SRC-TR 87-104

Comparing Minimax and Product in a Variety of Games

by

Ping-Chung Chi
Dana S. Nau

COMPARING MINIMAX AND PRODUCT IN A VARIETY OF GAMES

Ping-Chung Chi[1]
Computer Science Department

Dana S. Nau[2]
Computer Science Department and Institute for Advanced Computer Studies

University of Maryland
College Park, Md 20742

ABSTRACT

In applications of game tree searching, the most widely used back-up rule is the minimax rule. Research has shown that minimax does not perform well in some games—and this has led to interest in alternatives to minimax, such as the product rule.

This paper describes studies on a wide variety of games, including P-games, G-games, three-hole kalah, othello, and Ballard's incremental game. The results of these studies are as follows:

1. The product rule outperforms minimax in three-hole kalah. This is a remarkable result, since it is the first widely known game in which product has been found to be better than minimax.
2. The relative performance of minimax and product is related to a parameter called the rate of heuristic flaw (rhf). Thus, rhf has potential use as a predictor of when one should use product as an alternative to minimax.

Paper type: Science
Primary topic: Automated Reasoning
Sub-topic: Search

Address all correspondence to:
Dana S. Nau
Computer Science Dept.
University of Maryland
College Park, MD 20742
(301) 454-7932
dsn@mimsy.umd.edu

---

[1] This work has been supported in part by a Systems Research Center fellowship.

[2] This work has been supported in part by the following sources: an NSF Presidential Young Investigator Award to Dana Nau, NSF NSFD CDR-85-00108 to the University of Maryland Systems Research Center, IBM Research, and General Motors Research Laboratories.

1. INTRODUCTION

The discovery of pathological games [6,4] has sparked interest in the possibility that various alternatives to the minimax back–up rule might be better than minimax. Pearl [9,10] has suggested that one should consider a back–up rule called the product rule. Nau, Purdom, and Tzeng [8] did some experiments and found that in a class of board splitting games, product almost always performed better than minimax and that the product rule avoided pathology.

In earlier experiments on the game of kalah, Slagle and Dixon [12] found that a back–up procedure called "M & N" performed significantly better than minimax. However, the M & N rule closely resembles minimax. Until recently, poor performance of minimax relative to back–up rules significantly different from minimax has not been observed in more commonly known games.

Underlying the above issues is a more fundamental question: Why does minimax perform well in many games, but poorly in some others? This paper investigates a possible answer to this question. We have obtained following results:

(1) For a wide variety of games, a parameter called the rate of heuristic flaw appears to be a good predictor of how well minimax performs against the product rule. These games include three–hole kalah, othello, P–games, G–games, and possibly others. This suggests that rhf may serve not only as a guideline for whether it will be worthwhile to consider alternatives to minimax, but also as a way to relate other characteristics of game trees to the performance of minimax and other back–up rules.

(2) In studies of three–hole kalah, the product rule generally performed better than minimax. This is the first widely known game in which product has been found to be better than minimax—and it suggests the possibility of exploiting non–minimax back–up rules to achieve better performance in other games.

2. DEFINITIONS

By a *game*, we mean a two person, zero sum, perfect information game having a finite game tree. All of the games studied in this paper (othello, kalah, P–games, G–

games, and Ballard's incremental game) satisfy this restriction.

Let G be a two player, zero sum, perfect information game, whose players are called *max* and *min*. To keep the discussion simple, we assume that G has no ties, but this restriction could easily be removed. If n is a board position in G, let u(.) be the utility function defined as

$$u(n) = \begin{cases} 1 & \text{if n is a forced win node} \\ 0 & \text{if n is a forced loss node.} \end{cases}$$

Since evaluation functions are intended to estimate the utility values of nodes in a game, we consider an evaluation function to be a function from the set of all possible positions in G into the closed interval [0,1]. If e is an evaluation function and n is a node of G, then the higher the value e(n), the better n looks according to e. We assume that every evaluation function produces perfect results on terminal game positions. In other words, if n is a terminal node of G, then e(n) = u(n).

If m is a node of G, then the depth d minimax value of m is

$$M(m,d) = \begin{cases} e(m) & \text{if depth(m)=d or m is a terminal node} \\ \min\{M(n)\text{: n is a child of m}\} & \text{if min has the move at m} \\ \max\{M(n)\text{: n is a child of m}\} & \text{if max has the move at m,} \end{cases}$$

and the depth d product value of m is

$$P(m,d) = \begin{cases} e(m) & \text{if depth(m)=d or m is a terminal node} \\ \Pi\{M(n)\text{: n is a child of m}\} & \text{if min has the move at m} \\ 1-\Pi\{1-M(n)\text{: n is a child of m}\} & \text{if max has the move at m.} \end{cases}$$

Let G be a game tree, and let m and n be any two nodes chosen at random from a uniform distribution over the nodes at depth d of G. Let $better_e(m,n)$ (and $worse_e(m,n)$) be whichever of m and n looks better (or worse, respectively) according to e. Thus if e(m) > e(n), then $better_e(m,n) = m$ and $worse_e(m,n) = n$. If e(m) = e(n), we still assign values to $better_e(m,n)$ and $worse_e(m,n)$, but the assignment is at random, with the following two possibilities each having probability 0.5:

(1) $better_e(m,n) = m$ and $worse_e(m,n) = n$

(2) $better_e(m,n) = n$ and $worse_e(m,n) = m$.

Since e may make errors, exhaustive search of the game tree may reveal that $better_e(m,n)$ is worse than $worse_e(m,n)$, i.e., that

$$u(better_e(m,n)) < u(worse_e(m,n)).$$

In this case, the evaluation function has failed to give a correct opinion about m and n. An event like this is called a heuristic flaw at depth d. The rate of heuristic flaw at depth d, denoted by rhf(d), is defined to be the quantity

$$Pr[u(better_e(m,n)) < u(worse_e(m,n))].$$

3. THEORETICAL CONSIDERATIONS

3.1. When Rhf is Low

Consider a minimax game tree search terminating at depth d of a game tree. If rhf(d) is small, it is intuitively apparent that this search should perform quite well. The question is whether it will perform better than some other back–up rule.

For simplicity, assume that the game tree is binary. Assume further that it is max's move at some node c, and let m and n be the children of c. Let d be the depth of m and n. Then

$$\begin{aligned} (1) \quad Pr[u(c)=1] &= Pr[u(better_e(m,n))=1 \text{ or } u(worse_e(m,n))=1] \\ &= Pr[u(better_e(m,n))=1] + Pr[u(worse_e(m,n)) > u(better_e(m,n))] \\ &\approx Pr[u(better_e(m,n))=1] + rhf(d). \end{aligned}$$

The smallest possible value for rhf(d) is zero. If rhf(d) is close to zero, then from (1) we have

$$Pr[u(c)=1] \approx Pr[u(better_e(m,n))=1].$$

which says that the utility value of c is closely approximated by the utility value of its best child. But according to the minimax rule, the minimax value of c is the minimax value of the best child. This suggests that in this case one might prefer the minimax back–up rule to other back–up rules.

More specifically, let us compare the minimax rule to the product rule, in the extreme case where rhf(d)=0. In this case, whenever m and n are two nodes at depth d of G,

$$\Pr[u(better_e(m,n)) < u(worse_e(m,n))] = 0.$$

Therefore, since there are only a finite number of nodes at depth d, there is a value $k \in (0,1)$ such that for every node m at depth d,

$$u(m) = 1 \text{ if and only if } e(m) \geq k.$$

By mathematical induction, it follows that forced win nodes will always receive minimax values larger than forced loss nodes, so a player using a minimax search will play perfectly.

But if the search is a product rule search rather than a minimax search, then the search will not always result in perfect play. For example, suppose d=2 and k=0.5, and consider the tree shown in Figure 1. Node n1 is a forced loss node because it is a max node whose children are forced loss nodes, and node n2 is a forced win node because it is a max node having a forced win child. However, P(n1,1) > P(n2,1), so a player using the product rule would make an incorrect choice of move at node n.

The above comparison suggest that when rhf is small, the minimax rule should perform better than the product rule.

3.2. When Rhf is Large

Let m and n be any two nodes at depth d. In general, rhf can take on any value between 0 and 1. But if e is a reasonable evaluation function, and if $better_e(m,n)$ is a forced loss, then it should make it more likely that $worse_e(m,n)$ is also a forced loss. Thus, we assume that

$$\Pr[u(worse_e(m,n))=1 \mid u(better_e(m,n))=0] \leq \Pr[u(worse_e(m,n))=1].$$

Thus since u(.) must be either 0 or 1,

$$\begin{aligned} rhf &= \Pr[u(worse_e(m,n))=1 \ \& \ u(better_e(m,n))=0] \\ &\leq \Pr[u(better_e(m,n))=0] \ \Pr[u(worse_e(m,n))=1]. \end{aligned}$$

Suppose rhf is large, i.e.,

$$\text{rhf} \approx \Pr[u(\text{better}_e(m,n))=0]\ \Pr[u(\text{worse}_e(m,n))=1].$$

Then from (1),

$$\begin{aligned}\Pr[u(c)=1] \approx\ & \Pr[u(\text{better}_e(m,n))=1] \\ & + \Pr[u(\text{better}_e(m,n))=0]\ \Pr[u(\text{worse}_e(m,n))=1].\end{aligned}$$

Thus, if $e(\text{better}_e(m,n))$ and $e(\text{worse}_e(m,n))$ are good approximations of $\Pr[u(\text{better}_e(m,n))=1]$ and $\Pr[u(\text{worse}_e(m,n))=1]$, then

$$\begin{aligned}\Pr[u(c)=1] &\approx e(\text{better}_e(m,n)) + (1 - e(\text{better}_e(m,n)))\ e(\text{worse}_e(m,n)) \\ &= 1 - (1 - e(\text{better}_e(m,n)))\ (1 - e(\text{worse}_e(m,n))),\end{aligned}$$

which is precisely the formula for the product rule given in Section 2. This suggests that when rhf is large, the product rule might be preferable.

More specifically, consider an evaluation function e which returns correct values on terminal nodes, but on nonterminal nodes returns values that are completely unindicative of the true value of a node. This would happen if e always returned the same value (say, 0.5), or if e returned independent identically distributed random values. If e is used in games where the branching factor is not constant, the product rule will tend to choose nodes where one has a wider choice of moves than one's opponent. From this, it can be shown that the product rule will do slightly better than the minimax rule in a wide variety of games.

The above arguments are by no means conclusive, but they suggest that in a number of games, the minimax rule might perform worse than the product rule when rhf is large.

4. EMPIRICAL CONSIDERATIONS

The arguments given in Section 3 suggest that minimax should do better against product when rhf is low than it does when rhf is high. To test this conjecture, we have examined the following games: G-games, Ballard's incremental game, othello, P-games, and kalah. The results of this examination are discussed below.

4.1. G–Games

A G–game is a board–splitting game which was constructed to reveal the relationship between pathology and game graph structure [7]. For G–games it is easy to compute a perfect evaluation of each node, but in [7], two less–than–perfect evaluation functions e_1 and e_3 were used to compare the performance of minimax and product. The product rule did better than minimax when e_1 was used, and product did worse than minimax when e_3 was used.

For our purposes, the significance of this study is this: it can be proven that for every depth d, rhf(d) is higher using e_1 than it is using e_3. Thus, on G–games, product performs better against minimax when using the evaluation function having the higher rhf. This matches our conjecture.

4.2. Ballard's Experiments

Ballard [2] used a class of incremental games with uniform branching factor to study the behavior of minimax and non–minimax back–up rules. One of the non–minimax back–up rules was a weighted combination of the computational schemes used in the minimax and product rules. Among other results, he claimed that "lowering the accuracy of either max's or min's static evaluations, or both, serves to increase the amount of improvement produced by a non–minimax strategy." Since low accuracy is directly related to a high rhf, this would seem to support our conjecture. But since Ballard did not test the product rule itself, we cannot make a conclusive statement.

4.3. Othello

Teague [13] did experiments on the game of othello, using both a "weak evaluation" and a "strong evaluation." The weak evaluation was simply a piece count, while the strong one incorporated more knowledge about the nature of the game. According to Teague's study, minimax performed better than product 82.8% of the time with the strong evaluation, but only 63.1% of the time with the weak evaluation.

It would be difficult to measure the rhf values for othello, because of the immense

computational overhead of determining whether or not playing positions in othello are forced wins. However, since rhf is a measure of the probability that an evaluation function assigns forced win nodes higher values than forced loss nodes, it seems clear that the stronger an evaluation function is, the lower its rhf value should be. Thus, Teague's results suggest that our conjecture is true for the game of othello.

4.4. P–Games

A P–game is a board–splitting game whose game tree is a complete binary tree with random independent assignments of "win" and "loss" to the terminal nodes. P–games have been shown to be pathological when using a rather obvious evaluation function e1 for the games [5]—and in this case, the minimax rule performs more poorly than the product rule [8]. However, pathology in P–games disappears when a stronger evaluation function is used [1].

It can be proven that the stronger evaluation function (which we will call e2) has a lower rhf than e1. Both evaluation functions return values between 0 and 1, and the only difference between e1 and e2 is that e2 can detect certain kinds of forced wins and forced losses (in which case it returns 1 or 0, respectively).

Let m and n be any two nodes. If $e2(better_{e2}(m,n)) = 0$, then it must also be that $e2(worse_{e2}(m,n)) = 0$. But it can be shown that $e2(x) = 0$ only if x is a forced loss. Thus $u(worse_{e2}(n, m))=0$, so there is no heuristic flaw. It can also be shown that $e2(x) = 1$ only if x is a forced win. Thus if $e2(better_{e2}(m,n)) = 1$, then $u(better_{e2}(m,n))=1$, so there is no heuristic flaw.

Analogous arguments hold for the cases where $e2(worse_{e2}(m,n)) = 0$ or $e2(worse_{e2}(m,n)) = 1$.

The cases described above are the only possible cases where e2 returns a different value from e1. No heuristic flaw occurs for e2 in any of these cases, but heuristic flaws do occur for e1 in many of these cases. Thus, the rhf for e2 is less than the rhf for e1.

We tested the performance of minimax against the product rule using e1 and e2,

in binary P–games of depths 9, 10, and 11, at all possible search depths.[3] For each combination of game depth and search depth, we examined 3200 pairs of games. The study showed that for most (but not all) search depths, minimax performed better against product when the stronger evaluation function was used (for example, the following table shows the results for P–games of depth 11). Thus, this result supports our conjecture.

Search depth	% wins for minimax using e1	% wins for minimax using e2
2	51.0%	52.1%
3	52.5%	51.8%
4	49.9%	50.3%
5	50.7%	49.3%
6	46.2%	48.1%
7	46.7%	48.4%
8	44.9%	48.6%
9	47.2%	50.0%

4.5. Kalah

Slagle [11] states that "Kalah is a moderately complex game, perhaps on a par with checkers." A detailed description of the game of kalah can be found in [11].

If a smaller–than–normal kalah playing board is used, the game tree is small enough that one can search all the way to the end of the game tree. This allows one to determine whether a node is a forced win or forced loss—and thus rhf can be estimated by measuring the number of heuristic flaws that occur in a random sample of games. By playing minimax against product in this same sample of games, information can be gathered about the performance of minimax against product as a function of rhf. To get a smaller–than–normal playing board, we used three–hole kalah (i.e., a playing board with three bottom holes instead of the usual six), with each hole containing at most six stones.

One obvious evaluation function for kalah is the "kalah advantage" used by

[3] As is usual in studies of P–games, we used a probability of win for each terminal node of $\xi = 0.382$ [5].

Slagle [11]. We let e_a be the evaluation function which uses a linear scaling to map the kalah advantage into the interval [0,1]. [4] Evaluation functions more accurate than e_a can be obtained by using the product value P(m,2) based on e_a. Weighted averages of $e_a(m)$ and P(m,2) can be used to get evaluation functions with different rhf values:

$$e_a^w(m) = w\ e_a(m) + (1-w)\ P(m,2),$$

for w between 0 and 1. We measured the value of rhf, and compared the performance of minimax against product, using the following values for w: 0, 0.5, 0.95, and 1.

Using a method for generating random games similar to the method used by Slagle, we generated 1000 initial game boards for three-hole kalah. As discussed above, we used these boards both to measure rhf, and to compare the performance of minimax and product. Because of computational limitations, we only measured rhf at depth 4, and only played minimax against product using a search depth of 2. The results are summarized in the following table.

w	rhf(4)	% games won by product	% games won by minimax
1	0.135	63.4%	36.6%
0.95	0.1115	55.5%	44.5%
0	0.08	53.6%	46.4%
0.5	0.0765	51.2%	48.8%

Three observations can be made about these results:

(1) Product performs better with all four evaluation functions used. This suggests that product might be of practical value in kalah and other games.

(2) For the evaluation functions tested, the lowest rhf was obtained with w = 0.5. This suggests that a judicious combination of direct evaluation with tree search might do better than either individually. This idea needs to be investigated more fully.

[4] In a preliminary study reported in [3], we played minimax against the product rule in three-hole kalah and two modifications of kalah, using e_a. This study used a somewhat different definition of rhf than the one used here. The evaluation function had a different rhf value in each of the three games, and minimax did better in the game in which rhf was the lowest. This result motivated the more extensive studies reported in the current paper.

(3) The performance of product against minimax increases as rhf increases. This matches our conjecture about the relation between rhf and the performance of minimax and product.

5. P–GAMES WITH VARYING RHF

Section 4 shows that in a variety of games, minimax performs better against product when rhf is low than it does when rhf is high. But Section 4 does not give much idea of the specific relationship between rhf and performance of minimax versus product.

To address this problem, we did a Monte Carlo study of the performance of minimax against product on binary P–games, using an evaluation function whose rhf could be varied easily. For each node n, let r(n) be a random value, uniformly distributed over the interval [0,1]. The evaluation function e^w is a weighted average of u and r:

$$e^w(n) = w\ u(n) + (1-w)\ r(n).$$

When the weight $w = 0$, e^w is a completely random evaluation. When $w = 1$, e^w provides perfect evaluations. Furthermore, w has a straightforward relation to rhf: for all d,

$$\text{rhf}(d) = \begin{cases} \left(\frac{1-2w}{1-w}\right)^2 (1-\xi)\xi & \text{if } 0 \le w \le 0.5 \\ 0 & \text{if } 0.5 < w \le 1. \end{cases}$$

For $0 \le w \le 0.5$, this relationship is approximately linear, as shown in Figure 2. Furthermore, for $w \ge 0.5$, rhf = 0 (i.e., the evaluation function gives perfect performance with the minimax back–up rule).

In the Monte Carlo study, w was varied between 0 and 0.5 in steps of 0.01. For each value of w, minimax was played against product in 8000 pairs of binary P–games

of depth 6, with each player searching to depth 2.[5] The results of this study are shown in Figure 3, which graphs the fraction of games won by minimax against product, as a function of rhf. Figure 3 shows that minimax does significantly better than product when rhf is small, and product does significantly better than minimax when rhf is large.[6] Thus, in a general sense, Figure 3 supports our conjecture about rhf. But Figure 3 also demonstrates that the relationship between rhf and the performance of minimax against product is not always monotone, and may be rather complex.

6. CONCLUSIONS AND SPECULATIONS

The studies in this paper point out relationships between various characteristics of games and the performance of back–up rules in these games. They also suggest the possibility of improving the existing evaluation functions and back–up rules. The results are summarized below:

(1) Theoretical considerations suggest that for evaluation functions with low rhf values, minimax should perform better against product than it does when rhf is high. Our investigations on a variety of games, including G–games, Ballard's game, othello, P–games, and kalah, confirm this conjecture.

(2) The product rule plays better than minimax in the game of kalah with three bottom holes. This is a remarkable result, since it is the first widely known game in which product has been found to be better than minimax.

Previous investigations have proposed two hypotheses for why minimax might perform better in some games than in others: dependence/independence of siblings [5] and detection/non–detection of traps [10]. At first glance, these two hypotheses would appear to have little to do with each other. However, since sibling dependence generally makes rhf lower and early trap detection always makes rhf lower, these two characteristics have more to do with each other than has previously been realized.

One could argue that for most real games it may be computationally intractable

[5] i.e., 8000 randomly chosen initial playing boards, with each player having a chance to move first.

[6] Furthermore, the poor performance of minimax when rhf is large corroborates previous studies which showed that product did better than minimax in P–games using a different evaluation function [8].

to measure rhf, since one would have to search the entire game tree. But since rhf is closely related to the strength of an evaluation function, one can generally make intuitive comparisons of rhf for various evaluation functions without searching the entire game tree. This becomes evident upon examination of the various evaluation functions discussed earlier in this paper.

There are several problems with the definition and use of rhf. Since it is a single number, rhf is not necessarily an adequate representation for the behavior we are trying to study. Furthermore, since the definition of rhf is tailored to the properties of minimax, it is not necessarily the best predictor of the performance of the product rule. As a result of this, the relationship between rhf and the performance of minimax versus product can be rather complex (as was shown in Section 5). Further study might lead to better ways of predicting the performance of minimax, product, and other back–up rules.

REFERENCES

[1] Abramson, B., A Cure for Pathological Behavior in Games that Use Minimax, *First Workshop on Uncertainty and Probability in Artificial Intelligence*, 1985.

[2] Ballard, B. W., Non–Minimax Search Strategies for Minimax Trees: Theoretical Foundations and Empirical Studies, Tech. Report, Duke University, Durham, NC, July 1983.

[3] Chi, P. and Nau, D. S., Predicting the Performance of Minimax and Product in Game Tree Searching, *Second Workshop on Uncertainty and Probability in Artificial Intelligence*, 1986.

[4] Nau, D. S., Pathology on Game Trees: A Summary of Results, *Proc. First Annual National Conference on Artificial Intelligence*, Stanford, CA, pp. 102–104, 1980.

[5] Nau, D. S., An Investigation of the Causes of Pathology in Games, *Artificial Intelligence* **19**, pp. 257–278, 1982. (This paper was abstracted in *Zentralblatt fuer Mathematik* in 1983.)

[6] Nau, D. S., Decision Quality as a Function of Search Depth on Game Trees, *Journal of the ACM* **30**, 4, pp. 687–708, Oct. 1983. (An early version is available as Tech. Report TR–866, Computer Sci. Dept., Univ. of Maryland, Feb. 1980.)

[7] Nau, D. S., On Game Graph Structure and Its Influence on Pathology, *International Journal of Computer and Information Sciences* **12**, 6, pp. 367–383, 1983. (An early version is available as Tech. Report TR–1246, Computer Sci. Dept., Univ. of Maryland, Feb. 1983.)

[8] Nau, D. S., Purdom, P. W., and Tzeng, C. H., An Evaluation of Two Alternatives to Minimax, *First Workshop on Uncertainty and Probability in Artificial Intelligence*, 1985.

[9] Pearl, J., Heuristic Search Theory: Survey of Recent Results, *Proc. Seventh Internat. Joint Conf. Artif. Intel.*, Vancouver, Canada, pp. 554–562, Aug. 1981.

[10] Pearl, J., *Heuristics,* Addison–Wesley, Reading, MA, 1984.

[11] Slagle, J. R. and Dixon, J. K., Experiments with Some Programs that Search Game Trees, *JACM* **16**, 2, pp. 189–207, April 1969.

[12] Slagle, J. R. and Dixon, J. K., Experiments with the M & N Tree–Searching Program, *CACM* **13**, 3, pp. 147–154, March 1970.

[13] Teague, A. H., Backup Rules for Game Tree Searching: A Comparative Study, Master's Thesis, University of Maryland, College Park, MD, 1985.

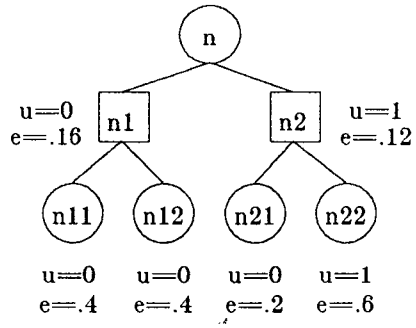

FIGURE 1: A case where product makes the wrong choice.

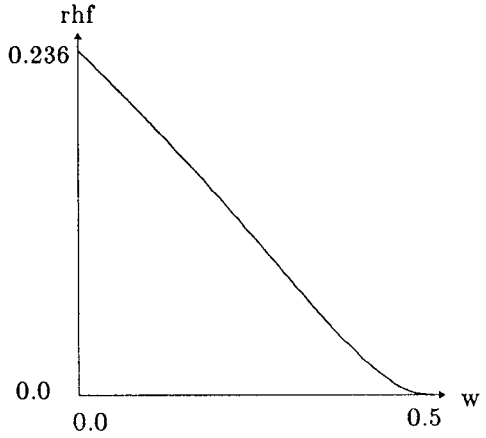

FIGURE 2: The relationship between w and rhf for e^{w}.

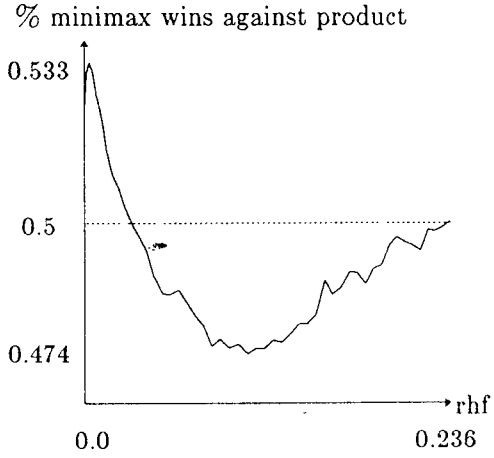

FIGURE 3: Performance of minimax against product using e^{w} as rhf varies.