Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

An Analysis of Decision Quality of Minimaxing vs. Product Propagation

Helmut Horacek
FR 6.2 Informatik
Universität des Saarlandes
Saarbrücken, Germany
horacek@ags.uni-sb.de

Hermann Kaindl
Institute of Computer Technology
Vienna University of Technology
Vienna, Austria
kaindl@ict.tuwien.ac.at

***Abstract*—The *minimaxing* approach to backing-up heuristic values has been very successful, e.g., in computer chess for making move decisions on the world-champion level, by employing very deep searches and effective pruning algorithms. From a theoretical point of view, however, it is not yet clear whether minimaxing or *product propagation* is better in terms of decision quality without deep searches.**

We present a systematic analysis of game trees with depth 2, where a single application of the competing back-up rules each (in a given branch from the root of the search tree) reveals their pure decision quality. Interestingly, product propagation tends to make better decisions per se more frequently, under realistic assumptions modeled after real game-playing programs. So, its decision quality may still make it a viable alternative for game trees where only shallow searches are affordable.

***Index Terms*—Back-up of heuristic values, minimaxing, product propagation.**

I. Background and introduction

Except for rare positions of very complex combinatorial games, there is no practical way of determining the exact status (the *true* value) of non-goal nodes that represent most of the positions in a game tree. Therefore, it is usually necessary to resort to *heuristic* estimates of their "goodness" or "strength" for one side. Such values are assigned by a so-called *static evaluation function* f which incorporates heuristic knowledge about the domain in question. In this paper it is sufficient to consider $f(n)$ as some function that evaluates each node n (that represents a position) with some error.

Given such an evaluator, the question remains how its values can be used for making reasonable decisions. Since the immediate application of f to the children of the given node usually does not lead to good decisions in practice, it seems natural to look ahead by searching deeper and evaluating the resulting nodes. For such a procedure it remains to be specified how deep the various branches should be searched (to constant or variable depth [1]) and, how the heuristic values should be backed up (i.e., propagated) towards the given node's children. In *two-person games* with *perfect information* the most successful approach for the back-up of values in practice has been *minimaxing* (for a description see, e.g., [2]). In the following, we assume that $f(n)$ assigns a value to a node n from the viewpoint of the moving side at n. We define here a special case where all branches are searched to the same depth.

Definition 1: A *minimax value* $MM_f(n)$ of a node n can be computed recursively as follows (in the *negamax* formulation):

(1) If n is considered *terminal*: $MM_f(n) \leftarrow f(n)$
(2) else: $MM_f(n) \leftarrow \max\limits_i(-MM_f(n_i))$ for all child nodes n_i of n.

$MM_f^d(n)$ is the minimax value of node n resulting from exactly d applications of the recursion (2) in every branch of the search tree. That is, $MM_f^d(n)$ defines the minimax value from a *full-width search* of the subtree rooted at n to a uniform depth d.

The use of minimaxing in computer chess and checkers practice is more or less ubiquitous. For instance, the special chess machine Deep Blue, which defeated the highest-rated human chess player for the first time in a match consisting of several games under tournament conditions, uses minimaxing as well as the checkers program Chinook, which has even become the official man-machine world champion [3].

Yet there has been some theoretical doubt on its usefulness as pointed out by Pearl [4]. In fact, minimaxing treats heuristic estimates as if they were true values. Pearl compared this to committing one of the deadly sins of statistics, computing a function of the estimates instead of an estimate of the function. So, there is some lack of theoretical foundation and explanation for the (very successful) use of minimaxing in game-playing practice.

Even to the contrary, Nau [5] showed that for certain classes of game trees the decision quality is degraded by searching deeper and backing up heuristic values using the minimax propagation rule. He called such behavior *pathological*. Essentially the same findings were reported independently by Beal [6]. Several subsequent studies like [4] provided some insight into minimax pathology, but for a long time there was no success in explaining the strong improvements with increasing search depth observed in practice.

Since the benefits of using minimaxing in practice had not been explained theoretically, different back-up rules have been proposed, such as *product propagation* [4] (in fact, this rule was already used much earlier by Slagle and Bursky [7]). It requires that an evaluation function $f'(n)$ returns values between 0 and 1 that are estimates of the probability that the position represented by node n is a forced win. $f'(n)$ assigns a value to a node n from the viewpoint of the moving side at n.

Definition 2: A *probability estimate* $PP_{f'}(n)$ of a node n can be computed recursively as follows (in the *negamax* formulation):

(1) If n is considered *terminal*: $PP_{f'}(n) \leftarrow f'(n)$
(2) else: $PP_{f'}(n) \leftarrow 1 - \prod_i (PP_{f'}(n_i))$ for all child nodes n_i of n.

$PP^d_{f'}(n)$ is the depth d estimate of node n resulting from exactly d applications of the recursion (2) in every branch of the search tree.

For the specific comparison of depth 2 searches, which is the topic of this paper, we compare a single application of the recursive component labeled as (2) under definitions for Minimaxing (Definition 1) and Product (Definition 2), backing up the leaf node values from depth 2 to depth 1. In depth 1, simply the position is chosen, which is assigned the superior value.

Product propagation is theoretically sound for *independent* probabilities. However, as noted already by Slagle and Bursky such probabilities are generally *not* independent in practice.

The remainder of this paper is organized in the following manner. First, we summarize previous work on comparing minimaxing with product propagation. Then we present our analysis of the decision quality of minimaxing and product propagation. In this respect, we illustrate representative examples of errors first. Finally, we elaborate on the back-up differences from searches to depth 2.

II. Previous work

While there has been a fair bit of attention on the theoretical problem of minimaxing pathology, we focus here on the work that directly compares minimaxing with product propagation under various conditions.

Nau [8] investigated product propagation as an alternative to minimaxing and found no pathology even in so-called P-games, where pathological effects had been shown for minimaxing. In fact, the values of the real leaf nodes of P-games directly correspond to the values of the squares in the initial board configuration, which are randomly assigned one of two values independently of the values of the other squares. Under these conditions as given in P-games, Nau's experiments resulted in a higher probability of correct move decision using product propagation compared to minimaxing. In so-called N-games (with incremental dependencies of true game-theoretic values), the results showed about the same probability of correct move decision for both back-up rules. In a P-game contest, a program based on product propagation scored marginally better than an otherwise identical program based on minimaxing.

In later work, Nau *et al.* [9] reported that product propagation scored better than minimaxing (at most search depths) in a P-game contest when counting the "critical" games only.[1] Further experiments showed, however, that minimaxing was better than product propagation (for search depths 3 and 4) in an N-game contest. The overall conclusion in [9] was that the minimax propagation method is often not the best method to use.

Nau [10] also used so-called G-games (with dependencies of true game-theoretic values in graphs where sibling nodes have many children in common) for comparing these propagation rules, which indicated some influence of the evaluation function used. G-game contests revealed that product propagation performed better than minimaxing if some evaluation function was used and worse than minimaxing if another function was used that is more accurate for these games. Results by Chi and Nau [11] confirmed this relationship of the respective advantages of these rules to the strength of an evaluation function used: the stronger the evaluation function the better for minimaxing.

Additionally, Chi and Nau compared these back-up rules on several games, including a small variant of kalah. Most interestingly, in this real game a program based on product propagation performed better than its opponent based on minimaxing.

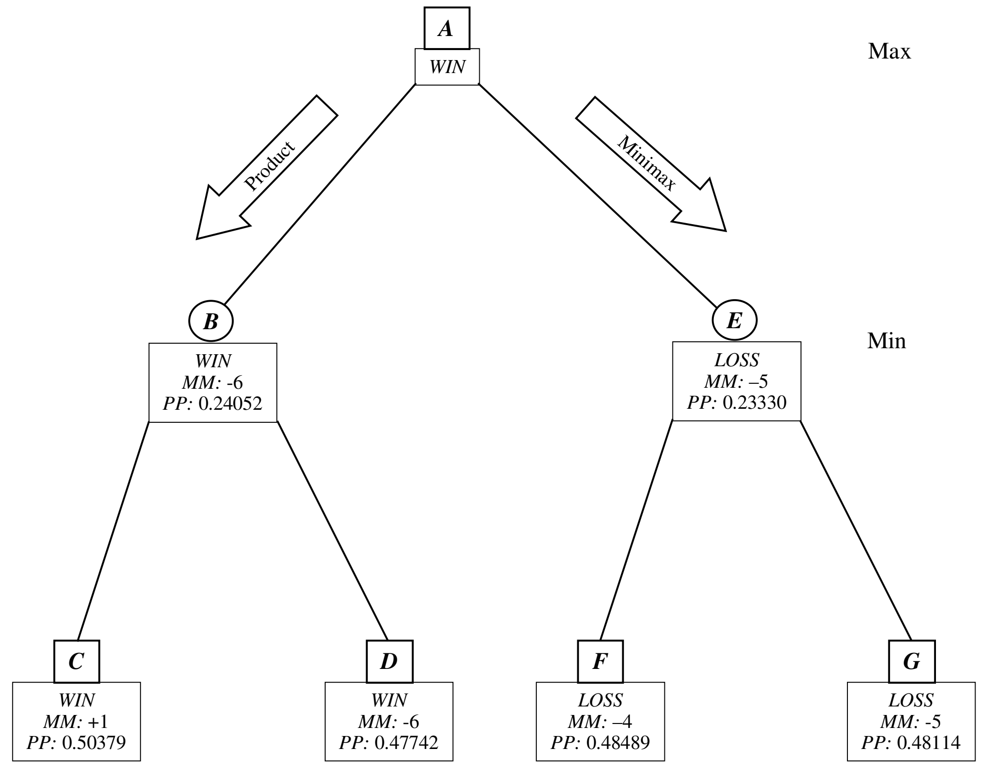
Since both programs searched to the *same depth*, however, these comparisons were unfair for minimaxing, which could have utilized well-known pruning procedures for searching much deeper with the same number of nodes generated. (For a comparison of pruning procedures see [12].) Still, there was some indication that product propagation may be the better rule for backing up heuristic values.

Kaindl and Scheucher [13] made simulation studies for comparing these backup rules in synthetic trees. These trees were randomly generated with properties found in certain real game trees (from computer chess), according to a tree-generation approach developed by them earlier [14]. Their results show that product propagation can be slightly better than minimaxing in such trees, but only under very unrealistic conditions. First, the actual probabilities to win must be available for product propagation. Second, both approaches must be able to search to the same depth.

Nau *et al.* [9] e.g., investigated combinations of minimaxing and product propagation. Their results suggested that the respective advantages could be combined by some combination of these back-up rules, but these combinations cannot be applied to the case of searching to depth 2, that we address in this paper.

In summary, the previous theoretical work on comparing minimaxing with product propagation was not conclusively in favor of one or the other back-up rule. In particular, it neither provided an explanation why product propagation has been neglected, e.g., in computer chess practice, nor whether this would be justified. It is clear though, that the pruning power

[1] For each initial game board, one game was played with one player moving first and another was played with his opponent moving first. For some game boards, one player was able to win both games of the pair. These are called critical games.



WIN, LOSS	True value
MM	Minimax back-up value or heuristic value of terminal node
PP	Product back-up value or probability to win of terminal node

Fig. 1. Move decision error by Minimax.

of alpha-beta and related algorithms has been the major reason. Unfortunately, this also seems to have been the reason for not studying the respective decision qualities further.

III. Analysis

In the following, we present an analysis of the differences between minimaxing and product propagation. First, we illustrate some typical move-decision errors caused by both back-up rules through two characteristic examples. Then we carry out a systematic analysis of the differences between the back-up rules according to combinations of evaluation errors. Finally, we quantify this analysis using the game tree model introduced in [13].

A. Two representative examples of errors

In order to get a better understanding of the relative qualities of these back-up rules, let us have a look at representative examples of move-decision errors caused. In fact, we randomly selected one example each from the tree searches to depth 2 in the games of the simulation studies presented in [13], that makes a difference to the move decision of Minimax and Product, respectively. "Minimax" and "Product" are players that use minimaxing (see Definition 1) and product propagation (see Definition 2), respectively.

In Fig. 1 and 2, intermediate and leaf nodes contain three values: the minimax value labeled by MM, the product propagation value labeled by PP, and the true value of the position associated with this node (either WIN or LOSS). A (terminal) position is considered to be erroneously evaluated if its true value is inconsistent with the side favored by minimax and product.

In the example of Fig. 1,[2] Minimax decides for the wrong move to the lost position E because of the gross evaluation error of position D in the other subtree. The wrong heuristic estimate is treated by Minimax as if it were a true value. Product compensates for this evaluation error in this example through taking into account the evaluation of position C as well.

In the example of Fig. 2, Product decides for the wrong move to the lost position E. In essence, the high probability to win estimated for position G more than outweighs the much smaller probability to win estimated for position F

[2]In the presentation of such an example, it is better to use the minimax instead of the negamax formulation for pedagogical reasons. Therefore, we show all values from the viewpoint of the moving side at A.

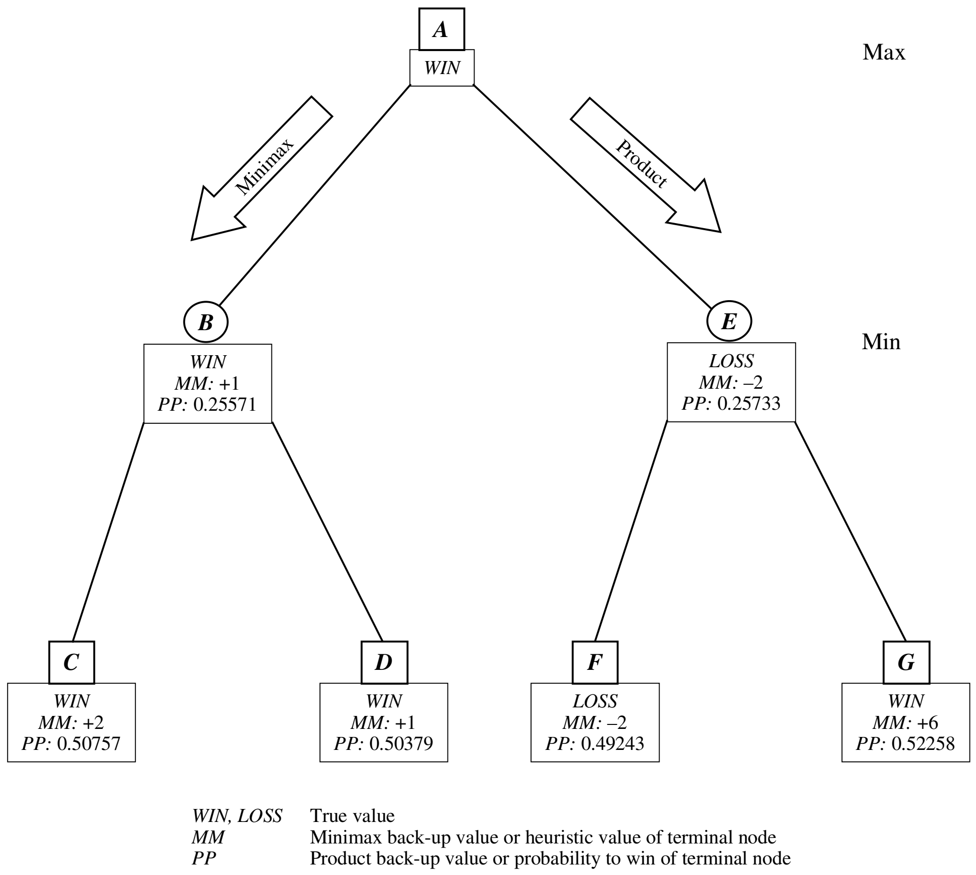


Fig. 2. Move decision error by Product.

(which amounts even to a small estimated probability to lose of 0.50757), compared to small probabilities to win estimated for both positions C and D in the other subtree. In contrast, Minimax makes the correct decision in this example because of avoiding the position F due to its negative evaluation. These examples are consistent with the major result of [11], [10] that relates the respective advantages of minimaxing and product propagation to the strength of the evaluation function. The example of Fig. 1 contains a gross evaluation error of a terminal node, while there is even *no* such error in Fig. 2. However, these examples suggest a different view of these back-up rules than that of Pearl found in [4], [2]. Pearl states that minimaxing may be useful for beating a fallible opponent, since this back-up rule contains a realistic model of a fallible opponent who shares the assessment of the terminal values by the player Minimax — such an opponent can be *predicted* to choose the moves evaluated best in the search tree of Minimax. Actually, it depends on the errors made by the evaluation function. Under the conditions in our model (and presumably in real computer chess programs), Product would play for the option of taking advantage of an error by a fallible opponent as illustrated in the example of Fig. 2: Product can be viewed to "hope" for a blunder by a fallible opponent, that might lead to position G with a high estimated probability to win. After all, Product takes into account the evaluations of *all* branches Since this move decision leads to a position with $LOSS$ status, it is objectively an erroneous decision, although at least in this example there is no single evaluation error in the sense that a node with true value $LOSS$ (WIN) is assigned a heuristic value with win (loss) estimate.

B. Qualitative analysis of back-up differences from searches to depth 2

We carried out a systematic analysis of the differences between the back-up rules according to combinations of evaluation errors. The basic situation in which differences between minimaxing and product propagation manifest themselves is illustrated in Fig. 3, for depth 2 and branching degree 2, with node labels as indicated. In order for this diagram to represent a *critical case*, two conditions concerning the values associated with the nodes must hold:

1) the back-up rules must select different moves;
2) one move leads to a won position, while the other leads to a lost position.

Whether the first condition holds can be derived from the definitions of the competing back-up rules. Without loss of gen-

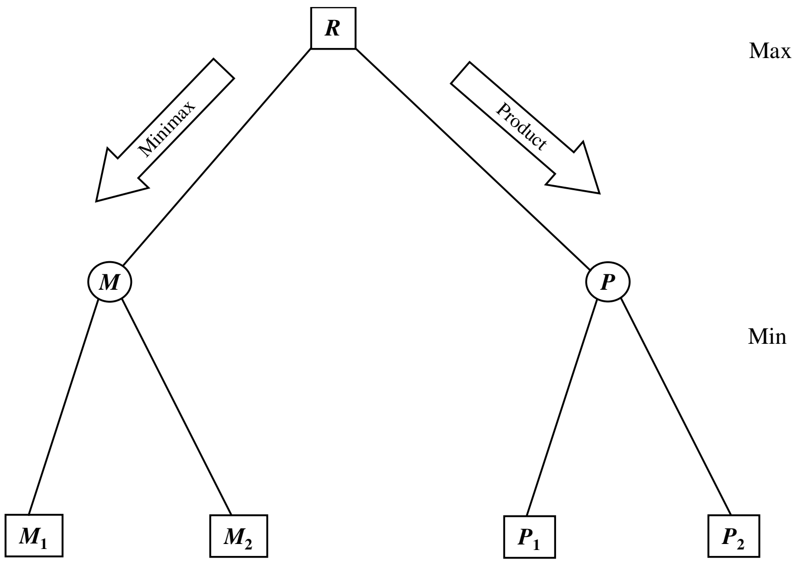


Fig. 3. Basic situation.

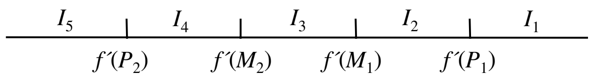


Fig. 4. Intervals of the draw value.

erality (due to symmetries), we assume that Max is on move in the root position R, M the node preferred by Minimax, P the one preferred by Product, and that $f'(M_1) \geq f'(M_2)$ and $f'(P_1) \geq f'(P_2)$ hold. Under these conditions, $f'(M_1)$ and $f'(M_2)$ must both be greater than the value of the smaller of the successor nodes of P, $f'(P_2)$, according to minimaxing. Conversely, product propagation demands that the product of the successors of P is greater than the product of the successors of M. Therefore, the value of the larger of the successor nodes of P, $f'(P_1)$, must be greater than both $f'(M_1)$ and $f'(M_2)$. Consequently, there exists a partial ordering between the values of these four nodes:

$$f'(P_2) \leq f'(M_2) \leq f'(M_1) \leq f'(P_1) \tag{1}$$

The second property, that makes a case critical depends on the true values associated with the competing nodes, M and P, which in turn depend on the true values of their successor positions. Hence, we have to distinguish which of the positions M_1, M_2, P_1, and P_2 are evaluated correctly and which ones are not. This yields 16 cases to be considered.

In our analysis, we also distinguish how the transition point between loss and win (the "draw value", 0 in terms of f, and 0.5 in terms of f') relates to these four values. This yields 5 cases, leading to a total of 80 cases. Among these cases, those are relevant where the move decision matters, that is, where M is won and P lost for the side to move, or vice versa. Together with the partial ordering on the evaluations of the positions under discussion, the draw value can be in one of the intervals as labeled in Fig. 4.

For illustration purposes, let us discuss two representative cases:

1) If there is no evaluation error, then only the case where the draw value falls in interval I_4 makes a distinction between the competing back-up rules, since M is won and P lost in I_4 (compare with the example in Fig. 2). If the draw value falls in interval I_5, then all values are "positive" (that is, $f > 0$, i.e., $f' > 0.5$) and both M and P are won. If the draw value falls in one of the intervals I_3, I_2, or I_1, then both M and P are lost.
2) Similarly, assume that the draw value is in interval I_3, and exactly one of the heuristic values is erroneous. Then this value must either be $f'(M_2)$ in order to make minimaxing superior to product propagation, or this value must be $f'(P_2)$ in order to favor product propagation. In the two remaining cases, the decision is irrelevant, since both M and P are lost.

The analysis of all 80 cases along similar lines yields the differences between minimaxing and product propagation summarized in Table I. The left part contains the constellations where minimaxing makes the right decision and product propagation does not so, and the right one those where only product propagation succeeds. The lines are ordered according to the number of evaluation errors associated with the four positions, and in each cell, the erroneous positions are indicated, together with the interval in which the draw value falls in each case.

Despite the high degree of abstraction of the information contained in its entries, we can use it to show several important points. Advantages and disadvantages of minimaxing and product propagation are balanced over these constellations —

	intervals favoring minimaxing	intervals favoring product propagation
no error	I_4	
one error (position)	$I_4(P_1)$	$I_2(P_2)$
	$I_3(M_2)$	$I_3(P_2)$
	$I_5(P_1)$, $I_5(P_2)$	$I_5(M_1)$, $I_5(M_2)$
two errors (positions)	$I_1(M_1, M_2)$	$I_5(M_1, M_2)$
	$I_2(M_1, M_2)$	$I_2(M_1, P_2)$, $I_2(M_2, P_2)$
	$I_3(P_1, M_2)$	$I_3(M_1, P_2)$
	$I_4(P_1, P_2)$	$I_4(M_1, P_2)$, $I_4(M_2, P_2)$
	$I_5(P_1, P_2)$	$I_1(P_1, P_2)$
three errors (positions)	$I_2(M_1, M_2, P_1)$	$I_4(M_1, M_2, P_2)$
	$I_3(M_2, P_1, P_2)$	$I_3(M_1, M_2, P_2)$
	$I_1(M_1, M_2, P_2)$, $I_1(M_1, M_2, P_1)$	$I_1(M_2, P_1, P_2)$, $I_1(M_1, P_1, P_2)$
four errors	I_2	

TABLE I
ANALYSIS OF PREFERENCES.

15 cases appear in each of the two columns. There is one extra case in favor of minimaxing in the constellation with no evaluation errors, and another case with the maximum number of errors, which are compensated by two additional cases with two evaluation errors each on the right side of Table I. Moreover, the cases in favor of minimaxing where the draw value falls in one of the intervals I_2 or I_4 have counterparts for product propagation with the other intervals, I_4 and I_2, respectively.

These asymmetries lead to an interesting result for the special case that the heuristic values have a uniform error distribution. Under this constellation, only the two extreme cases favoring minimaxing and the two average cases favoring product propagation, as well as the difference between the number of cases falling in either of the intervals I_2 and I_4 remain as decisive factors, whatever the distribution of values in a game tree is. For the trivial instance of board evaluator competence of exactly 50 percent. that is, guessing, these cases are leveled out completely. For the general case of constant error probabilities, the comparison between product propagation and minimaxing can be nailed down analytically to a simple difference, which makes the compensative factors favoring the competing back-up rules quite evident.

By putting together all the cases listed in Table I, substantial simplifications can be made. To start with, each pair of cases, one favoring minimaxing and the other favoring product propagation, both being associated with the same number of errors and falling in the same interval cancel themselves out. For example, there are two cases with exactly 1 error in interval I_3 — see line 3 in Table I —, one favoring minimaxing (the error occurs in position M_2), the other favoring product propagation (the error occurs in position P_2).

This leaves us with only eight remaining cases, half of which related to interval I_2, the other half related to interval I_4. For each of these subsets, there are two cases favoring minimaxing and two cases favoring product propagation, and the only difference lies in the number of errors associated with the complementing cases. More precisely, for the cases falling in interval I_4, those associated with zero or one error are in favor of minimaxing, those associated with two or three errors are in favor of product propagation. Conversely, for the cases falling in interval I_2, those associated with three or four errors are in favor of minimaxing, those associated with one or two errors are in favor of product propagation. The computation of the relative occurrence of these cases is done on the basis of p, the probability of error of the heuristic values. Each case comprises four positions, each of which may be erroneously evaluated (with probability p) or correctly (with probability $1-p$). The total frequency for a case amounts then to $p^n * (1-p)^{(4-n)}$ for exactly n positions being erroneously evaluated.

Thus, a calculation of the compensative benefits of the competing back-up rules, separately done for intervals I_4 and I_2 looks as follows (positive values are in favor of minimaxing, negative ones in favor of product propagation in (2), and vice versa in (3)):

$$I_4 : ((1-p)^4 + (1-p)^3 * p - (1-p)^2 * p^2 - (1-p) * p^3) \quad (2)$$

$$I_2 : ((1-p)^3 * p + (1-p)^2 * p^2 - (1-p) * p^3 - p^4) \quad (3)$$

For example, the first term $(1-p)^4$ in (2) expresses the case of no error (first line, left column in Table 1). Moreover, the term $(1-p)^3 * p$ appears in both (2) and (3). It represents the cases of exactly 1 error, both in line 2 in Table I (in the left column, in the scope of interval I_4, favoring minimaxing, in the right column in the scope of interval I_2, favoring product propagation).

The expressions in formulas 2 and 3 both contain a common factor $g(p) = (1-p)^3 + (1-p)^2 * p - (1-p) * p^2 - p^3$, so that (2) can be rewritten as $(1-p) * g(p)$ and (3) as $p * g(p)$.

Let $N(I_4)$ and $N(I_2)$ be the numbers of positions falling in intervals I_4 and I_2, respectively. If $(1-p) * N(I_4) > p * N(I_2)$ holds, minimaxing is superior to product propagation, otherwise the opposite is the case.

Apparently, the comparison depends on two complementary factors: the error probability, and the relation between the number of cases falling in intervals I_2 and I_4. Assessing the impact of the error probability is simple: the better the board evaluator is, the better this is for minimaxing. Assessing the impact of the relation between the number of cases falling in intervals I_2 and I_4 involves the following argument: In order for Product to make a move decision different from that of Minimax, the interval I_2 must be larger than I_4, otherwise $f'(P_1) * f'(P_2)$ will not be greater than $f'(M_1) * f'(M_2)$. Consequently, there will be more cases in which the draw value falls in interval I_2 than this is the case for interval I_4. Consequently, product propagation is superior to minimaxing for evaluators that are not too good, that is p does not get too low, so that $N(I_2) - N(I_4)$ is the decisive factor.

In order to assess which of the two complementary factors is larger and under what circumstances, we have to make concrete assumptions about the heuristic values in a search tree, in terms of their relative frequencies and dependencies within the tree structure. Since uniform error distribution is considered to be an unrealistic assumption in many game trees, we have also investigated the case of non-uniform error distribution. Instead of making arbitrary assumptions in this respect, we used the game-tree model of [14], since it appears to reflect the properties in games like chess better than other models. (More precisely, we used the function w_c from there with $c = 4$.) This tree model assumes a maximum increment value at each step, with all possible increment values occurring with the same frequency.

On the basis of this model, we have computed all possible search trees for depth 2, thereby varying the probability of error of the heuristic evaluation function systematically. It turned out that it requires the board evaluator to produce errors of 4.8% or less only, to make minimaxing competitive. If the error is greater than 4.8%, product propagation gives better results, while minimaxing does so for smaller error rates. This error rate corresponds according to our analysis to the following proportion of interval sizes: the number of cases where the draw value falls in interval I_2 is almost 20 times as much as the corresponding number for interval I_4, when using this game tree model. So, a small error rate is required to compensate for this disproportion, which makes product propagation superior for realistic board evaluators.

We have also let Minimax and Product simulate games in trees according to this model. For the given parameters, 3,886 critical cases result, in 1.579 percent of which Product wins more often than Minimax. Other parameters lead to similar results.

From these results, we got some evidence that Product has a slight but systematic advantage for depth 2 under realistic assumptions in game trees.

IV. Conclusion

In this paper, we reconsider product propagation as a candidate for backing-up heuristic values in game trees. Since this back-up rule does not allow for pruning algorithms as minimaxing does, however, especially the respective decision quality with minimal search is of interest. We provide the first comprehensive analysis of all the cases possible in a depth 2 search.

This systematic analysis confirms analytically the previous conjecture and empirical observation that a better evaluation function favors minimaxing in comparison to product propagation. More precisely, it provides a proof for uniform error distribution only, but independently from any specific tree model.

Our analysis also reveals yet unknown properties. In particular, it shows that product propagation tends to make better decisions than minimaxing more frequently, under realistic assumptions modeled after real game-playing programs.

As a consequence from this theoretical investigation, the decision quality of product propagation may still make it a viable alternative to minimaxing for game trees where only shallow searches are affordable.

References

[1] H. Kaindl, "Searching to variable depth in computer chess," in *Proc. International Joint Conference on Artificial Intelligence 1983*, Karlsruhe, August 1983, pp. 760–762.

[2] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.

[3] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "CHINOOK: The world man-machine checkers champion," *AI Magazine*, vol. 17, no. 1, pp. 21–29, 1996.

[4] J. Pearl, "On the nature of pathology in game searching," *Artificial Intelligence*, vol. 20, no. 4, pp. 427–453, 1983.

[5] D. Nau, "Pathology on game trees: a summary of results," in *Proc. First National Conference on Artificial Intelligence*, Stanford, CA, 1980, pp. 102–104.

[6] D. Beal, "An analysis of minimax," in *Advances in Computer Chess 2*, M. Clarke, Ed. Edinburgh University Press, 1980, pp. 103–109.

[7] J. Slagle and P. Bursky, "Experiments with a multipurpose, theorem-proving heuristic program," *Journal of the ACM*, vol. 15, no. 1, pp. 85–99, 1968.

[8] D. Nau, "Pathology on game trees revisited, and an alternative to minimaxing," *Artificial Intelligence*, vol. 21, no. 1–2, pp. 221–244, 1983.

[9] D. Nau, P. Purdom, and H. Tzeng, "Experiments on alternatives to minimax," *International Journal of Parallel Programming*, vol. 15, no. 2, pp. 163–183, 1986.

[10] D. Nau, "On game graph structure and its influence on pathology," *International J. Computer and Info. Sciences*, vol. 12, no. 6, pp. 367–383, 1983.

[11] P. Chi and D. Nau, "Comparison of minimax and product back-up rules in a variety of games," in *Search in Artificial Intelligence*, L. Kanal and V. Kumar, Eds. New York: Springer-Verlag, 1988, pp. 450–471.

[12] H. Kaindl, R. Shams, and H. Horacek, "Minimax Search Algorithms with and without Aspiration Windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 13, no. 12, pp. 1225–1235, December 1991.

[13] H. Kaindl and A. Scheucher, "Back-up of heuristic values: Minimaxing vs. product propagation," in *Proc. Thirteenth European Conference on Artificial Intelligence (ECAI-98)*, 1998, pp. 665–669.

[14] A. Scheucher and H. Kaindl, "Benefits of using multivalued functions for minimaxing," *Artificial Intelligence*, vol. 99, no. 2, pp. 187–208, 1998.