computer program tournaments were organized [Mueller, 1995]. Harrison named the most recent and most successful Go programs in his 2010 thesis [Harrison, 2010], which will be discussed here. We will also mention the techniques the programs use; these techniques are discussed in section 5.

GNU Go  GNU Go is a free, open source program, allowing many people to contribute to it. Its first version came out in 1999 and right now it is about six to seven stones weaker than the top commercial programs [GNU, 2010].

Many Faces of Go  The Many Faces of Go is one of the older Go programs available. It first came out in 1981 and was developed by David Fotland [David Fotland, 2010]. It has booked some successes in the late 80s and early 90s when it won the United States Computer Go competitions [David Fotland, 2010]. In 2008 it was the World computer go 19x19 and 9x9 champion [David Fotland, 2010].

Go++  Go++ is one of the most successful Go programs currently available. Since 1996 it was placed in the top 3 of every tournament it entered [Michael Reiss, 2010].

Crazy Stones  Crazy Stones was developed by Rémi Coulom. The development of this program was carried out as an initiative from the French Ministry of Research. It has won quite a few prizes since it started (about five years ago) [Remi Coulom, 2010]. It uses Monte-Carlo evaluation and has apparently been very successful doing so. In 2007 the Monte-Carlo search was improved with patterns learned by a Bayesian technique[Remi Coulom, 2007]. The Bayesian technique will not be discussed in this paper.
MoGo  MoGo is the result of Yizao's internship, where Rémi Coulom participated. He shared his experience from programming CrazyStone, resulting in another Monte Carlo based program using a tree search algorithm based on the UCT algorithm and patterns. There are high-performance clusters for main events [Wang and Gelly, 2007] [INRIA, 2010].

1.4.2 The success of Go Programs so far

The success of Go programs is usually defined by how well they do against professional Go players. There are many tournaments organized for this purpose, we will first mention the most popular ones. There have also been prizes for excellent computer Go programs, they will be described next. We conclude this section with some of the results of computer versus human games.

Tournaments  The Computer-Go website has a very useful and updated calendar with all the upcoming Computer Go tournaments in it [Nick Wedd, 2010a].

KGS Computer Go Tournament  This website organizes many online tournaments where computer Go developers can test their programs at all times [KGS Computer Go Tournament, 2010].

European Go Congress  The European Go Congress is organized annually and usually involves computer Go events too [European Go Congress, 2010].

US Go Congress  The US Go Congress is held annually as well, and sometimes involves a computer Go tournament as well [US Go Congress, 2010].

Computer Go Forum  This tournament is held annually as well and always includes computer go games (hence the name) [Computer Go Forum, 2010].

Prizes

Ing Prize  This prize was offered by the Taiwanese banker Ing Chang-ki and was the driver for many Go programmers to improve their programs. It was offered at the Ing Cup (a world computer Go congress). The winner of the tournament could challenge a young professional Go player with a handicap. If the computer won the match the prize would be rewarded and a new prize was announced: a larger one for winning a game with less handicap. The Ing prizes expired in 2000 when Chang-ki died. In 1997 Handtalk was the last program that claimed a prize of 250,000 NT dollars for winning a match with an 11-stone handicap against three 8 or 9 year old professionals [Nick Wedd, 2010b].

Selection of computer vs. human games  At the tournaments mentioned earlier in this chapter, Computer Go programs play against humans. This is a good way to test the program, since all humans are ranked. Therefore it is easy to see if the program has improved, even if they play against different humans. The Computer-Go website keeps a table with information about all the interesting human vs computer Go games [Nick Wedd, 2010b].

February 2010  Zen won the game against Nam Chi-hyueong (1p) on a 9x9 grid with no handicap.

August 2009  Fuego and MoGo won the game against Shen-Su Chang (6d) with 4 stones handicap on a 19x19 grid.

February 2009  MoGo won the game against Shi-Jim Yen 6d with no handicap on a 9x9 grid.

August 7 2008  MoGo won the game against Myungwan Kim (8p) with a handicap of 9 stones. The game was played on a 19x19 grid.

1.5 Artificial Intelligence techniques used in Go

This section describes the AI techniques that are most used in Computer Go programs. We start with the basic AI techniques needed to solve a game, then describe some possible ways to search the game tree, discuss machine learning and see how programs can divide the game into different stages to improve their performance. The section ends with some information about the difference between local and global factors in Go.
Chapter 1. Artificial Intelligence techniques used in Go

1.5.1 Basic AI techniques

All programs that play games have some basic requirements. They need a way to represent the different game states, a way to generate moves, define some goal states, and develop an evaluation function. Here we will describe this fundamental properties of Go programs. A lot of this information came from a paper called "AI Techniques Used in Computer Go" [Burmeister and Wiles, 1997], which describes just these techniques and how they were implemented in the programs available at the time.

State Representation  Go is a perfect information game, meaning that all the information about the state of the game is available at any time. Each intersection point of the grid is either empty or occupied by a black or white stone. The size of the state-space is therefore $3^{361}$ or $10^{172}$, compared to a state-space for chess of approximately $10^{50}$ and othello of $10^{30}$ [Allis, 1994]. However, describing the game state like that is not very useful for a computer. It will need information about the groups (adjacent stones), but this is very difficult in practice, because it is hard to determine which groups will be useful. The appropriate level of representation may vary on the sub-task being performed, e.g. tactical analysis, life-and-death analysis, or assessing territory [Burmeister and Wiles, 1997].

Move Generation  The average branching factor (number of possible moves at a certain moment in time) is very large (200) compared to that of chess (35) [Burmeister and Wiles, 1995]. This branching factor does involve all possible moves, even though some moves could be eliminated by a human at first sight. Goal-directed search is used to read out ladders of about 60 ply deep. These ladders usually have only one or two possible moves [Burmeister and Wiles, 1997].

Goal States  The goal of Go is to acquire more territory than your opponent, either by surrounding empty points or by capturing their stones. The goal state of Go is more complicated than that of chess (check mate), since it largely depends on the opponent’s stones, while check mate does not. The ending of a game in Go is not very clear. The game ends in principal when both players pass, and then the score is counted. However, if there is discussion about the score, the players will continue the game until they do agree on the score. It is difficult for a computer to determine when to pass [Burmeister and Wiles, 1997].

Evaluation Function  The life-and-death status of groups have to be considered to evaluate a certain Go position. It consumes a lot of time and is done by tactical search (see also the next section). Determining the winner from any position is P-space hard

1.5.2 Searching the Game tree

Some small games like Tic-Tac-Toe can be solved by searching the complete game tree. All possible moves and game states can be searched and the best move can easily be decided. However, Go is too complex for such a game tree search, with a game tree size of $10^{360}$ compared to $10^{123}$ for chess, and there are less than $10^{100}$ atoms in the universe. Searching parts of the game tree is possible. Here we describe some of the common ways to do this.

Goal Search  Goal search is used to look for nodes in the game tree that accomplish specific tasks, like establishing life or death, creating eyes, connections, cuts, safety or territory, and capture. These sub-goals may all help to reach the final goal: win the game by acquiring the most territory [Harrison, 2010].

Heuristic Evaluation Functions  Creating a good heuristic evaluation function is difficult, but they are used in most programs. Some useful heuristics are: moves that capture the opponent’s stones and moves that protect your own stones. These heuristic evaluation functions are usually quite complex. They largely decrease the branching factor, enabling the program to look further down in the game tree [Harrison, 2010].

Tactical search  Tactical search is used for purposes like determining whether strings are dead or alive, whether eyes are formed and determine the life-and-death status of groups. Tactical search is a heuristic device that has two move generators: one for attacking moves, and one for defensive moves. The moves suggested by them are sorted according to criteria like liberties of liberties and simple eye shapes. Next, an alpha-beta depth-first search. The search is limited to

Research assignment  5
a maximum number of nodes. Tactical search is a type of goal-oriented search [Burmeister and Wiles, 1997].

**Life-and-Death Search** Some programs use perform life-and-death analysis to determine the status of a group. Others just use a form of tactical search for this purpose. Life-and-death search is another goal directed type of search. There is a static life-and-death evaluator that determines the status of a group. The search is done to save or kill a group. If the goal is not yet achieved, more moves are generated and the search is continued. The life-and-death evaluator is called at each node during the search to determine if the goal has been achieved [Burmeister and Wiles, 1997].

**Monte-Carlo** Monte-Carlo search strategies are used a lot in computer Go programs and results have been quite good. The Monte-Carlo evaluation functions lead to precise evaluations by playing a number of random games from the current game state and infer the value of each of the possible next moves from these evaluations [Chaslot et al., 2006]. The move that does best on average is then chosen [Brugmann, 1993].

### 1.5.3 Machine learning

A lot of recent research has been done on the use of machine learning in Go, especially using neural networks [Donnelly et al., 1994] [Richards et al., 1998] [Lubberts and Miikkulainen, 2001] [de Groote, E., 2005]. In this subsection we will discuss some of the most used types of machine learning [de Groote, E., 2005]: supervised learning, reinforcement learning and we conclude the subsection with some information about neural networks.

**Supervised learning** With supervised learning, learning is done using examples. An example consists of two things: the input signal, and the desired output. This output could for example be decided by a human being. It is the task for the program to learn a function that produces the desired output given the input [de Groote, E., 2005].

**Reinforcement learning** In the past decades a lot of research on reinforment learning has been done by animal psychologists. In reinforcement learning, winning a game leads to a reward whereas losing the game causes a punishment. The difficulty here is that it is unclear which moves contributed to the loss or win. Two popular types of supervised learning are Temporal-Difference learning (used to learn evaluation functions) and Q-learning (moves are evaluated instead of positions) [de Groote, E., 2005].

**Neural Networks** Neural networks are often used to map the features of a game state to a desired output, since it uses an implicit representation of the game states. The state space of Go is simply too big for an explicit representation. The neural network learns by updating the weights of mappings between features of game states and outputs [de Groote, E., 2005]. A lot of research has been done on these so called evolutionary techniques in neural networks [Donnelly et al., 1994] [Richards et al., 1998] [Lubberts and Miikkulainen, 2001].

### 1.5.4 Different approaches for different stages of the game

Go can be divided into three phases of the game: the opening, midgame and endgame. Each of these phases requires different techniques to be used. The phases will be discussed here.

**Opening** The opening stage of the game, also called fuseki, can be compared to the book openings in chess. It is usually wise to start by placing stones in the corners. Specific sequences in the corners are called fuseki. Most good Go programs have databases with fuseki moves, usually between 5,000 and 50,000 [Donnelly et al., 1994]. Usually Go programs play most of their opening based on the openings stored in their look-up table. The look-up table is created by storing many successful openings of games played before.

**Midgame** The midgame is much more complex than the opening. The branching factor is usually between 200 and 300, making it impossible to look ahead more than a few moves when considering all possibilities. This is where AI techniques are needed most to select good moves and eliminate the others. There are some problems with midgame techniques: they are usually difficult to apply and error-prone [Donnelly et al., 1994]. Midgame is where the AI techniques
Chapter 1. Artificial Intelligence techniques used in Go

mentioned in this chapter (searching the game tree, machine learning) are most useful, since smart
techniques have to be used as there are too many possibilities to search the entire game tree.

**Endgame** The endgame, like the opening, is much easier than the midgame. The branching
factor is reduced and local patterns can be very useful. However, the score does not usually change
in the endgame so playing a good endgame does not contribute a lot to the score [Donnelly et al.,
1994]. The endgame can usually be played by searching the entire game tree, so the program
knows which move will have the best outcome.

Having considered the three phases, we can conclude that the endgame is usually not much of
a problem, just like the opening. However, the midgame is very difficult as the branching factor
may be very large and much of the research focuses on this phase of the game.

1.6 Conclusion

To answer the research question, we will first answer the second subquestion about what defines
the success of a Go program.

1.6.1 Successfulness of Go programs

The success of Go programs can be measured by looking at how well they do against humans.
There are many tournaments that allow the program developers to test their programs. However,
since most programs use a combination of techniques, it is hard to tell how successful a single
technique is. Computer Go is not yet anywhere near master level, but it will probably turn out
to be that a good Go program will need a combination of techniques. In 2010, there are Go
programs that have beaten 1p human players without a handicap. The 1p rank is the lowest level
of professional human players (there are 10 levels in total), so there are Go programs that are at
a professional level. However, they are still nowhere near grandmaster level yet. In this paper
we described some of the most used AI techniques in Computer Go. We discussed several ways
of searching the game tree and machine learning and looked at what the effect is of splitting the
game in different phases.

1.6.2 AI techniques used in successful Go programs

Most Go programs use a combination of techniques. All programs need things like state represen-
tation, move generation and an evaluation function. Usually, a table is used to look up opening
moves; the midgame is played by searching (parts of) the game tree or machine learning, and the
endgame is played by looking entirely into the game tree.

1.6.3 Successfulness of AI techniques

The success of Go programs can be measured by looking at how well they do against humans.
There are many tournaments that allow the program developers to test their programs. However,
since most programs use a combination of techniques, it is hard to tell how successful a single
 technique is. Computer Go is not yet anywhere near master level, but it will probably turn out
to be that a good Go program will need the combination of techniques. Even though we cannot
contribute the success of the programs to a single technique, we can say something about what
techniques have been tried in past programs but are not used anymore (they are not successful)
and the ones that are commonly used (they are).

In conclusion, the most successful AI techniques that are currently used in Go programs are Monte
Carlo and goal directed search. A lot of research has been done on machine learning, but this is
not yet used a lot in Go programs. Results of early studies are promising though.

1.7 References


Harrison, B. Move Prediction in the Game of Go. 2010.


