Incremental generation of possible moves in Shogi

Tsuyoshi SUZUKI, Nobuo INUI and Yoshiyuki KOTANI,
Tokyo University of Agriculture and Technology
2-24-16 Nakamachi, Koganei, Tokyo, 184-8588 JAPAN
go@fairy.ei.tuat.ac.jp, nobu@fairy.ei.tuat.ac.jp, kotani@cc.tuat.ac.jp

ABSTRACT

In othello and chess programs, the cost of generating possible moves is not a matter of importance because possible moves are not many on each position. Since there are many more possible moves in Shogi than the ones in chess, it is necessary to reduce the cost of generating possible moves. It is advantageous to search more nodes within the limited amount of time to reduce the cost of the generation. Presently, in Shogi programs, possible moves have been generated without using any previous data of move generation.

This paper proposes a new method to reduce the cost of the generation by using the data of moves in previous position. Firstly, we copy the previous data to the current data, and then we compute the difference between the previous position and the next position.

We examined this method in a Shogi program. Compared to a benchmark program that used a conventional move generation algorithm, our incremental technique produced a speed improvement of about 28%.

1 Introduction

The game of Shogi is a two-person complete information game like chess. Shogi is more complicated than chess. The reasons are as follows:

(1) 9 x 9 board
(2) dropped pieces can be reused
(3) there are many possible moves (max:600, average:80)
(4) each game lasts about 120 moves

A position is hard to evaluate, and a good evaluation function requires a great deal of knowledge. In spite of the effort in developing Shogi programs, the level of the best programs still remains low on the human scale.

An important guideline in developing a program is time constraints. This is crucial when there are many possible moves. The aim of this paper is to show the incremental generation of possible moves.

In section 2, we show the reason of using incremental generation. Presently, in Shogi programs, possible moves have been generated without using the previous position data. In our method, possible moves are generated by using the previous position data. We show the main idea of the incremental generation of possible moves in section 3. We also show the results of the experiment in section 4, and the conclusion is given in section 5.

2 Why is it necessary to generate incrementally?

Considering the average length of move sequence in actual games L and average branching factor B, there are $C=B^L$ states in games. In chess B=40, L=80 and $C=10^{128}$, but in Shogi B=80 L=120 and $C=10^{228}$, which shows complexity of games, and difficulty of developing the program.

Since there are many more possible moves in Shogi than the ones in chess, it is necessary to reduce the cost of generating possible moves. It is advantageous to search more nodes within the limited amount of time to reduce the cost of the generation.

Making observation of human Shogi players is very helpful in order to develop a Shogi program. For example, human players do not re-calculate each situation of the board after each move. In fact, he may calculate only situations that are affected by the last move. This remark was a motivation to find a mechanism about the incremental generation of possible moves.

The incremental generation of possible moves computes the difference between the previous position and the next position. Presently, in Shogi programs, possible moves have been generated each position fully. In Shogi, the cost of possible moves is not low. Since there is a little difference between the previous position and the next position, there is also a little difference between possible moves in the previous position and the next position.

Because the incremental generation of possible moves computes the difference between the previous position and the next position, it requires to compute less.

3 How to generate incrementally

In this section, we describe the method of the incremental generation of possible moves.

3.1 Previous method

Presently, in Shogi programs, possible moves have been generated without using the previous position data (we call this type-1).

Figure 1(1),(2) shows that 金 moves from 2三 to 3二, and Figure 2(1),(2) shows possible moves in Figure 1(1),(2) respectively.

There are eight moves in Figure 1(1). After 3二金 moves, there are nine moves in Figure 1(2). The previous method generating possible moves in Figure 1(1), and generate possible moves in Figure 1(2) without using the Figure 1(1) data after last move.

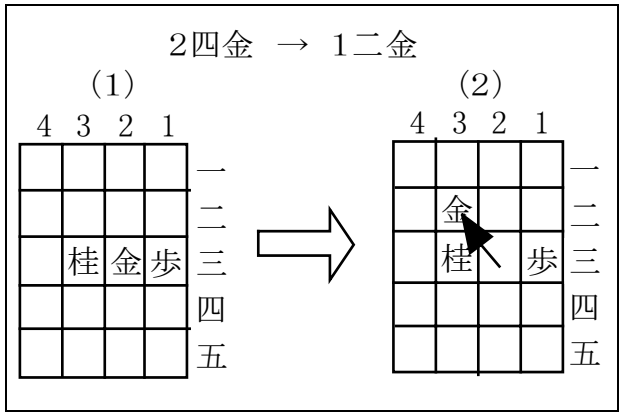

Figure 1: Example of change of possible moves

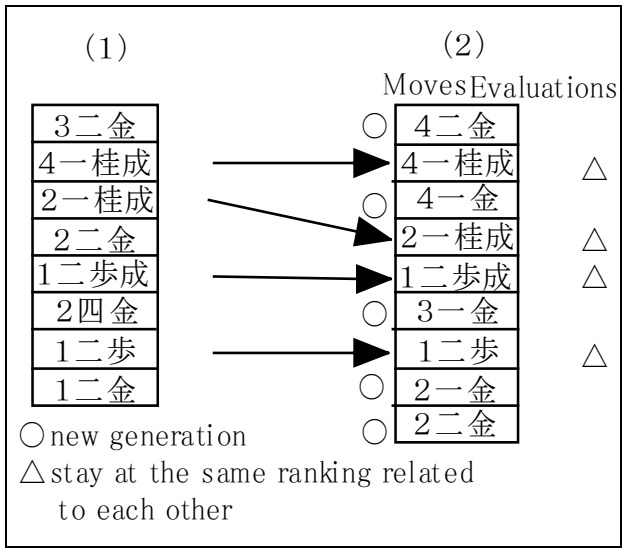

Figure 2: Possible moves in Figure 1

However, when we compare the list of possible moves in Figure 1(1) and the list of possible moves Figure 1(2), we find that:

(1) position is similar

(2) possible moves are similar

(3) move-ordering is similar

Hence, (3) helps alpha-beta search efficiently.

There is few difference between the previous position and the next position. Should we use the previous data?

3.2 Incremental generation of possible moves

As we point out in 3.1, there is a little difference of position, possible moves and move-ordering between the previous position and the next position. Average of possible moves in Shogi is about eighty, while last move changes a few parts of possible moves. Therefore, idea of the incremental generation of possible moves is that possible moves are generated by utilizing the previous position data.

The incremental generation of possible moves uses Black's possible-move-list and White's possible-move-list. These lists are updates after each move.

The incremental generation of possible moves is following (we call this type-2):

(1) possible moves in next position from the list of previous possible moves are selected

(2) moves from (1) are copied to the list of next possible moves

(3) the list of the new possible moves after the last moves is generated

(4) moves from (3) are added to the list of the next possible moves

(5) the list of the next possible moves is sorted out according to their evaluation

Figure 3 shows this operation in Figure 1. Firstly we select the possible moves in the list of the previous possible moves by the last one in the next position (Fig.3(1)) and their move are copied to the list of the next possible moves (Fig.3(2)). Then we generate the new possible moves after the last one, and their moves are added to list of the possible moves in next position. Finally we sort out the list of the next possible moves according to their evaluation.

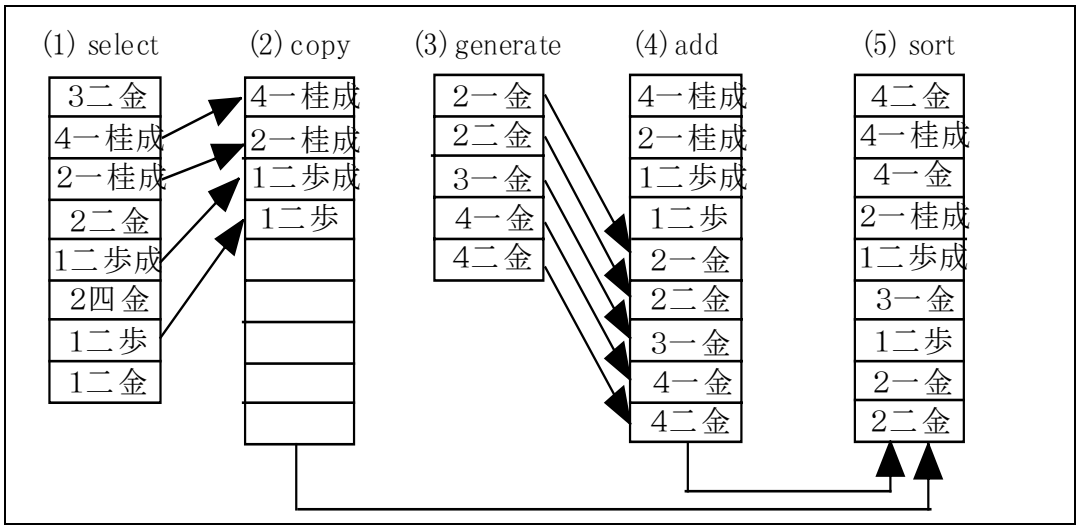

Figure 3: type-2 technique

3.3 Improvement on incremental generation of possible moves

We remark that the copied possible moves in the list of the next position are kept in their ranking. This helps sorting for the list of possible moves when to utilize alpha-beta game tree search. In general, the moves related in last moves obtain a high evaluation. By this fact, we get a following simple improvement (we call this type-3):

(1) possible moves in next position from the list of previous possible moves are selected

(2) moves from (1) is copied to the list of next possible moves

(3) the list of the new possible moves after the last moves is generated

(4) moves form (3) are added to top of the list of the next possible moves

(5) the list of the next possible moves is sorted out according to their evaluation

In Figure 3 the new generation moves added to last of the list of the possible moves. However in Figure 4 the new generation moves added to top of the list of the possible moves. By this means the sorting of the list of possible moves has more efficiency.

Most Shogi program searches top of the list of possible move by uses pre-pruning. Therefore we should sort the list of possible moves in the part of top of the list, which is for the next improvement.

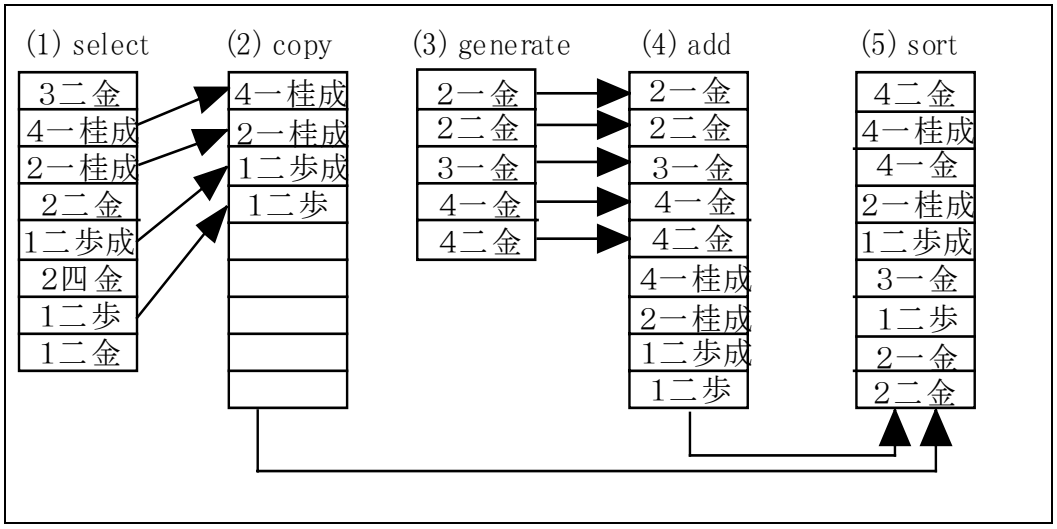

Figure 4: type-3 technique

We also propose that we first sort the new generated moves according to its evaluation, next we merge the new generated moves and the previous possible moves by the merge sort (we call this type-4).

(1) possible moves in next position from the list of previous possible moves are selected

(2) the list of the new possible moves after the last moves is generated

(3) the list of the new generation moves is sorted

(4) the new generated moves and the previous possible moves are merged

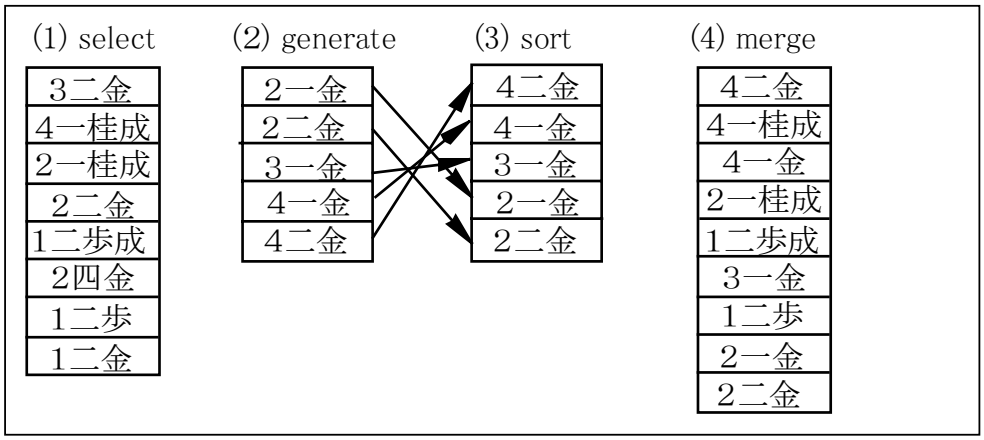

Figure 5: type-4 technique

Figure 5 shows this method in Figure 1. Moreover, we can also sort top of the list.

4 Results

We examined the type-1, type-2, type-3 and type-4 techniques by implementing them in a Shogi program. To allow our tests to evaluate solely the differences due to each move generation technique, we used an evaluation function that simply returned random numbers. We examined the experiments 50 games (about 5000 positions). Average length of move sequence in games is 97, average branching factor is 82 and 3-7 depths. The evaluation function consists of the value of pieces.

Figure 6 shows the result. In We define a unit as number of searched positions by type-1 program in a second (about 45000 nodes/sec).

Type-2, type-3 and type-4 generated the possible moves efficiently. There is little difference among type-2, type-3 and type-4. This means that sorting time of the possible moves is not influenced on search time.

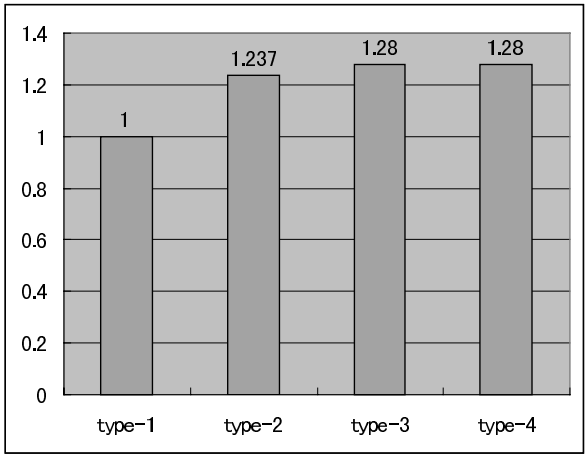

Figure 6: Result

Evaluation function requires the most time. Our method is no longer efficient for Shogi programs which search a few positions in a second, because our method is less used in game tree searching, but efficient for Shogi program which searches more 10000 position in a second. As shown in the figure, the number of the search positions in type-3 and type-4 increased about 28% compared to type-1.

5 Conclusion

This paper proposed the incremental generation of possible moves, and showed the result of the experiment. The incremental generation of possible moves is

(1) to utilize the previous position,

(2) to compute the difference between the previous position and the next position.

As the result of experiment, we can show that our method reduces the cost of move generation. Though we do not research the detail, our method is more effective for

positions that have a lot of possible moves.

Our program that is used for the experiment is still under development, we will examine the experiment which use the program with pre-pruning. We conjecture that our method is still efficient for such a program.

Reference

[1] Y.Kotani et al.: Computer Shogi (in Japanese), Science, Japan(1990)

[2] D.Levy and M.Newborn: How computer plays chess, Freeman and Company, New York (1990).

[3] H.Iida et al.: A Report on SHOUCHAN Project (in Japanese), Tokyo Univ. of A&T. (1994)