A Unified Theory of Heuristic Evaluation Functions and its Application to Learning

Jens Christensen
Computer Science Department, Stanford University,
Stanford, Ca. 94305

Richard E. Korf
Computer Science Department, University of California,
Los Angeles, Ca. 90024

Abstract

We present a characterization of heuristic evaluation functions which unifies their treatment in single-agent problems and two-person games. The central result is that a useful heuristic function is one which determines the outcome of a search and is invariant along a solution path. This local characterization of heuristics can be used to predict the effectiveness of given heuristics and to automatically learn useful heuristic functions for problems. In one experiment, a set of relative weights for the different chess pieces was automatically learned.

1. Introduction

Consider the following anomaly. The Manhattan distance heuristic for the Fifteen Puzzle is computed by measuring the distance along the two-dimensional grid of each tile from its current position to its goal position, and summing these values for each tile. Manhattan distance is a very effective heuristic function for solving the Fifteen Puzzle [4]. A completely analogous heuristic can be defined in three dimensions for Rubik's Cube: for each individual movable piece of the cube, count the number of twists required to bring it to its goal position and orientation, and sum these values for each component. Three dimensional Manhattan distance, however, is effectively worthless as a heuristic function for Rubik's Cube [5]. Even though Rubik's Cube is similar to the Fifteen Puzzle, the two heuristics are virtually identical, and in both cases the goal is achieved when the value of the heuristic is minimized, the heuristic is very effective in one case and useless in the other.

As another anomalous example, consider the games of checkers and Othello with material count as an evaluation function. Othello is a game played on an eight by eight square grid with pieces which are white on one side and black on the other. Each player alternately places pieces with his color showing on empty squares. Whenever a player places his pieces at both ends of a line of his opponent's pieces, the opponent's pieces are flipped over and become the property of the original player. The winner is the player whose color shows on the majority of the pieces at the end of the game. Material count is an evaluation function which sums the number of pieces belonging to one player and subtracts the total material of the other player. It turns out that material count is a fairly successful evaluation function for checkers but relatively ineffective for Othello, even though the winner is the player that maximizes his material in both cases.*

*Strictly speaking, a checkers game is won when the opponent has no more legal moves, but this almost always occurs as a result of the opponent losing all his pieces.

A challenge for any theory of heuristic evaluation functions is to explain these anomalies. An additional challenge is to present a consistent interpretation of heuristic functions in single-agent problems and two-player games. Surprisingly, the treatment in the literature of heuristic search in these two different domains has little in common. In single-agent searches, a heuristic evaluation function is viewed as an estimate of the cost of the remainder of the solution path. In two-person games, however, a heuristic function is vaguely characterized as a measure of the "strength" of a board position for one player versus the other.

2. A Unified Theory of Heuristic Evaluation Functions

One criterion which distinguishes the successful heuristics from the unsuccessful ones above is that in the successful cases, primitive moves in the problem space make only small changes in the value of the heuristic function. In the case of Manhattan distance for the Fifteen Puzzle, a single move changes the Manhattan distance by a single unit whereas for Rubik's Cube a single twist can change the Manhattan distance by as much as eight units (eight pieces move at once). Similarly, the material count in checkers rarely changes by more than a single piece during one move, but in Othello it can change by a large number of pieces (up to 18 in one case). This suggests a theory that evaluation functions which are relatively invariant over single moves are more effective.

A closely related idea was suggested by Lenat in the more general context of heuristic production rules [6]. A production rule has a left-hand side that specifies a situation where it is applicable, and a right-hand side that determines the action to be taken in that situation. Lenat argues that the power of heuristic production rules is derived from the fact that the appropriateness of a situation-action pair is a continuous function of both the situation and the action. In other words, if a particular action is appropriate in a particular situation, then 1) a similar action is likely to be appropriate in the same situation, and 2) the same action is likely to be appropriate in a similar situation. If we broaden the definition of action to include evaluation, and allow the situation variable to range over different states in the same problem space, then our notion of relative invariance over single moves becomes a special case of Lenat's continuity idea.

Of course, invariance over single moves is not enough to assure a useful evaluation function, since this can be trivially achieved by assigning all states the same constant value. The heuristic values must be tied to actual payoffs in the game, in particular to the values of the goal states. This suggests that when a heuristic function is applied to a goal state, it should return the exact value of that state.

Informally, we claim that an ideal heuristic evaluation function has two properties: 1) when applied to a goal state, it returns the outcome of the search; and 2) the value of the function is invariant along an optimal solution path. Taken together, these two properties ensure a function which is a perfect predictor of the outcome of pursuing any given path in the problem space. Therefore, a heuristic search algorithm using such a function should always make optimal moves. Furthermore, we claim that any successful evaluation function will satisfy these properties to some extent.

For example, the evaluation function for the A* algorithm [3] is $f(n)=g(n)+h(n)$ where $g(n)$ is the cost of the best path from the initial state to the node n and $h(n)$ is an estimate of the cost of the best path from node n to a goal state. Typically the h term is called the heuristic in this function, but for our purposes we will refer to the entire function f as the heuristic evaluation function. When this function is applied to a goal node, the h term is zero, the g term represents the cost of reaching the goal from the initial state, and hence f returns the cost of the path or the outcome of the search. If h is a perfect estimator, then as we move along an optimal path to a goal state, each move increases g by the cost of the move and decreases h by the same value. Thus, the value of f remains invariant along an optimal path. If h is not a perfect estimator, f will vary somewhat depending upon the amount of error in h. Thus, a good evaluation function for an algorithm such as A* will determine the outcome of the search and is relatively invariant over single moves.

Now consider a two-person game using minimax search and a static evaluation function. The static evaluation reflects the strength of a given board position. When applied to a state where the game is over, the function determines the outcome of the game, or which player won. This is often added as a special case to an evaluation function, typically returning positive and negative infinity for winning positions for MAX and MIN, respectively. When applied to a non-goal node, the function is supposed to return a value which predicts what the ultimate outcome of the game will be. To the extent that the evaluation is an accurate predictor, its value should not change as the anticipated moves are made. Thus, a good evaluation function should be invariant over the actual sequence of moves made in the game. Therefore, in both examples we see that a good evaluation function should have the properties of 1) determining outcome and 2) invariance over single moves.

2.1. Formal Description of the Theory

In this section we will define the properties of outcome determination and move invariance and show that these conditions are sufficient for perfect play by a heuristic search algorithm.

A heuristic function is said to *determine the outcome* of a search if when applied to any terminal or goal state, it returns the figure of merit for the task. This is the criterion against which success is measured. For example, in a single-person search where the task is to find a lowest cost path to a goal state, the outcome would be the actual cost of the solution path found. In a two-person game, the outcome might be either win, lose, or draw for a particular player, or a number indicating a score.

An *optimal move* from a given state is one which leads to a best outcome in the worst case. For example, in a single-person problem an optimal move is a move along a lowest cost path from the given state to a goal state. For a two-person game, an optimal move is determined by expanding the entire game tree to terminal values, minimaxing the terminal values back up the tree, and picking a move which forces a win if one exists, or forces a draw if no wins exist. If all moves result in a forced loss, all moves are optimal. Note that the optimal move is the best move given the current state of the problem or game. It is defined for all states, not just those on a globally optimal path from the initial state.

An algorithm exhibits *perfect play* if for all states, it makes an optimal move.

A heuristic function is said to be *move invariant* if the value it returns for any given state is equal to the value returned for the immediate successor which results from an optimal move.

A *heuristic search algorithm* is one which makes its decisions about what move to make next solely based on the minimum and/or maximum values of the heuristic evaluation function of the successors of the current state. Note that such an algorithm may or may not include lookahead. Lookahead is included by allowing it as part of the heuristic evaluation of a state. This definition encompasses all the standard heuristic search algorithms for one- and two-player games.

Our main theoretical result is the following: *Outcome determination plus move invariance are sufficient conditions for a heuristic evaluation function to guarantee perfect play by a heuristic search algorithm.* Its proof is as follows:

Move invariance requires that the heuristic value of any state and its successor resulting from an optimal move be the same. Since an optimal solution path is just a sequence of optimal moves, move invariance implies that the heuristic evaluations of all states along an optimal solution path from any given state are the same. Outcome determination ensures that the heuristic value of the goal at the end of such a path equals its exact value. Therefore, both properties together guarantee that the heuristic value of any given state is a perfect predictor of the eventual outcome of that state given perfect play. Thus, a heuristic search algorithm need only generate all successors of the current state, evaluate them, and choose the minimum or maximum value as appropriate to ensure optimal moves from every state.

While outcome determination and move invariance are sufficient conditions for perfect play, strictly speaking they are not necessary conditions. The reason is that a heuristic function with these properties could be composed with any function which preserves the maximum or minimum of a set without changing the moves that would be made. If we ignore such order-preserving functions, however, outcome determination and move invariance become necessary for perfect play.

Since outcome determination plus move invariance is equivalent to perfect prediction, one way of interpreting the above result is that perfect heuristics are necessary and sufficient for perfect play. On the surface, this seems somewhat contrary to well-known results such as the optimality of A* with inexact, but admissible, heuristics. Note, however, that A* doesn't commit itself to making any moves until it has searched the optimal solution path all the way to the goal, and hence it knows the exact outcome of the best move before it chooses that move.

2.2. Predicting Heuristic Performance

The decomposition of perfect prediction into the independent conditions of outcome determination and move invariance is useful for predicting heuristic performance qualitatively. For example, Manhattan Distance satisfies outcome determination in both the Fifteen Puzzle and Rubik's Cube, as does material count in both checkers and Othello. Both heuristics, however, differ markedly in move invariance in their two respective problems. Thus, our theory successfully distinguishes the useful from the useless heuristic in both cases. Furthermore, it provides a single, uniform interpretation of heuristic evaluation functions over both single-person and two-player games.

3. Learning Evaluation Functions

In addition to unifying the theory of heuristic functions, and making qualitative predictions about the performance of given evaluation functions for given problems, our theory can be used as the basis of a method for learning heuristic functions. The main contribution of the theory to this problem is that it decomposes the global property of perfect prediction into the two local properties of outcome determination and move invariance. Thus, we can search for heuristics that satisfy one of these properties, and then test to what extent the other is satisfied as well.

The basic idea is that since part of the characterization of a successful evaluation function is in terms of invariance over single moves, candidate evaluation functions can be optimized based on local information in a problem space. In particular, one can search for a function which is invariant over moves along a solution path. This technique was implicitly used by Samuel's [10] pioneering experiments on learning checkers evaluation functions, and by Rendell's [9] more recent work on learning heuristics for the Fifteen Puzzle. Below we describe some experiments which replace Samuel's ad hoc techniques with the well-understood method of linear regression, and extend the method to the domain of chess.

3.1. Description of the Method

We adopt the standard game-playing model of mini-max search with static evaluation at the search frontier [11]. While other learning experiments have focused on openings or endgames [2, 7, 8], we have addressed the mid-game. Samuel [10] observed that the most effective way of improving mid-game performance is to modify the evaluation function.

The first game program to improve its performance by learning was Samuel's checkers program [10]. Although it also employed other learning techniques, it is mostly known for learning the coefficients in its polynomial evaluation function. Samuel's idea was that the difference between the static evaluation of a board position and the backed-up value derived from a mini-max search could be used to modify the evaluation function. This is based on the assumption that for a given evaluation function, values based on looking ahead are more accurate than purely static evaluations. Samuel's program altered the coefficients of the function at every move where there was a significant difference between the value calculated by the evaluation function and that returned by the mini-max search. The idea is to alter the evaluation function so that it can calculate the backed-up value at the original state without having to look ahead. An ideal evaluation function eliminates the need for a mini-max search since it always calculates the correct value for any state.

The main difference between our approach and Samuel's is in how the value returned by the mini-max search is used to modify the evaluation function. Samuel employed an ad-hoc technique based on correlation coefficients and somewhat arbitrary correction factors. Our method is based on the well-understood technique of linear regression. In addition, while his investigation focused on checkers, our experiments have been carried out in the more complex game of chess.

3.2. Coefficient Modification by Regression

For pedagogical reasons, we will explain the technique using the simple example of a checkers evaluation function based only on the numbers of single pieces and kings. In other words, we want to determine the relative value of the kings and pieces in an evaluation function of the form $C_1F_1 + C_2F_2$ where F_1 and F_2 are the numbers of pieces and kings, respectively. Of course, there would also be terms for the opponent's material, but we assume that the coefficients have the same magnitude and opposite signs.

We start with an initial estimate of the coefficients, e.g. both equal to one. Given a particular board position, we can plug in values for F_1 and F_2. Then, we perform a look-ahead search to some depth, evaluate the nodes at the frontier using the initial estimate of the coefficients, and back-up these values using the mini-max algorithm, resulting in a numerical value for the original position. This information can be represented as an equation of the form $C_1F_1 + C_2F_2 = R$, where the C_i are the parameters of the equation, the F_i are the factors of the evaluation function or dependent variables, and the R is the backed-up mini-max value. One can then perform a linear regression on this data to determine the best-fitting values for the parameters of the equation, thus in effect establishing the coefficients of the factors in the evaluation function.

Unfortunately, the result of the regression is not the best choice of coefficients but rather a better estimate. The reason is that the right-hand sides of the equations are not exact but approximate values since they are based on the same estimated coefficients. Thus, the entire process must be repeated using the new coefficients derived from the regression. These iterations are continued until the values converge.

This iterative algorithm can be viewed as hill-climbing in the space of coefficients, with potentially all the normally associated problems of hill-climbing. In particular, there may exist values which are locally stable but not globally optimal. No effective way exists to detect such local stabilities except by drastically altering some of the coefficients in the regression analysis to see if different maxima are encountered. If that is the case, then these different evaluation functions can be played against each other to see which one is indeed the best.

This learning method can be applied to any game which can be implemented using mini-max search with static evaluation. Note that the learning is accomplished simply by the program playing games against itself, without any outside input.

Our method was first explored in the simple game of 4x4x4 tic-tac-toe, and performed remarkably well. We used six factors in the evaluation function, namely the number of rows, columns, and diagonals in which a side could win with either one, two, or three pieces of the same color already in place. Not only did it order the factors of the evaluation function in increasing order of the number of pieces in place, but it also quickly recognized which coefficients had incorrect signs and reversed them.

3.3. Experiments with Chess

As a serious test, we chose the game of chess and a simple evaluation function consisting only of material advantage. The experiment was to see if the learning program would approximate the classically accepted weights for the pieces: 9 for the queen, 5 for the rook, 3 for the bishop, 3 for the knight, and 1 for the pawn. The chess program was implemented using a two-ply (one full move) mini-max search with alpha-beta pruning and quiescence. 1400 half-moves were made between each regression. If neither side won during a game it was stopped after 100 half-moves and a new game was started. For purposes of the experiment, a win was assigned one more than the total initial material value, and the individual piece values were rounded off to the nearest 0.5. The pieces stabilized at: Queen, 8.0; rook, 4.0; bishop, 4.0; knight, 3.0; pawn, 2.0.

The above results were based on a search of only two ply, plus quiescence. This means that the chess program was playing a tactical game, trying to maximize material in the short run rather than to achieve checkmate. Since the equations correspond to moves from every phase of the game, the final values are average weights from the opening, midgame, and endgame. Berliner has observed, however, that the optimal evaluation function is in general a function of the stage of the game [1]. Because of the weakness in the end game caused by the lack of planning the chess program could not take advantage of the rook's increased strength during the end game. Other pieces might suffer from similar effects.

When we played the derived function against the classical function in one hundred games, the derived function won seventeen games and lost sixteen. The rest were draws. This does not mean that our derived function is optimal, only that it is as good as the classical one in the context in which it was learned, namely two ply search using only a material evaluation function.

4. Conclusions

We have presented a theory which unifies the treatment of heuristic evaluation functions in single-person problems and two-person games. The theory characterizes a useful heuristic function as one which determines the outcome of a search when applied to a terminal position, and is invariant over optimal moves. We have shown that these two properties are sufficient for perfect play by a heuristic search algorithm. This local characterization is useful for making qualitative predictions about the performance of given heuristics, and for the automatic learning of heuristic functions. In one experiment, our program was able to automatically learn a set of relative weights for the different chess pieces that are as good as the classical values in the context in which they were learned.

5. Acknowledgments

This research has benefitted from discussions with Bruce Abramson and Judea Pearl. This research was supported by the National Science Foundation under grant IST-85-15302, and by an IBM Faculty Development Award.

6. References

[1] Berliner, Hans.
On the construction of evaluation functions for large domains.
In *Proceedings of IJCAI-79*, pages 53-55. International Joint Conferences on Artificial Intelligence, Tokyo, Japan, August, 1979.

[2] Berliner, Hans, and Murray Campbell.
Using chunking to solve chess pawn endgames.
Artificial Intelligence 23(1):97-120, 1984.

[3] Hart, P.E., N.J. Nilsson, and B. Raphael.
A formal basis for the heuristic determination of minimum cost paths.
IEEE Transactions on Systems Science and Cybernetics 4(2):100-107, 1968.

[4] Korf, R.E.
Depth-first iterative-deepening: An optimal admissible tree search.
Artificial Intelligence 27:97-109, 1985.

[5] Korf, R.E.
Macro-operators: A weak method for learning.
Artificial Intelligence 26:35-77, 1985.

[6] Lenat, Douglas B.
The Nature of Heuristics.
Artificial Intelligence 19:189-249, 1982.

[7] Minton, Steven.
Constraint-based generalization, Learning game-playing plans from single examples.
In *AAAI-84*, pages 251-254. American Association for Artificial Intelligence, Austin, Texas, August, 1984.

[8] Quinlan, J. Ross.
Learning efficient classification procedures and their application to chess end games.
In Michalski, R.S., J.G. Carbonell, and T.M. Mitchell (editors), *Machine Learning*, pages 463-482. Tioga, Palo Alto, Ca., 1983.

[9] Rendell, L.
A new basis for state-space learning systems and a successful implementation.
Artificial Intelligence 20:369-392, 1983.

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

[11] Shannon, Claude E.
Programming a computer for playing chess.
Philosophical Magazine (Series 7) 41:256-275, 1950.