Games of No Chance
MSRI Publications
Volume **29**, 1996

Solving the Game of Checkers

JONATHAN SCHAEFFER AND ROBERT LAKE

Abstract. In 1962, a checkers-playing program written by Arthur Samuel defeated a self-proclaimed master player, creating a sensation at the time for the fledgling field of computer science called artificial intelligence. The historical record refers to this event as having solved the game of checkers. This paper discusses achieving three different levels of solving the game: publicly (as evidenced by Samuel's results), practically (by the checkers program *Chinook*, the best player in the world) and provably (by considering the 5×10^{20} positions in the search space). The latter definition may be attainable in the near future.

1. Introduction

Checkers is a popular game around the world, with over 150 documented variations. Only two versions, however, have a large international playing community. What is commonly known as checkers (or draughts) in North America is widely played in the United States and the British Commonwealth. It is played on an 8×8 board, with checkers moving one square forward and kings moving one square in any direction. Captures take place by jumping over an opposing piece, and a player is allowed to jump multiple men in one move. Checkers promote to kings when they advance to the last rank of the board. So-called international checkers is popular in the Netherlands and the former Soviet Union. This variant uses a 10×10 board, with checkers allowed to capture backwards, and kings allowed to move many squares in one direction, much like bishops in chess. This paper is restricted to the 8×8 variant, but many of the ideas presented here also apply to the 10×10 game.

People enjoy playing games of skill because of the intellectual challenge and the satisfaction derived from playing well. Many board games, such as chess and checkers, have too many possibilities for a human to understand them all. Hence they use knowledge and search to make their decisions at the board. The person with the best "algorithm" for playing the game wins in the long run. Without perfect knowledge, mistakes are made, and even World Champions will lose occasionally (for example, the world chess champion, Gary Kasparov, may lose three or four games a year).

This gives rise to an intriguing question: is it possible to program a computer to play a game perfectly? Can the game-theoretic value of checkers be determined? In other words, is it possible to solve the game? In recent years, some games have been solved, including Connect-Four [Allen 1989; Allis 1988], Qubic [Patashnik 1980], Nine Men's Morris [Gasser 1994] and Go-Moku [Allis et al. 1993].

This paper describes three ways in which it is possible to solve the game of checkers. Section 2 deals with *publicly* solving the game, creating the impression in the media that checkers has been solved. Section 3 deals with *practically* solving the game, creating a computer player that is better than all humans but is not perfect. Section 4 deals with *provably* solving the game, determining the game-theoretic value and a strategy for always achieving that value. For the game of checkers, publicly solving the game is a thing of the past, practically solving it is the present, and provably solving it is the near future.

2. Publicly Solving Checkers

In the late 1950's and early 1960's, Arthur Samuel did pioneering work in artificial intelligence using the game of checkers as his experimental testbed [Samuel 1959; Samuel 1967]. Thirty years later, his work is still remembered, both for the significance of his research contributions, and for the legacy of his checkers-playing program. In 1962, Robert Nealy, blind checkers champion of Stamford, Connecticut, lost a single game to Dr. Samuel's program. For the fledgling field of artificial intelligence, this event was viewed as a milestone and its significance was misrepresented in the media. Reporting of this event resulted in the game of checkers being labeled as "solved": computers were better than all humans. Thus, a 1965 article by Richard Bellin in the *Proceedings of the National Academy of Sciences* declared that " ... it seems safe to predict that, within ten years, checkers will be a completely decidable game" (vol. 53, p. 246), while Richard Restak, in the influential book *The Brain: The Last Frontier* (1979), stated that " ... an improved model of Samuel's checker-playing computer is virtually unbeatable, even defeating checkers champions foolhardy enough to 'challenge' it to a game" (p. 336).

How good was Samuel's program? Although Nealy advertised himself as a master, the highest level he achieved was the class below master [Fortman 1978]. Analysis of the fateful game showed that Nealy made a trivial mistake, possibly indicating that he was not taking his electronic opponent seriously. A match played a year later resulted in a decisive victory for Nealy, without loss of a single game.

As a result of the one win against Nealy, checkers was classified as an uninteresting problem domain and all the artificial intelligence research that might have used checkers as an experimental testbed switched to using chess. The perception that checkers is a solved game persists to the present time, and has been a major obstacle to anyone conducting research using this game. Modern artificial intelligence books now treat the subject of Samuel's program's performance more realistically, for example indicating that it "came close to expert play" [Barr and Feigenbaum 1981].

Dr. Samuel had no illusions about the strength of his program. In personal correspondence, he apologized for the problems created by the misconception that checkers was solved [Tinsley 1994] and admitted he had no idea how he could make his program good enough to compete with the world champion [Smith 1994].

3. Practically Solving Checkers

The first checkers-playing program is credited to Strachey [1952]. Samuel began his program in the early 1950's and continued working on it on and off for over two decades [Samuel 1959; 1967]. In 1965, the program played four games each against Walter Hellman and Derek Oldbury (then playing a match for the World Championship), and lost all eight games. In the late 1970's a team at Duke University headed by Eric Jansen developed the first program capable of defeating masters [Truscott 1978]. In May 1977, their program *PAASLOW* had the distinction of defeating Grandmaster Elbert Lowder in a nontournament game (the program recovered from a lost position), but lost their friendly match with one win, two losses and two draws [Fortman 1978]. The Duke team originally accepted a $5,000 challenge match to play the World Champion, Dr. Marion Tinsley, but eventually decided not to play [Anonymous 1979].

In 1989 and 1990, the First and Second Computer Olympiads were held in London, and three strong programs emerged [Levy and Beal 1989; 1991]. The programs *Colossus* (Martin Bryant) and *Checkers* (Gil Dodgen) were aimed at the PC market, while *Chinook* (Jonathan Schaeffer *et al.*) was a research project. *Colossus* and *Checkers* are still commercially available and are continually being updated. Both are ranked among the top twenty players in the world.

Chinook is a checkers program developed at the University of Alberta beginning in 1989. In 1990, it earned the right to play for the human World Checkers Championship, by coming an undefeated second to Tinsley, the World Champion, at the U.S. Open. This was the first time a program had earned the right to play for a human World Championship. Computers have played World Champions before—for example *BKG9* at backgammon [Berliner 1980] and *DeepThought* at chess [Kopec 1990]—but these were exhibition events with no title at stake.

The American Checker Federation and English Draughts Association (the governing bodies for checkers) refused to sanction the *Chinook*–Tinsley match, arguing that the World Championship was for humans, not machines. Eventually, they created the Man-Machine World Championship title to allow the match to proceed, but only after Tinsley resigned his "human" World Championship title in protest. In 1992, Tinsley defeated *Chinook* in their title match by a score of four wins to two with 33 draws [Schaeffer 1992; Schaeffer et al. 1993a].

In 1994, *Chinook* became World Champion when, after six games in their rematch (all draws), Tinsley resigned the match and the title citing health concerns. *Chinook* is the first computer program to become World Champion in any nontrivial game of skill. Obviously this was not a satisfactory way to win a World Championship. Subsequently, *Chinook* successfully defended its title twice against Grandmaster Don Lafferty. Unfortunately, with the recent passing of Marion Tinsley [Schaeffer 1995], mankind has lost its best chance of wresting the title away from the computer.

Chinook has not solved the game, but it is playing at a level that makes it almost unbeatable. In its last 242 games against the strongest players in the world (over 220 of which were against Grandmasters), *Chinook* only lost one game (to be fair, it also drew one game from a lost position). Since the program continues to automatically increase its checkers knowledge every day, *Chinook* will eventually be almost unbeatable. In effect, checkers will be solved in practice, if not in theory.

Chinook's strength comes from deep searches (deciding which positions to examine), a good evaluation function (deciding how favorable a position is), endgame databases (containing perfect information on all positions with eight pieces or less) and a library of opening moves [Schaeffer et al. 1992].

SEARCH: *Chinook* uses an iterative, alpha-beta search with transposition tables and the history heuristic [Schaeffer 1989]. Under tournament conditions (thirty moves an hour), the program searches to an average minimum depth of nineteen ply (one ply is one move by one player). The search uses selective deepening to extend lines that are tactically or positionally interesting. Consequently, major lines of play are often searched many plies deeper. It is not uncommon for the program to produce analyses that are thirty-ply deep or more.

KNOWLEDGE: The evaluation function has 25 heuristic components, each of which is weighted and summed to give a position evaluation. The game is divided into four phases, each with their own set of weights. The definition of the heuristics and their weights was arrived at only after long discussions with a checkers expert. The knowledge acquisition process goes on. The knowledge we add to the program nowadays tends to be subtle: it is harder to define and often covers exceptions to the general knowledge that is already programmed into *Chinook*.

DATABASES: All checkers positions with eight or fewer pieces (checkers or kings) on the board have had their game-theoretic value (win, loss or draw) computed [Lake et al. 1994]. The knowledge contained in the eight-piece databases exceeds human capabilities. Databases are discussed in more detail in the next section.

OPENING BOOK: The program has a library of 60,000 opening positions, each of which has been computer verified. Most of the positions originate from the checkers literature. However, computer verification has resulted in hundreds of corrections to the literature and some refutations of major lines of play (not published, until we have a chance to use them in a game!).

It is difficult for humans to compete with a program searching as deep as *Chinook* does. Only the best players in the world can maintain a level of vigilance sufficient to compensate for this. Nevertheless, it is clear that there is an important class of positions where even fifty-ply searches are insufficient to uncover the subtleties of the position [Schaeffer et al. 1993b]. The rare times that the program loses all have the same pattern. The opponent gets the program out of its opening library early in the game where the strategic aspects of the position require a search of roughly more than 25 ply. If the program does not have the proper knowledge to play this position, it may make a mistake and, against only the very best players, possibly lose a game. With a deeper search, an improving evaluation function, more endgame databases and a growing opening library, the probability of this happening is rapidly decreasing.

Although *Chinook* is not perfect and can still lose a game, in practice checkers can be regarded as “solved”.

4. Provably Solving Checkers

What does it mean to solve a game? Allis [1994] defines three levels:

ULTRA-WEAKLY SOLVED: The game-theoretic value for the initial position has been determined.

WEAKLY SOLVED: The game is ultra-weakly solved and a strategy exists for achieving the game-theoretic value from the opening position, assuming reasonable computing resources.

STRONGLY SOLVED: For all possible positions, a strategy is known for determining the game-theoretic value for both players, assuming reasonable computing resources.

Ultra-weakly solving a game is of theoretical interest but means little in practice. For example, Allis cites the game of Hex, which is known to be a win for the first player on all diamond-shaped boards, but no one knows a strategy for achieving this result. Strongly solved games include simple domains such as tic-tac-toe, and more complex domains such as chess and checkers endgames (for which databases enumerating all possibilities have been constructed).

Note the qualification on resources in the above definitions. In some sense, a game such as chess is solved since we have an algorithm (alpha-beta searching) that could return the game-theoretic value given an unlimited amount of time. The resource constraints preclude such solutions.

To these definitions, we can add a fourth:

Ultra-strongly solved: For all positions in a strongly solved game, a strategy is known that improves the chances of achieving more than the game-theoretic value against a fallible opponent.

This point is very important in practice. Knowing that a position is, for example, a draw only means that this is the best one can achieve against a perfect opponent. Many opponents (particularly humans) are fallible, and selecting a move based on the probability of opponent error increases the expected outcome of a game.

How difficult is it to solve a game? Allis, van den Herik and Herschberg argue that there are two dimensions [Allis et al. 1991]:

Space complexity: the number of positions in the search space.

Decision complexity: the difficulty required to make correct decisions.

A first impression suggests that, the larger the search space, the harder the problem is to solve. This is not true, as illustrated by a trivial variant of the game of Go on a 19×19 board (361 squares). In this game, the two players alternate making moves, where a move consists of placing a stone on an empty square. The first person to fill in 181 squares wins [Allis et al. 1991]. This game has a search complexity of $3^{361} \approx 10^{170}$, but is trivial to solve.

All the games solved thus far have either low decision complexity (10^{30} positions for Qubic, 10^{105} positions for Go-Moku) or low space complexity (10^{11} positions for Nine Men's Morris) or both (10^{14} positions for Connect-Four). Checkers is rated as having high decision complexity (more complex than 9×9 Go and the play of bridge hands, on a par with backgammon and 10×10 draughts, and less complex than chess) and moderate space complexity (10^{20} positions versus unsolved games such as backgammon with 10^{19} positions, chess with 10^{44} positions, and Scrabble with 10^{150} positions).

A bound on the space complexity of checkers can be derived. The naive calculation of 5^{32} (each of 32 squares can be occupied by a checker or king of either color, or be empty) can be improved upon. Let b be the number of black checkers, w the number of white checkers, B the number of black kings, W the number of white kings and f the number of black checkers on the first rank. In the following, 12 is the maximum number of pieces per side, 32 is the number of squares on the board that can be occupied, 28 is the number of squares that it is legal to place a checker on (checkers become kings on the last rank) and 4 is the number of squares on black's first rank (these squares cannot hold any white

pieces	positions
1	120
2	6,972
3	261,224
4	7,092,774
5	148,688,232
6	2,503,611,964
7	34,779,531,480
8	406,309,208,481
9	4,048,627,642,976
10	34,778,882,769,216
11	259,669,578,902,016
12	1,695,618,078,654,976
13	9,726,900,031,328,256
14	49,134,911,067,979,776
15	218,511,510,918,189,056
16	852,888,183,557,922,816
17	2,905,162,728,973,680,640
18	8,568,043,414,939,516,928
19	21,661,954,506,100,113,408
20	46,352,957,062,510,379,008
21	82,459,728,874,435,248,128
22	118,435,747,136,817,856,512
23	129,406,908,049,181,900,800
24	90,072,726,844,888,186,880
total	500,995,484,682,338,672,639

Table 1. Search space complexity of checkers.

checkers). The number of positions having n or fewer pieces on the board is

$$\sum_{b=0}^{\min(n,12)} \sum_{B=0}^{\min(n,12)-b} \sum_{w=0}^{\min(n-b-B,12)} \sum_{W=0}^{\min(n-b-B,12)-w} \sum_{f=0}^{\min(b,4)} \mathrm{Num}(b,B,w,W,f)-1,$$

where

$$\mathrm{Num}(b,B,w,W,f)=\binom{4}{f}\binom{24}{b-f}\binom{28-(b-f)}{w}\binom{32-b-w}{B}\binom{32-b-w-B}{W}.$$

The subtraction of one position in the above formula handles the impossible case of zero pieces on the board. This yields a search complexity of roughly 5×10^{20} positions (Table 1).

This number is misleading, since it includes positions that are not legally reachable from the start of the game. For example, although there are 9×10^{19}

plausible positions with 24 pieces on the board, only a small fraction of this number are legal. It is possible to place 24 pieces on the board and have 10 of them be kings. However, there is no legal move sequence from the start of the game that can reach such a position, since the forced capture rule precludes this. It is possible to have 24 pieces on the board with two of them being kings, but this move sequence is contorted and cooperative and has no relationship to a reasonable playing strategy.

A bound on the number of reasonable positions one might encounter in trying to prove checkers is obtained using the following assumptions. First, consider only those positions where the difference between the number of black and white pieces is less than three, that is, $|(b + B) - (w + W)| < 3$. More lopsided positions will always be a win for the strong side unless (1) there is an immediate capture present to restore the material balance (which reduces the position to one satisfying the above condition), or (2) the pieces of the strong side are constricted in a manner that is impossible to reach either by the rules of the game, or by considering a reasonable playing strategy.

Within the subset of positions created by the above assumption, a number of positions can be further discarded because they are prevented from occurring by the rules of the game or by reasonable playing strategy. For example, the rules condition eliminates all positions with 24 kings on the board. The reasonable playing strategy assumption enables us to consider only those positions that will occur in normal play. One observation is that as the number of pieces on the board decreases, the probability of there being multiple kings increases. The number of kings can be restricted so that scenarios such as 12 kings versus 12 kings are eliminated. Define a subset of the search space that limits the number of kings for positions with 12 or more pieces as follows:

$$(B \leq \lceil 24 - \tfrac{1}{2}n \rceil,\, W \leq \lfloor 24 - \tfrac{1}{2}n \rfloor) \cup (B \leq \lfloor 24 - \tfrac{1}{2}n \rfloor,\, W \leq \lceil 24 - \tfrac{1}{2}n \rceil)$$

subject to the conditions $12 \leq n \leq 24$ and $|(b + B) - (w + W)| < 3$. Even this assumption is generous: it includes unlikely positions such as 6 black kings against 6 white checkers.

Summing the positions satisfying these conditions reduces the search space to slightly less than 10^{18} positions. Still included in this number are some positions not legally reachable from the start of the game. We have found no way to mathematically define this set. It is possible that the exclusion of these positions may reduce the plausible search space from 10^{18} to 10^{17} or less.

A search space of 10^{18} still seems impossibly large. If 10^6 positions were solved per second, $100,000$ years of computing would be required to enumerate all the positions. What makes the problem of finding the game-theoretic value of checkers feasible is the idea of a proof tree. When analyzing a position that is a win, one need only consider one winning move; the alternatives moves are either inferior or redundant. When considering a position that is a loss or a draw, all alternatives must be considered (since you have to prove you cannot do better). In the best case—a game that is provably winning—the search only has to explore roughly the square root of the search space: a single move at winning positions and all moves at losing positions. This means that a proof of the game-theoretic value of checkers might require a search of as few as 10^9 positions. Of course, knowing which 10^9 positions out of 10^{18} is the hard part.

To weakly solve checkers (determine the game-theoretic value), we have devised a three-pronged approach:

BOTTOM-UP: The only positions with known game-theoretic values, as defined by the rules of the game, are those positions where one side has no pieces or no moves (a loss). Using retrograde analysis, the value of these terminal positions can be backed up to positions many moves from the end of the game [Thompson 1986]. We have built endgame databases that solve all 443,748,401,247 positions with eight or fewer pieces on the board [Lake et al. 1994].

Endgame databases essentially reduce the number of moves required to be played from the start of the game before one reaches a position with a known game-theoretic value.

MIDDLE: We have collected thousands of positions with more than eight pieces on the board for which we have determined the game-theoretic value. They come from games played by *Chinook*, where the search was deep enough to demonstrate that an endgame database value could be backed up to the root position of the search. For example, we have proved that a position with as many as 22 pieces on the board is a win. The proof is simpler than one might expect, because forced capture moves in the analysis quickly reduce the number of pieces to eight.

In effect, we are building a middlegame database of solved positions with more than eight pieces on the board.

TOP-DOWN: A proof consists of demonstrating a sequence of moves from the start of the game to the endgame databases that achieves the maximum possible value (of win, draw or loss). Once the start of the game is assigned an accurate value, the game theoretic value of checkers has been determined. In tournament checkers, the first three half-moves are randomly chosen (or balloted) among a set of 144 opening sequences. In other words, the proof can be treated as 144 subproblems.

Starting at the beginning of the game and doing a search is counterproductive: the search will be too large. Instead, we use published checkers analysis to identify the "best" lines of play for each side in each opening. By playing down these lines, we try to solve positions further down the line of play (thereby creating new entries for the middlegame database) and then use those results to try and solve positions closer to the start of the line. In effect, the opening is used to guide the search to the parts of the search space most likely to be profitable.

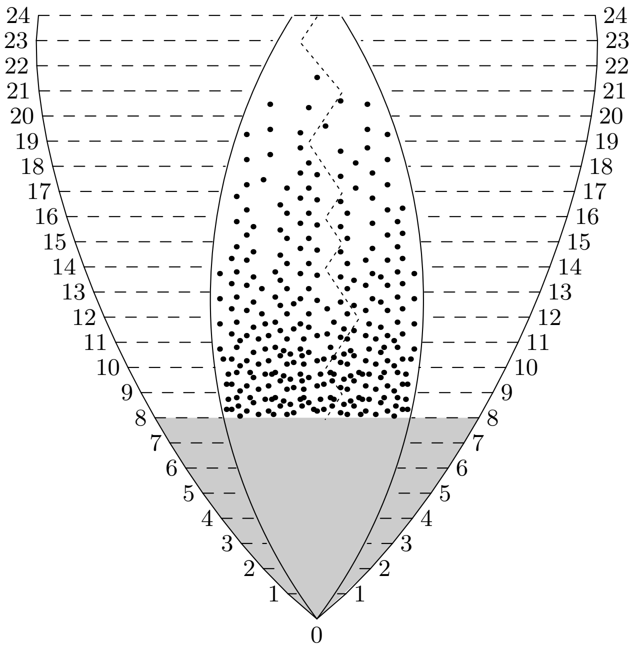

Figure 1. Checkers search space.

Figure 1 illustrates this three-pronged approach. Despite the deceptive appearance, the diagram is drawn to scale based on the numbers in Table 1. It shows the number of pieces on the board (vertically) versus the logarithm of the number of positions (base 10) with that many pieces on the board (horizontally). Many of the positions are not legally reachable by the rules of the game, and the logarithm of the estimated illegal number is shown in the region of dashed lines. The endgame phase of the proof is the shaded area: all positions with eight or fewer pieces. The middlegame database contains positions with more than eight pieces for which a value has been proved (shown as dots in the diagram). Note that a dot is misleading because each represents the root of a search tree which has been solved (and, by implication, all positions in that proof tree must also have been solved). None of these lower-level solved positions is presented in the diagram. The dotted line represents an opening line of play (as suggested by the checkers literature). It goes from the start of the opening to the endgame databases. By repeatedly searching that line, more middlegame database positions can be solved, thereby simplifying the task of proving the start position.

Critical to the success of this approach is the search algorithm chosen. Our initial experience is with two search approaches: proof number search (PNS) and alpha-beta. Allis has been successful with his PNS approach [Allis et al. 1994], a variation on McAllester's conspiracy numbers algorithm [McAllester 1988]. PNS builds a tree to variable depth trying to answer a binary question (for example, is this position a win?). Each node in a search tree whose value has not been proven is given a *proof* and *disproof* number. The proof number indicates the number of nodes in the tree whose value has to be determined to prove the assertion (the position is a win), while the disproof number indicates the number of nodes in the tree whose value has to be shown false to prove the assertion untrue (the position is not a win). The search algorithm examines the path of least resistance, the line of play with the lowest proof/disproof number. These numbers exploit the nonuniform branching factor in the search tree. For example, in chess, a forced sequence of checking moves leaves the opponent with few responses and, hence, few positions that need to be proven. The algorithm is intuitively appealing and has been successfully used to solve Connect-Four and Qubic.

There is a simple alternative search method to consider: alpha-beta. It can be used to prove the value of a position by modifying the evaluation function to return ∞ for a win, 0 for a draw and $-\infty$ for a loss or unproven position. Under the pessimistic assumption that all unproven positions are losses, a nonloss value returned by a search is a useful result and can only be improved by finding the correct value for the unproven positions. Thus, if a search returns a win value we know the position is a win, and if it returns a draw value, the position is at least a draw (it might still be a win).

Initial experience with the two algorithms has had some surprising results. Generally alpha-beta significantly outperforms PNS, but occasionally PNS wins by an enormous amount. Experiments performed by Victor Allis and Jonathan Schaeffer on the games of the 1994 *Chinook*–Oldbury match showed that many positions could be solved with 25 ply of search and were quickly found by alpha-beta. PNS, on the other hand, would explore some lines to excessive search depths looking for a more complicated solution than was needed. PNS excels when there is a nonuniform branching factor in the search tree. Ignoring capture moves (which are forced), the number of legal moves in a position changes little from move to move. Thus PNS has little information to guide its search. Research needs to be done on finding a more suitable heuristic for indicating to PNS where success is likely to be found.

Figure 2 illustrates the potential for deep alpha-beta searching. There are eighteen pieces on the board, so the endgame databases seem quite distant. It is white's tenth move (or ninth if you consider that white's first move was balloted). *Chinook* (white) played its checker from g1 to h2, whereupon Tinsley immediately remarked "You're going to regret that". On move 36, *Chinook* resigned. Since this game was played in 1990, the endgame databases have expanded from six to eight pieces and the search depths of the program have significantly increased. Today, the program can prove that the move g1 to h2 is a loss and b4 to a5 is a draw. This proof was done using alpha-beta by starting at the end of the game (move 36), searching that position and proving it to be lost, adding that result to the middlegame database, and then backing up to move 35 and trying to prove that position, etc. Thus this position, only ten

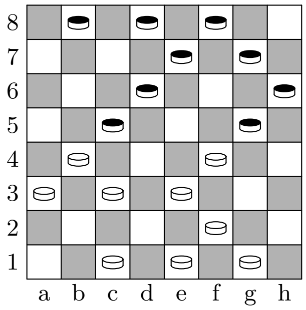

Figure 2. *Chinook*–Tinsley, 1990: White to play

moves from the start of the game (nine if you exclude balloted moves) has been solved. Subsequent searches of other positions are now simplified, since if they encounter this complicated position in their search, they need look no further; the middle game database contains the result.

The above discussion has concentrated on weakly solving the game. Ultra-strongly solving the game requires additional information; the program must choose the move to maximize its chances of improving its result. For example, given a drawn (lost) position, the program must select the move that maintains the draw (loss) and maximizes the chances of the opponent making a mistake that changes the game's outcome. In general this is a difficult problem, since maximizing difficulty is usually a function of the strengths and weaknesses of the opponent.

The biggest software advance in *Chinook* in the past two years has been an improvement in its ability to differentiate between drawing moves. In most game-playing programs, the value of a draw is 0, equality. *Chinook* differentiates between drawn positions by assigning them a value in the range of $+\frac{1}{3}$ of a checker to $-\frac{1}{12}$ of a checker. Note the asymmetric range of values. A very weak draw is still a draw and must have a score close to zero. If not, a weak nondrawing move might lead to a position with a higher score and be selected as the move to play. In tournaments, this scheme has resulted in several drawn games becoming wins when the program made it difficult for the opponent to find the drawing line.

The current status of the project is:

- Our endgame database consists of 4.4×10^{11} positions, that is, all positions with eight or fewer pieces. Completing the nine-piece databases will require computational and storage requirements beyond our current capabilities. We have begun the computing but expect to have only a small percent of the 10^{12} positions completed in 1995.
- The opening lines have been determined. This was done in consultation with our checkers expert, Martin Bryant, and his extensive literature on the game.
- The middlegame database contains only positions that occurred in *Chinook*'s tournament games. We are currently starting a systematic search of one

particular opening where it appears likely we can prove a draw. Success in one opening is all that is necessary to validate our approach.

5. Conclusions

With current technology, we believe it is possible to weakly solve the game of checkers. A database containing a proof tree for the game may contain as few as 10^9 nodes—a small number by today's standards. Strongly solving the game will be much harder, since it requires knowledge of all 10^{18} legally reachable positions in the game. The combination of weakly solving the game with the deep search, endgame databases and draw differentiation facilities in *Chinook* means that it will be possible to build a program that is close to ultra-strongly solving the game in practice.

Acknowledgements

Many people have contributed to *Chinook* over the years, including Martin Bryant, Joseph Culberson, Randal Kornelson, Brent Knight, Paul Lu, Steve Sutphen, Duane Szafron and Norman Treloar. Aske Plaat provided detailed feedback on the final draft of this paper. The support from Silicon Graphics International and the MPCI project at Lawrence Livermore Laboratories is greatfully appreciated. This research was supported by the Natural Sciences and Engineering Research Council of Canada (grant number OGP-8173) and the Netherlands Organization for Scientific Research (NWO).

References

[Allen 1989] J. D. Allen, "A note on the computer solution of connect-four", pp. 134–135 in *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad* (edited by D. N. L. Levy and D. F. Beal), Ellis Horwood, Chichester, England, 1989.

[Allis 1988] L. V. Allis, "A knowledge-based approach to connect-four. The game is solved: White wins", M.Sc. thesis, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, 1988.

[Allis 1994] L. V. Allis, "Searching for solutions in games and artificial intelligence", Ph.D. thesis, Dept. of Computer Science, University of Limburg, 1994.

[Allis et al. 1991] L. V. Allis, H. J. van den Herik, and I. S. Herschberg, "Which games will survive", pp. 232–243 in *Heuristic Programming in Artificial Intelligence: The Second Computer Olympiad* (edited by D. N. L. Levy and D. F. Beal), Ellis Horwood, Chichester, England, 1991.

[Allis et al. 1993] L. V. Allis, H. J. van den Herik, and M. P. H. Huntjens, "Go-Moku solved by new search techniques", pp. 1–9 in *Proc. AAAI Fall Symposium on Games: Planning and Learning*, AAAI Press, Report FS9302, Menlo Park, California, 1993.

[Allis et al. 1994] L. V. Allis, M. van der Meulen, and H. J. van den Herik, "Proof-number search", *Artificial Intelligence* **66** (1994), 91–124.

[Anonymous 1979] Anonymous, "Computer Checkers: The Tinsley Challenge", *Personal Computing* (July 1979), 88–89.

[Barr and Feigenbaum 1981] A. Barr and E. Feigenbaum, *The Handbook of Artificial Intelligence*, Kaufmann, Los Altos, CA, 1981.

[Berliner 1980] H. Berliner, "Backgammon computer program beats World Champion", *Artificial Intelligence* **14** (1980), 205–220.

[Fortman 1978] R. L. Fortman, "Duke vs. Lowder", *Personal Computing* (November 1978), 26–28.

[Gasser 1994] R. Gasser, "Harnessing computational resources for efficient exhaustive search", Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1994.

[Kopec 1990] D. Kopec, "Advances in Man-Machine Play", pp. 9–32 in *Computers, Chess and Cognition* (edited by T. A. Marsland and J. Schaeffer), Springer, New York, 1990.

[Lake et al. 1994] R. Lake, J. Schaeffer, and P. Lu, "Solving large retrograde analysis problems using a network of workstations", pp. 135–162 in *Advances in Computer Chess* 7 (edited by H. J. van den Herik et al.), University of Limburg, Maastricht, Netherlands, 1994.

[Levy and Beal 1989] *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad* (edited by D. N. L. Levy and D. F. Beal), Ellis Horwood, Chichester, England, 1989.

[Levy and Beal 1991] *Heuristic Programming in Artificial Intelligence: The Second Computer Olympiad* (edited by D. N. L. Levy and D. F. Beal), Ellis Horwood, Chichester, England, 1991.

[McAllester 1988] D. A. McAllester, "Conspiracy numbers for min-max search", *Artificial Intelligence* **35** (1988), 287–310.

[Patashnik 1980] O. Patashnik, "Qubic: $4 \times 4 \times 4$ tic-tac-toe", *Math. Mag.* **53** (1980), 202–216.

[Samuel 1959] A. L. Samuel, "Some studies in machine learning using the game of checkers", *IBM J. Res. Devel.* **3** (1959), 210–229.

[Samuel 1967] A. L. Samuel, "Some studies in machine learning using the game of checkers: recent progress", *IBM J. Res. Devel.* **11** (1967), 601–617.

[Schaeffer 1989] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice", *IEEE Trans. on Pattern Anal. and Machine Intelligence* **11** (1989), 1203–1212.

[Schaeffer 1992] J. Schaeffer, "Man versus machine: The Silicon Graphics World Checkers Championship", Technical Report 92-20, Dept. of Computing Science, University of Alberta, 1992.

[Schaeffer 1995] J. Schaeffer, "Marion Tinsley: Human Perfection at Checkers?", pp. 115–118 in this volume.

[Schaeffer et al. 1992] J. Schaeffer et al., "A World Championship caliber checkers program", *Artificial Intelligence* **53**(2–3) (1992), 273–290.

[Schaeffer et al. 1993a] J. Schaeffer, N. Treloar, P. Lu, and R. Lake, "Man versus machine for the World Checkers Championship", *AI Magazine* **14**(2) (1993), 28–35.

[Schaeffer et al. 1993b] J. Schaeffer, P. Lu, D. Szafron, and R. Lake, "A re-examination of brute-force search", pp. 51–58 in *Proc. AAAI Fall Symposium on Games: Planning and Learning*, AAAI Press, Report FS9302, Menlo Park, California, 1993.

[Smith 1994] H. Smith, personal communication citing letter sent to Smith by A. L. Samuel (1994).

[Strachey 1952] C. S. Strachey, "Logical or nonmathematical programs", *Proceedings of the ACM Conference, Toronto*, 1952.

[Thompson 1986] K. Thompson, "Retrograde analysis of certain endgames", *J. International Computer Chess Assoc.* **9**(3) (1986), 131–139.

[Tinsley 1994] M. Tinsley, personal communication citing letter sent to Tinsley by A. L. Samuel (1994).

[Truscott 1978] T. Truscott, "The *Duke* checkers program", preprint, Duke University, 1978.

Jonathan Schaeffer
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada T6G 2H1

Robert Lake
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada T6G 2H1