The Role of Games in Artificial Intelligence

Murray Campbell
IBM T.J. Watson Research Center
mcam@us.ibm.com

Games as Test Beds for AI Research

Games are excellent test beds for exploring many types of artificial intelligence research. They abstract some aspects of complex real-world domains, allowing the problem solver to focus on critical issues of search and knowledge representation. This fact was recognized in the infancy of the computer age, most notably by Claude Shannon[4] in his seminal paper on computer chess: "(1) The problem is sharply defined both in the allowed operations (the moves) and in the ultimate goal (checkmate); (2) it is neither so simple as to be trivial nor too difficult for satisfactory solution; (3) chess is generally considered to requiring 'thinking' for skillful play; a solution of this problem will force us either to admit the possibility of mechanized thinking or to further restrict our concept of 'thinking'; (4) the discrete structure of chess fits well into the digital nature of modern computers."

The advantages of games for AI research become clearer when considering some of the complexities of the real world:

1. There are multiple goals, sometimes conflicting, often poorly defined
2. The allowable actions are rarely explicit, change over time, and may be effectively infinite
3. Most relevant information is either hidden or too voluminous to absorb and process
4. The behavior and motivations of other agents is hard to discern and predict
5. Building sensors and effectors to interact with the world is a significant technical challenge all by itself.

Games make simplifying assumptions on these conditions, perhaps selectively allowing some complexity in one or two dimensions. Otherwise, the problem is so daunting that it is hard to even know where to begin.

This paper will examine some of reasons why games have been and continue to be excellent domains for Artificial Intelligence research. Different games stress different aspects of intelligence, which allows progress to be made in each area. Deep Blue, the first chess-playing computer to defeat the human world champion in a regulation match, will be described in some detail, with emphasis on the search and evaluation function. Finally, a number of areas for future research are described.

Issues Explored in Game-Playing Systems

A wide variety of games have been the subject of AI research[3]. To achieve high performance in these games,

Figure 1: Deep Blue defeated Garry Kasparov in a 1997 match.

many issues must be explored, including search control, real-time decision making, imperfect information, randomness, learning, and multi-agent environments.

Almost all games require some form of search control, i.e., a systematic method of exploring possible future game states. This is because the size of the search space is too large to exhaustively explore. Selective search is one option for controlling search, exploring more deeply those states that are deemed most important. An indirect approach is state abstraction, combining multiple game states into one, which also reduces complexity.

Most games have real-time aspects to them. Making a reasonable decision in a limited time is a key area of research, and numerous techniques have been developed to address this issue. For example, the technique of "iterative deepening" (successively exploring deeper and deeper in the move tree) ensures that there is always a reasonable move to be made. In some cases, it is possible to reason explicitly about the expected benefit of additional computation versus its cost, using decision theoretic techniques.

In many games, information is hidden from the players. It is typical to hypothesize properties of this missing information based on evidence that is revealed as the game progresses and make move decisions based on these assumptions. For example, one could conduct Monte Carlo simulations to identify a good move.

Randomness in games is in part a search control problem, where the value of a future move being explored must be weighted by the probability of its occurrence. Monte Carlo simulation is also one technique that is useful to deal with randomness.

The use of learning to improve game-playing behavior has long been an important research area, going back to Arthur Samuel's checkers program in the 1950s. A wide variety of techniques have been applied, with varying degrees of success. However, the goal of very high-performance game playing systems that are able to learn completely from scratch has only be realized in very limited domains.

Multi-player games add an additional layer of complexity. Standard searching methods often become intractable, and it is often necessary to model the opponents to better predict them, thereby constraining the search.

Case Study: The Deep Blue Chess System

The Deep Blue[1)] computer chess system was developed at IBM Research in the mid-1990s. In 1997, Deep Blue became the first computer to defeat the reigning World Chess Champion in a regulation match (**Figure 1**).

The most obvious feature of Deep Blue was its computational power. Deep Blue was a massively parallel system, comprised of a 30-processor IBM RS/6000 SP computer and 480 single-chip chess search engines. The SP computer (leftmost in **Figure 2**) was made up of two frames, each containing 15 general-purpose computers (middle in **Figure 2**). Each SP processor, in addition to the standard components,

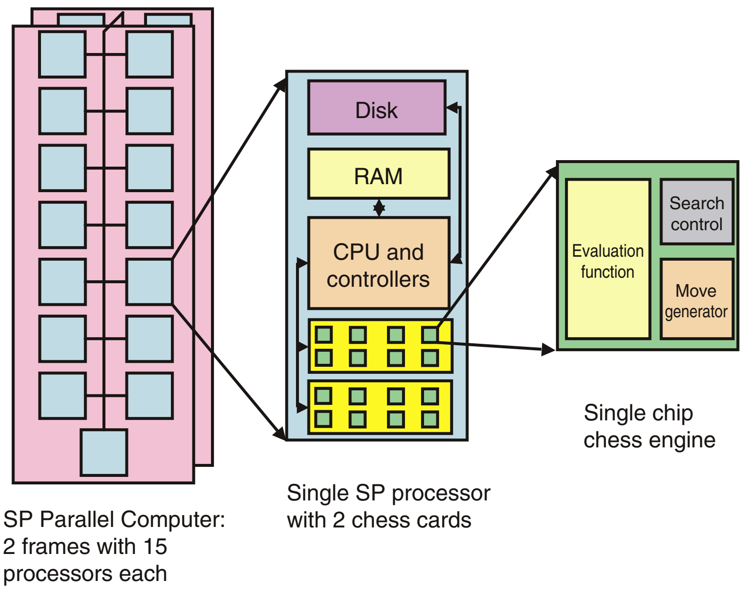

Figure 2: The components of the Deep Blue system.

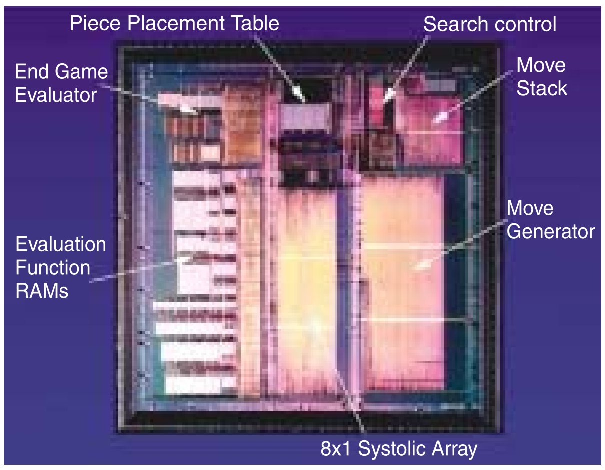

Figure 3: The VLSI single-chip chess search engine used in Deep Blue.

contained two microchannel cards, each of which contained 8 single-chip chess search engines (rightmost in **Figure 2**). Each chip was capable of examining over 2 million chess positions per second (**Figure 3**).

Deep Blue managed the parallelism through the use of a static processor tree. One SP processor was designated as the master, and it controlled the remaining 29 SP processors. The master processed the levels of the search tree closest to the root. It then passed jobs to the worker processors. Each of the workers processed additional levels in the search tree, and then passed on jobs to the single-chip search engines (**Figure 4**).

The overall average Deep Blue system speed was 126 million chess positions per second. This is in stark contrast to human Grandmaster chess players, who have been observed to examine 1-2 chess positions per second. Clearly the human approach to chess is dramatically different to that taken by Deep Blue.

In spite of this huge speed advantage, a naive brute-force approach to chess would have been insufficient to defeat the top human Grandmasters. In part, this is because brute-force approaches to chess (and most other games) are exponential with search depth. This severely limits the maximum depth of search, and human Grandmasters can easily demonstrate the ability to search well beyond the maximum depth

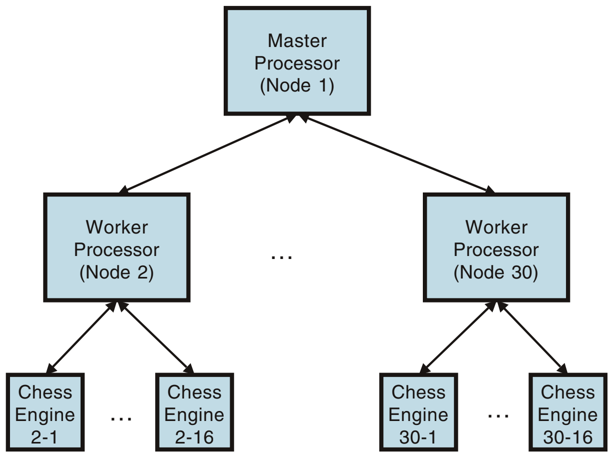
Master Processor (Node 1)
Worker Processor (Node 2)
…
Worker Processor (Node 30)
Chess Engine 2-1
…
Chess Engine 2-16
Chess Engine 30-1
…
Chess Engine 30-16

Figure 4: Deep Blue is organized as a 3-level hierarchy.

a brute-force system can reasonably be expected to attain. The key for human players is "selective search", i.e., searching the tree of possible chess moves to a variable depth. Properly done, this allows "important" move sequences to be searched more deeply, and "unimportant" sequences to be searched less deeply. **Figure 5** illustrates this idea. On the left, a brute force search can reach a given fixed depth. A selective search, on the right, allocates some of the available resources to searching more deeply following important positions (shown as stars in the figure). This allows the selective search to search more deeply on important moves.

Considerable effort was spent on implementing and improving the effectiveness of Deep Blue's selective search. Our approach was to "extend" interesting sequences more deeply. A second approach, complementary to the first, is to "prune" uninteresting sequences early. Deep Blue's selective search was purely extension based (we considered pruning too risky) and used a novel algorithm called "dual credit with delayed extensions", which attempted to adhere to the following principles:

1. Extending forced/forcing pairs of moves: The most important method humans use to search deeply is to examine forcing sequences in greater detail. A forcing sequence is one where one or both sides have very few reasonable choices of moves (other choices lead to an immediate loss). We used a method based on our "singular extensions" algorithm to identify forcing sequences.
2. Forced moves are expectation dependent. A move can be forced for a player if that player expects to have an advantage, but not forced if the player expects the position to be even.
3. Fractional extensions: It is usually not feasible to fully extend forcing sequences, since this can cause the search to "explode". It is possible to measure the "forcedness" of a sequence, and extend the sequence in proportion to the degree of forcedness.
4. Delaying extensions: A series of forcing moves is usually much more interesting than an isolated forcing move, so delaying an extension until additional forcing moves are seen can be quite effective.
5. Dual credit: Chess is a two-player game, and sometimes both sides can be forced in sequence. The dual credit system ensures that the search is bounded.
6. Preserve the search envelope: Under some conditions a selective search can loop unless careful track is kept of previously visited states. These states are recorded in a hash table.

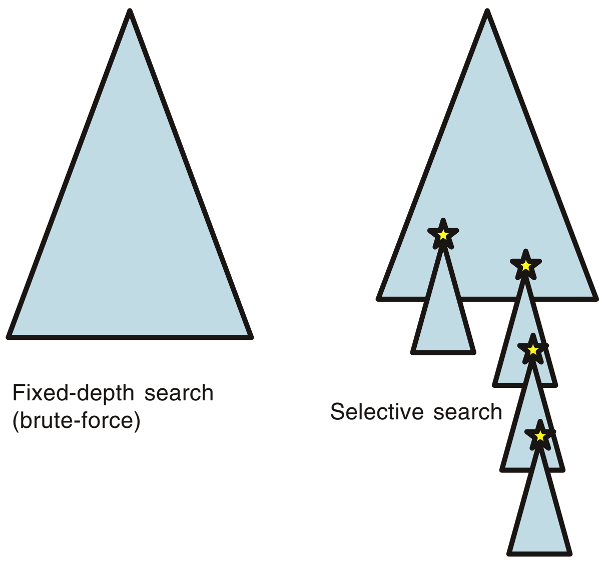

Figure 5: Brute-force versus selective search.

The resultant search was highly selective. A brute-force Deep Blue would have searched roughly 14 ply ahead (a ply is a white move or a black move). The Deep Blue selective search algorithm ensured that all move sequences were examined to a minimum of 12 ply, but was able to search the important move sequences 30-40 ply deep. This is undoubtedly an important factor in Deep Blue's 1997 victory.

A less visible but equally important aspect of Deep Blue was the "evaluation function". An evaluation function, given a chess position, produces a scalar value representing the goodness of the position. Positive values are good for white, while negative values are good for black. The Deep Blue evaluation function was quite complex, recognizing approximately 8000 patterns, each of which was assigned a value.

The complexity of the Deep Blue evaluation function was a mixed blessing. On the one hand, many important features relevant to assessing a chess position allow a more accurate evaluation. On the other hand, assigning appropriate number values to each of the patterns was an extremely difficult problem. It is particularly true because the values are context dependent, varying with the characteristics of the current state. We experimented with a number of methods for automatically tuning these values, with limited success. It is fair to say that the best way to tune a complex chess evaluation function is an open research problem, and that the "evaluation function" of human Grandmasters is in many cases superior to Deep Blue.

One other interesting aspect of Deep Blue that is relevant to AI is the "extended book". The extended book was a mechanism to allow Deep Blue to take advantage of past human Grandmaster experience. A collection of 700,000 Grandmaster games were processed, and for some of the positions reached in these games, a score was computed estimating the value of each of the moves that Grandmasters had previously played. The score was based on a large number of factors, including the number of times a move had been played, the strength of the players making the move, and the results of the move. Deep Blue would use this information to bias its search, preferring moves that had high scores. Of course if Deep Blue found a strong new move that had never been played before, it was free to choose it.

To summarize, Deep Blue used a radically different approach to playing chess than a human Grandmaster. While Deep Blue had a less sophisticated evaluation function than human players, it could compensate for this by means of an incredibly thorough and deep search. This, along with various other techniques, allowed Deep Blue to play chess at the world-class level.

Future Directions

Steady progress continues to be made in improving the performance of PC-based chess programs. Some commercially available programs (e.g., Deep Fritz, Deep Junior) are now challenging the world's top Grandmasters. However, the focus of game-playing research has shifted to a number of other areas.

Creating effective learning systems is a serious challenge for many games. In chess, for example, preliminary steps have been taken to learn feature values in an evaluation function, using either examples for Grandmaster play or by reinforcement-learning-style approaches. There is considerable work to do before an evaluation function can be learned from scratch, especially in discovering potentially useful features (although Gerry Tesauro's TD-Gammon is a notable exception).

In the area of search, games with large branching factors (a large number of moves available at each turn) are proving difficult for computer players. The main reason is that techniques that have proven successful in games like chess and checkers do not scale well to games with large branching factors. Highly selective search is essential for such games, i.e., search in a more human style.

Games with imperfect information and randomness present a whole new set of problems. In a card game like bridge, Matthew Ginsberg's program GIB has sophisticated methods for dealing with hidden cards, and is approaching world-class level. In poker, another card game, opponent modeling is an important factor (because of bluffing), and a group at the University of Alberta has developed a program that is also approaching world-class level.

While research in traditional strategy games like chess and Go continues, the future of Artificial Intelligence game playing research seems to be moving towards interactive games[2]. There are significant benefits in developing intelligent characters and complex virtual worlds that make interactive games more realistic and interesting. Each genre of interactive computer games (e.g., action, role-playing, adventure, strategy, sports) have their own unique challenges. The game roles that AI can help with include believable tactical enemies, partners, support characters, etc. Interactive games will provide a wide variety of challenges to AI researchers for the foreseeable future.

Acknowledgement I would like to thank Feng-hsiung Hsu and A. Joseph Hoane, Jr., my collaborators on the Deep Blue project.

References

1) Campbell, M., Hoane, A.J. and Hsu, F.H.: Deep Blue, Artificial Intelligence, Vol.134, pp. 57-83 (2002).

2) Laird, J.E. and van Lent, M.: Human-level AI's Killer Application: Interactive Computer Games, AAAI National Conference on Artificial Intelligence (2000).

3) Schaeffer, J.: A Gamut of Games, AI Magazine, Vol.22, No.3, pp.29-46 (2001).

4) Shannon, C.: Programming a Computer for Playing Chess, Philosophical Magazine, Vol.41, pp.256-275 (1950).

(Recieved October 5, 2003)