Computer Go

Jens Lehmann

Dresden University of Technology

July 1, 2004
Introduction
Introduction to Go

- popular board game
- invented in China before 2000BC
- brought to Japan 7th century AD
- simple rules, but complex game
Popularity

- most popular in Eastern Asia (Korea, Japan, China)
- professional players, who play the game fulltime
- one of four arts in ancient China along with music, poetry and painting
- popularity in the western world increases
Computer Go

- a lot of research effort
- only moderate success
- all programs can be beaten by average club players

Go is the biggest challenge for AI game programmers today.
Rules of Go
Basic Setup

- played on square board with horizontal and vertical lines
- common sizes 9x9, 13x13 and 19x19
- two players: Black and White (playing alternatingly)
- only one kind of stone
- stones played on intersections of the board
- player must either put a stone on the board or pass
- two consecutive passes end the game
Figure: empty 9x9 go board
Capturing Stones

- stones cannot be moved, but they can be captured
- to capture a group of stones it must be completely surrounded
- being surrounded = there are opponent stones on every adjacent horizontal or vertical intersections
- suicide = a move, which captures one’s own stone
Figure: Black moves ...
Figure: ... and captures White
Figure: Black’s turn
Figure: corner group captured
Ko-rule

- Ko-rule: it is not allowed to repeat the previous board position
- avoids infinite move sequences
- there are different rulesets for Go, some only forbid repeating the *previous*, others forbid repeating *any* board position
Figure: White’s turn
Figure: Black must not play F5
Aim of the Game

- evaluating a game depends on the used ruleset, but they have some things in common
- goal: conquer a greater part of the board than your opponent
- dead stones are removed from the board after the end of a game
- dead stones: stones, which cannot avoid capture
- players can discuss about whether a stone is dead or not
Game Evaluation using Chinese Rules

- remove dead stones
- area scoring method: count all controlled intersections of a player
- controlled intersection: empty, but completely surrounded intersection or intersection with an own stone
Game Evaluation using Japanese Rules

- remove dead stones
- territory scoring method: one point for each empty controlled intersection
- subtract the stones the opponent has captured
Figure: a final board position
Figure: dead stones removed
Komi

- Black has an advantage, because it moves first
- to compensate this White receives additional points (komi)
- komi varies between 5.5 and 7.5 (independent of board size)
- fraction number avoids draws
Handicap

- handicap balances game between differently skilled players
- Black is always the weaker player in handicap games
- handicap allows Black to put a certain number of stones on the board before White can move
Go ranking system

- kyu rating:
  - 30 kyu...1 kyu
  - kyu = student

- dan rating:
  - 1 dan...7 – 9 dan
  - dan = master

- professional:
  - 1 pro...9 pro

- created in the 16th century in Japan
- new players get a kyu rank (lower is better)
- competent players get dan rating (higher is better)
- professional players are on an additional dan ranking scale (higher is better)
- difference of one kyu or dan rank: one handicap stone
- difference of one pro dan rank: \( \frac{1}{3} \) handicap stone
Common Problems and Terminology
Definition: Point, adjacent Point

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

Definition (adjacent points)
Two points are adjacent, if they are neighbours on the same horizontal or vertical line.
Definition: connected, Block, Liberty

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

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

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

Jens Lehmann

Introduction

Rules of Go
- Basic Setup
- Capturing Stones
- Ko
- The Aim of the Game
- Komi and Handicap
- The Go Ranking System

Common Problems and Terminology
- Life and Death
- Territory

State of the Art in Computer Go
- Why do Computers play so badly?
- Why do Humans play so good?

Techniques used in Computer Go
- Patterns
- Knowledge Representation
- Game Tree Search

Outlook
Life and Death

Life and Death: Classifying stones as dead, alive or unsettled is a key concept of Go.

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

- capturing dead stones is often a bad move
- Chinese rules: dead stones are removed and captures do not count
- Japanese rules: capturing makes only sense, if more stones are captured than needed for capturing
Eyes

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

- an eye can be build by surrounding a small area e.g. a single point
- the opponent cannot fill this area immediately, because this would be a suicide
- opponent must first cover all other liberties of a group, before it can close eye and capture group → this is a sure liberty
Definition (alive)

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

- if a group has two (true) eyes, it cannot be captured by the opponent
- opponent cannot fill both sure liberties at the same time
- some eyes can be destroyed and therefore do not make a group of stones alive \(\rightarrow\) false eyes
false and true eyes

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

Definition (unsettled)
A group of stones, which is neither dead nor alive, is unsettled.
Figure: Life and Death
Go has complex strategic component
some moves do not have immediate consequence, but play a decisive role later in the game
important for reasoning about the strength of Go programs vs. humans
example: territory
Territory

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

- goal of Go: claim more territory than opponent
- Which moves gain most territory?
Territory

For surrounding nine points . . .

► . . . twelve stones are needed in the center
► . . . nine stones are needed on the side
► . . . six stones are needed in the corner

At the beginning try to gain control over the corners.
Go strategy

- good player needs feeling for "balance of power" on board
- many stones in corner ensure control over it, but lose control elsewhere
- being too greedy is dangerous, because the opponent can undermine your territory

Conclusion: Seemingly subtle differences can be important later on.
State of the Art in Computer Go
State of the Art in Computer Go

- programs lag far behind their human opponents
- difference in playing strength probably greater than in any other popular board game
- programs are unlikely to reach master level strength anytime soon
- final frontier in computer games research
"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.” (Dr. Piet Hut)

- about 10 person years of effort spend for top-level Go programs
- best programs play master level moves often
- full game performance is still very low
- difficult to evaluate strength, because of different playing style
- strength estimated at 15 to 5 kyu
"Brute-force searching is completely and utterly worthless for Go." (David Fotland)

- new term: ply = a move by one of the players, "half" move
- chess: about 35 legal moves per turn
- 9x9 Go: similar to chess
- 19x19 Go: about 200 legal moves per turn
- computing several plies ahead works very well for chess, but not for 19x19 Go

But: Even on a 9x9 Go board computers are not much stronger than on a 19x19 board.
Position Evaluation

- a full position evaluation in Go is a very hard task
- leads to a lot of subproblems
- group status (dead, alive or unsettled) cannot easily be tested → often requires locally calculating several plies ahead
- local balance on a board can change significantly with each move
Why are Computers bad at playing Go?

- high branching factor makes looking ahead very expensive
- when looking ahead not even a single position can be easily and accurately evaluated
- in Go other techniques must be used to create good programs (section 5)
Why do Humans play so good?

- often forgotten point of view
- humans quickly recognize subtle differences in Go positions
- can judge early, if a group can be captured or avoid capture
- can sense life and death → do not waste moves in already won or lost fights
Why do Humans play so good?

- humans recognize abstract patterns and draw intuitive conclusions
- feeling, which stones can form a group, and become powerful
- in general good feeling for the strategic component of Go
- strategy is very important in Go
- a good long time strategy can be better than winning a local fight
“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.” (Dr. I. J. Good, 1965)
Why do Humans play so good?

- humans can "read" Go games very good
- game is mostly static (except captures), stones do not move
- fits human perception $\rightarrow$ humans can look ahead and evaluate positions very good
Techniques used in Computer Go
What are patterns?

Parts of a pattern:

<table>
<thead>
<tr>
<th>map</th>
<th>a set of points and their state (black, white or empty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>corner, edge or side pattern</td>
</tr>
<tr>
<td>context</td>
<td>constraints, which must be satisfied for a pattern match (liberty count, safety of stones on the boundary of a pattern)</td>
</tr>
<tr>
<td>information</td>
<td>knowledge, which can be used in case of a pattern match</td>
</tr>
</tbody>
</table>
pattern matching

- two-dimensional pattern must be compared with full board position
- a pattern can have 16 different instances by rotating, flipping and changing colors
- sophisticated filtering techniques used to reduce needed computations
- usually hundreds of patterns match in a board position
- simple technique:
  - save pattern matches
  - during next turn keep all pattern whose situation (map and constraints) has not changed
Figure: two matches of a corner pattern
Usage of a Pattern

- patterns are mostly generated by experts by hand, although approaches for automatic pattern generation exist
- contains information about:
  - strong or bad moves (concerning goals)
  - position evaluation
  - possible connections between groups
- data is evaluated by main routine, because e.g. making an eye can make a group alive or be unimportant, if the group is already alive
Knowledge Representation

- Go has several layers of representation
- simplest representation: status of every field on the board (black, white or empty)
- representations explained here: blocks, territory and groups
- other representations are: connectors/dividors, chains and more
Blocks

- we defined blocks as a maximum set of connected stones
- important property of a block: number of liberties
- heuristic: five or more liberties means local safety
- less than five liberties: use local forward search to determine, if it is save
- tactical and strategical status of a block is not necessarily the same → blocks need to be connected to a higher level structure
Figure: tactical and strategic safety
Territory

- we defined territory as an area surrounded and controlled by a player
- Go programs must recognize whether it controls a part of the board
  - player with more territory wins
  - can be used to connect to other groups
Territory

- finding territory
  - search for areas of high influence and test, if the opponent can destroy the influence or not (if yes: test, if it can be made safe by a move)
  - connectivity method: 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
- if potential territory is found the program can compute, if it is possible to create two eyes (or if a stone can be played in opponent territory to avoid the creation of two eyes)
Groups

- high level concept
- loosely defined as set of stones with a certain minimum influence
- groups may become territory
- easily picked out by humans, difficult for computers
- groups not implemented in every program
- today’s implementations are reasonable, but not excellent
The Role of Groups

- if a program is able to find groups, it can intelligently extend them (or destroy opponent groups)
- essential at the beginning of a game: long before territories can be secured the board can divided in areas of influence
- necessary for good strategic moves
- recognizing groups is a prerequisite for connecting them; often results in two eyes and a big alive group
Game Tree Search

- brute force full board search works only a few plies ahead
- most top programs rarely use brute force search
- search algorithms restricted to certain goals and mostly local → reduces branching factor, needs no complex evaluation function
Single Goal Search

- goal often limited to a single structure e.g. a block or group
- simple example: ladder (a sequence of capturing threats with forced replies of the defender)
- searching only for ladders (a single goal) programs can compute a lot of plies ahead and determine, if a ladder is successful
- other goals:
  - capture opponent block
  - determine if group is alive/dead
  - connect/cut two blocks
  - eye status
  - local game score
Figure: the ladder
Multiple Goal Search

- like single goal search, but with AND and OR combinations of goals (usually both single goals fail)
- examples:
  - capture block1 or block2
  - make eyes or attack
- heuristics used for determining potential successful combination of goals
- only rudimentary implemented
Full Board Search

- full board search is not brute force search, because of the high number of legal moves
- only a few promising moves are picked (usually by statically analysing the board)
- still only narrow and shallow
- sometimes used for finding weak enemy groups
- no agreement on full board search
Outlook
Outlook

Possible future directions of research:

- more processing power → stronger program
- neural networks
- combinational game theory
- sure win for high handicaps
- Go on small boards
- special hardware
For further reading I

**Martin Mueller.**
Computer go.

**H. Jaap van den Herik, Jos W.H.M. Uiterwijk and Jan van Rijswijck.**
Games solved: Now and in the future.
For further reading II

