Param Sodhi

Professor Brooks

CS 386

23 September 2024

Assignment 2

a) What were the engineering advances that led to Deep Blue's success? Which of them can be transferred to other problems, and which are specific to chess?

Deep Blue's victory over Garry Kasparov was significant in the history of artificial intelligence for several reasons. Some of the technological advances that contributed to its success include specialized hardware, optimized search algorithms, and hueristics tailored to the chess subject at hand.

Deep Blue was built with custom hardware specifically designed to optimize chess computations. For example, the Deep Blue engine implemented parallel process capabilities, which means that multiple processors are used to evaluate various chess positions. This change allows Deep Blue to make decisions much faster than an engine using a single processor. Deep Blue also employs optimized algorithms, such as Alpha Beta. Alpha-beta, also known as pruning, works by stopping the evaluation of a move when it discovers an option that is worse than a previously considered move. It only passes alpha and beta values to node children and returns all other values to higher nodes. This addition reduces computational burden. Deep Blue software also had a large database of chess knowledge and access to libraries, allowing it to look up positions rather than having to calculate each scenario each time.

All three of these overarching ideas can and are being used in other areas today. For example, numerous engines now benefit from specialized hardware that focuses on the task at hand rather than a generic piece of hardware. Some large companies include Microsoft and Apple, who do build specialized hardware for their programs. AlphaBeta pruning can be applied to processes that require any sort of decision-making, and the concept of using domain-specific knowledge to train your model enables individuals to create more sophisticated and responsive models. Overall, Deep Blue was a step forward in enhancing how AI affects our world and how we go about creating engines.

b) AlphaZero is compared to a number of modern game-playing programs, such as StockFish, which work similarly to Deep Blue. The paper shows that AlphaZero is able to defeat StockFish even when it is given only 1/100 of the computing time. Why is that? Please frame your answer in terms of search and the number of nodes evaluated.

AlphaZero sets itself apart from other game engines, such as Stockfish, by employing a unique search strategy and techniques that always prioritize quality parsing over quantity parsing. The distinction between AlphaZero and Stockfish is that they use different types of searching algorithms to process a game. On the one hand, Stockfish employs a brute force approach, which means that they attempt to evaluate as many moves as possible within the depth of their search. It uses Alpha-Beta search in the same way that Deep Blue does, which, while efficient, requires a large number of calculations to be considered effective. AlphaZero, on the other hand, began using a technique known as selective search based on a tree strategy. Instead of increasing the number of nodes, AlphaZero opts to use an algorithm known as MCTS. After a quick Google search, I discovered that MCTS stands for Monte Carlo Tree Search. This MCTS

employs deep neural networks to learn optimal moves and game strategies via extensive self-play. This approach enables AlphaZero to concentrate computational resources on the most relevant and significant moves rather than extending attempts in a wide range of possibilities. In the piece entitled "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," it reinforces this concept with the statement: "AlphaZero searches just 80 thousand positions per second in chess and 40 thousand in shogi, compared to 70 million for Stockfish and 35 million for Elmo." AlphaZero takes a more "human-like" approach to searching. Instead of using brute force, it simulates games to generate a probability distribution over the next moves, assigning a likelihood to each possible move based on how frequently that move was visited in its simulations and how successful those moves were. As a result, AlphaZero can make more informed and effective decisions with only 1/100 of the computing time that StockFish requires.