A Model of Two-Player Evaluation Functions[1]

Bruce Abramson[2] and Richard E. Korf[3]

Abstract

We present a model of heuristic evaluation functions for two-player games. The basis of the proposal is that an estimate of the *expected-outcome* of a game situation, assuming random play from that point on, is an effective heuristic function. The model is supported by three distinct sets of experiments. The first set, run on small, exhaustively searched game-trees, shows that the quality of decisions made on the basis of exact values for the expected-outcome is quite good. The second set shows that in large games, estimates of the expected-outcome derived by randomly sampling terminal positions produce reasonable play. Finally, the third set shows that the model can be used to automatically learn efficient and effective evaluation functions in a game-independent manner.

I. Introduction: The Problem

Heuristic search theorists have studied static evaluation functions in two settings: single-agent puzzles and dual-agent games. In single-agent domains, the task is typically to find a lowest cost path from the initial state to a goal state. The role of the heuristic evaluation function is to estimate the cost of the cheapest such path. This provides a rigorous definition of single-player evaluators, offers an absolute measure of heuristic quality (its accuracy as an estimator), allows any two functions to be compared (the more accurate estimator is the better heuristic), and has spawned a large body of results that relate evaluator accuracy to both solution quality and algorithmic complexity of heuristic searches.

Unfortunately, the meaning of heuristic evaluation functions for two-player games is not as well understood. Two-player evaluators are typically described as estimates of the "worth"[Nilsson, 1980], "merit", "strength"[Pearl, 1984], "quality"[Winston, 1977], or "promise"[Rich, 1983] of a position for one player or the other. The literature is uniformly vague in its interpretation of game evaluation functions. One popular school of thought contends that a static evaluator should estimate a node's actual minimax value, or the value that would be returned by searching forward from the given position all the way to the terminal nodes of the tree, labelling the leaves with their actual outcomes, and then minimaxing the leaf values back up to the original node. Under this definition, the best heuristic is the function that most accurately approximates the minimax value over all possible game positions. The difficulty with this proposal is that it provides no way of judging the quality of a heuristic, comparing two different evaluators, or learning a heuristic function, because actual minimax values can only be computed by exhaustively searching the entire game tree below the given node. In real games, this is a computationally intractable task for all but end-game positions.

Alternatively, the quality of a heuristic can be defined operationally by the quality of play that it produces. This definition allows any two heuristic functions to be compared by playing them against each other. There are two major drawbacks to this approach. First, it compares entire control strategies, not just evaluation functions. The quality of a program's play can be affected by a number of factors, including backup techniques (minimax is not necessarily optimal when the values are only estimates), and lookahead depth (the relative performance of two functions may be different at different depths), as well as evaluator strength. Second, comparitive studies fail to provide an absolute measure of the quality of a heuristic function.

We introduce a new model of two-player evaluators that resolves all of these difficulties. The *expected-outcome model*, described in section 2, provides a rigorous definition of an evaluator's objective, an absolute standard for gauging its accuracy, and a viable method for performing a priori comparisons. Section 3 outlines a series of experiments that shows that, at least in its most basic form, the model leads to reasonable play in real games. Some conclusions and directions for future research are then given in section 4.

[1] This research was supported in part by NSF Grant IST 85-15302, an NSF Presidential Young Investigator Award, an IBM Faculty Development Award, and a grant from Delco Systems Operations.

[2] Department of Computer Science, Columbia University, and Computer Science Department, University of California at Los Angeles

[3] Computer Science Department, University of California at Los Angeles

II. Expected-Outcome: The Proposed Model

In a broad sense, the purpose of an evaluation function in a two-player domain is to indicate whether a given node on the search frontier will result in a victory. The standard assumption, forwarded by proponents of approximating minimax values, has been that this corresponds to an estimate of the outcome that would be arrived at by perfect play. Our new model is based on a different set of assumptions. We view the actual outcome of a game as a random variable and investigate what the game's payoff would be, given random play by both sides. Although the assumption of random play seems unrealistic, it is important to recall that in a two-player game, evaluation functions are normally applied only at the frontier of the search. By definition, the frontier is the limit beyond which the program cannot gather any further data about the game tree, and in the absence of any other information, random play is the only practical assumption. Furthermore, there is a common belief that any player, including a random one, should find it easier to win from a "strong" position than from a "weak" one. Thus, a technique for determining strong positions for a random player may help indicate strong positions for a perfect one, as well. In any event, our approach stands in stark contrast to the usual one, and the question of its utility is primarily empirical, not intuitive.

Any effective evaluator designed under our assumptions should indicate the expected value of the outcome variable, or the *expected-outcome* of the given position.

Definition: *Expected-Outcome Values*
The expected-outcome value of a game-tree node, G, is given by a player's expected payoff over an infinite number of random completions of a game beginning at G, or

$$EO(G) = \sum_{leaf=1}^{k} V_{leaf} P_{leaf},$$

where k is the number of leaves in the subtree, V_{leaf} is a leaf's value, and P_{leaf} is the probability that it will be reached, given random play. It is important to note that P_{leaf} is not necessarily equal to $\frac{1}{k}$. The probability that a leaf will be reached is one over the product of its ancestors' branching factors; a node with no siblings is twice as likely to be reached as a node with one sibling. Leaves are only equiprobable in trees in which all nodes of equal depth are constrained to have identical branching factors, thereby making all paths equally likely.

Ignoring the issue of plausibility for a moment, this model has a number of attractive features. First, it is precise. Second, it provides an absolute measure of heuristic quality, (namely the accuracy with which it estimates the expected value), hence a means of directly comparing two heuristic functions. Finally, and most importantly, it provides a practical means of devising heuristics — expected values can be approximated by random sampling. Along with their many advantages, of course, expected values (and other statistical parameters) do bear a serious onus: they can be very misleading when population sizes are relatively small. Thus, care must be taken not to rely too heavily on expected-outcome values in end-game play.

Most interesting games generate trees that are too complex and irregular to be discussed analytically. Although it is possible to show that on trees with uniform branching factors and depths expected-outcome functions make optimal decisions, when the uniformity disappears, the guaranteed optimality is lost. Since the ultimate criterion by which an evaluator is judged is its performance in actual competition, we ran three sets of experiments to verify both the rationality of our assumptions and the strength of our model in real games. In the first set, we generated the complete game-trees of tic-tac-toe and 4-by-4 Othello, calculated the exact numbers of wins, losses, and draws beneath every position, and compared the exact expected-outcome function with a well-known standard evaluator for the same game. We found that the quality of the decisions made by expected-outcome was superior to that of the standard evaluators. While these results are encouraging, they are limited to games that are small enough to be searched exhaustively. In the second set of experiments, we used the full 8-by-8 game of Othello. Since this game is too large for exact values to be calculated, we estimated expected-outcome by averaging the values of a randomly sampled subset of the terminal positions beneath the given node. This estimated expected-outcome evaluation was pitted directly (no lookahead) against a standard evaluator, with the result that expected-outcome significantly outplayed the standard. Unfortunately, the cost of implementing the random sampler was prohibitive. In the final set, we attempted to produce an *efficient* estimator by performing a regression analysis on the expected-outcome estimates returned by the sampler, to automatically learn the coefficients in a polynomial evaluator for Othello. Once again, the results were positive: the learned coefficients played as well as a set of coefficients that had been designed by an expert, even though the learning procedure had no information about the game other than the rules and the values of terminal positions. Taken as a whole, this series of experiments offers strong empirical support for the expected-outcome model.

III. Supporting Evidence

One of the most attractive features of expected-outcome is its domain-independence. The model's reliance on nothing more than a game's rules and outcomes indicates that it should be equally applicable to all two-player games. In addition to being a source of great strength, however, this generality also makes the model somewhat difficult to test thoroughly. Different implementations on different games are quite likely to yield different results. This section describes a series of experiments that demonstrate the utility of expected-outcome to at least one class of games, those with finite-depth trees and outcomes drawn from $\{win, loss, draw\}$. The requirement of finite-depth trees simply means that the game will eventually terminate. Without this rule, a chess game could, at least in theory, continue indefinitely. Variants of two games that meet both requirements, tic-tac-toe and Othello, were selected for testing. Tic-tac-toe is a game that should be familiar to everyone; Othello, although of growing popularity, may not be. The standard game is played on an 8-by-8 board. The playing pieces are discs which are white on one side and black on the other. Each player, in turn, fills a legal vacant square with a disc showing his own color. Whenever the newly placed disc completes a sandwich consisting of an unbroken straight line of hostile discs between two friendly ones, the entire opposing line is captured and flipped to the color of the current mover. A move is legal if and only if at least one disc is captured. When neither player can move, the one with the most discs is declared the winner. (For a more detailed description, see [Frey, 1980] [Maggs, 1979] [Rosenbloom, 1982]).

A. Decision Quality

The first step in determining a model's theoretical accuracy is investigating its *decision quality*, or the frequency with which it recommends correct moves. In the case of expected-outcome, the question is how often the move with the largest (or smallest, as appropriate) percentage of win leaves beneath it is, in fact, optimal. Since optimal moves are defined by complete minimax searches, (searches that extend to the leaves), their calculation is contingent upon knowledge of the entire subtree beneath them. Thus, for this first set of experiments, fairly small games had to be chosen. Moreover, in order to compare the decision quality of expected-outcome with that of a more standard function, popular games (or variations thereof) were needed. Four games that met both requirements were studied, although only two of them, 3-by-3 tic-tac-toe and 4-by-4 Othello, have game-trees that are small enough to generate entirely. The other two, 4-by-4 tic-tac-toe and 6-by-6 Othello, were chosen because they are small enough for large portions of their trees to be examined, yet large enough to offer more interesting testbeds than their smaller cousins.

For each game studied, every node in the tree (beneath the initial configuration) was considered by four functions: complete-minimax, expected-outcome, a previously studied standard, and worst-possible-choice. The decisions recommended by these evaluators were compared with the optimal move, or the move recommended by minimax, and a record was kept of their performance. Minimax, by definition, never made an error, and worst-possible-choice erred whenever possible. Expected-outcome, unlike complete-minimax, did not back up any of the values that it found at the leaves; its decisions were based strictly on evaluations of a node's successors. Finally, the standard evaluators were taken from published literature and calculated using only static information: the open-lines-advantage for tic-tac-toe [Nilsson, 1980], and a weighted-squares function for Othello based on the one in [Maggs, 1979]. Open-lines-advantage is known to be a powerful evaluator; weighted-squares is less so. Nevertheless, its study does have scientific merit. Weighted-squares were the first reasonable expert-designed Othello functions, and the more sophisticated championship-level evaluators became possible in large part due to the feedback provided by their performance [Rosenbloom, 1982]. Since the purpose of these experiments was not to develop a powerful performance-oriented Othello program, but rather to test the decision quality of a new model of evaluation functions, a useful comparison can be provided by any well thought out game-specific function, albeit less-than-best.

The results of these experiments were rather interesting and quite positive. Without going into detail, their most significant feature was the evaluators' relative error-frequency — in tic-tac-toe, expected-outcome made roughly one-sixth as many errors as open-lines-advantage, and in Othello about one-third as many as weighted squares. The basic point made by these experiments is that in all cases tested, expected-outcome not only made fewer errors than the standard functions, but chose the optimal move with relatively high frequency. This indicates that guiding play in the direction of maximum win percentage constitutes a reasonable heuristic. Thus, the expected-outcome model has passed the first test: exact values generally lead to good moves.

B. Random Sampling Strategies

According to the the decision quality results, if complete information is available, moving in the direction of maximum win percentage is frequently beneficial. Unfortunately, these are precisely the cases in which optimal moves can always be made. Since probabilistic (and for that matter, heuristic) models are only interesting when knowledge is incomplete, some method of estimating expected-outcome values based on partial information is needed. The obvious technique is random sampling. Expected-outcome values, by their very definition, represent the means of leaf-value distributions. In the second set of experiments, a sampler-based estimate of expected-outcome was pitted against a weighted-squares function in several matches of (8-by-8) Othello. These experiments, like those which investigated decision quality, were designed as pure tests of evaluator strength — neither player used any lookahead. The aim of these tests, then, was to show that sampler-based functions can compete favorably with those designed by experts, at least in terms of their quality of play. As far as efficiency goes, there is no comparison. The sampler was fairly cautious in its detection of convergence to a value; many samples were taken, and as a result, the sampling player frequently required as much as an hour to make a single move [4]. The static function,

[4]Convergence was detected by first sampling N leaves and developing an estimate, then sampling an additional N and finding

on the other hand, never required more than two seconds. The time invested, however, was quite worthwhile: in a 50-game match, the sampler crushed its weighted-squares opponent, 48-2.

Veteran Othello players may feel that the number of victories alone is insufficient to accurately gauge the relative strength of two players. Perhaps of even greater significance is the margin of victory — the single most important feature in determining a player's USOA (United States Othello Association) rating [Richards, 1981]. Over the course of 50 games, the weighted-squares total of 894 discs was 1,079 shy of the 1,973 racked up by the sampler. A statistical analysis of the disc differentials indicates that the sampler should be rated roughly 200 points, or one player class, ahead of the weighted-squares player.

These studies show that, efficency considerations aside, sampler-based functions can compete admirably. It is important, however, to keep the results in their proper perspective. As a demonstration of the world's best Othello evaluator, they are woefully inadequate — the absence of lookahead makes the games unrealistic, the difference in computation times skews the results, and the competition is not as strong as it could be. Their sole purpose was to establish estimated expected-outcome as a function at least on par with those designed by experts, and the data clearly substantiates the claim. Expected-outcome functions, then, do appear to make useful decisions in interesting settings. Given no expert information, the ability to evaluate only leaves, and a good deal of computation time, they were able to play better than a function that had been hand-crafted by an expert. Thus the second challenge has been met, as well: in the absence of perfect information, an expected-outcome estimator made reasonably good decisions.

C. Learning Expected-Outcome Functions

Like most products, evaluation functions incur costs in two phases of their existence, design and implementation. The inefficiency of sampler-based functions is accrued during implementation; their design is simple and cheap, because an effective sampler need only understand the game's rules and be able to identify leaves. Static evaluators, on the other hand, rely on detailed game-specific analyses, frequently at the cost of many man-hours and/or machine-hours. To help reduce these design costs, a variety of automatic tools that improve static evaluators have been developed, the simplest of which attempt to determine the relative significance of several given game features. Techniques of this sort are called *parameter learning* [Samuel, 1963] [Samuel, 1967] [Christensen and Korf, 1986], and should be applicable to learning the relationship between game features and expected-outcome values. While this reliance on predetermined game features will inevitably limit conformity to the model's ideal, scoring polynomials are the backbone of most competitive game programs, and if done properly, the learned functions should combine the statistical precision and uncomplicated design of sampler-based functions with the implementation efficiency of static evaluators. The next set of experiments involved learning static expected-outcome estimators of just this sort.

To find a member of the weighted-squares family that estimates the expected-outcome value, a regression procedure was used to learn coefficients for the features identified by the original, expert-designed function. Since the exact expected-outcome value is not computable in interesting games, an estimated value had to be used as the regression's dependent variable. Thus, the value that was approximated was not the actual expected-outcome, but rather the estimate generated by the random sampler described in the previous section. The output of the regression led directly to the development of static estimators of the desired form. In addition, the statistical measures of relationship between the independent and dependent variables indicated that the selected game features are reasonable, albeit imprecise, estimators of expected-outcome. This is directly analogous to the assertion that weighted-squares functions can play up to a certain level, but for championship play, additional factors must be considered [Rosenbloom, 1982].

For the third, and final set of experiments, four members of the weighted-squares family of Othello evaluators were studied, two of expert design [5] and two learned by regression analysis. These evaluators differ only in the coefficients assigned to each of the game features. To ascertain the relative strength of the coefficient sets, a tournament was played. Unlike the functions studied in the decision quality and random sampling experiments, all four weighted-squares evaluators are efficiently calculable. This allowed the ban on lookahead to be lifted and more realistic games to be studied. The rules of the tournament were simple. Every pair of functions met in one match, which consisted of 100 games each with lookahead length fixed at 0, 1, 2, and 3. Between games, the players swapped colors. Over the course of 400 games, *no evaluator was able to demonstrate substantial superiority over any other.* Not only were the scores of all matches fairly close, but the disc differential statistics were, as well. An analysis of the victory margins shows that with probability .975, no two of the functions would be rated more than 35 USOA points apart. Since roughly 200 points (actually, 207 [Richards, 1981]), are necessary to differentiate between player classes, the rating spread is rather insignif-

another estimate. If the discrepancy between them was within the tolerable error bounds, the estimate was accepted. Otherwise, another $2N$ were sampled, and so on, until convergence was detected. For the sampler used in these experiments, the original sample size was $N = 16$ leaves, and the maximum needed was 1024.

[5] The first expert function was taken directly from [Maggs, 1979], while the second, which was also used in the previous section's random sampling experiments, modified the first to account for my personal experience.

icant — it should be clear that all four functions are essentially equivalent.

In addition to offering a method of comparing evaluator strength, disc differentials suggest another application of expected-outcome: assign each node a value equal to the expected disc-differential of the leaves beneath it. A fifth weighted-squares function was learned to estimate the expected-outcome of this multi-valued leaf distribution (all outcomes in the range $[-64, 64]$ are possible), and entered into the tournament. Its performance was noticeably stronger than that of the other functions, although not overwhelmingly so, with victory margins between 39 and 145, and ratings 25 to 85 points above its competitors.

Thus, the coefficients learned by the regression analysis procedure are at least as good as those designed by experts. Of course, it is possible to contend that a function's strength is derived primarily from its feature set, not its coefficient set. If this is true, any two members of the same family should perform comparably, and it's not surprising that the new functions competed favorably with the old. To dissipate any doubts that may arise along these lines, some further family members were generated. Each of the four evaluators in the initial tournament played an additional match against a weighted-squares cousin with a randomly generated set of coefficients. All four random functions were demolished — they rarely won at all, and would be rated at least a player class behind the four that had been intelligently designed. With its strong showing in the tournament, the expected-outcome model has met the third challenge: an effeciently calculable estimator played fairly well.

IV. Conclusions

Our proposed model of two-player evaluation functions, the expected-outcome model, suggests new directions for rethinking virtually every element of game programming. For example, in addition to the obvious benefits of a rigorous and practical definition for evaluators, the model implies a significantly different approach to the programming of two-player games. The standard Shannon Type-A program does a full-width search to a fixed depth and then estimates the values of the nodes at that depth [Shannon, 1950]. The program in the second set of experiments (random sampling) does a full-depth search but only of a subset of the nodes. In a Shannon type-A strategy, uncertainty comes from the estimates of the positions at the search horizon, whereas in our model, uncertainty is due to sampling error. Furthermore, the new model avoids one of the major disadvantages of all previous approaches, the need for a game-specific evaluation function based on a set of handcrafted, carefully tuned, ad hoc features. In sharp contrast to this reliance on outside expertise, the expected-outcome model requires only well-defined leaf values, the rules of the game, and a game-independent sampling strategy.

It is, of course, unreasonable to expect the initial implementation of any new model, regardless of inherent merit, to match the achievements of thirty-five years of progressive research. Whether expected-outcome will eventually replace minimax as the standard model for game design, or simply augment it by providing a degree of precision to some of its more ambiguous components, remains to be seen. What this paper has shown is that the estimation of expected-outcome functions defines a viable, domain-independent role for two-player evaluation functions. We believe that the new model warrants the serious further study that is currently in progress.

Acknowledgements

We would like to thank Othar Hansson, Andrew Mayer, Dana Nau, and Judea Pearl for providing us with helpful discussions and suggestions.

References

[Christensen and Korf, 1986] Jens Christensen and Richard Korf. A unified theory of heuristic evaluation functions and its application to learning. In *Proceedings of the fifth National Conference on Artificial Intelligence*, 1986.

[Frey, 1980] Peter W. Frey. Machine othello. *Personal Computing*, :89–90, 1980.

[Maggs, 1979] Peter B. Maggs. Programming strategies in the game of reversi. *BYTE*, 4:66–79, 1979.

[Nilsson, 1980] Nils J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Company, 1980.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison Wesley, 1984.

[Rich, 1983] Elaine Rich. *Artificial Intelligence.* McGraw Hill, 1983.

[Richards, 1981] R. Richards. The revised usoa rating system. *Othello Quarterly*, 3(1):18–23, 1981.

[Rosenbloom, 1982] Paul S. Rosenbloom. A world-championship-level othello program. *Artificial Intelligence*, 19:279–320, 1982.

[Samuel, 1963] A.L. Samuel. Some studies in machine learning using the game of checkers. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, McGraw-Hill, 1963.

[Samuel, 1967] A.L. Samuel. Some studies in machine learning using the game of checkers ii — recent progress. *IBM J. Res. Dev.*, 11:601–617, 1967.

[Shannon, 1950] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.

[Winston, 1977] P.H. Winston. *Artificial Intelligence.* Addison Wesley, 1977.