To appear in Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems

The Challenge of Go as a Domain for AI Research: A Comparison Between Go and Chess

Jay Burmeister and Janet Wiles
Departments of Computer Science and Psychology
University of Queensland, QLD 4072 Australia
jay@cs.uq.edu.au
http://www.psy.uq.edu.au/~jay/research/research_page.html

Abstract

Go provides artificial intelligence (*AI*) and cognitive science researchers with an easily specified formal domain in which skills of human intelligence cannot be matched by currently known programming techniques. Go is a much more widely played game than chess (principally in Japan, Korea and China), yet it is not well known to AI and cognitive science researchers and our goal in this paper is to introduce some of the challenges of the game to the AI community in the form of a comparison with chess.

Go has been called a possible "task par excellence for AI" by Berliner [1] and we conclude that Go is a domain in which the development of new programming techniques is not only possible but is in fact necessary.

1.0 Introduction

Computer programs are challenging human performance in almost every level of game-playing endeavour: the world backgammon champion is a neural network program [7]. The performance of chess programs - originally the drosophila of research into search techniques in AI - is at the Grandmaster level: in 1994, the highest rated chess player in the world, Kasparov, was beaten by a computer chess program in a timed tournament, though he is yet to lose an untimed match. However, these top programs have long since ceased to inspire or teach AI and cognitive science researchers anything about incorporating the flexibility and skills of human cognition into computer programs.

A common misconception drawn from decades of chess research is that brute force techniques, utilising good search and evaluation algorithms, is sufficient to solve any problem once it has been formally specified. Go is a domain that contradicts this common misconception. It is easy to formally specify the rules of Go, however, all current programs fall short of human performance even to the level of a beginner-intermediate player.

Initially, we were under the impression that the difference in program performance between chess and Go is related to the relative branching factor and hence the relative complexity of chess and Go. Although it is true that Go has a much larger branching factor which has considerable consequences for programming (as shown in Table 1), we have come to realise that the differences between strategy and tactics in the two games is more important.

In chess, good evaluation functions for board positions can usually be estimated by tactical means alone - that is, searching through a tree of possible moves until a major change in positional strength is uncovered. In Go, tactics considerations involve fighting over specific groups of stones (defined in section 2.1) whilst strategy considerations involve building groups of stones that will have massive influence on the game at much later stages. A mastery of strategy and tactics is required for human players to play well in both chess and Go. In chess programs, tactical skills combined with long lookahead search techniques suffices to produce competent play. Such techniques have failed in Go programs, for reasons that we review below.

In section 2 of this paper we compare the features of chess and Go that may be relevant to understanding the cognitive challenge of Go and the difficulties involved in designing Go programs. We then present a brief history of research projects involving Go and commercial Go-programming efforts in section 3.

2.0 The Complexity of Chess and Go

In this section we briefly present the rules of Go, which are exceedingly simple to learn, and assume that the reader will be familiar with the rules of chess because of its higher profile in Western culture. We then compare the features of chess and Go.

2.1 Rules of Go

Go is a 2-player, perfect information game, played on a board which consists of a grid made by the intersection of horizontal and vertical lines. The size of the board is generally 19x19, however, 9x9 and 13x13 boards are also used, especially for children and beginners. Players alternate in placing black and white *stones* on the intersection points of the board (including edges and corners of the board). The aim of Go is to capture more territory than the opposing player by surrounding it with walls of stones.

The neighbours of any given point are the intersection points that are horizontally or vertically adjacent to it (rooks move in chess). Empty points that neighbour a stone are called its *liberties*. One or more stones of the same colour can be *linked* into *strings* by being orthogonally connected to each other. Any stone or string which has no liberties is captured and is removed from the board. This is the only instance in which stones, once placed, are moved. Each captured stone is worth one point of territory in scoring at the end of the game.

The repetition of a previous board position is prohibited by the ko rule in order to avoid infinite loops. Ko situations arise when a stone which captures a single opponents stone can itself be captured by the opponent replacing the captured stone with a new stone on the next move.

Although Go has relatively few rules, there are many subtle aspects to playing and scoring that arise as consequences of these rules. Certain configurations of strings cannot be captured and are said to be *alive*. Conversely, a string which cannot be saved from capture regardless of what moves its owner makes is considered to be *dead*. Dead stones need not be removed during the game, but can be taken off as prisoners at the end. A stone cannot be played on a point which has no liberties, unless it captures an opponent's stones, creating liberties for itself in the process.Groups of stones are the main perceptual units concerning a player throughout the game. The most important attribute of a group at the end of the game is its safety i.e., whether it is dead or alive.

2.2 A Comparison of the Features of Chess and Go

This section contains expanded comparisons of the twelve features of chess and Go from Table 1.

1. There are fewer types of pieces in Go than chess (in chess there are 6 types of pieces, whereas in Go each player has only one type, called a stone). However, the board size is much greater in Go (8x8 squares in chess compared to a 19x19 grid in Go).

2. The size of the board, and the relative freedom in the placement of stones mean that there are many more moves made in a typical game of Go than in chess (about 80 in chess compared to about 300 in Go).

3. Stones can be placed anywhere on the board, making for a very large average branching factor in the selection of each move (estimated at around 200), whereas pieces in chess are constrained to a much smaller set of legal moves (branching factor of about 35). The diversity of chess pieces is exploited to reduce the branching factor in chess programs in a way that is not possible in Go (see discussion of evaluation functions in point 7 below). The branching factor also plays a role in the different stages of both games (beginning, middle and end). In the opening stage of chess, there are many well-known openings, and these are often followed for up to 10 moves. In Go, the number of sensible openings is much larger, and set openings are rarely followed for more than about 3 moves. However, there are sequences of moves in local skirmishes, known as joseki, that reflect optimal play (for both sides) in tactical battles in the corners. Skill in the use of joseki libraries lies in selecting the right joseki to optimize interactions with stones outside the region of interest or diverge from the known sequence to serve other interests.

Table 1: A Comparison the Features of Chess and Go

	Features	Chess	Go
1	board size	8 x 8 squares	19 x 19 grid
2	# moves per game	~80	~300
3	branching factor	small (~35)	large (~200)
4	end of game and scoring	checkmate (simple definition - quick to identify)	counting territory (consensus by players - hard to identify)
5	long range effects	pieces can move long distances (e.g., queens, rooks, bishops)	stones do not move, patterns of stones have long range effects (e.g., ladders; life & death)
6	state of board	changes rapidly as pieces move	mostly changes incrementally (except for captures)
7	evaluation of board positions	good correlation with number and quality of pieces on board	poor correlation with number of stones on board or territory surrounded
8	programming approaches used	amenable to tree searches with good evaluation criteria	too many branches for brute force search, pruning is difficult due to lack of good evaluation measures
9	human lookahead	typically up to 10 moves	even beginners read up to 60 moves (deep but narrow searches e.g., ladders)
10	horizon effect	grandmaster level	beginner level (e.g., ladders)
11	human grouping processes	hierarchical grouping [5]	stones belong to many groups simultaneously [6]
12	handicap system	none	good handicap system

4. Both games can be won by resignation of the opponent, or by an outright win - checkmate in chess, or surrounding more territory and taking more prisoners in Go. In chess, the end of a game is easy to determine: checkmate is simply defined in terms of the position and threats to the king. In Go, a game is finished when both players pass, although a game can be resumed should there be a disagreement during scoring. It is frequently difficult for beginners to know when the end of a game has been reached. Beginners typically prolong play beyond the point where experts would stop, and their difficulties are echoed by Go programs, which also have difficulty deciding when to end a game. The natural end of the game occurs when playing additional stones reduces a player's score, either by filling in already secured territory, or giving unnecessary prisoners to the opponent. Go programs are often inefficient in both these ways. There are several different rule systems for scoring in Go: they are all similar, based on calculating the sum of territory surrounded plus prisoners taken during the game for each player, and taking the difference between these scores (a typical margin in a professional game would be less than five points).

5. In both chess and Go, pieces can have long range effects. In chess the effect is directly a result of some pieces' ability to move many squares (e.g, queens, rooks, bishops). In Go, a stone once placed on a grid point does not move (although it can be captured and removed from the board). However, a group (or pattern of stones) does have long range effects since it can play a role in a capturing race, or can affect the life and death struggle of another group. For example, a stone played in the path of a ladder (a group of stones involved in a certain type of capturing race) can change the potential to link two stones later in the game - even though the stones are on the other side of the board (for a computational approach to addressing this problem see Burmeister, Wiles & Purchase [2]).

6. The state of the board changes rapidly in chess, as pieces move positions. In Go, the board is only gradually changing, as stones are added to the existing configurations. This gradual change compensates significantly for the greater memory load imposed by the larger board. It also allows accurate readout of board positions later in the game - it is even possible for beginners to readout ladders up to 60 moves ahead (a deep but narrow search). The only time the physical state of the board changes significantly is when a group is captured, and the stones are removed from the board. Groups worth as much as 30 points are often won or lost in a game as trade-offs for other groups, even though the final difference in scores may be just 2 points.

7. Evaluation of board positions (in expert human play) typically reflects both tactical and strategic factors in both chess and Go. However, in chess, there is a good correlation between the likelihood of winning from any stage of the game and the number and quality of pieces on each side. Thus in computer chess programs, strategic factors have not been essential in evaluating board position. In Go, there is no strong correlation between winning a tactical struggle over a group, and winning the game, since each tactical struggle over a group requires moves that are not contributing to another group. Early in the game, players strive to achieve influence on the board, rather than directly taking territory. In fact, taking territory at the start of the game can indicate an over-concentrated position, that will not be effective in containing the opponent's territorial moves much later in the game. Algorithmic approaches to measuring the influence of a position are a standard part of most Go programs but there are no methods to confirm their efficacy, except for the strength of the program itself which is usually very weak.

8. For all the reasons discussed above, programming approaches to chess are amenable to tree searches, with good evaluation criteria. Such approaches have not succeeded in Go, because the branching factor is too large for brute force search techniques, and pruning is not a viable option without good evaluation measures.

9. In tactical evaluation, there appear to be substantial differences between the search requirements of chess and Go. Even though at the absolute beginner level, players in both games might start by mentally searching only a few moves ahead, in Go, even beginners will soon learn that there exist special patterns (e.g., ladders) for which there is only one sensible move. Ladders consist of long sequences of forced moves (up to 60 moves) are easily mentally traced until they reach a boundary such as the edge of the board. Partial search of the ladder gives no indication of its outcome if there are stones in its path. Such deep and narrow searches do not occur in chess. In both chess and Go, experts do search through many alternative possible moves, however, in chess, broad searches are rarely more than 10 moves deep, and chess programs that can lookahead 12 moves have an excellent chance of determining near optimal play. Because of the presence of ladders in Go, search trees cannot be uniformly both broad and deep, and because of the problems with evaluation of board positions (see point 7 above) sensible pruning techniques are difficult to devise.

10. The horizon effect in a heuristic search occurs where a search to a specific depth elicits an evaluation that is radically different from the evaluation that would be obtained if the search was a little deeper. This effect is common in sacrifice moves in chess. It is not a phenomenon in beginner games but can be a problem at the Grandmaster level. In Go, the horizon effect is seen in assessment of even beginner level play (e.g., ladders as described in point 9 above).

11. Chess was used in the early 1970s to study human grouping processes, and Chase and Simon [5] showed that expert chess players view the board in terms of hierarchical groups of stones. In trying to replicate the results with Go players, Reitman [6] found a completely different structure, in which stones are seen as part of many intersecting groups. Identifying the eventual group to which stones belong is critical in determining their safety, however, from Reitman's research, it appears that no clean assignments can be made until the end stage of the game. Such results indicate sources of potential problems for computer Go programs that assign stones into hierarchical groups to estimate features such as influence.

12. One major difference between playing chess and Go lies in the standards of the opponents. In chess, a similar standard is required for an even game. In Go, there is a very effective handicap system, in which a weaker player starts with a predefined number of stones (usually 2-9) placed at influential points on the board. Thus the greater skill of one player is offset by the material advantage of the weaker one. In such contests (and most Go games are of this type) neither player can assume an infallible opponent, for the weaker player tries to simplify the game and play safely, whereas the stronger player must take more risks and exploit the weakness of the other player. Although the final score of a Go game is often used as an indication of the relative strengths of the players, the important factor is winning, by however small a margin, rather than taking chances that may increase the margin. Thus, estimating the opponent's level of skill is a critical aspect of playing Go. It is common to rank the strength of computer Go programs according to their first game against a human of a given level. In subsequent games, the human player will often overwhelm the program that had won previously, because the human will learn the program's weaknesses, but the program does not change.

3.0 Go Research Efforts and Commercial Go Programs

Research into aspects of Go began as early as the 1960s and Go has been used as a domain in a wide variety of research including pattern recognition, heuristic tree analysis, representative search, machine learning, software engineering, human perception and cognition, memory, and planning (an in depth treatment of research involving Go and commercial Go programs can be found in [4]).

Other than early academic work on Go, most research work using Go as a domain is not intended to produce working Go programs. Since traditional AI techniques do not produce good results in Go, new techniques can be developed without concern that they are unnecessarily complicated methods for achieving what can already be realised in a simpler way (as they would be viewed in the computer chess field).

There are many resources available on the Internet that would be of benefit to Go related research projects. These resources include archive sites, a news group, a mailing list for discussing aspects of Go programming, computer Go competitions, and bibliographies containing computer Go related research (for a full description of the resources available and information on how to access them see [3] and [4]).

The inducement of over US$1 million for the first Go program that can play at the level of Shodan (i.e., a competent game, but nowhere near the peak of human performance) has helped to increase the amount of effort expended on programming Go, especially commercial efforts. However, the best computer Go programs currently play at about the level of someone who has read a few introductory Go books and played one to two games per week for a year compared to chess programs which play at Grandmaster level.

4.0 Conclusions

Our conclusions from the comparison of the features of chess and Go is that the performance of Go programs will not match those of chess programs if they are programmed using the same techniques used in chess. Thus, Go is a domain that provides AI and cognitive science researchers with an opportunity to develop new approaches and programming techniques. Furthermore, Go is a domain which requires such changes to enable research progress to be achieved.

Acknowledgements

We thank the Department of Computer Science at the University of Queensland for the facilities to carry out this project and for providing a Summer Research Associate position to the first author and the University of Queensland for subsequently providing a University of Queensland Postgraduate Research Scholarship to the first author. We would also like to thank Andrew Hussey for proof reading the penultimate draft of this paper.

References

[1] Berliner, H. J. A chronology of computer chess and its literature. *Artificial Intelligence*, **10**, pages 201 - 214, 1978.

[2] Burmeister, J., Wiles, J. and Purchase, H. On relating local and global factors: a case study from the game of Go. In the *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, 1995.
(Available on the Internet at http://www.psy.uq.edu.au/~jay/papers/l-g.factors.ps.Z).

[3] Burmeister, J. and Wiles, J. Accessing Go and Computer Go Resources on the Internet. To appear in the *Proceedings of the Second Game Programming Workshop in Japan*, September 1995.
(Available on the Internet at http://www.psy.uq.edu.au/~jay/papers/comp-go.internet.ps.Z).

[4] Burmeister, J. and Wiles, J. An Introduction to the Computer Go Field and Associated Internet Resources. Technical Report 339, Department of Computer Science, University of Queensland, 1995.
(Available on the Internet at http://www.psy.uq.edu.au/~jay/go/CS-TR-339.html).

[5] Chase, W. G. and Simon, H. A. Perception in chess. *Cognitive Psychology*, **4**, pages 55 - 81, 1973.

[6] Reitman, J. S. Skilled perception in Go: Deducing memory structures from inter-response times. *Cognitive Psychology*, **8**, pages 336 - 356, 1976.

[7] Tesauro, G. Practical issues in temporal difference learning.In J. E. Moody, S. J. Hanson and R. P. Lippmann, editors, Advances in Neural Information Processing Systems 4, pages 259 - 266. Morgan Kaufmann Publishers, San Mateo, 1992.