# Monte Carlo Simulation Adjusting

**Nobuo Araki*, Masakazu Muramatsu, Kunihito Hoki, Satoshi Takahashi**
Graduate School of Informatics and Engineering, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585 Japan
*a1341001@edu.cc.uec.ac.jp
http://gi.cs.uec.ac.jp:10140/Araki_complementary/ReadMe.html

## Abstract

In this paper, we propose a new learning method *simulation adjusting* that adjusts simulation policy to improve the move decisions of the Monte Carlo method. We demonstrated simulation adjusting for $4 \times 4$ board Go problems. We observed that the rate of correct answers moderately increased.

## Introduction

The Monte Carlo (MC) method has driven recent advances in computer Go (Coulom 2007; Gelly and Silver 2007). A lot of game simulations are performed to achieve good move decisions. Simulation means that the program self-plays one game from a given position to the end. The probability of move selection in the simulation is often decided by softmax policy:

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_b e^{\phi(s,b)^T\theta}}, \tag{1}$$

where $\phi(s,a)$ is the feature vector of move $a$ on game position $s$ and $\theta$ is the weight vector. Because the quality of simulations affects the performance of MC searches, recent state-of-the-art Go programs have simulation policies that are carefully adjusted by hand and by machine learning.

In this paper, we propose a new learning method for simulation policy that aims to improve the move decisions of the MC method. In the first experiment, we used a set of $4 \times 4$ Go problems, Kuroneko-no Yonro (Hsu 2012), to measure the performance. These problems are relatively difficult, but the correct answers are available.

Our method is similar to simulation balancing (Silver and Tesauro 2009; Huang, Coulom, and Lin 2010). While simulation balancing requires desirable evaluation values (minimax values or the results of deep MC searches) to adjust the policy, our method requires desirable moves. Desirable moves can be powerful learning signals because plenty of expert moves in game records are available for Go. In fact, a popular learning method (Coulom 2007) uses such desired moves.

## Simulation Adjusting

Our purpose is to find the value of $\theta$ such that the best moves calculated by the MC method match the correct moves. We set $z$ as the game result (1 or 0), $V(s,a) = \mathbb{E}_{\pi_\theta}[z|s,a]$ as the winning rate of move $a$ at a given position $s$ calculated by using simulation policy $\pi_\theta$, $V(s,a^{\max})$ as the winning rate of the best move $a^{\max} = \arg\max_a V(s,a)$, and $V(s,a^*)$ as the winning rate of the correct move $a^*$. Our approach adjusts $\theta$ to match $\max_a V(s,a)$ with $V(s,a^*)$. The objective function value to minimize is

$$J(\theta) = \mathbb{E}_{\rho(s)}[(V(s,a^{\max}) - V(s,a^*))^2], \tag{2}$$

where $\mathbb{E}_{\rho(s)}$ denotes the expectation over the distribution of actual states $\rho(s)$. In the discussion below, we use the same notations as the paper (Silver and Tesauro 2009) for $\rho(s)$ and $\psi$.

To carry out the steepest descent, we obtain the gradient of $J(\theta)$ as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\rho(s)}[2(V(s,a^{\max}) - V(s,a^*))k(s)], \tag{3}$$

where $k(s)$ is $\nabla_\theta(V(s,a^{\max}) - V(s,a^*))$. We set $s_0 = s$ as a given position, $s_{i+1} = s_i \circ a_i$ as the game position that appears after move $a_i$ at game position $s_i$, $s_{T+1}$ as the end game position, and

$$\begin{aligned} k(s) &= \mathbb{E}_{\pi_\theta, a_0 = a^{\max}}[z\sum_{t=1}^{T}\psi(s_t,a_t)] \\ &\quad -\mathbb{E}_{\pi_\theta, a_0 = a^*}[z\sum_{t=1}^{T}\psi(s_t,a_t)]. \end{aligned} \tag{4}$$

In accordance with equations (3) and (4), the update algorithm is Algorithm 1. First, we compute the maximum winning rate $V_{\max}$, its move $a^{\max}$, and the winning rate of the correct move $V^*$ using the MC method. Second, we compute $\psi$ as defined in the paper (Silver and Tesauro 2009), and $k = h_{\max} - h^*$. Third, we update $\theta$ with step size $\alpha$ and repeat this procedure until $\theta$ converges. In our experiment, we set $\alpha = \frac{1.0}{10 \times \log(L \times 2.0 + 3.0)}$ where $L$ is the iteration number so that the step size decreases in accordance with the progress of the iterations.

**Algorithm 1** Adjusting $\theta$

```
$\theta \leftarrow 0$
for $L = 0$ to loop limit do
  for all $s_0 \in$ training set do
    $V_{\max} \leftarrow 0, V^* \leftarrow 0, a^{\max} \leftarrow$NULL
    $a^* \leftarrow$ teacher move in $s_0$
    for all $a_0 \in$ all legal moves of $s_0$ do
      $V \leftarrow 0$
      for $i = 1$ to $M$ do
        simulate$(s_0, a_0, \ldots, s_T, a_T; z)$ using $\pi_\theta$
        $V \leftarrow V + \frac{z}{M}$
      end for
      if $V > V_{\max}$ then
        $V_{\max} \leftarrow V, a^{\max} \leftarrow a_0$
      end if
      if $a_0 = a^*$ then
        $V^* \leftarrow V$
      end if
    end for
    $h_{\max} \leftarrow 0, h^* \leftarrow 0$
    for $j = 1$ to $N$ do
      simulate$(s_0, a^{\max}, \ldots, s_T, a_T; z)$ using $\pi_\theta$
      $h_{\max} \leftarrow h_{\max} + \frac{z}{N} \sum_{t=1}^{T} \psi(s_t, a_t)$
    end for
    for $j = 1$ to $N$ do
      simulate$(s_0, a^*, \ldots, s_T, a_T; z)$ using $\pi_\theta$
      $h^* \leftarrow h^* + \frac{z}{N} \sum_{t=1}^{T} \psi(s_t, a_t)$
    end for
    $\theta \leftarrow \theta - \alpha(V_{\max} - V^*)(h_{\max} - h^*)$
  end for
end for
```

## Experiments

We test Algorithm 1 on Kuroneko-no Yonro (Hsu 2012), a set of problems of $4 \times 4$ Go. The correct move for each problem is known and unique.

We construct the feature vector based on previous work (Coulom 2007) and the author's Go knowledge, e.g., $3 \times 3$ patterns, distance to the previous $n$ moves, ladder, nakade, semeai (capturing races) with a small number of moves, capturing stones in various situations, self-atari, escaping self-atari, tactical moves at board corners, atari in various situations, distance from board edge, and cut.

We select problems where black wins with 0.0 komi under the Chinese rules and has more than one legal move. We divide them into training data A (60 problems) and test data B (10 problems). For each legal move, we carry out 50 simulations to compute its winning rate ($M$= 50) and 50 simulations to compute $\psi$ ($N$= 50). We set the loop limit to 150. For each loop, we carry out one closed test (CT) for data A and 100 open tests (OTs) for data B. The results are shown in Figure 1. The number of correct answers (CA) is the average number of correct answers of B over 100 different pseudorandom number sequences. We also adjust the policy by using the popular method (Coulom 2007) with data A and test this policy on data B (CA of OT by the Minorization-Maximization (MM) method (Coulom 2007)). This figure shows that the objective function value (OF) of the closed test (OF of CT) decreases, and the number of correct answers of the open test (CA of OT) increases and is greater than the number of correct answers of the open test by MM (CA of OT by MM). We can also see that the result is improved in the first 10 iterations, and after that the number of correct answers oscillates.

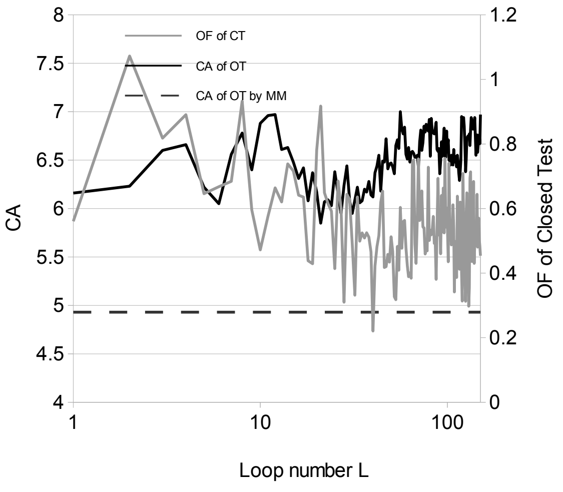

Figure 1: Result of closed test (training data of A and B and test data A) and open test (training data A and test data B.)

## Conclusion and Future Work

We observed that the objective function value decreased and the number of correct answers moderately increased. Increasing the size of training data may improve the number of correct answers, but this is future work. Performing the test on actual expert games ($9 \times 9$ and $19 \times 19$ games) and tuning the step size $\alpha$ are also future work.

## References

Coulom, R. 2007. Computing Elo Ratings of Move Patterns in the Game of Go. *ICGA Journal* 30(4).

Gelly, S., and Silver, D. 2007. Combining Online and Offline Knowledge in UCT. In *International Conference on Machine Learning, ICML 2007.*

Hsu, C. 2012. *Kuroneko-no Yonro*. Nihon Ki-in.

Huang, S.-C.; Coulom, R.; and Lin, S.-S. 2010. Monte-Carlo Simulation Balancing in Practice. In *Computers and Games*, 81–92.

Silver, D., and Tesauro, G. 2009. Monte-Carlo Simulation Balancing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 945–952. New York, NY, USA: ACM.