AI Magazine Volume 22 Number 3 (2001) (© AAAI)

A Gamut of Games

Jonathan Schaeffer

■ In 1950, Claude Shannon published his seminal work on how to program a computer to play chess. Since then, developing game-playing programs that can compete with (and even exceed) the abilities of the human world champions has been a long-sought-after goal of the AI research community. In Shannon's time, it would have seemed unlikely that only a scant 50 years would be needed to develop programs that play world-class backgammon, checkers, chess, Othello, and Scrabble. These remarkable achievements are the result of a better understanding of the problems being solved, major algorithmic insights, and tremendous advances in hardware technology. Computer games research is one of the important success stories of AI. This article reviews the past successes, current projects, and future research directions for AI using computer games as a research test bed.

Games are ideal domains for exploring the capabilities of computational intelligence. The rules are fixed, the scope of the problem is constrained, and the interactions of the players are well defined. Contrast the game world with the real world—the game of life—where the rules often change, the scope of the problem is almost limitless, and the participants interact in an infinite number of ways. Games can be a microcosm of the real world (for example, the role of game theory in economics, social interaction, and animal behavior), and successfully achieving high computer performance in a nontrivial game can be a stepping stone toward solving more challenging real-world problems.

Historically, games have been a popular choice for demonstrating new research ideas in AI. Indeed, one of the early goals of AI was to build a program capable of defeating the human world chess champion in a match. This challenge proved to be more difficult than was anticipated; the AI literature is replete with optimistic predictions. It eventually took almost 50 years to complete the task—a remarkably short time when one considers the software and hardware advances needed to make this amazing feat possible. Often overlooked, however, is that this result was also a testament to human abilities. Considering the formidable computing power that DEEP BLUE used in its 1997 exhibition match against world chess champion Garry Kasparov (machine: 200,000,000 chess positions a second; man: 2 a second), one can only admire the human champions for withstanding the technological onslaught for so long.

Computer game research was started by some of the luminaries in computing science history. In 1950, Claude Shannon published his seminal paper that laid out the framework for building high-performance game-playing programs (Shannon 1950). In 1951, Alan Turing (1953) did a hand simulation of his computer chess algorithm (a lack of resources prevented him from actually programming it); the algorithm lost to a weak human player. Around this time, Arthur Samuel began work on his famous checkers-playing program, the first program to achieve notable success against human opposition (Samuel 1967, 1959). By 1958, Alan Newell and Herb Simon had begun their investigations into chess, which eventually led to fundamental results for AI and cognitive science (Newell, Shaw, and Simon 1958). An impressive lineup to say the least!

In the half century since Shannon's paper, enormous progress has been made in constructing high-performance game-playing programs. In Shannon's time, it would have seemed unlikely that within a scant 50 years checkers (8×8 draughts), Othello, and Scrabble programs would exist that exceed the abilities of the best human players,[1,2] and backgammon and chess programs could play at a level comparable to the human world champion. These remarkable accomplishments are the result of a better understanding of the problems being solved, major algorithmic insights, and tremendous advances in hardware technology. The work on computer games has been one of the most successful and visible results of AI research. For some games,

Considering the formidable computing power that DEEP BLUE used in its 1997 exhibition match against world chess champion Garry Kasparov (machine: 200,000,000 chess positions a second; man: 2 a second), one can only admire the human champions for withstanding the technological onslaught for so long.

one could argue that the Turing test has been passed (Krol 1999).

This article discusses the progress made in developing programs for the classic board and card games. For a number of games, a short history of the progress in building a world-class program for the game is given, along with a brief description of the strongest program. In each case, a single feature of the program that is a major contributor to the program's strength is highlighted. The histories are necessarily brief. I apologize in advance to the many hard-working researchers and hobbyists whose work is not mentioned here.

Enabling Technologies briefly summarizes some of the major advances in technology that facilitated the construction of world-championship–caliber programs. Success in Computer Games reports the past successes where computers have met or exceeded the best human players (backgammon, checkers, chess, Othello, and Scrabble). Current Research Efforts highlights games of current academic interest (bridge, go, and poker). The Future of Computer Games discusses some of the future challenges of using games as a research test bed for AI.

Although this article emphasizes the AI viewpoint, one should not underestimate the engineering effort required to build these programs. One need only look at the recent success of the DEEP BLUE chess machine to appreciate the effort required. This project spanned 8 years (12 if one includes the pre-IBM time) and included several full-time people, extensive computing resources, chip design, and grand master consultation. Some of the case studies hint at the amount of work required to construct these systems. In all cases, the successes reported in this article are the result of consistent progress over many years.

Enabling Technologies

The biggest advances in computer game playing have come as a result of work done on the alpha-beta search algorithm. This algorithm received the most attention because of the research community's preoccupation with chess. With the DEEP BLUE victory over world chess champion Garry Kasparov, interest in methods suitable for chess has waned and been replaced by activity in other games. One could argue that the chess victory removed a ball and shackle that was stifling the creativity of researchers who were building high-performance game-playing systems.

The alpha-beta research led to a plethora of search enhancements, which significantly improved the efficiency of the search. Some of these enhancements include iterative deepening, caching previously seen subtree results (transposition tables), successor reordering, search extensions and reductions, probabilistic cutoffs, and parallel search. The results are truly amazing. Even though there is an exponential difference between the best case and the worst case for an alpha-beta search, most high-performance game-playing programs are searching within a small constant of the best case (Plaat et al. 1996).

Sadly, the community of researchers involved in this work has done a relatively poor job of selling the technology, resulting in many of the ideas being reinvented for other domains. For example, many search techniques pioneered with alpha-beta have become standard in other search domains, with few realizing the lineage of the ideas.

At the heart of many game-playing programs is an evaluation function. Early on, game developers quickly encountered the knowledge-acquisition bottleneck and traded quality of knowledge for speed of the program. Simple evaluation functions, linear combinations of easily identifiable features, were the mainstay of computer game programs for many decades. Alternative approaches, such as modeling human cognitive processes, turned out to be much harder to do than initially expected and generally resulted in poor performance. Game programmers quickly learned that a little heuristic knowledge, when combined with deep search, can produce amazing performance results. Indeed, one could argue that the viability of brute-force search, once a term with negative connotations in the AI community, is one of the main research results from games-related research (Ginsberg 1996b).

In the last decade, new techniques have moved to the forefront of games research. Two in particular are mentioned here because they are likely to play a more prominent role in the near future: (1) Monte Carlo simulation and (2) temporal-difference learning.

Monte Carlo simulation has successfully been applied to games with imperfect or nondeterministic information. In these games, it is too expensive to search all possible outcomes. Instead, only a representative sample is chosen to give a statistical profile of the outcome. This technique has been successful in bridge, poker, and Scrabble.

Temporal-difference learning is the direct descendent of Samuel's machine learning research (Sutton 1988). Here, a database of games (possibly generated by computer self-play) can be used to bootstrap a program to find a good combination of knowledge fea-

tures. The algorithm has successfully been applied to backgammon and has recently shown promise in chess and checkers (Schaeffer, Hlynka, and Jussila 2001).

The most obvious hardware advance is simply speed. To read about Samuel's checkers-playing program running on a 1963 machine that could execute 15 million additions a minute (Pfeiffer 1964) starkly brings home the point that orders of magnitude more computing power makes many things possible. Indeed, considering the paucity of computing power at Samuel's disposal, one can only be filled with admiration at what he achieved.

Computer games research pioneered competitions to assess the quality of the systems being developed. Since 1970, there have been annual computer chess tournaments. There is now an annual Computer Olympiad that brings together many of the top programs and their developers in head-to-head competition.[3] The competitive spirit has spread throughout the AI community; competitions now exist for other applications, including theorem proving, planning, and natural language.

Success in Computer Games

In a number of games, computers have enjoyed success that puts them on par or better with the best humans in the world. In some sense, these games are now the past, in that active research to develop high-performance programs for them is on the wane (or is now nonexistent). These include games where computers are better than all humans (checkers, Othello, Scrabble) and those where computers are competitive with the human world champion (backgammon, chess).

Backgammon

The first concerted effort at building a strong backgammon program was undertaken by Hans Berliner of Carnegie Mellon University. In 1979, his program, BKG9.8, played an exhibition match against the then newly crowned world champion Luigi Villa (Berliner 1980a, 1980b). The stakes were $5,000, winner take all. The final score was seven points to one in favor of the computer, with BKG9.8 winning four of the five games played (the rest of the points came from the doubling cube).

Backgammon is a game of both skill and luck. In a short match, the dice can favor one player over another. Berliner writes that "in the short run, small percentage differences favoring one player are not too significant. However, in the long run a few percentage points are highly indicative of significant skill differences" (Berliner 1980a, p. 215). Thus, assessing the results of a five-game match is difficult. Afterwards, Berliner (1980b, p. 71) analyzed the program's play and concluded that

> There was no doubt that BKG9.8 played well, but down the line Villa played better. He made the technically correct plays almost all the time, whereas the program did not make the best play in eight out of 73 non-forced situations.

BKG9.8 was an important first step, but major work was still needed to bring the level of play up to that of the world's best players.

In the late 1980s, IBM researcher Gerry Tesauro began work on a neural net–based backgammon program. The net used encoded backgammon knowledge and, training on data sets of games played by expert players, learned the weights to assign to these pieces of knowledge. The program, NEUROGAMMON, was good enough to win first place in the 1989 Computer Olympiad (Tesauro 1989).

Tesauro's next program, TD-GAMMON used a neural network that was trained using temporal difference learning. Instead of training the program with data sets of games played by humans, Tesauro was successful in having the program learn using the temporal differences from self-play games. The evolution in TD-GAMMON from version 0.0 to 3.0 saw an increase in the knowledge used, a larger neural net, and the addition of small selective searches. The resulting program is acknowledged to be on par with the best players in the world and, possibly, even better.

In 1998, an exhibition match was played between world champion Malcolm Davis and TD-GAMMON 3.0 (at the AAAI-98 conference). To reduce the luck factor, 100 games were played over 3 days. The final result was a narrow eight-point win for Davis. Both Davis and Tesauro have done extensive analysis of the games, coming up with similar conclusions:[4]

> While this analysis isn't definitive, it suggests that we may have witnessed a superhuman level of performance by TD-GAMMON, marred only by one horrible blunder redoubling to 8 in game 16, costing a whopping 0.9 points in equity and probably the match!

A notable feature of TD-GAMMON is its neural net evaluation function. The net takes as input the current board position and returns as output the score for the position (roughly, the probability of winning) (Tesauro 1995). The net has approximately 300 input values (Tesauro 2001). The latest version, TD-GAMMON 3.0, contains 160 hidden units. Each unit takes a linear

Sadly, the community of researchers involved in this work has done a relatively poor job of selling the technology, resulting in many of the ideas being reinvented for other domains. For example, many search techniques pioneered with alpha-beta have become standard in other search domains, with few realizing the lineage of the ideas.

sum of the weighted values of its input and then converts it to a value in the range –3 to 3 (a backgammon is worth three points, a gammon two, and a win one). The conversion is done with a sigmoid function, allowing the output to be a nonlinear function of the input. The resulting neural net has approximately 50,000 weights that need to be trained.

The weights in the hidden units were trained using temporal-difference learning from self-play games. By playing the program against itself, there was an endless supply of data for the program to train itself against. In a given game position, the program uses the neural net to evaluate each of the roughly 20 different ways it can play its dice roll and then chooses the move leading to the maximum evaluation. Each game is played to completion, and then temporal-difference learning is applied to the sequence of moves. Close to 1,500,000 self-play games were used for training TD-GAMMON 3.0.

Tesauro's success with temporal-difference learning in his backgammon program is a major milestone in AI research.

Tesauro's success with temporal-difference learning in his backgammon program is a major milestone in AI research.

Checkers

Arthur Samuel began thinking about a checkers program in 1948 but did not start coding until a few years later. He was not the first to write a checkers-playing program; Christopher Strachey (1952) predated him by a few months. Over the span of three decades, Samuel worked steadily on his program, with performance taking a back seat to his higher goal of creating a program that learned. Samuel's checkers player is best known for its single win against Robert Nealey in a 1963 exhibition match. From this single game, many people erroneously concluded that checkers was a "solved'" game.

In the late 1970s, a team of researchers at Duke University built a strong checkers-playing program that defeated Samuel's program in a short match (Truscott 1979). Early success convinced the authors that their program was possibly one of the 10 best players in the world. World champion Marion Tinsley effectively debunked that, writing that "the programs may indeed consider a lot of moves and positions, but one thing is certain. They do not see much!'' (Tinsley 1980). Efforts to arrange a match between the two went nowhere, and the Duke program was quietly retired.

Interest in checkers was rekindled in 1989 with the advent of strong commercial programs and a research effort at the University of Alberta—CHINOOK. CHINOOK was authored principally by Jonathan Schaeffer, Norman Treloar, Robert Lake, Paul Lu, and Martin Bryant. In 1990, the program earned the right to challenge for the human world championship. The checkers federations refused to sanction the match, leading to the creation of a new title: The World Man-Machine Championship. This title was contested for the first time in 1992, with Marion Tinsley defeating CHINOOK in a 40-game match by a score of 4 wins to 2. CHINOOK's wins were the first against a reigning world champion in a nonexhibition event for any competitive game.

There was a rematch in 1994, but after six games (all draws), Tinsley resigned the match and the title to CHINOOK, citing health concerns. The following week he was diagnosed with cancer, and he died eight months later. CHINOOK has subsequently defended its title twice and has not lost a game since 1994. The program was retired from human competitions in 1997 (Schaeffer 1997).

The structure of CHINOOK is similar to that of a typical chess program: search, knowledge, database of opening moves, and endgame databases (Schaeffer 1997; Schaeffer et al. 1992). CHINOOK uses alpha-beta search with a myriad of enhancements, including iterative deepening, transposition table, move ordering, search extensions, and search reductions. CHINOOK was able to average a minimum of 19-ply searches against Tinsley (using 1994 hardware), with search extensions occasionally reaching 45 ply into the tree. The median position evaluated was typically 25-ply deep into the search.

A notable feature in CHINOOK is its use of endgame databases. The databases contain all checkers positions with 8 or fewer pieces, 444 billion (4×10^{11}) positions compressed into 6 gigabytes for real-time decompression. Unlike chess programs, which are compute bound, CHINOOK becomes input-output bound after a few moves in a game. The deep searches mean that the database is occasionally being hit on the first move of a game. The databases introduce accurate values (win/loss/draw) into the search (no error), reducing the program's dependency on its heuristic evaluation function (small error). In many games, the program is able to back up a draw score to the root of a search within 10 moves by each side from the start of a game, suggesting that it might be possible to determine the game-theoretic value of the starting position of the game (one definition of "solving" the game).

CHINOOK is the first program to win a human world championship for any game. At the time of CHINOOK's retirement, the gap between the program and the highest-rated human was 200 rating points (using the chess rating scale) (Schaeffer 1997). A gap this large means that

the program would score 75 percent of the possible points in a match against the human world champion. Since then, faster processor speeds mean that CHINOOK has become stronger, further widening the gap between man and machine.

Chess

The progress of computer chess was strongly influenced by an article by Ken Thompson that equated search depth with chess-program performance (Thompson 1982). Basically, the paper presented a formula for success: Build faster chess search engines. The milestones in chess program development become a statement of the state of the art in high-performance computing:

1978–1980: The pioneering programs from Northwestern University, most notably CHESS 4.6 (Slate and Atkin 1977), ran on a top-of-the-line Control Data computer and achieved the first major tournament successes.

1980–1982: BELLE, the first program to earn a U.S. master title, was a machine built to play chess. It consisted of 10 large wire-wrapped boards using LSI chips (Condon and Thompson 1982).

1983–1984: CRAY BLITZ used a multiprocessor Cray supercomputer (Hyatt, Gower, and Nelson 1990).

1985–1986: The HITECH chess machine was based on 64 special-purpose VLSI chips (one to a board square) (Ebeling 1987; Berliner and Ebeling 1989).

1985–1986: WAYCOOL used a 256-processor hypercube (Felten and Otton 1988).

1987–present: CHIPTEST (and its successors DEEP THOUGHT and DEEP BLUE) took VLSI technology even further to come up with a full-board chess chip (Hsu 1999; Hsu et al. 1990a, 1990b).

In 1987, CHIPTEST shocked the chess world by tying for first place in a strong tournament, finishing ahead of a former world champion and defeating a grand master. The unexpected success aroused the interest of world champion Garry Kasparov, who played a two-game exhibition match against the program in 1989. Man easily defeated machine in both games.

The DEEP BLUE team worked for seven years to improve the program, including designing a single-chip chess search engine and making significant strides in the quality of their software. In 1996, the chess machine played a six-game exhibition match against Kasparov. The world champion was stunned by a defeat in the first game, but he recovered to win the match, scoring three wins and two draws to offset the single loss. The following year, another exhibition match was played. DEEP BLUE scored a brilliant win in game two, handing Kasparov a psychological blow from which he never recovered. In the final, decisive game of the match, Kasparov fell into a trap, and the game ended quickly, giving DEEP BLUE an unexpected match victory, scoring two wins, three draws, and a loss.

It is important to keep this result in perspective. First, it was an exhibition match; DEEP BLUE did not earn the right to play Kasparov.[5] Second, the match was too short to accurately determine the better player; world-championship matches have varied from 16 to 48 games in length. Although it is not clear just how good DEEP BLUE is, there is no doubt that the program is a strong grand master.

What does the research community think of the DEEP BLUE result? Many are filled with admiration at this feat of engineering. Some are cautious about the significance. John McCarthy (1997) wrote that "in 1965, the Russian mathematician Alexander Kronrod said, 'Chess is the *Drosophila* of artificial intelligence.' However, computer chess has developed much as genetics might have if the geneticists had concentrated their efforts starting in 1910 on breeding racing *Drosophila*. We would have some science, but mainly we would have very fast fruit flies."[6]

In retrospect, the chess "problem" turned out to be much harder than was expected by the computing pioneers. The DEEP BLUE result is a tremendous achievement, and a milestone in the history of computing science.

From the scientific point of view, it is to be regretted that DEEP BLUE has been retired, the hardware unused, and the programming team disbanded. The scientific community has a single data point that suggests machine might be better than man at chess. The data are insufficient, and the sample size is not statistically significant. Moreover, given the lack of interest in DEEP BLUE from IBM, it is doubtful that this experiment will ever be repeated. Of what value is a single, nonrepeatable data point?

DEEP BLUE and its predecessors represent a decade-long intensive effort by a team of people. The project was funded by IBM, and the principal scientists who developed the program were Feng-Hsiung Hsu, Murray Campbell, and Joe Hoane.

The notable technological feature of DEEP BLUE is its amazing speed, the result of building special-purpose chess chips. The chip includes a search engine, a move generator, and an evaluation function (Cambell, Hoane, and Hsu 2001; Hsu 1999). The chip's search algorithm is based on alpha-beta. The evaluation function is implemented as small tables on the chip; the

values for these tables can be downloaded to the chip before the search begins. These tables are indexed by board features and the results summed in parallel to provide the positional score.

A single chip is capable of analyzing over two million chess positions a second (using 1997 technology). It is important to note that this speed understates the chip's capabilities. Some operations that are too expensive to implement in software can be done with little or no cost in hardware. For example, one capability of the chip is to selectively generate subsets of legal moves, such as all moves that can put the opponent in check. These increased capabilities give rise to new opportunities for the search algorithm and the evaluation function. Hsu (1999) estimates that each chess chip position evaluation roughly equates to 40,000 instructions on a general-purpose computer. If so, then each chip translates to a 100 billion instruction a second chess supercomputer.

Access to the chip is controlled by an alpha-beta search algorithm that resides on the host computer (an IBM SP-2). Each of the 30 SP-2 processors could access 16 chips. The reported cumulative performance, 200,000,000 positions analyzed a second, falls short of the peak speed (over 1 billion positions a second) because of the inherent difficulty of getting good parallel performance out of the alpha-beta algorithm. This massive amount of computing allows the program to search deeper, significantly reducing the probability that it will make an error (as Kasparov found out to his regret).

The AI community gave a collective sigh of relief when DEEP BLUE defeated Kasparov. It was time to move on to new challenges in the field.

Othello

The first major Othello program was Paul Rosenbloom's (1982) IAGO, achieving impressive results given its early-1980 hardware. It dominated play against other Othello programs of the time but played only two games against world-class human players, losing both. The program's ability to predict 59 percent of the moves played by human experts was extrapolated to conclude that the program's playing strength was of world-championship caliber.

By the end of the decade, IAGO had been eclipsed. Kai-Fu Lee and Sanjoy Mahajan's program BILL represented a major improvement in the quality of computer Othello play (Lee and Mahajan 1990). The program combined deep search with extensive knowledge (in the form of precomputed tables) in its evaluation function. Bayesian learning was used to combine the evaluation-function features in a weighted quadratic polynomial.

Statistical analysis of the program's play indicated that it was a strong Othello player. BILL won a single game against Brian Rose, the highest-rated American Othello player at the time. In test games against IAGO, BILL won every game. These results led Lee and Mahajan to conclude that "BILL is one of the best, if not the best, Othello player in the world.'' As usual, there is danger in extrapolating conclusions based on limited evidence.

With the advent of the Internet Othello server (IOS), computer Othello tournaments became frequent. In the 1990s, they were dominated by Michael Buro's LOGISTELLO. The program participated in 25 tournaments, finished first 18 times, second 6 times, and fourth once. The program combined deep search with an extensive evaluation function that was automatically tuned, which when combined with an extensive database of opening moves and a perfect end-game player, is a winning recipe for Othello.

Although it was suspected that by the mid-1990s, computers had surpassed humans in their playing abilities at Othello, this was not properly demonstrated until 1997, when LOGISTELLO played an exhibition match against world champion Takeshi Murakami. In preparation for the match, Buro (1997, p. 189) that

> BILL played a series of games against different versions of LOGISTELLO. The results showed that bill, when playing 5-minute games running on a PentiumPro/200 PC, is about as strong as a 3-ply LOGISTELLO, even though BILL searches 8 to 9 plies. Obviously, the additional search is compensated for by knowledge. However, the 3-ply LOGISTELLO can only be called mediocre by today's human standards.
>
> Two explanations for the overestimation of playing strength in the past come to mind: (1) during the last decade human players have improved their playing skills considerably, and (2) the playing strength of the early programs was largely overestimated by using ... nonreliable scientific methods.

LOGISTELLO won all six games against Murakami by a total disc count of 264 to 120 (Buro 1997), which confirmed what everyone had expected about the relative playing strengths of man and machine. The gap between the best human players and the best computer programs is believed to be large and effectively unsurmountable.

Outwardly, LOGISTELLO looks like a typical

alpha-beta–based searcher (Buro 2001). However, the construction of the evaluation function is novel. The program treats the game as having 13 phases: 13–16 discs on the board, 17–20 discs, …, and 61–64 discs.[7] Each phase has a different set of weights in the evaluation function. The evaluation-function features are patterns of squares comprising combinations of corners, diagonals, and rows. These patterns capture important Othello concepts, such as mobility, stability, and parity. LOGISTELLO has 11 such patterns, which with rotations and reflections yield 46. The patterns include a 3×3 and a 5×2 configuration of stones anchored in a corner and all diagonals of length greater than 3.

The weights for each entry in each pattern (46) for each phase of the game (11) are determined by linear regression. More than 1.5 million table entries need to be determined. The data were trained using 11 million scored positions obtained from self-play games and practice games against another program (Buro 1995). The evaluation function is completely table driven. Given a position, all 46 patterns are matched against the position, with a successful match returning the associated weight. These weights are summed to get the overall evaluation that approximates the final disc differential.

Michael Buro (1997, p. 193) comments on the reasons why LOGISTELLO easily won the Murakami match:

> When looking at the games of the match the main reasons for the clear outcome are as follows:
>
> 1. Lookahead search is very hard for humans in Othello. The disadvantage becomes very clear in the endgame phase, where the board changes are more substantial than in the opening and middlegame stage. Computers are playing perfectly in the endgame while humans often lose discs.
> 2. Due to the automated tuning of the evaluation functions and deep selective searches, the best programs estimate their winning chance in the opening and middlegame phase very accurately. This leaves little room for human innovations in the opening, especially because the best Othello programs are extending their opening books automatically to explore new variations.

Scrabble

The first documented Scrabble program appears to have been written by Stuart Shapiro and Howard Smith and was published in 1977 (Shapiro and Smith 1977). In the 1980s, a number of Scrabble programming efforts emerged, and by the end of the decade, it was apparent that these programs were strong players. With access to the entire Scrabble dictionary in memory (now over 100,000 words), the programs held an important advantage in any games against humans.

At the first Computer Olympiad in 1989, the Scrabble winner was CRAB written by Andrew Appel, Guy Jacobson, and Graeme Thomas (Leavy and Beal 1989). Second was TYLER written by Alan Frank. Subsequent Olympiads saw the emergence of TSP (Jim Homan), which edged out TYLER in the second and third Olympiads. All these programs were very good and quite possibly strong enough to be a serious test for the best players in the world.

Part of their success was a result of the fast, compact Scrabble move generator developed by Andrew Appel (Appel and Jacobson 1988). Steven Gordon (1994) subsequently developed a move generator that was twice as fast but used five times as much storage.

Brian Sheppard began working on a Scrabble program in 1983 and started developing MAVEN in 1986. In a tournament in December 1986, MAVEN scored eight wins and two losses over an elite field, finishing in second place on a tie breaker. Sheppard describes the games against humans at this tournament:[8]

> MAVEN reels off JOUNCES, JAUNTIER, and OVERTOIL on successive plays, each for exactly 86 points, to come from behind against future national champion Bob Felt. MAVEN crushed humans repeatedly in offhand games. The human race begins to contemplate the potential of computers.

In the following years, MAVEN continued to demonstrate its dominating play against human opposition. Unfortunately, because it did not compete in the Computer Olympiads, it was difficult to know how strong it was compared to other programs at the time.

In the 1990s, Sheppard developed a pre–endgame analyzer (when there were a few tiles left in the bag) and improved the program's ability to simulate likely sequences of moves. These represented important advances in the program's ability. It was not until 1997, however, that the opportunity arose to properly assess the program's abilities against world-class players. In 1997, a two-game match between MAVEN and Adam Logan at AAAI-97, one of the best players in North America, ended in two wins for the human. Unfortunately, the match was not long enough to get a sense of who was really the best player.

In March 1998, the *New York Times* sponsored

an exhibition match between MAVEN and a team consisting of world champion Joel Sherman and runner-up Matt Graham. It is not clear whether the collaboration helped or hindered the human side, but the computer won convincingly by a score of six wins to three. The result was not an anomaly. In July 1998, MAVEN played another exhibition match against Adam Logan (at AAAI-98), scoring nine wins to five.

Shortly after the Logan match, Brian Sheppard wrote:[9]

> The evidence right now is that MAVEN is far stronger than human players.... I have outright claimed in communication with the cream of humanity that MAVEN should be moved from the "championship caliber'' class to the "abandon hope'' class, and challenged anyone who disagrees with me to come out and play. No takers so far, but maybe one brave human will yet venture forth.
>
> No one has.

MAVEN divides the game into three phases (Sheppard 2001): (1) early game, (2) pre–end game, and (3) end game. The early game starts at move one and continues until there are 9 or fewer tiles left in the bag (that is, with the opponent's 7 tiles, there are 16 or fewer unknown tiles). In the pre–end-game and end-game phases, specialized searches are performed, taking advantage of the limited amount of unknown information.

In the early game phase, the program uses simulations to get a statistical analysis of the likely consequences of making a move. Typically, 1,000 three-ply simulations are done when making a move decision. The move leading to the highest average point differential is selected. The issue with the simulations is move generation. On average, there are over 700 legal moves to a position, and the presence of 2 blanks in the rack can increase this figure to more than 5000![10] Contrast this number, for example, with chess, where the average number of moves to consider in a position is roughly 40. Thus, MAVEN needs to pare the list of possible moves down to a small list of likely moves. Omitting an important move from this list will have serious consequences; it will never be played. Consequently, MAVEN uses multiple move generators, each identifying moves that have important features that merit consideration. These move generators are as follows:

Score and rack: This generator finds moves that result in a high score and a good rack (tiles remaining in your possession). Strong players evaluate their rack based on the likeliness of the letters being used to aid upcoming words. For example, playing a word that leaves a rack of QXI would be less preferable than leaving QUI; the latter offers more potential for playing the Q effectively.

Bingo blocking: Playing all 7 letters in a single turn leads to a bonus of 50 points (a bingo). This move generator finds moves that reduce the chances of the opponent scoring a bingo on his/her next turn. Sometimes it is worth sacrificing points to reduce the opponent's chances of scoring big.

Immediate scoring: This generator finds the moves with the maximum number of points (which becomes more important as the end of the game nears).

Each routine provides as many as 10 candidate moves. Merging these lists results in typically 20 to 30 unique candidate moves to consider. In the early part of the game, only the score-and-rack generator is used. In the pre–end-game, there are four: the three listed above plus a pre–end-game evaluator that "took years to tune to the point where it didn't blunder nearly always."[11] In the end game, all possible moves are considered.

The move-generation routines are highly effective at filtering the hundreds or thousands of possible moves:[12]

> It is important to note that simply selecting the one move preferred by the score-and-rack evaluator plays championship-caliber Scrabble. My practice of combining 10 moves from multiple generators is evidence of developing paranoia on my part. "Massive overkill" is the centerpiece of maven's design philosophy.

Obviously, this move filtering works very well, given the level of the program's play. The Scrabble community has extensively analyzed MAVEN's play and found a few minor errors in the program's play. Postmortem analysis of the Logan match showed that MAVEN made mistakes that averaged nine points a game. Logan's average was 40 points a game. MAVEN missed seven fishing moves—opportunities to exchange some tiles (69 points lost), some programming errors (48 points lost), and several smaller mistakes (6 points lost). The programming errors have been corrected. If a future version of MAVEN included fishing, the error rate would drop to less than one point to a game. MAVEN would be playing nearly perfect Scrabble.

Of the points lost because of programming errors, Brian Sheppard [13]

> It just drives me crazy that I can think up inventive ways to get computers to act intelligently, but I am not smart enough to implement them correctly.
>
> The soliloquy of every games programmer!

So, You Think You Are Good at Scrabble?

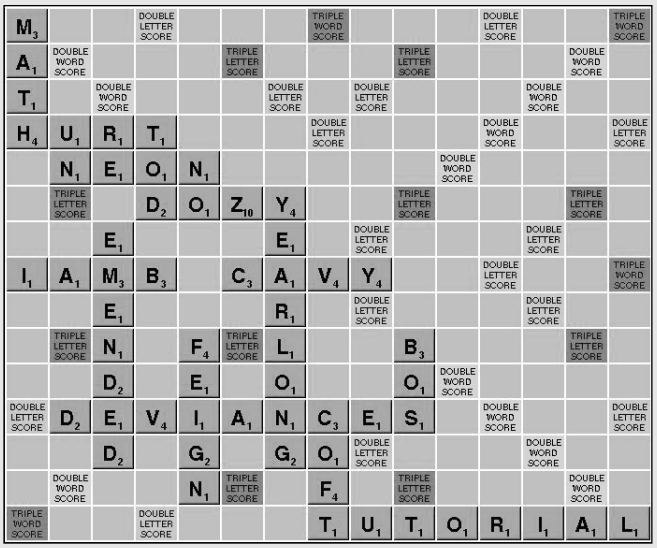

Figure A. MAVEN *Plays BOS (j10), Scoring 26 Points.*

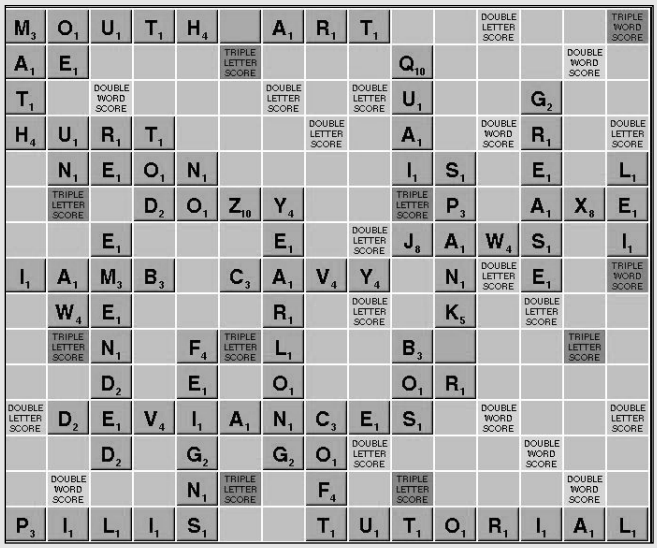

Figure B. MAVEN*—Logan, Final Position.*

At AAAI-98, MAVEN played an exhibition match against Adam Logan, one of the top Scrabble players in North America. Logan won three of the first four games of the match, but MAVEN won six of the next seven. Going into the critical twelfth game, MAVEN led by a score of seven wins to four. The following annotations are based on comments from Brian Sheppard. The columns of a Scrabble board are specified from left to right by the letters a to o. Rows are specified from top to bottom using the numbers 1 to 15. Moves are specified by giving the square of the first letter of the word. If the coordinate begins with a number, then the word is placed horizontally. If the coordinate begins with a letter, then the word is placed vertically. The blank is referred to by "?."

Follow along yourself. How good are the moves that you find?

MAVEN (ACNTVYZ) plays CAVY at 8f, 24 pts, MAVEN = 24 Logan = 0. The alternative is ZANY, scoring 32 points, but leaving a poor selection of letters in the rack.

Logan (EGLNORY) plays YEARLONG at g6, 66 pts, MAVEN = 24 Logan = 66. The only bingo! A 50 point bonus.

MAVEN (ADNNOTZ) plays DOZY at 6d, 37 pts, MAVEN = 61 Logan = 66. AZLON (10e,34,NTD) or ZOON (11e,26,ADNT) can also be considered.

Logan (ADEFOTV) plays OFT at h13, 21 pts, MAVEN = 61 Logan = 87. Of course, you also considered VOTED (5A,27,AF), OVA (H13,21,DEFT), FOVEAL (10b,22,DT), and ADVENT (12c,22,FO).

MAVEN (AENNNOT) plays NEON at 5b, 15 pts, MAVEN = 76 Logan = 87.

Logan (ACDEEIV) plays DEVIANCE at 12b, 96 pts, MAVEN = 76 Logan = 183. Another bingo!

MAVEN (AHINRTU) plays HURT at 4a, 34 pts, MAVEN = 110 Logan = 183.

Logan (DDEEMMN) plays EMENDED at c7, 26 pts, MAVEN = 110 Logan = 209.

MAVEN (ABEINNP) plays IAMB at 8a, 33 pts, MAVEN = 143 Logan = 209.

Logan (AILMTTU) plays MATH at a1, 27 pts, MAVEN = 143 Logan = 236. Strong players also consider UTA (3a,20,ILMT), which scores fewer points but gets rid of the annoying "U."

MAVEN (EFGNNPS) plays FEIGN at e10, 18 pts, MAVEN = 161 Logan = 236. FENS (j9,24,GNP) scores more points, but FEIGN keeps better tiles.

Logan (AILORTU) plays TUTORIAL at 15h, 77 pts, MAVEN = 161 Logan = 313. Adam Logan's third bingo!

MAVEN (?ABNOPS) plays BOS at j10, 26 pts, MAVEN = 187 Logan = 313. See figure A. Sheppard considers this to be a "fantastic move" and one of the most difficult moves in the game.

Logan (IILPRSU) plays PILIS at 15a, 34 pts, MAVEN = 187 Logan = 347. PILIS, PULIS, PILUS, and PURIS are all good.

MAVEN (?AKNPRS) plays SPANKER at k5, 105 pts, MAVEN = 292 Logan = 347. The only bingo, reviving Maven's chances despite the 160-point deficit.

Logan (EEEORUS) plays OE at b1, 12 pts, MAVEN = 292 Logan = 359. The best move, dumping extra vowels.

MAVEN (?HJTTWW) plays JAW at 7j, 13 pts, MAVEN = 305 Logan = 359.

Logan (AEEGRSU) plays GREASE at m3, 31 pts, MAVEN = 305 Logan = 390. AGER (L9,24,ESU) also merits consideration.

MAVEN (?HRTTWX) plays AX at 6m, 25 pts, MAVEN = 330 Logan = 390. Maven's second brilliant move, choosing AX over GOX (13G,36) and sacrificing 11 points.

Logan (EIIILQU) plays LEI at o5, 13 pts, MAVEN = 330 Logan = 403.

MAVEN (?AHRTTW) plays WE at 9b, 10 pts, MAVEN = 340 Logan = 390.

Logan (AIIIOQU) plays QUAI at j2, 35 pts, MAVEN = 340 Logan = 438. A 98-point lead and only a few moves are left in the game. Obviously, it's all over.

MAVEN (?AHRTTU) plays MOUTHPART at 1a, 92 + 8 pts, MAVEN = 440 Logan = 438. See figure B. Wonderful! MAVEN scores exactly 100 points, edging Adam Logan by 2. Sheppard writes that "MAVEN steals the game on the last move. Adam, of course, was stunned, as it seemed that there were no places for bingos left on this board. If I hadn't felt so bad for Adam, who played magnificently, I would have jumped and cheered" (Brian Sheppard, personal communication, 1999). This game put MAVEN up by eight games to four, so winning the match was no longer in doubt.

How often do you score 438 points in a game of Scrabble…and lose?

Just in case some of the words used in this game are not part of your everyday vocabulary, here are a few useful definitions (taken from the commercial version of MAVEN):

Bos: a pal.

Fens: marshes.

Foveal: a shallow anatomical depression.

Gox: gaseous oxygen.

Pilis: a Philippine tree.

Uta: a type of lizard.

Zoon: whole product of one fertilized egg.

Other Games

Superhuman performance has been achieved in several lesser-known games. For example, for both the acient African game of Awari and the recently invented Lines of Action, there seems little doubt that computers are significantly stronger than all human players. In Awari, databases containing all positions with 38 or fewer stones on the board have been constructed (the game starts with 48 stones), suggesting that a perfect computer player will soon be available. In Lines of Action, the program MONA won the world mail-play championship in competition against most of the top human players. The gap between the top programs and the best humans appears to be large and growing.

For some games, computers have been able to determine the result of perfect play and a sequence of moves to achieve this result (van den Herik, Uiterwijk, and van Rijswijck 2001).[14] In these games, the computer can play perfectly, in the sense that the program will never make a move that fails to achieve the best-possible result. Solved games include Nine Men's Morris (Gasser 1995), Connect-4 (Allis 1988), Qubic (Allis 1994), Go Moku (Allis 1994), and 8 × 8 Domineering (Breuker, Uiterwijk, and van den Herkik 2000).

This article has not addressed one-player games (or puzzles). Single-agent search (A*) has successfully been used to optimally solve instances of the 24-puzzle (Korf 2000; Korf and Felner 2001) and Rubik's Cube (Korf 1997).

Current Research Efforts

In the past decade, a number of games have become popular research test beds. These games are resistant to alpha-beta search, either because of the large branching factor in the search tree or the presence of unknown information. In many respects, the research being done for these games has the potential to be much more widely applicable than the work done on the alpha-beta search-based programs.

Bridge

Work on computer bridge began in the early 1960s (Berlekamp [1963], for example), but it wasn't until the 1980s that major efforts were undertaken. The advent of the personal computer spurred numerous commercial projects that resulted in programs with relatively poor capabilities. Perennial world champion Bob Hamman once remarked that the commercial programs "would have to improve to be hopeless'' (Ginsberg 1999). A similar opinion was shared by another frequent world champion, Zia Mahmood. In 1990, he offered a prize of £1,000,000 to the person who developed a program that could defeat him at bridge. At the time, this bet seemed safe for the foreseeable future.

In the 1990s, several academic efforts began using bridge for research in AI (Frank 1998; Ginsberg 1999; Smith, Nau, and Throop 1998a, 1998b; Ginsberg 1996b). The commercial BRIDGE BARON program teamed up with Dana Nau and Steve Smith from the University of Maryland. The result was a victory in the 1997 World Computer Bridge Championship. The program used a hierarchical task network for the play of the hand. Rather than build a search tree where each branch was the play of a card, they would define each branch as a strategy, using human-defined concepts such as finesse and squeeze (Smith, Nau, and Throop 1998a, 1998b). The result was an incremental improvement in the program's card play, but it was still far from being world-class caliber.

Beginning in 1998, Matthew Ginsberg's program GIB started dominating the computer bridge competition, handily winning the World Computer Bridge Championship. The program started producing strong results in competitions against humans, including an impressive result in an exhibition match against world champions Zia Mahmood and Michael Rosenberg (held at AAAI-98). The match lasted two hours, allowing 14 boards to be played. The result was in doubt until the last hand, before the humans prevailed by 6.31 international match points (IMPs). This result was the first notable man-machine success for computer bridge-playing programs. Zia Mahmood, impressed by the rapid progress made by GIB, withdrew his million pound prize.

GIB was invited to compete in the Par Contest at the 1998 World Bridge Championships. This tournament tests the contestant's skills at playing out bridge hands. In a select field of 35 of the premier players in the world, the program finished strongly in twelfth place. Michael Rosenberg won the event with a score of 16,850 out of 24,000; GIB scored 11,210. Of the points lost by GIB, 1,000 were because of time (there was a 10-point penalty for each minute spent thinking), 6,000 were because GIB did not understand the bidding, and 6,000 were because GIB was unable to handle some hands where the correct strategy involves combining different possibilities (Ginsberg 1999).

The name GIB originally stood for "Goren in a box," a tribute to one of the pioneers of bridge. Another interpretation is "Ginsberg's Intelligent Bridge.'' Previous versions of GIB used a fast search to play out a hand. It simu-

lated roughly 50 different scenarios for the placement of the opponent's cards and chose the play that maximized the expected score (Ginsberg 1999). For the play of the hand, Ginsberg has developed a new version of the algorithm that eliminates the simulations and replaces it with perfect information (Ginsberg 2001).

A challenging component of the game is the bidding. Most previous attempts at bridge bidding have been based on an expert-defined set of rules. This is largely unavoidable because bidding is an agreed-on convention for communicating card information. GIB takes this bidding one step further, building on the ability to quickly simulate a hand (Ginsberg 1999). The program has access to a large database of bidding rules (7,400 rules from the commercial program MEADOWLARK BRIDGE). At each point in the bidding, GIB queries the database to find the set of plausible bids. For each bid, the rest of the auction is projected using the database, and then the play of the resulting contract is simulated. GIB chooses the bid that leads to the average best result for the program.

Although intuitively appealing, this approach does have some problems. Notably, the database of rules might have gaps and errors in it. Consider a rule where the response to the bid 4♠ is incorrect in the database. GIB will direct its play toward this bid because it assumes the opponents will make the (likely bad) database response. As Ginsberg writes, "It is difficult to distinguish a good choice that is successful because the opponent has no winning options from a bad choice that appears successful because the heuristic fails to identify such options'' (Ginsberg 1999, p. 588).

GIB uses three partial solutions to the problem of an erroneous or incomplete bidding system. First, the bidding database can be examined by doing extensive offline computations to identify erroneous or missing bid information. This is effective but can take a long time to complete. Second, during a game, simulation results can be used to identify when a database response to a bid leads to a poor result. This may be evidence of a database problem, but it could also be the result of effective disruptive bidding by GIB. Finally, GIB can be biased to make bids that are "close" to the suggested database bids, allowing the program the flexibility to deviate from the database.

To summarize, GIB is well on the way to becoming a world-class bridge player. The program's card play is already at a world-class level (as evidenced by the Par Contest result), and current efforts will only enhance the program's abilities. The bidding needs improvement, an effort that is currently being addressed. Had Zia Mahmood not withdrawn his offer, he might eventually have lost his money.

Go

The history of computer go has not been dominated by hardware advances, as seen in computer chess. Computer go tournaments proliferated in the 1990s, and the organizers had the benefit of the chess experience. Two tournament rules were instituted that had a significant impact on how program development would occur. The first required all competitors to run on a commercially available single-processor machine, which had the advantage of putting all the programs on a level playing field by factoring out most hardware differences. The second rule required that an entire game had to be completed in 30 minutes for each player. Because games could be as long as 180 moves a side, programmers were faced with critical cost-benefit decisions in their implementations. The rules had the advantages of making tournaments easy to organize (no expensive hardware setup or modem connections needed) and ensuring that competitions could be completed quickly with lots of games being played.

The first go program was written by Al Zobrist in 1970 (Zobrist 1970). Walter Reitman and Bruce Wilcox began researching go programs in 1972 (Reitman et al. 1974), an effort that has continued for Wilcox to the current day. These early efforts produced weak programs; there was no obvious single algorithm to build a program around, as alpha-beta had done for chess. The difficulty in writing a go program became evident; a strong program would need lots of patterns and knowledge, with only a limited dependence on search.

Computer go tournaments began in 1984 with a short-lived series of annual tournaments at the USENIX Conference. In 1987, the First International Go Congress was held, and there have been annual events ever since. The mid-1990s were dominated by the program HANDTALK, written by Zhixing Chen. HANDTALK remained stagnant for a few years while it was being rewritten. During this time, Michael Reiss' GO4++ assumed front-runner status. Chen's new program, GOEMATE, now appears to be the best. Although the top programs claim a performance level of as many as 3 kyu on the go rating scale (a middle amateur level), most experts believe that the programs are much weaker than that (around 8 kyu).

The Ing Prize has been set up as an incentive to build strong go programs. The grand prize of roughly $1.5 million will be won by the devel-

opers of the first program to beat a strong human player on a 19 × 19 board. To qualify to play for the grand prize, a program must win a number of matches of increasing difficulty. Currently, the programs have to defeat three junior players (ages 11 to 13). Don't let their age fool you; they are very strong players! The winner of the annual International Go Congress gets the chance to play. To qualify for this event, a program must finish in the top three in one of the North American, European, or Asian championships.

Go has been resistant to the techniques that have been successfully applied to the games discussed in this article. For example, because of the 19 × 19 board and the resulting large branching factor, alpha-beta search alone has no hope of producing strong play. Instead, the programs perform small, local searches that use extensive application-dependent knowledge. David Fotland, the author of the MANY FACES OF GO program, identifies over 50 major components needed by a strong go-playing program. The components are substantially different from each other, few are easy to implement, and all are critical to achieving strong play. In effect, you have a linked chain, where the weakest link determines the overall strength.

Martin Müller (author of EXPLORER) gives a stark assessment of the reality of the current situation in developing go programs (Müller 1999, pp. 105–106):

> Given the complexity of the task, the supporting infrastructure for writing go programs should offer more than is offered for other games such as chess. However, the available material (publications and source code) is far inferior. The playing level of publicly available source code…, though improved recently, lags behind that of the state-of-the-art programs. Quality publications are scarce and hard to track down. Few of the top programmers have an interest in publishing their methods. Whereas articles on computer chess or general game-tree search methods regularly appear in mainstream AI journals, technical publications on computer go remain confined to hard to find proceedings of specialized conferences. The most interesting developments can be learned only by direct communication with the programmers and never get published.

Although progress has been steady, it will take many decades of research and development before world-championship–caliber go programs exist (Mueller 2001).

Poker

There are many popular poker variants. Texas Hold'em is generally acknowledged to be the most strategically complex variant of poker that is widely played. It is the premier event at the annual World Series of Poker.[15] Until recently, poker has been largely ignored by the computing academic community. There are two main approaches to poker research (Billings 1995): One approach is to use simplified variants that are easier to analyze. However, one must be careful that the simplification does not remove challenging components of the problem. For example, Findler (1977) worked on and off for 20 years on a poker-playing program for 5-card–draw poker. His approach was to model human cognitive processes and build a program that could learn, ignoring many of the interesting complexities of the game.

The other approach is to pick a real variant and investigate it using mathematical analysis, simulation, and ad hoc expert experience. Expert players with a penchant for mathematics are usually involved in this approach. None of this work has led to the development of strong poker-playing programs.

There is one event in the meager history of computer poker that stands out. In 1984 Mike Caro, a professional poker player, wrote a program that he called ORAC (Caro spelled backwards). It played one-on-one, no-limit Texas Hold'em. Few technical details are known about ORAC other than it was programmed on an Apple II computer in Pascal. However, Caro arranged a few exhibitions of the program against strong players:[16]

> It lost the TV match to casino owner Bob Stupak, but arguably played the superior game. The machine froze on one game of the two-out-of-three set when it had moved all-in and been called with its three of a kind against Stupak's top two pair. Under the rules, the hand had to be replayed. In the [world series of poker] matches, it won one (from twice world champion Doyle Brunson—or at least it had a two-to-one chip lead after an hour and a quarter when the match was canceled for a press conference) and lost two (one each to Brunson and then-reigning world champion Tom McEvoy), but—again—was fairly unlucky. In private, preparatory exhibition matches against top players, it won many more times than it lost. It had even beaten me most of the time.

Unfortunately, ORAC was never properly doc-

umented and the results never reproduced. It is highly unlikely that ORAC was as good as this small sample suggests. No scientific analysis was done to see whether the results were the result of skill or luck. As further evidence, none of the current-day commercial efforts can claim to be anything but intermediate-level players.

In the 1990s, the creation of an internet relay chat (IRC) poker server gave the opportunity for humans (and computers) to play interactive games over the internet. A number of hobbyists developed programs to play on IRC. Foremost among them is R00LBOT, developed by Greg Wohletz. The program's strength comes from using expert knowledge at the beginning of the game and doing simulations for subsequent betting decisions.

The University of Alberta program POKI, authored by Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron, is the first serious academic effort to build a strong poker-playing program. POKI plays on the IRC poker server and, like R00LBOT, is a consistent big winner. Unfortunately, because these games are played with fictitious money, it is hard to extrapolate these results to casino poker.

To play poker well, a program needs to be able to assess hand strength (chances that you have the current best hand), assess hand potential (chances that additional cards will improve your hand), model the opponents (exploiting tendencies in their play), handle deception (misleading information given by the opponents), and bluff (deceive the opponents). In strategic games such as chess, the performance loss by ignoring opponent modeling is small; hence, it is usually ignored. In contrast, not only does opponent modeling have tremendous value in poker, it can be the distinguishing feature between players at different skill levels. If a set of players all have a comparable knowledge of poker fundamentals, the ability to alter decisions based on an accurate model of the opponent can have a greater impact on success than any other strategic principle.[17]

To assess a hand, POKI compares its cards against all possible opponent holdings. Naively, one could treat all opponent hands as equally likely; however, this skews the hand evaluations compared to more realistic assumptions. Many weak hands are likely to have been folded early on in the game. Therefore, for each possible opponent hand, a probability (or weight) is computed that indicates the likelihood that the opponent would have played the hand in the observed manner.

The simplest approach to determining these weights is to treat all opponents the same, calculating a single set of weights to reflect reasonable behavior, and use them for all opponents. An offline simulation was used to compute the expected value for each possible hand; these results closely approximate the ranking of hands by strong players. This is called generic opponent modeling (GOM) (Billings et al. 1998). Although rather simplistic, this model is quite powerful in that it does a good job of skewing the hand evaluations to take into account the most likely opponent holdings.

Obviously, treating all opponents the same is clearly wrong; each player has a different style. Specific opponent modeling (SOM) customizes the calculations to include opponent-specific information. The probability of an opponent holding a particular hand is adjusted by feeding into a neural net the betting frequency statistics gathered on this opponent from previous hands. These statistics usually provide enough information to differentiate, for example, aggressive playing styles from conservative ones.

In competitive poker, opponent modeling is much more complex than portrayed here. For example, players can act to mislead their opponents into constructing an erroneous model. Early in a session, a strong poker player might try to create the impression of being very conservative, only to exploit this image later in the session when the opponents are using an incorrect opponent model. A strong player has to have a model of each opponent that can quickly adapt to changing playing styles.

At best, POKI plays at the strong intermediate level (Billings et al. 2001). A considerable gap remains to be overcome before computers will be as good as the best human players. Recent research has focused on trying to build "optimal" playing strategies (Koller and Pfeffer 1997).

Other Games

Several less well-known games are providing interesting challenges. The following three examples all have one property in common: a large branching factor.

Shogi, often referred to as Japanese chess, is very popular in Japan, with major tournaments each year culminating in a computer world championship. From the search point of view, Shogi is more challenging than chess: 9×9 board (versus 8×8 for chess), 40 pieces (32 for chess), 8 piece types (6), 80 to 120 average branching factor (40), and ability of the captured pieces to be returned to the board (removed from the board). Checkmating attacks are critical in Shogi; the programs need specialized checkmate solvers. These solvers

have had some spectacular successes. For example, programs are now capable of solving composed problems with a solution length of over 1500 ply! Nevertheless, the best programs play at the master's level, but world-championship–level play is still a few decades away (Iida, Sakuta, and Rollason 2001).

Hex is an elegant game with a simple rule set: alternate placing a stone of your color on an empty square. One player tries to create a chain of stones connecting the top to the bottom of the board. The other player tries to connect the left side to the right side. It can be shown mathematically that the game is a first-player win and that draws are not possible. QUEENBEE was the first program to achieve success against strong programs (van Rijswijk 2000). The program uses alpha-beta search with a novel evaluation function. Hexy is currently the strongest program in the world and is competitive with strong human players for smaller board sizes. The program uses a specialized search for virtual connections, utilizing a theorem-prover-like technique to prove that two points not connected can be connected by a series of moves (Anshelevich 2001, 2000).

A recently invented game that has become popular for games researchers is Amazons. It is played on a 10 × 10 board, with each player having four queens. Pieces move like a queen in chess, but after moving, they shoot an arrow in any direction. The square on which the arrow lands now becomes a wall and cannot be occupied by a queen. In effect, each move reduces the playing area available. If you run out of moves, you lose. In the opening phase of the game, there can be several thousand moves to choose from. The best programs typically search five ply ahead (deeper in the end game). Because of the territorial nature of the game, Amazons is often touted as a research stepping stone between the search-intensive approaches used in chess and the knowledge-intensive approaches used in go. AI research into this game is only three years old. The best programs play reasonably well but are not yet competitive with strong human players (van den Herik 2000).

Interesting research is also being done on puzzles. Recently, major advances have occurred in building programs that can solve crossword puzzles. PROVERB (Michael Littman, Greg Keim, and colleagues) scores remarkably well (over 95 percent of the words correct) on the *New York Times* crossword puzzles without understanding the clues (Keim et al. 1999; Keim, and Shazeer 2001)!

Another challenging puzzle is SOKOBAN. Here the large branching factor (could be over 100), and deep solution lengths (some optimal solutions are over 700 moves) make for a daunting search. On a standard test set, the program ROLLING STONE can only solve 57 of 90 problems (Junghanns and Schaeffer 2001, 1999).

The Future of Computer Games

In the realm of board and card games, go will continue to taunt AI researchers for many decades to come. As well, new games will come along to provide interesting challenges. For example, the game of Octi was invented to be resistant to computer algorithms.[18] It is characterized by having a large branching factor, making deep search impractical. However, Octi has the additional dimension that a move can change the capabilities of a piece, making it challenging to design an evaluation function.

The research into board and card games is, in some sense, historically motivated because these challenges were interesting at the dawn of the computing age. However, with the advent of home computers, new forms of computer games and a $20 billion (and growing) industry has emerged: interactive computer games. There are numerous products on the market covering the gamut of action games (for example. shoot'em-up games such as *Quake*), role-playing games (for example, player goes on a quest, as in *Baldur's Gate*), adventure games (for example, navigating through a scripted story, as in *King's Quest*), strategy games (for example, controlling armies in a war, such as in *Command and Conquer*), "God" games (for example, evolving a simulated population, as in *SimCity*), and sports (for example, controlling a player or coaching a team, such as *FIFA'01*) (Laird and van Lent 2000). Historically, these games have been long on graphics and short on AI.[19]

John Laird has promoted interactive computer games as an opportunity for the AI research community (Laird and van Lent 2000). Many interactive computer games require computer characters that need to interact with the user in a realistic, believable manner. Computer games are the ideal application for developing human-level AI. There is already a need for it because human game players are generally dissatisfied with computer characters. The characters are shallow, too easy to predict, and, all too often, exhibit artificial stupidity rather than artificial intelligence. This has led to the success of online games (such as *Ultima Online*), where players compete against other humans. The current state of the art in developing realistic characters can be described as primitive, with simple

rule-based systems and finite-state machines the norm. The lack of sophistication is the result of the lack of research effort (and, cause and effect, research dollars). This is changing because more game companies and researchers recognize that AI will play an increasingly important role in game design and development. The quality of the computer graphics might draw you to a product, but the play of the game will keep you using the product (and buying the sequel). AI is critical to creating a satisfying gaming experience.

Finally, the last few years have seen research on team games become popular. The annual RoboCup competition encourages hardware builders and software designers to test their skills on the soccer field.[20]

Although this article has emphasized building games programs that can compete with humans, there are many other AI challenges that can use games as an interesting experimental test bed. Some sample projects include data mining, learning, and annotators.

Data mining: There are large databases of end-game positions for chess, checkers, and awari. It is dissatisfying that all a program can do is look up a specific position in the database. If the exact position is in the database, you get useful information, else nothing. Surely there must be some way of mining the data to learn the principles of strong end-game play. As well, there are large databases of chess opening moves. Can these databases be analyzed to discover new opening ideas? Can one characterize an opponent's strengths and weaknesses? Can the data be extrapolated to similar positions?

Learning: Using temporal-difference learning to tune an evaluation function is just the precursor to other exciting applications of learning technology to games. For example, research in applying learning algorithms can result in more focused and informed game-tree searches, better opponent modeling in poker, and adaptive characters in commercial games.

Annotators: Developing annotators that can provide an interesting and informative analysis of a game is a challenging problem. There have been some attempts at automating the commentary for chess games (the International Computer Chess Association has an annual competition), but the results are mediocre. It is hard to differentiate between the trivial and the interesting, the verbose and the informative, all the while anticipating the questions humans would like answered in the commentary. An interesting example is the work done on providing computer commentary to RoboCup games (Frank et al. 2001).

Games will continue to be an interesting domain for exploring new ideas in AI.

Conclusions

Shannon, Turing, Samuel, Newell, and Simon's early writings were pioneering, realizing that computer games could be a rich domain for exploring the boundaries of computer science and AI. Software and hardware advances have led to significant success in building high-performance game-playing programs, resulting in milestones in the history of computing. With it has come a change in people's attitudes. Whereas in the 1950s and 1960s, understanding how to build strong game-playing programs was at the forefront of AI research, today it has been demoted to lesser status. In part, this is an acknowledgment of the success achieved in this field—no other area of AI research can claim such an impressive track record of producing high-quality working systems. However, it is also a reflection on the nature of AI itself. It seems that as the solution to problems become understood, the techniques become less "AIish."

The work on computer games has resulted in advances in numerous areas of computing. One could argue that the series of computer-chess tournaments that began in 1970 and continue to this day represents the longest running experiment in computing science history. Research using games has demonstrated the benefits of brute-force search, something that has become a widely accepted tool for a number of search-based applications. Many of the ideas that saw the light of day in game-tree search have been applied to other algorithms. Building world-championship–caliber games programs has demonstrated the cost of constructing high-performance AI systems. Games have been used as experimental test beds for many areas of AI. And so on.

One could argue that the series of computer-chess tournaments that began in 1970 and continue to this day represents the longest running experiment in computing science history.

Arthur Samuel's concluding remarks from his 1960 paper are as relevant today as they were when he wrote the paper (Samuel 1960):

> Programming computers to play games is but one stage in the development of an understanding of the methods which must be employed for the machine simulation of intellectual behavior. As we progress in this understanding it seems reasonable to assume that these newer techniques will be applied to real-life situations with increasing frequency, and the effort devoted to games … will decrease. Perhaps we have not yet reached this turning point, and we may still have much to learn from the study of games.

Acknowledgments

I would like to extend my deepest admiration to the brave human champions who accepted the challenge of a computer opponent. In most cases, the champion had little to gain but everything to lose. Malcolm Davis, Garry Kasparov, Adam Logan, Zia Mahmood, Marion Tinsley, Michael Rosenberg, and Takeshi Murakami made it possible to scientifically measure the progress of game-playing programs.

The initial impetus for this article came almost two years ago when Marvin Zelkowitz suggested I write an article for *Advances in Computers 50,* reflecting back on the 40 years since Arthur Samuel wrote an article on computer games in volume 1 of this series. This article was eventually worked into a talk that was presented at AAAI-00. I want to thank David Leake for encouraging me to write this article.

Financial support was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Alberta's Informatics Circle of Research Excellence (CORE).[21]

Notes

1. Othello is a registered trademark of Tsukuda Original, licensed by Anjar Co.

2. Scrabble is a registered trademark of the Milton Bradley Company, a division of Hasbro, Inc.

3. See www.msoworld.com.

4. E-mail message from G. Tesauro, 14 August 1998.

5. To be fair, it is unlikely that the International Chess Federation will ever allow computers to compete for the world championship.

6. The *drosophila* is the fruit fly. The analogy is that the fruit fly is to genetics research as games are to AI research.

7. Note that there is no need for a phase for less than 13 discs on the board because the search from the first move easily reaches 13 or more discs.

8. E-mail message from B. Sheppard, 9 March 1999.

9. Personal communication with B. Sheppard, 1999.

10. As a frequent Scrabble player, I painfully admit that the number of words that I find is considerably smaller than this!

11. E-mail message from B. Sheppard, 1 June 1999.

12. E-mail message from B. Sheppard, 1 June 1999.

13. Personal communication with B. Sheppard, 1999.

14. This is in contrast to the game of Hex, where it is easy to prove the game to be a first-player win, but computers are not yet able to demonstrate this win.

15. The 2000 winner of this event was Chris Ferguson, whose research career began in AI (he has published with Richard Korf [Powley, Ferguson, and Korf 1993]).

16. E-mail message from M. Caro, 13 March 1999.

17. The importance of opponent modeling can be seen in the First and Second International RoShamBo (rock, paper, scissors) competitions (www.cs.ualberta.ca/~games).

18. www.octi.net.

19. For example, path finding is a critical component of many games, yet it took until 1996 for the industry to "discover" A*.

20. www.robocup.com.

21. Portions of this article were published in Schaeffer (2000) and are reproduced with permission.

References

Allis, V. 1994. Searching for Solutions in Games and Artificial Intelligence. Ph.D. dissertation, Department of Computer Science, University of Limburg.

Allis, V. 1980. A Knowledge-Based Approach to Connect-Four. The Game Is Solved: White Wins. M.Sc. thesis, Department of Computer Science, Vrije Universiteit.

Anshelevich, V. 2001. A Hierarchical Approach to Computer Hex. *Artificial Intelligence.* Forthcoming.

Anshelevich, V. 2000. The Game of Hex: An Automatic Theorem-Proving Approach to Game Programming. In Proceedings of the Seventeenth National Conference on Artificial Intelligence,189–194. Menlo Park, Calif.: American Association for Artificial Intelligence.

Appel, A., and Jacobson, G. 1980. The World's Fastest Scrabble Program. *Communications of the ACM* 31(5): 572–578, 585.

Berlekamp, E. 1963. A Program for Playing Double-Dummy Bridge Problems. *Journal of the ACM* 10(4): 357–364.

Berliner, H. 1980a. Backgammon Computer Program Beats World Champion. *Artificial Intelligence* 14(2): 205–220.

Berliner, H. 1980b. Computer Backgammon. *Scientific American* 242(6): 64–72.

Berliner, H., and Ebeling, C. 1989. Pattern Knowledge and Search: The supreme Architecture. *Artificial Intelligence* 38(2): 161–198.

Billings, D. 1995. Computer Poker. M.Sc. thesis, Department of Computing Science, University of Alberta.

Billings, D.; Davidson, A.; Schaeffer, J.; Szafron, D. 2001. The Challenge of Poker. *Artificial Intelligence.* Forthcoming.

Billings, D.; Papp, D.; Schaeffer, J.; and Szafron, D. 1998. Opponent Modeling in Poker. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 493–499. Menlo Park, Calif.: American Association for Artificial Intelligence.

Breuker, B.; Uiterwijk, J.; and van den Herik, J. 2000. Solving 8 3 8 Domineering. *Theoretical Computer Science* 20(1–2): 195–206.

Buro, M. 2001. Improving Heuristic Min-Max Search by Supervised Learning. *Artificial Intelligence.* Forthcoming.

Buro, M. 1997. The Othello Match of the Year: Takeshi Murakami vs. LOGISTELLO. *Journal of the International Computer Chess Association* 20(3): 189–193.

Buro, M. 1995. Statistical Feature Combination for

the Evaluation of Game Positions. *Journal of Artificial Intelligence Research* 3:373–382

Campbell, M.; Hoane, J.; and Hsu, F.-H. 2001. DEEP BLUE. Artificial Intelligence. Forthcoming.

Condon, J., and Thompson, K. 1982. Belle Chess Hardware. In *Advances in Computer Chess 3,* ed. M. Clarke, 45–54. New York: Pergamon.

Ebeling, C. 1987. *All the Right Moves*. Cambridge, Mass.: MIT Press.

Felten, E., and Otto, S. 1988. A Highly Parallel Chess Program. In *Proceedings of the Conference on Fifth Generation Computer Systems,* 1001–1009. New York: Springer-Verlag.

Findler, N. 1977. Studies in Machine Cognition Using the Game of Poker. *Communications of the ACM* 20(4): 230–245.

Frank, I. 1998. *Search and Planning under Incomplete Information: A Study Using Bridge Card Play*. New York: Springer Verlag.

Frank, I.; Tanaka-Ishii, K.; Okuno, H.; Nakagawa, Y.; Maeda, K.; Nakadai, K.; and Kitano, H. 2001. And the Fans Are Going Wild! SIG plus MIKE. In *Proceedings of the Fourth International Workshop on RoboCup*. New York: Springer-Verlag. Forthcoming.

Gasser, R. 1995. Efficiently Harnessing Computational Resources for Exhaustive Search. Ph.D. dissertation, Institute of Theoretical Computer Science, ETH Zürich.

Ginsberg, M. 1999. GIB: Steps toward an Expert-Level Bridge-Playing Program. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 584–589. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Ginsberg, M. 1996a. Do Computers Need Common Sense? In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning*, 620–626. San Francisco, Calif.: Morgan Kaufmann.

Ginsberg, M. 1996b. Partition Search. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 228–233. Menlo Park, Calif.: American Association for Artificial Intelligence.

Gordon, S. 1994. A Faster Scrabble Move Generation Algorithm. *Software Practice and Experience* 24(2): 219–232.

Hsu, F. 1999. IBM's DEEP BLUE Chess Grandmaster Chips. *IEEE Micro* 19(2): 70–81.

Hsu, F.; Anantharaman, T.; Campbell, M.; and Nowatzyk, A. 1990a. A Grandmaster Chess Machine. *Scientific American* 263(4): 44–50.

Hsu, F.; Anantharaman, T.; Campbell, M.; and Nowatzyk, A. 1990b. Deep Thought. In *Computers, Chess, and Cognition,* eds. T. Marsland and J. Schaeffer, 55–78. New York: Springer Verlag.

Hyatt, R.; Gower, A; and Nelson, H. 1990. Cray Blitz. In *Computers, Chess, and Cognition,* eds. T. Marsland and J. Schaeffer, 111–130. New York: Springer Verlag.

Iida, H.; Sakuta, M.; and Rollason, J. 2001. The State of the Art in Computer Shogi. *Artificial Intelligence*. Forthcoming.

Junghanns, A., and Schaeffer, A. J. 2001. Enhancing Single-Agent Search Using Domain Knowledge. *Artificial Intelligence* 129(1–2): 219–251.

Junghanns, A., and Schaeffer, J. 1999. Domain-Dependent Single-Agent Search Enhancements. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 570–575. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Keim, G.; Shazeer, N.; Littman, M.; Agarwal, S.; Cheves, C.; Fitzgerald, J.; Grosland, J.; Jiang, F.; Pollard, S.; and Weinmeister, K. 1999. PROVERB: The Probabilistic Cruciverbalist. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, 710–717. Menlo Park, Calif.: American Association for Artificial Intelligence.

Koller, D., and Pfeffer, A. 1997. Representations and Solutions for Game-Theoretic Problems. *Artificial Intelligence* 94(1): 167–215.

Korf, R. 2000. Recent Progress in the Design and Analysis of Admissible Heuristic Functions. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, 1165–1170. Menlo Park, Calif.: American Association for Artificial Intelligence.

Korf, R. 1997. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 700–705. Menlo Park, Calif. : American Association for Artificial Intelligence.

Korf, R., and Felner, A. 2001. Disjoint Pattern Database Heuristics. *Artificial Intelligence*. Forthcoming.

Krol, M. 1999. Have We Witnessed a Real-Life Turing Test. Computer 32(3): 27–30.

Laird, J., and van Lent, M. 2000. Human-Level AI's Killer Application: Interactive Computer Games. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, 1171–1178. Menlo Park, Calif.: American Association for Artificial Intelligence.

Lee, K.-F., and Mahajan, S. 1990. The Development of a World Class Othello Program. *Artificial Intelligence* 43(1): 21–36.

Levy, D., and Beal, D., eds. 1989. *Heuristic Programming in Artificial Intelligence*. New York: Ellis Horwood.

Littman, M.; Keim, G.; and Shazeer, N. 2001. Solving Crossword Puzzles by Computer. *Artificial Intelligence*. Forthcoming.

McCarthy, J. 1997. AI as Sport. *Science* 276: 1518–1519.

Mueller, M. 2001. Computer Go. *Artificial Intelligence*. Forthcoming.

Müller, M. 1999. Computer Go: A Research Agenda. *Journal of the International Computer Chess Association* 22(2): 104–112.

Newell, A.; Shaw, J.; and Simon, H. 1958. Chess-Playing Programs and the Problem of Complexity. *IBM Journal of Research and Development* 2(2): 320–335.

Pfeiffer, J. 1964. Man vs. Machine in the Mechanical Age. *Popular Mechanics,* August, 52–57,172–173.

Plaat, A.; Schaeffer, J.; Pijls, W.; and de Bruin, A. 1996. Exploiting Graph Properties of Game Trees. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 234–239. Menlo Park, Calif.: American Association for Artificial Intelligence.

Powley, C.; Ferguson, C.; and Korf, R. E. 1993. Depth-First Heuristic Search on a SIMD Machine. *Artificial Intelligence* 60(2): 199–242.

Reitman, W.; Kerwin, J.; Nado, R.; Reitman, J.; and Wilcox, B. 1974. Goals and Plans in a Program for Playing Go. In Proceedings of the Association of Computing Machinery National Conference, 123–127. New York: Association of Computing Machinery.

Rosenbloom, P. 1982. A World-Championship-Level Othello Program. *Artificial Intelligence* 19(3): 279–320.

Samuel, A. 1967. Some Studies in Machine Learning Using the Game of Checkers: Recent Progress. *IBM Journal of Research and Development* 11(6): 601–617.

Samuel, A. 1960. Programming Computers to Play Games. In *Advances in Computers, Volume 1,* ed F. Alt, 165–192. San Diego, Calif.: Academic.

Samuel, A. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 3(2): 210–229.

Schaeffer, J. 2000. The Games Computers (and People) Play. In *Advances in Computers 50,* ed. M. Zelkowitz, editor, 189–266. San Diego, Calif.: Academic Press.

Schaeffer, J. 1997. *One Jump Ahead: Challenging Human Supremacy in Checkers*. New York: Springer-Verlag.

Schaeffer, J.; Hlynka, M.; and Jussila, V. 2001. Termporal Difference Learning Applied to a High-Performance Game-Playing Program. Paper presented at the Seventeenth International Joint Conference on Artificial Intelligence, 4–10 August, Seattle, Washington.

Schaeffer, J.; Culberson, J.; Treloar, N.; Knight, B.; Lu, P.; and Szafron, D. 1992. A World Championship Caliber Checkers Program. *Artificial Intelligence* 53(2–3): 273–290.

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

Shapiro, S., and Smith, H. 1977. A Scrabble Crossword Game-Playing Program. Technical Report, 119, Department of Computer Science, State University of New York at Buffalo.

Sheppard, B. 2001. World Championship–Caliber Scrabble. *Artificial Intelligence*. Forthcoming.

Slate, D., and Atkin, L. 1977. Chess 4.5—The Northwestern University Chess Program. In *Chess Skill in Man and Machine,* 82–118. New York: Springer-Verlag.

Smith, S.; Nau, D.; and Throop, T. 1998a. Computer Bridge: A Big Win for AI Planning. *AI Magazine* 19(2): 93–105.

Smith, S.; Nau, D.; and Throop, T. 1998b. Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1079–1086. Menlo Park, Calif.: American Association for Artificial Intelligence.

Strachey, C. 1952. Logical or Non-Mathematical Programmes. In Proceedings of the Association for Computing Machinery Meeting, 46–49. New York: Association of Computing Machinery.

Sutton, R. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3:9–44.

Tesauro, G. 2001. Programming Backgammon Using Self-Teaching Neural Nets. *Artificial Intelligence*. Forthcoming.

Tesauro, G. 1995. Temporal-Difference Learning and td-gammon. *Communications of the ACM* 38(3): 58–68.

Tesauro, G. 1989. Neurogammon Wins Computer Olympiad. *Neural Computation* 1(3): 321–323.

Thompson, K. 1982. Computer Chess Strength. In *Advances in Computer Chess 3,* ed. M. Clarke, 55–56. Oxford, U.K.: Pergamon.

Tinsley, M. 1980. Letter to the Editor. *Scientific American,* August.

Truscott, T. 1979–1980. The Duke Checkers Program. *Journal of Recreational Mathematics* 12(4): 241–247.

Turing, A. 1953. Digital Computers Applied to Games. In *Faster Than Thought,* ed. B. Bowden, 286–295. London: Pitman.

van den Herik, J. 2000. The Fifth Computer Olympiad. *International Computer Games Association Journal* 23(3): 164–187.

van den Herik, J.; Uiterwijk, J.; and van Rijswijck, J. 2001 Games Solved: Now and in the Future. *Artificial Intelligence*.

van Rijswijk, J. 2000. Computer Hex: Are Bees Better Than Fruit Flies? M.Sc. thesis, Department of Computing Science, University of Alberta.

Zobrist, A. 1970. Feature Extractions and Representation for Pattern Recognition and the Game of Go. Ph.D. dissertation, University of Wisconsin.

Jonathan Schaeffer is a professor of computing science at the University of Alberta. He received his B.Sc. from the University of Toronto (1979) and his M.Math (1980) and Ph.D. (1986) degrees from the University of Waterloo. His research interests are in AI and parallel/distributed computing. He is best known for his work on computer games. He is the creator of the checkers program CHINOOK, the first program to win a human world championship in any game. He is also a cofounder of the bioinformatics software company BioTools, Inc. His e-mail address is jonathan@cs.ualberta.ca.