Developments on Monte Carlo Go

Bruno Bouzy and Bernard Helmstetter

Advances in Computer Games 10, pp. 159–174

Presented by Markus Enzenberger

Computer Go Seminar, University of Alberta, April 2004

Overview

- Present a Monte Carlo approach simpler than **[Bruegmann, 1993]** based on **[Abramson, 1990]**
- Study enhancements like Progressive Pruning and All Moves as First heuristic
- Experimental comparison by playing test games on 9×9

Related Work

Abramson [1990]

- Expected-outcome model
- Heuristic is expected value given random play
- Domain-independent, efficiently calculable

Bruegmann [1993]

- Simulated Annealing
- Optimize priority of playing a move
- Decrease randomness over time

Basic Idea

- Based on Abramson [1990]
- Play a number of completely random games and evaluate them
- Choose a move by 1-ply search, maximize expected outcome
- Only domain-dependent knowledge is definition of an eye

Olga and Oleg

Two implementations were used in the experiments.

They used different definitions of an eye.

Both definitions are fast to compute and wrong in some cases.

Olga

Empty intersection surrounded by stones of one color with two liberties or more

less restrictive, slower

Oleg

Empty intersection surrounded by stones belonging to the same string

more restrictive, faster

Enhancements

- Progressive Pruning
- “All Moves as First”
- Temperature and Simulated Annealing
- Depth 2 Enhancement

Experiments

- 100 games on 9×9 board
- Alternating colors
- Standard deviation 15 points

  $\Rightarrow$ Standard error 1.5 points

Progressive Pruning

After an initial number of games statistically inferior moves are no longer selected

r_d: Ratio that defines when a move M_1 is inferior to M_2
in terms of their standard deviations

σ_e: Standard deviation for equality
Defines when a move M_1 is considered to be equal to M_2

Olga uses hard pruning

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

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

$\Rightarrow$ Use $\sigma_e = 0.2$, $r_d = 1$

All Moves as First

Optimizing move values no matter when they are played in the game (Gobble [Bruegmann 1993])

Speed-up: Number of random games independent of number of legal moves

Does not work well when move order is important (because of captures)

vs Olga(Basic)	vs Olga(PP)
$+13.7$	$+4.0$

Number of random games

Experiments performed with Oleg($N = 10000$)

1000	100000
-12.7	$+3.2$

$\Rightarrow$ Use $N = 10000$

Temperature and Simulated Annealing

Temperature: Play moves with non-uniform probability

$$\exp(Kv)$$

Results vs Oleg($K = 2$)

K	0	5	10	20
mean	-8.1	$+2.6$	-4.9	-11.3

$\Rightarrow$ Use $K = 5$

Simulated Annealing: Optimize move order, switch moves in priority list with probability based on temperature

Oleg(Simulated Annealing) vs Oleg($K = 5$)

$$\boxed{+1.6}$$

Depth 2 Enhancement

Use Monte-Carlo evaluation at leaf nodes of a depth-2 search

Prune moves in Monte-Carlo proven to be inferiour at depth 1

Depth$= 2$ vs Depth$= 1$

Olga	Oleg
-2.1	-2.4

- Performace is worse !
- $\max$ operator increases standard error of root node
- More games needed

All against All Tournament

- **GNU Go** 3.2
- **Indigo** 2002
- **Olga**(Depth=1, $r_d = 1$, $\sigma_e = 0.2$, PP, NOT All Moves as First)
- **Oleg**($K = 2$, NOT PP, All Moves as First)

	Olga	Indigo	GNU Go
Oleg	$+10.4$	-4.9	$+31.5$
Olga		$+1.8$	$+33.7$
Indigo			$+8.7$

Strength and Weaknesses

- Very little knowledge
- Likes to make strongly connected shapes
- Tactically weak
- Still too slow for larger boards

Perspectives

- Add tactics (as pre- or post-processing)
- Use domain-dependent pseudo-random games
  (e.g. patterns that influence the probabilities)
- Explore the locality of Go
- Define sub-goals