AN ENHANCEMENT TO THE ITERATIVE, ALPHA-BETA, MINIMAX SEARCH PROCEDURE

by William Fink, author of the Sfinks chess programs

The greatest problem facing any chess program employing a minimax search procedure is the problem of the exponential growth in the number of board positions to be evaluated.

Alpha-beta pruning has been the primary method of combating this problem without the loss of accuracy. For some time it has been known that the efficiency of alpha-beta pruning depended upon the order in which moves were searched. If the best moves were searched first, the maximum number of cut-offs would occur. One method of causing the better moves to be searched first was to employ an iterative search and on each succeeding iteration, first search the moves from the principal variation (best line of play) from the preceding iteration.

The enhancement, about to be described, is a more general application of the preceding idea. Simply stated, the idea is to remember the principal variation for each legal move and search it first on the next iteration. (To distinguish these variations from the best line of play over the set of all legal moves, they will be referred to as alternate variations.) This idea was first suggested to me by Charles Heath. He used it in his reversi program.[1] I have since seen the idea briefly mentioned in Advances in Computers.[2] It was very easy to add this enhancement to my chess program, and it gave an improvement in search times of up to 15%.

The Implementation

My current program, Sfinks 3.0, uses a full-width iterative minimax search procedure and employs alpha-beta pruning. On the first ply, all the moves are generated and then sorted in a list. Eight bytes of storage are used for each legal move: four for the representation of the move, two for a link to the next move in the sorted list, and two for the move's value. To implement a version of the idea, it was necessary to double the amount of memory used by the move list. That is, for every move, eight additional bytes were set aside. In the first four additional bytes per move, the "best" response from ply 2 to the move on ply 1 was saved. In the remaining four additional bytes, the "best" reply from ply 3 to the move from ply 2 was saved. In other words, the best variation for each legal move was stored, but only for two additional plies.

1. Master Reversi for 32K Model I TRS-80 from INSTANT SOFTWARE, INC.

2. "Recent Progress in Computer Chess" by M. M. Newborn, 1979, in Advances in Computers, Vol. 18, pp. 59-117, Academic Press

There were two problems to be solved: 1) where to find the moves of the alternate variations so that they could be saved, and 2) when to play and search these moves for an evaluation.

The moves to be saved were found in two places during a tree search. Whenever the principal variation (best line of play over the set of all legal moves) was found and saved at ply depth of 2, the moves from plies 2 and 3 could be saved directly into the space provided for the alternate variation for the move being considered on ply 1. Also, whenever an alpha-beta cut-off occurred and the depth of the search was 2, either one (from ply 2) or two moves (from plies 2 and 3) were saved, depending upon the maximum depth of the iteration.

It was easier to determine when to play and search the moves saved. When checking to see if a move from the principal variation should be played, the list of best values for each ply is scanned from the value for current depth -1 down to the value for depth 1 to see if all values are initial values ($\pm\infty$). If they are, a move from the principal variation is played and searched. If they are all initial, except for the value for ply 1, and the depth of the search is 2 or 3, the appropriate move is fetched from the alternate variation to be searched.*

The implementation just described is one that effectively enhances the alpha-beta cut-offs in the Sfinks chess program which runs on a 32K TRS-80 microcomputer. This or other versions of the idea might be tried on different computers. If the host computer has a lot of memory and the entire tree is saved for the first couple of plies, the idea can most likely be implemented at a greater depth with a beneficial effect, also.

*In Sfinks 3.0, the moves on the deeper plies are generated in stages and duplication of moves to be searched sometimes occurs (i.e., when the expected alpha-beta cut-off fails). This causes only a relatively insignificant slow down and can be ignored.