Research on Techniques of Computer Game

Hongkun Qiu[1], Fuxiang Gao[2], Yajie Wang[1] ,Yan Ge[1], Quan Qiu[3], Huimin Sun[1]

1. Engineering Training Center, Shenyang Aerospace University, Shenyang 110136, China
E-mail: qiuhongkun_cn@sina.com

2. Institute of Information Science and Engineering, Northeastern University, Shenyang 110819, China
E-mail: gaofuxiang@ise.neu.edu.cn

3. School of Information Science and Engineering, Shenyang ligong University, Shenyang 110168, China

E-mail:962441891@qq.com

Abstract: Computer game is regarded as a new research field of Artificial Intelligence. It's very difficult to design and implement a high-level computer game system. Some techniques of computer game were presented with an example from a chess game - Connect-six. Firstly, introduced the game tree and searching techniques. Secondly, the key technologies for computer games were discussed, including establishment and application of Hash Table, improvement the efficiency of pruning. Finally, some contrasting experiments were performed to test the performance of the program which used the previously mentioned techniques. The experiments' results indicated that these techniques are very useful for computer game. And they have been confirmed by a series of competitions and tests. By these researches, a high-level game system can be produced and the results and conclusions of this paper also have some reference values to other computer game procedures.

Key Words: Computer Game; Minimax Algorithm; Alpha-Beta Pruning; Hash Table; Evaluation

1 INTRODUCTION

With the development of computer technology, the idea of playing chess by computer was come true. Computer game (CG) is regarded as a new research field, and it comes from the old games. The research of computer game originated from 50s' in the 20th century. Nowadays it has been widely accepted by a large number of students and scholars. Computer Games is an important branch of Artificial Intelligence (AI) which is one of the active research areas in recent years. It is considered to be one of the most challenging research directions in the field of artificial intelligence. It is based on artificial intelligence and kinds of computer game techniques. Research results of it contribute to AI not only in methods but in theories as well as remarkable influences to social and science. [1]

Precursors of AI once expressed earnestly: If people can master the very core of chess-playing, perhaps they can master the core of human behavior. Some computer game, for example, Chinese chess has won great success and that of chess is going through boom, the search techniques of which provide a well frame of reference to other computer game systems. But different rules in different games bring on specialties of each game. Thus, deeply researches of each game on its principle and its inherence law are needed in order to design a concrete computer game system.

Some games, such as Chinese chess, Go, Checker, Surakarta, Amazon and Connect-six, are complete information extended game, that is to say, the participant's strategies only rely on the game situation in the game .[2,3] Comparing with some games like Chinese chess, Go, Checker, and Connect-five, that have the mature computer game algorithm, there is few research on the other chess games nowadays. So it is significant to research on technologies of computer game in the field of artificial intelligence. In relate to the respective characteristic of each game, it is very important to establish a reasonable and efficient computer game system in the following research. The results and conclusions of a special game system also have some reference value to other computer game procedures.

Because of the complexity of computer games, it's very difficult to design and implement a set of computer game system. Some technologies of computer game must be thought about.

2 GAME TREE SEARCHING TECHNIQUES

Game trees are important in artificial intelligence because one way to pick the best move in a game is to search the game tree using the minimax algorithm or its variants. In game theory, a game tree is a directed graph whose nodes are positions in a game and whose edges are moves. The complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position. The number of leaf nodes in the complete game tree is the number of possible different ways the game can be played.

Computer-person games can also be represented as and-or trees. For the first player to win a game, there must exist a winning move for all moves of the second player. This is

978-1-4673-5534-6/13/$31.00 ©2013 IEEE

represented in the and-or tree by using disjunction to represent the first player's alternative moves and using conjunction to represent all of the second player's moves.

With a complete game tree, it is possible to solve the game – that is to say, find a sequence of moves that either the first or second player can follow that will guarantee either a win or tie. The algorithm is generally called backward induction or retrograde analysis.

The game tree for Connect-five is easily searchable, but the complete game trees for larger games like Go and Chinese chess are much too large to search. The search space is a very huge number to these games. For example, the number of possible different ways the Chinese chess can be played is estimated to be a total of 10^{144}, and the game tree for Go even has 10^{172} leaf nodes. Instead, a chess-playing program searches a partial game tree: typically as many plies from the current position as it can search in the time available.

For computer game, move generation process is also the game tree expansion process, as shown in Figure 1.

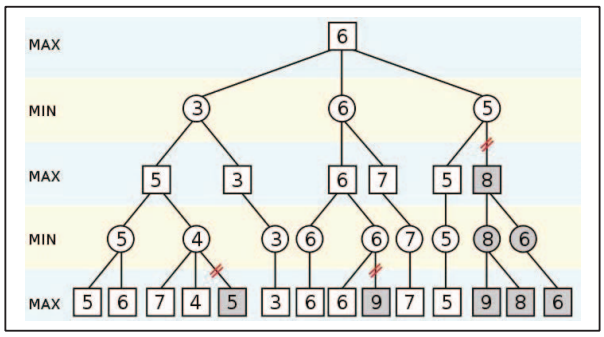

Figure 1. An illustration of four degree game tree alpha-beta pruning

In the game tree expansion process, the node number according to the current situation changes from several thousand to several hundred thousand. It is of great workload to the program. Therefore, a good planning is made to simplify game tree search.

Game tree search is a very general technique; it describes any algorithm that explores the space of possible solutions in a sequential fashion, moving in one step from a current solution to a nearby one. The generality and flexibility of this notion has the advantage that it is not difficult to design a game tree search approach to almost any computationally hard problem; the counterbalancing disadvantage is that it is often very difficult to say anything precise or provable about the quality of the solutions that a game tree search algorithm finds, and consequently very hard to tell whether one is using a good search algorithm or a poor one.

Game tree search algorithms are generally heuristics designed to find good, but not necessarily optimal, solutions to computational problems, and by talking about what the search for such solutions looks like at a global level.

3 HASH TABLE

3.1 The establishment of hash structure

Usually hash structure should at least have two members, first member is used to store situation, the second member is used to store the corresponding forms situation, if necessary can also add other members to improve Hash table search efficiency. To use hash table should pay attention to the storage situation. The hash value should be 64-bit integer because using the 64-bit integer is difficult to cause hash conflict. And it will more likely cause hash conflict to use 32-bit integer for larger game tree.

For examples, some of important data structures of hash structure are defined by using C++ language as shown in bellow:

```
// Define start library hash structure.
Struct HashJoseki
{
__int64 checksum[8]; // Storage situation.
   Step step[8];  // Storage situation the corresponding forms
};
//Define search history of the hash structure.
struct HashHistory
{
   bool pruning;
// If  the move is listed in the calculation, linear pruning is not accepted.
__int64 checksum; // Existing hash situation.
   vector<Step> MoveList; // Save the current situation of the move.
};
```

3.2 Hash structure applied to start to the library

In the use of the library start first start in the library all situation and corresponding rook read to memory, when found computer to deal with the situation and start the library situation is consistent, is directly used in the library start forms, which requires hash table to store start library, and hash table should be greater than the size of the start of the size of the library, in order to expand the size of the library can start.

Some codes of the Search Class, which defined by C++ language, are listed in bellow:

```
//Search hash table for the existing moves.
bool Search::SearchCM()
{
   for(int k=0;k<CMNum;k++)
      for(int i=0;i<8;i++) // Search moves.
```

```
    {
      if(HashKey64==HashCM[k].checksum[i])
      {
        HashStep=HashCM[k].step[i];
// If the current hash value equal to the moves of the hash value, direct return and set the  move as a good step.
        return true;
      }
    }
  }
  return false;
}
```

3.3 Hash structure applied to generate forms and screening

When the search is completed, it have already search all the situations and the corresponding forms of stored in the corresponding depth hash table. When it shows the same situation again in the next time, it may use hash table directly to store in the forms for expansion, and don't have to repeat formation.

Some key codes in program of Hash generated forms are listed in bellow:

```
// If found in the hash table rook, adopted it directly
if(LookUpHashTable(Depth)==true)
  MoveList=HashList.MoveList;
```

When the hash table find rook, the hash table should be form according to the historical value reordering. The key values should be sorted from the big to the small. And this should improve the pruning efficiency. Some key codes are listed in bellow:

```
// Hash rook screening
for(iter=MoveList.begin();iter!=MoveList.end();iter++)
{
  iter->SValue=iter->LValue;  // Forms the historical value assignment to forms used to sort of reference value
}
If(LookUpHashTable(Depth)==true)
{
  sort(MoveList.begin(),MoveList.end(),ComSValue); //Step descending or ascending according to the maximum or minimum search
MoveList.resize(MoveList.size()*RATIO);// According to the proportion of the RATIO rook screening
}
```

4 IMPROVE THE EFFICEINCY OF PRUNING

It is possible to solve a game using only a subset of the game tree. A move need not be analyzed in many games if there is another move that is better for the same player (for example alpha-beta pruning can be used in many deterministic games).

Any subtree that can be used to solve the game is known as a decision tree, and the sizes of decision trees of various shapes are used as measures of game complexity. [4]

Alpha-beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.). It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision. [5]

The grayed-out subtrees need not be explored (when moves are evaluated from left to right), since we know the group of subtrees as a whole yields the value of an equivalent subtree or worse, and as such cannot influence the final result. The max and min levels represent the turn of the player and the adversary, respectively.[6,7]

In uses Alpha-Beta pruning program, if for each layer node make the ideal sort, can the search depth expanded to two times the left and right sides without extra time-consuming; If the each layer node make the not ideal sort, it is in the same search depth, uses Alpha-Beta pruning of the search and uses Alpha-Beta pruning of the search the amount of time roughly the same. So to node sort ideal level directly affects the Alpha-Beta pruning efficiency, and introducing hash table can improve the node sort ideal degree. When to find the hash table forms, to expand the rook game before the tree should be first to determine a more ideal standard forms of sorting screening, and then spread to the game of the tree. Ideal standard should be the node according to the search.[8,9,10]

To expand the search depth and spend little time, some techniques can be applied. When the game tree is searched to the fifth layer, for example, it is obviously to waste a lot of time spreading the first five layers. Whether or not it be searched deeply based on the following situations.

A. When the chess game formed double threat situation or can form double threat situation, the game tree should continue to be spread.

B. The game tree should be spread if meet the single threat situation and can have a rook form his own single threat situation.

C. When above two conditions missed, interrupt expansion immediately.

However, it is necessary to adopt a more restrictive way on expanding the searching depth. Otherwise the expanding

process will spend too much time. It may be a very long time to expand the searching depth to ten layers, for example, and it may be a very short time to expand the search depth to two layers. So the search depth should be adjusted to the best state, and its situation assessment value will be as final return value in the computer evaluation.

5 EXPERIMENTS

Some contrasting experiments were performed to test the performance of the program which used the previously mentioned techniques. One screen-shot of the experiments are shown in Figure 2. As an example, the left figure in Figure 2 showed five layers depth search and no extension depth screen, by contrast, the right figure in Figure 2 showed five layers depth search and expands the depth to nine layers search. In the left figure of Figure 2, the black rook is obviously bad, because white can continuous forming double threat. The experiments and measurements showed that the white chess threatened black until the black lost. In right figure the black rook inhibit the development of white and black chess type more favorable. The contrast results show the superiority of the depth of extension. The contrast experiment results are listed in the follow Table 1, and it indicated that these techniques are very useful for computer game.

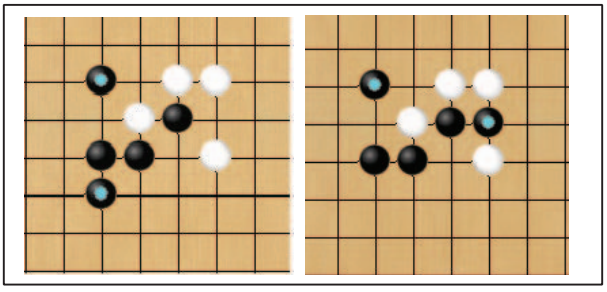

Figure 2. Contrast Experiment single depth VS extension depth

Table1. Contrast Experiment

Experiment Item	Original version VS New version
Average Using Time	1'25 VS 3'28
Winning rate	0% VS 100%

6 CONCLUSION

Some techniques were discussed in this paper. A game system of Connect-six, for example, was formed. In this game system, some techniques such as Alpha-Beta pruning and Hash table technique were used. And the correctness and efficiency of the techniques had been proved by a series of tests and competitions. By these researches, a high-level game system can be produced. In the same way, the results and conclusions of this paper also have some reference values to other Computer Game procedures.

REFERENCES

[1] H.van den Herik, J.Uiterwijk, and J.van Rijswijck. Games solved: Now and in the future. Artificial Intelligence, 2002, 134:277-311.

[2] M. Campbell, A. Hoane, and F.-H. Hsu. Deep blue. Artificial Intelligence, 2002, 134:57-83.

[3] J.Schaeffer, Y.Bjornsson, N.Burch, A.Kishimoto, M.Muller, R.Lake, P.Lu, and S.Sutphen. Solving checkers. International Joint Conference on Artificial Intelligence, 2005, pp. 292-297.

[4] Chang-ming Xu, Z.M. Ma, Xin-he Xu. A Method to Construct Knowledge Table-Base in k-in-a-row Games [C]. ACM Symposium on Applied Computing, 2009: 929-933.

[5] Xin-he Xu, Jiao Wang. Key Technologies Analysis of Chinese Chess Comprter Game. Journal of Chinese Computer Systems. 2006, 6.

[6] Shang-bin Jiao, Lin Ding. Research on Translation Table Heuristic Algorithm. Computer Engineering and Applications. 2010, 6.

[7] Akihiro Kishimoto, Jonathan Schaeffer. Distributed Game-Tree Search Using Transposition Table Driven Work Scheduling. Proceedings of 2002 International Conference on Parallel Processing. 2002, 8.

[8] Alexander Reinefeld. A Minimax Algorithm Faster than Alpha-Beta. 7th Advances in Computer Chess Conference, 1993.

[9] YANG Xiao-bin, LI Wen. Alpha-Beta-Based Optimized Game Tree Search Algorithm. 3rd International Conference on Computational Intelligence and Industrial Application, 2010.

[10] Joel Veness, Alan Blair. Effective Use of Transposition Table in Stochastic Game Tree Search. IEEE symposium on Computational Intelligence and Games. 2007, 4.