Expected-Outcome: A General Model of Static Evaluation

BRUCE ABRAMSON, MEMBER, IEEE

Abstract—**Static evaluation is one of the least understood aspects of two-player games. The *expected-outcome* model proposes that the proper evaluation of a game-tree node is the expected value of the game's outcome given random play from that node on. Although this characterization may appear somewhat naive, experiments performed on variants of tic-tac-toe, Othello, and chess, substantiate its viability as a general paradigm of evaluator design. First, expected-outcome is considered in its ideal form, where it is shown to be a powerful heuristic. Next, the ability of a simple random sampler that estimates expected-outcome to outduel a standard Othello evaluator is demonstrated. Finally, the sampler is combined with a linear regression procedure to produce efficient expected-outcome estimators. Overall, the expected-outcome model of two-player games is shown to be precise, accurate, easily estimable, efficiently calculable, and domain-independent.**

Index Terms—**Artificial intelligence; decision making; experimentation, simulation, and analysis; game-trees, two-player games, and evaluation functions; heuristic search; machine learning; probability and statistics.**

I. Introduction

THE mathematical study of games predates digital computers by several decades [22]. With the coming of the computer age, the marriage of games as decision making models, to computers as decision making machines, was swift and natural. In 1950, with computer science in its infancy and the term "artificial intelligence" as yet unborn, Claude Shannon's *Programming a Computer for Playing Chess* [20] begat the computer game. The original objective behind game programming was to understand how decisions are made. Four decades of research on computer chess have led to special purpose chess architectures that play at and beyond the master level [4], [8], but left many fundamental issues in decision-making unresolved.

One key issue faced by all decision makers is the quantification and comparison of potential domain configurations; functions that assign values to states in a game are known as static evaluators. In the past, evaluators have been designed ad hoc, in terms of the specific game being studied. The closest thing to a standard model of static evaluators defines an ideal function as one that returns a node's complete minimax value (the value that would be attained by performing a minimax search to the leaves), and an heuristic function as one that estimates it [20]. Even this characterization, however, is generally viewed as an implicit property of the function rather than as an explicit design objective.

At first glance, this estimated-minimax model appears promising. A closer look, however, reveals several problems. In addition to lacking the necessary precision to extend beyond two-player games, estimated minimax values offer few suggestions about how to design, judge, compare, and learn appropriate functions. Design is impossible because minimax values are as difficult to estimate as they are to calculate. In addition, the determination of evaluator accuracy must be done comparatively—there is no known standard against which it can be judged. Defining an evaluator's strength as proportional to the strength of its play, however, fails to provide the desired absolute measure of heuristic function quality. Furthermore, the outcome of head-to-head competition may be quite misleading; a program's decision quality and performance level depend on its entire control strategy, not just its evaluation function (see [1] for my recent survey of control strategies). Finally, a model that fails to indicate how to design, judge, or compare functions is unlikely to offer valid techniques for learning them. These shortcomings indicate that despite its empirically demonstrated strength as a control strategy (partial or estimated), minimax offers little insight into the design of static evaluators.

This paper proposes an unambiguous, domain-independent model of static evaluation, namely the projected outcome of a game given random play; this *expected-outcome* model considers the relative merit of game-tree nodes rather than board positions. By making this distinction, the model immediately increases a function's extensibility from a specific game to the class of two-player zero-sum games of perfect information. In Section II, a node's expected-outcome is defined as the expected value of the leaves beneath it. At first glance, this expected-outcome model appears quite naive—the implicit underlying assumption of random play is clearly inaccurate. In addition, the obvious method of estimating these values, namely random sampling of leaves, involves traversing paths all the way from the node being evaluated to the bottom of the tree—not a particularly efficient technique. Since accuracy and efficiency are the two major concerns in system design, these objections are potentially damn-

Manuscript received January 19, 1988; revised August 14, 1989.
This work was supported in part by the National Science Foundation under Grants IST 85-15302 and IRI-8910173, and a grant from the USC Faculty Research Initiation Fund.
The author is with the Department of Computer Science, University of Southern California, Los Angles, CA 90089.
IEEE Log Number 8932051.