

Re-searching from Shallow Depths to Order Moves Saves Leaves Only When Shallow Values Track Deep Ones

— iterative deepening that orders children by the previous search's values reads fewer leaves than a single search in random order, but on trees where shallow and deep values are unrelated it reads more —the separator is whether shallow values predict the order of moves at depth — [Paper 951]

Yuuki YAMAGISHI (山岸 勇輝)

Yamagishi Complexity Torus Lab, Japan

September 2026

YOI: <https://www.yamagishiyuuki.com/yoi/951>

— *From thinking to syncing.* —

Abstract

The number of leaves alpha-beta search reads changes greatly with the order of children, but without knowing the true values one cannot use the best order. This paper compares the total leaves evaluated by iterative deepening, which re-searches from depth 1 upward and orders children by the values returned by the previous search, with a single search in random order and with the formula for the best order, on trees whose position scores are related between shallow and deep searches and on trees whose scores are not. No new theorem or law is claimed. First, on trees where shallow and deep values are related, re-searching reads fewer leaves: when a child's score is its parent's score plus a random number, the total leaves of iterative deepening were 0.547664 times those of a single random-order search for branching 4 and depth 6, 0.408755 times for branching 4 and depth 8, and 0.437245 times for branching 6 and depth 6 (Section 2). Second, on trees where they are unrelated, re-searching reads more leaves: when position scores are independent random numbers at each depth, the total leaves of iterative deepening were 1.507492, 1.729632 and 1.378168 times those of a single random-order search (Section 2). Third, on related trees, the ordering at the last depth comes close to the best order: leaves read at the last depth alone were 1.829396, 1.946184 and 2.235112 times the best-order formula, against 6.961942, 15.300718 and 13.563032 times on unrelated trees (Section 3). Fourth, every method gives the same root value: for all three sizes and both kinds of tree, the root values of iterative deepening, random order and best order matched full minimax, and the leaves of the best order matched the Knuth-Moore formula (127, 511 and 431) (Section 2). Fifth, the separator is whether shallow values predict the order of moves at depth: re-searching costs the leaves of the shallow searches, but if the ordering is right, more branches are cut in the deep search and the cost is recovered. If it is not right, only the cost remains (Section 4). **Keywords:** iterative deepening, alpha-beta search, move ordering, game tree, leaves read, related evaluations.

Scope of this paper (scope note): No new theorem or law is claimed: alpha-beta search, the formula for leaves read in the best order (Knuth and Moore), iterative deepening that re-searches with increasing depth, and ordering moves by the results of the previous search are all known. These are set trees: three uniform trees, branching 4 and depth 6, branching 4 and depth 8, and branching 6 and depth 6, with 6 random trees each. There are only two ways of making position scores, and they correspond to no real game's evaluation. Other techniques are not included: reuse of identical positions (transposition tables), narrowed search windows and time management are not included. Ordering uses only the values returned by the previous search (bounds for cut-off children). Speed is not discussed. What is counted is the number of leaves evaluated; actual time and program strength are not treated. Relation to earlier papers: Paper 926, in its scope note, did not treat actual game programs, leaving out evaluation functions, iterative deepening, transposition tables and move-ordering techniques themselves; this paper sits in the seats of iterative deepening and move ordering (no earlier occupant; transposition tables are not entered). Paper 926 showed that the number of leaves alpha-beta search reads ranges, with the order of children, from over eight tenths of all to the order of their square root. Here it is counted how close to the best order one gets when the ordering is built from shallow searches without knowing the true values. What is added: setting the total leaves of iterative deepening beside random and best order, showing that gain and loss switch between related and unrelated trees, comparing the leaves at the last depth alone with the best order, confirming that root values agree, and placing the separator on "whether shallow values predict the order of moves at depth".

1. Notation and the Question

Trees: uniform trees with branching b and depth d . Position scores are made in two ways. (1) Related trees: a child's score is its parent's score plus a standard normal random number (shallow and deep scores are related). (2)

Unrelated trees: independent standard normal random numbers at each depth.

Search methods: (1) Random order: children in random order, one alpha-beta search to depth d . (2) Iterative deepening: re-search at depths 1, 2 and so on up to d , ordering children at each position by the values returned in the previous search, largest first (random order for positions not searched before). (3) Best order: order by the true values from a full search and search once. What is counted is the number of leaves evaluated; for iterative deepening it is the total over all depths. The leaves of the best order follow the Knuth-Moore formula $b^{\lceil d/2 \rceil} + b^{\lfloor d/2 \rfloor} - 1$.

Question: if the ordering is built from shallow searches without knowing the true values, are leaves saved even after paying for the re-searches?

2. Leaves Read

Computation 2.1: mean over 6 trees.

branching, depth, tree	iterative deepening total	random order once	best order (formula)	deepening/random	deepening/best
4, 6, related	439.5	802.5	127	0.547664	3.460630
4, 6, unrelated	1425.3	945.5	127	1.507492	11.223097
4, 8, related	1862.8	4557.3	511	0.408755	3.645466
4, 8, unrelated	12072.8	6980.0	511	1.729632	23.625897
6, 6, related	1586.8	3629.2	431	0.437245	3.681748
6, 6, unrelated	7975.0	5786.7	431	1.378168	18.503480

For all six combinations the root values of the three methods matched full minimax, and the leaves of the best order matched the formula.

Observation 2.2

on related trees re-searching reads fewer leaves, on unrelated trees more. For all three sizes, the total of iterative deepening on related trees was below a single random-order search: on unrelated trees all three were above it, 1.729632 times at branching 4 and depth 8. The same method was a gain or a loss depending on how the tree was made.

3. Ordering at the Last Depth

Computation 3.1: leaves evaluated at the last depth of iterative deepening alone.

branching, depth	related tree (ratio to best order)	unrelated tree (ratio)
4, 6	232.3 (1.829396)	884.2 (6.961942)
4, 8	994.5 (1.946184)	7818.7 (15.300718)
6, 6	963.3 (2.235112)	5845.7 (13.563032)

Observation 3.2

on related trees the ordering built from shallow searches comes to about twice the best order. At the last depth alone, related trees read 1.829396 to 2.235112 times the leaves of the best order: unrelated trees read 6.961942 to 15.300718 times.

4. The Separator

Observation 4.1

the separator is whether shallow values predict the order of moves at depth. Iterative deepening adds the cost of reading leaves at shallow depths: on related trees, the ordering built from shallow searches brought the leaves at the last depth down to 1.829396 to 2.235112 times the best order (Computation 3.1), and the cost was recovered. On unrelated trees the ordering did not hit, and only the added cost remained.

Observation 4.2

what is not said. Which kind of tree real game evaluations resemble, counts with transposition tables or narrowed windows, and program speed are not treated. What is shown is that on the two set kinds of tree, the gain and loss of iterative deepening switch with whether shallow and deep searches are related.

5. Roll Call and Closing

1. two kinds of tree and three search methods (Section 1): **related or unrelated**.
2. down to **0.408755** of random order on related trees (Computation 2.1): **re-searching reads fewer**.
3. up to **1.729632** times on unrelated trees (Computation 2.1): **re-searching reads more**.
4. last depth 1.829396 to 2.235112 times best order (Computation 3.1): **the ordering hits**.
5. root values and best-order formula agree (Computation 2.1): **the method does not change the answer**.
6. whether shallow searches predict move order (Observation 4.1): **the separator; the scope note of Paper 926**.

Closing: Re-searching from shallow depths to order moves saves leaves only when shallow values track deep ones. When a child's score is its parent's plus a random number, the total leaves of iterative deepening were 0.408755 to 0.547664 times those of a single random-order search, and the ordering at the last depth came to about twice the

best order: with independent random numbers at each depth, iterative deepening read more, 1.378168 to 1.729632 times. The separator is whether shallow values predict the order of moves at depth. Placed among the earlier papers: the effect of ordering that Paper 926 counted with and without knowing the true values was counted as iterative deepening, which builds its ordering from shallow searches. To be honest, which kind real game evaluations resemble is not claimed. That re-searching can be a gain or a loss depending on whether shallow searches hit, and no further.

References

1. Knuth, D. E., & Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*.
 2. Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*.
 3. Yamagishi, Y. (2026). *The Number of Leaves Alpha-Beta Search Reads Ranges, with the Order of Children, from Over Eight Tenths of All to the Order of Their Square Root* [Paper 926]. YOI: <https://www.yamagishiyuuki.com/yoi/926>
-

— *From thinking to syncing.* —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)