Minimaxing
Theory and Practice

Hermann Kaindl

Empirical evidence suggests that searching deeper in game trees using the minimax propagation rule usually improves the quality of decisions significantly. However, despite many recent theoretical analyses of the effects of minimax look-ahead, however, this phenomenon has still not been convincingly explained. Instead, much attention has been given to so-called pathological behavior, which occurs under certain assumptions. This article supports the view that pathology is a direct result of these underlying theoretical assumptions. Pathology does not occur in practice, because these assumptions do not apply in realistic domains. The article presents several arguments in favor of minimaxing and focuses attention on the gap between their analytical formulation and their practical meaning. A new model is presented based on the strict separation of static and dynamic aspects in practical programs. finally, certain methods of improving minimax look-ahead are discussed, drawing on insights gained from this research.

Existing models for analyzing properties of minimaxing seem to have been designed primarily to support formal analysis, rather than the proper modeling of the games for which minimaxing has proved very successful in practice. The discovery and analysis of pathological game trees, for which minimaxing does not work, is important but seems to have no correspondence to practical observations. Conjectures about practice based on models with assumptions that do not apply are rather dangerous.

This article takes the opposite approach to understanding the benefits of minimaxing. After summarizing existing models according to their practical relevance, we present a new model that was developed primarily from observations. This model seems to capture the essential features of the method, but introduces new complications, despite efforts to simplify it as much as possible. We conclude with a discussion of the consequences of such investigations for the proper use of minimaxing.

Background

First, let us briefly review the concept of minimaxing. Most current computer programs for two-person, perfect-information games (including chess, checkers, and kalah) use minimaxing as their basic method for choosing a move from a given position. While such games could be completely solved in principle using game theory, the combinatorial explosion inherent in "interesting" games makes this totally infeasible within any practical time and space limits. Hence, in contrast to the usual attempt in one-person games (problems) to search for a real goal-node, the usual paradigm here is a bounded look-ahead to artificial terminal nodes.

Except in very rare cases such as the game nim, there is no known way of directly evaluating the exact status (the true value) of such artificial terminal nodes. Consequently the use of heuristic estimates is usually the only practical resort. Such heuristic values are assigned by so-called *static evaluation functions*. In practical applications these heuristic values are usually not restricted to the game-theoretic values, but range over an interval of integer values. Theoretically, real numbers can also be used. While there is no satisfactory theory of the actual semantics of these numbers, it is intuitively clear that they should induce a partial order in the various positions of a game according to their worth in the sense of "goodness" or "strength" for one side. Conceptually it is rather problematic to map all considerations about a position into a single number, especially as there is no measure of its reliability. Nevertheless, up to now this simple scheme has been superior to more complicated ones in game-playing practice.

Under the assumption that the look-ahead tree (graph) of moves for both sides is searched deeper than one *ply* (the technical term for a half-move, or move by one side), the natural question arises as to what to do with the heuristic values of the terminal nodes. More precisely, how can they be used for making a reasonable move decision? When exactly one ply is searched, it seems clear that the best choice is the move resulting in the successor position with the best value (the maximal one, assuming that higher values are better for the side that is moving). When there are several such moves, one of them may be chosen

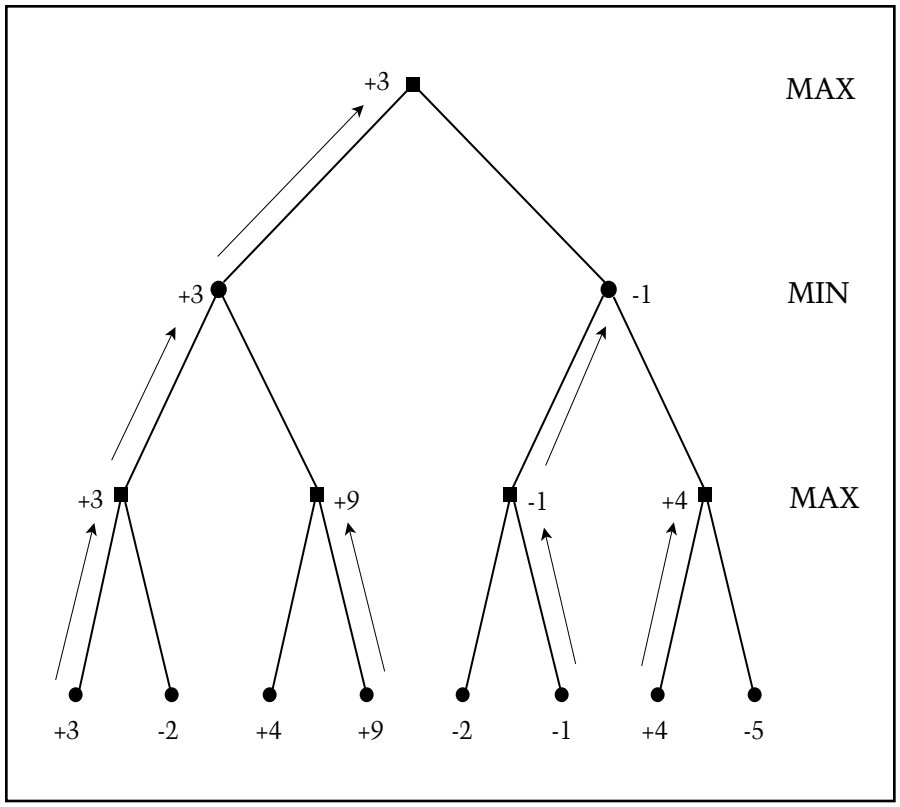

Figure 1. An Example of Minimaxing.

arbitrarily. But how should the heuristic estimates be backed-up (propagated) from deeper in the tree?

For a long time, there was universal agreement to proceed as follows: If one player (called MAX) is on move from a position within the tree, take the maximum of the successor values; if the other player (called MIN) is on move, take the minimum. By use of this back-up rule, a so-called *minimax value* of the whole tree can be computed recursively. figure 1 illustrates the way values are propagated toward the root. Usually, the primary interest is not the minimax value of the tree itself, but rather the move to be selected. In accordance with the back-up rule, the arc (move) leading to the best (by convention the maximal) backed-up value of the immediate successors is chosen. More precisely, the choice is one of what may be several moves leading to the maximum.

While any reasonable method of searching can be used, the usual choice is an iteration of successively deeper depth-first searches in the form of backtracking, to minimize storage requirements. In fact, it is possible to ignore large parts of the look-ahead tree without changing the result. There are algorithms that guarantee the same minimax value and move choice while saving considerable effort. The best-known of these is the alpha-beta algorithm which, for instance, in figure 1 must only search the part of the tree shown in boldface. Since these backward-pruning possibilities are not of primary interest in this article, the reader is referred to Campbell and Marsland (1983) or various textbooks on AI.

An important question, then, is how deeply the different branches should be searched within the given time limits. In practice, there is a tendency to search each branch to about the same depth (of course, with the exception of backward pruning), although there is an important potential for improvements through searching to variable depth (Kaindl 1983b). However, from a theoretical point of view the crucial question is: Why search at all? As heuristic values have to be used anyway, those of the immediate successor positions may be used directly for selecting a move. Unfortunately, experience shows that decisions based on this method are, with only very few exceptions, much worse than those based on deeper and deeper searches using minimaxing.

While the basic concepts of minimaxing (with the exception of search techniques and backward pruning) were proposed very early by two famous researchers, Shannon (1950) and Turing (1953), the success of this technique remains even today a theoretical mystery.

Is Minimaxing Pathological?

Nau (1979, 1980, 1983b) has proved that for certain classes of game trees the decision quality is degraded by searching deeper and backing up terminal values using the minimax propagation rule. Nau called such behavior *pathological.* The key assumptions for these game trees follow.

1. The trees are uniform, in that each nonterminal node has exactly m + n children nodes.
2. Every "critical" node has m children with the same true value, followed by n children with the opposite true value (for the remaining nodes, the true values of the children are identical according to the game-theoretic relationship).
3. There is an independent (identical) distribution of error by the static evaluation function that estimates values for terminal nodes.

Recently, Schrüfer (coauthor of a competition chess program) modified this model, distinguishing two different error parameters for "overestimating" and "underestimating," respectively (Schrüfer 1986, Section 1). While his finding that pathology can be avoided—given low error rates—is quite illustrative, the requirement for a negligible probability of just one of the two errors (underestimating) could not be related satisfactorily to observations in computer chess practice.

In later work, Nau (1982, 1983a, 1983c) investigated a class of "real" games, called Pearl's games or P-games, in which pathology actually occurs. Similar classes of games, called N-games and G-games, were shown not to be pathological. What is the essential difference between P-games and N- or G-games?

Our description of such games can be rather cursory, since they are amply described in Nau (1982, 1983a, 1983c). A *P-game* is played on a board measuring $b^{k/2}$ by $b^{k/2}$, where b > 1 and k > 0 are integers. For the example shown in figure 2, which is taken from Nau (1982, 1983a) b = 2 and k = 4 were chosen. The initial configuration of the board for a P-game is constructed by randomly assigning each square of its board one of two possible values, independently of the values of the other squares. We have indicated these values as W and L to emphasize their correspondence to the true values WIN and LOSS on leaf nodes. The first player moves by dividing the board vertically into b sections of equal width, and discarding all but one of the sections. The opponent's move consists of doing the same horizontally with the remaining part of the board. The play continues in this manner until only one square is left. If the square has value W, the last player (MAX) wins. If it has the value L, MAX loses, and the opponent (MIN) wins.

Figure 2 shows a complete game tree for this example, with the true values WIN and LOSS indicated for the boards from MAX's viewpoint. The numbers denote the heuristic values of these boards, computed by the static evaluation function described in Nau (1982) (by simply counting the number of W squares). The interested reader is encouraged to figure out whether a 1- and a 2-ply search using minimaxing will choose the correct move for MAX in the position marked by an asterisk. Note that for k[<=] 7, even for P-games, no pathological behavior has been found in Nau (1982).

Due to the board-splitting character of these games, the values of the real leaf nodes directly correspond to the (randomly assigned) values of the initial board configuration. This causes the values of sibling nodes to be completely independent of each other, which results in the occurrence of pathological behavior of minimaxing in P-games.

N-games have rules identical to those of P-games, with the one essential exception that the initial playing board is set up differently, in an "incremental" manner, so that the strength or weakness of a board position tends to be roughly the same for sibling nodes. This dependence of values is sufficient to prevent the occurrence of pathology. Nau's investigation of *G-games* (Nau 1983c) also shows nonpathological behavior. These games are a modification of P-games, such that sibling nodes have many children in common, which results in a different type of correlation between them.

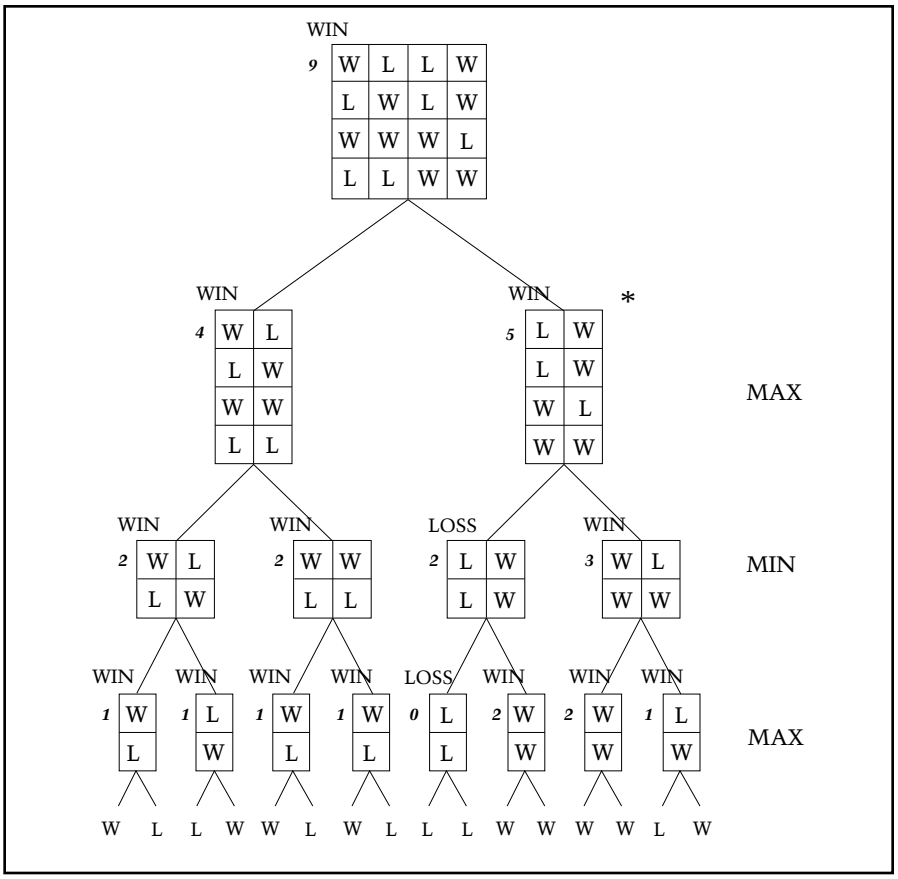

Figure 2. A Game Tree for a P-Game with b = 2 and k = 4.

Essentially the same findings were reported independently by Beal (1980, 1982) and Bratko and Gams (1982). They began by assuming independent distributions for the true as well as the heuristic values, and then rejected one of these assumptions in favor of dependence as a "clustering" of true values. A model in Schrüfer (1986, Section 2) also relates the effects of deeper minimaxing more generally to the distribution of true values. Thus, it seems clear that independence of the true node values or independence of the errors made by the static evaluation function (when evaluating terminal nodes) is a necessary condition for pathology in minimaxing.

Michon (1983) found nonpathological behavior for games with "inert" structure, despite the assumption of independent terminal values. Consequently, pathological behavior of minimaxing seems to be restricted to games with the properties of independence of nodes as well as of noninertness (games with constant branching degree are *noninert).*

However, Pearl (1983, p. 452) still left room for doubt by stating:

> "Error amplification due to minimaxing is an established fact which may significantly degrade the quality of decisions in practical games. The absence of search-depth pathology in common games only means that the deterioration in decision quality due to minimaxing is masked by other processes and so is not sufficient to show up as a weakening ability with increasing search-depth."

	Rate	P3	P4	P5	P6	P7	P8
P3	1091		4				
P4	1332	16		5 ½			
P5	1500		14 ½		4 ½		
P6	1714			15 ½		2 ½	
P7	2052				17½		3 ½
P8	2320					16 ½	

Table 1. Tournament of the Chess Program BELLE.

Version Pi searches to a depth of i plies and then enters its quiescence evaluation; for example, P8 scores 16 ½ : 3 ½ versus P7.

This conclusion seems to be based on the following statement about chess:

> "The values of successor positions appear tied down to the value of the father position partly because we only consider normally the select set of successors which are reasonable to play. It is hard to conceive of a chess position so strong that it cannot be spoiled abruptly if one really tries to make a stupid move" (Pearl 1983, p. 444).

On the contrary, it is not even necessary to conceive of such positions, but only to look into those actually investigated by the full-width search of a real chess program. Since such a search tries stupid moves most of the time, most of the terminal positions are so strong for one side that they cannot be spoiled by a single stupid move. See also Beal (1983, p. 167). Of course this clustering of values in games like chess is hard to prove formally, but it should be noted that Beal (1982) examined the king and pawn versus king ending (as a complete database on this ending is available) and concluded that the clustering factor suffices to make minimax lookahead beneficial and not pathological.

Regardless of whether such clustering really occurs in practice, it should be clear that neither the true values of nodes nor the errors of static evaluation are independent in game trees of chess and related games. The relationships among pieces on the board are strong, and they always change incrementally when the pieces are moved. The values of positions as well as the errors of evaluation depend on these relationships. If one player in such a game had twice as much material as the other, it would be absurd to believe that the values of descendant nodes would be randomly distributed. Furthermore, since all the nodes visited during a search are connected by a graph, the search tree, it is unrealistic to assume their independence. Consequently, the notion of pathology in such a domain is also unrealistic.

The argument given here is not intended to belittle the theoretical work on minimax pathology. Of course, the discovery that minimaxing can be detrimental under certain conditions and the investigations of these conditions are important. However, these findings should not be interpreted so that readers are tempted to believe that minimaxing is in principle a bad method. Practical programs for two-person games based on minimaxing give good results, and clearly better ones than programs using the other methods tried so far. Therefore, isn't it at least as important to investigate more thoroughly those conditions that lead to the beneficial behavior observed in practice? As these conditions are not understood well enough for formal treatment, one should at least try to approach such an understanding by intuitive reasoning.

Why Is Minimaxing Beneficial in Practice?

Thompson (1982) conducted an experiment on the influence of search depth on the playing strength of full-width searching chess programs. Table 1 (from Thompson 1982, p. 56) shows the results from this "BELLE tournament." Thompson found that an additional ply of search with an identical static evaluation function is equivalent to about 250 Elo rating points (Elo 1978). Although such a formula seems to be a doubtful means for extrapolating the speed necessary for a machine to defeat the human World Champion, it is consistent with the experience gained from computer chess tournaments. A similarly dramatic benefit of searching deeper using the minimax propagation rule seems to be true for checkers and kalah.

Although demonstrating this benefit appears to have been the initial motivation for the theoretical analysis of minimaxing, the discovery of pathological behavior in minimaxing on certain unrealistic game trees was the first surprising result. In addition, the succeeding investigations could not provide theoretical support for the dramatic benefit observed in practice. For instance, Table 5 in Nau (1982) shows only modest improvements in decision quality with increasing search depth for N-games.

The dependencies in real game trees

reflect the structure of real games, and consequently pathology does not occur there; but clustering relationships between the true values alone cannot account for the large benefits observed, mainly because they seem to be an outer phenomenon. This relationship of the true values may be induced at least partially by a more fundamental relationship between true and heuristic values within whole subtrees.

Let us briefly look at some extremely simple examples from the very domain for which Shannon (1950) and Turing (1953) made their proposals (and where minimaxing is truly useful). figure 3 presents fragments of chess positions showing simple tactical patterns. These examples will give the reader a very crude idea of what happens in a typical computer chess program. To follow these examples, it is only necessary to know how chess pieces move, and that the relative heuristic values of queen, rook, and pawn are about 9, 5, and 1.

In the following, we assume a static evaluation function that simply sums up the relative heuristic values of the pieces on the board for each side and then subtracts these sums. (Although the evaluation functions actually used in chess programs may include various positional and strategic subtleties, the material term is by far the dominating one.) In figure 3a a full-width, 1-ply search will select Rh3xQh7 (rook moves to square h7 to capture queen) for White on move. Since this move changes the static values drastically, a position containing this pattern cannot be considered *quiescent.* (The importance of quiescence had already been recognized by Shannon and Turing, and this concept will be treated more theoretically in the next section.) In fact, such simple issues are successfully handled by most programs within their *capture quiescence search:* selectively searching captures even beyond the horizon of a full-width search. However, when the side on move has an *en prise* piece—a piece in risk of capture—more subtle issues become relevant (Kaindl 1982, 1983a).

Especially, multiple en prise pieces raise more difficult problems. The interested reader is encouraged to figure out how a minimax search may discover whether a rook can be gained by the "forking" move of the White pawn from g2 to g4, depending on the remainder of the board partly shown by figure 3b. (As a hint, possibilities such as one of the rooks putting the White king in check may be crucial.) Nevertheless, such issues usually occur in more complicated combinations. For example, in figure 3c White has to exchange queens before the forking pawn move can be successful.

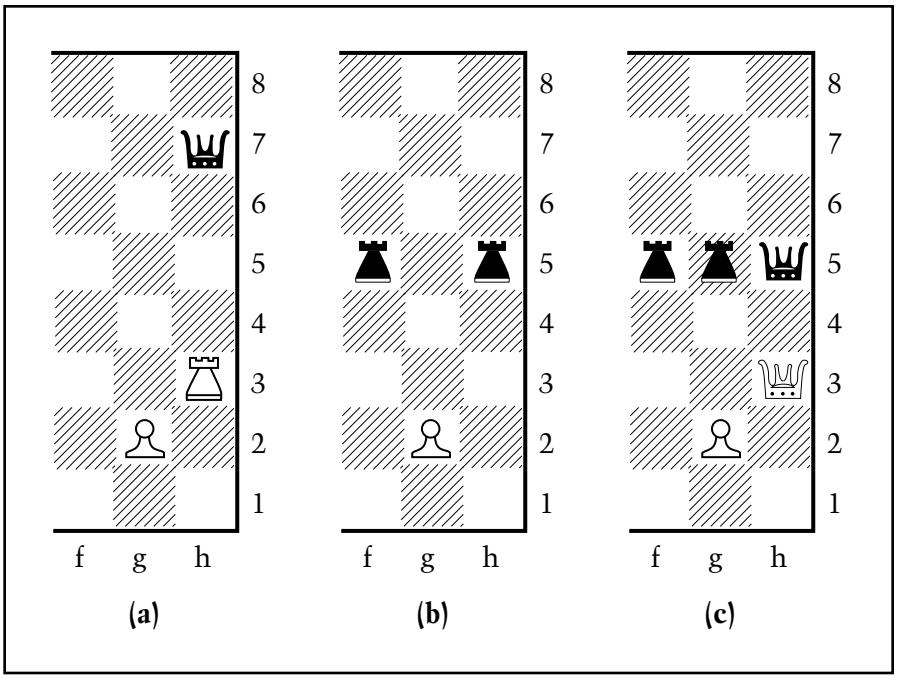

Figure 3. Fragments of Chess Positions Showing Tactical Patterns.

These examples have only shown rather simple tactical issues. Certain positional issues, such as how to maneuver a piece to a central square, are handled in a similar way. Actually, there is a strong temptation to think that the reason for the benefits of searching deeper is trivial. But do any of the previously published attempts to prove the benefits of minimaxing model convincingly what these issues really signify in an abstract minimax tree?

A common argument is that of "improved visibility," which was simply interpreted and modeled by Pearl (1983) in the sense that the accuracy of the static evaluation function improves as the game proceeds. Mathematical analysis of this model shows that a very strong improvement is necessary to combat pathology, but this is based on the artificial assumption of the independence of values. Thus, a similar analysis based on dependent values might be of interest.

For the game of chess, no such improvement in accuracy can be seen when considering the static evaluation functions used in actual programs. However, to compute a minimax value, the relation of the terminal values to the minimax value is important. This is utilized by the TEST procedure of the SCOUT algorithm (Pearl 1980). As most of the terminal values of a chess tree are very strong for one side (as was argued before in favor of clustering), these relations are generally evaluated very accurately, and this relative accuracy improves with increased search depth. See also Bratko and Gams (1982, p. 12-14).

Another argument investigated by Pearl (1983) is that of "the avoidance of traps." Again the model is oversimplified, considering only actual terminal nodes (that is, mates and stalemates) at all levels of the game tree as "traps." Pearl's analysis shows that the presence of such traps makes minimax look-ahead beneficial, mainly because such real terminal nodes can be evaluated without error. Again this

model is based on the assumption of independence of all the other values, which probably explains why the existence of real terminal nodes has such a strong influence on the analysis.

In computer chess practice this type of trap alone does not seem to make a strong contribution to the benefits of minimaxing. Statistics compiled at the fourth World Computer Chess Championship with the program MERLIN suggest that the number of actual terminal nodes in a search tree is usually less than 0.1 percent and seldom reaches 1 percent. Moreover, even if not a single terminal node is visited during the search, minimax look-ahead is very useful.

The avoidance of traps in a more general sense is actually one of the major tasks of minimax look-ahead in chess: to recognize forced material gain or loss. Such a search discovers "double attacks" (see figure 3b), "piece overloading," "decoying," and all the other tactical chess elements about which the usual static evaluation functions have no explicit knowledge. Because tactics plays a major role in games like chess, checkers, or kalah, searching deeper is a very successful method for improving the performance of programs for such games. In backgammon, however, forced variations are not very important. Thus Berliner's (1980) backgammon program, based primarily on knowledge and not on search, was very successful.

The minimax propagation rule backs up exactly one value for each position, the one it assumes to be best. Consequently, this rule is well suited for games in which each player has exactly one move from each position; in fact, it is a generalization of the game-theoretic relationship. Of course it is important that this heuristic value is sufficiently reliable. Fortunately, the dominating factor for evaluation in chess, checkers (material balance), and kalah (number of stones) is sufficiently reliable most of the time. It is also interesting to reconsider the difference between P- and N-games from this point of view. According to the tables in Nau (1982, 1983a), the same static evaluation function is clearly more accurate for N-games than for P-games. (D′ (1, k) provides a measure of the accuracy of the evaluation function for nodes with distance k from the end of the game.)

For the game of chess, no such improvement in accuracy can be seen when considering the static evaluation functions used in actual programs.

There is insufficient space to illustrate the role of tactics more thoroughly here, but the benefits of minimax look-ahead can also be seen in a more general way. The static evaluation functions used in practice usually incorporate only *static knowledge* (that is, there is no attempt to evaluate the outcome of immediate impending combat), whereas the *dynamic* aspects are evaluated conveniently by a full-width minimax search. In terms of planning, the static evaluation functions can only measure the current state of goal achievement (for example, the material advantage of a position), but cannot realize whether and how such a goal can be reached (for example, by forcing the opponent in such a way that material gain cannot be prevented).

It is interesting to note that it is exactly this difference between static and dynamic evaluation, together with a fixed search horizon, that causes one of the major defects of the minimax method, the *horizon effect;* see also Berliner (1973). Berliner (1981, p. 585) showed that this effect does not influence the played move very often when searching is sufficiently deep. Nevertheless it is worth trying to avoid it by investigating important variations more deeply than others (Kaindl 1981, 1982, 1983a, 1983b).

A Model Based on Quiescence

The most commonly used technique to counter the horizon effect is the quiescence search. The importance of quiescence has been known ever since the early work on computer game-playing; Turing (1953) called quiescent positions "dead." However, there is an important point often missed in this regard: Quiescence is not only related to the properties of a certain game, but even more so to the static evaluation function used. A position is *nonquiescent* if its value can be changed drastically by moves or move sequences. For example, in chess the usual capture quiescence search has to be considered in relation to the common use of materially dominated static evaluation functions (as in figure 3a). Beal defined a node as "consistent" if its heuristic value is the same as the backed-up value from a 1-ply search over its descendants (Beal 1980, p. 106). Most interestingly, although this definition was part of a model of quiescence search and forward pruning, Beal did not relate this model to a model showing pathological behavior presented earlier in the same article, in the sense of using quiescence search and forward pruning to overcome pathology.

Now let us generalize the notion of consistency in order to get a model of quiescence that reflects the empirically derived notion.

Definition: Given a static evaluation function f returning heuristic values (HV), a node k is *n-ply-quiescent* if and only if the static heuristic value HV of k assigned by f is equal to the minimax value of k resulting from a full-width search of the subtree below k up to depth n, where the terminal nodes are assigned heuristic values by f.

In practice, the heuristic evaluators usually return a wide range of values. Therefore, strict equality would be achieved very seldom, and a relaxation, for example, in the sense of "small difference," would be more realistic. However, to make the model as simple as possible, we will restrict ourselves here to a two-valued function. In this context, the criterion of equality seems appropriate.

The following model is an attempt to concentrate on the strict separation

of the dynamic and static aspects discussed above, using the definition of n-ply-quiescence. While a relatively complex relationship between the true values and heuristic estimates is introduced, the overall model has been kept as simple as possible.

1. The tree structure has a uniform branching factor b.
2. True values of nodes (TV) are either WIN or LOSS.
3. True values have the game-theoretic relationship.
4. Heuristic values (HV) are either +1 (estimating WIN) or -1 (estimating LOSS).
5. Probabilities of error e+ and e- are defined as follows (k being a node):

   e+ (k) = P (HV(k) = +1 | TV(k) = LOSS)

   e- (k) = P (HV(k) = -1 | TV(k) = WIN)
6. For all nodes l (those n-ply-quiescent according to the definition) and for all nodes m (those not n-ply-quiescent), the following conditions hold:

   e+ (l) < e+ (m)

   e- (l) < e- (m)

   e+ (l) < e- (m)

   e- (l) < e+ (m)
7. The number of nodes that are n-ply-quiescent is small compared to the number of nodes that are not n-ply-quiescent.

Assumptions 1 to 5 are fairly standard in many models and are given here for completeness. However, assumptions 6 and 7 are new and model the properties that are very likely the ones responsible for the dramatic benefits observed in practice. Assumption 6 introduces the relationship between the errors of evaluating quiescent and nonquiescent nodes. The number n of n-ply-quiescence can be treated as a parameter of the model, and it seems likely that there is a strong relationship between n-ply-quiescence and the benefits of searching the subtree n plies deep. The assumption itself, that the heuristic values are much more reliable for quiescent positions than for nonquiescent ones, is undoubtedly justified by practical observations. (Why else are resources spent for quiescence searching?) Assumption 7 models the dynamic nature of tactical domains such as chess, checkers, or kalah, where at least up to now, no static evaluation functions have been written that are sufficient to capture their dynamic aspects.

Unfortunately, this model has so far resisted formal analysis, because of the relatively complex relationships it contains. Thus, the statement that it models observed behavior remains a conjecture. Nevertheless, as an illustration it could help us to approach a deeper understanding.

Michon (1983) investigated the influence of quiescence analysis formally and found it to be beneficial. Unfortunately, he considers quiescence mainly from a quantitative point of view, focusing on the natural termination of a quiescence search. However, the criterion for quiescence used in his model is not related to the static evaluation used to model the empirically derived notion of quiescence discussed above. Therefore it is doubtful whether this analysis provides an explanation for the actually observed benefits of minimaxing. In fact, a theorem based on this model indicates that the probability of error converges toward zero, regardless of how poor the static evaluation is (Michon 1983, p. 93). Considering the extreme case in which the static evaluation function systematically considers all LOSSes as WINs and vice versa, we can note that such behavior has not yet been observed in practice.

Consequences for the Proper Use of Minimaxing

In practice, searching to a strictly fixed depth is rather unusual, as simple criteria for relaxing the depth restriction—that is, quiescence (making direct capture moves) or forcedness (forced response when the king is in check)—have given very good results. In the future, extensively searching to variable depth probably will further improve results, since this effectively utilizes the benefits of minimax searching while simultaneously reducing its defects.

Michon (1983, p. 78) used the same criterion for quiescence and to decide whether a move should be counted as a ply of depth. (When a move is not counted, then the corresponding path is searched deeper, and in effect, the search horizon becomes variable.) For the latter purpose Michon's criterion seems to fit much better, but there it models the notion of forcedness rather than quiescence. Although these concepts are related in some sense, it seems useful to distinguish between them, especially for their application in practice. It appears to be safe to use the criteria for forcedness also for quiescence. For example, the criterion "replies to check" is used successfully for both concepts in most chess programs. Responses when the king is in check are usually small in number, and consequently the player on move is usually forced. Hence, a reliable static evaluation of such a position is fairly difficult, and the result is nonquiescence.

In the future, extensively searching to variable depth probably will further improve results

However, the inverse process is rather dangerous and must be handled very carefully. Ken Thompson performed an experiment with the Computer World Champion of 1980, BELLE, which can serve as empirical evidence here. Thompson changed BELLE to count capture moves as only half a ply: Whenever two (four ...) capture moves in a path occur, this path is searched one (two ...) ply deeper. This procedure does not seem unreasonable, considering the enormous utility of the capture quiescence search. However, he changed it back after having had bad experience with it in several games. The reason for this is rather simple: Capture moves are very useful for refuting bad lines within the tree (in the sense of alpha-beta cutoffs), but these refutations became much more expensive as a result of this change. The subtrees below the refuting capture moves usually had to be searched deeper.

Even before these empirical data were available, a model was created (Kaindl 1983b) that also includes the "interest" of a position in the decision

whether to extend the horizon from it. When comparing estimated values of the interest with the aspiration window (represented by the current values of α and ß), one can try to avoid wasting effort on variations that are not likely to become important for the result of the search.

Conclusion

This discussion is an attempt to help bridge the gap between theory and practice for minimax look-ahead. The arguments in favor of minimaxing and the model presented should also support efforts to obtain the desired theoretical evidence of the benefits observed in practice. Unfortunately, the model resulting from the empirical approach used here has resisted formal analysis up to now because of the relatively complex relationships between the true values and heuristic estimates of whole subtrees. However, it can serve as an illustration of an approach to deeper understanding. finally, the insights gained from such investigations should help to improve practical applications.

Postscript

Although this article is devoted to minimaxing, we should note that a different method of propagating values has recently become popular in the context of theoretical analysis of game tree searching. This method considers heuristic estimates as independent probabilities of winning. It was first used by Slagle and Bursky (1968) and much later was again proposed by Pearl (1983). There is some evidence that it may be rather useful for games where minimaxing is pathological (Nau 1983a, 1983c). Unfortunately, no suitable tree-pruning procedure (such as alpha-beta for minimaxing) is available. Thus, it appears that for computer chess practice, this method has more conceptual defects than minimaxing (Horacek, Kaindl, and Wagner 1987).

References

Beal, D. F. 1983. Recent Progress in Understanding Minimax Search. In Proceedings of the ACM Annual Conference, 164-169. New York: Association for Computing Machinery.

Beal, D. F. 1982. Benefits of Minimax Search. In *Advances in Computer Chess* 3, ed. M. R. B. Clarke, 17-24. Oxford: Pergamon.

Beal, D. F. 1980. An Analysis of Minimax. In *Advances in Computer Chess* 2, ed. M. R. B. Clarke, 103-109. Edinburgh, Scotland: Edinburgh University Press.

Berliner, H. J. 1981. An Examination of Brute Force Intelligence. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 581-587. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Berliner, H. J. 1980. Backgammon Computer Program Beats World Champion. *Artificial Intelligence* 14(1): 205-220.

Berliner, H. J. 1973. Some Necessary Conditions for a Master Chess Program. In Proceedings of the Third International Joint Conference on Artificial Intelligence, 77-85. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Bratko, I., and Gams, M. 1982. Error Analysis of the Minimax Principle. In *Advances in Computer Chess* 3, ed. M. R. B. Clarke, 1-15. Oxford: Pergamon.

Campbell, M. S., and Marsland, T. A. 1983. A Comparison of Minimax Tree Search Algorithms. *Artificial Intelligence* 20(4): 347-367.

Elo, A. 1978. *The Rating of Chess Players, Past and Present.* London: Batsford.

Horacek, H.; Kaindl, H.; and Wagner, M. 1987. Probabilities in Game-Playing: Possible Meanings and Applications. In Proceedings of the Third Austrian Meeting on Artificial Intelligence, 12-23. Berlin: Springer-Verlag.

Kaindl, H. 1983a. Quiescence Search in Computer Chess. In *Computer Game-Playing: Theory and Practice,* 39-52. Chichester, England: Ellis Horwood. Also in *SIGART Newsletter* 80, ed. M. A. Bramer, 124-131, 1982.

Kaindl, H. 1983b. Searching to Variable Depth in Computer Chess. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 760-762. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kaindl, H. 1982. Dynamic Control of the Quiescence Search in Computer Chess. In Proceedings of the Sixth European Meeting on Cybernetics and Systems Research, ed. R. Trappl, 973-978. Amsterdam: North-Holland.

Kaindl, H. 1981. Realisierung von langfristigem Planen und Ma[Help - beta sign]nahmen gegen den Horizont-Effekt im Computer-Schach. Doctoral diss., Technical Univ. of Vienna.

Michon, G. 1983. Recursive Random Games: A Probabilistic Model for Perfect Information Games. Ph.D. diss., Univ. of California at Los Angeles.

Nau, D. S. 1983a. Pathology on Game Trees Revisited, and an Alternative to Minimaxing. *Artificial Intelligence* 21(1,2): 221-244.

Nau, D. S. 1983b. Decision Quality as a Function of Search Depth on Game Trees. *Communications of the ACM* 30(4): 687-708.

Nau, D. S. 1983c. On Game Graph Structure and Its Influence on Pathology. *International Journal of Computer and Information Sciences* 12(6): 367-383.

Nau, D. S. 1982. An Investigation of the Causes of Pathology in Games. *Artificial Intelligence* 19(3): 257-278.

Nau, D. S. 1980. Pathology on Game Trees: A Summary of Results. In Proceedings of the first National Conference on Artificial Intelligence, 102-104. Menlo Park, Calif.: American Association for Artificial Intelligence.

Nau, D. S. 1979. Quality of Decision Versus Depth of Search on Game Trees. Ph.D. diss., Duke Univ.

Pearl, J. 1983. On the Nature of Pathology in Game Searching. *Artificial Intelligence* 20(4): 427-453.

Pearl, J. 1980. Asymptotic Properties of Minimax Trees and Game-Searching Procedures. *Artificial Intelligence* 14(2): 113-138.

Schrüfer, G. 1986. Presence and Absence of Pathology on Game Trees. In *Advances in Computer Chess* 4, ed. D. F. Beal, 101-112. Oxford: Pergamon.

Shannon, C. E. 1950. Programming a Computer for Playing Chess. *Philosophical Magazine* 41(7): 256-275.

Slagle, J. R., and Bursky, P. 1968. Experiments with a Multipurpose, Theorem-Proving Heuristic Program. *Communications of the ACM* 15(1): 85-99.

Thompson, K. 1982. Computer Chess Strength. In *Advances in Computer Chess* 3, ed. M. R. B. Clarke, 55-56. Oxford: Pergamon.

Turing, A. M. 1953. Digital Computers Applied to Games. In *Faster than Thought,* ed. B. V. Bowden, 286-295. London: Pitman.