

Monte-Carlo Go Developments

by B. Bouzy and B. Helmstetter

Tsan-sheng Hsu

徐讚昇

tshsu@iis.sinica.edu.tw

<http://www.iis.sinica.edu.tw/~tshsu>

Abstract

- Introducing the original ideas of using Monte-Carlo simulation in computer Go.
- Adding new ideas to pure Monte-Carlo approach for computer Go.
 - Progressive pruning
 - All moves as first heuristic
 - Temperature
 - Simulated annealing
 - Depth-2 tree search
- Conclusion:
 - With the ever-increasing power of computers, we can add more knowledge to the Monte-Carlo approach to get a reasonable solution for computer games.

Basics of Go

- **intersection**: a cell where a stone can be placed or is placed.
- two intersections are **connected**: they are either adjacent vertically or horizontally
- **string**: a connected, i.e., vertically or horizontally, set of stones of one color
- **liberty**: the number of connected empty intersections
- **eye**:
 - Exact definition: very difficult to be understood and implemented.
 - Approximated definition:
 - ▷ *An empty intersection surrounded by stones of one color with two liberties or more.*
 - ▷ *An empty intersection surrounded by stones belonging to the same string.*

Basics of Monte-Carlo simulation

■ Algorithm:

- Play a large number of almost random games from a position to the end, and score them.
- Evaluate a move by computing the average of the scores of the random games in which it had played.

■ Details:

- No filling of the eyes when doing a random game.
 - ▷ *The only domain-dependent knowledge used in the original version of GOBBLE in 1993.*
- Scores: the difference of the the total numbers of stones for the two sides

Results

- **Original version: GOBBLE 1993.**
 - Score is not good compared to other Go programs.
- **Enhanced versions**
 - Adding more knowledge.
 - Adding more techniques.
 - ▷ *Much more than what is discussed in this paper.*
 - MoGo won 19x19 version at 2007.

Move evaluation and ordering

■ Evaluation of the moves:

- Moves were evaluated according to the average score of the games in which they were played, not only at the beginning but at every stage of the games provided that it was the first time one player had played at the intersection.
- Moves are considered independent of positions.

■ Selection of the moves:

- Moves are ordered according to the current evaluation values.
- Simulating annealing was used to control the probability that a move could be played out of order.
- The amount of randomness put in the games was controlled by the **temperature**.
 - ▷ *The temperature was set high in the beginning, and then gradually decreased.*
 - ▷ *Simulating annealing is not required, but was used in the original 1993 version.*

Progressive pruning I

- **Information about a move:**
 - Each move has a mean value m and a standard deviation σ .
 - ▷ *Left expected outcome $m_l = m - \sigma \cdot r_d$.*
 - ▷ *Right expected outcome $m_r = m + \sigma \cdot r_d$.*
 - ▷ *r_d is a ratio fixed up by practical experiments.*
 - A move M_1 is statistically inferior to another move M_2 if $M_1.m_r < M_2.m_l$.
 - Two moves M_1 and M_2 are statistically equal if $M_1.\sigma < \sigma_e$ and $M_2.\sigma < \sigma_e$ and no move is statistically inferior to the other.
 - ▷ *σ_e is called standard deviation for equality.*
 - ▷ *Its value is determined by experiments.*
- **After a minimal number of random games (100 per move), a move is pruned as soon as it is statistically inferior to another move.**
 - This process is stopped when
 - ▷ *there is only one move left, or*
 - ▷ *the moves left are statistically equal, or*
 - ▷ *a maximal threshold of iterations is reached.*
 - Current threshold is 10000 multiplied by the number of legal moves.

Progressive pruning II

■ Two different pruning rules.

- **Hard**: a pruned move cannot be a candidate later on.
- **Soft**: a move pruned at a given time can be a candidate later on if its value increases.

■ Selection of r_d .

- The greater r_d is,
 - ▷ *the less pruned the moves are;*
 - ▷ *the better the algorithm performs;*
 - ▷ *the slower is plays.*

- Results:

r_d	1	2	4	8
score	0	+ 5.6	+ 7.3	+9.0
time	10'	35'	90'	150'

■ Selection of σ_e .

- The smaller σ_e is,
 - ▷ *the fewer equalities there are;*
 - ▷ *the better the algorithm performs;*
 - ▷ *the slower is plays.*

- Results:

σ_e	0.2	0.5	1
score	0	-0.7	-6.7
time	10'	9'	7'

Progressive pruning III

■ Conclusion:

- r_d plays an important role in the move pruning process.
- σ_e is less sensitive.

All-moves-as-first heuristic I

- **How to perform statistics for a completed random game?**
 - Basic idea: its score is used for the first move of the game only.
 - All-moves-as-first: its score is used for all moves played in the game as if they were the first to be played.
- **Advantage:**
 - All-moves-as-first helps speeding up the experiments.
- **Drawbacks:**
 - The evaluation of a move from a random game in which it was played at a late stage is less reliable than when it is played at an early stage.
 - Recapturing.
 - ▷ *Order of moves is important for certain games;*
 - ▷ *Modification: if several moves are played at the same place because of captures, modify the statistics only for the player who played first.*
 - Some move is good only for one player.
 - ▷ *It does not evaluate the value of an intersection for the player to move, but rather the difference between the values of the intersections when it is played by one player or the other.*

All-moves-as-first heuristic II

■ Results:

- Basic idea is very slow: 2 hours vs 5 minutes.
- Relative scores between different heuristics.

All-moves	basic idea	PP
0	+13.0	+ 4.0

- Number of random games N : relative scores with different N .

N	1000	10000	100000
scores	-12.7	0	+3.2

▷ *Finally settle for 10000.*

Temperature

- **Constant temperature:** consider all the legal moves and play one of them with a probability proportional to $\exp(K \cdot v)$, where
 - K is the temperature, and
 - v is the current value.
 - Results:

K	0	2	5	10	20
score	-8.1	0	+2.6	-4.9	-11.3
- **Simulated annealing:**
 - increases K from 0 to 5 over time does not enhance the performance.

Depth- i enhancement

- **Algorithm:**
 - Enumerate all possible positions from the root after i moves are made.
 - For each position, use Monte-Carlo simulation to get an average score.
 - Use a minimax formula to compute the best move from the average scores on the leaves.
- **Current result: depth-2 is worse than depth-1 due to oscillating behaviors normally observed in iterative deepening.**
 - Depth-1 overestimates the root's value.
 - Depth-2 underestimates the root's value.
 - It is computational difficult to get depth- i results for $i > 2$.

Putting everything together

- **Two versions:**
 - Depth =1, $r_d = 1$, $\sigma_e = 0.2$ with PP, and basic idea.
 - $K = 2$, no PP, and all-moves-as-first.
- **Still worse than GnuGo, a Go program with lots of domain knowledge in 2004 by more than 30 points.**
- **Conclusion:**
 - Add tactical search: for example ladders.
 - As the computer goes faster, more domain knowledge can be used.
 - Exploring the locality of Go with statistics.

Comments

- The “reliability” of a Monte-Carlo simulation depends on the number of trials it performs.
 - The rate of convergence is important.
 - Do enough number of trials, but not too much for the sake of saving computing time.
- Adding more knowledge can slow down each simulation trial.
 - There should be a tradeoff between the amount of knowledge added and the number of trials performed.
 - Similar situation in searching based approach:
 - ▷ *How much time should one spent on computing the evaluating function for the leaf nodes?*
 - ▷ *How much time should one spent on searching deeper?*
- Knowledge, or patterns, about Go can be computed off-lined by Monte-Carlo methods.

References and further readings

- B. Bruegmann. Monte Carlo Go. unpublished manuscript, 1993.
- * B. Bouzy and B. Helmstetter. Monte-Carlo Go developments. In H. Jaap van den Herik, Hiroyuki Iida, and Ernst A. Heinz, editors, *Advances in Computer Games, Many Games, Many Challenges, 10th International Conference, ACG 2003, Graz, Austria, November 24-27, 2003, Revised Papers*, volume 263 of *IFIP*, pages 159–174. Kluwer, 2004.
- Haruhiro Yoshimoto, Kazuki Yoshizoe, Tomoyuki Kaneko, Akihiro Kishimoto, and Kenjiro Taura. Monte Carlo Go has a way to go. In *AAAI*, 2006.