# Computer Go

Jens Lehmann

July 1, 2004

This manuscript was written during the seminar "Game playing computers and artificial intelligence" of the Department of Computer Science at the Dresden University of Technology in the summer term 2004. It introduces the game of Go in general and describes the current state of the art in Computer Go.

# Contents

# 1 Introduction

Go is a very popular board game, which offers an overwhelming variety of strategic and tactical possibilities to the player. Despite its simple rules the game is very complex and believed to be the most challenging existing board game. According to the legends the game was invented in China before 2000BC and brought to Japan in the 7th century AD. The game is still most popular in eastern Asia, especially in Japan, China and Korea, but the popularity throughout other parts of the world increases. Go is not only fun to play, but also teaches intellectual abilities, concentration and balance.

In the countries where Go is most popular there are professional players, who learn the game fulltime from their childhood. Some live near their master and are teached until the early twenties. There was a time in China where Go was one of the Four Arts, with music, poetry and painting being the other ones.

After chess Go is the board game, which most research efforts were put into. Despite all these efforts, there has only been moderate success in developing a good computer Go program. All programs can be beaten by average club players. For this reason Computer Go is the biggest challenge for AI game programmers today.

The intention of this paper is to find reasons for humans being so much stronger than computers nowadays, to present techniques used in Computer Go and give an outlook on future possibilites.

# 2 Rules of Go

## 2.1 Basic Setup

Go is played on a square board with horizontal and vertical lines on it. You can see an example in figure 1. Usual boardsizes are 9x9, 13x13 and 19x19. There is a black and a white player, who alternatingly play stones. Stones are played on the intersections of lines (not in the fields like in other games). The black player moves first. Once a stone is placed it cannot be moved. A player can also "pass", which means not to do anything in his turn. Two consecutive passes end the game.

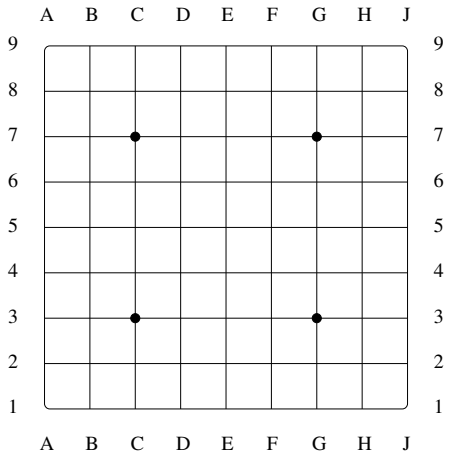


Figure 1: empty 9x9 go board

## 2.2 Capturing Stones

During a Go game stones can be captured. Stones of the opponent player are captured, if they are completely surrounded by your stones. If Black plays a stone on E6 in figure 2 it captures the white stone E5 like shown in figure 3.

A stone or a block of stones is surrounded, if there are opponent stones in every adjacent (horizontally or vertically across the lines of the board) intersections. The block G1-H1 in figure 4 has one free intersection at J1. By filling this intersection (figure 5) Black can capture the white stones. What would happen, if it were White's turn and White decides to play J1? The White stones would be surrounded by Black and therefore are captured. Such a behaviour, where a played stoned is immediately captured is called "suicide". Most rulesets forbid suicide behaviour.

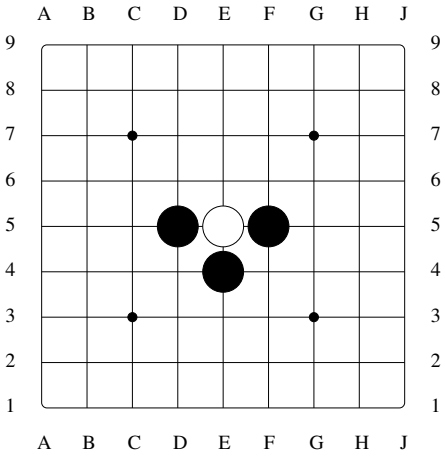


Figure 2: Black moves ...

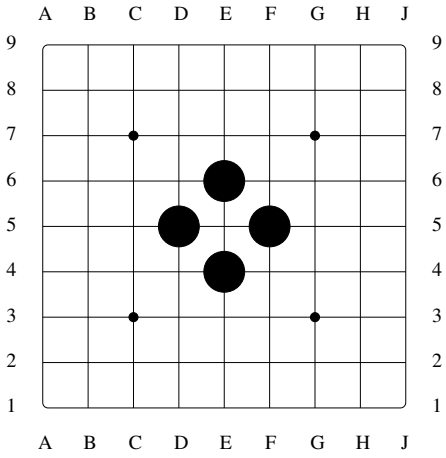


Figure 3: ... and captures White

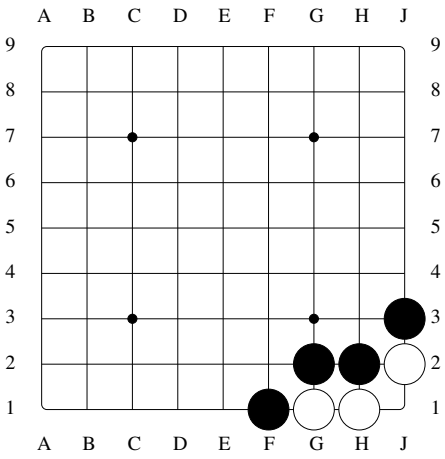


Figure 4: Black's turn

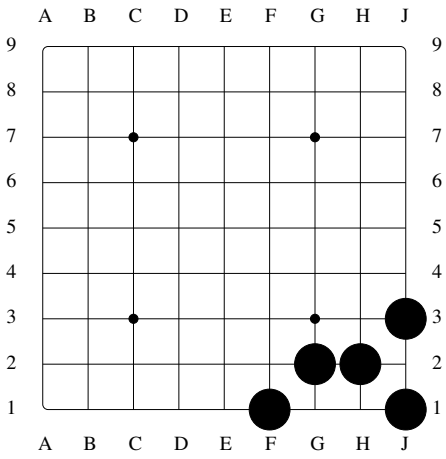


Figure 5: corner group captured

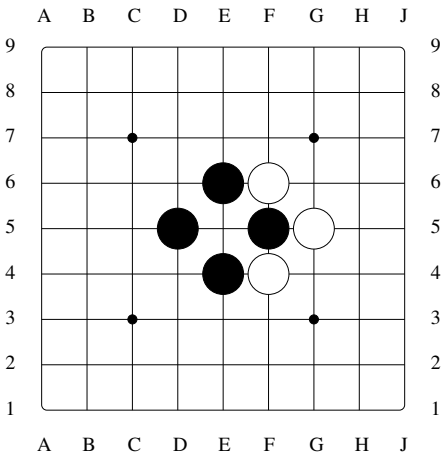


Figure 6: White's turn

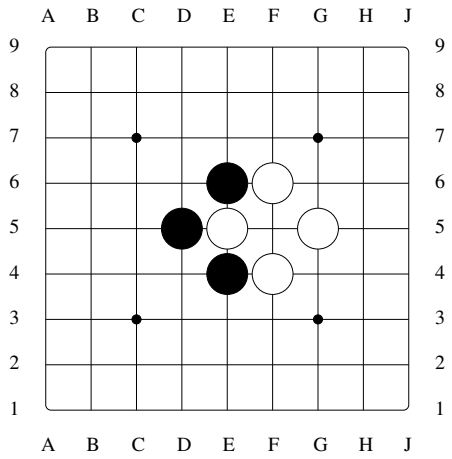


Figure 7: Black must not play F5

## 2.3 Ko

The Ko-rule states, that it is not allowed to repeat the previous board position. A typical situation for a Ko is shown in figure 6. White can play E5 in this board configuration. This is not a suicide, because White captures the black stone at F5 and so prevents getting captured itself. After this move Black could strike back at F5, which would lead to the board configuration before Black's last move and possibly to an infinite move sequence. That is why the Ko-rule forbids repeating the previous board position. Black cannot play F5 immediately, but if it plays a move somewhere else on the board it can later play F5 again to capture White's stone. The rulesets differ in handling repition of *any* previous board position (not just *the* previous one). Please see [6] in question.

## 2.4 The Aim of the Game

The purpose of Go is to conquer a greater part of the board than your opponent. Figure 8 shows a final board position (a position is final, if both players have passed). There are different rulesets for evaluating a game. The most popular ones are the Chinese and Japanese rules. In both rulesets dead stones have to be removed from the board after the end of a game. Dead stones are stones which cannot avoid being captured. In figure 8 the white stones at B7 and C7 are dead. The Chinese rules now use the area scoring method, which means to count all controlled intersections of a player. This can either be any empty intersection, which is completely surrounded by his stones or an intersection, on which he played a stone. In the example White gains 36 points and Black 45 points, so Black is the winner. The Japanese rules use the territory scoring method. Every player gets one point for every empty (after dead stone removal!) controlled intersection and must substract the number of stones captured from him. In the example White has 21 controlled empty intersections and Black has 27 of them. Black has one stone more than White on the board. This

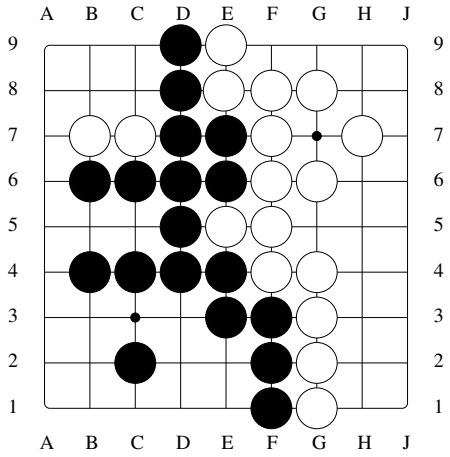


Figure 8: a final board position

means, that either both parties captured the same number of stones (if White passed first) or Black captured one stone more (assuming no one passed before the final two passes). In both cases Black is the winner.

Unlike in most other games the players can discuss about certain stones being dead or not after a finished game. The rulesets differ in resolving disputes. Please see [7] for Japanese rules, [8] for Chinese rules and [6] for a detailed comparision between popular rulesets. Generally different rulesets rarely lead to different results.

## 2.5 Komi and Handicap

Usually Black has a slight advantage, because it moves first. To compensate for this White receives additional points, called komi. The komi can varies between 5.5 and 7.5. Usually a fraction number is preferred, because this avoids draws. An interesting fact is, that komi is independant of boardsize.

A handicap is used to balance the game between differently skilled players. Black always takes the role of the weaker player in handicap games. The handicap allows Black to place a number of stones on the board before White can move. The greater the difference in skill the higher the handicap should be. Handicap stones are usually placed on the star points of a Go board. The star points on a 9x9 board are C3, C7, G3 and G7. They are usually marked on the board in some way to make orientation easier.

## 2.6 The Go Ranking System

$$\overbrace{\overbrace{30kyu \ldots 1kyu}^{\text{kyu rating}}}^{\text{kyu=student}} \quad \overbrace{\overbrace{1dan \ldots 7-9dan}^{\text{dan rating}} \quad \overbrace{1pro \ldots 9pro}^{\text{professional}}}^{\text{dan=master}}$$

Starting with the 16th century AD a ranking system for Go was created in Japan. New players get a kyu rank. 1 kyu is the highest kyu rank and a rank of about 30 kyu is the rank of a Go beginner. Competent amateur players receive a dan rating from 1 to 9 with 9 dan being the highest rank. 1 dan is slightly better than 1 kyu. In Europe ranks up to 7 dan are recognised, in Japan ranks up to 8 dan. The difference between these ranks is about one handicap stone. In China, Japan and Korea there is an additional ranking set for professional players, again reaching from 1 dan to 9 dan, with 9 dan being the highest rank. A pro 1 dan player has comparable strength to the strongest amateur players. The difference between professional ranks is about 1/3 handicap stone.

If a player beats his handicap often this is an indicator that his real strength is higher than his current rank, so he will climb one rank upward. In different groups, where players are not mixed, it can happen that different ranks are awarded for the same playing strength. This is one reason why players usually append "in this group" or "in this country" to their rank.

# 3 Common Problems in Go and Terminology

The aim of this section is to give the reader a feeling for the game and explain common terms of Go players. This section is not related to Computer Go, but builds up the basics for understanding the techniques explained in the following sections.

To make it easier to speak about Go, I introduce some simple definitions:

**Definition 3.1 (point)**
*An intersection of a horizontal and vertical line on a Go board is called a* point.

**Definition 3.2 (adjacent points)**
*Two points are* adjacent, *if they are neighbours on the same horizontal or vertical line.*

**Definition 3.3 (connected)**
*Two stones are* connected, *if they are on adjacent points and have the same color.*

**Definition 3.4 (block)**
*A* block *is a maximum set of connected stones.*

**Definition 3.5 (liberty)**
*A* liberty *is an empty point adjacent to a stone or a block of stones.*

As an example the stone C4 in figure 9 has three liberties and the block G9-G8-G7-H7-J7 has 8 liberties. Obviously a block without a liberty cannot exist, because it would have been captured. Maximizing the liberties of a block is an important concept in Go, since it makes capturing stones very hard for the enemy.

## 3.1 Life and Death

A key concept in Go is to classify a group of stones as alive, dead or unsettled.

**Definition 3.6 (dead)**
*A* dead *stone is a stone, that cannot avoid capture.*

Stones can be dead, even though they are still on a board. It is important to note, that in most cases it does not make sense to immediately capture dead stones (unless needed for strengthening own stones). If Chinese rules are used, they are removed from the board anyway and captures do not play any role. In the case of Japanese rules capturing makes only sense, if you capture more stones, than you must place stones on the board for capturing them.

How can a group avoid capturing? Maximizing the liberties of a group is one way. Another possibility is preventing the opponent from being able to cover all liberties of a group. How can this be done? To show this I introduce the term "eye". In figure 9 you can see, that the black stones in the right lower corner are completely enclosing an area (H2). White cannot cover this area immediately, but has to cover all other liberties of this group first, before it can play H2 to capture it. This leads to the next definition:

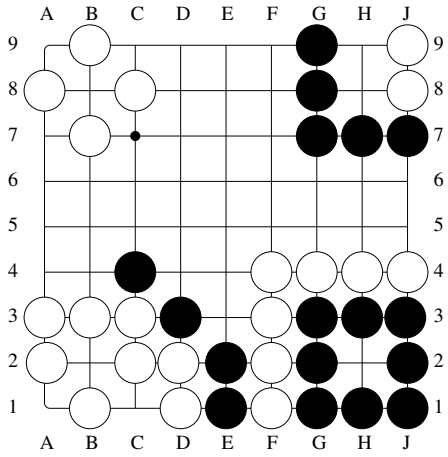


Figure 9: Life and Death

**Definition 3.7 (eye)**
*An enclosed area providing one sure liberty is called an* eye.

Thinking further it seems impossible for the enemy to capture a group of stones with two eyes. The white group in the lower left corner of figure has three eyes. There is no way Black can capture this group. We call such a group alive.

**Definition 3.8 (alive)**
*A group of stones is* alive, *if it cannot be captured by the opponent.*

However not every group with two eyes can avoid capture. The white group in the upper left corner of figure 9 can be captured by destroying the eye at B8 after playing C9, C7, D8 and B8. Such an eye is a false eye.

**Definition 3.9 (true eye)**
*A* true eye *is an eye, that the opponent can only close by destroying all blocks forming the eye or by forcing the player to close the eye.*

**Definition 3.10 (false eye)**
*A* false eye *is an eye, which is not a true eye.*

**Definition 3.11 (unsettled)**
*A group of stones, which is neither dead nor alive, is* unsettled.

## 3.2 Territory

Go has a very complex strategic component. Some good moves do not have immediate consequences, but play an important role much later in the game. Some simple examples for this fact are presented in this section. They are important for reasoning about the strength of Go computers versus humans.

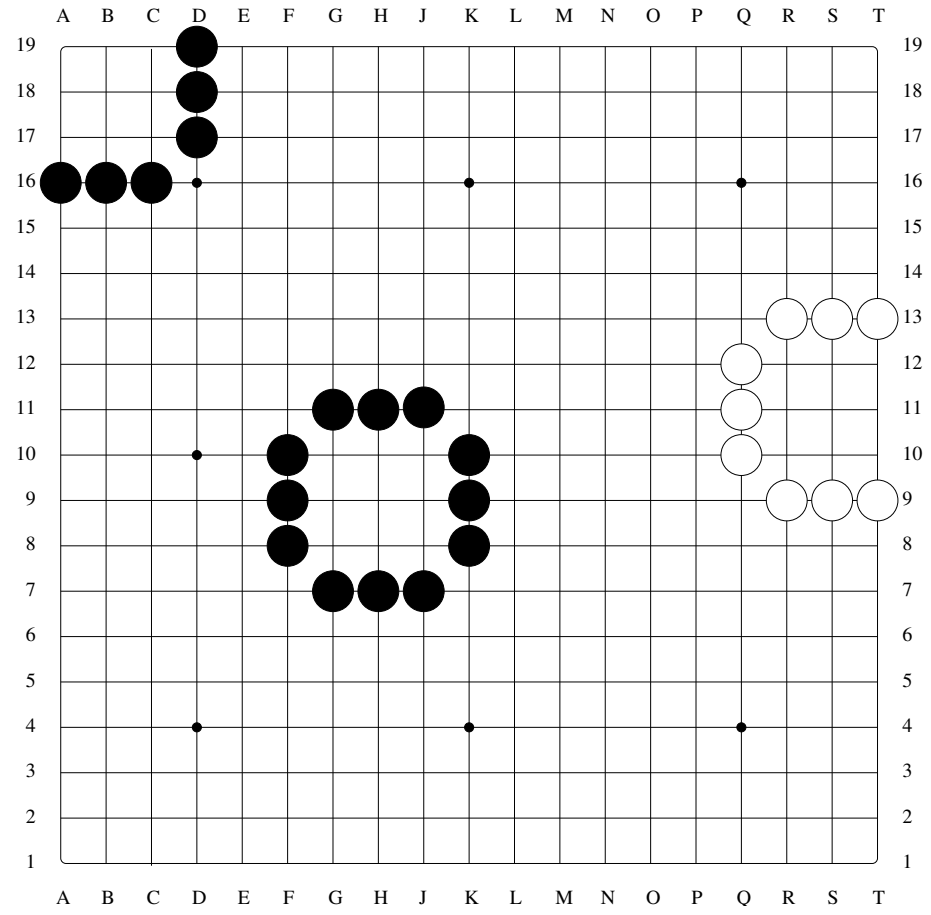


Figure 10: making territory on a 19x19 board

To discuss some of these moves the term territory is introduced:

**Definition 3.12 (territory)**
*An area surrounded and controlled by one player is called* territory.

The goal of Go is to claim more territory than the opponent. When the game starts one must try to surround and control as much territory as possible. It is obvious that a move at A1 does not give control over significant parts of the board, but where else on the board should one play? Figure 10 shows a 19x19 go board. This is the boardsize used in Go tournaments (Computer Go is often played on 9x9 as well as 19x19). It is more than four times as large as a 9x9 board and gives one a good idea of the problems unexperienced players as well as computers may have.

The figure shows how you can control territory by surrounding it. Counting the number of stones needed for surrounding a territory of nine points, we find out that only six stones are needed in the corner, nine stones are needed on the side and twelve stones in the middle of the board. Therefore a good strategic move at the beginning of the game will try to gain control over a corner. On a 19x19 board D4 (or by symmetry D16, Q4, Q16) is a very popular move for starting a game.

A good player needs a feeling how the "balance of power" is on the board. Placing too many stones in one corner can ensure control over it, but puts the player in a bad position elsewhere. However, being too greedy is not good either, because this allows the opponent to gain territory within the area you apparently controlled. Go is a game of balance. A move on the second line often does not gain enough. A move on the fifth or sixth line can be undermined by the opponent playing on the third line. Seemingly subtle differences can be very important later on. Understanding this is necessary to follow the explanations in the next section.

# 4 The State of the Art in Computer Go

After having laid out a solid foundation, this is the first section devoted entirely to Computer Go. As already stated briefly in the introduction, Go programs lag far behind their human opponents. One can go even further and say that the difference in playing strength between Go programs and humans is greater than in any other popular board game. Some games have been completely solved and in other games like chess programs have reached master level strength. Go programs are not anywhere near reaching this goal despite considerable research efforts and about 10 person years of effort spend for the top programs. This shows how hard it is to write strong programs. Go is believed to be the final frontier in computer games research.

Measuring the strength of programs is not an easy task, because computer play differs a lot from human play. Some of the best programs can play master level moves most of the time, especially in situations with limited possibilites and situations involving known patterns. However the full game performance is still below master level. Usually a program plays enough "bad" moves to lose a game. The strength of Go programs is currently estimated at about 15 to 5 kyu in rating system presented in section 2.6. Dr. Piet Hut makes it very clear in a New York Times article [4]: "If a reasonably intelligent person learned to play Go, in a few months he could beat all existing computer programs. You don't have to be a Kasparov.".

## 4.1 Why are Computers so bad at playing Go?

The most obvious reason why Go is a difficult game for computers is the large number of possible moves in each turn. David Fotland, author of a powerful Go program, states "Brute-force searching is completely and utterly worthless for Go." [4].

A move made by one of the players is called a ply. In a chess game there are about 35 legal moves in a turn. Fast and effective chess programs can look twelve or more plies ahead. In Go about 200 legal moves have to be considered each turn, so the branching factor is much higher. Although the large search space is often believed to be the main reason for the complexity of Go for computers, this is not completely true. Even on a 9x9 board with much less possibilites (comparable to chess) Go programs are still very weak and not much stronger than on a 19x19 board (see [3, section 2] for a more detailed analysis of this fact).

The reason for this is the difficulty of static move evaluation. In most board games pieces on the board can be assigned a certain value and a full evaluation is relatively easy. In Go this a much harder task ([3, section 4]). A full evaluation involves solving a lot of subproblems. Groups (own and opponent ones) have to be probed, if they are dead, alive (eye status!), unsettled or something in between. There is no easy way to do this. Often locally calculating some plies ahead is required, if a group does not follow a known pattern. The local balance on a Go board can change significantly with each move unless a program makes very defensive moves securing certain groups (which very likely leads to

problems elsewhere).

As a conclusion both principles, looking ahead and evaluating a position, face enourmous barriers in computer Go and require more processing power than available. Some of the techniques used in real Go programs are covered in section 5.

## 4.2 Why are humans so good at playing Go?

This section describes an often forgotten point of view. To find out why the difference in playing strength is so significant it is not sufficient to analyse why computers are bad at playing Go, but there may also be reasons why humans are especially good at it.

Human players are able to recognize the impact of subtle differences in Go positions, which can have a decisive effect later. Contrary to computers they can judge very early, if a group can be captured (or respectively avoid capture). For a computer this is often impossible, because of the wide range of possible outcomes. Expert players develop a sense for the balance of power on the board. They can sense life and death and therefore do not spend more moves than necessary on an already won or lost fight.

Recognizing patters and drawing intuitive conclusions is another strength of humans. They can feel, if some stones can slowly form a group and become powerful. Humans have a better intuition for the strategic component of a game. Strategy in Go is much more important than e.g. in chess. A good strategy means to make moves with a longtime positive effect opposed to winning a local fight. According to an article of the IntelligentGo.org foundation [9] Dr. I. J. Good wrote in 1965(!) in the "New Scientist":

> "The experienced player will often be unable to explain convincingly to a beginner why one move is better than another. A move might be regarded as good because it looks influential, or combines attack and defence, or preserves the initiative, or because if we had not played at that vertex the opponent would have done so; or it might be regarded as bad because it was too bold or too timid, or too close to the enemy or too far away. If these and other qualitative judgments could be expressed in precise quantitative terms, then good strategy could be programmed for a computer, but hardly any progress has been made in this direction."

Another point is that humans can "read" a Go game very good, because it is mostly static except the capturing of stones. The stones themselves do not move on the Go board unlike many other games. This visual nature of the game fits human perception very well and cannot be modelled in a simple way in a computer program.

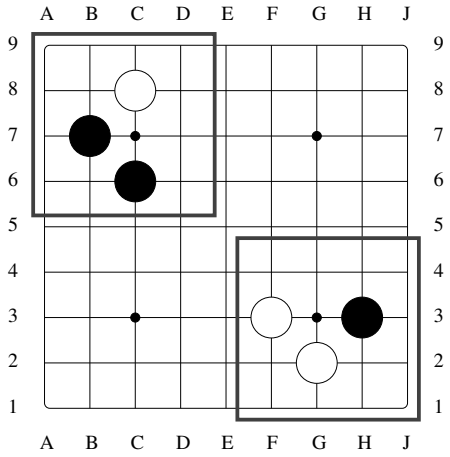


Figure 11: two matches of a corner pattern

# 5 Techniques used in Computer Go

This section discusses technical approaches currently used in the most popular Go programs. Of course there is a variety of different apporaches, but due to the introductive nature of this document only a few of them are presented here.

## 5.1 Patterns

A pattern consists of a number of points and the state each point can have (white, black or empty). Figure 11 shows an example pattern. Usually corner, edge and center patterns are distinguished, because the edges often have an effect on the pattern.

A pattern is two-dimensional and must be compared with a full board position. It can have eigth different instances by rotating and flipping horizontal or vertical. Additionally colors can be interchanged. In figure 11 both matches are instances of the same pattern. One can imagine that it needs a lot of processing power to compare each pattern against each possible location of the pattern on the board. Sophisticated filtering techniques help to reduce the needed computations. A simple time-saving technique is to save pattern matches and mismatches. During the next moves all those matches (and mismatches) can be kept whose situation has not changed, which means their prerequisites are still fulfilled (respectively not fulfilled).

Patterns are a way to represent Go knowledge and are present in almost every program. The patterns are mostly generated by hand. A program can contain thousands of patterns. There exist approaches for automatic pattern generation, but at the point of the time of writing this document most top level programs use hand generated patterns. For this reason automatic pattern generation is not covered here.

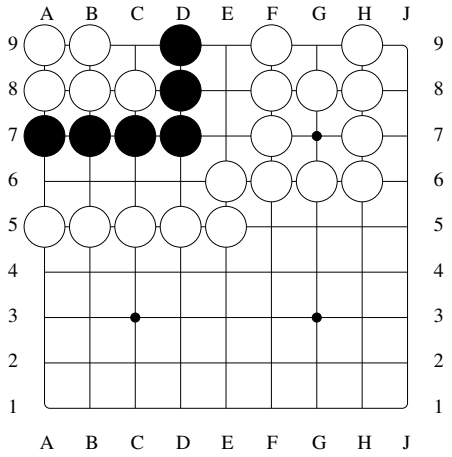


Figure 12: tactical and strategic safety

In addition to the stone positions a pattern contains context information. This includes constraints, which must be satisfied for a pattern match. These can be liberty count and safety of stones on the boundary of a pattern. If a board position satisfies a pattern, the saved knowledge of the pattern can be used. This can be information about strong moves, position evaluation or possible connections between groups. The data provided by the pattern is evaluated by the main routine. If a pattern contains a move sequence forming an additional eye of a group this can be very useful, but unimportant if the group is already alive. Some programs give bonuses to moves suggested by a pattern [1, section 3.1.3] , because they are usually created by expert players and less error prone.

Of course good patterns can make a Go program stronger and save computation time, because it is not necessary to search some plies forward once a pattern has matched. On the other hand there can be hundreds of patterns matching in a single board position [1, section 3.1.2]. Algorithms for dealing with overlapping or even contradicting patterns are needed to resolve issues.

## 5.2 Knowledge Representation

In Go there are several layers of knowledge representation. The most basic layer is the status (white, black or empty) of each single point of the board. Effective representation of this layer is not explained here. However the next parts cover some of the more interesting representations of knowledge about a Go board.

### 5.2.1 Blocks

Blocks (see Definition 3.4) are still a very basic layer of representing a Go board. An important property for a block is the number of liberties. A standard heuristic is that five or more liberties mean local safety [1, section 3.2.4]. If a block

has fewer liberties a forward search can determine if it is safe. Strategic safety is more important than tactical safety. Fig 12 is an example by Mueller [1]. The black block is tactically safe and has lots of liberties, but it is strategically dead, because it cannot make two eyes. (White can play B8 after Black attacks at C9.) The white block in the upper left corner is tactically dead, because it cannot avoid capture, but strategically it is part of a territory belonging to White. For this reason Blocks are usually connected to a higher level structure to get the strategical status of groups and territory. This way a program does not unnecessarily waste moves in a strategically dead area.

### 5.2.2 Territory

It is essential for the strategy of a Go program to recognize wether it controls a part of the board or not. One reason is, that the player with more territory (see Definition 3.12) wins the game. Another reason is that a sorrounded territory is safe and can be used to connect to other groups, which makes them alive. Finally this knowledge can be used to effectively surround opponent stones.

One possibility to find territory is to search in areas of high influence and then test wether the opponent can possibly destroy the influence or not. This is especially important for potential territory, which can be made safe by a low number of moves. Another method is to look for connecting blocks (either directly connected or by eyes). A third method uses connectivity as criterion. A point is territory if an opponent stone on this point cannot connect to any stone outside, which is alive, and cannot live by itself.

In order to avoid unnecessary moves inside own enclosed areas, the program can compute, wether it is possible in all cases to create two eyes and make the area alive or not. This can also be used to test, if it is possible to make good moves in areas with high influence by the opponent to stop him from securing his area by making eyes.

### 5.2.3 Groups

Groups are a high level concept in Go. They are loosely defined as a set of related stones with a certain minimum influence. While groups are often easily picked out by humans they are a greater problem for computers [3]. Not every Go program defines groups [1, section 3.2.8] as a seperate concept, but they can be a very important means for generating good moves. Once a group and its area of influence is roughly known by a program it can intelligently play stones to extend own groups or invade opponent groups. At the beginning of a game this is essential. Even after only a few moves, long before you can think about securing areas, you can divide the board in areas influenced by one of the players. Making a good strategic move at this point of time can greatly enhance playing strength. According to Mueller [1] today's implementations of groups are reasonable, but still much weaker than humans. Being able to recognize groups is a prerequisite for connecting them. Connecting two weak groups and therefore forming two eyes can decide a game.

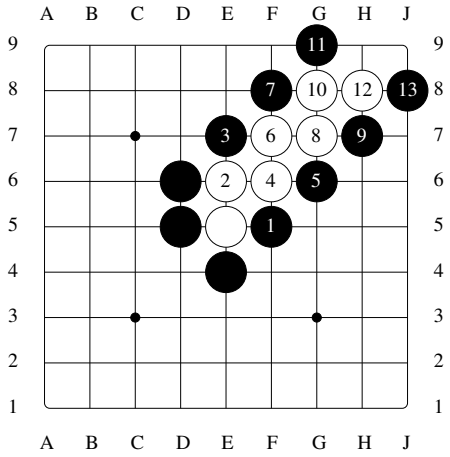


Figure 13: the ladder

## 5.3 Game Tree Search

This section covers examples of the use of search in Go programs. As already mentioned in the section "The State of the Art in Computer Go" a brute force full board search for more than a few plies is technically impossible and not used in the top programs [3, section 6.2]. However there is a great variety of restricted search algorithms used in Go programs. These searches are limited to certain goals. By only searching for one (or a few) goals the number of possible moves is relatively low and there is no complex evaluation function needed. These were the major drawbacks of full board search.

### 5.3.1 Single-Goal Search

A goal is often limited to a single strucure e.g. a block or a group. The most simple example is a ladder. A ladder is a sequence of capturing threats. The defender moves are always forced replies. An example can be seen in figure 13. The program may need to compute up to 60 plies in order to be able to say if a ladder is succesful. This is only possible with a specialized ladder search. A ladder is just one example, other common goals include "capture an opponent block", "determine if a region is alive/dead", "connect/cut two blocks", "determine the eye status", "local game score" or "safety of a territory". Usually not all of these goals are implemented in Go programs, but some of them.

### 5.3.2 Multiple-Goal Search

Multiple-goal search works like single-goal search except that AND and OR combinations of single-search goals are allowed. A goal could be "capture block 1 or block 2" (double threat) or "make eyes or attack". It is not possible to probe for every combination of goals, so heuristics must be used to examine,

which combination of goals can be succesful in a certain situation. Multiple-goal search is only rudimentary implemented in recent Go programs.

### 5.3.3 Full Board Search

Full board search in Go is rarely a brute-force search, because of the high branching factor. Several method exists to pick only a few promising moves. Usually an extensive static analysis is used to limit the number of moves. Shortcuts can be introduced to play urgent moves immediately. Currently there is no agreement how full board search should be performed. It is sometimes used for finding weak enemy groups. Because full board search can only be very narrow and shallow it is not a key to success, unless the processing power of the used computers increases significantly.

# 6 Outlook

While the last sections hopefully informed the reader well about the current state of computer Go, this section gives a brief overview over possible future directions of research (ideas taken from [1, 9]) in computer Go research.

**More Processing Power → stronger Program** So far no one could prove, that more processing power automatically leads to better programs. Doing so could help constructing powerful programs.

**Sure Win Programs for high Handicaps** One approach could be to write a program, which wins a game with a certain number of handicap stones for sure (by demonstration or even proof). After that one could try to reduce the number of handicap stones.

**Go on small Boards** On small Go boards games can be completely solved. The largest square board size solved is 4x4 (see [2, section 3.5.]). An interesting idea is to construct perfectly playing programs for small board sizes and the increase the size.

**Special Hardware** Implementing some parts of a Go program in hardware may make them better. Especially full board search could be greatly enhanced. This was done succesfully in chess. However in Go this alone probably won't lead to a master level program, but it leaves some potential for future programs.

**Neural Networks** Neural networks with temporal difference learning [10] are used by NeuroGo [11] as a part of the program. There has been no breakthrough achieved by these methods yet, although they enhance playing strength.

**Combinational Game Theory** Combinational game theorie splits a game in subgames and treats them independly. For every subgame the best move is chosen and out of these an overall best move is computed. This could work in Go endgames. Further information can be found at [3, section 8].

While most other board games can be played very well by computers Go remains a big challenge for researchers. The simple rules of Go unfold to an enourmous variety and complexity. Writing a competitive Go program may teach us a lot about human thought.

## References

[1] Martin Müller. Computer go. *Artificial Intelligence*, 134(1-2):145–179, 2002.

[2] H. Jaap van den Herik, Jos W.H.M. Uiterwijk and Jan van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.

[3] Bruno Bouzy and Tristan Cazenave. Computer go: an ai oriented survey. *Artificial Intelligence*, 132(1):39–103, October 2001.

[4] George Johnson. To test a powerful computer, play an ancient game. *New York Times*, July 1997.

[5] www.wikipedia.org. articles about "go", "go rules", "computer go" and "go strategy".

[6] British Go Association. Comparision of go rules, September 2003. http://www.britgo.org/rules/compare.html.

[7] Japanese rules. http://www-2.cs.cmu.edu/~wjh/go/rules/Japanese.html, April 1989.

[8] Chinese rules. http://www-2.cs.cmu.edu/~wjh/go/rules/Chinese.html, 1992.

[9] The Intelligent Go Foundation. Overview of computer go. http://www.intelligentgo.org/en/computer-go/overview.html.

[10] Nicol N. Schraudolph, Peter Dayan and Terrence J. Sejnowski. Learning to evaluate go positions via temporal difference methods. Technical report, IDSIA, 2000.

[11] Markus Enzenberger. NeuroGo. http://www.markus-enzenberger.de/neurogo.html. A Go program, which is partly based on neural networks.

[12] Gnu Go. http://www.gnu.org/software/gnugo/gnugo.html. A strong and succesful open source go playing program.