Monte-Carlo Go Reinforcement Learning Experiments

Paper by Bruno Bouzy, Guillaume Chaslot, IEEE 2006

Presented by Markus Enzenberger.
Go Seminar, University of Alberta.

November 16, 2006

Indigo project

- 9th KGS, 19x19, Dec 2005 (Formal: 3rd/7, Open: 1st/9)
- 8th KGS, 9x9, Formal, Nov 2005 (4th/11)
- 7th KGS, 19x19, Open, Oct 2005 (2th/7)
- 2005 WCGC, Tainan, Taiwan, Sept 2005 (6th/7)
- 10th CO, Taipei, Sept 2005 (19x19: 4th/7, 9x9: 3rd/9)
- 9th CO, Ramat-Gan, Jul 2004 (19x19: 3rd/5, 9x9: 4th/9)
- 8th CO, Graz, Nov 2003 (19x19: 5th/11, 9x9: 4th/10)
- Comp. Go Festival, Guyang, China, Oct 2002 (6th/10)
- 21st Century Cup, 2002, Edmonton, Canada (10th/14)
- Mind Sport Olymp. 2000, London, England (5th/6)
- Ing Cup 1999 Shanghai, China (13th/16)
- Ing Cup 1998 London, England (10th/17)

Basic Monte-Carlo Go

- General MC model (Abramson)
- all-moves-as-first heuristic (Brügmann)

- Standard deviation, random games, $9\times9 \approx 35$
- Precision: 1 point
- 1000 games $\equiv$ 68 % confidence
- 4000 games $\equiv$ 95 % confidence

Progressive pruning

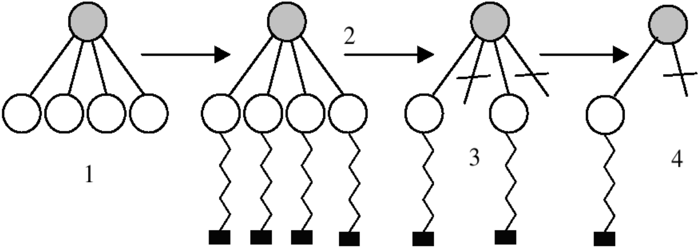


Fig. 1. Progressive pruning: the root is expanded (1). Random games start on children (2). After several random games, some moves are pruned (3). After other random games, one move is left, and the process stops (4).

Advantages of MC

- Bouzy, Helmstetter: MC programs on par with Indigo2002
- MC takes advantage of faster computers
- Heavily knowledge based programs less robust
- Global view (avoids errors due to decomposition)
- Easy to develop

Indigo2003

Two ways of associating Go knowledge with MC

- Pre-select moves for MC
- Insert knowledge in random games
  ("pseudo-random" := non-uniform probability)

Two Modules of Indigo2003

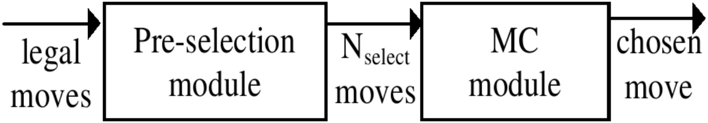


Fig. 2. The two modules of Indigo2003: the pre-selection module selects N_{select} moves by the mean of lot of knowledge, and local tree searches, additionnally yielding a conceptual evaluation of the position. Then, among the N_{select} moves, the MC module selects the move to play by the mean of random simulations.

Pseudo-Random (PR) player

- Capture-escape urgency: fill last liberty of one-liberty string
  urgency linear in string size
- Cut-connect urgency: 3×3 patterns
  (smaller: insufficient; larger: lower urgency)
- Total urgency is sum of both urgencies
- Probability of move proportional to total urgency

- 3x3 pattern urgency table
- 3^8 3x3 pattern (center is empty)
- 25 dispositions to the edge
- #patterns = 250,000
- one-liberty urgency

MC (Manual) vs MC(Zero)

- Manual: urgencies assigned to 3×3 patterns assigned by human expert
- correspond to Go concepts such as cut and connect
- contains non-zero and high values only very sparsely

board size	9x9	13x13	19x19
mean	+8	+40	+100
% wins	68%	93%	97%

TABLE II

RESULTS OF $MC(Manual)$ VS $MC(Zero)$ FOR THE USUAL BOARD SIZES.

Reinforcement Learning and Monte-Carlo Go

- Q-learning for learning action values
- p_1 better than p_2 (at random level) does not necessarily mean that $MC(p_1)$ better than $MC(p_2)(at MClevel)$
- Randomization vs. determinism

The randomness dimension

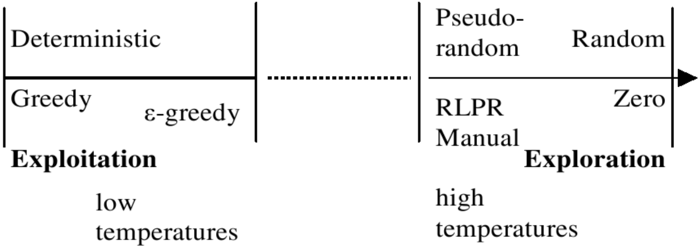


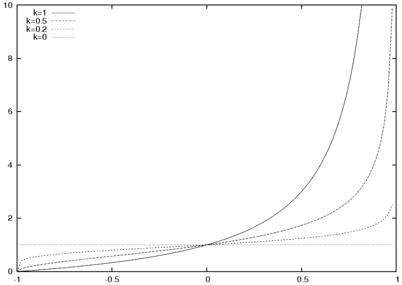
Fig. 3. The randomness dimension: the completely deterministic programs are situated on the left. $Zero$ is situated on the right. On the left of $Zero$, there are the PR programs and $Manual$. On the right of deterministic programs, there are ϵ-greedy programs [1]. The temperature indicates a randomisation degree: 0 for deterministic programs, and infinite for $Zero$, the uniform probability player.

Experiment 1: One program vs. population

- Patterns have action values $Q \in]-1, +1[$
- Urgency:

$$U = \left(\frac{1+Q}{1-Q}\right)^k$$

- Value updates after playing a block of b games:
$Q_{\text{play}} = Q_{\text{play}} + \lambda^b Q_{\text{learn}}$



Experiment 1a: one unique learning program

- RLPR $\gg$ Zero
- RLPR $<$ Manual
- MC(RLPR) $\ll$ MC(Manual)

- Highly dependent on initial conditions
- Q-values learnt in first block,
  later only these Q-values are increased

Experiment 1b: a population of learning programs

- Evolutionary computing
- Test against Zero and Manual
- Select and duplicate best programs according to some scheme

Starting population = Zero

- RLPR = Zero + 180
- RLPR = Manual - 50
- MC(RLPR) $\ll$ MC(Manual)

Starting population = Manual

- RLPR = Zero + 180
- RLPR = Manual + 50
- MC(RLPR) = MC(Manual) - 20

- RLPR programs have a tendency to determinism
- Using EC lowers the problem but does not remove it completely

Experiment 2: relative difference at MC level

- MC(RLPR) playes agains itself again
- Update rule to avoid determinism
  for a pair of patterns a, b:

$$exp(C(V_a - V_b)) = u_a/u_b$$

$$\delta = Q_a - Q_b - C(V_a - V_b)$$

$$Q_a = Q_a - \alpha\delta$$

$$Q_b = Q_b - \alpha\delta$$

9×9

- MC(RLPR) = MC(Manual) + 3

19×19 (learning on 9×9)

- MC(RLPR) = MC(Manual) - 30

Conclusion

- Determinism was identified as obstacle
- Evolutionary computing was used to avoid determinism (strong dependence on intial conditions)
- Relative differences were used to avoid determinism