The Historical Development of Computer Chess and its Impact on Artificial Intelligence

David Heath and Derek Allum
Faculty of Science and Computing,
University of Luton,
Park Square,
Luton LU1 3JU
United Kingdom
david.heath@luton.ac.uk
derek.allum@luton.ac.uk

Abstract

In this paper we review the historical development of computer chess and discuss its impact on the concept of intelligence. With the advent of electronic computers after the Second World War, interest in computer chess was stimulated by the seminal papers of Shannon (1950) and Turing (1953). The influential paper of Shannon introduced the classification of chess playing programs into either type A (brute force) or type B (selective). Turing's paper (1953) highlighted the importance of only evaluating 'dead positions' which have no outstanding captures. The brute force search method is the most popular approach to solving the chess problem today. Search enhancements and pruning techniques developed since that era have ensured the continuing popularity of the type A method. Alpha-beta pruning remains a standard technique. Other important developments are surveyed. A popular benchmark test for determining intelligence is the Turing test. In the case of a computer program playing chess the moves are generated algorithmically using rules that have been programmed into the software by a human mind. A key question in the artificial intelligence debate is to what extent computer bytes aided by an arithmetic processing unit can be claimed to 'think'.

Introduction

With the advent of computers after the end of the Second World War, interest in the development of chess playing programs was stimulated by two seminal papers in this area. The paper by Shannon (1950) remains even to this day to be of central importance while the paper by Turing (1953) is equally influential.

The minimax algorithm was first applied in a computer chess context in the landmark paper of Shannon. He also introduced the classification of chess playing programs into either type A or B. Type A are those that search by 'brute force' alone, while type B programs try and use some considerable selectivity in deciding which branches of the game tree require searching.

Alpha-beta pruning was first formulated by McCarthy at the Dartmouth Summer Research Conference on Artificial Intelligence in 1956. However, at this stage no formal specification of it was given, but it was implemented in game playing programs of the late 1950s. Papers by Knuth and Moore (1975) and Newborn (1977) have analysed the efficiency of the method and it has been proved that the algorithm returns exactly the same move as that obtained by full minimaxing or, alternatively, a move of the same value.

The success of type A 'brute force' programs using exhaustive search, minimaxing and alpha-beta pruning, transposition tables and other search enhancements has had the unfortunate effect of minimising interest in the development of type B programs. The work of Simon and Chase (1973) established that most humans only consider a handful of plausible moves. The power and ability of the chess grandmaster resides in his ability to select the correct subset of moves to examine.

In contrast the brute force programs search the entire spectrum of initial moves diverging from some given position referred to as the root. These initial moves fan out, generating a game tree which grows exponentially with depth. Apart from the work of Botvinnik et.al in recent years, there has been no significant progress in developing a type B strategy program which

would reduce the initial span of the search tree. The successful development of algorithms that introduced selectivity into the search engine, so that the program followed similar lines of thought to those of a chess grandmaster, would considerably reduce the amount of searching required.

Turing's paper (1953) highlighted the importance of only evaluating 'dead positions' which have no outstanding captures. It is from this paper the term 'Turing dead' is taken. Most chess programs search to a quiescent position (Turing dead) and then evaluate the position as a function of the material balance and other features. This evaluation function encapsulates chess-specific knowledge typically relating to pawn structures and king safety.

History

The historical evolution of computer chess programming techniques and knowledge can be conveniently discussed in three broad eras. Each era is characterised by its own particular developments, some of which can be directly linked to increased processor power, the availability of new hardware devices and others to algorithmic advances.

The boundary between each era is not always precise as it is sometimes not easy to draw a clear dividing line across a time continuum. Any such process is artificial. However, these broad historical eras are (Donskoy and Schaeffer 1990):

(i) 1st era (1950 - c1975)

(ii) 2nd era (c1975-c1985)

(iii) 3rd era (c1985 onwards)

The first *pioneering* era as stated above runs from 1950-c1975. Here there is a definite point at which this history commences, marked by the publication of Shannon's paper and the advent of electronic computers. These computers, although originally regarded as revolutionary and powerful, had but a fraction of the computing power of the current generation of microprocessors. Indeed hardware limitations characterise the first era of computer chess development, requiring highly selective techniques in order to produce moves in an acceptable time. The earliest programs were, therefore, Shannon type B.

The first significant chess playing program was by Bernstein (1957) and ran on an IBM 704 computer, capable of performing approximately 42,000 operations per second. This was not a 'brute force' program as it only selected the best seven moves for consideration using heuristics based on chess lore. Compared to the sophisticated brute force programs of today which generate the full span of moves at the root, this is a very limited range of moves.

The Bernstein program was built around the strategy of working to a plan. As it was incapable of doing a full width search due to time considerations, it selected its quota of seven moves for deeper analysis by seeking the answer to eight questions. Once the moves were selected, they were analysed to a search depth of 4 ply. The potential replies by the opponent were also selected on the basis of seeking answers to the same set of questions.

Bernstein's program played at a very elementary level and the first program to attain any recognisable standard of play was that of Greenblatt (1968). For a number of years this remained the most proficient chess program and played at an Elo strength of approximately 1500. It had carefully chosen quiesence rules to aid tactical strength and was also the first program to use transposition tables to reduce the search space. However, the Greenblatt program also used an initial selection process to minimize the size of the game-tree as the computing hardware of that era was incapable of achieving the computing speeds of today. Again this program, because of its selectivity at the root node, falls into the first era.

The first program to achieve full width search and make 'brute force' appear a viable possibility was Chess 4.5. This program was developed for entry into the ACM 1973 Computer Chess contest by Slate and Atkin, using the experience they had gained in programming earlier selective search chess programs. The techniques used by Slate and Atkin (1977) are still in use today although they have been refined and improved over the years. These standard techniques initiated what may be classed as the second *technology* era giving rise to programs typically searching to a fixed depth as fast as possible and then resolving the horizon problem by extending checks at the cutoff depth and considering captures only in a quiescence search.

The second era of computer chess also saw more emphasis placed on the development of dedicated chess hardware. Typical of such developments was Ken Thompson's chess machine *Belle* which won the ACM North American Computer Chess Championship in 1978. The special purpose hardware increased the speed of *Belle* enabling it to analyse 30 million positions in 3 minutes and search exhaustively to 8 or 9 ply in middlegame positions. It also had an extensive opening book. *Belle* won the 3rd World Computer Chess Championship in 1983, achieving a rating of over 2200 in 1980 and was the first program to receive a Master rating. It was for *Belle* that Thompson devised his first end game database, solving the KQKR problem and hence partially removing the perceived weakness at that time of computers in endgame positions.

In 1975 Hyatt commenced work on *Blitz* which was then entered in the ACM 1976 North American Computer Chess Championship. Initially *Blitz* was selective and relied on a local evaluation function to discard moves. However, the availability of the world's fastest supercomputer, the *Cray*, enabled the program appropriately renamed *Cray Blitz*, to use brute force and in 1981 it was searching approximately 3000 nodes per second and consistently achieving six ply searches. This rate of analysis was improved by the use of assembly language and the availability of the Cray XMP computer with multiprocessing facilities, allowing 20,000 - 30,000 nodes to be searched per second in 1983.

The third stage, aptly named *algorithmic* by Donskoy and Schaeffer (1990), extends onwards from the mid 1980s to the current time. This has seen some considerable activity in the refinement of the basic tree searching algorithms used (see later). It has also seen the emergence of personal computers on a scale originally unenvisaged. The widespread practice of incorporating vast opening books and more cd-roms for end games. This current phase has been most fruitful and seen a considerable increase in the playing strength of chess programs to the point where the top commercial software programs are generally recognised as being superior to humans at speed chess and in sharp tactical positions. Under strict tournament conditions the most highly rated player of all time Kasparov has now lost a game, but not the match, against *Deep Blue*.

Move Ordering Techniques

The previous section has outlined the historical development of computer chess. The transition to Shannon B type programs to Shannon A is not solely attributable to increased computing power. It also partly arose out of increased understanding of the alpha-beta algorithm which was subjected to deep analysis by Knuth and Moore (1975). Other techniques for increasing the efficiency of the search and pruning mechanisms also became more prominent at the beginning of the second era.

Producing a cutoff as soon as possible with the alpha-beta algorithm considerably reduces the size of the search tree. Consequently, move ordering is an important aspect of achieving maximum speed since, if we know the best move in a certain situation producing it early rather than late, will have beneficial results. The worst case is where the moves are examined in such an order as to produce no cutoffs, generating the maximum number of nodes to be analysed. This is the maximal tree which grows exponentially with depth, d, so that the number of nodes examined assuming a uniform branching factor, b, is given by b^d. The best situation is where the moves are all well-ordered and provide immediate cutoffs, producing the minimal tree which, although it also grows exponentially with depth, is very much reduced in size by the cutoffs. In between these two extremes we have game trees generated by chess programs which, initially are unordered, but become progressivley more ordered as the depth of search increases.

Algorithmic developments and various heuristics are moving the programs closer to the minimal tree. This progressive improvement in reaching closer and closer to the minimal tree is borne out by *Belle*, which it is estimated came within a factor of 2.2 of the minimal tree, *Phoenix* within a factor of 1.4 and *Zugzwang* within an impressive factor of 1.2 (Plaat 1996).

Iterative deepening is a useful device for allowing moves to be re-ordered prior to the depth being increased as well as providing a method for limiting search depth in response to time constraints. Originally introduced in 1969, it has become standard practice in brute force programs.

The killer heuristic is another move ordering technique using information from the alpha-beta pruning algorithm to facilitate it. Whenever a certain move causes a cutoff in response to

another move, then it is likely that it is able to refute other moves in a similar position. Such a move is a 'killer move' and can be stored in a killer table so that it can be tried first in order to cause another cutoff. Captures and mate threats provide the commonest form of such 'killer moves'. In this context the null move can provide a list of the most powerful moves for the opponent and assist in the move ordering process.

The killer heuristic was described in detail by Slate and Atkin (1977) although it had been used in earlier chess programs. However, the benefits of this technique are controversial with Hyatt claiming an 80% reduction while Gillolgly observed no such significant reduction in size (Plaat 1996).

A generalisation of the killer heuristic is the history heuristic introduced by Schaeffer (1989). This extends the ideas of the killer heuristic to include all the interior moves which are ordered on the basis of their cutoff history.

Tree Searching Algorithms

Chess programs have used till now almost exclusively one of a variety of related algorithms to search for the best move. Constant improvements in the search algorithm have been sought to reduce the number of nodes visited, and thus speed up the program.

We now briefly describe a number of variations upon the alpha-beta algorithm. These depend on having some knowledge, albeit imperfect, of what is the most likely principal variation, the series of moves down the tree consisting of the best play.

The usual call to alpha-beta at the root node has an infinite window. In aspiration search some initial estimate of a likely value is made together with a window, or band of values this likely value could lie in. Clearly the success of this algorithm depends on the initial estimates. If a value is returned which lies outside the aspiration search window then a re-search needs to be done with full alpha-beta width. There is thus a trade-off to be made between a greater degree of pruning using the limited window and the amount of re-search which has to be done.

Principal variation search (PVS) is a variation of alpha-beta search where the search at a particular level is ordered according to some evaluation function. The node which is found to be the most likely member of the principal variation is searched with a full width alpha-beta search, while others are searched with a minimal window where $\beta = \alpha + 1$.

With perfect move ordering, all moves outside those estimated to be part of the principal variation will be worse than those estimated to be on it. This is proved when the minimal window search fails low. Should this search fail high, a re-search has to be done with full alpha-beta width.

A refinement on PVS is NegaScout. This is very similar to PVS and incorporates the observation that the last two plies of a tree in a fail-soft search always return an exact value.

There have been a number of further derivations of the so-called zero window alpha-beta algorithm, which use a zero (actually unity) window at all nodes, unlike PVS which uses zero window only away from the principal variation. One of these is described by Plaat et al. (1996a) and has been given the name MTD(f). It relies on a large memory to hold a transposition table containing records of previous visits to nodes. We recall that in fact games like chess really take the form of graphs rather than trees, so that a particular board position might be visited from many different paths. The memory should be as large as feasible, since it serves to hold as many as possible of the results of visiting nodes.

Since the memory enhanced test algorithms operate with zero window, they, of necessity, need to do many re-searches. It is the use of memory to help speed up the production of results that makes these algorithms competitive in speed. Although Plaat et al. (1996a) claim their version is faster in practice than PVS, R. Hyatt, as he wrote in the rec.games. chess.computer Usenet newsgroup, in April 1997, still uses PVS for his Crafty chess program. The following pseudo-code from Plaat et al. (1996a) illustrates their MTD(f) version.

```
function MTDF(node-type,f,d)→ g
   g := f;
   upperbound := +∞;
   lowerbound := -∞;
   repeat
   if g = lowerbound then
     β := g + 1;
   else
     β := g;
     g:=ABWM(root,β-1,β, depth);
     if g < β then
            upperbound := g;
```

```
    else
            lowerbound := g;
until lowerbound >= upperbound;
  return g;
```

The algorithm works by calling ABWM a number of times with a zero window search. Each call returns a bound on the minimax value. These bounds are stored in *upperbound* and *lowerbound*, forming an interval around the true minimax value for that search depth. When the upper and lower bounds collide, the minimax value is found.

ABWM (AlphaBetaWithMemory) is a conventional alpha-beta algorithm which has extra lines to store and retrieve information into and out of a transposition table.

Interest has also been shown in the B* algorithm as well as those involving conspiracy numbers. These are more complex than algorithms involving alpha-beta searches, and seem to get closer to the Shannon B kind of algorithm where some expert knowledge of the game is programmed in to guide the machine.

Impact on AI and Philosophy

In the *pioneering* era, both computer chess and AI were emerging disciplines. At that early stage computer chess was regarded as a testbed for AI search and problem solving techniques. The early chess programs were primitive but are now much more sophisticated and have achieved spectacular results. However, as recognised by Donskoy and Schaeffer (1990), the dominantion of AI by computer chess has now changed and it is currently viewed as a small part of AI with its methods, identified primarily as brute force, regarded within the AI community as simplistic.

However, the impact of computer chess on AI can be gauged by the number of methods developed within the chess environment that are domain independent and applicable elsewhere. It is, for instance, undeniable that computer chess has made significant contributions to search techniques which find applications in other areas such as theorem proving and problem solving in general. It has also emphasised the development of special hardware to solve the chess problem, a viewpoint which is echoed in other areas of AI (Horacek 1993).

The computer scientists and AI enthusiasts that initiated the pioneering era of computer chess were partly motivated by a desire to investigate the processes of human reasoning. They argued that if it was possible for computers to solve the chess problem, it must be possible for computers to tackle other difficult problems relating to planning and applications to the economy. This preoccupation with intelligence formed the core of Turing's work, commencing with the creation of Turing machines and then the formulation of the Turing test which still remains of central importance in the AI debate today.

Turing himself worked in the period 1940-45 at Bletchley Park as an Enigma code-breaker. An indifferent chess player himself, he was, however, in the constant company of the best British chess players of that era, thus creating the environment and ideas for the final phase of his work. The Turing test states that if the responses of a computer are indistinguishable from those of a human it possesses intelligence. Consequently, it can be argued that a computer playing chess possesses intelligence as, in many cases, its moves are identical to those of chess experts. This is the strong AI viewpoint.

It was precisely to repudiate these claims of intelligence that Searle (1980) put forward his famous Chinese Room argument against strong AI. Chess programs produce their moves algorithmically. Suppose another human being having no prior knowledge of chess *per se*, performs the same algorithms, does he also understand the game? The weak AI viewpoint is that computers lack any such intelligence or understanding. This viewpoint is supported by Penrose (1995) who cites an example of a position in which Deep Thought blundered because of its lack of understanding of a particular position, despite being capable of deceiving us into thinking that it really has some chess intelligence.

Conclusion

The strength of the top chess playing programs is continually increasing. In this paper we have reviewed the main techniques that have enabled programmers to achieve such a spectacular increase in playing strength commencing with the earliest exploratory programs to the sophisticated software of today. During this period, we have seen significant increases in processing power which, despite the prevalent von Neumann architecture of today, is still increasing in power. Consequently, we foresee

the possibility of further increases in playing strength.

Future work will probably refine the traditional Shannon A type algorithms even more. For example, new avenues of research still exist on application-independent techniques of exploiting the fact that computer chess trees are really graphs (Plaat 1996b). However, there is a limit on the extent to which this can proceed. The consideration of Shannon B type algorithms is an area requiring further investigation and development.

References

Botvinnik, M.; Cherevik, D.; Vladimirov, V., and Vygodsky, V. 1994. Solving Shannon's Problem: Ways and Means, *Advances in Computer Chess* 7, van der Herik H. J, Hersberg I.S, and Uiterwijk, J.W.H.M.(eds) Maastricht, University of Limburg.

Donskoy, M.V and Schaeffer, J. 1990. Perspectives on Falling from Grace, in *Computers, Chess and Cognition*, Marsland, T and Schaeffer, J. (eds) New York, Springer Verlag.

Horacek, H. 1993. *Computer Chess, its Impact on Artificial Intelligence*, ICCA Journal 16(1):31-36.

Knuth, D.E, and Moore, R.W. 1975. An analysis of alpha-beta pruning. *Artificial Intelligence* 6(4):293-326.

Newborn, M.M. 1977. An analysis of alpha-beta pruning. *Artificial Intelligence* 6:137- 153.

Penrose, R. 1995. in *Shadows of the Mind*, London, Vintage.

Plaat, A.1996. Research Re:Search and Re-search. Ph.D.diss., Dept. of Computer Science, Erasmus University.

Plaat, A., Schaeffer, J., Pijls, W., and de Bruin, A. 1996a. Best-first fixed-depth minmax algorithms, *Artificial Intelligence* 87:255-293.

Plaat, A., Schaeffer, J., Pijls, W., and de Bruin, A. 1996b. Exploiting Graph Properties of Game Trees, *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregon.

Searle, J.R. 1980. Minds, brains amd programs. in *The behavioural and brain sciences*, Vol 3. Cambridge, Cambridge University Press.

Schaeffer, J.1989. The history heuristic and alpha-beta search enhancements in practice, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):1203-1212.

Shannon, C.E. 1950. Programming a Computer for Playing Chess. *Philosophical Magazine* 41(7): 256-275.

Simmon, H.A and Chase, W.G. 1973. Skill in Chess. *American Scientist*, 61, 482-488.

Slate, D and Atkin, L.1977. Chess 4.5:The North Western University Chess Program, in *Chess Skill in Man and Machine*, P.Frey(ed.), 82-118. New York, Springer Verlag.

Turing, A. M. 1953. Digital computers applied to games. in *Faster than Thought*, Bowden, B.V.(ed). London, Pitman.