Ch.4 Enhancing the Simulation Strategy with Knowledge

This chapter is partially based on the publications:

- B. Bouzy and G.M.J-B. Chaslot (2006). Monte-Carlo Go Reinforcement Learning Experiments. IEEE 2006 Symposiumon Computational Intelligence in Games(eds. G. Kendall and S. Louis), pp. 187-194.

- G.M.J-B. Chaslot, C. Fiter, J-B. Hoock, A. Rimmel, and O. Teytaud (2010). Adding Expert Knowledge and Exploration in Monte-Carlo Tree Search. Advancesin Computer Games Conference (ACG 2009) (eds. H.J. van den Herik and P.H.M. Spronck), Vol. 6048 of Lecture Notes in Computer Science (LNCS), pp. 1–13, Springer-Verlag, Heidelberg, Germany.

What we discuss in this Chapter

- In this chapter we will discuss two different simulation strategies that apply knowledge:

1. urgency-based simulation.
2. sequence-like simulation.

Urgency-Based simulation

- Urgency-Based simulation is basically a 2 step method.

- In the first step, an urgency value, U_j , is computed for each move j.
- In the second step, taking the urgencies into account a move is randomly drawn.

The probability of each move

- M is the set of all possible moves for a given position.

$$p_j = \frac{U_j}{\sum_{k \in M} U_k}$$

About the Urgency-based Simulation

- If the urgency-based simulation strategy is too random, the level of play of the MCTS program will be close to the level of a program that draws plain randomly.

- If the urgency-based simulation strategy is too deterministic, the simulations will be too similar, which will lead to a lack of exploration and hence to meaningless Monte-Carlo simulations.

Computing Urgencies in the Game of Go

- In order to compute the urgency value of each move.

- Bouzy (2005) computed for his program INDIGO the urgency as the sum of two values:
  1. the capture-escape value V_{ce}.
  2. the pattern value V_p.

The meaning of the two values

V_{ce} depends on

1. the number of stones that could be captured.
2. the number of stones that could escape a capture by playing the move.

$V_p(i) = \sum_j w_j \times m_{i,j}$ where w_j is the weight of pattern j, and $m_{i,j}$ is 1 if move i matches pattern j and is 0 otherwise.

Sequence-Like Simulation

- This simulation strategy consists of selecting each move in the proximity of the last move played. This leads to moves being played next to each other, creating a sequence of adjacent moves.

How to select

- To select which move to play in the neighborhood of the last move, 3×3 patterns similar to the ones proposed by Bouzy (2005) were used. After each move, the program scans for 3 × 3 patterns at a Manhattan distance of 1 from the last move. If several patterns are found, one is chosen randomly. The move is then played in the centre of the chosen pattern. If no pattern is found, a move is chosen randomly on the board.

Learning Automatically the Simulation Strategy

- **Learning from Matches between Programs**
- **Learning from the Results of Simulated Games**
- **Learning from Move Evaluations**
- **Learning from the Mean-Squared Errors on a Set of Positions**
- **Learning from Imbalance**

Learning from Matches between Programs

- fitness function has two problems :

1. the number of parameters can be quite huge.
2. it is difficult to evaluate how each pattern contributed to the victory.

Learning from the Results of Simulated Games

- Learning from the results of simulated games consists of playing games between two simulation strategies .(let S_1 and S_2 be these strategies), and observe the results $r_1, r_2, ..., r_n$ of these games.
- The learning algorithm is then applied after each game, based on the decisions that have been made by S_1 and S_2 for the game i, and the result r_i.

Learning from Move Evaluations

- how each move should be evaluated.
- we chose as a move evaluation, denoted v_i for a move i, to use fast Monte-Carlo Evaluations.

Learning from Move Evaluations

- how the weights should be related to the move evaluations.

$$e^{C\times(v_a - v_b)} = \frac{w_a}{w_b}$$

- w_i as the weight of the pattern that matches for the move i. Second, we chose to associate the move evaluation v with the weight w such that for every pair of legal moves (a, b) in a board position.

Learning from the Mean-Squared Errors on a Set of Positions

- The learnt simulation strategy was as good as using expert patterns, but decreased the number of simulations. Hence, MOGO still plays with expert patterns.

Learning from Imbalance

- The imbalance is the difference between the errors made by the first player and the errors made by the second player.
- The underlining idea is that it is fine to make mistakes in the simulation if the other player makes mistakes as well.

	Learning from Imbalance	Learning from Move Evaluations
9×9	win	
19×19		win

Conclusions of Ch.4

1. **Avoiding big mistakes is more important than playing good moves.** If a move has a high probability to be a bad move, it should be avoided with a high probability.

2. **Simplifying the position.**

3. **Balancing exploration and exploitation.** The simulation strategy should not become too stochastic, nor too deterministic.

Conclusions of Ch.4

1. **Learning from matches between programs.**
   - The drawback of this method is that it is relatively slow, since learning can only be done in low dimensions.
   - The advantage of this method is that it is able to learn simultaneously the simulation strategy together with other parts of MCTS.
2. **Learning from the results of simulated games.**
3. **Learning from move evaluations.**
   - This method performed better than learning from the results of simulated games.
4. **Learning from the mean-squared errors on a set of positions.**
5. **Learning from imbalance.**

Ch.5 Enhancing the Selection Strategy with Knowledge

This chapter is based on the following publications:

- G.M.J-B. Chaslot, M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, and B. Bouzy (2007). Progressive Strategies for Monte-Carlo Tree Search. Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007) (eds. P. Wang et al.), pp. 655–661.

- G.M.J-B. Chaslot, M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, and B. Bouzy (2008c). Progressive Strategies for Monte-Carlo Tree Search. New Mathematics and Natural Computation, Vol. 4, No. 3, pp. 343–357.

- G.M.J-B. Chaslot, C. Fiter, J-B. Hoock, A. Rimmel, and O. Teytaud (2010). Adding expert knowledge and exploration in Monte-Carlo Tree Search. Advances in Computer Games Conference (ACG 2009) (eds. H.J. van den Herik and P.H.M. Spronck), Vol. 6048 of *Lecture Notes in Computer Science (LNCS)*, pp. 1–13, Springer-Verlag, Heidelberg, Germany.

What we discuss in this chapter

- *progressive bias* : Progressive bias directs the search according to knowledge.

- *progressive widening* : Progressive widening first reduces the branching factor, and then increases it gradually.

Progressive Strategies

- They are inaccurate when the number of simulations is low and when the branching factor is high.
- Such strategies use (1) knowledge and (2) the information available for the selection strategy.
- A progressive strategy chooses moves according to knowledge when a few simulations have been played, and converges to a standard selection strategy with more simulations.

Progressive Bias

- To direct the search according to – possibly time-expensive – heuristic knowledge.
- For that purpose, the selection strategy is modified according to that knowledge.
- The influence of this modification is important when a few games have been played, but decreases fast (when more games have been played) to ensure that the strategy converges to a pure selection strategy.

$$k \in argmax_{i \in I} \left(v_i + C \times \sqrt{\frac{\ln n_p}{n_i}} + f(n_i) \right)$$

- We chose $f(n_i) = H_i/n_i+1$, where H_i represents heuristic knowledge, which depends only on the board configuration represented by the node i.
- The variables v_i = the value of move, n_i = the visit count of i, n_p = the visit count of p, and C = coefficent

Progressive Widening

- Reducing the branching factor artificially when the selection strategy is applied
- Increasing it progressively as more time becomes available.

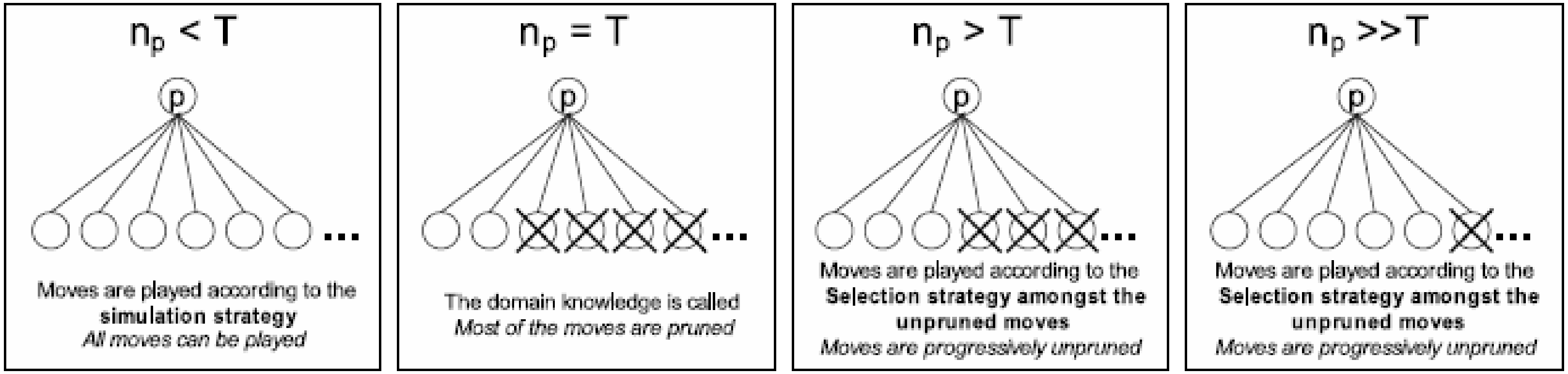

Figure 5.1: Progressive widening.