SEARCHING TO VARIABLE DEPTH IN COMPUTER CHESS

Hermann KaindJ
Rennweg 43/3/17
1030 Wien
Austria

ABSTRACT

This paper discusses some methods for guiding the search of conventional chess programs to variable depth. The motivation for investigating such methods comes from the fact that searching to a fixed depth causes difficult problems (e.g. the horizon effect). The first section deals with certain improvements of the quiescence search and a demonstration of their beneficial effects. The method of not counting moves as a ply of depth is investigated then and the results of extending it somewhat are reported. As this method seems to be too simple nevertheless, a more general model tor extending the horizon of the full-width search to variable depth is proposed.

I INTRODUCTION

Although most of today's better chess programs perform a simple quiescence search and do not count replies to check as a ply of depth there is a fundamental philosophy to search every branch to the same depth. The main reason for this lies in the tact that in chess it is very difficult to discriminate good moves from bad ones statically without error. Moreover it has been shown consistantly, that programs which perform forward pruning are inferior. They often prune moves at a lower level which seem to be bad, but would prove good, deeper in the tree (e.g. sacrifices). On the contrary, brute-force programs can discover everything within their horizon as their search is full-width. A thorough discussion of advantages and disadvantages of this memethod was given by Berliner in 1981 *[4].*

A principal problem when searching to a fixed depth is the horizon effect (I 2J, [3] as it is extremely difficult to overcome it by pure static analysis. Actually in deep searches this effect does not influence the played move very often [4] but nevertheless it should be worth trying to avoid it further by investigating important variations more thoroughly ([10], [5]).

In despite of the enormous speed of the special chess automaton BELLE [6] the combinatorial explosion of full-width searches should not be forgotten completely. It seems to be very difficult to build a machine which will be significantly faster than BELLE. Thus it also seems to be very difficult to further strengthen chess automata this way. Therefore it should not be disadvantageous to make the search more efficient guiding it to variable depth.

II QUIESCENCE SEARCH

A usual method for searching to variable depth is the so-called quiescence search: a selective search is performed sprouting from certain positions (mostly at the horizon of a full-width search) to reach quiescent (dead) positions which can be evaluated more accurately. Its main purpose is to avoid the horizon effect and sometimes it also can find deeper combinations. Indeed most of today's successful performance programs employ a quiescence search, but a severely limited one. Mostly only captures and replies to check are considered and sometimes promoting and checking moves.

A model for a more informed (than usual) quiescence search is given elsewhere [18], [91). On the implementation of this model into our brute-force chess program MERLIN and the control of this search together with results it is reported in [8], | 10]. The exceptional features of this quiescence search are:
- the consideration of hung pieces which are subject to capture by the side not on move (moves by such pieces and protective moves);
- the killer quiescence search which tries killer moves from the full-width search beyond its horizon;
- the check quiescence search which only tries very forcing checks.

Unfortunately it is not always possible to achieve a dynamic value using this model. E.g., let us assume a position with a hung piece which can"neither be moved nor protected. Assuming further that the quiescence search cannot select any move, no value or a pessimistically estimated claim value (|3], |J0j) would result. To estimate such a situation realistically by a static analysis is also risky. Therefore MERLIN tries a null move there letting the opponent capture immediately. This way quiescent positions may result which can be evaluated more accurately. Nevertheless the resulting value for the position in question can itself be clearly wrong, but experiments have shown that this occurs seldom.

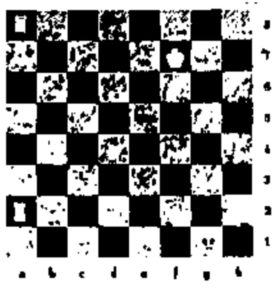
Figure 1
White to Play

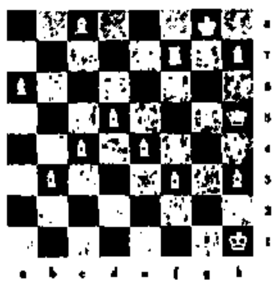
Figure 2
White to Play

Of course there are also other threats to be worried about. MERLIN considers threats of pawn promotion or mate in the following way: A simple static analysis decides, if there could be such a threat, and eventually a null move analysis is activated to investigate it further.

The position of Figure l ([3], Figure 4.8) seems to be easy (at least for human chess players). But e.g. after l.Ra8-h8 Ra2-al + 2.Kgl-f2 Ral*a7 the correct move 3.Rh8-h7 + winning the rook is not easy to discover within the quiescence search of a brute-force chess program (if not trying all the checking moves there). By further checks with the black rook this effect can last tor several ply. MERLIN avoids it using the killer quiescence search in the following way, when searching to a full depth of 3 ply: After l.Ra8-h8 Ra2*a7 2.Rh8-h7 the black rook is won. Thus Rh8-h7 is stored as a killer move and tried again after e.g. l.Ra8-h8 Ra2-al 2.Kg 1-f 2 Ral*a7, winning the black rook.

Let us assume now, that alter e.g. l.Ra8-h8 Ra2-a1 + 2.Kgl-f2 Ral-a2, 3. Kf2-c3 the quiescence search is reached. There Black to move can (according to the usual scheme) decide to accept the static value, or try to better it by certain moves. Let us assume further, that the moves result in correct values (e.g. 3...Ra2*a7 4.Rh8-h7t killer quiescence search). But the usual static evaluation functions do not account for the threat of pawn promotion. Thus the value of the whole variation would be accidental.

MERLIN recognizes in this position the opponent's (White's) opportunity to promote the pawn from a7. It asks the search, if this promotion is really a threat by trying a null move. Essentially this means, that the side to move is changed and then only tried to promote the pawn from a7. As there the threat is recognized to be real, MERLIN refuses to evaluate the position in question statically. Instead it tries to find counter moves (e. g. to add control to the promoting square or to block it) but cannot find one there. So MERLIN plays l.Ra8-h8 alter a 3 ply search, announcing the main variation l...Ra2*a7 2.Rh8-h7+ Kf7-e6 3.Rh7*Ra7.

(II MOVES NOT COUNTING AS A FLY OF DEPTH

The best chess programs now do not count responses to check as a ply of depth. This method actually makes the horizon of the full-width search variable. It also entails only a small risk of explosion, as there are few legal moves in positions with the king checked. MERLIN improves this method somewhat applying it only to those moves whose optimistically estimated value is better than the actual best value reached so tar. The reason is that the extension of the search should be also worth its costs. In situations after a sacrifice this criterion also expresses a principle of compensation: Those moves should be investigated further where the side owning more material is forced by a check. Additionally MERLIN does not count certain very forcing checks.

In the position of Figure 2 ([12], Figure (4) mate in 11 ply is possible. Not counting replies to check a search to "only" 7 ply is sufficient. Additionally not counting certain checks MERLIN reaches the position after l.Qh5*h7 + Kg8*Qh7 2.Rg5-h5+ Kh7-g8 3.Nh4-g6 within a full-width 2 ply search. There it recognizes statically by use of simple criteria that the black king might be in danger. As mentioned before MERLIN activates a null move analysis in such situations. Here the result is that the rook actually threatens to mate the black king. Thus no static value is assigned and certain counter moves are tried (e.g. moves by the black rook from f7 to vacate an escape square tor the king). As these cannot prevent mate either, MERLIN claimed mate in 9 ply after 20.5 sees, of cpu time on a CDC Cyber 170/720 and searching 623 nodes (58% within the quiescence search). Strictly speaking this was not absolutely correct, as Black can delay mate for further 2 ply sacrificing his queen on g2 or h2 giving check. Normally MERLIN does not try such checks within its quiescence search as they Jose material. It is not clear if this would be an important flaw (e.g. PARADISE [12] does not even consider these moves). Nevertheless MERLIN was told then to try every check within its quiescence search if mate actually threatens. This helped to find the correct depth of mate (searching some few nodes more). As it also helps to make sure that there is no complicated side effect by a check, this method seems to be worth its costs.

To compare MERLIN's performance *on* this position with that of other programs, see Wilkins, 1980 [12] (It should be noted, that there exist computers which are 20 times or more faster than the one used by MERLIN.).

Despite of good results it seems that the method of not counting moves as a ply of depth is too simple for further extension. The main disadvantages are:
- a move can be counted or not only as a whole;
- this decision is made within the tree by use of little information;
- mostly this decision is only based on an estimate, how forced or forcing a move is.

But the idea in itsell to extend the horizon of the full-width search to variable depth seems to be very promising. This way it is possible to explore variations deeper without reducing the chance of discovery (compared to the quiescence search). This aspect seems to be important, as it often occurs in chess, that forced moves should be followed by one or more moves which cannot easily be detected statically by conventional programs. Thus a more general model for extending the horizon of the full-width search to variable depth seems to be necessary.

IV A MODEL FOR EXTENDING THE HORIZON

Knuth and Moore report on an interesting approach by Floyd in 1965 (see [11], loc cit.) to search to variable depth:
Each move is assigned a "likelihood" which is related to its forcedness and plausibility. When the product of all "likelihoods" leading to a position becomes less than a given threshold, this position is considered to be terminal and evaluated statically.

Such an approach seems up to now not have been tried in large-scale experiments. The reason is, that today's chess programs have not enough knowledge available to assign such values accurately enough to every move. The only exception may be PARADISE (for the subproblem of tactics) but when having much knowledge available, a best-first search seems to be more adequate than a depth-first search to variable depth (see also [3], [13]).

Consequently current performance programs should, only when there is a reason they know, search a branch

deeper, but in a subtler way than not counting certain moves. This should be possible as follows:
Iterative deepening of full-width searches is retained, but at the horizon of the full-width search (before beginning with the quiescence search) the decision is made, if the horizon is to be extended or not. This decision could be conferred to the following routine:

```
FUNCTION  EXTEND-HORIZON
          (FORCEDNESS (actual variation),
          INTEREST (actual position),
          EXTENSION) : BOOLEAN;
```

The FORCEDNESS of the actual variation could be measured according to Floyd's scheme. But then again the question arises, how to assign an appropriate "likelihood" to every move. The following scoring system seems to be more manageable in this respect:
A move which does not "count" is assigned a value of 0 (regarding to a "likelihood" of 1), moves which "count only half" are scored by 0.5 etc. The remaining moves which are not forcing or forced (or at least the program does not have knowledge about) get a score of 1. The sum of these values is subtracted from the number of ply in this variation resulting in a measure of its FORCEDNESS.

The inclusion of the INTEREST of the actual position as a parameter to EXTEND-HORIZON shall provide for the inclusion of the level of aspiration into the decision to extend the horizon or not. It is useless and expensive to search a forced variation deeper and deeper whose final value cannot influence the rest of the search. The level of aspiration is mostly represented by ALPHA and BETA in conventional programs [11]. So it seems appropriate to compare an estimate of the value of the actual position to these bounds. Unfortunately such values cannot be estimated very accurately sometimes (especially in situations after a sacrifice). Thus the INTEREST of the actual position should also include a notion of "compensation" (in the chess sense).

The parameter EXTENSION is to avoid that the horizon is extended over and over again in the same branch. Thus the actual extension (difference of the actual depth and the given search depth) should be included into this decision.

Although this model is not complete either it seems to remove the main disadvantages of simply not counting certain moves as a ply of depth:
- moves can be assigned also non-integer values, so that a smoother variability is possible;
- the decision is made at the latest possible moment (at the horizon itself) and therefore the whole variation can be analysed;
- the decision is based additionally on an estimate of the resulting position's value and consequently the search should become more effective and controllable.

V <u>CONCLUDING REMARKS</u>

The method of selecting a null move is not new (e.g. see [1]). But the use of it to achieve a dynamic value and to investigate a threat seems to be exceptional for brute-force programs. For a further discussion of this method see [13].

Extending the horizon of the full-width search aims at having the longest branches of the tree coincide with the variations most likely to be important. It can help against the horizon effect, too. The more knowledge the program has available the more variability of the search seems to be admissible.

For building better chess programs an enlargement of their positional chess knowledge seems to be even more important than an improvement of their search. So my colleague H.Horacek has built a general positional evaluation function which probably is one of the biggest and most detailed of today's programs for the whole game of chess. I myself have proposed a model for giving knowledge about positional long-range plans to chess programs which was also implemented for the example of the so-called "minority attack" ([7], [8]).

However a program of course is also slowed down by such supplements. Anyway an improvement of the search can result in the deeper exploration of interesting variations as well as in having more time available for each node to evaluate it more thoroughly.

REFERENCES

[I] Adelson-Velskiy, G.M., V.L.Arlazarov and M.V.Donskoy "Some Methods of Controlling the Tree Search in Chess Programs" <u>Artificial Intelligence</u> 6 (1975) 361-371.
[2] Berliner, H.J. "Some Necessary Conditions for a Master Chess Program" In <u>Proc. I3CAI-73</u>. Stanford, August, 1973, pp. 77-85.
[3] Berliner, H.3. "Chess as problem solving: The development of a tactics analyser". Ph.D.Dissertation. Comp. Science Dep., Carnegie-Mellon Univ. Pittsburgh, 1974.
[4] Berliner, H.3. "An Examination of Brute Force Intelligence" In Proc. I<u>JCAI-81</u>. Vancouver, August, 1981, pp. 581-587.
[5] Berliner, H.3., Personal communication, 3an. 5, 1983.
[6] Condon, 3.H. and K.Thompson, "Belle Chess Hardware" In <u>Advances in Computer Chess 3</u> (ed.M.R.B.Clarke), Pergamon Press, 1982.
[7] Kaindl, H. "Positional long-range planning in computer chess" In <u>Advances in Computer Chess 3</u> (ed.M.R.B. Clarke), Pergamon Press, 1982.
[8] Kaindl, H. "Realisierung von langfristigem Planen und Massnahmen gegen den Horizont-Effekt im Computer-Schach". Doctoral Dissertation. TU of Vienna, 1981.
[9] Kaindl, H. "Quiescence Search in Computer Chess" <u>SIGART Newsletter</u> 80 (1982) 124-131.
[10] Kaindl, H. "Dynamic Control of the Quiescence Search in Computer Chess" In <u>Proc. EMCSR-82</u>. Vienna, Austria, April, 1982, pp. 973-978.
[II] Knuth, D.E. and R.W.Moore, "An Analysis of Alpha-Beta Pruning" <u>Artificial Intelligence</u> 6 (1975) 293-326.
[12] Wilkins, D. "Using Patterns and Plans in Chess" <u>Artificial Intelligence</u> 14 (1980) 165-203.
[13] Wilkins, D. "Using Knowledge to Control Tree Searching" <u>Artificial Intelligence</u> 18 (1982) 1-51.