The role of games in understanding computational intelligence

Jonathan Schaeffer, University of Alberta

The AI research community has made one of the most profound contributions of the 20th century to mankind's knowledge. This research has led to the realization that intelligence is not uniquely human. Using computers, it is possible to achieve human-like behavior in nonhumans. In other words, it is possible to create the illusion of human intelligence in a computer.

This idea has been vividly illustrated throughout the history of computer games research. Unlike most of the early work in AI, game researchers were interested in developing high-performance, real-time solutions to challenging problems. This led to an ends-justify-the-means attitude: the result—a strong chess program, for example—was all that mattered, not the means by which it was achieved. In contrast, much of the mainstream AI work used simplified domains, while eschewing real-time performance objectives. This research typically used human intelligence as a model: all one had to do was emulate the human example to achieve intelligent behavior. The battle (and philosophical) lines were drawn.

The difference in philosophy can be easily illustrated. The human brain and the computer are different machines, each with its own sets of strengths and weaknesses. Humans are good at, for example, learning, reasoning by analogy, and image processing. Computers are good at numeric calculations, repetitious computations, and memorizing large sets of data. These machine architectures are largely complimentary: the human's processing strengths are the computer's weaknesses and the computer's strengths are human weaknesses. Given a problem to be solved and a specified architecture (human brain or silicon computer), a good solution should cater to the strengths of the machine being used, not the weaknesses. When viewed in this light, it is not surprising that the unhuman-like approaches have won out.

Building high-performance game-playing programs has been one of AI's major triumphs. This is due, in part, to the success achieved in games such as backgammon, chess, checkers, Othello, and Scrabble, where computers are playing as well as or better than the best human players. However, it is also due to the examples it set to the research community. These include tackling challenging problems (rather than trivial subsets, as is still often seen in AI research) and the emphasis on the results of the system without regard for the methods used to achieve those results (the ends justify the means).

This essay illustrates a number of techniques used by game-playing programs to achieve the illusion of human-like intelligence.

Brute-force search

Humans are poor searchers. They cannot search quickly, and usually not optimally. In contrast, computers are very good at search. Considering millions of possibilities per second, looking for a solution in a maze (or tree) of possibilities is easy to do in a computer. However, humans are very good at discovering, generalizing, and using knowledge; computers are primitive in comparison. Even after 50 years of research, no one understands how to represent and manipulate knowledge effectively. Hence, many computer-based solutions for games programs trade off knowledge for search. They use large, deep searches to compensate for inadequate knowledge (so-called brute-force search, sometimes used in a derogatory context). Search itself is dynamic knowledge.

This idea culminated in the Deep Blue victory in an exhibition match with world chess champion Garry Kasparov in 1997. Deep Blue used a 32-processor IBM SP-2 computer, with each processor connected to 16 specially designed chess chips.[1] Each chip was capable of quickly searching large chess trees. The result was a chess machine whose search considered all possible moves at least 12 ply (one ply is one move by one side) into the future, while selectively extending the search considerably deeper for interesting moves.

Deep Blue searched 200 million chess positions *per second*.[1] Kasparov considered two. Given those numbers, to some observers it was not a surprise that the computer won the match. The real surprise was how long humans have been able to withstand the technological onslaught. The next generation of Deep Blue chess chips will be 10 times faster than those used in the Kasparov match. Improved computer technology improves the perceived "intelligence" of compute-bound AI applications.

Brute-force search is now an accepted tool in the AI researchers' toolbox, and has been used to achieve many notable successes (with games and in other domains).

Large memory

Computer memory is cheap. The price per byte of disk storage is plummeting. This trend will continue for the foreseeable future.

Human memories are notoriously fallible, have a fixed capacity, and degenerate with time. Storing large amounts of data is impractical. Humans compensate for this by distilling large amounts of knowledge into a manageable number of rules (or heuristics) that are effective at reconstructing the data. Computer memory, on the other hand, can be made infallible, can be expanded to fit the needs, and can be preserved forever. Hence, a brute-force approach to storage can be used: save everything.

The Chinook checkers program (8×8 draughts) uses this idea.[2] The game theoretic result (win, loss, or draw) for all positions with eight or fewer pieces on the board were computed: roughly 444 billion positions (4.44×10^{11}). This knowledge lets the program play perfectly when it reaches a position in the database (the program will always win a won position and never lose a drawn position). It also significantly affects the search, in that it introduces perfect knowledge. It is not uncommon to see a position with 20 pieces on the board being searched deeply enough to back up a database score to the root of the search. Typically, Chinook can announce the final result of the game (assuming that the human opponent does not make a mistake) within 15 moves of the start.

The database was compressed into six gigabytes for real-time decompression. While the early versions of the program were I/O bound, continually accessing the database on disk to look up position values during a search, the current version preloads the entire database into random-access memory. The resulting speed benefits only serve to widen the gap between Chinook's capabilities and what the best humans can achieve.

Unexplainable knowledge

How does one acquire knowledge for use in a game-playing program? The traditional approach is to consult human domain experts and attempt to distill rules for strong play from them. This is a difficult task, especially given the difficulty humans have in expressing their subconscious decision-making processes. For a computer solution, one ideally wants to eliminate the weak link—the human expert. The computer should be able to discover and refine all the knowledge it needs. While this goal remains elusive in general, there have been some notable successes in the games domain.

The Othello program Logistello plays Othello better than all humans.[3] The program evaluates positions using 11 patterns that, with reflections and rotations, comes to 46. Assigning a score to each possible value for each pattern results in roughly 1.2 million evaluation scores that need to be determined. Using self-play (Logistello playing games against itself) and linear regression, the program can incrementally learn the value of these parameters. With relatively little effort (roughly a month of computation), the program achieves world-class ability. In effect, this is a brute-force approach to integrating knowledge. Program designers can include as much (or as little) knowledge as they like, and let the parameter-learning process decide what is used and how important it is. The TD-Gammon backgammon program pioneered similar techniques where self-play and temporal difference learning of a neural net resulted in play that is comparable to that of the human world champion.[4]

There is little in the way of useful information that humans can extract from the large number of seemingly random numbers with which Logistello uses to evaluate positions. Just as computer program designers have difficulty understanding human knowledge, so too do humans have difficulty understanding computer "knowledge."

Simulations

The early research into games was restricted to two-player, perfect-information games. With the decline of interest in chess research in the 1990s, efforts switched to other games, including those with multiple players and having imperfect information. Imperfect information provides an interesting challenge. In these types of games, humans observe their opponent's actions and make inferences as to the missing information. Human intuition and experience can be amazingly accurate. Computers have difficulty approximating experience and intuition, but are capable of precise probability calculations. The computer's solution is simulation: instantiate the missing information many times, each time calculating the likely outcome. In this way, a statistical profile can be obtained that indicates the computer's best course of action.

Bridge[5], poker[6], and Scrabble[7] programs—all games of imperfect information—use similar techniques to achieve their success: they simulate hundreds or thousands of scenarios. For example, the bridge program GIB decides on what card to play by simulating the play of the hand.[5]The program internally deals out cards that are consistent with the bidding to the opponents. It then plays out the hand to the end to see which card play leads to the best result (most tricks won). It repeats this process roughly 100 times, each time with different cards for the opponent. After enough simulations, it becomes clear which card play, on average, leads to the best result. The program does not understand well-known bridge concepts such as finesse or squeeze; everything is done using uninformed search.

Comprehension without understanding

Documents contain human-understandable information. Because this information was designed to be easily understood by a human (such as text, sounds, images), it is often difficult for a computer to figure it out. This communications gap between man and machine is an imposing obstacle to technology. No one wants to re-express human knowledge in computer-understandable terms unless absolutely necessary. We need to make computers understand documents. The computer solution is to create the illusion of understanding, without actually doing any comprehension.

The Proverb program solves crossword puzzles.[8] Human crossword solvers use the semantics of the clues to deduce the answers needed for the puzzle. The clues are intended for a human audience, and include a combination of factual information, common knowledge, missing words, and word plays. Writing a computer program to understand the semantics is a challenging problem. Proverb's solution is to not understand the clues. It includes a variety of solvers (or agents) that examine the words in the clues to identify likely answers. For example, Proverb uses a database of clues and answers from previous puzzles to find an answer to a clue 34% of the time. Specialized agents can go out on the Internet and query, for example, dictionary, history, geography, and movie databases. Each agent returns a set of answers. Proverb combs through the plausible answers, trying to fit them into the puzzle grid and satisfy all the constraints. Proverb scores 95% letters correct on the *New York Times* crossword puzzles, without understanding the meaning of any of the clues.

Comprehension without understanding is a powerful technique. Similar ideas are being applied to, for example, classifying Web pages. Given an arbitrary Web page, is it possible to classify its type (its personal page, company, course, and so forth)? Statistical techniques can do an effective job. For example, counting the frequency of words appearing on the page can be a powerful indicator of the type of the page. No human would ever explicitly compute word frequencies to do this.

Computer games research is a microcosm for AI research. By liberating the computer from blindly following the human example, amazing feats of intelligence are possible with relatively little computer programming effort. Our notion of intelligence will never be the same.

References

1. F-h. Hsu, "IBM's Deep Blue Chess Grandmaster Chips," IEEE Micro, March-April, 1999, pp. 70-81.
2. J. Schaeffer, *One Jump Ahead*, Springer-Verlag, 1997.
3. M. Buro, "Logistello—A Strong Learning Othello Program," 1997, www.neci.nj.nec.com/homepages/mic/ps/log-overview.ps.gz.
4. G. Tesauro, "Temporal Difference Learning and TD-Gammon", Communications of the ACM, Vol. 38, No. 3, 1995, pp. 58-68.
5. M. Ginsberg, "GIB: Steps Toward an Expert-Level Bridge-Playing Program," International Joint Conference on Artificial Intelligence, 1999, pp. 584-589.
6. D. Billings, L. Peña, J. Schaeffer and D. Szafron, "Using Probabilistic Knowledge and Simulation to Play Poker," AAAI National Conference, 1999, pp. 697-703.
7. B. Sheppard, private communication, October, 1998.
8. G. Keim, N. Shazeer, M. Littman, S. Agarwal, C. Cheves, J. Fitzgerald, J. Grosland, F. Jiang, S. Pollard and K. Weinmeister, "Proverb: The Probabilistic Cruciverbalist," AAAI National Conference, 1999, pp. 710-717.

Jonathan Schaeffer is a professor in the Department of Computing Science at the University of Alberta. His research interests include heuristic search and parallel-computing environments. He received a BSc from the University of Toronto and an M.Math and a PhD from the University of Waterloo. He is the author of the checkers program Chinook, the first program to win a human world championship in any game. He has developed world-class programs for chess and poker, as well as worked on Sokoban. He is a member of the IEEE, ACM, AAAI, and ICCA. Contact him at the Dept. of Computing Science, Univ. of Alberta, Edmonton, Alberta, Canada T6G 2H1; jonathan@cs.ualberta.ca; www.cs.ualberta.ca/~jonathan.