Improved alpha-beta pruning of heuristic search in game-playing tree

ZHANG-Congpin CUI-Jinling
（Key Laboratory for Intelligent Information Processing, College of Computer and Information Technology, Henan Normal University, Xinxiang 453007）

Abstract

The game playing is an import domain of heuristic search, and its procedure is represented by a special and/or tree. Alpha-beta pruning is always used for problem solving by searching the game-playing-tree. In this paper, the plan which child nodes are inserted into game-playing-tree from large value of estimation function to small one when the node of no receiving fixed ply depth is expand is proposed based on alpha-beta pruning. It improves effect of search.

1. Introduction

Game playing is a kind of competitive actions with intellect such as chess, tic-tac-toe in which the two players alternate moves. It is an important domain of heuristic search. The simplest game playing is two-person game of perfect information which the two player alternate moves, and it has the property that both players have perfect information about the location of the pieces on the board, the result of game playing is a tie or one person wins and another loses. In this paper, the object of research is two-person game of perfect information, and it provides reference for complicated problem of game playing.

In a two-person game of perfect information, each player in turn has to determine a next move. This next-move decision requires thinking about how your opponent will move in response, how you will respond to their response, and so on.

The procedure of game playing is represented by a special and/or tree--game playing tree.

2 Game playing

A game playing tree is a tree consisting of nodes that represent options for the players to move in a two-person game. The nodes implicitly represent states of the game. The state resulting from a move is completely determined by the move and the state immediately prior to the move. It has three characteristics. The first, the root of a game playing tree represents the initial state of the game. The second, and-nodes and or-nodes arise by turns. One player of two-person game of perfect information is called side A, and the other is called side B. Two sides both have many options in a move, the options are children nodes. According to standpoint of the side A, the relation of the side A's options is or, because the side A can decide which children node is chosen. On the contrary, the relation of the side B's options is and, because the side A has no decision, and the side A must consider every options of the side B, especially, the worst state to the side A. The third, the standpoint of one-side is taken from beginning to end in the whole procedure of game playing, such the side A's viewpoint. Terminal nodes indicate moves that result in the side A of the two-person winning the game are solution nodes, the terminal nodes indicate moves that result in adversary(the side B) winning the game are non solution nodes[1].

In order to solve problem the search techniques are used for two-person game of perfect information. Normally a part of game playing tree is produced. The search techniques normally include blind search and heuristic search, especially heuristic search is always used. Heuristic search for two-person game of perfect information includes MINIMAX and alpha-beta pruning.

In MINIMAX search terminal nodes indicate moves that result in one of the two players winning the game and assigned a value of either 1 or -1 depending on which player wins. If the side A is represented, the objective of the side A is to reach a node with value 1, and objective of adversary is to reach a node with value -1. If it is possible for a game to end in a draw, we assign nodes resulting in a draw with a value of 0. In order to choose a next move, we need to evaluate the consequences of every option, including those corresponding to internal nodes in the game playing tree. When the side A chooses the next move, it will choose the biggest value of its children nodes, the side B is contrary. This procedure is impractical because the game playing tree for the most games is extremely large [2]. And alpha-beta pruning is widely applied.

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.527

3. Alpha-beta pruning

Although it is generally difficult to determine exactly far ahead to search, there are some cases in which, given the nodes that are already evaluated, there is no use in extending the search at other nodes. Consider the partially expanded game playing tree shown in Figure 1. Nodes corresponding to options of the side A are shown as squares, and nodes corresponding to options of the side B are shown as circles. Note that, if the side A chooses the F, it can do no better then 1. The side A will choose E over F no matter what it learns in expanding L and M, so it might prune away the children nodes of F. As well, Figure 1 illustrates another case in which we can prune the search space. At node G, the side B has option with value 7. In this case, there is no need to expand D since the side B can always choose C (the value 2) and do better than 7.

The general case for pruning a node is as follows. Suppose that n is a node at which the side A gets to move and that n has a sibling with a value v. Suppose further that there is some node m on the path from the root of the tree to n such that the side B gets to move and m has a sibling with a value greater than v. In this case, the node n can be pruned from the search space. A similar argument works for nodes at which the side B gets to move.

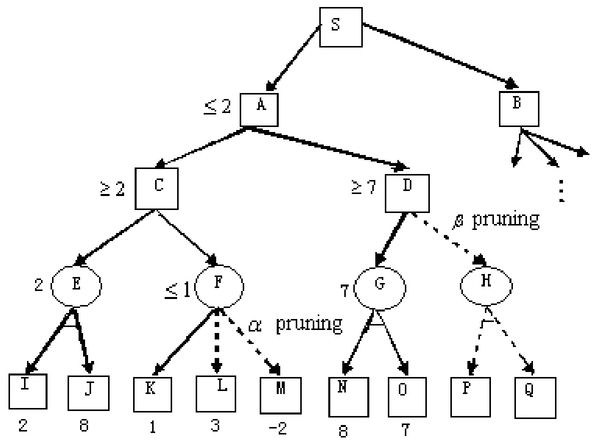

Figure 1 alpha-beta pruning

To implement this pruning strategy, it is necessary to keep track of the current best estimated value for a node even though the sub-tree rooted at that node has only partially been explored. When we encounter a new node, if it is a terminal node or we have decided not to expand the tree further, then we apply the evaluation function and assign the resulting value to the node. How much of a reduction of search space is actually realized depends a lot on how the children of a node are ordered in the list of nodes to consider next. In order to prune effectively, we have to consider the children order of node when the node is expanded [3].

4. Improved alpha-beta pruning

It is obvious that the size of the search space is related to the estimated value for the first child of nodes. The estimated value for the root of the tree will be equal to value of a terminal node which is the best node. In the procedure of backing up the value at a root node, if the first searching node is the best node, the size of the search space will be reduced and the effectiveness of search will be improved mostly. In the best case, and-node firstly expands the smallest estimated value of the children, and or-node firstly expands the largest estimated value of the children. In this case, alpha-beta search far enough ahead, let the depth of the tree is d, the number of node is a half number of MINIMAX search. In the worst case, alpha-beta search is the same as MINIMAX. How much of a reduction is actually realized depends on how the children of a node are ordered in the list of nodes to consider next. In order to improve effectiveness of search it is a method to adjust the order of the children of a node [4].

In the procedure of alpha-beta search, nodes are expanded by deep-first search of the game playing tree until the depth of d, and then the estimated values of terminal nodes are computed. In the paper, children of the node which is being expanded are ordered by their estimated value. Here is a complete algorithm for being improved the pruning method.

1 Set N to be the list consisting of the single element, m.

2 Let n be the first node in N.

3 If n=m and n has been assigned a value, then exit returning this value.

4 Try to prune n as follows. If n is a or-node, let v be the minimum of the values of sibling of n, and u be the maximum of the values of sibling of ancestors of n that are minimizing nodes. If v≥u, then you can remove n and its siblings and any successors of n and its siblings from N. if m is an and-node, then proceed similarly switching min for max, max for min, and ≤ for ≥.

5 If n cannot be prune, then if n is a terminal node or we decide not to expand n, assign n the value determined by the evaluation function and back up the value at n.

6 Otherwise, remove n from N, and assign the children initial values of n, add the nodes to the front of N. If the node n is and-node, add the nodes from large value to small one. If the node n is and-node, add the nodes from small value to large one.

7 Return to step 2.

In addition, the best state is stored for the next search.

Here is tic-tac-toe to which improved alpha-beta search is applied. Figure 2 illustrates tic-tac-toe, has 9 spaces, in which the two players alternate moves. One who marks firstly on the grid in a line will win. The mark of the side A is ×, another is O.

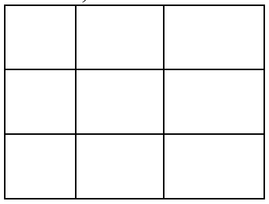
Figure 2 tic-tac-toe

We need an evaluation function, e(p), which when applied to a state p returns an estimate of its value. The evaluation function is e(+p)-e(-p), e(+p)is the number of × in a line, e(-p) is the number of O in a line. If p lead to a win for the side A with certainty, then e(p)=+∝. If p lead to a win for the side B with certainty, then e(p)=-∝.

Here is tic-tac-toe to which improved alpha-beta search is applied. Figure 3 shows the game playing tree in which the nodes of MINIMAX search are shown when depth of search is 2.

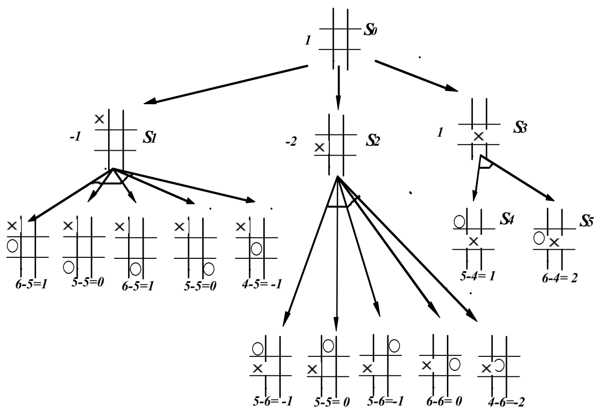

Figure 3 nodes of MINIMAX search of tic-tac-toe

Table 1 lists respective result of different strategy for search.

Table 1

search technique	number of nodes (search depth=2)	number of nodes (search depth=4)
MINIMAX	16	462
α-β pruning	12	121
Improved α-β pruning	7	50

With depth going deeper, improved α-β search will prune away more unnecessary nodes. Although time is spent in ordering the nodes, time in searching is cut down mostly.

5. Conclusions

In this paper, we study an improved α-β pruning technology. In the procedure of improved alpha-beta search, the estimated values of nodes are computed, while nodes are expanded. Based on the estimated values, the nodes are inserted into the game playing tree. The experiment shows that the deeper the tree is, the more nodes are cut down.

6. Acknowledgement

The work described in this paper was jointly supported by key young teacher plan project of Henan province and scientific and technological challenges project of Henan province (2007520026).

7. References

[1] P.R.Cohen, E.A.Feigenbaum. The Handbook of Artificial Intelligence [M]. 1982(3):45-80

[2] Clancy W.J. Heuristic Classification. Artificial Intelligence [J]. 1985(27): 289-350

[3] George F.luger. Artificial Intelligence Structures and Strategies for Complex Problem Solving Fifth edition [M]. BeiJing: China Machine Press，2006

[4] Thomas Dean, James Allen, Yiannis Aloimonos. Artificial Intelligence: Theory and Practice[M]. BeiJing：Publishing House of Electronics Industry，2003