Thesis Proposal: The Expected-Outcome Model of Two-Player Games

CUCS - 230-86

Bruce Abramson
Department of Computer Science, Columbia University
Computer Science Department, University of California at Los Angeles *

November 2, 1986

Abstract

This paper introduces a new, crisp definition of two-player evaluation functions. These functions calculate a node's *expected-outcome* value, or the probability that a randomly chosen leaf beneath it will represent a win. The utility of these values to game programs will be assessed by a series of experiments that compare the performance of expected-outcome functions with that of some popular, previously studied evaluators. To help demonstrate the domain-independence of these new functions, the experiments will be run on variants of several games, including tic-tac-toe, Othello, and chess. In addition, the paper outlines a new probabilistic model of game-trees which involves rethinking many long-accepted assumptions in light of the newly defined expected-outcome functions.

*Current address:
Computer Science Department
University of California at Los Angeles
Los Angeles, California 90024

Contents

1 Introduction

This document is a proposal for my doctoral dissertation. The thesis underlying this work is that the proper model for the study of two-player games is probabilistic in nature, not deterministic. The standard model of minimaxing statically estimated values in game-trees dates back to 1950 [Sha50], and has served as the basis of nearly all game programs. One of the most conspicuous flaws of this model is the absence of a precise definition of what these static values estimate. Several attempts have been made to deal with this shortcoming, including the replacement of point-valued estimates with ranges [Ber79] or probability distributions [Pal85] that describe the likely locations of "actual" values, and the introduction of easily recognizable inexact goals that frequently correspond to the ultimate goal of winning the game [Bot84]. These approaches represent steps in the right direction in that they acknowledge the impossibility of estimating unspecified values, and concentrate instead on describing parameters of those values that are likely to be

useful. Even the suggestion of distribution-based evaluators falls short of the mark, however, because it, too, fails to provide the necessary useful definition [1]. This is a universal failure of two-player game models — the quantity being estimated by the static evaluator is vaguely defined, and there are no general guidelines as to how it should be estimated. Nevertheless, all of these models rely heavily on the *face-value principle*, the assumption that statically determined values precisely correspond to actual payoffs [Pea84].

Unlike the standard model, which starts by specifying a backup strategy and leaves the role of the evaluator undetermined, the model introduced here starts by defining the *expected-outcome* value of a node as the probability that random play from that node will result in victory. In addition to providing a precise role for the evaluator, the probabilistic nature of these values makes them fairly simple to estimate. Furthermore, this definition makes it possible to reassess virtually every component of two-player game theory; its implications and variants include new backup strategies and the development of an evaluation function that calculates the probability of victory assuming reasonable (not strictly random) play. Clearly, a complete analysis of all of its potential ramifications is beyond the scope of a single dissertation. The thrust of my research is to develop the basic model and some of its most immediate extensions, and lay the groundwork for future analyses. I hope to show that the expected-outcome model is elegant, general, useful, and powerful, that it constitutes a significant contribution to the fields of game design and heuristic analysis, and that it warrants serious further study.

The rest of the proposal proceeds as follows: Section 2 elaborates on some of the problems with standard approaches to two-player games and demonstrates the need for a new model. The basic expected-outcome model is described in section 3. In addition, a few of its implications are outlined. Section 4 contains a description of experimental work, some already completed and some still in progress, that will determine the strength of expected-outcome functions and their usefulness to actual game programs. Section 5 describes the anticipated contributions of this research. The two

[1] To resolve this problem, Palay introduced the concept of the *delphic* value, or "the value returned by an oracle when viewing the state in question using the same scale as used by the evaluation function". He admits, however, that no computational method is envisioned, and recommends extended minimax and expert consultation as methods of approximating delphic values.

appendices include a description of the standard evaluators studied and a brief outline of the experimental work, respectively.

2 Background: The Problem

One of the most widely applied problem-solving techniques in artificial intelligence is the heuristic search of state-space graphs. Many interesting problems can be represented as path-finding problems on large graphs, that is graphs with more nodes than can be examined in a reasonable amount of time [Poh70]. To find appropriate paths from an initial state to a goal state, then, some nodes must be ignored. Guidelines that indicate which nodes should be ignored and which should be examined are called heuristics, and the resulting systematic search of the graph is known as an heuristic search. The most desirable heuristics are easy to calculate, highly accurate, lead to good solutions, and are applicable to many problems. Heuristic search theory is, in large part, the study of necessary tradeoffs among these features and the design of heuristics that combine them in the desired proportions. One type of heuristic that has received a great deal of attention is static evaluation, or the estimation of a node's merits based solely on directly detectable features. In many domains, such as puzzles, an adversary (frequently nature) plays its hand before the game begins. The problem-solving agent controls the search and concentrates on finding the cheapest path to a goal state. The intuitive evaluation function in these systems is an estimate of the cost of that cheapest path. This type of evaluator has been successfully applied to numerous single-agent domains. In addition, the rigorous definition it provides has led to many analytic studies which compare the strength of any two such functions, relate a function's accuracy to the quality of the solution it generates, and determine the complexity of various algorithms [Pea84].

Most problems, however, can not be modeled effectively as single-agent searches. In particular, there is a fairly simple class of problems known as two-player zero-sum games of perfect information, (this class includes many popular parlor games such as chess, checkers, Othello, and Go), in which the two players are perfect adversaries and decisions made by one player must take the opposing player's possible responses into account. The goal in these domains is to win the game, and thus the intuitive definition of a two-player

evaluator is an estimate of whether a given node will result in a winning state. Unfortunately, no firm understanding of what this means has ever been developed, and few analytic studies have been performed to either define or investigate a function's accuracy. The remainder of this section illustrates how the failure to rigorously define the aim of two-player evaluators has led to an abundance of difficulties in two-player search systems and caused the majority of work done in the field to be rather ad hoc.

Static evaluators come into play in two-player domains via a modification of the strategy that would be optimal if all of a tree's nodes could be examined, the *minimax algorithm* [NM44]. In cases where information about all possible eventual outcomes of an action is available, leaf values, (usually win, loss, and draw), are minimaxed up the complete tree, and optimal decisions are based on this backed-up information. Most interesting games, however, generate trees that are too large to be searched exhaustively [2]. Thus, the tree is searched as deeply as possible, domain-specific static evaluators are applied to tip nodes at the search frontier, and the estimated values are minimaxed up the partial tree. Decisions are then made as if the backed-up information were exactly correct [Sha50]. This face-value principle of perfect estimates is generally taken for granted, not because it is believed to be true, but rather because no better assumptions present themselves [Pea84]. Given the reliance on accurate estimates, then, the importance of determining a precise, useful aim for the evaluator should be obvious. Nevertheless, no such definition has ever been developed. In general, two-player evaluators are described as either indications of a position's "worth" [Nil80] or estimates of the value that would be returned by a complete minimax search from that position all the way to the leaves [Pea84]. Neither of these definitions provides useful information. The major difficulty with the first lies in its vagueness, while the second offers no helpful instructions as to how an evaluator should be designed — the complete minimax value of a node is as hard to estimate as it is to calculate. For this reason, all standard-model evaluators have been based on game-specific features identified by experts.

The definitions given above pose several other difficulties, as well. First, the range of exact node values is limited to the range of possible leaf values.

[2] As an example of tree size, some complete game-trees have been estimated at 7^{60} nodes for Othello, 10^{40} for checkers, 10^{120} for chess, and $361! (> 10^{650})$ for GO.

All useful evaluation functions, however, return much larger ranges. The common resolution offered for this discrepancy that the larger ranges not only determine the eventual winner, but account for the ease of victory as well [Sha50]. Second, evaluation functions that attempt to estimate a node's complete-minimax value implicitly assume that play will be perfect beyond the search frontier despite a complete lack of knowledge about that portion of the tree. This difficulty has been almost universally overlooked in the past because the perfect-play assumption is useful as a defense mechanism. After all, defense against a perfect opponent should work against an imperfect one as well. Nevertheless, this approach does have its drawbacks — although it defends against perfect play, it may make defense against imperfect play considerably more difficult than necessary [Bra80].

More significantly, the use of partial-minimax to back up information has never been justified; it has been adopted because it would be optimal if the information were accurate. There is no reason to assume that it is proper when the values being backed up are estimates. One of the most obvious drawbacks to the strategy is its absolute reliance on the single best child of each parent. If that one crucial estimate is wrong, the entire strategy falls apart. It has been fairly well established that strategies that account for multiple children outperform minimax. Studies done on the M&N algorithm demonstrated that adding "some (experimentally determined) function of the M maximum or N minimum values" to the minimax value increases the accuracy of the decisions made [SD70]. In addition, the discovery of a phenomenon known as *minimax pathology* [Nau83] has led to the investigation of *product propagation* [Pea81], a strategy that backs up the product of the children's values. Despite the clearly inaccurate assumption of independence among sibling nodes required to justify this strategy, some surprising experiments have been run in which product propagation outperformed minimax [NPT83] [CN86]. This result alone should be enough to indicate that minimax is not always the strongest possible strategy.

Other difficulties with the standard model are somewhat subtler. The lack of a general guideline for evaluator design points to one such shortcoming: there is no known a priori method for determining the accuracy of a function. In a game whose leaf values are restricted to win, loss, and draw, (true for all games considered), a perfect evaluator returns the appropriate value of the three. Thus, the implicit task of an evaluator is classification, and a

function's strength is directly proportional to its ability to correctly identify wins, losses, and draws. The obvious way to determine strength, then, is to divide the range of returned values into three classes and calculate the percentage of actual values that were classified correctly. The problem with this approach is that it completely ignores the issue of decision quality, or the frequency with which the best move is made. Under this system, a function that incorrectly classifies a few nodes, albeit badly, receives a high rating, while one which makes many inconsequential misclassifications gets a low one. For example, consider two evaluation functions, A and B. Function A classifies 90% of the nodes correctly, but most of the erroneous 10% are losses which are given very high values (or wins given very low values). Any time one of these losses is available it will be selected despite the existence of many wins, which were correctly classified but given lower values. Function B, on the other hand, only classifies 60% of the nodes accurately, but most of its errors occur with values near the boundaries. Thus, B's errors rarely affect play, and when they do, the results tend to be far from disastrous, say the selection of an easy draw over a difficult win. If two such functions were pitted against each other, B would probably win more games. Thus, only a posteriori comparisons are possible. This, in turn, leads to a great many problems, most notably the inadequate testing of functions before they are actually used.

Although the above scenario may appear somewhat contrived, it is actually related to a well known game-tree phenomenon, the horizon effect, or the difficulty of knowing what lies just beyond the search frontier [Ber73]. Errors of the type made by function A frequently arise from the evaluation of board positions that occur in the middle of a combination of moves or a series of exchanges. These nodes are not *quiescent*, and they should not be evaluated statically. Evaluations that are made on non-quiescent nodes are highly unreliable because the static information is likely to change rapidly as soon as the horizon is extended [Sha50]. Unfortunately, non-quiescent nodes are not always easily recognizable, and are thus frequently unavoidable. The problem of quiescence can be viewed as a necessary outcome of the standard definition of the evaluator. Since the evaluator is designed to estimate the value of a node as it relates to the goal, there is no logical point for terminating search other than reaching a goal node [Bot84] [Ber79]. Thus, the search frontier must be set arbitrarily, and anomalies of the horizon abound.

In short, the absence of a precise definition of what an evaluation function is estimating sends a ripple of problems throughout the search system: backup algorithms can't be justified, functions can't be compared, and frontiers can't be set intelligently. Although each of these shortcomings has been discussed in the past, the effects that they have on the design and performance of game-playing programs is not always identifiable. Most previous work has concentrated on one (occasionally two) of these difficulties; never have all three been addressed simultaneously. I have presented a survey of this work in [Abr86]. The point of departure of my research from previous work is my interest in the general structure of game-trees. The next section outlines the expected-outcome model, an evaluator that considers the relative merit of nodes in a game-tree, rather than that of features on a board. Although the information contained in the two representations is equivalent, existing board-based evaluators are useful only in the games for which they were designed, while the new tree-based model should be applicable to any problem that can be represented as a game-tree, (or at the very least a large class of them).

3 Expected-Outcome: The Proposed Model

The purpose of an evaluation function in a two-player domain is to estimate whether a given node *on the search frontier* will result in a win. The proposed model contends that this corresponds to the probability that a randomly chosen leaf beneath the node in question will be a win. To determine this probability, consider the leaves' numeric values, (say one, zero, and one-half, for win, loss, and draw, respectively [3]), add the values of all leaves in the subtree beneath the given node, and divide by the number of leaves. The proper interpretation of this function is the expected value of the subtree, or an *expected-outcome* function.

At first glance, the assumption of random play may appear unreasonable.

[3]There are several equally reasonable methods for assigning numeric values to leaves. One other formulation that has been studied regards a draw as noise unless it is the only value possible. In this system, wins are one, losses zero, and draws don't count as leaves. For games with relatively few draws, the assignments are essentially equivalent. Only in games with a high density of draw leaves, such as chess, does the distinction become interesting. This is discussed in more detail later.

It is important to recall, however, that evaluation functions in two-player games are normally applied only at the frontier of the search. By definition, the frontier is the limit beyond which a program cannot search deeper in the game-tree, rendering information from the subtrees beneath it highly unreliable. The expected-outcome model effectively divides the tree into two sections. The lower part, which extends from the tips to the leaves, is too large to be understood completely. The sole available information is static, a description of what the subtree looks like (at least in terms of leaf composition). Only the upper part is well understood; various backup strategies can be applied to determine not only its configuration, but the dynamic flow of control through it as well. The absence of reliable control information from the tree's lower portion indicates that although play there will not actually be random, the assumption of random play is, in fact, credible, and certainly more realistic than the common assumption of perfect play.

Under this new model, the vaguely defined problem of devising evaluation functions for games is reduced to the precise problem of approximating the percentage of win leaves beneath a given node. Since this percentage corresponds to a statistical mean, it can be approximated by using the well-understood technique of random sampling. The remainder of this section will outline some of the open problems that can be addressed by the adoption of expected-outcome as the model for two-player static evaluators. They are not all issues that I expect to resolve as part of my dissertation, but they should help illustrate the model's widespread potential.

3.1 Benefits of the Model

In its purest form, calculating expected-outcome values involves knowing all leaf values in a tree. Since this is as impractical as performing complete minimax searches, some approximation technique must be adopted. Random sampling can fill this need in one of two ways: by replacing static evaluators or by designing evaluation functions. If a random sample is taken beneath every node on the search frontier, expected-outcome values can be assigned across the frontier and backed up the tree, and the need for static evaluation is completely eliminated. The idea of guiding search based on randomly sampled leaves has a certain aesthetic appeal. The standard approach performs

full-width searches as deeply as time permits. Sampling strategies augment this with full-depth searches to as wide a group of leaves as possible. Play can then be directed towards the subtree that has the most overall promise, rather than towards a single data point on the frontier (whose value may or may not be an anomaly of the horizon). One major advantage of this approach is that random sampling is a well understood statistical technique that estimates distribution means to within the limits of a known confidence interval, rendering the a priori comparison of two functions elementary. The drawback to sampling during play is, of course, the cost of preparing the sample. Leaves in a subtree can only be found by traversing entire paths. The computational effort expended on the full-depth portion of the search cuts down on the depth of the full-width portion. In many cases this sacrifice may be too costly to be acceptable.

The increased costs can be avoided if, rather than using the samples in the place of static evaluators, they are used to design new evaluators. If appropriate statically available domain features can be identified, random sampling can be combined with a variation-of-parameters learning technique [Sam63] [Sam67] [Gri74] [CK86] to determine coefficients that approximate expected-outcome values. Although a certain amount of expertise is needed to find the most appropriate features, all games have characteristics that can be distinguished even by a novice, (such as the different pieces in chess or different squares in Othello), and if necessary, modified using factor analysis or some other statistical method. Since these experiments are run only once and never during actual play, a great many points can be sampled at no extra cost to the player. This approach allows the full-width component to retain its maximum depth and the confidence interval to retain most of its significance as a function comparator. Its disadvantages, on the other hand, include the reliance on expertise to identify domain features and the use of an estimated estimate to guide search.

In addition to offering a means of comparing functions, the new model suggests an approach to developing reasonable, justifiable backup strategies. Rather than being viewed as point probabilities, the expected-outcome values on the search frontier actually represent the means of probability distributions. Unlike discrete point values, which are distinct, distributions tend to overlap. Thus, the face-value principle, which always recommends the node of best static value, (thereby justifying minimax), is no longer applicable.

Consider a simple illustration of the problem with minimax: some node in the tree has children valued at .8, .1, .72, .65, and .25. A standard max operation would return the value .8 and recommend moving in the direction of the corresponding child. If the parent is the current state, this clearly represents the best move. If, on the other hand, the move being considered actually lies somewhere in the future, (between the current move and the search frontier), the assumption that the .8 node will be chosen represents a premature commitment. The minimax model, in making this commitment, implies that *if* this position is reached, the .8 node *will* be chosen. A more accurate assessment of the situation is that *if* this position is reached, decisions that will be based on information found several levels down the tree *are most likely to* select the .8 node. The premature committment of minimax comes from the assumption that the estimates are entirely accurate. If $Prob[error = 0] = 1.0$ and the values being estimated are precise and deterministic, the distributions collapse to points, and the .8 is clearly optimal. In the absence of information about the accuracy of the evaluator, perfection may be the safest assumption. The notion of confidence intervals, on the other hand, allows the face-value principle to be dropped, and, coupled with the probabilistic interpretation of the evaluator, should lead to a family of new backup strategies.

The proper method for backing up distributions is not immediately obvious. Palay has analysed some distribution-based variants of the B* algorithm in which products of polynomial representations of the probability distribution functions (PDF's) are backed up [Pal85]. Although this method yields improved results over the originally proposed B* ranges [Ber79], PDF multiplication creates many problems: the independence and continuity of the distributions must be assumed, and a class of reasonable distributions that is closed under multiplication must be found. The success of the M&N algorithm [SD70], however, suggests a simpler approach. Recall that this algorithm added some experimentally determined function of the M or N best values to the minimax value. This immediately suggests backing up a weighted sum of a node's children, an operation with several advantages: it is easy to calculate, easy to justify, requires no independence assumptions, and keeps all distribution classes closed. Furthermore, knowledge of the mean and confidence interval and access to other distribution parameters should make the determination of the proper coefficients for each node a tractable

analytic task, rather than an experimental one. As part of my research, I plan to develop a method for calculating these coefficients. I am also optimistic that knowledge of the confidence interval will help overcome the major obstacle that has been faced by previous attempts to devise such backup algorithms: the lack of α-β pruning analogs. Although strategies more accurate than minimax have been devised, the increased accuracy has never sufficed to offset the lost efficiency. It may, in fact, turn out that minimax with α-β is optimal because of efficiency considerations. If so, this will constitute the first analytic justification for minimaxing estimated values.

The third area of difficulty, the determination of intelligent criteria for terminating search, can be addressed by considering a possible extension of the basic model, expected-reasonable-outcome. As it stands, the decision criterion of an expected-outcome function is rather simple: choose the subtree with the largest percentage of win leaves. The immediate objection to this strategy takes the form of an adversary argument, or the description of a case in which the percentage of wins is deceptive. Any instance of the horizon effect [Ber73], (originally presented to highlight the difficulty of quiescence), should also cause expected-outcome to select the wrong move. For example, a large percentage of the descendants sired by a chess node halfway through a queen trade will lead to victories. Upon completion of the trade, however, the queen will have been sacrificed to protect the king, and many of her stoutest offspring will have been summarily executed in their prime (i.e. most of the win leaves will be pruned immediately upon completion of the trade). In particular, expected-outcome will suffer in instances in which random play is blatantly inaccurate, or when there is an "obvious" next move available just beyond the frontier. It should be possible, however, to extend the model to sample non-random paths below the frontier. An iterated sampling procedure, using reasoning similar to that incorporated into the backup algorithms, could rely on the evaluation function derived on the previous iteration to determine the relative probabilities with which successor nodes will be chosen. Sampling paths with the appropriate probabilities would lead to an expected-reasonable-outcome function, which would indicate the expected value of the outcomes lying along reasonable lines of play, and thereby avoid the negative effects of an horizon.

By way of summary, then, the expected-outcome model allows all three major problems to be addressed. The most basic difficulty, the absence of

a definition of the evaluation function, is resolved by the basic statement of the model. The secondary problems, brought about by the lack of a definition, should be resolvable through the model's implications and extensions. Backup algorithms become justifiable when statistical confidence intervals are known, and an intelligent criterion for setting the search frontier can be developed through a simple iteration process. These features help make the new model desirable. The next section presents some experimental evidence that shows that it is reasonable, as well.

4 Supporting Evidence

The appeal of the expected-outcome model lies in its elegance, its crisp definition, and its domain-independence. In addition, it offers straightforward solutions to some long-standing open problems. The ultimate criterion by which an evaluator is judged, however, is performance in actual competition, not aesthetic appeal. Unfortunately, the general absence of both actual competitors and absolute standards make performance rather difficult to test thoroughly in a laboratory. Determining the utility of expected-outcome to game programming, then, must follow a somewhat different track. This section describes experiments that address some key questions: Do expected-outcome functions make good decisions? Are they useful in real games? Can they be automatically learned? Do they imply powerful backup strategies? and How do they relate to the ad hoc evaluators that have been studied in the past?

4.1 Completed Work

4.1.1 Decision Quality

The first step in determining whether a model is of practical use is investigating how often it recommends good decisions. If moving in the direction of the maximum win percentage generally leads to good moves, expected-outcome functions are powerful heuristics. Of course, "generally" and "good moves" are both subjective terms. In order to test the *decision quality* of a function,

(the frequency with which it makes correct decisions [4]), these terms must be defined. Perhaps the simplest definition of a "good move" is one which is as good as the optimal move, or one with the best complete-minimax value. (In other words, a minimax search to the leaves would either choose the move in question or one with the same value). Although other definitions are possible, the complete minimax procedure does offer a useful absolute standard for judging a move's quality — an incorrect move involves choosing a draw over a win or a loss over a draw. Unfortunately, no such standard exists for the term "generally". One datum that should be enlightening, however, is how the expected-outcome functions compare with previously studied expert-designed game-specific evaluators.

The first issue to be addressed, then, is how often the move with the largest (or smallest, as appropriate) percentage of win leaves beneath it is, in fact, optimal. In addition to testing the strength of the decision criterion, the comparison of expected-outcome with complete minimax will indicate how often the assumptions of random play and perfect play beyond the search frontier recommend different moves. Calculating either the complete-minimax value or the expected-outcome value, however, requires knowledge of the entire tree. 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 studied, yet large enough to offer more interesting testbeds than their smaller cousins. In the case of 4-by-4 tic-tac-toe, select patterns were frozen in place to generate eleven initial configurations. The patterns were chosen so that nearly all leaves would be considered at least once. As the complete tree is ascended, the density of examined nodes decreases. For 6-by-6 Othello, ten initial configurations were generated by moving randomly for the first twenty moves — only

[4]Technically speaking, decisions are made by control strategies, which are sets of rules for move selection that combine a static evaluation function with a dynamic lookahead procedure. If, however, the dynamic component of a strategy is ignored and the move with the best static value is chosen, the decisions may be attributed to the evaluators.

the last twelve were made with the help of evaluation functions. Although neither of these methods of generating initial configurations guarantees impartiality, the consistency of the results indicates that the techniques used probably did not have a major effect on the outcome. For each game, every node in the tree (beneath the initial configuration) was considered by four evaluation functions: minimax, expected-outcome, a popular standard, and worst-possible-choice, and a record was kept of each function's performance. Minimax, by definition, never made an error, and the worst-possible-choice function erred whenever possible.

In all four cases, the result was the same: expected-outcome made relatively few errors and outperformed the standard evaluators. Not surprisingly, the percentage of errors made by both evaluators increased as the tree was descended. Whereas most evaluators explain degraded performance by claiming that they were not tailored to end-game play, the new model offers a more precise explanation: statistical parameters like the mean of a distribution are only useful for large populations, and end-game nodes have relatively few descendants. These results are summarized in Tables 1 and 2 and Graphs 1 through 3. For each tree considered, the number of decision nodes, possible errors, and errors made by each function are shown. A node is considered a decision node if its best successor (as chosen by minimax) is not a leaf. The reason for this definition is that most evaluators recognize leaves as special cases. Thus, performance when leaves are available is not truly indicative of a function's accuracy. It is important to note that although the expected-outcome value, like the complete-minimax value, was calculated exactly by searching ahead to the bottom of the tree, expected-outcome did not back up any values; decisions were based strictly on evaluations of a node's successors. The standard evaluators were taken from published literature and calculated using only static information: the open-lines-advantage for tic-tac-toe [Nil80], and a weighted-squares function for Othello [Mag79]. Appendix 1 contains a detailed description of these functions.

The relative number of incorrect decisions in a search space is a fair basis for comparing two evaluation functions. The percentage of possible errors that they make, on the other hand, provides some insight into the absolute accuracy of the functions. With the exception of 4-by-4 Othello, expected-

outcome made only a small percentage of the possible errors [5]. The extremely small percentage of decisions in which errors were possible in that game, however, indicates that it may not be a fair gauge of decision quality — the worst-possible-choice selector was still better than 90% accurate. In terms of the standard evaluators, the results are consistent with observed performance. The open-lines-advantage function for tic-tac-toe is known to be fairly strong. When implemented with sufficient lookahead, it can both force a draw and take advantage of its opponent's errors. (In the 3-by-3 case, a one-ply lookahead is generally sufficient). Weighted-squares, with a mean accuracy of roughly 57% on trees with an average branching factor around seven is reasonable, but not overwhelmingly effective as an Othello evaluator. This is not surprising. A thorough analysis of the game showed that weighted-squares strategies were overly simplistic, unable to account for issues like mobility and stability, and not sensitive to the differences between opening, mid-game, and end-game strategies. An evaluator that took these items into consideration was able to play at world-championship level [Ros82]. Nevertheless, the study of weighted-squares does have scientific merit. The purpose of these experiments was not to develop the best performance-oriented Othello program, but rather to test the validity of a new model of evaluation functions. For this task, any well thought out, game-specific function offers a useful comparison, even if it is not the strongest known evaluator for the game in question. The basic result of these experiments is that in all cases tested, expected-outcome made fewer errors than the standard functions. 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: it generally leads to good moves.

4.1.2 Estimating Expected-Outcome Values

The results of the decision quality experiments are rather encouraging. They indicate that in instances where complete information is available, moving

[5] Small is, of course, relative. In some of the tests runs on 6-by-6 Othello, the error percentage was as high as one-quarter. Nevertheless, this is still small enough to indicate that for the overwhelming majority of nodes, the proper successor lay in the direction of maximum win percentage. Even in the case of 4-by-4 Othello, for that matter, the error percentage was less than one-half.

in the direction of maximum win percentage is frequently beneficial. Unfortunately, these are precisely the cases where complete-minimax searches are possible and optimal moves can always be made. Since probabilistic (and for that matter, heuristic) models are only interesting when predictions are based on incomplete information, some means of estimating expected-outcome values based on partial information is needed. The obvious technique for deriving these estimates is random sampling. The next issue that must be investigated, then, is whether estimated expected-outcome functions are of any use in real games. The second set of experiments is designed to address two questions: Do estimated expected-outcome functions make good decisions in interesting games? and Is random sampling useful as an expected-outcome estimator?

Expected-outcome values, by their very definition, represent the means of leaf-value distributions. Any sampler that wishes to estimate these values must make certain assumptions about the distributions. One assumption that is both reasonable and useful is that a path leading to a draw contributes no real information, and is thus nothing more than noise. This is useful because binary random variables are easier to deal with than ternary ones, and reasonable due to gaming intuition and experience — players rarely (if ever) play to draw while the possibility of victory remains. Thus, draw leaves found by a sampler rooted at a mid-game node are not attractive to either player. Draws become significant only when one of the other outcomes is unattainable, in which case they are valued at .5 (this rarely occurs, and then only in the end-game, where populations are small and expected-outcome functions of dubious accuracy, anyway). Otherwise, win leaves are valued at 1, loss leaves at 0, and draw leaves ignored. With draws disregarded as noise, the leaf-value distribution describes a binary-valued random variable, with $Pr[WIN] \stackrel{\text{def}}{=} p$ and $Pr[LOSS] \stackrel{\text{def}}{=} q = (1 - p)$. A random sampler that tallies leaves sampled (excluding draws) and wins found develops the estimate $\hat{p} = \frac{WINS}{LEAVES}$. By the law of large numbers, $\hat{p}$ should converge to p as the number of points sampled ($LEAVES$) increases.

Determining p through random sampling is not difficult. As is the case with most statistical estimates, perfection is not expected; two confidence parameters, α and ϵ, define the probability $(1 - \alpha)$ with which $|p - \hat{p}| \leq \epsilon$. Using the normal approximation to the binomial distribution, the number of

sample points needed for this degree of certainty is $S = \frac{z^2_{\alpha/2}pq}{\epsilon^2}$, where $z_{\alpha/2}$ is the z-score of $\alpha/2$, or the number of standard deviations from the mean such that the area under the distribution curve from $\mu - (z_{\alpha/2})\sigma$ to $\mu + (z_{\alpha/2})\sigma$ is $(1-\alpha)$. For example, to achieve 95% certainty that $|p-\hat{p}| \leq .05$, set $\alpha = .05$, $\epsilon = .05$, $z_{\alpha/2} = 1.96$, and $S \approx 1537pq$. Since the term pq reaches a maximum of .25 at $p = q = .5$, $S \leq 385$. If the confidence is relaxed somewhat and only 90% certainty is required, the formula indicates that $S \leq 271$. Sampling this many points is expensive and frequently unneccessary. In general, pq will be considerably smaller than .25 and good estimates for p will be found fairly quickly. The sample sizes given by the formula are actually worst case scenarios — the only reasonable assumption if the number of samples is set in advance. If convergence is checked for intermittently during the course of sampling, on the other hand, the sampler may stop when a reasonable estimate has been found. One approach to on-the-fly convergence detection is to sample N leaves, count the wins, and set $\hat{p}_0 = \frac{WINS}{N}$. Keeping a running tally of wins, sample another N points, and set $\hat{p}_1 = \frac{WINS}{2N}$. Continue this procedure, with $\hat{p}_i = \frac{WINS}{2^iN}$, until $|\hat{p}_i - \hat{p}_{i-1}| \leq \epsilon$. If this is true, the first 2^{i-1} leaves and the second 2^{i-1} leaves sampled contain nearly identical proportions of wins. Thus, it is reasonable to assume that the overall win ratio, or p, is fairly close to the estimate given by $\hat{p}_i$.

This technique was used in the second set of experiments, which pitted an estimated expected-outcome function vs. a weighted-squares function (see Figure 2a) in four 100-game matches of (8-by-8) Othello. The expected-outcome function's sampler set $\epsilon = .05$ and $N = 8$, but stopped checking for convergence if none had been found by the time 256 samples were taken. It is important to note that these estimates of p are far fom perfect — if they were completely accurate, the moves would be deterministic, (as they are for weighted-squares or any other standard evaluator), and all games would be identical. Arbitrary tie-breaking rules account for minor variations in play, but even so, the number of different games should be fairly small. The proof of imperfection, then, is that in the first match, all 100 games were distinct. An investigation of the remaining two matches is unnecessary; it is clear that the values used are estimates, not exact expected-outcome values. Thus, tighter estimation procedures should lead to stronger expected-outcome functions, just as more careful game-specific analyses led to stronger

standard functions. This analogy helps justify the adoption of weighted-squares as the benchmark against which the initial sampler-based function is judged: weighted-squares were the first reasonable expert-designed Othello functions, and stronger evaluators became possible in large part due to the feedback provided by their performance [Ros82].

These experiments, like those run on decision quality, were designed as pure tests of evaluator strength — neither player used any lookahead. Unlike decision quality, however, these matches have no absolute standard to be judged against. This immediately decreases the precision possible in interpreting their outcome. Fortunately, the relation of the experiments' results to their intent should not cause much controversy. To be of any use, sampler-based functions must compete favorably with those designed by experts. In each of the matches, the competition was about even; in no case did either evaluator win enough games to make a viable claim of superiority. From the sampler's viewpoint, the win-loss-draw scores of the four matches were 46-48-6, 41-53-6, 48-49-3, and 54-41-5, for an overall total of 189-191-20. It is important to keep these results in their proper perspective. As a demonstration that estimated expected-outcome yields the world's best Othello evaluator, the experiments are woefully inadequate — the absence of lookahead makes the games unrealistic, the difference in computation times [6] 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 on par with those designed by experts, and the data clearly substantiates the claim.

Expected-outcome functions, then, do appear to be useful in real settings. Given no expert information, the ability to evaluate only leaves, and a good deal of computation time, they were able to perform on par with a function that had been hand-crafted by an expert. Thus, both questions have been answered in the affirmative: expected-outcome functions can be estimated by a sampler, and the estimates do lead to good moves.

[6] For most of the cases tested, the sampler needed between one and ten minutes per move. By contrast, the weighted-squares function rarely took more than two seconds to statically evaluate all possibilities and select a move.

4.2 Work in Progress

4.2.1 Learning Expected-Outcome Functions

Given that estimated expected-outcome functions make reasonable moves in real games, attention can now be focused on the next question: Is the model of practical use to real game programs? The major drawback to sampling strategies is the cost of preparing the sample. The time discrepancy between the sampler and the weighted-squares player of the estimation experiments is clearly unacceptable. If lookahead were introduced, the time required to sample would rapidly mushroom beyond reasonable limits. In section 3, it was pointed out that the time spent sampling detracts from time spent extending the frontier. This added cost can be avoided if a static evaluator that recognizes a node's expected-outcome value can be devised. If this is possible, the costs can all be attributed to a learning preprocessor and have no effect on the actual game. Perhaps the simplest way of learning such a function for Othello is to start with significant board features that have already been identified by an expert. The equivalence classes of squares defined by the weighted-squares function should be able to serve this purpose. A parameter learning technique, such as learning by regression analysis [CK86], can be used to develop coefficients for the features.

The third set of experiments will start by using this technique to learn an evaluation function. This learned function, which is a static estimator of the expected-outcome value, will play a series of games against the original weighted-squares function. In these matches, lookahead length will be varied to see what effect, if any, this has on the relative strength of the functions. If the learned function plays on par with the original, this furthers the claim that the expected-outcome model is, in fact, a reasonable one. The possible supremacy of the original function, on the other hand, should not be taken as an immediate condemnation of the new model. There are several factors that could contribute to uneven play, and they must all be investigated before the model may be rejected. First, it is possible that the significance of some squares change frequently and radically throughout the game. This problem can be alleviated by using a slightly more complex function, one that uses different coefficients for different stages of the game. Second, the equivalence classes of squares may be completely irrelevant to the expected-outcome value. The way around this problem is to use a statistical procedure

that learns features as well as coefficients, such as factor analysis. In fact, a strong case could be made for starting with this type of experiment and never relying on features identified by an expert at all. I have chosen to save it for a fallback, however, so that the model itself can be studied, and not be confused with the statistical techniques used in its implementation. Third, there may be something to the backup procedure, minimax, that favors the standard function. The simplest way to test this, of course, is to play a match with no lookahead. If this turns out to be the case, it should offer some insight into the relationship between minimax and various evaluation functions, and suggest some methods for tailoring new functions to the backup strategy.

These experiments have not been started, and there are, of course, a myriad of unforseeable circumstances that could arise. I believe that the three problems mentioned above are the most likely, and that the resolutions given for them should lead to experiments that convey useful information about the model's strengths, as well as a few allusions to its relationship with previously proposed evaluators.

4.2.2 New Backup Strategies

The major purpose of the learning experiments is to demonstrate that expected-outcome information can be efficiently incorporated into real games. One of the potential difficulties suggested in the context of these experiments was that standard weighted-squares functions may somehow be able to exploit the quirks of minimax better than their expected-outcome counterparts. Although this is rather unlikely, it should be possible to design a backup strategy that avails itself of the information provided by expected-outcome values. In section 3, the potential for developing a new family of backup strategies, based on the definition of evaluation functions as probability distributions, was mentioned. Using this interpretation, it becomes evident that the "quality" of nodes frequently overlap. Under the minimax model, given a choice between two nodes of values v_1 and v_2, $v_1 > v_2$, the v_1 node is always recommended to the maximizer and the v_2 node to the minimizer. If the evaluator is reasonable, these moves will be correct more often than not. If, however, the values are viewed as the means of two random variables, X_1 and X_2, respectively, a mixed strategy is likely to be stronger than a pure one. For example, if there are N choices to be made between nodes of values v_1 and

v_2, the maximizer could expect to note improved performance by selecting the v_1 node aN times and the v_2 node bN times, where $a + b = 1$ and $1.0 \geq a \geq b \geq 0.0$. The parent of these nodes, then, is a random variable defined by $X_{parent} = aX_1 + bX_2$, with a mean evaluated at $v_{parent} = av_1 + bv_2$. The appropriate values for a and b depend on the distributions of X_1 and X_2.

Once again, the relegation of draws to noise status is useful. This assumption, first introduced in the estimation experiments, reduces all X_i to binary valued random variables, with $p_i \stackrel{\text{def}}{=} Pr[\text{node i will lead to a win leaf}] = v_i$ [7]. Since p_i is known, so is $q_i = (1 - p_i)$. This is enough information to specify all moments of the distribution, and should ease the development of an analytic technique for determining appropriate weights. In addition to everything else, this approach is intuitively appealing. In binomial distributions, means are dependant on p, while variances depend on the product pq. Thus, random variables for which $p \approx 0.0$ or $p \approx 1.0$ will have very small variances, and the largest possible variance occurs at $p = q = .5$. Nodes with large and small values of p, then, will stand out as clear choices; the overlap with distributions defined by other nodes will be minimal. Intermediate-valued nodes, on the other hand, will have large variances and overlap greatly. This is as it should be: oustanding moves should be given values near 0 and 1, moves of dubious quality should not.

The new backup strategy, once derived, suggests two interesting areas of study: experimental testing and accuracy vs. efficiency analysis. Testing can be accomplished by taking the learning experiments one step further. With lookahead depth fixed, a program using the new strategy and the learned function can be matched against a program minimaxing either evaluator. The outcome of these matches should indicate the relative utility of the backup strategies. Even if the new technique makes more accurate decisions than minimax given equivalent information, its use may not be preferable in actual game situations. One of the most salient features of minimax is its companion algorithm, α-β-pruning. When implemented together, these algorithms can deepen search greatly, and effectively increase the amount of useful information available to the decision maker. It is possible that the

[7] The meaning of this probability actually changes as the tree is ascended. On tip nodes, it corresponds to the probability of a random path leading to a win. Otherwise, the path is weighted (in the example above by a and b) until the tips, and random from then on.

confidence in the estimates of p, (the tolerable error), combined with the variance and higher moments, may define an α-β analog for the new backup strategy. In any case, the tradeoffs of accuracy for efficiency should offer some insight into the propriety (or lack thereof) of perpetuating minimax usage.

4.2.3 Expected-Outcome in Chess

Backup strategies aside, if the learning experiments work as anticipated, the expected-outcome model will have been shown to be reasonable for at least one class of games, those with relatively few draws. Not all games belong to this class. In chess, for example, most games end in a draw. This could cause serious problems for the random sampling procedure. It is conceivable that these draws introduce so much noise into the system that the number of samples needed to differentiate between moves that are likely to lead to wins and those likely to lead to losses is prohibitive. The fourth set of experiments involves the learning of coefficients for the chess pieces to develop a static estimator of the expected-outcome value. This is completely analogous to the experiments described above for Othello, and faces the same set of problems. Feedback from the Othello experiments may be helpful in deciding exactly how to implement the tests on chess.

Regardless of the outcome of this experiment, the result will be interesting. If the learned coefficients approximate the generally accepted values, a strong case can be made for the claim that previous evaluators have unwittingly been estimating the percentage of wins beneath a node. If the learned coefficients are significantly different from the accepted ones, but compete favorably with them in a series of games, it may represent a contribution to our understanding of chess, as well as our understanding of heuristics. Finally, if the learned coefficients perform poorly, it will highlight some of the differences between chess and Othello. Even in this case, however, expected-outcome should not be dismissed out of hand for chess. The model outlined here is the simplest possible application of the underlying ideas. Many involved extensions and refinements are possible. The expected-reasonable-outcome model mentioned in section 3, for example, may be necessary for a game as complex as chess. In any event, the learning of chess coefficients that approximate expected-outcome values should provide some insight into the power

and applicability of the new model.

5 Contributions

The focal point of my dissertation is the introduction and development of a new probabilistic model of evaluation functions for two-player games, the expected-outcome model. The thesis underlying this model is that knowledge of the percentage of wins beneath a given node in a game tree is general, powerful, and leads to good moves. The work described in this proposal develops the basic model and takes it through a series of experiments that will determine whether it is a useful technique for designing game programs. Experiments that have already been concluded are strong enough to show that expected-outcome functions make good moves. Furthermore, I am rather optimistic that the model will also prove to be useful in the design of efficiently calculable static evaluators. In its strictest sense, this alone should constitute a significant contribution to the fields of game programming and heuristic analysis, as the first general-purpose, domain-independent technique for the design of two-player evaluation functions. In a broader sense, however, the contributions of the model and its possible extensions are profound.

The scheme for devising backup strategies could either justify the use of minimax or propose a superior alternative. Although this idea has yet to be developed fully, it is significant as the first illustration of the potential benefits of a well defined aim for static evaluators. Further benefits abound, but rely, for the most part, on extensions of the basic model that are beyond the scope of the current work. Perhaps the simplest variant removes the assumption of random play beyond the search frontier, which, although justifiable, is incorrect. A more powerful evaluator, (mentioned in section 3), assumes that reasonable moves will always be made, and sums the values of all moves deemed reasonable. Evaluators of this nature define an entire spectrum of reasonable-play assumptions bridging the gap between randomness and perfection. This approach retains much of the defensive power of the perfect-play assumption without falling prey to its weaknesses, and involves only a simple iteration of the learning procedure. At least in principle, the learned function should be strengthened with every iteration, to the point where it may even asymptotically approach the value returned by a complete

minimax search.

Other extensions of the model are interesting, as well. As things stand, the only statistical parameter being used is the mean of the leaf distribution. One variant would use additional statistical information to make even stronger decisions. In particular, knowledge of a distribution's variance and skew should be helpful in detecting quiescence and predicting what lies beyond the search frontier. A second variant would consider games in which paths to the leaves are too long to be searched entirely. Instead, random deep searches could be applied beyond the frontier and expert-designed functions used to estimate the value of nodes at that depth. Although this procedure loses much of the elegance of the original model and resurrects many of the problems inherent in standard approaches, it may yield improved performance without encountering cumbersome calculations.

Unless they are accompanied by other advances in the field, however, even functions derived through some of these extended models may not be sufficient to achieve two long-standing goals in game design, grandmaster level in chess and competence in Go. Nevertheless, the adoption of a useful definition for two-player evaluation functions should serve as an important point of departure for future game designers, lead to the development of probabilistic analyses of the general structure of games and game-trees, and perhaps even suggest an approach towards the unification of the theories of one- and two-player games.

Acknowledgements

A lot of work has gone into developing the ideas outlined in this proposal, and many people have pointed me in useful directions. Virtually every issue outlined has been discussed (or debated) with my advisor, Richard Korf. Other people who have been particularly helpful include Peter Allen, Michael Foster, Jonathan Gross, John Kender, Michael Lebowitz, Andrew Mayer, Judea Pearl, and Igor Roizen.

This document is not a discussion of completed work. It was prepared as a written form of a proposal for my doctoral dissertation. This proposal was presented at Columbia University on August 18, 1986 in an open colloquium to a faculty committee consisting of Richard Korf, Jonathan Gross, and Michael Foster, and approximately twenty attendees. This research has been

sponsored in part by the National Science Foundation under grant IST-85-15302.

References

[Abr86] Bruce Abramson. *Control Strategies for Two-Player Games.* Technical Report, Columbia University, May 1986.

[Ber73] Hans J. Berliner. Some necessary conditions for a master chess program. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pages 77–85, 1973.

[Ber79] Hans Berliner. The b* tree search algorithm: a best-first proof procedure. *Artificial Intelligence*, 21:23–40, 1979.

[Bot84] M.M. Botvinnik. *Computers in Chess: Solving Inexact Search Problems.* Springer-Verlag, 1984. trans. A. Brown.

[Bra80] M.A. Bramer. Correct and optimal strategies in game playing programs. *The Computer Journal*, 23:347–352, 1980.

[CK86] 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.

[CN86] Ping-Chung Chi and Dana S. Nau. Predicting the performance of minimax and product in game-tree searching. In *Proceedings of the 2nd Workshop of Uncertainty in Artificial Intelligence*, pages 49–55, 1986.

[Gri74] A.K. Griffith. A comparison and evaluation of three machine learning procedures as applied to the game of checkers. *Artificial Intelligence*, 5:137–148, 1974.

[Mag79] Peter B. Maggs. Programming strategies in the game of reversi. *BYTE*, 4:66–79, 1979.

[Nau83] Dana S. Nau. Decision quality as a function of search depth on game trees. *JACM*, 30:687–708, 1983.

[Nil80] Nils J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Company, 1980.

[NM44] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[NPT83] Dana S. Nau, Paul Purdom, and Chun-Hung Tzeng. *Experiments on Alternatives to Minimax.* Technical Report, University of Maryland, October 1983.

[Pal85] Andrew J. Palay. *Searching With Probabilities.* Pitman, 1985.

[Pea81] Judea Pearl. Heuristic search theory: a survey of recent results. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 24–28, 1981.

[Pea84] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison Wesley, 1984.

[Poh70] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1:193–204, 1970.

[Ros82] Paul S. Rosenbloom. A world-championship-level othello program. *Artificial Intelligence*, 19:279–320, 1982.

[Sam63] 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.

[Sam67] A.L. Samuel. Some studies in machine learning using the game of checkers ii — recent progress. *IBM J. Res. Dev.*, 11:601–617, 1967.

[SD70] James R. Slagle and John K. Dixon. Experiments with the m & n tree-searching procedure. *CACM*, 13:147–154, 1970.

[Sha50] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.

[Ste83] Stephen L. Stepoway. Reversi: an experiment in game-playing programs. In M.A. Bramer, editor, *Computer Game Playing: Theory and Practice*, Ellis Horwood Limited, 1983.

Appendix 1: Standard Evaluation Functions

The evaluation function used for tic-tac-toe is rather simple. Each line (row, column, and diagonal) on the board represents a potential win. A line is open to a player if his opponent has no marks in it. The open-lines-advantage evaluation function for position p, $E_{ttt}(p)$, is

$$E_{ttt}(p) = \begin{cases} \infty & \text{if } p \text{ is a win for X} \\ -\infty & \text{if } p \text{ is a win for O} \\ OPEN_p(X) - OPEN_p(O) & \text{otherwise} \end{cases}$$

where $OPEN_p(Player)$ = the number of lines open to Player in p.

This function was used in [Nil80] to demonstrate issues related to minimax search and static evaluation. It has an intuitive appeal, and works rather well when combined with lookahead. On 3-by-3 tic-tac-toe, a one ply lookahead is sufficient to force a draw, which is, in fact, the outcome of the game when both players are perfect. Examples of this function on 3-by-3 and 4-by-4 boards can be found in Figure 1.

The function used for Othello is a bit more complex. The basis of a weighted-squares strategy is the realization that not all squares on the board are of equal value; edge squares are less likely to be flipped than squares in the center, and corner squares will never be flipped at all. Squares immediately adjacent to the corner, on the other hand, are frequently detrimental to the player moving there first, because they allow the other player to take the corner. Once the corner has been taken, however, they lose that special status. Weighted-squares strategies have been discussed in the literature and implemented in many programs [Mag79] [Ste83]. The function used for 8-by-8 Othello is a weighted-squares strategy, based on the one presented in [Mag79], and modified to account for the author's personal experience with the game. Although it is not immediately clear how the 8-by-8 function should be applied to smaller boards, the convention chosen was to consider squares based on their locations with respect to the corners. The values for each square are shown in Figure 2. Note that the large numbers were retained because they were not derived scientifically in the first place. Thus, although the 64 in the corners clearly came from the number of squares on an 8-by-8 board, there is no reason to believe that it is not a reasonable value

for a corner on a smaller board as well. Leaves were recognized when neither player was able to move, (this occurs trivially when the board is full), and classified as wins, losses, or draws as dictated by the final score.

Appendix 2: Summary of Experiments

This is a brief list of the experiments discussed in section 4. It is included as a quick reference guide to the goals of my research and how I plan to meet them.

1. **Decision Quality:** Do expected-outcome functions make good decisions?
   Four small games were tested. In all cases, expected-outcome usually chose the optimal move. By way of comparison, some well known evaluators erred more frequently.
   Conclusion: Yes, at least for the games tested.

2. **Estimation:** Do estimated expected-outcome functions make good moves in actual games?
   A random sample of leaves beneath each node was taken to estimate the expected-outcome value (with draws regarded as noise). The sampler stopped when it either converged to a reasonable value or had examined 256 leaves. This function was pitted against a standard weighted-squares function in 100-game matches of Othello, with no lookahead used by either player.
   Conclusion: In at least one implementation, they play as well as a reasonable, expert-designed function. Thus, it is fair to assume that the decisions made by the two functions are of roughly the same quality.

3. **Learning:** Is it possible to devise a static evaluation function that approximates the expected-outcome value?
   A variation-of-parameters technique, (specifically, learning through regression analysis), will be applied to a large set of Othello configurations to learn coefficients for a set of board features. The features have been identified by the designer of the weighted-squares function studied in the previous experiments, and the coefficients will be learned to approximate the expected-outcome value. The learned function will be pitted against the original function in a series of games played with varying lookahead.
   Status: Not yet begun.
   Projected Conclusions: The coefficients learned through a purely

mechanized process are as good as or better than those determined by an expert.
Possible Problems: First, reliance on an estimated estimate may be insufficient to exploit the power of expected-outcome. Second, the features identified may either be inadequately correlated to the expected outcome, or be unstable throughout the course of the game.
Resolutions: If the features are irrelevant to what is being learned, more complicated statistical procedures, such as factor analysis, may help find better board features. If the problem is instability, learning different coefficients for different phases of the game should solve the problem.

4. **Backup Strategies:** Devise a strategy that backs up weighted sums of random variables, rather than minimum and maximum point values.
Comment: Unlike the other areas outlined, this is essentially an analytic task, not an experimental one. A variety of weighted-sum techniques are possible, and it is diffcult to project a priori which one will turn out to be correct.
Status: Under consideration.
Demonstrative Tests: Continue the experiments run on the learned evaluation functions, playing a minimax strategy vs. the weighted-sum strategy.
Analytic Issues: Explore the possibility of developing an α-β analog, and consider tradeoffs between accuracy and efficiency.

5. **Chess:** Can the learning experiments be applied to a more complex game, such as chess?
A similar regression analysis learning technique will be used to learn coefficients for the different chess pieces.
Status: Not yet begun.
Possible Outcomes:

   - The values learned approximate the generally accepted ones: this outcome strengthens the claim that two-player evaluators should, in fact, be estimating the expected-outcome value.

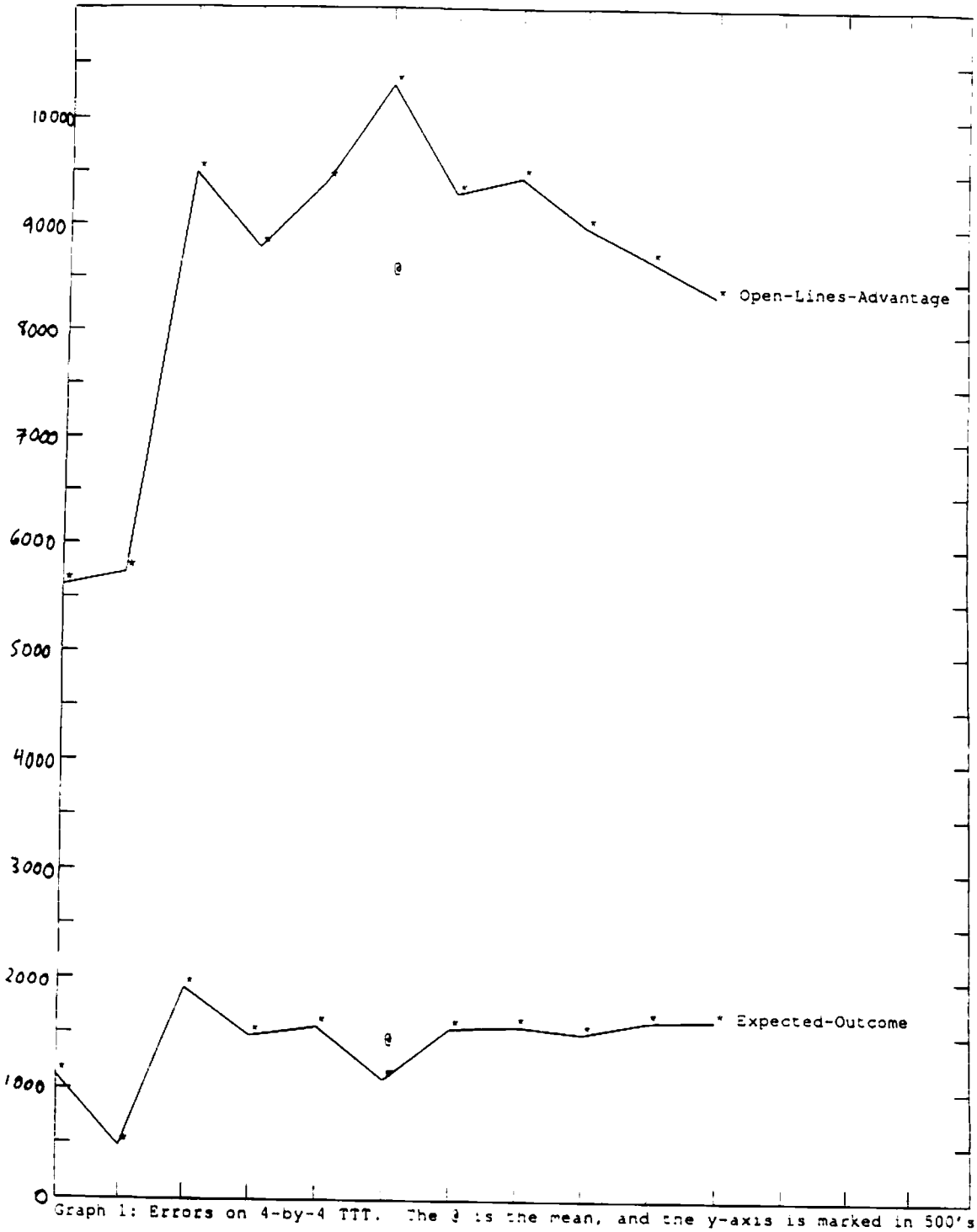
- The learned values differ from the standards ones, but compete favorably with them: this outcome also strengthens the claim that expected-outcome is a reasonable model, and may offer some insight into the design of better chess evaluators.
- The learned values differ from the standards, and do not compete favorably with them: this indicates a fundamental difference between the games of chess and Othello, and points to the boundaries of usefulness of the expected-outcome model in its pure form. It may also suggest which extension of the model is useful to games like chess.

Decision Quality					
Game	Decisions	Errors Made			Initial Configuration
		Standard	Expected-Outcome	All Possible	
3x3 Tic-tac-toe	2474	534	236	1180	Empty Board
4x4 Tic-tac-toe	82293	5612	1112	25030	XOOX in 5-6-9-10
	81551	5734	476	26105	XXOO in 5-6-9-10
	89764	9488	1908	32226	XOOX in 8-9-10-11
	88544	8791	1483	27619	XOXO in 8-9-10-11
	88687	9415	1566	30655	XXOO in 8-9-10-11
	87301	10302	1090	28920	XOOX in 12-13-14-15
	88654	9301	1549	30544	XOXO in 12-13-14-15
	88687	9447	1572	30775	XXOO in 12-13-14-15
	82509	8988	1508	30264	XOOX in 0-5-10-15
	82363	8680	1620	28439	XOXO in 0-5-10-15
	82152	8350	1632	27032	XXOO in 0-5-10-15
4x4 Othello	69308	3396	3112	6632	4 center squares filled
6x6 Othello	193877	6024	2994	11537	20 random moves made
	362356	8002	2394	16113	20 random moves made
	415792	20705	10194	44054	20 random moves made
	336338	4845	928	11176	20 random moves made
	907095	44815	17948	104835	20 random moves made
	482945	14586	4380	33277	20 random moves made
	813139	72121	24825	146675	20 random moves made
	523215	31027	8762	63791	20 random moves made
	104260	10077	3241	19082	20 random moves made
	899214	58215	16418	139201	20 random moves made

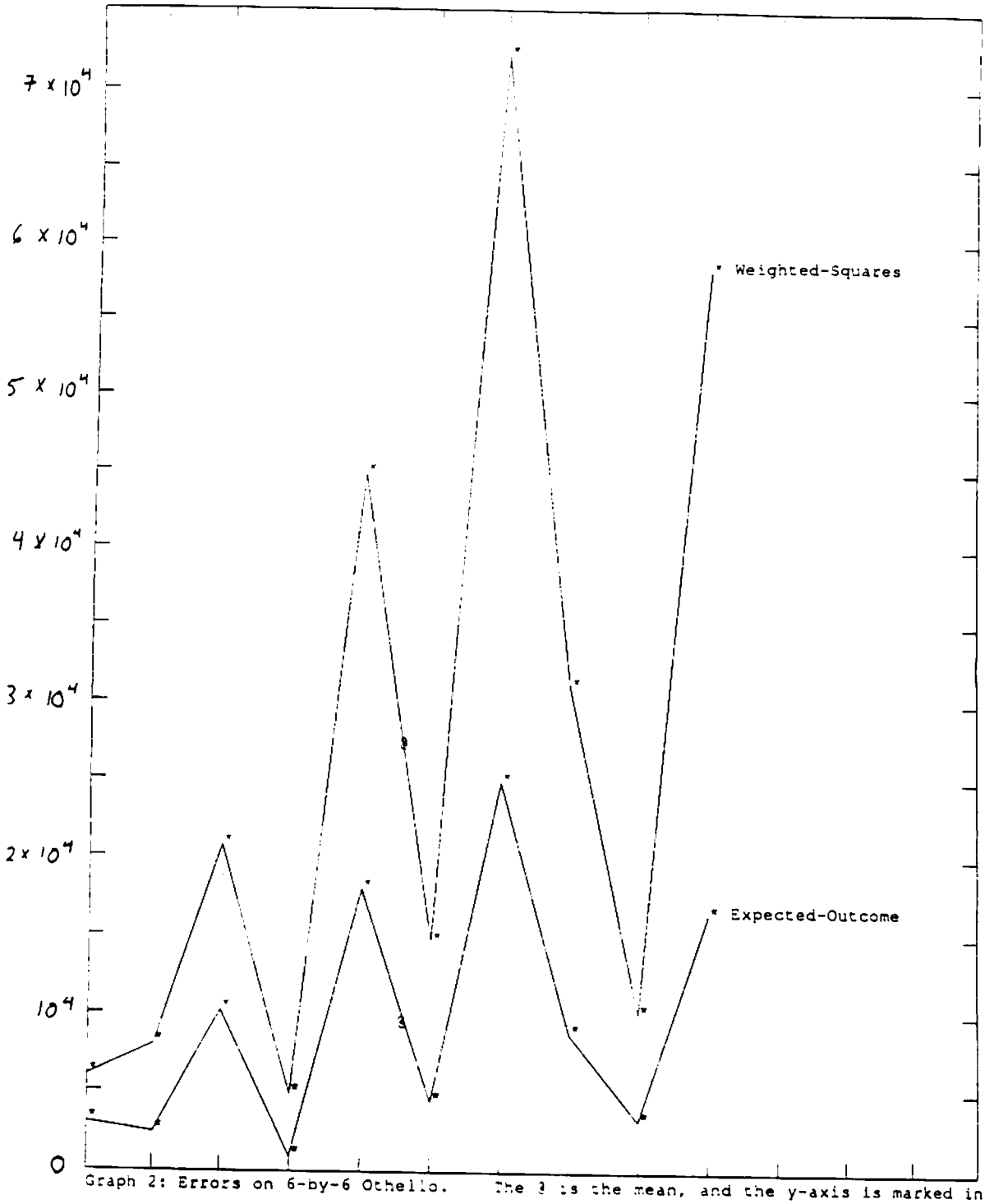
Table 1: This table contains the output of the decision quality experiments. The decisions column records the number of nodes whose best successor, as selected by minimax, was not a leaf. The number of possible errors was determined by an evaluation function that always chose the successor with the worst minimax value; if it did not err, no error was possible. For the other two functions, a choice was considered an error if the minimax value of the selected successor was not the correct minimax value. In all Othello games, the game starts with the four center squares filled with two black discs occupying one diagonal, and two white discs the other. In addition, for the tests run on 6-by-6, the initial configurations were generated by making the first twenty moves randomly. For 4-by-4 tic-tac-toe, the initial configurations were generated by freezing the specified patterns in the squares indicated (the numbering scheme can be found in Figure 1). Graph 1 displays the errors made by expected-outcome and open-lines-advantage on 4-by-4 tic-tac-toe, and Graph 2 those made by expected-outcome and weighted-squares on 6-by-6 Othello. The data points on these graphs are arranged in the order of the trials, as indicated above. The shape of the curves is not particularly significant; the major point of interest should be the errors made by each of the evaluators on identical subtrees.

4-by-4 Tic-tac-toe (11 trials)			
	Maximum	Minimum	Mean
Decision Nodes	89764	81551	85682.27
Possible Errors	32226	25030	28873.54
Errors by Open-Lines-Advantage	10302	5612	8555.27
Errors made by Expected-Outcome	1908	476	1410.55
Possible Errors/Decision Nodes	0.366796	0.304157	0.336799
Errors by Open-Lines/Decision Nodes	0.118006	0.068195	0.099550
Errors by Expected-Outcome/Decision Nodes	0.021256	0.005837	0.016410
Errors by Open-Lines/Possible Errors	0.356224	0.219651	0.294773
Errors by Expected-Outcome/Possible Errors	0.060373	0.018234	0.048482
6-by-6 Othello (10 trials)			
	Maximum	Minimum	Mean
Decision Nodes	907095	104260	503823.09
Possible Errors	146675	11176	58974.10
Errors made by Weighted-Squares	72121	4845	27041.70
Errors made by Expected-Outcome	24825	928	9208.40
Possible Errors/Decision Nodes	0.183023	0.033228	0.106776
Errors by Weighted-Squares/Decision Nodes	0.096653	0.014405	0.050635
Errors by Expected-Outcome/Decision Nodes	0.031086	0.002759	0.017480
Errors by Weighted-Squares/Possible Errors	0.528089	0.418208	0.471246
Errors by Expected-Outcome/Possible Errors	0.259513	0.083035	0.161974

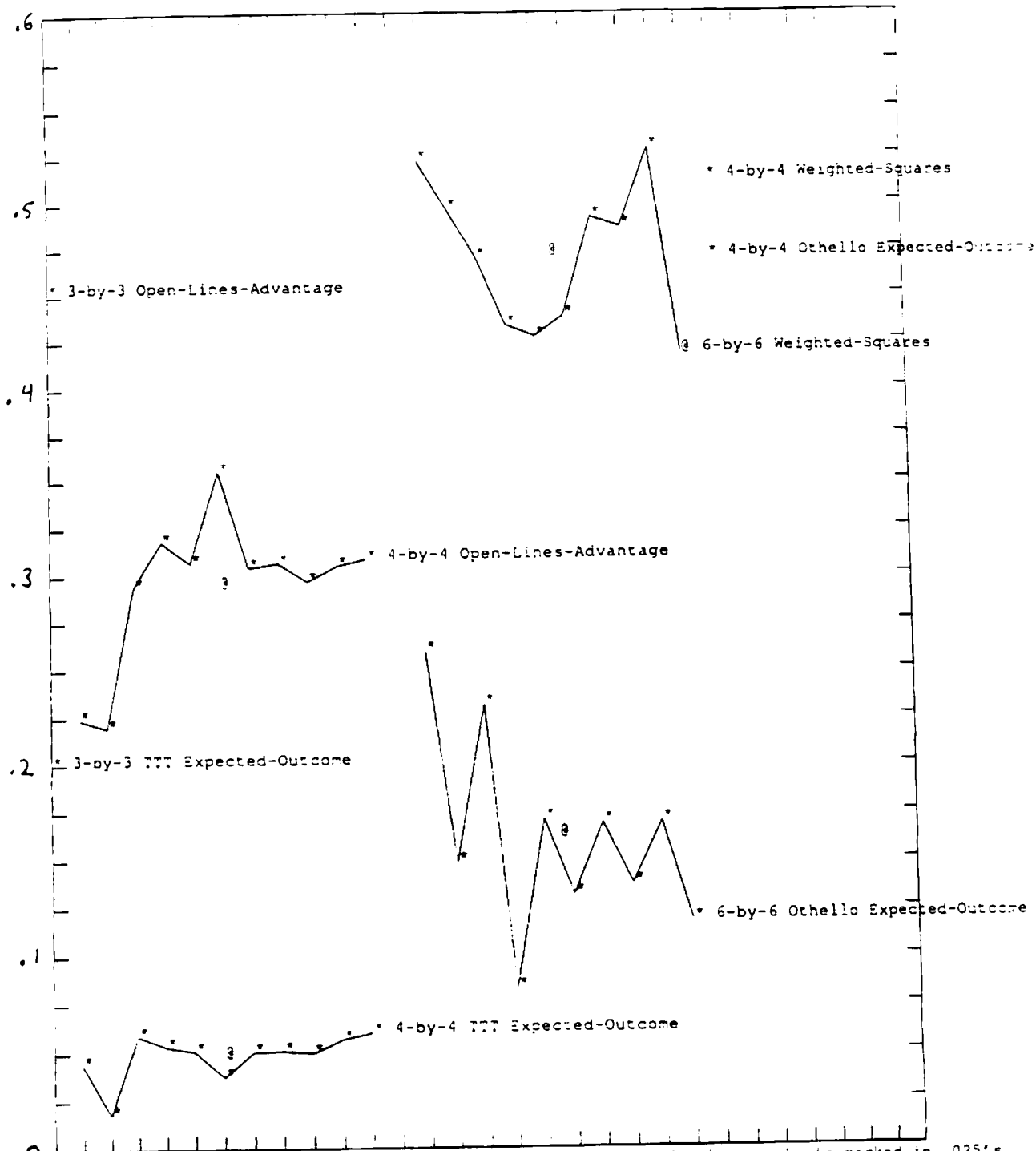
Table 2: This table contains some statistical information about the raw data presented in Table 1. For the two games with more than one trial run, the maximum, minimum, and mean of several points of interest are shown. The number of tests run is insufficient for any additional statistical parameters (e.g. standard deviation) to be significant. The percentage of decisions in which errors were possible is indicated by the number of possible errors divided by the number of decision nodes. The percentages of erroneous decisions and possible errors made by each of the functions are shown above and displayed graphically in Graph 3. Once again, the shapes of the curves are essentially meaningless; the important feature is the relative error-percentages made by two evaluators on the same tree.



Graph 1: Errors on 4-by-4 TTT. The @ is the mean, and the y-axis is marked in 500's.



Graph 2: Errors on 6-by-6 Othello. The 3 is the mean, and the y-axis is marked in 5000's.



Graph 3: Percent of possible errors. The @'s are means, adn the y-axis is marked in .025's

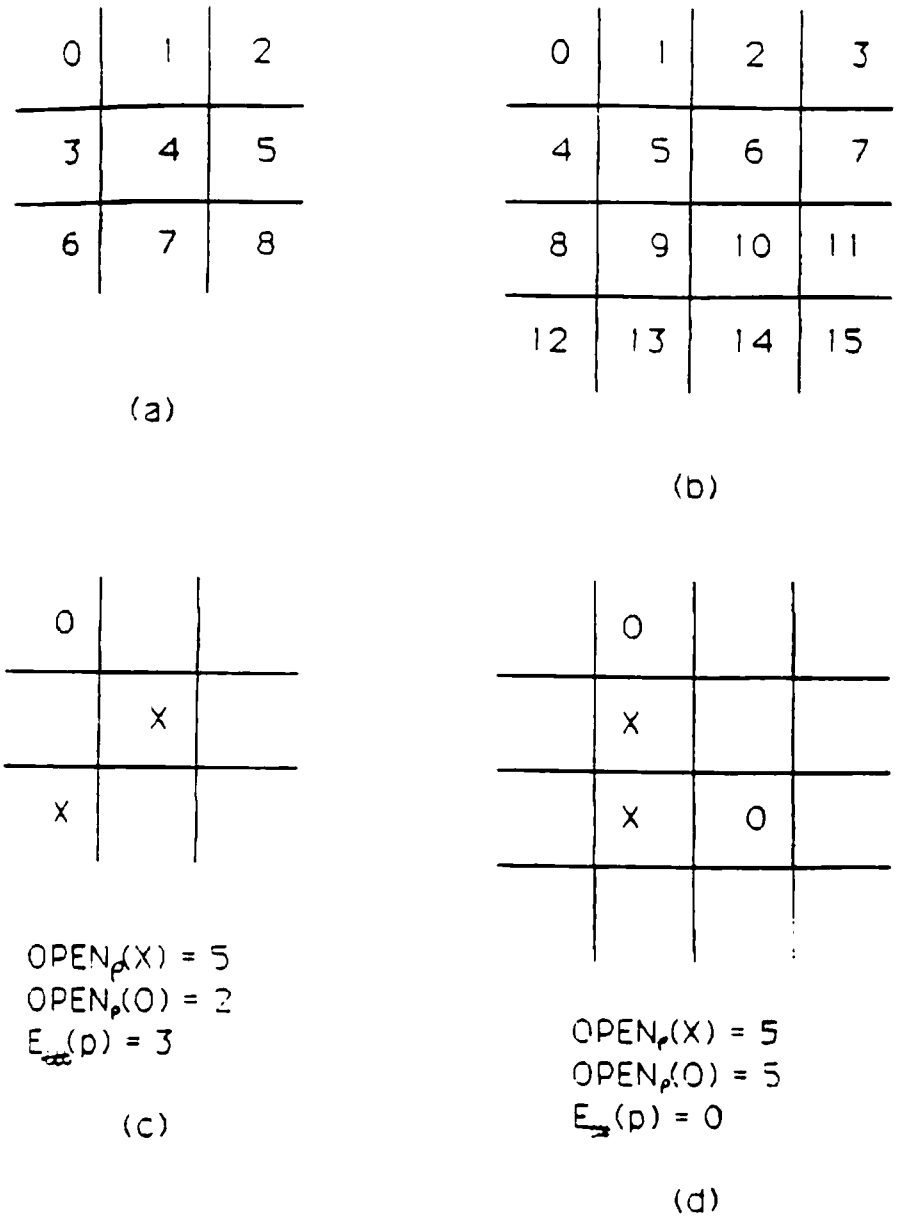


Figure 1 - Figures (a) and (b) show the numbering of the squares for 3-by-3 and 4-by-4 tic-tac-toe, respectively. For the tests run on 3-by-3, the initial configuration was the empty board. The initial configurations for the 4-by-4 tests were as specified in Table 1. Figures (c) and (d) illustrate the use of the open-lines-advantage function.

64	5,-30	10	5	5	10	5,-30	64
5,-30	-2 -40	-3	-3	-3	-3	-2 -40	5,-30
10	-3	2	1	1	2	-3	10
5	-3	1	1	1	1	-3	5
5	-3	1	1	1	1	-3	5
10	-3	2	1	1	2	-3	10
-5,30	-2 -40	-3	-3	-3	-3	-2 -40	5,-30
64	5,-30	10	5	5	10	5,-30	64

(a)

64	5,-30	10	10	5,-30	64
5,-30	-2 -40	-3	-3	-2 -40	5,-30
10	-3	2	2	-3	10
10	-3	2	2	-3	10
5,-30	-2 -40	-3	-3	-2 -40	5,-30
64	5,-30	10	10	5,-30	64

(b)

64	5,-30	5,-30	64
5,-30	-2 -40	-2 -40	5,-30
5,-30	-2 -40	-2 -40	5,-30
64	5,-30	5,-30	64

(c)

Figure 2 - A weighted-squares function for (a)8-by-8, (b)6-by-6, and (c)4-by-4 Othello. Squares with two values take on the smaller one when the adjacent corner is empty.