Combining Generalization and Localization Pattern Matching in the Game of Go

Yuuki Doi (0810041)

School of Information Science,
Japan Advanced Institute of Science and Technology

2011/2/8

Keywords: Computer Go, Monte Carlo Go, Pattern Matching.

In this thesis, we propose a new pattern matching which is easier to match than previous methods and covers larger area in the game of Go, and we evaluate the performance. We also improve the skill of our program by making the simulation quality more accurate with using a proposing method.

The skill of Go programs has not improved by *Minmax* search, which has been successfully applied in Chess or Shogi. The reason is the hugeness of the state space and the search space of Go. In 1993, Monte-Carlo method for Go (Monte-Carlo Go) was proposed and brought rapid advancements in computer Go, and it has improved the skill of Go programs. Monte-Carlo Go does *playout*, which simulates a game randomly from a position to the end, and repeats it. The position is evaluated by the results of simulation. i.e., wining and /losinge counts. It is known that the quality of *playouts* remarkably affects the skill of Go programs, and the enhancement is an important subject in this domain. In this thesis, we focus on enriching the quality of *playout* by introducing a more accurate pattern matching to positions.

In the early days, the depth of Monte-Carlo Go was just one ply, and the *playout* consisted from completely random moves. This approach has a problem that it tends to select an unprofitable move which turns out to be a bad move by looking forward with some depth, and consequently, it tends to expect opponent 's mistakes. A tree search algorithm and an evaluation function using pattern matching overcame this problem, and improved the skill of Go program. We use the same approach in this thesis.

A pattern in Go is a set of stone locations around a focused intersection, i.e., a candidate move. If an intersection is matched to a pattern, the intersection is scored by a corresponding value to the coefficient of the pattern. Common patterns are symmetrical to a given intersection, and the shapes are variable size of square, diamond or circle. Historically, this pattern had been used for a static evaluation function in *Minmax* search in order to decrease the number of legal moves to be searched. The patterns have a trade-off problem; a big pattern requires many game records to learn an adequate coefficient value, while small one has less ability to evaluate a move accurately. We have proposed various methods for this problem, but they had both good and bad points. Especially in *playouts*

of Monte-Carlo Go, patterns are harder to match to positions, because many positions in *playout* are not similar to one appeared in normal game records.

In this thesis, we propose Collage Pattern, which divides intersections around a move and covers them by several small patterns. It can covers the same area where a classical large patterns covers, and the each part of the patterns can be learned adequately because the size is small. Moreover it is expected to covers larger area than classical patterns. We implemented this method in a Go Program NOMITAN, which was studied and developed in JAIST. We had several experiments to evaluate the proposing method from variable aspects.

We compare traditional methods and Collage Pattern, and evaluate their performances of pattern matching, *playout* and strength in game playing. We propose some hierarchical evaluation items, and evaluate from many points such as, *playouts*, probability and order important things for strength. One of the contribution of this thesis is to organize these evaluation items.

Experimental results showed that Collage Pattern was improved in the performance of the generalization capability. The probability distribution was not improved so much. The *playout* speed was slower and program based on the pattern does not show very good performance. However, we obtained positive result for maximum size of pattern matching in *playout*s. Hence it has a potential of improving the performanceby adjusting the probability distribution and by changing the objective function for the learning. As future work, we would like to evaluate the quality of *playout* by fattening the evaluation items, and create an unrivaled strong Go program.