RESEARCH REVIEW: DEEP BLUE

RESEARCH PAPER BY IBM WATSON TEAM

This research paper describes the techniques and rationale that went behind the design of Deep Blue chess machine that defeated the then-reigning World Chess Champion Garry Kasparov in a six-game match in 1997. At the start of the paper, the authors discuss the series of machines that led up to the creation of Deep Blue viz. ChipTest, Deep Thought, Deep Thought 2, Deep Blue I and Deep Blue II (referring to the computer that won against Garry Kasparov in 1997). Deep Blue inherited a lot of features from its previous counterparts; nonetheless, several improvements were made to make it competitive. Factors that contributed to its success are described in the subsequent paragraphs.

In comparison to Deep Blue I (the computer that lost to Garry Kasparov in 1996), Deep Blue II incorporated a **single-chip chess search engine with a more complex evaluation function** (with *increased number of features*) that was nearly 1.5 times faster than its predecessor. It was **a massively parallel system** with multiple levels of parallelism, with around 500 processors available to participate in the game tree searches. The system included significant **search extensions with non-uniform searches** so that it could *search to a reasonable minimum depth in the game tree*. **The gist of how it worked is as follows**: a master chip searched the top levels of the game tree and then distributed "leaf" positions to the workers who perform a few levels of additional search, and then distribute their leaf positions to the chess chips which search the last available levels of the tree.

The search mechanism of Deep Blue was a **hybrid software/hardware search**. The software search was flexible and could change as needed whereas the hardware search while inflexible, was faster. To maintain a balance between the speed of the hardware search and the efficiency/complexity of the software search, the chips only carried out shallow searches.

The chess chip was divided into three parts: *the move generator, the evaluation function and the search control.* The move generator, an 8x8 array of combinatorial logic (representing a chessboard) was controlled via a hardwired finite state machine. The move generator computed all possible moves. In order to generate moves with minimum latency and the first move being as close the best possible (to make search process efficient), the evaluation function of Deep Blue composed of "slow-evaluation" and "fast-evaluation". The features recognized in both evaluation functions had programmable weights for easy adjustment of their relative importance. The overall evaluation function was the sum of the feature values. Search Control monitored the quality of search by ensuring 'progress' with inclusion of components like a repetition detector.

The paper further goes into more details of the features that constituted the evaluation function, describing the heuristics used to score a particular game state. The game agent used ideas such as *quiescence search, iterative deepening, transposition tables, and NegaScout*. Deep Blue also had **two important databases** – *opening book* which was chosen to emphasize on the positions that Deep Blue played well and a large extended book that allows a large grandmaster game database to influence Deep Blue's play, particularly during endgames. To manage a suitable move in a required time frame, **time control** was also implemented.

The authors conclude their research listing out areas for additional improvement that could have otherwise resulted in better or worse results. This seminal paper is a great inspiration to understand the many possibilities that are available for creating a compelling game agent.