Deep Blue: Research Review [1]

John Carpenter, PEng.

I. Introduction

The 1997 chess match Deep Blue v. Kasparov was arguably one of the most influential demonstrations of artificial intelligence ever shown. At 18, the chessmaster Kasparov routinely defeated the grandmasters before him with his wildly agressive and creative manner. His tactics and skill made him one of the best players to ever play chess, and he jumped at the opportunity to prove even the best computer couldn't defeat a grandmaster at chess.

In 1997 computers were little more than small calculators. They were functional at raw calculations but were never expected to be able to defeat a human opponent. Computers possesed no algorithm for creativity, deception or trickery. They simply ran algorithms. Kasparov would dictate the play and force the computer into situations which required elegance, not brute force to resolve.

However Deep Blue managed something no computer had done before and soundly defeated Kasparov in a series of matches widely followed by millions around the world. In its win Deep Blue was described as being more creative, and inventive compared to Kasparovs rigid approach. Which is ironic since, Deep Blue was for all purposes, simply a very smart algorithm.

II. Building a House with Toothpicks

Deep Blue operated at a blistering 11 GFLOPs [2] processing millions of moves per second, spread amoung hundreds of processors. It used specially designed chips and hardware designed solely to work through the problem of calculating the best move in chess. That would seem like an insurmoutable speed at that time but it should be noted the laptop this report has been created with can routinely calculate between 40-60 GFLOPs. What was far more interesting was the clever algorthms to determine the board state and evaluate positions

The Deep Blue architecture utilized a brute force quiescence search that implemented a number of very exact optimizations with respect to chess. Given the combinatorial complexity of chess, Deep Blue was rarely able to calculate out more than 6-8 moves ahead. This put a lot of dependence upon efficient calculations within the tree search and more importantly a precise board evaluation for each node.

For the search, Deep Blue utilized a Negamax[3] with alpha-beta pruning algorithm. The Negamax algorithm is similar to the minimax algorithm, however instead of changing between the min and max of a set of nodes, Negamax assumes that a good player moves and a bad opponent moves are equally valued

$$max(a, b) = -min(-a, -b)$$

. This reduces the calculations to a single form and reduces the calculations within the tree.

Secondly, it utilizes a *null window* search to speed the pruning in the alpha-beta phase. The null window sets a fixed value for alpha and beta to narrow the search when the game results are nearly certain. They mention that additional searches may be required if they are looking for exact values, but the speed performance for the null windows was sizable.

Finally, they add a number of chess constraints that are particular to the game. These contraints limit searches where previous experience has shown that the position is not viable. Effectively pruning the tree before spending the time in calculating the value of its children.

III. Teaching an Algorithm to Understand Chess

At the core of the Deep Blue algorithm is a very complicated evaluation model. The model attempts to "score" the current game position using a book of roughly 8000 chess positions and their advantages within the game. Those positions ranged from a single piece location to a full board configuration and the summation of those positions in the current board is the relative score of that position. The determination of these scores was a tedious effort involving teams of grandmasters and developers. Having a table of results in which to lookup the scores sped the search up significantly. The complexity of the search became effectively a linear problem.

Today this process of training with known datasets, and applying relative weights has become a well-known process in modern machine-learning algorithms. Deep Blues implementation shows the practical aspects of machine-learning can be applied to brute-force search algorithms quite effectively. Using a dictionary of succesful positions and weights as a heuristic is a effective and fast way to evaluate almost any position.

IV. Conclusions

Deep Blue implemented a novel approach to the brute-force game agents methods by merging and optimizing learned results into it's model. This combination of both raw calculations and learned behaviour produced a masterful chess machine that was able to defeat on the strongest chess minds of our time. As a game agent, its search and evaluation model work surprisingly well and even exhibits "creative" behaviour.

References

[1] Feng-hsiung Hsu, Murray S. Campbell, and A. Joseph Hoane, Jr.. 1995. *Deep Blue system overview*. In Proceedings of the 9th international conference on Supercomputing (ICS '95). ACM, New York, NY, USA, 240-244.

[2] *Deep Blue (chess computer)*. (2017, February 21). Retrieved March 02, 2017, from https://en.wikipedia.org/wiki/Deep$_{B}lue_{(}chess_{c}omputer)Negamax.(2016, November13).InWikipedia, TheFreeEncyc$
55, *March*3, 2017, $fromhttps : //en.wikipedia.org/w/index.php?title = Negamaxoldid =$ 749294742