Construction of Opening Book in Connect6 with Its Application

Jun-jie Tao [1], Chang-ming Xu [2], Kang Han [2], Xin-he Xu [2]

1. Software College of Northeastern University, Shenyang 110004
E-mail: tjjhui@hotmail.com

2. College of Information Science and Engineering, Shenyang 110004

Abstract: An opening book is an important part in most game-playing computer programs. The so-called opening book is a database which contains an ocean of grandmaster's game records. Usually it is constructed manually by experts, or by selecting the excellent position from the massive raw game records by statistical. At the same time, some opening books can study and be optimized from new positions. But in connect6, it is impossible to construct an excellent opening book only by experiences from human beings. The reasons are: firstly, the average branch factor in Connect6 is huge; Secondly, the threat-based moves broadly existed in many positions, it makes the positions with small differences have opposite results. We use computer to generate the book automatically. This paper presents the way we construct the opening book of cennect6, and the convenience it takes when we test the program.

Key Words: opening book, iterative-deepening, connect6, Threat-Based Search

1. INTRODUCTION

Generally speaking, there are common beginnings in a lot of computer games which can be stored in computer in advance. The so-called opening book is a database which contains an ocean of grandmaster's game records or the excellent positions produced between computers and grandmasters.

Usually the strategies of searching and evaluation in the program were not adroit enough or sometimes even fixed, so it may waste a lot of time while received a bad move. After using opening book we need not to search each position at the beginning of the game, the move can be hit in the opening book. It helps us to save a lot of time and greatly reduces the probability of strategic errors as well. Majority of today's games adopt opening book, like chess [1], Chinese chess [2], and checkers [3] and so on. Lots of the opening books are fabricated by experts selecting from the existed heuristic positions, or some good positions generated by automatically position generating program searching the game search tree. There are shortcomings no matter what method above. To experts selecting positions, positions selected by expert is much too less, then experts may select the positions carelessly, the bad or even wrong positions may cannot be avoided. To the automatically generating positions, it is so hard to find a perfect evaluation method to satisfy all the positions, so the position may be not excellent as well.

This paper based on the rules and characteristics of connect6 is talking about a new method to construct whole opening book of connect6 and its applications.

This work is supported by National Nature Science Foundation under Grant 60774097

2. Connect6 and Its Rules

Connect(m, n, k, p, q) denotes a family games of k-in-a-row, like Connect(15, 15, 5, 1, 1) represents the famous connect5. Connect6 [4][5] is one of the most interesting, it is more fairer and easier.

In connect6 so-called Connect(m, n 6, 2, 1), the two sides are black and white. On the $m\times n$ board, except black puts only q=1 stone the first step of the game, the two sides put p=2 stones in each turn. The one who gets 6 sequence stones in one line include horizontal, vertical and diagonal. There is no eat and the stone cannot be putted on the cross filled by another stone. There is also no limitation of the size of the board, although, usually the board is 19×19. Connect6 is more complex than the chess which is proven, the complexity of state space and search tree space were 10^{172} and 10^{140}.It is considered as one of the most ideal object to study computer game.

It is noteworthiness that except the first step, black and white put two stones in one turn, which leads the great branch factor. At the same time, Connect6 is considered fair, it always draw at the end of the game while the level of the two sides are similar.

3. Constructing Opening books in Connect6

Most of the game programs like Chess, Chinese chess, checkers use opening books. It is proved that using the opening books promotes the performance of programs. However the importance of opening books varies from different games. In connect6 the opening book is also very important because of the widely existence of threats which may lead to lose. We must point out that the factor of connect6 is huge, so it can not image an excellent opening book which just depends on the experiences of human beings' especially grandmasters' experiences.

This paper talks about constructing a connect6 opening book by "enumerate and threat-based search". We decrease the total number of positions which should be enumerated by limiting the areas, at the same time we calculate all the positions' theoretical values of the search tree by off-line searching. Last we save the theoretical value of each position and the path lead to wining position to the opening book. There are 4 steps to construct the opening book:

- Select the size and areas of the board
- Draw up the method of storing and searching
- Generate positions and divide them
- Search and accomplish the fabrication

We will introduce the searching method for it is the core of the fabrication.

3.1 Factors Influencing the Size of Opening Books

There are two factors influenced the size of opening book: first, the number of stones one position include. Second, the size of board to putting stones.

1)7 stones position: every position in the opening book has 7 stones. There are two reasons: ① the total number of 9 stones positions, 2,054,455,634, is too large, but the total number of 7 is 184,072,680 which is acceptable. ② the win positions in 3 stones positions and 5 stones positions are not so many as 7 stone position. Synthesize above 7 stones position is better. The common 2 stones positions are not too much, so some common 3 stones positions as the foundation to enumerate the 7 stones positions.

2) 7×7 board fixed center: most of the 7 stones positions are in the board 7×7, and usually the value of 7 stones position which is not in board 7×7 is unknown. So every position in the opening book was fixed on the 7×7 board the center of which is at the center point. If the first stone is not at the center point of the board, we can move the board to the center by plus offset $x–x`$, $y–y`$(x, y is the originally, $x`$, $y`$ is the center after move). At the same time, the different orders to put stones are not considered in the same positions. In this way, for each one of the 3 stones position the number of positions should be generated is C(46, 2) × C(44, 2) = 979110.

C(n,m) is the Combination formula:

$$C(n,m)=n!/(m!\times(n-m)!) \qquad (1)$$

3.2 Methods of Storing and Retrieving Opening Books

To make index convenient, we save all theoretical values of 7 stones positions into several independent files. In each file, each position is derived from a same 3 stones ancestor. Any 3 stones beginning position corresponds one independent file.

Hash technology is widely used because many opening books of game programs are constructed by an ocean of scattered positions. Furthermore chess go and some other chess use SGF[6] files as the standard.

The opening book in this paper include all the positions with n stones (n<=7) are different to the opening books above. Hash is not the excellent method and SGF will decrease the efficiency because all the positions should be included and they are sequential. So the save and index methods should be reconsidered. In fact it is more like an endgame database[7][8].the whole opening book has 3 questions as follows:

- The memory allocate to each position
- Build the mapping relationship between the position and the entry address
- The way how to load

First, the question of the memory allocated to each position. Obviously, a two bits item is enough for each position in the opening book, considering the value of the search tree are win, lose and draw.

Then build the mapping relationship between positions and the entry addresses. It should be assured: 1） each position can be index in the opening book; 2） the index is easy to calculate; 3） avoid redundant. To assure 1 and 2 the easiest method is allocate a table for each position; the index of the position with n stones is defined as:

$$index = s_1+a_1\times s_2+\ldots+a_{i-1}\times s_i+\ldots+a_{n-1}\times s_n \qquad (2)$$

Among them, s_1, s_2, …, s_i, …, s_n represent the element number on the board($1\le i\le n$); let k=49, then the const coefficient a_i ($1\le i\le n$-1) is defined as:

$$a_i = (k-1)\times\ldots\times(k-i)\times(k-i+1) \qquad (3)$$

Notice that, different order may lead the same position. For example, exchange two black stones s_i and s_j, the position is the same as it is before. But in the formula (2) index is different after exchanging stones, which means the same positions will be saved many times. So the blacks and whites should be separated to be numbered to assure the index is invariable when exchanging any black stones or any white ones. In another way, the index is only related to the location of the stones, but unrelated to the order. So let

$$index = index_b + C(n, nb)\times index_w \qquad (4)$$

$index_b$ can be calculated by (5) and $index_w$ by (6). We number the nb black stones on board respectively as sb_1, sb_2, …, sb_i, …, sb_n, and $sb_1<sb_2<\ldots<sb_i<\ldots<sb_{nb}$. Then

$$index_b = sb_1+a_1\times sb_2+\ldots+a_{i-1}\times sb_i+\ldots+a_{n-1}\times sb_n \qquad (5)$$
$$a_i = (k-1)\times\ldots(k-nb)\times(k-nb+1),\ k=49$$

We number the $n–nb$ black stones on board respectively as sw_1, sw_2, …, sw_i, …, sw_{n-nb}, and $sw_1 < sw_2 < \ldots < sw_i < \ldots < sw_{n-nb}$.

$$index_w = \mathrm{sw}_1+\mathrm{a}_1\times sw_2+\ldots+a_{i-1}\times sw_i+\ldots+a_{n-1}\times sw_{n-nw} \qquad (6)$$
$$\text{among, } a_i = (k-1)\times\ldots(k-n+nb)\times(k-n+nb+1),$$
$$k=49-nb$$

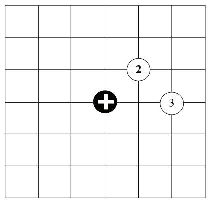
Theorem 1: formula (4) can uniquely indicate one position.

Proving: assume $x_1\ne y_1$ $x_2\ne y_2$, let $x_1+C(n, nb)\times x_2 = y_1+ C(n, nb)\times y_2$. Then $(x_1–y_1)+C(n, nb)\times(x_2–y_2) = 0$, because $x_1–y_1 < C(n, nb)$, $x_1=y_1$ $x_2=y_2$, so it is impossible that $x_1\ne y_1$ and $x_2\ne y_2$.

3.3 Generating Positions for Search

Paragraph 3.2 talks about the method to store and search the opening book. This paragraph will talk about how to save the value of the search tree into the opening book. The total number of 7 stones positions in the opening book is 184,072,680.each position should be searched to get the theoretical value. In fact, it is redundant to search every position because a lot positions are equivalent, which means the positions are the same after one or more rotating or flipping.

Obviously, the equivalent positions have the same results after searching, so only one should be searched. There are so many equal positions in the 7 stones position. For example figure 1 is a common 3 stones opening position (+ represents the center point)



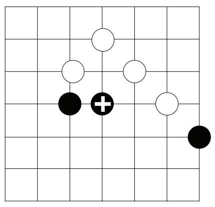


Fig. 1: 3 stones opening position Fig. 2: 7 stones opening position

We can get lots of 7 stones positions by enumerating, arbitrarily choose one of them assumed to be showed in figure 2. we can get four positions A、B、C、D in figure 3 by rotating it and four other positions E、F、G、H by flipping horizontally

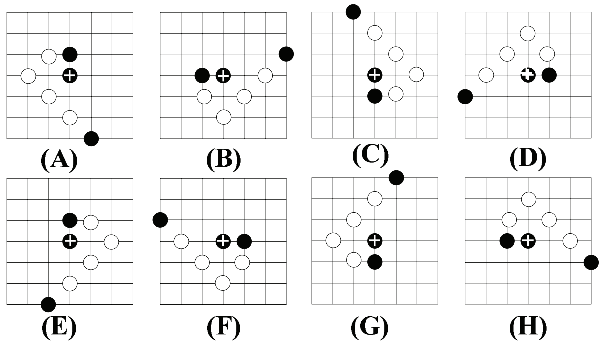


Fig. 3: 8 equivalent positions after rotating and flipping.

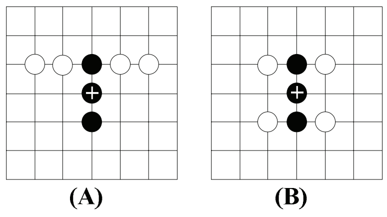


Fig. 4: Non-8 equivalent positions after rotating and flipping.

There are two situations that the number of equivalent positions is not 8 after rotating or flipping, in figure 4 positions A has four equivalent positions while position B has only two. Overall, the number is much less after rotating and flipping.

The opening book is large, generally, we save it in files. Along with the game process going, lots of the positions can not be encountered. So it is a waste memory to load all the positions into memory.

A simple method is divided the position files in accordance with the ancestor node, the positions with the same ancestor are in one file. At last load the positions in the small files by the indexes of the positions and retrospect the specific position then search it, save the result into relevant file to construct the opening book.

Last it must be known for a certain position how to search its value in the opening book. First calculate the position's index, because all the files include 100 positions, so the name of the file which the position in it is index/100.at the same time, index%100 is the number of the position in the file. Now we can easily find the value by the index of the position.

4. Search

Aft Most of the game programs like Chess, Chinese chess, checkers use opening books. It is proved that using the opening books promotes the performance of programs. However the importance of opening books varies from different games. In connect6 the opening book is also very important because of the widely existence of threats which may lead to lose. We must point out that the factor of connect6 is huge, so it can not image an excellent opening book which just depends on the experiences of human beings' especially grandmasters' experiences.

4.1 Thread-based Search

Threats are common in connect6, so the search algorithm for constructing the opening book in this paper is threat-based search. Threat-based search is the algorithm that creates threat after Attacker (A) has made move and Defender (D) must reply this move or A will win. Search begins from A, in this paper we assume that black is A (the result is correct if we exchange the two sides) .So it's black's turn after 7 moves, we use two-threats-based search to search the position. In the Figure 5, '1' represents able to form two threats, '0' represents unable to form two threats or Defender creates two threats after defending,'2' represents the win position .In the figure there are 3 ways to form two threats for position A, but for position C only E forms two threats, once defender do not moves E attacker unable to win. For position D attacker can form two threats in each defend, moreover for the only two positions H and I defender lose, so position A is the won position. The pseudo code of the threat-based search is as follows:

```
Attack（POSITION $i$）{
    if（WIN=Evaluation（$i$））
        Return WIN;
    for（each position with two-threats）
    {
              if（WIN=defend（$P_i$））
                 Return WIN;
    }
    Return NON-WIN;
}
```

```
Defend  POSITION i  {
    for  each position after eliminate
    threats
    {
      if  NON-WIN =Attack  P_i
              Return NON-WIN;
    }
    Return WIN;
}
```

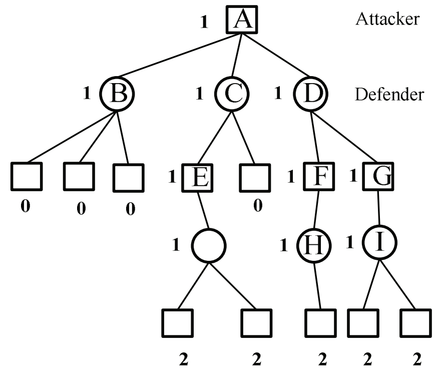


Fig. 5: Threat-based search tree

4.2 Depth-first iterative-deepening

For depth-first search, one branch will be searched until the end, but it is not sure the answer is in the first several branches. It is time consuming while searching with fixed depths may unable to find the answer. Depth-first iterative-deepening search algorithm adds a depth condition, when the ply is equal to depth the value of this position is unknown, then we let depth plus 1 and search the position again, repeat it until get the answer.

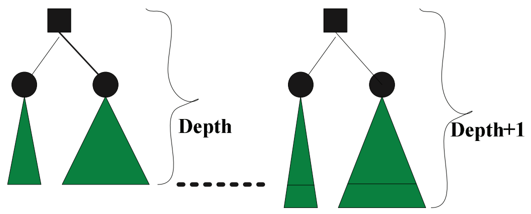


Fig. 6: Iterative-deepening

The result is best in the searching process which means the shortest path. In figure 6, “0” represents unknown, “1” represents win, “2” represents non-win. The result is unknown in figure 7(a). The max ply plus 1 then we research it in figure 7(b), when search to J, J is proved to win then E is proved bottom-up, after that C is proved to win through E and F. at last A is proved to win.

In the searching process, the node may be searched many times, so the hash table technology is necessary in the iterative-deepening search algorithm. It is proved by lots of programs that the usage of hash table is bigger when the depth is larger.

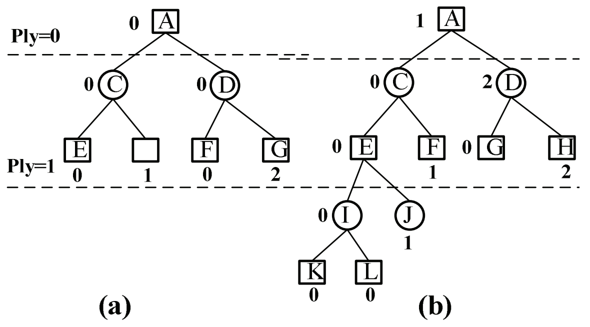


Fig. 7: Iterative-deepening search

4.3 Converting Binary Value Search to Triple Value Search

Though each node in the search progress has a bi-value(win, unknown), we can know the tri-value by a special method, first to the perspective of the attacking side to verify if the attacking side in the current situation is win. If win for the current position is that the attacking side to win or exchange perspectives for two sides to do the search again. If defender is win the current position is lose for attacker or the position is unknown. Through this way the bi-value becomes tri- value.

4.4 Shortcomings

The theoretical values of the search are win, lose or draw in theoretical. But for current search, some of the positions are still unknown, and that is because not all positions have two threats while the threats in connect6 widely exist, some positions have only one or even no threats. It is common for current program cannot find the definite result to make the program powerful, so we are forced to single-threat based search. It is obviously more powerful to search the position again by single-based search, and the total number of unknown position will decrease a lot. But it is time consuming to use single-threat-based search because of the flexibility of the left move after defender destroyed the only one threat. So it became impossible to use the single-threat-based search for such a lot of positions in the opening book. However, I believe, after a certain improved even non-threat-based search should be used to optimize opening book.

5 Experiment

5.1 Configuration of Machine

Intel(R) Core(TM)2 Duo CPU E8400 @ 3.0GHz 1.98 GHz,2.00GB memory.

5.2 Experiments and Results

Few positions in the file is complex, to assure the tree is not too deeply searched we limited the max depth to 16 plies(a total of 64 moves).

In order to speed up the progress we spent a total of 28 machines to search simultaneously. In the search process, only the individual position of the depth is greater than 16 plies (refer to search a total of 64 moves), 99.998% of the positions are in depth of 16 plies.

The following are five position files extracted in the searching process.

Tab. 1: Data in files

	S1	S2	S3	S4	S5
Num	100	100	100	100	100
Proven	83	61	52	96	98
DisProven	17	39	48	4	2
Unknown	0	0	0	0	0
Ave_t	248.75	780.17	123.91	138.17	94.
Ave_p_t	230.05	217.97	189.63	141.65	95.66
MAXPly	10	8	6	6	5

S_i $1 \leq i \leq 5$ represents position files, there are 100positions in each position. Proven represents the position which is wins, Disproven in the contrary. Unknown represents the position with out an exact result. Ave_t represents average time used for searching; Ave_p_t represents the average time to prove the position. One ply represents one round(two moves each total 4moves), MAXPly is the max ply searched.

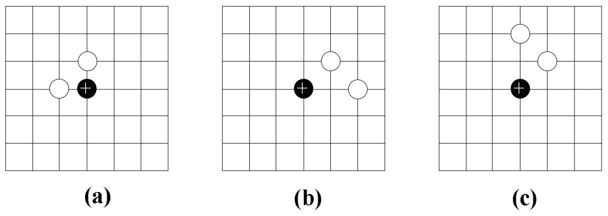


Fig. 8: Common 3 stones positions

We construct 3 three stones positions as Figure 8. These 3 positions include total 2,937,330 positions. After symmetrical about 52.32% Redundant were removed. Only 1,400,488 positions should be searched. In the positions 341,787 of them are proved to be win, about 24.405%; 1,058,670 can not to be proved win which is 75.593% of the total positions. The number of unknown positions is 31, about 0.002%.

5.3 Application in Test

The opening book not only can be used at the start of the game, but also can be used in the test part. We use some other algorithm to search the positions. It is hard for us to test the accuracy of the program; it may contain some small errors while the test positions get all right results. We extract some of the positions to test the program, making the credibility of the correctness of the program greatly increased.

6. Forward

It should be noted that the opening book can be perfected, there are only two-threat win positions in current opening book, the single-threat win position even the non-threat win position should be in the opening book. At the same time we should research new ways to decrease the number of the nodes to search in the algorithm. Iterative-deepening is largely a waste, there should be better ways to achieve better search.

REFERENCES

[1] KARAPETYAN A, RICHARD J.Lorentzgen (References) generating an opening book for amazons.4th International Conference on Computers and Game[C]. Ramat-Game, Israel, 2004.

[2] Wei Qin-gang ▪ Wang Jiao ▪ Xu Xin-he ▪ Nan Xiao-fei A study and design of opening-book of computer Chinese Chess CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS 2007

[3] J. Schaeffer, Y. Bj¨ornsson, N. Burch, A. Kishimoto, M. M¨uller, R. Lake, P. Lu and S. Sutphen: "Solving Checkers", Proceedings of IJCAI-05, pp.292-297, 2005.

[4] Wu I-C, Huang D-Y, A New Family of k-in-a-row Games, in Proceedings of The 11th Advances in Computer Games Conference, 88-100, 2005.

[5] Wu I-C, Huang D-Y, Chang H-C, Connect6, ICGA Journal, 28, 4, 234-242 ▪ 2005.

[6] http://www.red-bean.com/sgf/index.html

[7] K. Thompson, Retrograde analysis of certain endgames, ICCA J. 9 (3) ▪ 131–139 ▪ 1986.

[8] K. Thompson, 6-piece endgames, ICCA J. 215–226 ▪ 1996.

[9] Allis, L., van den Herik, H., and Huntjens, M. (1993). Go-Moku and Threat-Space Search. Report CS 93-02, Department of Computer Science, University of Limburg, Maastricht, the Netherlands.

[10] Hsieh Ming-Yu, Tsai Shi-Chun, on the fairness and complexity of generalized k-in-a-row games, Theoretical Computer Science, 385, 1-3, 88-100 ▪ 2007.

[11] M. Buro. Toward Opening Book Learning. ICGA Journal, volume 22, no. 1 ▪ 98-102 ▪ 1999.

[12] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence, 27:97–109, 1985.

[13] T. Lincke. Strategies for the Automatic Construction of Books. In Marsland, T, Frank, I. editors, Proceedings of the Second International Conference on Computers and Games (CG2000), volume 2063 of Lecture Notes in Computer Science. Hamamatsu Japan. Springer-Verlag ▪ 74-86 ▪ 2001.

[14] http://www.elephantbase.net/