Using Global Corresponding Move Bias on MCTS

Cheng-Wei Chou [1], Shi-Jim Yen [2] and Jung-Kuei Yang [3]

[1, 2] Dept. Of Computer Science and Information Engineering, National Dong Hwa University, Taiwan
[3] Dept. of Applied Foreign Languages, Lan Yang Institute of Technology, I Lan, Taiwan

***Abstract*—In this paper we propose an efficient method to improve Monte Carlo Tree Search (MCTS) for the reuse information between different branches of MCTS and get very well experimental results. MCTS cannot share the result of semeai problem between different branches, but this paper try to use Global Corresponding Move Bias to improve this problem. The experiment result shows although this method is simple, but useful.**

Keywords- MCTS, UCT, Computer Game, Go

1. INTRODUCTION

The game of Go is a very old and popular game. [1] In recent years, the strength of Go program is benefited by the Monte Carlo Tree Search (MCTS) algorithm [2]. However, a common problem still perplexes the researchers of the computer Go field. When applying MCTS, some branches have clearly found the best moves of some semeai problems locally, but other ones still require finding the best move again, as in fig. 1. If the found results could be reused, the performance of tree search could be increased, and the strength of Go programs could be improved.

This paper proposes a method called *global corresponding move bias* to improve this problem. This method is inspired by RAVE [3] and LGRF [4]. Useful information is obtained by gathering the total visit counts of the responding moves of every location on the board. The information could be used to guide the direction of MCTS and make it more efficient. Related work, included MCTS, RAVE and LGRF will be introduced in section2. The detail of our method will be described in section 3. The experimental results of section 4 demonstrate that this method can improve the strength of Go program. This paper is concluded in section 5

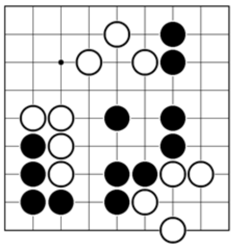


Figure 1 : The semeai problem of right bottom corner may be calculated several times in the same search tree.

2. RELATED WORK

2.1 MCTS

MCTS is a well-known and useful method in the field of Computer Game, especially for the game of Go. This method uses Monte Carle method to evaluate situations. Basically, MCTS has four basic stages, as fig. 2. MCTS repeats these four stages to build a search tree in memory until time out. Finally, MCTS select the best child by reward or visit counts of children of root. It is a best-first method, but uses some method, for example, UCB, to deal with the Exploitation vs. Exploration problem.

This search method builds an Asymmetric tree in memory incrementally. Those stages will be described below:

Selection : Every time, MCTS starts from the root, selects the best child in every level until reaches the leaf node of current tree. The strategy UCT [8], as formula 1, is the famous method to decide which child is the best. In formula 1, vi means past rewards of node i, Ti means the visit counts of parent of node i, N means the visit counts of node i. Bias is a parameter to balance exploitation and exploration.

$$v_i + bias \times \sqrt{\frac{ln\,N}{T_i}} \qquad (1)$$

Expansion : One (or more) new node(s) will be add the current tree. These nodes usually mean new boards by playing legal moves.

Simulation : This stage will play the game to end from the position of leaf node, and obtain result of the game.

Backpropagation : The result of simulation will update all nodes of the best sequence which are selected in *Selection* stage.

2.2 RAVE

The complete name of RAVE is Rapid Action Value Estimation. MCTS needs a few simulations to get a trusted value to decide a branch is good or not. However, when the numbers of simulation are not enough, MCTS will waste pleasure simulations to bad branches. RAVE uses the information of every simulation to estimate how quality of moves quickly. Of course, the quality of estimation is not very precise, so the effect of this method will decrease follow the simulation numbers increasing. This method has been proved very useful for Go and other games.

2.3 LGRF

In simulation step, the normal method is to use some heuristics with priority. However, the priorities usually are fixed.

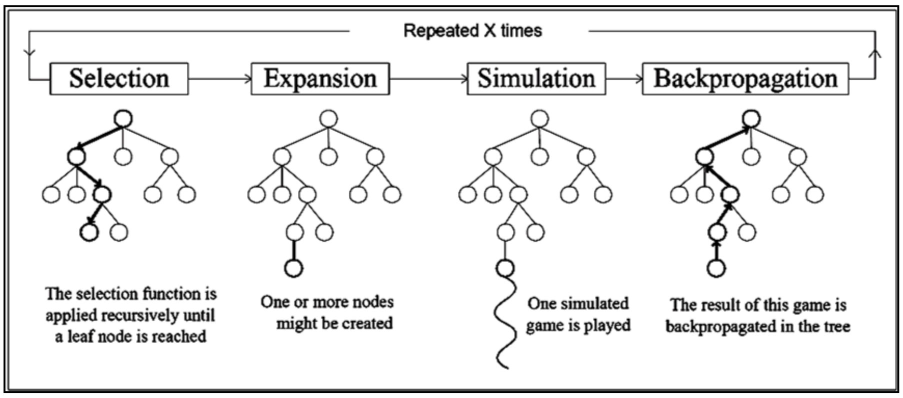

Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes might be created
One simulated game is played
The result of this game is backpropagated in the tree

Figure 2 : the four stages of Monte Carlo Tree Search.

H. Baier and P. D. Drake proposed a method, LGR (Last Good Reply), this method will remember the last win reply. They improved LGR to LGRF (Last Good Reply Forget), this method will forget the lost reply. These methods improve the quality of simulation and increase the strength of Go program.

3. GLOBAL CORRESPONDING MOVE BIAS

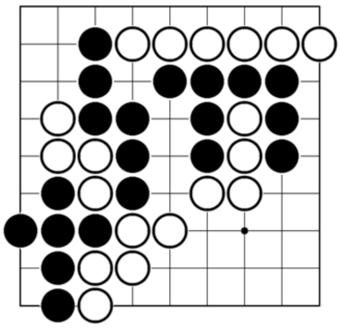
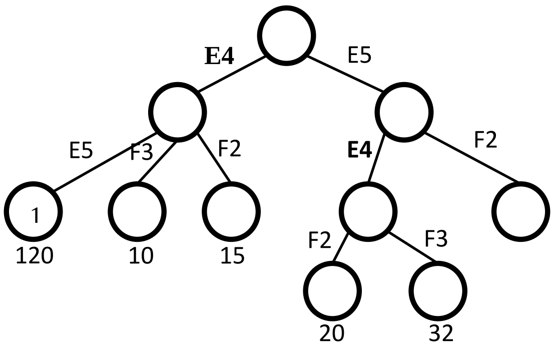


Figure 3 : Now turn is black. Black win the left bottom semeai problem and doesn't need to play anymore.

There is a problem in MCTS, the result of semeai problem in some branches of search tree cannot be used in other branches. For example, as fig. 3, there is a semeai problem in left bottom, and the result is black can capture white stones. Although in some branches, the result may be calculated, however, others branches cannot reuse the result, even the problem is almost same.

Of course, this is a big problem, and it is difficult to cut or distinguish if a situation or a semeai problem of certain branch is similar to another. Hence, we try to use more simple method. Because a semeai problem usually need ten or more steps, it is difficult to be reused. Our method try to focus on one move, this idea is inspired by LGR and LGRF. We try to record the replies of some move in search tree, and don't consider the situation is the same or not. We call this method as Global Corresponding Move Bias (GCMB). GCMB is similar to LGRF, but they are different. LGRF is used in playout, but GCMB is used to search tree. GCMB is similar to RAVE, too. However, RAVE use the move of both playout and search tree, but GCMB just uses the reply moves in search tree. Unexpectedly, this simple method obtains possible results. The remaining part of this section will describe the detail of GCMB.

In MCTS, we maintain a global table and record the total visit counts of responding moves of every move which is in the search tree regardless of the position. As the example in fig. 4, the visit counts of responding moves of move E4 are recorded. However, fig. 4 doesn't consider the colors. In fact, the information is recorded separately for different colors. These values will be cleared before beginning of search every time.



Global corresponding move bias of E4

Move	Visit Count
E5	120
F3	42
F2	30

Figure 4 : An example of the global corresponding move bias.

This information is used to update the initial value of prior knowledge. Originally, when the new children of search tree are created, the RAVE visits and RAVE wins are given according of the quality of the children to effect the direction of MCTS, called prior knowledge. The quality of children are decided by some heuristic of Go, for example, capture, escape, and pattern matching. In our method, the information of global corresponding moves is referenced to add the RAVE wins when the children nodes are created as the formula 2.

$$RAVE\_wins_i = RAVE\_wins_i + RAVE\_visits_i * GCMB_i \qquad (2)$$

$RAVE\_wins_i$ and $RAVE\_visits_i$ means the initial RAVE wins count and RAVE visits count of $child_i$, respectly. $GCMB_i$ means the percentage of child's move and farther's move. This method is launched only when the total visit counts of all responding moves are bigger than a threshold.

4. Experiment Results

4.1 Scalability

In the experiment, our program plays 4000 games against Fuego 0.4.1 SVN 1157 in the 9x9 board with 7.5 komi.

Table 1 : The main result for GCMB

Simulations per move	No GCMB	With GCMB
10000	57.5% (±0.8)	61.3% (±0.8)
60000	54.7% (±0.8)	60.2% (±0.8)

Table 1 demonstrates that global corresponding move bias could improve the strength of our program regarding of the simulation times per move.

4.2 Threshold Value

The second experiment is to change the threshold value. The simulation time per move is 10000. Table 2 shows the result. Although the threshold value is bigger, the quality of GCMB is better. But because the threshold value is bigger, the less nodes use GCMB to tune the prior knowledge. Finally, the winning rates of the two different threshold values are close.

Table 2 : The result for changing threshold value

Threshold value	Winning rate
100	61.3% (±0.8)
500	60.4% (±0.8)

4.3 Inner Data

The values of GCMB are used to tune the values of prior knowledge. We hope GCMB could help MCTS to focus on the better branches. In order to measure the effectiveness of GCMB, we compare the GCMB values and the real result of search. If the GCMB values could match the real searching result, the effectiveness of GCMB could be measured. First, the GCMB values are recorded when a node creates its children. Second, when the search is end, we calculate the probabilities of the best child of a node which visits count is greater than one thousand appears in the best N replies of GCMB. Table 3 shows the result of this experiment.

Table 3 : The probabilities of best move appear in best N replies of GCMB.

Total counts \ N	1	3	5	7
100~500	0.208	0.395	0.516	0.6
500~1000	0.249	0.453	0.574	0.66
1000~2000	0.291	0.498	0.626	0.706
>2000	0.358	0.6	0.715	0.785

5. CONCLUSION & DISCUSSION

In this paper, we try to obtain the information by recording the information of global corresponding moves and use the information to improve the performance of MCTS. The experiment results prove this method could steady increase the winning rate of the MCTS Go program.

Acknowledgments

The authors thank anonymous reviewers for their valuable comments, and thank the National Science Council of the Republic of China (Taiwan) for financial support of this research under contract numbers NSC 96-2628-E-259-020-MY3 and NSC 98-2221-E-259-021-MY3.

References

[1] Martin Müller. Computer Go. *Artificial Intelligence*, 134:145-179, 2002..

[2] Gelly, S., Wang, Y., Munos, R., & Teytaud, O. (2006). Modification of UCT with patterns in Monte-Carlo Go (Technical Report 6062). INRIA.

[3] Gelly, S. and Silver, D. (2011) Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*. Vol. 175, Issue 11, July 2011, pp. 1856-1875.

[4] H. Baier and P. D. Drake "The power of forgetting: Improving the last-good-reply policy in Monte Carlo Go", IEEE Trans. Comput. Intell. AI Games, vol. 2, no. 4, pp.303 -309 2010.