Game search trees

Eyal Amir

January 5, 1996

For the time being, this document will describe the more 'exotic' search techniques.

1 SSS*

The source for the following algorithm is found in [Stockman].

1.1 Preliminaries

Definition 1.1 *An* AND/OR tree *G is a (non-empty) tree where all immediate successors of a node in the tree are of the same type, AND or OR. The ype of the root node may be either AND or OR.*

Definition 1.2 *A* solution tree *T of an AND/OR treee G is a tree with the following characteristics:*

1. *The root node of the AND/OR tree G is the root node of the solution tree T.*
2. *If a nonterminal node of G is in T, then all of its immediate successors are in T, if they are of type AND, and exactly one of its immediate successors is in T if they are of type OR.*
3. *All the terminal nodes of T represent "solved problems".*

Definition 1.3 *If T is a soltion tree of AND/OR tree G rooted in node p, the* value of *T if denoted as $f_T(p)$, and is defines as the minimum value of all terminal nodes in T.*

The best solution trees will be those whose value is maximum over the set of all possible solution trees. This maximum value is the same as that computed by minimax definitions and procedures.

Definition 1.4 *Let G be an AND/OR tree with set of terminals V_T. A* minimax evaluation on the nodes n of G *is a real-valued function denoted as $g(n)$ or $g_G(n)$ and defined as follows.*

1. *If n has immediate successors of type OR:*

$$g(n) = max\{g(n_i)\}$$

   for all immediate successors n_i of n.

2. *If n has immediate successors of type AND:*

$$g(n) = min\{g(n_i)\}$$

   for all immediate successors n_i of n.

3. *If n is terminal:*

$$g(n) = f(n)$$

   where f is a real-valued "static evaluation function"

Theorem 1.5 *Let $T(p)$ be a solution tree rooted at a node p of an AND/OR tree and let $g(p)$ be the minimax ecaluation of p. Then*

$$g(p) \geq f_T(p)$$

Moreover $g(p) = f_{T_0}(p)$ for some solution tree T_0.

1.2 The Algorithm's main idea

The SSS* is a Best-First search procedure that maintains upper bounds on the values partially developed candidates strategies, selecting the most promissing ones for further development. A strategy specifies one response, of the player to each of the opponent's move. The development process continnues until one game strategy is fully developed, at which point it represents the optimal strategy.

1.3 Computational Aspects

The SSS* is optimal interms of avarage number of nodes examined; but its superior pruning power is more than offset by the substantial storage space and bookkeeping required.

2 B*

The source for the following algorithm is found in [Berliner].

2.1 The Algorithm's main idea

Instead of one heuristic function and one calue per node, we have upper and lower bounds on each node, as well as an upper-bound heuristic function (and a lower-bound heuristic function. Now, cdeciding on a successor can be done in one fof the twoways (or a combination of the two):

1. Taking the node with the best upper-boundmm aand trying to increase the lower bound, so that the lower bound of it, is better than the upper-bound of all otherpossible successors.

2. Taking the node with the second-best upper-bound and try to decrease its upper-bound, so that it will be less than the best-node (with the best upper-bound).

The way to do the decision of How to choose the strategy is not specified, but some options are mentioned:

D If $\Sigma_{i=2}^{b} d_i^2 < d_1^2$ (where d_i is the depth from which the optim bound was backed-up to successor i, and n_1 is the best-node) Then DISPROVER-EST Else PROVEBEST

R The criterion info above is divided by the range of the node.

In other words, the D criterion favours subtrees that have not yet been explored deeply, and the R criterion favours searching nodes with larger ranges.

2.2 From the Author

2.2.1 Positive

1. No effort limits are needed.

2. The investigation goes only in directions that need to be furthely purused. The method used here is sometimes refered to as "forewaed-peuning" (pruning from the root).

3. The deepening is done (as SSS* does) while "jumping" from one side of the tree, to another, due to changes in values of examined nodes.

4. There is an option to exercise two search strategies instead of being forced to pursue the best alternative to a great depth.

2.2.2 Negative

1. Need to keep the expanded tree in memory

2. Sometimes difficult to find monotonic upper and lower bound funtions. In that case, errors will occur in the choice of the arc. It is argued by the author though, that such erros occur with other search techniques with inaccurate heuristic functions (assuming adequacy - if we stop the search before goal is reached).

Remark 2.1 *B* with only PROVEBEST strategy, and stopping the deepening only when upper=lower bounds at a node, is equivalent to A* with a mode of finding the best arc at the root, without the complete path itself.*

2.3 Computational Remarks

- The more parameters (alternatives) used in choosing the strategy, the better the result.

- For search trees ≥ 1000 we expand $0.32 \times \#nodes$ searched by Best-First Search.

References

[Stockman] G. C. Stockman, *A Minimax Algorithm Better than Alpha-Beta?*, Artificial Intelligence 12 (1979), 179-196.

[Berliner] H. Berliner, *The B* Tree Search Algorithm: A Best-First Proof Procedure*, Artificial Intelligence 12 (1979), 23-40.

This is /u/eyal/ongo/quals/games.tex.