Studies in Human and Computer Go: Assessing the Game of Go as a Research Domain for Cognitive Science

Jay Madison Burmeister BSc BA (Hons)

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electrical Engineering
and
School of Psychology

The University of Queensland
Australia

October 2000

Statement of Originality

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text, and the material has not been submitted, either in whole or in part, for a degree at this or any other university[1].

Jay Burmeister

October 2000

1 I gratefully acknowledge the author of the Statement of Originality whose identity is unfortunately shrouded in the mists of time.

Abstract

This thesis assesses the game of Go as a research domain for Cognitive Science by investigating some of the research issues within the domain of Go, and in particular, by assessing Go as a research domain for both Artificial Intelligence and Cognitive Psychology. The following general assessment questions were used:

- *To what extent does the domain of Go facilitate the investigation of research questions?*
- *What type of research issues can be investigated within the domain of Go?*
- *Does investigation within the domain of Go give rise to further interesting research questions?*
- *Do useful generalisations arise from research conducted within the domain of Go? What types of generalisations can be made?*
- *In what ways does the domain of Go facilitate an exchange of ideas between disciplines?*

The methodology used to answer these questions involved case studies of Computer Go programs and experimental studies in the memory field.

In Computer Go the research issues addressed were related to knowledge, search and their interaction. The AI techniques used in Computer Chess are generally not successful in Computer Go. It is commonly believed that the large branching factor in Go is the reason why brute-force search techniques used in Computer Chess do not work in Computer Go. However, the main reason for the failure is that static evaluation of Go positions is not possible due to the necessity of life and death analysis (although the large branching factor is certainly a contributing factor to the failure). Case studies of two Go programs (Many Faces of Go and Go4^{++}) as well as an overview of the current top Go programs are presented Together they show how the techniques developed by the Computer Go community address the difficulties associated with programming a game that resists the brute-force search approach.

The memory research issues addressed the use of Go for studying the factors that affect memory for Go moves and the extent to which inference impacts on memory performance for sequences of Go moves. Three experimental studies comprising experiments and case studies were conducted using beginner, experienced, and master Go players as participants. The experimental studies were instrumental in the development of a new experimental paradigm for testing memory performance for sequences of moves from a Go game, and in particular, the development of two new experimental tasks, namely the sequential pennies-guessing task and the sequential construction task. An extension of this paradigm led to the development of a potential technique for learning Go involving the use of interactive memory tasks. The results of the experimental studies indicated that inference impacts to some extent on memory performance, that memory for sequences of moves is related to Go skill, and that Go is a better domain than chess for investigating memory for sequences.

Following the assessment of Go as a research domain for Artificial Intelligence and Cognitive Psychology, this thesis concludes that:

- the game of Go provides a good research domain for Artificial Intelligence, facilitating investigation into areas such as planning, problem solving, pattern recognition, and opponent modelling; and
- the game of Go provides a good research domain for Cognitive Psychology, facilitating investigation into areas such as memory, implicit learning, pattern recognition, perceptual learning, problem solving, and attention.

Based on these conclusions, this thesis finds that:

The game of Go provides a good research domain for Cognitive Science.

List of Publications

Burmeister, J. and Wiles, J. (1995) Accessing Go and Computer Go resources on the Internet. In H. Matsubara, editor, *Proceedings of the Second Game Programming Workshop in Japan '95*, pages 75 - 84, Kanagawa. Computer Shogi Association.

Burmeister, J., Wiles, J. and Purchase, H. (1995) The integration of cognitive knowledge into perceptual representations in Computer Go. In H. Matsubara, editor, *Proceedings of the Second Game Programming Workshop in Japan '95*, pages 85 - 94, Kanagawa. Computer Shogi Association.

Burmeister, J. and Wiles, J. (1995) The challenge of Go as a domain: A comparison between Go and chess. In *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, pages 181 - 186, Perth. IEEE Western Australia Section. (refereed)

Burmeister, J., Wiles, J. and Purchase, H. (1995) On relating local and global factors: A case study from the game of Go. In *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, pages 187 - 192, Perth. IEEE Western Australia Section. (refereed)

Burmeister, J. and Wiles, J. (1996) The use of inferential information in remembering Go positions. In H. Matsubara, editor, *Proceedings of the Third Game Programming Workshop in Japan '96*, pages 56 - 65, Kanagawa. Computer Shogi Association.

Burmeister, J., Wiles, J. and Purchase, H. (1999) The integration of cognitive knowledge into a perceptual representation: Lessons from human and Computer Go. In J. Wiles and T. Dartnall, editors, *Perspectives on Cognitive Science: Theories, Experiments and Foundations, Volume II*, pages 239 - 257. Ablex, Connecticut. (refereed)

Burmeister, J. and Wiles, J. (1999) AI techniques used in Computer Go. In R. Heath, B. Hayes, A. Heathcote and C. Hooker, editors, *Proceedings of the Fourth Conference of the Australasian Cognitive Science Society*. University of Newcastle. (refereed)

Burmeister, J., Saito, Y, Yoshikawa, A. and Wiles, J. (2000) Memory performance of master Go players. In H. J. van den Herik and H. Iida, editors, *Games in AI Research*, pages 271 - 286. Universiteit Maastricht. (refereed)

Acknowledgements

I am grateful to my wife, Sharyn, and my daughter, Casey, for providing me with their love and support throughout my PhD. I dedicate this thesis to them in honour of their sacrifices which made its completion possible.

I am grateful to Janet Wiles, my supervisor, mentor, and friend, who deserves much of the credit for this project and thesis. Janet sparked my initial interest in research, and taught me how to conduct and communicate research. Janet's guidance, wisdom, support, and friendship has made it possible for me to achieve my potential.

I would like to thank my associate supervisor, Mike Humphreys, for his help with the Cognitive Psychology side of this project. My thanks also go to my auditor, Roger Duke, for making report time more a pleasure than a pressure, and for being an understanding and supportive boss during the final stages of writing my thesis. I am grateful to Helen Purchase for her Honours supervision during Janet's absence and for her contributions to the early publications that resulted from my Honours work and developed into my PhD.

I would like to thank Robert Thomas and Jennifer Downie for their help, without which I most likely would not have been able to finish this project.

I am grateful to Andrew Hussey for his assistance with writing the software used to conduct the experiments.

This thesis document has benefited through the critical analysis and comments of Scott Bolland, Kerry Chalmers, Simon Dennis, Jennifer Hallinan, Mike Humphreys, Andrew Hussey, Mike Norris, Guy Smith, and Brad Tonkes, for which I am grateful.

I was very fortunate to be able to spend 3 months conducting research in Japan at NTT. I am grateful to my colleagues, Yasuki Saito and Atsushi Yoshikawa, for sharing their experience, knowledge, and facilities with me, and for providing the opportunity to meet members of the Go and Computer Go communities. I also appreciate Yaski-san, Yoshikawa-san, and the all the Kame-G members for making my stay not only a rewarding research visit, but also a wonderful time that I will never forget. Special thanks to Makiko Ishikawa for the friendly way in which she helped me to cope with my domestic and administrative concerns, and Takehiko Ohno for his technical support. I am also grateful to NTT for the financial support the provided for me during my stay in Japan.

The Go and Computer Go communities have been very good to me. I would particularly like to thank David Fotland, Mick Reiss, Martin Müller, Ken Chen, Tristan Cazenave, and Zhixing Chen for sharing the details of their Go programs and assisting me to communicate them. I would like to thank Masahiro Okasaki of the Go Culture Study Group for making the 9x9 professional games available to me and Peter Robinson for providing me with the Shusaku book. I would also like to thank Darren Cook, Chihiro Mizuuchi, Hidekazu Takagi and Yasutoshi Yasuda for the help they gave me while I was in Japan.

Thanks to my office mates, Rafael Brander, Rachael Gibson, Peta Wyeth, and Leesa Murray, and my lunch mates, Andrew Hussey, Anthony MacDonald, and David Leadbetter, who provided me with encouragement and attentive ears to pour out my troubles and share my triumphs. Thanks to the members of the Cognitive Science Research Group, the Department of Computer Science and Electrical Engineering, and the School of Psychology for their helpful discussions, and for providing a stimulating environment in which to conduct my research. Thanks also to the administrative and technical staff of the Department of Computer Science and Electrical Engineering, and the School of Psychology for their help.

I am thankful to my examiners, Tom Downs, Neil Charness and the anonymous examiner, for their helpful comments and practical suggestions for improving this thesis.

I am grateful to the University of Queensland, the Department of Computer Science and Electrical Engineering, and the School of Psychology for providing me with a scholarship, travel funding, employment, and resources.

Contents

Section I The Game and the Goal

1 Introduction 1

1.1 Goal 4
1.2 Method 5
1.3 Statement of Thesis 7
1.4 Overview 8

2 Overview of the Game of Go 11

2.1 The Game of Go 11
2.2 Rules 12
2.3 Linking Strings into Groups and Armies 17
2.4 Advanced Concepts 19
2.5 Feature Comparison Between Go and Chess 25

Section II The Computer Go Field 31

3 Early Work in the Computer Go Field 33

3.1 The Early Programs 33
3.2 Discussion 44

4 State-of-the-Art in the Computer Go Field: Case Studies and Analyses of Current Go Programs 47

4.1 Case Study 1: Many Face of Go 48
4.2 Case Study 2: Go4++ 57
4.3 Game-Tree Search and AI Techniques in Competitive Go Programs 62
4.4 Summary 65

5 Research Issues in Computer Go **67**

5.1 Discussion 67
5.2 Future Directions 75
5.3 Conclusions 76

Section III Studies in Memory and Learning for Go Sequences **79**

6 The Cognitive Psychology of Go: Literature Review and Experimental Paradigms **81**

6.1 Seminal Cognitive Studies in Chess 82
6.2 Cognitive Studies Using Go as a Research Domain 88
6.3 Discussion 95

7 Study 1: Sequential Pennies-guessing Task **99**

7.1 The Sequential Pennies-guessing Task 99
7.2 Experiment 7.1: Sequential Pennies-guessing Task 100
7.3 Results 105
7.4 Discussion 106

8 Study 2: Sequential Construction Task **109**

8.1 The Sequential Construction Task 110
8.2 Experiment 8.1: First Sequential Construction Task 111
8.3 Experiment 8.2: Second Sequential Construction Task 118
8.4 Summary: Experiments 8.1 and 8.2 124
8.5 Case Studies: Memory Performance of Master Go Players 124
8.6 General Discussion: Studies 1 and 2 131

9 Study 3: Learning Go Using an Interactive Memory Task **133**

9.1 Experiment 9.1: Learning Go 134
9.2 Discussion 139

10 Go as a Domain for Studying Sequential Aspects of Memory and Learning 143

10.1 Summary of Results 143
10.2 Discussion 144
10.3 Conclusions 145

Section IV Go Provides a Good Research Domain for Cognitive Science

11 Conclusions, Contributions and Future Work 149

11.1 Synergy: Exchanging Ideas Between the Computer Go and Cognitive Psychology Fields 149
11.2 Conclusion: Go is a Good Research Domain for Cognitive Science 153
11.3 Summary of the Contribution of this Thesis 154
11.4 Future Work 156

References 159

Appendices 165

A Common Go Terms and Their English Equivalents 165

B The Predecessors to The Many Faces of Go: G2 and Cosmos 167

B.1 G2 167
B.2 Cosmos 171
B.3 The Many Faces of Go 174

C The Experimental Software 175

List of Figures

1.1 Layout of this thesis document 9

2.1 Stone Liberties 12

2.2 Stone Capture 13

2.3 Strings 13

2.4 String Liberties 14

2.5 String Capture 14

2.6 Suicide 14

2.7 Ko 15

2.8 Territory 16

2.9 Examples of Links 18

2.10 Cutting Strings 18

2.11 Groups and Armies 19

2.12 Single Eye 20

2.13 Capturing a String with One Eye 20

2.14 Strings with Two Eyes Cannot be Captured 21

2.15 Life and Death 21

2.16 Seki 22

2.17 Ko Threats and Ko Fight 23

2.18 Ladder 24

2.19 Ladder-breaker Stone 24

3.1 Zobrist's Influence Function 34

3.2 Ryder's influence function. 38

7.1 Sequential presentation in the episodic condition 102

7.2 The entire position 103

7.3 Feedback: wrong stone selected. 104

7.4 Feedback: correct stone selected 104

7.5 Sequential Pennies-guessing Task: experienced and beginner results 106

8.1 Initial stone presented in the inferential condition. 113

8.2 Feedback: wrong position selected 114

8.3 Sequential Construction Task: Experiment 8.1 results 115

8.4 Sequential Construction Task: Experiment 8.2 results 120

8.5 Comparison of Experiments 8.1 and 8.2 results: experienced participants . . . 121

8.6 Comparison of Experiments 8.1 and 8.2 results: beginner participants 122

8.7 8d episodic trial results 126

8.8 8d and 6d(1) episodic results 127

8.9 Master participants' episodic (500 milliseconds) and inferential results. 128

9.1 Feedback: wrong position selected 137

9.2 Results: S1 139

9.3 Results: S2 140

Chapter 1

Introduction

The long-term progress of research in a field is affected by the interaction between research questions and research domains. Investigating research questions requires research domains; investigation within research domains gives rise to further research questions. Thus one of the prerequisites for research progress is the provision of good research domains.

The characteristics of good research domains are that they enable research questions to be investigated, that investigations conducted within them give rise to further interesting research questions, and ideally, that they allow generalisations to be made beyond the domain itself. In an interdisciplinary field such as Cognitive Science good research domains facilitate research in different disciplines and enable an exchange of ideas between disciplines.

Cognitive Science

Cognitive Science is an interdisciplinary field concerned with the study of the mental processes that underlie human cognitive abilities. The disciplines involved in Cognitive Science include Philosophy, Linguistics, Neuroscience, Artificial Intelligence, and Cognitive Psychology. Two of these disciplines are of particular interest in this thesis. Artificial Intelligence (*AI*) is concerned with constructing artificial means of performing tasks, that, if performed by humans, would require intelligence. Cognitive Psychology is concerned with the phenomena and underlying mechanisms of human intelligence and cognitive abilities.

Games as Cognitive Science Research Domains

Games such as chess have long been accepted as research domains in AI and Cognitive Psychology. In AI, games can be formally specified and provide non-trivial domains without all the problems associated with real world complexity. In Cognitive Psychology, games provide actual human domains (rather than contrived artificial domains) in which there are experts who have mastered the complexity of the domain. Chess in particular has proven to be a good research domain. The metaphor of the *Drosophila fruit-fly* was invoked by Simon and Chase in connection with the use of chess in Psychology:

> *As genetics needs its model organisms, its* Drosophila *and* Neurospora*, so psychology needs standard task environments around which knowledge and understanding can cumulate. Chess has proved to be an excellent model environment for this purpose.* Simon & Chase (1973, p. 394)

The Drosophila metaphor has also been used in AI circles with chess being referred to as *the Drosophila of AI* by pioneer AI researcher McCarthy (1990).

Chess as a Research Domain for Cognitive Psychology and AI

It is instructive to briefly overview the utility of chess as a Cognitive Science research domain. As noted by Charness (1992), although chess research has not been central to Cognitive Science, it has made a significant impact on the field. In Cognitive Psychology, according to Charness (1992), chess has been used as a means to study perception, pattern recognition, encoding, memory, and problem solving. Other uses of chess listed by Charness include: developing theories about the architecture of the human cognitive system; a domain for cognitive modelling; an empirical domain to study chunking; studying the nature of expertise in general and chess expertise in particular; and contributing to an understanding of chess playing itself. In AI, chess has primarily been used to study search and evaluation processes, leading to the development of search techniques such as minimax and alpha-beta pruning.

Results from psychological research into chess have shown that chess players rely less on searching than on a thorough knowledge of chess patterns and an ability to access and use them effectively. In the early stages of the Computer Chess field, AI researchers tried to incorporate as much knowledge as possible into their chess playing systems. However, the performance of such systems did not keep pace with the performance of brute-force systems that could more effectively exploit search rather than knowledge. Thus, although chess programs now play chess well compared to human chess masters, they have ceased to contribute to the psychological understanding of human cognitive abilities. A divergence in the direction of chess research in AI and Cognitive Psychology resulted from the success of brute-force AI techniques. The synergy between AI and Cognitive Psychology chess research has not continued, particularly with respect to the influence of Cognitive Psychology on the construction of chess programs.

The Game of Go

A potential new *Drosophila* for AI according to McCarthy (1990), has emerged in the game of Go. In chess, programs based on brute-force AI techniques perform very well (the chess program Deep Blue beat the reigning world champion Kasparov in a best-of-six-game series in 1997). However, the brute-force AI techniques used in Computer Chess do not work in Computer Go:

> *... even if a full-width search program were to become World Chess Champion, such an approach cannot possibly work for GO, and this game may have to replace chess as the* task par excellence *for AI.* Berliner (1978, p. 206)

It is acknowledged within the Computer Go field that the current state-of-the-art Go programs play at around the level of someone who has played a few games a week for a year and has studied some introductory Go books. According to McCarthy (1990), entirely different investigations from those conducted in chess research would be necessary in Computer Go.

Current Go programs, just like human Go players, rely more heavily on knowledge than on search to play Go well. Typically, Go programs limit the number of suggested moves for which search-trees are generated rather than performing full-width search. The generation of good moves to explore requires the effective use of Go knowledge. Since Go programs rely more heavily on knowledge than chess programs, an understanding of how Go knowledge is acquired, organised, and used by humans may provide valuable lessons which lead to improvements not only in Go programs but also to a better understanding of how to use knowledge effectively in AI in general. Thus, unlike Computer Chess research, Computer Go research may benefit from psychological investigations of Go players.

Cognitive Psychology may benefit both from the investigation of the various cognitive aspects of Go and from Computer Go itself. Go provides an excellent domain for investigating many algorithms required for cognitive processes such as perception and pattern recognition, learning, attention, evaluation, opponent modelling, problem solving, expertise, planning, knowledge representation, and memory. Computer Go, unlike Computer Chess, may contribute to the psychological understanding of human cognitive abilities, and thus, a potential synergy between AI and Cognitive Psychology Go research may exist and prove fruitful for both disciplines and therefore for Cognitive Science.

A Note on the Usage of the Term “Research Domain”

The term “research domain” can be used in a broad or a narrow sense in a similar way to the term “literature”. In the broad sense, the term “research domain” may refer to an entire field of study just as the term “literature” can be used to refer to the entire literature of a field. In the narrow sense, the term “research domain” may refer to a particular narrowly defined domain in which research is conducted (e.g., the game of chess); similarly, the description of work conducted in a narrowly defined research domain may be referred to as the “literature”. Thus the term “research domain” just like the term “literature” describes a continuum. In this thesis, the term “research domain” will be used in its narrowest sense unless otherwise indicated, that is, “research domain” will refer to a particular narrowly defined domain in which research is conducted.

1.1 Goal

The goal of this thesis is to assess the game of Go as a research domain for Cognitive Science by investigating the research issues within the domain of Go. The assessment will be with respect to the assessment questions described below in section 1.1.1. Since Cognitive Science is an interdisciplinary field, the goal of the thesis will be refined into sub-goals related to AI and Cognitive Psychology in section 1.1.2. Section 1.2 outlines the methodology used in this thesis and the assessment questions for the AI and Cognitive Psychology sub-goals. The thesis of this thesis is stated in section 1.3.

1.1.1 Assessment Questions

The characteristics of a good research domain (*RD*) as identified above can be expressed as general questions:

RD1 *To what extent does the research domain facilitate the investigation of research questions?*

RD2 *What type of research issues are able to be investigated within the domain?*

RD3 *Does investigation within the research domain give rise to further interesting research questions?*

RD4 *Can useful generalisations be made beyond the research domain itself? And if so, what types of generalisations can be made?*

The assessment questions RD1 - 4 can be adapted to Go (*GO*) so that they can be used as questions for assessing Go as a research domain:

> ***GO1*** *To what extent does the domain of Go facilitate the investigation of research questions?*
>
> ***G02*** *What type of research issues can be investigated within the domain of Go?*
>
> ***GO3*** *Does investigation within the domain of Go give rise to further interesting research questions?*
>
> ***GO4*** *Do useful generalisations arise from research conducted within the domain of Go? What types of generalisations can be made?*

Due to the interdisciplinary nature of Cognitive Science (*CS*), a further assessment question is necessary in assessing Go as a research domain:

> ***CS1*** *In what ways does the domain of Go facilitate an exchange of ideas between disciplines, that is, between AI and Cognitive Psychology?*

1.1.2 Sub-goals

To address question CS1, the goal of the thesis can be further refined into two sub-goals:

- to assess Go as a research domain for AI; and
- to assess Go as a research domain for Cognitive Psychology.

To perform the assessments associated with the sub-goals, the assessment questions GO1 - 4 will be refined and made specific for both AI (*AI1 - 4*) and Cognitive Psychology (*CP1 - 4*) in sections 1.2.1 and 1.2.2 respectively.

1.2 Method

The first part of question GO1 and question GO2 (and their respective AI and Cognitive Psychology correlates) will be addressed by concrete example. The methodology used in this thesis will be case studies in the Computer Go field and experimental studies in the memory field. Question CS1 will be addressed by considering past, present, and future interactions between the Computer Go and the Cognitive Psychology Go communities.

1.2.1 Assessing Go as an AI Research Domain

Go will be assessed as a research domain for AI by investigating the Computer Go field.

AI Assessment Questions

The specific assessment questions for assessing Go as a research domain for AI (*AI*) are:

> ***AI1*** *To what extent does the domain of Computer Go facilitate the investigation of knowledge and search, and their interaction?*
>
> ***AI2*** *What type of AI research issues can be investigated in the domain of Computer Go?*
>
> ***AI3*** *Do interesting research questions arise from investigations of search, knowledge, and their interaction within the domain of Computer Go?*
>
> ***AI4*** *Do generalisations useful to AI arise from research conducted within the domain of Computer Go? What types of generalisations can be made beyond the domain of Computer Go?*

The specific research issues comprising the questions AI1 and AI2 will be justified in Section II. The questions AI1 and AI2 will be addressed by a survey of the Computer Go field, including a survey of the literature and detailed case studies of current Go programs. Question AI3 will be addressed by considering whether the issues of the research domain exist elsewhere in AI. Question AI4 will be addressed by considering whether generalisations from within the research domain may have an impact outside the domain itself.

1.2.2 Assessing Go as a Cognitive Psychology Research Domain

Go will be assessed as a research domain for Cognitive Psychology primarily through a series of experimental memory studies.

Cognitive Psychology Assessment Questions

The specific assessment questions for assessing Go as a research domain for Cognitive Psychology (*CP*) are:

> ***CP1*** *To what extent does the domain of Go facilitate the investigation of memory for moves? To what extent does the domain of Go facilitate the investigation of the impact of inference on memory performance?*
>
> ***CP2*** *What type of Cognitive Psychology research issues can be investigated in the*

domain of Go?

CP3 *Do interesting research questions arise from investigations of memory for moves within the domain of Go? Do interesting research questions arise from investigations of the impact of inference on memory performance within the domain of Go?*

CP4 *Do generalisations useful to Cognitive Psychology arise from research conducted within the domain of Go? What types of generalisations can be made beyond the domain of Go?*

The specific research issues comprising the questions CP1 and CP2 will be justified in Section III. Questions CP1 and CP2 will be addressed by analysis of a series of memory experiments and case studies. Question CP3 will be addressed by considering whether the issues of the research domain exist elsewhere in Cognitive Psychology. Question CP4 will be addressed by considering whether generalisations from within the research domain may have an impact outside the domain itself.

1.2.3 Scope

This thesis builds on AI and Cognitive Psychology research that has used Go as a research domain. Detailed literature surveys are provided for the Computer Go field (Chapters 3 and 4) and cognitive studies of human Go performance (Chapter 4).

Although references to chess in AI and Cognitive Psychology research will be made where relevant, a comprehensive survey of games research in general and of chess research in AI and Cognitive Psychology in particular, will not be provided in this thesis.

1.3 Statement of Thesis

The conclusions resulting from the assessments associated with the sub-goals are that:

- the game of Go provides a good research domain for Artificial Intelligence facilitating investigation into the areas of planning, problem solving, pattern recognition, and opponent modelling; and
- the game of Go provides a good research domain for Cognitive Psychology facilitating

investigation into the areas of memory, implicit learning, pattern recognition, perceptual learning, problem solving, and attention.

Based on those conclusions, the thesis of this work can be stated as:

The game of Go provides a good research domain for Cognitive Science.

1.4 Overview

This thesis is organised as three streams: Cognitive Science, AI, and Cognitive Psychology (see Figure 1.1). The Cognitive Science stream is supported by the AI and Cognitive Psychology streams that essentially run in parallel with minimal interaction between them.

This thesis consists of 4 sections. Section I is concerned with Go and its assessment as a Cognitive Science research domain. Section II is concerned with the sub-goal of assessing Go as a research domain for AI. Section III is concerned with the sub-goal of assessing Go as a research domain for Cognitive Psychology. Section IV asserts the thesis of this work, namely that the game of Go provides a good research domain for Cognitive Science, by drawing together the conclusions of sections II and III.

Section I: The Game and the Goal

Chapter 2 provides an overview of the game of Go including the rules and descriptions of aspects of the game that are relevant to following chapters.

Section II: The Computer Go Field

Chapter 3 surveys the early work in the Computer Go field, assesses game-tree search in Go, and discusses the challenges of Computer Go for AI. Chapter 4 surveys the contemporary Computer Go field and provides in depth case studies of two current Go programs based on issues identified in Chapter 3. Section II closes in Chapter 5 by arguing that Go provides a good research domain for AI.

Section III: Studies in Memory and Learning for Go Sequences

Chapter 6 introduces Section III and provides a survey of seminal memory studies in chess and cognitive studies in Go. This is followed by a series of experimental memory studies that

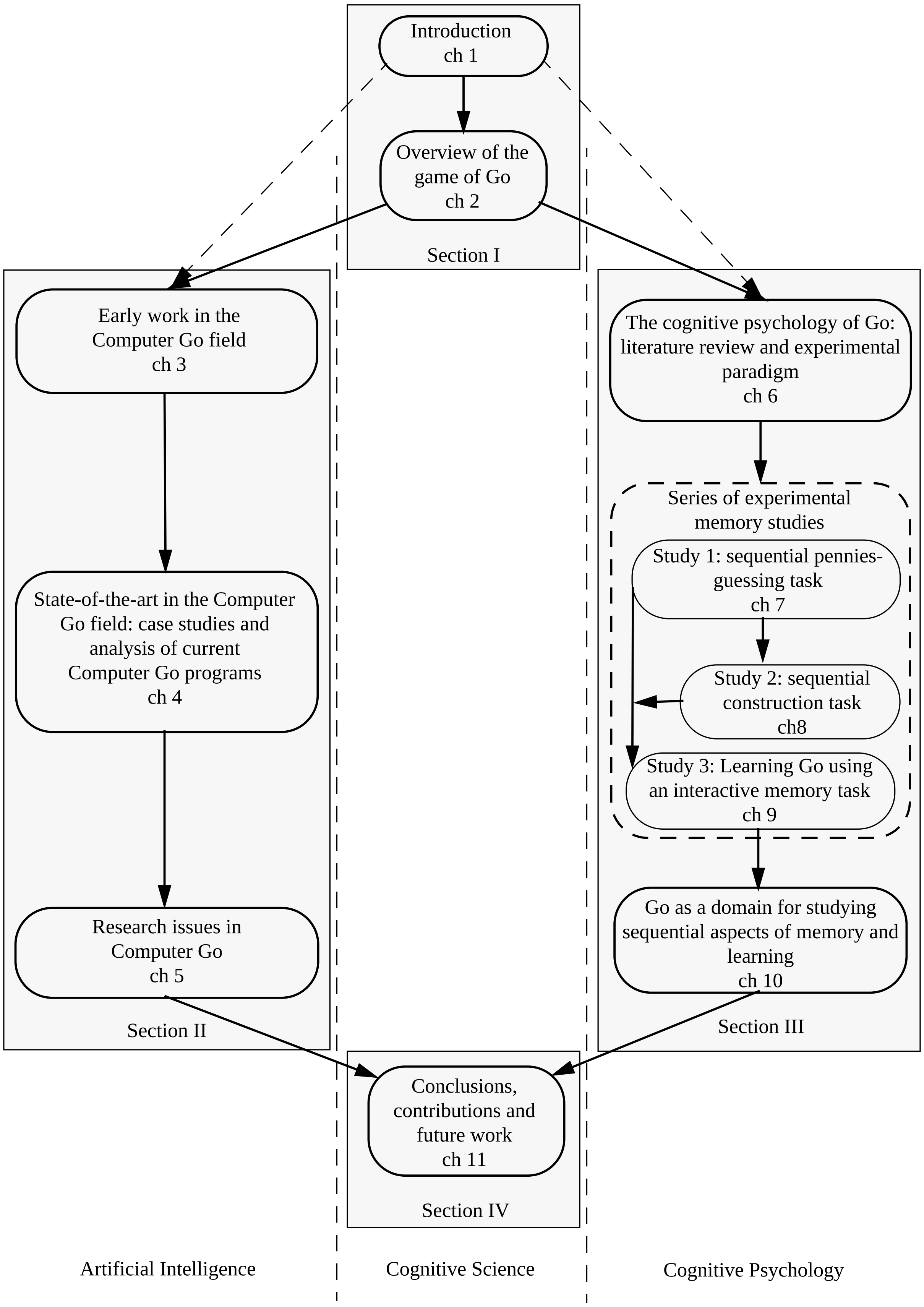

Introduction
ch 1
Overview of the game of Go
ch 2
Section I
Early work in the Computer Go field
ch 3
State-of-the-art in the Computer Go field: case studies and analysis of current Computer Go programs
ch 4
Research issues in Computer Go
ch 5
Section II
The cognitive psychology of Go: literature review and experimental paradigm
ch 6
Series of experimental memory studies
Study 1: sequential pennies-guessing task
ch 7
Study 2: sequential construction task
ch8
Study 3: Learning Go using an interactive memory task
ch 9
Go as a domain for studying sequential aspects of memory and learning
ch 10
Section III
Conclusions, contributions and future work
ch 11
Section IV
Artificial Intelligence
Cognitive Science
Cognitive Psychology

Figure 1.1: Layout of this thesis document comprises three streams related to Cognitive Science, AI, and Cognitive Psychology. The AI and Cognitive Psychology streams run in parallel with minimal interaction, supporting the Cognitive Science stream.

concretely demonstrate the utility of Go as a research domain. Chapters 7 and 8 describe experiments based on tasks (sequential pennies-guessing and sequential construction respectively) in which the participants reconstructed the order of play (i.e., the sequence of moves). Implicit learning was observed in those experiments and Chapter 9 reports on case studies involving learning Go through the use of interactive memory tasks. Section III closes in Chapter 10 by arguing that Go provides a good research domain for Cognitive Psychology.

Section IV: Go Provides a Good Research Domain for Cognitive Science

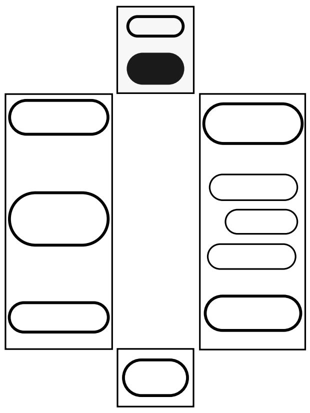
Chapter 11 draws together the conclusions from Sections II and III (i.e., that the game of Go provides a good research domain for AI and Cognitive Psychology) to support the thesis that the game of Go provides a good research domain for Cognitive Science.

Chapter 2

Overview of the Game of Go

Introduction

This chapter provides an overview of the game of Go, including the rules and some of the key concepts. Readers familiar with the rules and tactics of Go need not read section 2.1 and 2.2. Section 2.3 sets out the terminology used in this thesis for concepts such as groups and armies. Features of Go and chess are compared in section 2.5. A list of Go terms together with a brief explanation of their meaning is provided in Appendix A.



2.1 The Game of Go

Go is a two player, perfect information board game in which chance plays no part. It is an ancient game which originated between 2500 and 4000 years ago and enjoys a similar status in Japan, China, Taiwan and Korea[1], to chess in Western countries.

Go is played on a board that consists of a grid made by the intersection of horizontal and vertical lines. The size of the board is generally 19 x 19, although 9 x 9 and 13 x 13 sized boards are also used for playing shorter games. Intersection points are connected along the horizontal and vertical lines such that the neighbours of any given point are the intersection points that are horizontally and vertically adjacent to it. The two players alternate (with the Black player moving first) in placing black and white *stones* on the intersection points of the board, including edges and corners. The aim of Go is to capture more territory (see section 2.2.4) and prisoners than the other player.

A game of 19x19 Go is usually around 250-300 moves long and is generally described in three

1. Other names for Go are “Igo” in Japan, “Wei Qi” in China, and “Baduk” in Korea.

phases: the opening or *fuseki*[2], the mid-game, and the end-game or *yose*. Opening move sequences are called *joseki*. They are similar to opening books in chess and are typically based around open corners.

2.2 Rules

Go is a very simple game with few rules. There are several variations on the rules but the differences are generally insignificant and are usually to do with scoring and *ko* (see section 2.2.3). This section is not intended to give an exhaustive description of tournament rules, but rather to set out what would be considered sufficient rules to play a “club” game using Japanese rules and scoring.

2.2.1 Liberties and Capture

Empty points that neighbour a stone are called its *liberties* (points marked L in Figure 2.1). Any stone that has no liberties is *captured* and removed from the board (white stone in Figure 2.2). Once placed on the board, stones do not move (other than when they are captured and removed from the board).

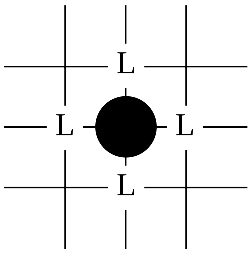


Figure 2.1: Stone Liberties. The points marked L are liberties of the black stone.

2.2.2 Strings and String Capture

Stones of the same colour can be joined into *strings* by being horizontally or vertically connected to each other (see Figure 2.3). Single stones can also be described as strings (e.g., black stone marked 1 in Figure 2.3). Instead of the stones in a string having individual liberties,

2. In this thesis (following widespread Western usage), Japanese terms for Go concepts will be introduced where appropriate.

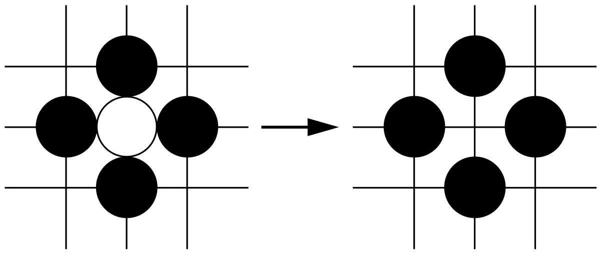


Figure 2.2: Stone Capture. The white stone has no liberties and is thus captured and removed from the board.

the liberties of the constituent stones of the string become the liberties of the string (points marked L in Figure 2.4). To be captured, a string must be surrounded by opponent stones such that no stone in the string has any liberties (black stones in Figure 2.5). When a player fills the penultimate liberty of a stone or string, the string is said to be in *atari* (similar to being in check in chess). A player cannot commit *suicide* by placing a stone in a position that leads to its immediate capture (point marked X in Figure 2.6).

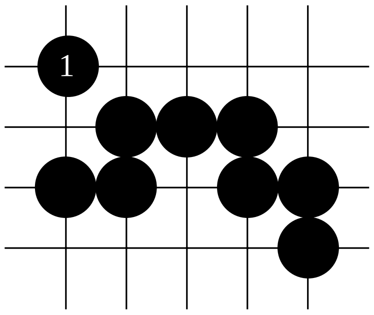


Figure 2.3: Strings.The eight black stones (excluding the stone at 1) form a string.

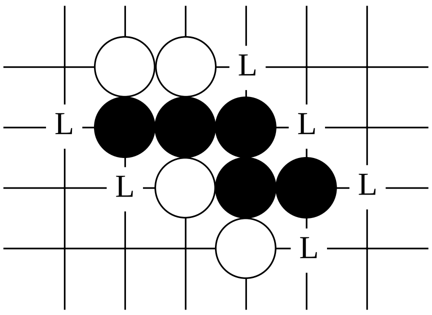


Figure 2.4: String Liberties. The points marked L are liberties of the black string.

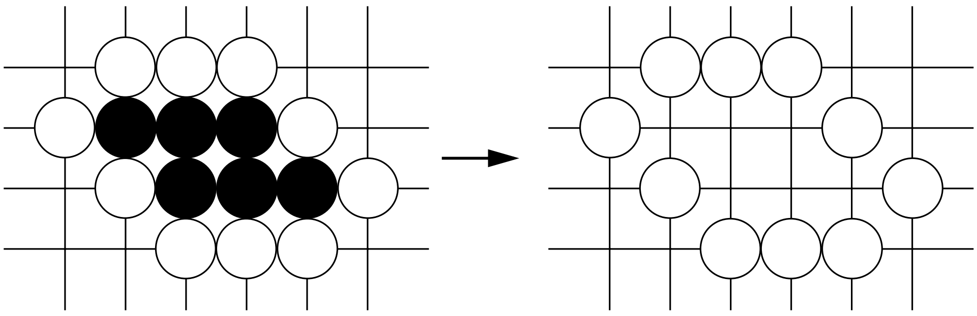


Figure 2.5: String Capture. The black string has no liberties and is thus captured and removed from the board.

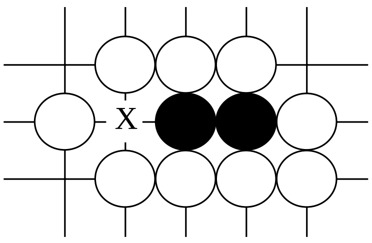


Figure 2.6: Suicide. Black cannot play at X since the whole black string would be captured.

2.2.3 Repetition of Board Positions - Ko

To stop endless cycles occurring, a player cannot play a stone that causes a previous board position to be repeated. In general, the only situation in which a cycle can arise is called a *ko*. In Figure 2.7, a white stone at 1 is captured by a black stone at 2 which could itself be captured by a white stone being played at 1 again. Such a situation is prohibited by the rule against repetition.

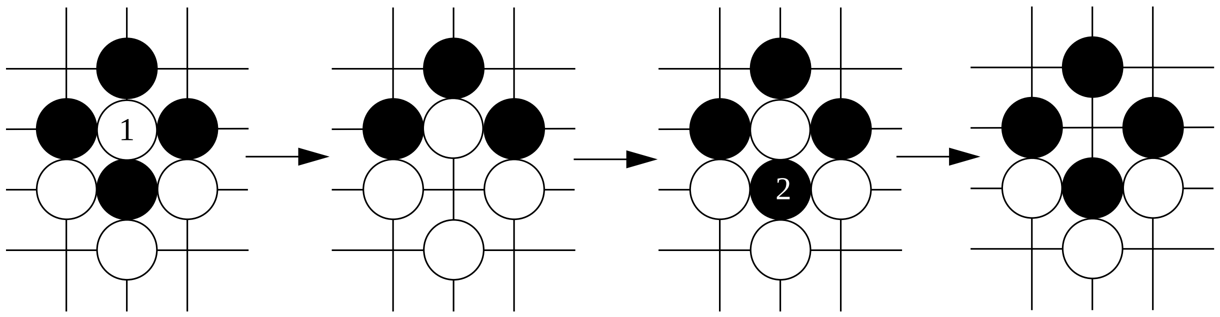


Figure 2.7: Ko. After White plays a stone at 1, the black stone is captured and removed from the board. Black can not immediately play a stone at 2 to capture the white stone since this would result in an endless repetition of board positions.

2.2.4 Territory

Territory is determined at the end of the game (see next section) and consists of the empty intersection points that are surrounded by a player’s stones (points marked B in Figure 2.8). Determining territory also involves removing the opponent’s *dead* stones (i.e., stones that are dead but have not been captured as described in section 2.4.2) which count as prisoners.

2.2.5 Winning the Game

A player may *pass* at any time and the game is over when both players pass consecutively. The players calculate their scores by adding the number of prisoners they have captured (comprising those captured during the game and the dead stones removed once scoring has commenced) to the amount of territory they have surrounded. The player with the largest score wins. Disputes during scoring (generally over whether certain stones are dead or alive) can be settled by an exchange of a stone and the continuation of the game until the matter is settled.

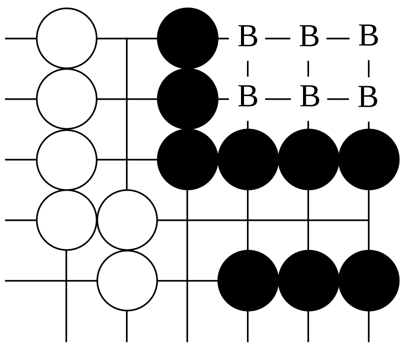


Figure 2.8: Territory. The points marked B are considered Black territory and count as 1 point each at the end of the game for scoring purposes.

2.2.6 The Go Handicap and Ranking Systems

Go has a sophisticated handicap and ranking system. Players are ranked according to their ability, with a complete novice being ranked at approximately 30 *kyu*. After playing 10 to 15 games, a player's rank would probably be approximately 20 kyu. After reaching 1 kyu, further improvement would result in a rank of 1 *dan* or first-degree master. Amateur ranks continue up to 8 dan. Thus, kyu ranks count down and dan ranks count up. Professional ranks start at the equivalent of 9 dan amateur and extend from 1 dan to 9 dan.

Increasing effort is required to improve to the next rank as a player improves. To rise from 20 kyu to 10 kyu would require playing 1 or 2 games a week for approximately a year and possibly reading a few books on Go tactics and strategy. The differentiation between professional ranks is also smaller than that between amateur ranks: four professional ranks are about the same as one amateur rank (Bate, 1997). Many years of full-time play are required to reach professional ranking.

To provide balance for both weak and strong players, the weak player is given handicap stones at the start of the game. In the Chinese rules, the handicap stones are placed at the weaker player's discretion whereas in the Japanese rules the handicap stones are placed on set points called *hoshi* points. The weaker player is given a number of handicap stones equal to the difference between the player's rankings. For instance, a 10 kyu player would give 5 handicap stones to a 15 kyu player. The relative value of handicap stones diminish as the board size increases: one handicap stone on a 9x9 board is worth two on a 13x13 board and four on a 19x19 board.

Playing first without handicap is worth an advantage of about 5 points of territory. It is customary for the weaker player to play first using black stones. If both players are ranked equally, the White player is given a 5 point bonus or *komi*. In tournaments the komi is usually 5.5 points so as to avoid ties.

2.3 Linking Strings into Groups and Armies

Strings can be linked into larger structures called groups and armies[3] which are the main focus during a game of Go.

2.3.1 Links/Connections

The only *link* between stones that is recognized in the rules of Go is that between horizontally and vertically adjacent stones, called *nobi*. In practice however, there are several virtual links (or *connections*) that are recognized by experienced Go players (see Figure 2.9). The *ikken tobi* link is one that can be physically connected as long as black plays next. The *kosumi* link can be made as long as black plays next or each of the connecting points are empty if white plays first. Of the other links, physical connection depends on the context of the surrounding stones. If two strings are denied a physical connection, they are said to be *cut* (see Figure 2.10). *Connectivity* is a computational measure of the likelihood that two strings can be connected.

2.3.2 Groups

A *group* is composed of strings of one colour that are in close proximity to each other such as strings A1 and A2, and strings B1 to B4 in Figure 2.11. Being in close proximity means that they can be securely linked by one of the links illustrated in Figure 2.9. For the B group, string B1 is linked to string B2 by an ogeima link; string B2 is linked to string B3 by an ikken tobi link; and string B3 is linked to string B4 by a kogeima link. Groups are the main perceptual units concerning the player throughout the game. The most important attributes of a group are whether it is *alive* (see section 2.4.2), and whether it can create two *eyes* (see section 2.4.1) or

3. Other members of the Computer Go community, for example, Chen (1989) and Müller (1995), use the alternative terms “block”, “chain”, and “group” for “string”, “group”, and “army” respectively.

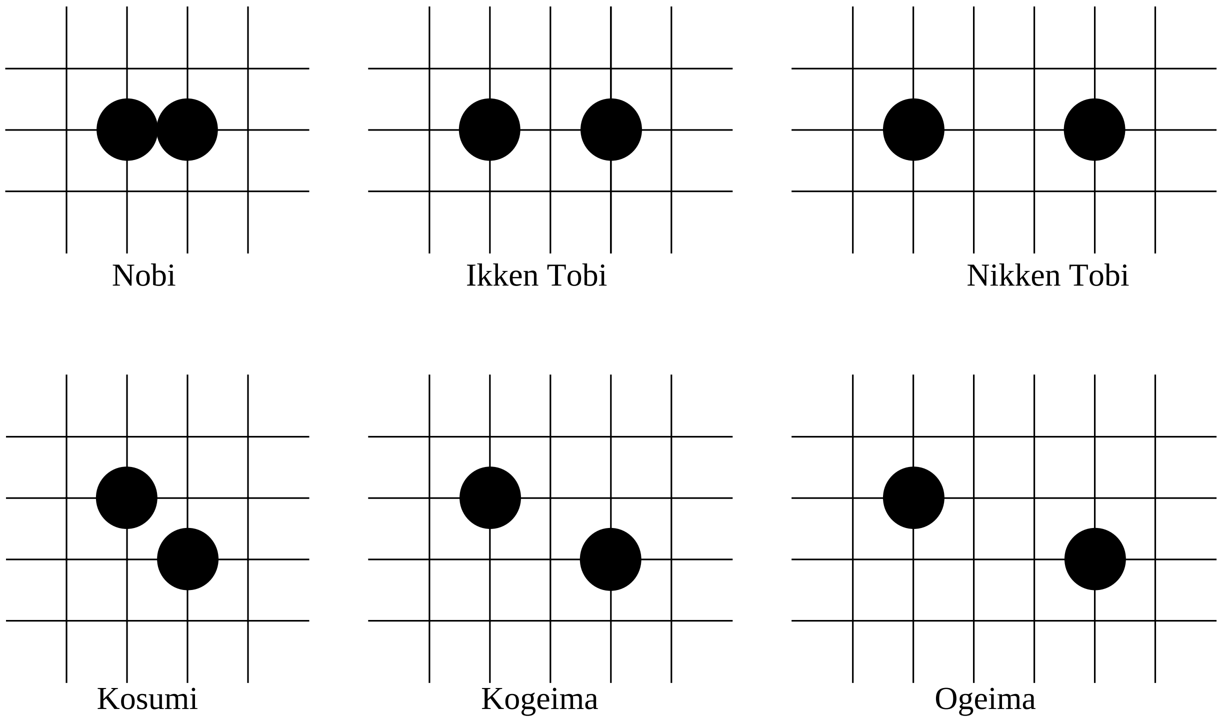


Figure 2.9: Examples of Links. Note that alternate English names for the links are: one point jump (ikken tobi); two point jump (nikken); diagonal (kosumi); knight's jump (kogeima); and large knight's move (ogeima).

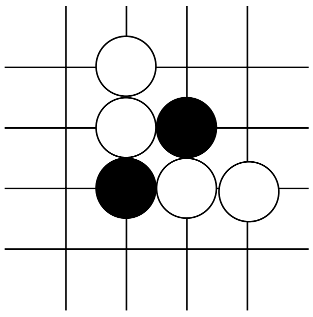


Figure 2.10: Cutting Strings. The two black stones have been cut by the white strings.

connect to a group with two eyes to ensure survival.

2.3.3 Armies

An *army* is a loose federation of groups of one colour that are in close proximity to each other. In Figure 2.11, the two groups A (comprised of strings A1 and A2) and B (comprised of strings B1 to B4) form one army. Being in close proximity means that there are no intervening enemy

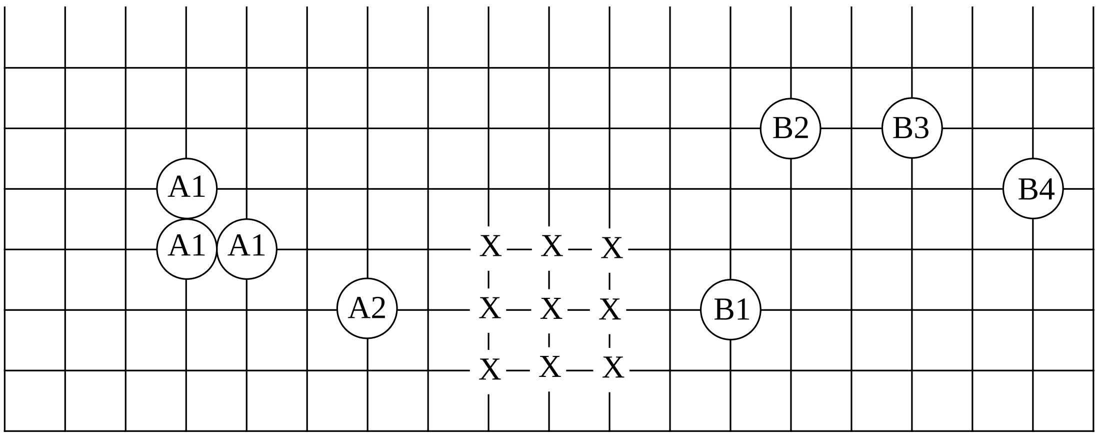


Figure 2.11: Groups and Armies. A group is composed of strings of one colour that are in close proximity to each other such as A1 and A2, and B1 to B4. An army is a loose federation of groups of one colour that are in close proximity to each other (i.e. the two groups above form an army). The army would be made more secure by playing at one or more of the points marked X.

strings between the groups. Although armies cover a large territory, they are not as secure for defending territory as groups. It is thus important for players to convert armies into groups whilst stopping their opponents from doing the same. Converting an army into groups implies playing stones between strings of separate groups, so that they can be linked together as described above (points marked X in Figure 2.11).

2.4 Advanced Concepts

Playing Go well requires an understanding of many advanced concepts that are beyond the scope of this thesis to describe. However, there are several concepts that are essential for an adequate understanding of Go. These concepts include *eyes*, *life and death*, *ko threats*, *sente* and *gote*, and *ladders*. The concept of *influence* is included even though it is not necessarily an essential Go concept but because it had an impact on early Computer Go programs.

2.4.1 Eyes

Eyes are formed when a string surrounds an empty intersection point (point marked E in Figure 2.12). Strings that contain a single eye can be captured by first being surrounded and then the eye being filled (see Figure 2.13). Filling the eye does not violate the suicide rule since the opponent’s captured stones are first removed, creating liberties for the last stone played. A string that has two eyes (points marked E in Figure 2.14) cannot be captured since playing in the first eye would be subject to the suicide rule.

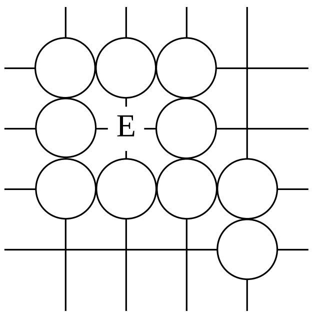


Figure 2.12: Single Eye. The string contains a single eye (the point marked E).

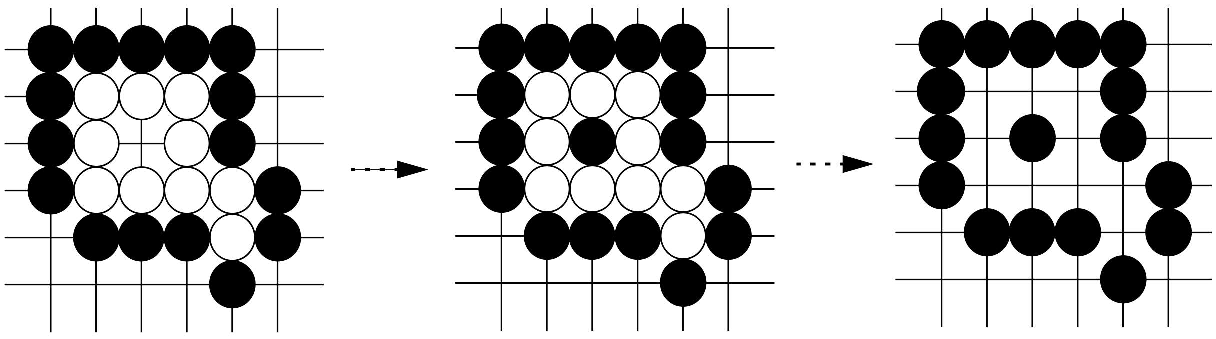


Figure 2.13: Capturing a String with One Eye. The string is first surrounded and then the eye is filled. The white stones are removed because they have no liberties, creating liberties for the black stone that filled the eye. Thus, playing in the eye does not violate the suicide rule. Note that this figure describes one move.

2.4.2 Life and Death

Strings that have two eyes are said to be *alive*. In general, a string is alive if it cannot be captured even if the opponent moves first (see black group on the left of Figure 2.15). A string is considered to be *dead* if there is no way to stop it from being captured even if its owner moves first (see black group in the middle of Figure 2.15). A string is *unsettled* if its life and death status depends on which player moves next (see black group on the right of Figure 2.15).

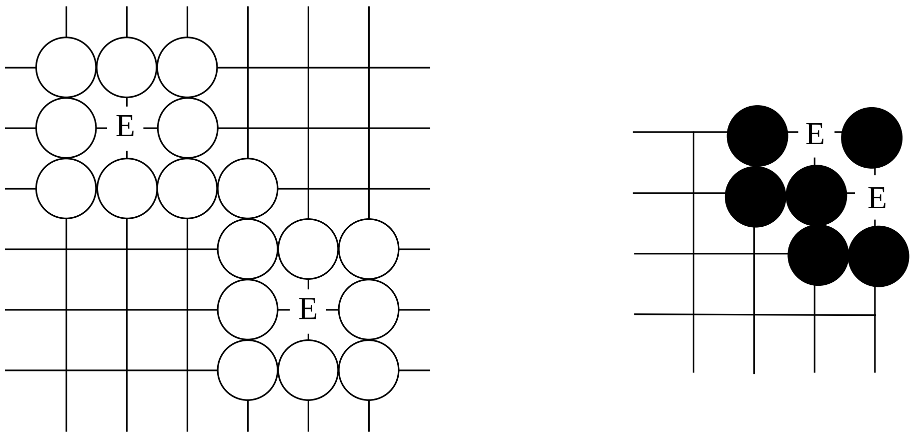


Figure 2.14: Strings with Two Eyes Cannot be Captured. The points marked E are eyes. Since the string has two eyes, it cannot be captured.

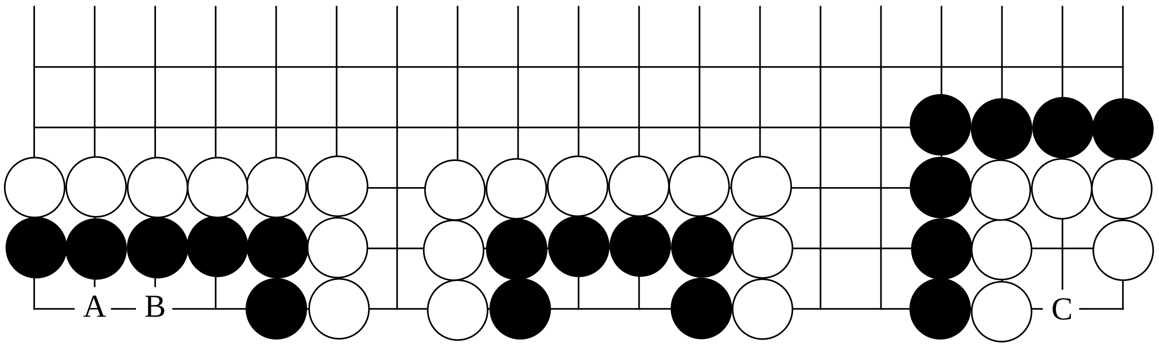


Figure 2.15: Life and Death. The black group on the left is alive because it can make 2 eyes regardless of where White plays: if White plays at A, Black plays at B making 2 eyes (and vice versa). The black string in the middle is dead since Black cannot make 2 eyes. The white string on the right is unsettled: whichever player plays next “wins”. If White plays at C then the string has 2 eyes. However, if Black plays at C, White cannot make 2 eyes and the string is dead.

2.4.2.1 Seki

It is possible for strings to be in a configuration in which they have mutual life (called *seki*). In such a situation, neither player can play to kill the opponent’s string because in doing so, the player would place their own string in atari (see Figure 2.16).

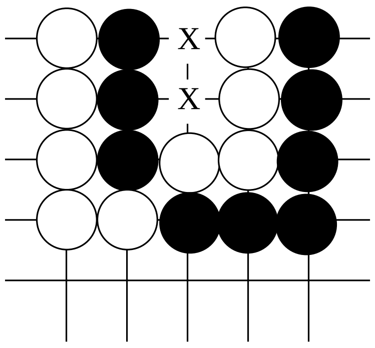


Figure 2.16: Seki. If either player plays at one of the points marked X to place the opponent's string in atari, their own string will be in atari and can be captured. Thus, neither player will play at the points marked X and the inner strings are mutually alive, that is, in seki.

2.4.3 Ko Fights and Ko Threats

A ko situation can lead to a *ko fight* (see Figure 2.17) in which the player who loses a stone in the ko (white stone at 4 captured by black at 1) threatens an opponent stone elsewhere (at 2) and then recaptures the opponent stone in the ko (at 4) after the opponent responds to the threat (at 3).

2.4.4 Ladders

Ladders are a capturing race between a group that is almost enclosed, and a surrounding group. A kogeima link is an implicit link between two stones (see Figure 2.9) that are not adjacent and have no intervening stones between them. Depending on context, a kogeima link can be considered to be a virtual link in most situations. An attempt by an opponent to stop a physical joining of the two stones by cutting them may result in a ladder (left side of Figure 2.18). The ladder begins with White trying to stop Black from converting the virtual kogeima link into a concrete link between the stones K1 and K2. Initially, White plays at 1 and Black responds at 2. White stops Black making the connection by playing at 3. When Black plays at 4, the white stone at 1 is threatened with capture. To defend this stone, White plays at 5. Black once again threatens capture by playing at 6. A cycle of threat and response continues and results in a ladder being formed. Black wins the ladder if a board edge is encountered (right side of Figure 2.18).

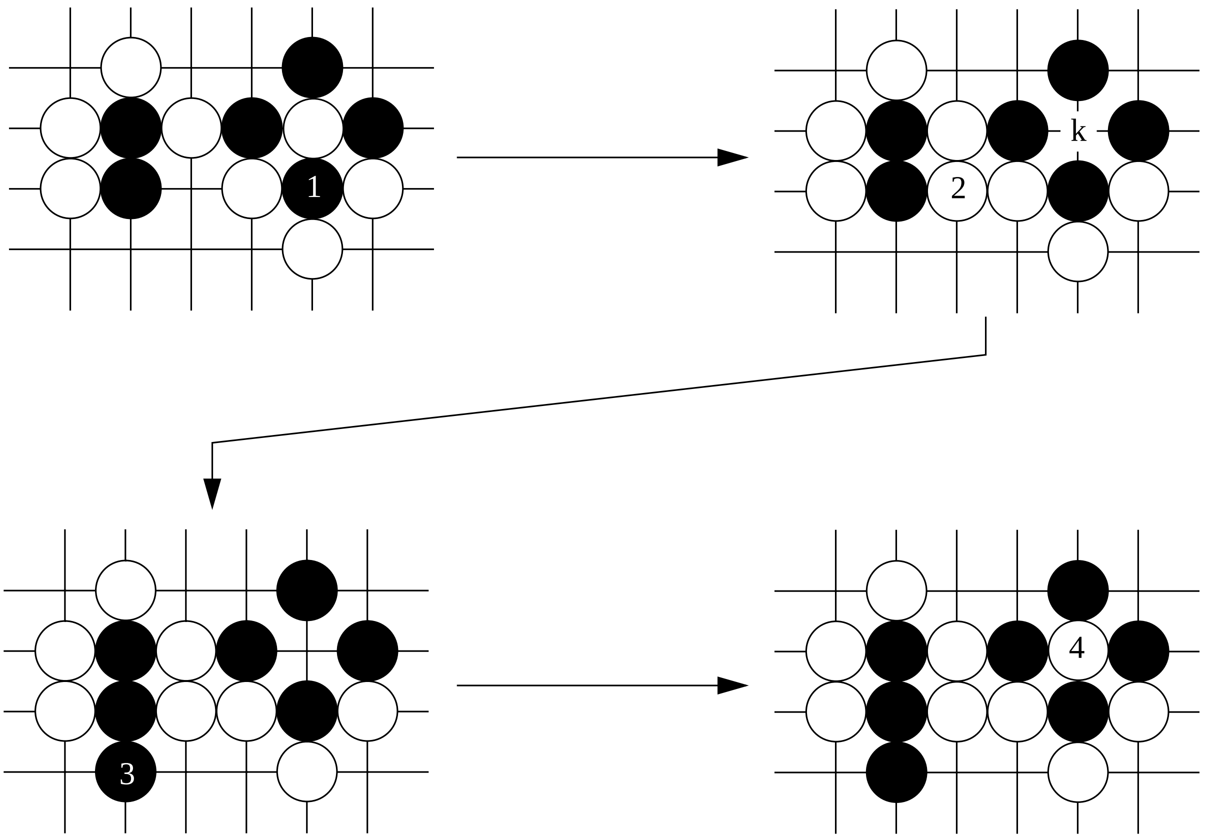


Figure 2.17: Ko Threats and Ko Fight. A white stone is captured by the black stone played at 1. White can not immediately play at k because of the ko rule. White plays a ko threat at 2, threatening to capture the black string. Black responds at 3 to save the string. White may now play at 4 and recapture the black stone played at 1 since doing so will not repeat a previous board position. This sequence of moves is a ko fight.

2.4.4.1 Ladder-breakers

The points on which the black and white stones (1 through 18) are played are called the *ladder points*. If before trying to cut the kogeima link, the White player already had a white stone on any of the ladder points, the white string would escape capture. Such a stone is called a *ladder-breaker*. As an example, consider the situation where White has a ladder-breaker already on a ladder point prior to White playing at 1 (marked as 0 in Figure 2.19). The ladder would initially unfold as before, but White would escape capture by playing at 11, threatening to capture both of the black stones at 4 and 8, and cutting Black's kogeima link in the process. Thus, it can be seen that the context of the ladder points needs to be taken into consideration in evaluating the strength and security of a kogeima link.

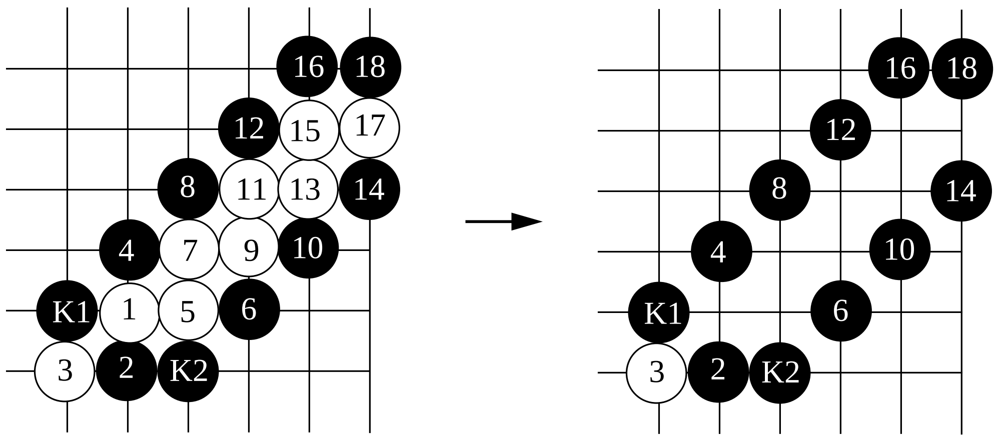


Figure 2.18: Ladder. A ladder results from White trying to save a threatened stone. Black wins the ladder when the edge of the board is encountered. The white stones are then removed from the board.

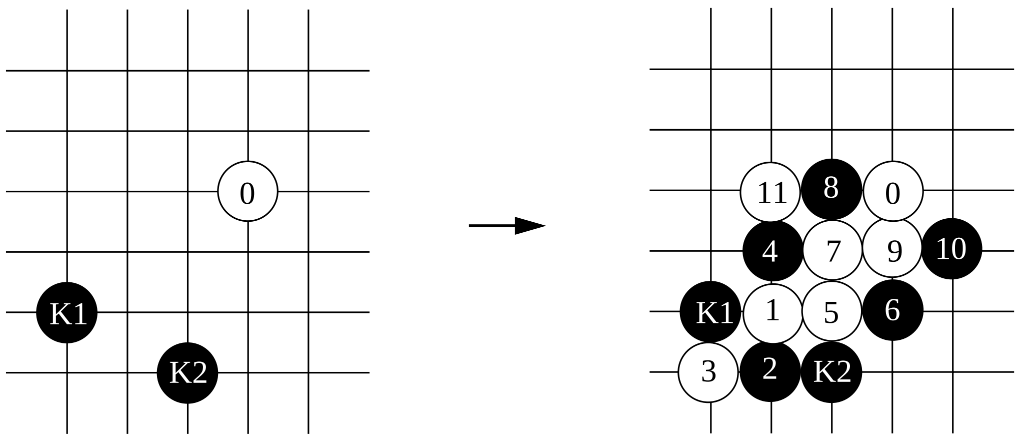


Figure 2.19: Ladder-breaker Stone. A stone on one of the ladder points is called a ladder-breaker and alters the outcome of the ladder. The ladder-breaker stone (white 0) allows White to attack Black’s stones (black 4 and 8) and cut the kogeima link between K1 and K2.

The winner of a ladder can be determined *a priori* and therefore the losing player will not initiate it. Other than when played by novices, ladders do not appear in normal play which makes them difficult to learn by observation. Novices are typically taught to “read” ladders by experienced players or by reading Go books.

2.4.5 Sente and Gote

Skilful players can engineer the game so that they gain the initiative for a number of moves and dictate the response of their opponent by using forcing moves (a simple example is the threat moves in a ladder). In such a case, a player is said to have *sente* and the opponent to have *gote*. Sente usually occurs when players place their opponent in a situation of having to save stones and only having one option available to do so. Sente allows a player to control the course of the game so as to execute a plan of several moves without the opponent being able to frustrate the plan in midstream.

2.4.6 Influence

Stones can be considered to spread *influence* to the empty points and stones around them: a stone played near other friendly stones is more likely to eventually live than a stone played near enemy stones; an empty point more heavily influenced by one side than the other is more likely to be turned into territory by that side. The influence of many stones in close proximity have an additive effect and opposing influence cancels. Influence is an abstract concept although there have been explicit computational algorithms developed to model it.

2.5 Feature Comparison Between Go and Chess

A comparison of thirteen features of Go, over-the-board tournament chess play, Computer Go and Computer Chess relevant to the material contained in later chapters is summarised in Table 2.1 and is described in detail in the associated paragraphs.

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 (a 19x19 grid in Go compared to 8x8 square board in chess).

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 250 - 300 in Go).

	Features	Chess	Go
1	board size	8 x 8 squares	19 x 19 grid
2	# moves per game	~80	~250 - 300
3	branching factor	small (~35)	large (~200)
4	end of game and scoring	usually checkmate (simple definition - quick to identify) although other methods are harder	both players pass in succession and score by 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, ko threats)
6	state of board	changes rapidly as pieces move	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	perceptual structures	pieces belong to small chunks	stones belong to many groups simultaneously
12	handicap system	none/poor	good handicap system
13	typical computer tournament time limits	generous (e.g., 120 minutes to play 40 moves)	severe (e.g., 50 minutes to play 125 moves)

Table 2.1: A Comparison the Features of Chess and 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 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 optimise interactions with stones outside the region of interest or diverge from the known sequence to serve other interests.

4. Both games can be won by resignation of the opponent, or by an outright win (e.g., checkmate in chess, or surrounding more territory and taking more prisoners in Go). In chess, although the end of a game is usually easy to determine (checkmate is simply defined in terms of the position and threats to the king), there are other means of ending the game that are harder to define such as “draws” that can be reached by a number of different routes: agreement by both players, 50-move rule, 3-fold repetition of position claimed by one player, perpetual check (that eventually leads to 3-fold repetition), insufficient material, and stalemate[4]. 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 tend to 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).

4. Thanks to Neil Charness for bringing these difficult chess endings to my attention.

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.

6. The state of the board changes rapidly in chess as pieces move positions. In Go, the board is only gradually changes, 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 and 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.

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) and 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 look ahead 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. Chase and Simon (1973a) showed that expert chess players view the board in terms of small chunks of pieces. In trying to replicate the results with Go players, Reitman (1976) found that the perceptual structure in Go involved stones being seen as a part of many intersecting or overlapping groups. Identifying the group to which stones will eventually 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 a source of potential problems for computer Go programs in estimating features such as influence.

12. One major difference between playing chess and Go lies in the standards of the opponents.

Although no longer used in modern chess, there were handicapping systems in the prior two centuries, such as giving a weaker player first move and a pawn, or a minor or major piece advantage, etc. Some have used time control differences (5 minutes versus 30 minutes)[5]. But I would agree that the handicapping system in chess is very poor relative to that in Go. Thus currently 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 (called hoshi points). Thus the greater skill of one player is offset by the material advantage of the weaker one. In such contests (and many 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, while the program does not change.

13. The typical time limits on Computer Go tournaments are more severe than the typical time limits for Computer Chess tournaments. For the Fourth FOST Cup (1998), each individual program had 50 minutes to complete 125 individual moves (i.e., a game consisted of a minimum of 250 individual moves played within 100 minutes) as compared to Computer Chess tournament time limits which may be as generous as 120 minutes to play 40 individual moves for each program (i.e., a game consists of a minimum of 80 individual moves played within 240 minutes).

5. Thanks to Neil Charness for bringing the historical handicap system in chess to my attention.

Section II

The Computer Go Field

Meta-level Rationale

Section II is concerned with assessing Go as a research domain for AI by addressing the assessment questions AI1 - 4 from section 1.2.1, which are repeated below:

> ***AI1*** *To what extent does the domain of Computer Go facilitate the investigation of knowledge, search and their interaction?*
>
> ***AI2*** *What type of AI research issues can be investigated in the domain of Computer Go?*
>
> ***AI3*** *Do interesting research questions arise from investigations of search, knowledge, and their interaction within the domain of Computer Go?*
>
> ***AI4*** *Do generalisations useful to AI arise from research conducted within the domain of Computer Go? What types of generalisations can be made beyond the domain of Computer Go?*

Overview

Question AI1 is addressed by demonstrating that the issues of search, knowledge and their interaction are core issues in Computer Go. The demonstration involves detailed case studies of current top Go programs (Chapter 4), as well as surveys of both the early (Chapter 3) and contemporary work in the Computer Go field (Chapter 4).

Question AI2 is addressed by surveying some of the approaches which have been developed in order to overcome the difficulties of managing the interaction between search and knowledge in Computer Go (Chapter 5).

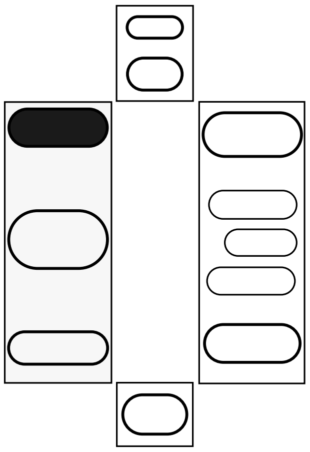
Question AI3 is addressed by considering how the domain of Computer Go can be used to answer research questions related to other areas of interest to AI such as planning, problem solving, pattern recognition, and opponent modelling (Chapter 5). Question AI4 is addressed in Chapter 5 by considering the generalisations useful to AI that can be made beyond Go itself.

Chapter 3

Early Work in the Computer Go Field

Introduction

The best-known AI programming approach to games is brute-force search. Using a brute-force search approach involves a full-width search of the game-tree to a sufficient depth to be able to select the best move at the root of the tree (i.e., to choose the next move for the current position). The selection of the next move is based on evaluating positions in the game-tree using an evaluation function. Games of any interest are too large to exhaustively search and so the game-trees are pruned in order to make the problem tractable.



Early work in the Computer Go field began around the end of the 1960s and beginning of the 1970s when the full-width brute-force approach began to dominate knowledge-based approaches in the Computer Chess field. However, the Computer Go field did not embrace the brute-force search approach used in the Computer Chess field.

This chapter describes some of the early Computer Go programs in section 3.1 and concludes with a discussion of the reasons why brute-force search was not used in earlier Go programs in section 3.2.

3.1 The Early Programs

Three of the earliest Go programs have been selected as representative of the early work in the computer Go field. Zobrist's program (Zobrist, 1970) is described in section 3.1.1; Ryder's program (Ryder, 1971) is described in section 3.1.2; Reitman and Wilcox's program (Reitman, Kerwin, Nado, Reitman & Wilcox, 1974; Reitman & Wilcox, 1975, 1978, 1979a, 1979b; Wilcox, 1988) is described in section 3.1.3.

3.1.1 Zobrist

The earliest full Go program was written by Zobrist as part of a PhD thesis on pattern recognition (Zobrist, 1970).

3.1.1.1 Influence Function

Zobrist introduced the idea of using an influence function to segment the board into black and white territories. The influence function computed a numeric value for every point on the board. Black stones were given a value of +50 and white stones a value of -50 and empty points were given a value of 0. Each point had 1 added to it for each neighbouring point with a positive influence value. Similarly, each point had 1 subtracted from it for each neighbouring point with a negative influence value. This process was repeated another three times (for a total of four times). This influence function spread black and white influence numerically and Zobrist then segmented the board into areas of contiguous positive and negative numeric values (see Figure 3.1).

				1				
			2	2	2			
		2	4	6	4	2		
	2	4	8	10	8	4	2	
1	2	6	10	12	10	6	2	1
	2	4	8	10	8	4	2	
		2	4	6	4	2		
			2	2	2			
				1				

Figure 3.1: Zobrist’s Influence Function. Pattern of influence radiating from a single black stone. Note that the final influence value for the white stone would be 62 rather than 12 resulting from the addition of the initial 50 and the 12 gained from the spreading influence.

The program constructed an internal representation of the board which was stored as numeric values in various 19x19 arrays. The representation included arrays containing information such as:

- the occupation of a point (black, white, empty),;
- number of white and black neighbours, both horizontally, vertically and diagonally;
- the number of stones and liberties for each string;
- the number of empty points adjacent to each empty point; and
- size (including empty points) and number of stones in each segment.

The program also used a second influence function to store an internal representation that measured the influence of stones. It was similar to the influence described above except that occupied points were initially given values of +100 and positive points radiated +3 to their neighbours whilst negative points radiated -4 to their neighbours. Since the computer played black (positive values), this influence function overstated the opponent’s influence value so that the program would play cautiously.

3.1.1.2 Selecting a Move

Zobrist’s program used summed feature evaluation to choose its next move. Once an internal representation was constructed, the program performed pattern recognition over the entire board which amounted to searching for certain numeric patterns contained in the program’s various arrays. With each pattern would be stored an associated move and a numeric value reflecting the priority of the move. When a pattern match was made, the numeric value would be added to the point corresponding to the associated move. An attempt was made to match each pattern “known” to the program across the entire board, rotating and reflecting the patterns as necessary. At the end of this process, the point with the highest value became the program’s next move.

The program exhibited weaknesses in its play which Zobrist wished to overcome. He divided the game into four phases: corner and edge play at the beginning, extension into the centre of the board, defence and attack of loosely staked out territories, and stabilising surrounded territories. To enable the program to play appropriate moves during the various phases, only appropriate patterns (along with patterns that were relevant throughout the entire game) were allowed to be matched during those phases.

3.1.1.3 Results and Improvements

Another addition to the program was limited lookahead or heuristic search. During the pattern recognition process, local areas would be suggested for further examination by lookahead which was carried out to a depth of three moves. The program handled saving/capturing strings, connecting/cutting strings, ladders, and forming eyes using lookahead.

The performance of Zobrist's program was modest, beating two novices but performing poorly against experienced players.

3.1.2 Ryder

Ryder's program (Ryder, 1971) can be seen as an extension of Zobrist's approach in as much as it used an influence function and move selection was based on summed feature evaluation. However, Ryder added both strategic and tactical considerations and also used lookahead extensively. The distinction drawn by Ryder between tactics and strategy was that tactical considerations dealt with short-term objectives whilst strategic considerations dealt with long-term objectives.

Ryder saw Go as a game that required a balance between acquiring actual or potential territory and avoiding the loss of important stones to the opponent. He formulated three areas of interest that were designed to help maintain this balance: board organisation showing the degree of control of each side over various regions; identification of areas of best play for both sides; and analysis of strings to determine whether they were safe or endangered.

Due to the intractability associated with game-tree search in Go, Ryder's program used the combined analyses of many local tactical problems performed by lookahead, together with other analyses rather than searching the game-tree.

3.1.2.1 Selecting a Move

Move generation was accomplished in two passes. In the first pass, the tactical status of all strings were determined. Armies, walls (defined in next section) and groups were identified and each instance was subjected to a process of developmental analysis resulting in the determination of areas and endangered groups for each player. Each legal move was then examined and scored using summed feature evaluation in an attempt to eliminate as many as possible from future consideration, with the 15 best moves being further analysed in the second pass. The legal moves were examined with respect to eighteen features (or patterns). The score obtained by a move was the result of a weighted sum of the features associated with it.

The second pass began with an analysis of the best 15 moves from the first pass with respect to tactical and naive strategic theories. The score obtained by a move from these analyses was added to the score obtained in the developmental analysis. The move with the highest combined score after the first and second passes was then recommended as the next move. Second pass analysis would stop before all 15 moves were analysed if an “obviously” best move was identified.

Ryder’s program used a forward-pruning method to search the game-tree that resulted during lookahead. In forward pruning, a decision based on various criteria was made regarding the desirability of pursuing a given branch during lookahead (i.e., whether the future implications of a particular move were worth examining). The danger in forward pruning is that the best move may be discarded before it has been examined. In Ryder’s program, the criteria for terminating a particular search tree included whether a target string had two eyes, whether a saving move resulted in more than five liberties, or whether it was unlikely that the target would be captured.

3.1.2.2 Influence Function

Like Zobrist, Ryder also used an influence function to provide a numeric value to indicate the degree of tactical control each stone exerted over its neighbours. His influence function was similar to Zobrist’s in that black influence was positive and white influence was negative and the influence value at each point was the sum of the influence values propagated by its nearby neighbours. Ryder’s influence function was simpler than Zobrist’s in that each stone contributed an influence value to its nearby neighbours according to a table of values (see Figure 3.2).

Either too much or too little influence at a point indicated a problem with playing at that point. Too much influence indicated that there was an over-concentration to the detriment of the global situation, whereas too little influence indicated that there was the possibility of capture by the opponent. Optimal moves were those that were close to the peak of the value/influence curve. Influence was regarded by Ryder to be a local property of a stone although he admitted that there

			1			
	1	2	5	2	1	
	2	6	13	6	2	
1	5	13	60	13	5	1
	2	6	13	6	2	
	1	2	5	2	1	
			1			

Figure 3.2: Ryder's influence function.

were cases (such as ladder-breakers), in which stones exerted control on distant parts of the board. In such situations where the influence value was not a reliable measure, tactical evaluation was used to overcome the short-comings of the influence function.

One situation in which the influence function was not a good measure was that of connectivity. Ryder defined two terms: *connectedness* and *strongly related*. The definition for connectedness was that a point was connected to colour X if it was occupied by a non-dead stone of colour X, or, if it was empty, it was adjacent to at least one stone of colour X and none of the opposite colour. An extension to connectedness was half-connectedness. An empty point was half-connected to colour X if it was adjacent to at least two stones of colour X and one or more stones of the opposite colour. A point was strongly related to colour X if it was occupied by, or was adjacent to, a stone of colour X. A point could also be strongly related to colour X if it was diagonally adjacent to a stone of colour X with at least one empty point (kosumi link) or no more than one empty point away (ikken-tobi link). Groups were defined as sets of points that were connected or half-connected.

The influence function was particularly suited to determining walls and armies. Walls were defined as a connected set of points in which the influence values were equal to or larger than a predefined wall threshold. Similarly, armies were defined as a set of points (including empty points) in which the influence values were equal to or larger than a predefined army threshold.

The tactical concepts employed in Ryder’s program included liberty estimation, ko, ladders and snapbacks[1], and both eyes and false eyes. Ryder admitted that improvement was needed in the program’s tactical ability which he described as “blunder avoidance”.

3.1.2.3 Representation

Information was included in the representation if it satisfied any of the following criteria: it was basic and was needed in many calculations; it took a significant effort to re-calculate and was used more than once; or it was accessed many times and was constant through many if not all of those accesses.

Groups and armies were represented in a link-based manner. There was a link from each point in a group or an army to the entity descriptor of the group or army it belonged to. Each entity had a link to one of the points that comprised it which was the base point for that entity. Searches for points near entities were initiated from the base point.

The global representation of the board was contained in a 21x21 array (19x19 with an empty point border around the outside). By using various bits in the memory location of each point in the array, the following information/links were able to be stored: area number; army number; wall number; group number; string number; and current state (black, white or empty).

3.1.2.4 Results and Improvements

Ryder reported the results of only one game which was lost to a novice who had just learned the rules and had been given some rudimentary tips on the tactics and strategies of Go. With regard to Ryder’s main interest, he reported that the tactical analysis was quite an efficient searching mechanism that provided valuable information for move generation and compared favourably to the powers of analysis exhibited by human beginners. He was also of the opinion that only a small effort would be required to improve the program’s tactical strength.

Future improvements suggested by Ryder included re-using previously calculated information in searching, the incorporation of distant ladder-breakers into tactical analysis and improving the definition of endangered groups. Ryder identified the developmental analysis stage which

1. Snapbacks are situations in which a friendly stone is captured by the addition of an enemy stone to an existing enemy string that gains one liberty as a result of the capture (i.e., the enemy string is in atari). The enemy string is immediately captured by a friendly stone played where the first enemy stone was captured from (this is not a ko situation since more than one stone is captured and therefore the board position is not repeated).

consumed around 10% of the processing time as needing attention. One possible reason for the poor performance of Ryder's program was that it made no distinction between the opening, mid-game and end-game, and was therefore playing inappropriate tactical moves in the end game in particular. Ryder recognised that the addition of such distinction would be of importance in future improvements.

3.1.3 Reitman and Wilcox

Walter Reitman and Bruce Wilcox began using Go to conduct AI research in 1972 investigating pattern recognition (Reitman, Nado & Wilcox, 1978; Reitman & Wilcox, 1975, 1978), planning (Reitman, Kerwin, Nado, Reitman & Wilcox, 1974) and replication of human perceptual and cognitive abilities in Go. Indeed, the objective of their program was to model human Go play and to this end, they videotaped protocols and post-game analyses of a strong Go player to try to ascertain how humans play Go[2]. The program that resulted from that research has been described both as the Reitman-Wilcox Go program and the INTERIM.2 Go program. Subsequently, Wilcox rewrote the INTERIM.2 program to produce Nemesis, a commercial Go program (Wilcox, 1985).

According to Reitman and Wilcox, the three essential ingredients to a Go program were perception, knowledge and coordination (Wilcox, 1988). They tried to incorporate into their program's representation as many of the perceptions of a skilled Go player as they could. They identified and stored two types of Go knowledge: process knowledge that included how to kill a group, and how to make territory; and evaluation knowledge that enabled the judgement of the stability of groups, and the estimation of relative values. Coordination of the program's perceptions and knowledge was achieved through appropriate control structures which allowed it to choose the next move.

The INTERIM.2 program created and maintained a representation that was a selectively updated, multilevel network which Reitman and Wilcox said was analogous to the type of perceptual and cognitive representation a skilled Go player would create (Reitman & Wilcox, 1979b). There was a hierarchy of experts and critics that operated on the network which chose

2. Kerwin, J. and Reitman, W. (1973) Video game no. 3: a Go protocol with comments. Mental Health Research Institute, Communication No. 300, University of Michigan, Ann Arbor, Michigan, as cited in Reitman and Wilcox (1975).

the next move. Lookahead was selective and goal-driven rather than full-board.

3.1.3.1 The INTERIM.2 Program

INTERIM.2 summarised information “up” and filtered decisions “down” (Wilcox, 1988). Information contained in low-level representations (e.g., strings) were summarised and used in high-level representations, (e.g., groups). The program made decisions at a high level about what to “focus on” and these decisions were filtered down to lower levels and influenced decisions made at those levels.

3.1.3.2 Data Structures and Control Flow

The program stored its representation in a structure called GAMEMAP which was indexed through a set of interrelated arrays called GAMEBOARDS. In each GAMEBOARD array there were pointers to every instance of a particular data structure (e.g., STRING-BOARD contained pointers to every string on the board). The GAMEBOARDS progressed from simple to complex: TYPEBOARD, STRINGBOARD, LINKBOARD. Special GAME-BOARDS such as LENSBOARD, WEBBOARD, SECTORBOARD and TACTICSBOARD were designed to contain representations that corresponded to a skilled human player’s perceptual and cognitive representations.

The flow of control in the INTERIM.2 program oscillated between MOVE and REFLEX. MOVE selectively updated the components of GAMEMAP that were affected when stones, black or white, were placed on the board. REFLEX was responsible for choosing the program’s next move which was achieved by stepping through an ordered list of functions and accepting the first move returned by them.

3.1.3.3 The Tactician (PROBE)

The program’s tactician, PROBE, was used to answer specific questions and therefore performed narrow, goal-driven lookahead instead of general purpose, full-board lookahead. The basic assumption behind PROBE was that if the results of a small number of intelligently chosen lines were the same for a given question, then those results would most likely be the answer to that question. Although such an assumption is not always valid, Reitman and Wilcox felt it was a small price to pay for a quick search and was not dissimilar to the penalties experienced when human searches failed for similar reasons. PROBE was designed to model how skilled human Go players performed tactical problem solving by selective lookahead (Reitman & Wilcox, 1979a).

PROBE would be employed to answer a specific question and would propose a reasonable initial move. The initial move would be hypothetically played resulting in a new hypothetical board position. PROBE would then propose another reasonable move based on the hypothetical board position. This process would continue until a position was reached that could be judged as either a success or a failure with respect to the specific question that PROBE was trying to answer. If the position was a success, PROBE returned the initial move as the result. If the position was a failure, the hypothetical moves were retracted until there was a position in which a reasonable alternative move could be made and the process was repeated. In classical lookahead, all moves are examined in this fashion (i.e., lookahead is full-width) so that the best move could be discovered (e.g., brute-force search in chess programs). In PROBE, not even all reasonable moves were examined. Each side was limited to two or three failed variations before the search was terminated and a failure result was returned.

The selection and examination of reasonable moves was conducted by a hierarchy of experts which contained a large amount of specific Go knowledge. The experts formed, evaluated and implemented goal-directed “chains of thought” (Wilcox, 1988). There were a variety of experts including experts that generated goals and sub-goals; experts that proposed moves to realise goals; experts that used exclusively the current board position and current goals; and experts that used all goals and subgoals. The control structures implemented by Reitman and Wilcox in PROBE allowed the knowledge contained in the various experts to be organised, easily debugged, and readily extended (Wilcox, 1988). The goal-directed behaviour of PROBE resulted in very restrictive searching which typically resulted in around 60 - 80 hypothetical moves per lookahead (Wilcox, 1988).

3.1.3.4 Perceptions and Representations

INTERIM.2 maintained many different data structures in its representation. Point occupancy (black, white, empty) was used to determine strings. A basic property of each string was whether it could be captured or saved. The links recognised by INTERIM.2 were kosumi, ikken-tobi, nikken-tobi, kogeima and ogeima. Links were classified as normal (connectable), dead (unconnectable but potentially useful) or string capture (linkages between strings capturing an opponent string). Basic properties of linkages included whether they could be joined or cut, whether they crossed opponent links, and why they were created. Special edge linkages that were links from strings up to the fourth line to the edge of the board were also recognised. Sector lines were long range links and were used in stability assessment and move generation. Virtual strings were strings connected by unbreakable links. Groups were sets of strings joined together by links. Enclosed territory was a set of empty points or dead stones that were enclosed by strings, linkages and edge linkages. The motivation for moves (e.g., why a link was created) was explicitly tracked to facilitate move selection and to ease debugging.

Influence was radiated using Ryder's algorithm. Reitman and Wilcox saw influence as misleading in certain situations and therefore developed the notion of influence blocks. Areas of empty, contiguous, same-signed influence points that did not include linkage points were collected into a structure. The perimeter of these structures were located and then iteratively "shrunk" to provide nested loops. Since shrinking did not take place across stones or linkage points, it was possible that the loops created in the shrinking process would become arc segments or strips. Eventually individual structures emerged that were called influence blocks. Points contained in an influence block were considered more likely to be turned into territory the further they were away from the perimeter of the block.

Creating and maintaining INTERIM.2's representation was described by Reitman and Wilcox as "perception" and the data structures used were also sometimes referred to as "perceptual representations". Two components of the perception system employed by the INTERIM.2 program are particularly interesting. *Webs* were used to determine what regions could be "seen" from a stone and *lenses* were used to monitor actual patterns of stones in a game using the program's knowledge of Go patterns.

A web was constructed around a group as concentric "circles" of liberty points (circumferential strands) to a maximum depth of nine levels. Radial strands emanated from the groups. The web did not continue behind other stones, either friendly or unfriendly. Links between stones and between stones and the edge of the board (up to three moves away) also stopped a web. The continuous loop of the outermost points of a web comprised its perimeter. The points between the group and the perimeter were considered to be transparent. The perimeter of a web was useful information for a group in deciding where potential threats might come from, where it could run to avoid those threats, the location of nearby friendly groups, where linkages might be attacked, etc.

To Reitman and Wilcox, local stone configurations and their transformation by sequences of moves was a fundamental aspect of Go. Sequences of moves associated with standard configurations (e.g., attacking or defending a kogeima link) are recognised by competent Go players. Lenses were designed to monitor such configurations and suggest moves that were known pattern continuations.

A *lens* comprised a set of *fields*. Each field consisted of a set of points and watched for the development of a sequence of moves. The lens in turn supplied information about the suggested moves and global status information to the tactician. When a field recognised a configuration, it supplied suggested moves and other associated information to the lens. When fields no longer recognised patterns they were discarded, which caused a lens to get smaller over time. This in effect amounted to the “focus” of a lens becoming more concentrated over time.

3.1.3.5 Performance

The results of three games played by the INTERIM.2 program are reported in Reitman and Wilcox (1978). Two were won against players ranked 22 and 34 kyu and one was lost against a player ranked 4 kyu. The computed ranking of the INTERIM.2 program from these games was 27 kyu.

Reitman and Wilcox were pleased with the performance of the INTERIM.2 program in regard to their initial objectives of modelling human play. It played conservatively, choosing to defend endangered groups in preference to attacking opponent groups. It defended well; the only cases in which it lost groups (other than due to bugs) were when several groups were simultaneously endangered.

3.2 Discussion

The amount of knowledge used in a Go program steadily increased at the beginning of the Computer Go field. The early Go programs encoded Go knowledge in the form of patterns and also influence functions. The use of patterns was generally via summed feature evaluation to provide enough information to effect forward pruning, that is, to enable the program to concentrate on the most promising candidate moves. The use of the influence function was generally to identify control of territory. However, influence was unreliable in certain circumstances and had to be either be supplemented with other information or modified to reflect higher-level knowledge.

As mentioned in the introduction, the Computer Go community was established around the time that the Computer Chess community was turning to brute-force search in preference to knowledge-based approaches. The early Go programs did not use brute-force search primarily because of the size of the game-tree[3]:

- there are 361 points on a 19x19 Go board compared to 64 squares on a chess board;
- the average branching factor of Go is approximately 200 compared to approximately 35 for chess;
- the average Go game is between 250 and 300 plies compared to around 80 for chess; and
- the size of the state-space (i.e., the number of possible positions) is approximately 10^{172} (i.e., approximately 3^{361}) compared to approximately 10^{50} for chess and 10^{30} for othello (Allis, 1991). The size of the game-tree (i.e., the number of possible games) is estimated[4] to be 10^{575} compared with estimates of 10^{123} for chess and 10^{55} for othello (Allis, 1991).

Although not commonly recognised, a contributing factor to the failure of brute-force search was the lack of adequate evaluation functions that would enable the large game-trees to be pruned.

Search in the early programs was local rather than full-width and was used for tactical purposes such as capturing/saving strings, ladders, connecting/cutting etc. Since search was not performed at the full-board level, it was necessary to implement a way to forward prune moves, that is, to select which moves would be examined and which would not. Another difficulty was that searches could only be allocated a certain amount of resources and had to be terminated when those resources were consumed.

Summary

The early Go programs can be characterised as being knowledge-intensive rather than search-intensive. Local search requires forward pruning, and knowledge is needed to select which

3. Computationally, Go is a very difficult game: determining the winner from an arbitrary position is P-space hard (Lichtenstein & Sipser, 1980) and deciding whether a player can force a win is exponential-time complete (Robson, 1983).
4. The estimate of 10^{575} is based on an average branching factor of 200 and average game length of 250 ply (i.e., 200^{250}). The size of the game-tree may be as high as 10^{690} if the average game is considered to be 300 ply (Burmeister & Wiles, 1995). The estimate of 10^{360} provided by Allis (1991) is based on an average branching factor of 250 and an average game length of 150 ply.

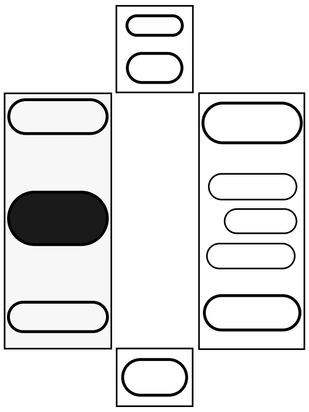
parts of the game-tree to search and which parts not to search.

The question that follows is how the Computer Go field evolved, particularly with respect to brute-force search versus knowledge. In the next chapter, the current state-of-the-art in the Computer Go field will be examined through case studies of Many Faces of Go and Go4^{++} and a general overview of competitive programs.

Chapter 4

State-of-the-Art in the Computer Go Field: Case Studies and Analyses of Current Go Programs

Introduction



Go is one of the last formal game domains in which computer performance is not competitive against even moderate human players. As mentioned in Chapter 1, the current state-of-the-art Go programs play at about the level of someone who has played a few games a week for a year and studied some introductory Go books. The unclaimed Ing Prize (US $1.6M for the first Go program to beat a professional player in a best-of-seven match without handicap) is by no means the only inducement to programmers. The Computer Go field flourished in the 1980s when Computer Go tournaments began and the first commercial programs were released. There are several annual Computer Go tournaments, notably the FOST[1] Cup which awards approximately US $20,000 for 1st place.

This chapter investigates the depth and breadth of the Computer Go field by investigating some of the contemporary competitive Go programs. In-depth descriptions are given in the case studies of Many Faces of Go in section 4.1 and Go4++ in section 4.2 The breadth of field is covered in section 4.3 by describing several Go programs with respect to state representation, move generation, goal states, evaluation, tactical search, and influence functions.

1. Foundation for the Fusion of Science and Technology.

4.1 Case Study 1: Many Face of Go

David Fotland is the author of The Many Faces of Go (*MFG*) which is one of the best commercial Go programs. Fotland began writing Go programs in 1981 and released MFG in 1990 (G2 and Cosmos, the predecessors to Many Faces of Go are described in Appendix B[2]). The description of MFG below extends Fotland's description of knowledge representation in MFG (Fotland, 1993). This case study of MFG was only possible with the help of David Fotland who kindly shared previously unpublished[3] details of MFG in 1996[4] by email and in person at FOST '96. In particular, Fotland shared details of the tactician, life and death analysis and scoring, the static life and death evaluator, the life and death search engine, the score evaluator, and how MFG chooses a move.

4.1.1 Components of MFG

4.1.1.1 Data Structures

The dynamic data describing a board position is stored in three classes of data structures: incremental, locally recalculated, and globally recalculated. The data in the incremental data structures is updated upon addition or removal of a stone and includes low-level information such as lists of empty points, number of liberties etc. Locally recalculated data is updated as it is needed, that is, it is only updated for regions of the board that have been affected by a particular move or move sequence, and includes connection strength, and eyes. Globally recalculated data is updated for the whole board and includes group strength and influence. The locally and globally recalculated data structures are maintained by the evaluation function. The dynamic data is stored globally and is generated progressively in several passes of increasing abstraction. The dynamic data used in MFG includes point environment, string data, connection data, eye data, potential eye data, group data, territory, and score.

Information pertaining to connectivity between two strings is stored in a connection data structure. Each connection has a list of connection points, that is, a liberty of one string that is

2. The descriptions of G2 and Cosmos were largely derived from the literature but also included some personal communication with Fotland.
3. Fotland has publicly discussed many aspects of MFG on the Computer Go mailing list (*cgm*) which is an email forum for discussing Computer Go issues. As at September 1999, instructions for joining the list can be found at http://www.hsc.fr/computer-go/ and the address of the list is computer-go@hsc.fr.
4. The nature of Go programs is that they are constantly evolving. Thus, this description is a time stamp of MFG in 1997.

no more than three points away from the other string. Thus, the connections recognized by MFG include hane, one point jump (ikken tobi), knights move (kogeima), two point jump (nikken tobi), large knights move (ogeima), three point jump, and bent three point. Other special connections near the board edge are handled through the pattern database. The data stored in the connection data structure includes the strings involved in the connection(s), lists of one, two and three point connections (i.e., as measured from the connection points), the type of each connection, and the strength of each connection (e.g., already cut, cuttable, shared connection, connected with aji, connected solidly). The type and strength data is stored in a locally recalculated data structure whereas the connection data is stored in an incremental data structure.

The eye data structure records information pertaining to either a block of empty points or dead opponent strings that are partially or completely enclosed by friendly stones. Data stored in the eye data structure includes colour, a list of the points in the eye, vital points, and eye type (e.g., one point, two point, line eye, big eye, and dead group eye). The number of eyes (to a resolution of 1/8 of an eye) achievable under various conditions are recorded including the number of eyes if the opponent moves twice, the number of eyes if the opponent moves next, and the number of eyes if the program moves next. A board point can only be recorded as belonging to one eye. The eye data structure is a locally recalculated data structure since eye data does not lend itself to incremental update. In hindsight, Fotland says that a pattern based approach to recognizing eyes would be preferable (Fotland, 1993).

The evaluation of group strength and the generation of attacking and defending moves involves identifying potential eye space and running points. The potential eye data structure contains type, value and location data for potential eyes. An example of data stored in the potential eye data structure is “extend along edge” (type), value is the number of new points of eye space, and location is the liberty at the start of the extension (i.e., the first point played in the extension). Running points are stored by type (e.g., running towards friendly/unfriendly stones, in several different lists. Potential eyes and running points are stored in a globally recalculated data structure. The types of potential eyes include play on eye vital point, connect to friendly group, capture tactically threatened string, extend along edge of board, block enemy incursion into eye space, and run away into centre.

4.1.1.2 Tactician

Every string with three or less liberties and many strings with four liberties are read by the tactician. Each string is read twice, once with White moving first and once with Black moving first. The tactician determines whether a string is unconditionally captured (can be captured if the friendly side moves first, wins kos, and ignores threat to nearby groups), unconditionally threatened (can be captured if the opponent side moves first and the friendly side wins kos and ignores threats on nearby groups), conditionally threatened (can be captured if the opponent side moves first and friendly side answers threats to nearby groups and also wins kos, or may have to sacrifice nearby group), or unconditionally alive (can not be captured even if the opponent side moves first, wins kos and the friendly side answers nearby group threats). The tactician relies on simple heuristics concerned with the number of liberties and connectivity, that is, pattern matching is not used in the tactician.

The tactician has two separate move generators; one to generate attacking moves and one to generate defensive moves. The moves suggested by the move generators are sorted according to criteria that include second order liberties, cut points, and simple eye shapes. Once sorted, an alpha-beta depth-first search is employed with the performance of the search depending on the quality of the move sorting (Fotland, Computer Go mailing list[5]).

Tactical searches are goal directed and are limited to a maximum number of nodes. For string captures this limit is approximately 100, however, when only one move is suggested, it is not counted towards the node limit. In this way, ladders can be read without problems. The number of nodes for a search is allocated to the branches according to the value given to the moves by the first ply move generator and thus different branches may end at different depths. The branching factor at each successive ply is progressively constrained by the tactician. The branching factor falls from five at the first ply to one or two by the fifth ply.

Since the data structures in MFG are separated into incremental, locally recalculated and globally recalculated, low-level local tactical searches can be executed quickly by only updating the incremental data structures. However, the possible goals of such a search are limited since the high-level data structures are not recalculated. It is not possible, for example, for MFG to search for *x* liberties and two eyes.

5. 5 March, 1993.

The tactician is used to read connections and eyes. A cutting stone can be examined by the tactician to determine whether or not it will be captured. Stones on the diagonal of eyes can be examined to determine whether they can be captured. These types of search do not exceed approximately 12 ply since the program actually plays worse if more plies are considered.

MFG uses caching to reduce the amount of time spent reading. Since strings are read twice, tactical results related to eyes, capture or threat, and cuts are cached rather than re-evaluated. In the course of a game, MFG performs approximately 100,000 -150,000 tactical searches, visiting between 1.5 - 2 million nodes at approximately 2,000 - 2,500 nodes per second. At each move, approximately 1,000 tactical searches are performed at between 20 - 30 per full-board evaluation. On average, each tactical search visits between 10 - 20 nodes although many searches visit less than 5 nodes and a few searches per evaluation are terminated due to the node limit.

4.1.1.3 Influence Function

MFG uses an influence function that is inversely proportional to distance. Influence is radiated to a fixed maximum distance of nine although when lower playing levels are chosen, the distance is smaller. Stones and connection impede the radiation of influence. Black and White influence are radiated separately (i.e., they are not summed together to give a single value for each board point) so that their relative values can be compared. Dead stones radiate negative influence and thus, since Black and White influence is maintained separately, dead stones reduce their own influence rather than increase their opponent’s influence. The initial influence value radiated from a stone depends on its strength.

Influence is not used to determine group connectivity. The use of influence in MFG includes determining thickness and territory, identifying surrounded groups, and generating moyo building or reducing moves. Another way in which influence is used is to determine a group’s running ability. A group’s ability to run towards a liberty depends on the relative values of friendly and unfriendly influence nearby. The sum of a group’s ability to run toward each of its liberties determines its running ability whilst the gradient of the influence determines the direction in which it should run.

4.1.1.4 Pattern Database

Each pattern in the pattern database has a move tree associated with it. The pattern database contains approximately 1,200 patterns of size 8x8 and approximately 6,900 suggested moves with the average number of moves stored for each pattern being about five.

Each point in a pattern is marked as black, white, empty, white or empty, black or empty, black or white, or anything (i.e., do not care). Patterns are categorized according to whether they are in the middle of the board or whether they contain edges and/or corners.

A successful match of a pattern depends on both matching the stones in the pattern with those on the board and matching any attributes associated with the pattern. Each pattern has as many as ten attributes that pertain to stones and include whether they were dead or alive, minimum number of liberties, and connectivity between stones.

To match over 1,000 patterns on a 19x19 board in 16 different orientations requires approximately three million primitive operations. Since performance is a critical issue, the patterns are compiled into a bit array which facilitates fast bit-parallel comparisons. By using an 8-bit hash function and bit-wise operations, the number of primitive operations is reduced to approximately 350,000. This number of operations still takes enough time to make it too expensive to match all patterns more than once per move.

Patterns from the pattern database are evaluated after being played on the board in their entirety, that is, they are evaluated at the endpoints of their move-trees. A small subset of patterns are used to read local sequences based on shape. Fotland has begun evaluating the strength of one point jump connection using patterns.

A fast pattern matcher is necessary for a strong Go program according to Fotland. Pattern matching can be used to suggest and sort moves, and to classify connections and eyes. Reading is employed to determine whether eyes are false. Pattern matching on its own is not adequate for life and death and group strength determination (Fotland, Computer Go Mailing list[6]). The eye and connection patterns are hand coded decision trees.

6. 8 April, 1993.

One method that Fotland tried to increase the number of patterns in the pattern database was to take them from professional games. When replaying a professional game, if MFG did not consider the move played by a professional or the move played was low on the suggestion list, Fotland would cut-and-paste an 8x8 section from the game into the pattern database, modifying it to include the anything (do not care) points and adding the attributes. Fotland discontinued this method since most of the missed moves were tactical (i.e., based on lookahead not shape) or depended on a larger context than the 8x8 section that was placed in the pattern database.

4.1.1.5 Joseki Database

The joseki database contains approximately 45,000 moves. The entire joseki database is a single acyclic graph (i.e., a graph containing no loops) with an empty corner at its root. Moves in a joseki sequence can be one of several types including urgent, complex, trick, and bad. The urgent moves have priority whilst the complex and trick moves are only played if MFG is behind. The bad moves are not for the program to play, rather they are used by the program to know how to respond to bad moves played by its opponent.

Joseki patterns are evaluated by being played on the board in their entirety and evaluating the resulting board position. MFG selects joseki sequences that are appropriate given the status of the other corners (Fotland, 1993).

4.1.1.6 Life and Death Analysis and Scoring

Life and death analysis in MFG consists of a life and death search engine and a static life and death evaluator that is used by the life and death search engine. The value returned by full board evaluation is ultimately the score for the position returned by the score evaluator which estimates the score and employs the static life and death evaluator.

4.1.1.7 Static Life and Death Evaluator

The static life and death evaluator determines the life and death status of every group in a multiple pass process without using lookahead by progressively creating complex structures from simple structures. The tactician is used by the static life and death evaluator and is called at least twice for each appropriate string in a group. An improvement in the static life and death analysis is achieved by making a third call to the tactician in order to determine whether threatened strings are conditionally or unconditionally threatened.

A group’s life and death status (or strength) is quantified as one of 25 values. The life and death values are divided into five main categories: very alive, alive, unsettled, weak and dead. Determining the strength of a group includes considering potential connections and potential eyes. The life and death status of groups were used in various ways including guiding the life and death reading search engine (see next section), and to evaluate territory based on group strength (see section 4.3.4).

4.1.1.8 Life and Death Search Engine

A life and death search engine is used to perform life and death searches on particular groups and is similar to the tactician except that it performs life and death analysis on groups rather than tactical analysis on strings.

A life and death search is goal directed (e.g., to save or kill a group). If the goal of the search is not achieved at a particular node, appropriate moves are generated by its own move generators and the search is continued. The static life and death evaluator is called at each node during a life and death search in order to determine whether the goal has been achieved.

The life and death search engine uses depth-first alpha-beta search. There is no fixed depth to the search tree; the depth to which a particular branch is searched is heuristically determined. The size of the search tree is constrained and is typically approximately 20 nodes in size and up to seven plies deep.

Life and death search is slow and full trees are cached to reduce the overhead associated with re-searching the trees. The slowness of life and death search also means that it is not used during full board evaluation.

4.1.1.9 Score Evaluator

An estimate of the score of a position is returned by the score evaluator and is based on territory estimates which are based on group strength and a few other considerations. The static life and death evaluator is called to determine group strengths (life and death status) for all the groups on the board. Quiescence search (see section 4.1.2.3.) is achieved by the score evaluator recursively calling itself.

4.1.2 How MFG Plays a Move

The general approach taken by Fotland in MFG is a four step process. Firstly, the position is strategically evaluated and goals chosen. Secondly, candidate moves are generated and full board lookahead performed. Thirdly, the moves are evaluated either in relation to the strategic goals (e.g., were they achieved) or by evaluating the resulting full board positions. Fourthly, the best move is selected.

According to Fotland, improvements to MFG’s performance come primarily from improving the sorting of the candidate moves (i.e., considering the better moves first) and by improvements in the evaluation function, particularly improvements in eye shape (see tactician, section 4.1.1.2) and territory estimation (see score evaluator, section 4.1.1.9).

4.1.2.1 Strategic Evaluation

A strategy function is used to focus attention on important areas of the board before each move (Fotland, 1993). The dynamic data is examined to determine what phase the game is in, the score for a pass move, the value of taking sente, and whether there are any urgent offensive or defensive moves that need to be made.

During strategic evaluation, the static life and death evaluator determines which groups are neither very alive nor dead (i.e., groups that are alive, unsettled, or weak). Life and death searches are conducted on these groups and the results stored. It is possible that the status of a group determined to be very alive or dead by the static life and death evaluator may change as a result of life and death searching even though it is not explicitly examined by the life and death search engine. The moves associated with saving/killing such groups receive high priority during move generation (see next section).

4.1.2.2 Candidate Move Generation

Candidate moves are generated and up to the best ten are evaluated before one is selected as MFG’s next move. Heuristic values are associated with each candidate move so that they can be sorted.

Candidate moves are generated by a rule-based expert system and include fuseki (e.g., shimari, kakari, and joseki moves), moves on the edge (e.g., invasions and extensions), playing in the centre, conservative play when ahead, risky play when behind, influence (e.g., defending a moyo running), pattern matching (e.g., cutting and connecting, surrounding and escaping, invasion and defence, endgame, killing and saving groups, and shape-based moves), group defensive moves (e.g., making eyes, running, and fighting semeais), contact fights (e.g., blocking, extending for liberties, and hane), ko threats, and dame.

A limited form of full board lookahead is achieved by storing move sequences in the pattern and joseki databases. Move sequences attached to patterns in the pattern and joseki databases are played onto the board and evaluated at their endpoints. Thus if a particular joseki is suggested as appropriate the stones in that sequence are played and the resulting board position is evaluated. This procedure enables MFG to "play" to the end of a lookahead sequence and minimax the score returned by the evaluation function for the resulting board position.

Move generation is affected by the phase that the game is in. Certain rules will only fire in certain phases of the game. The fuseki phase lasts until joseki sequences have been pursued in all the corners. The middle game lasts until at least move 120 and MFG enters the end game when there are no unsettled groups on the board. Transition between middle game and end game phases may occur many times during a game.

The types of moves generated are also affected by the relative score. MFG plays more conservatively when it is far ahead of its opponent and makes unsound invasions when it is far behind its opponent. The relative score is classified into many states including way ahead (over 40 points), ahead (20 - 40 points), about even, behind (over 10 points), and way behind (over 20 points)

Urgent offensive and defensive moves are examined initially and if sufficiently urgent are played without any further considerations. Urgent moves can be suggested by the pattern matcher or by the life and death search engine (i.e., moves that change the life and death status of groups determined to be very alive or dead by the static life and death evaluator).

4.1.2.3 Board Evaluation

The evaluation function consists of the static life and death evaluator and the score evaluator. The final value returned by the evaluation function is the estimate of the score provided by the score evaluator.

In order to try and overcome the difficulties associated with the horizon effect, Fotland chose to use quiescence search rather than modify the evaluation function. In quiescence search, positions are evaluated when they become quiet rather than when they are in a state of flux. Positions that are not quiet are characterized by oscillations in the evaluation values returned for successive moves that makes it difficult to make good decisions based on the evaluations. MFG uses quiescence search to overcome this problem by either the score evaluator recursively calling itself (i.e., generating further moves) until a position becomes quiet or by using patterns suggested as obvious local answers to make the position quiet. MFG will generate up to six plies of moves in a quiescence search which improves its performance by up to four stones (Fotland, Computer Go mailing list[7]).

4.1.2.4 Move Selection

Since the evaluation function is very slow (Fotland, Computer Go mailing list[8]), a limit is placed on the number of full board evaluations that can be performed each move, depending on the playing level. MFG can only look at five to ten moves in detail (depending on playing level) before selecting the move it finally plays. A safeguard against errors in the evaluation function is that only reasonable moves are suggested for full board evaluation (Fotland, 1993).

By employing the strategy function, the move suggestion expert system, and the joseki and pattern databases, MFG fully examines a small number of moves, playing the one that receives the highest score from the evaluation function (i.e., receives the highest value from the score evaluator).

The score returned from a full board evaluation is also modified by the addition of sente if applicable. Thus if MFG plays a sente move, the value of sente is added to the evaluation for the resulting position. The value of sente varies from seven points during the fuseki to one or two points in the endgame and is determined by the strategy function.

4.1.2.5 Size, Performance and Timeline

With some help from Ishi Press and a professional artist, Fotland rewrote Cosmos’ user interface and released it as MFG in 1990. Between then and March 1997, MFG has had three releases with various improvements. MFG contains approximately 40,000 lines of C code with approximately another 20,000 lines of user interface code. MFG was awarded a 6 kyu certificate from the Nihon-Kiin (Japanese Go Association) at FOST ‘96.

7. 30 September, 1993.
8. 16 August, 1994.

4.2 Case Study 2: Go4^{++}

Michael Reiss started writing Go programs in 1983 and his program Go4^{++} is one of the best commercial Go programs[9]. Go4^{++} has evolved from various predecessors over that time. Reiss had never previously published an in-depth description of Go4^{++} and generously assisted my compilation of this description in person at FOST'96 and by email[10].

Reiss' programming philosophy is to use simple algorithms on a large amount of data rather than complex algorithms on a small amount of data. Go4^{++} calculates everything from first principles: complex concepts are calculated from simple concepts which are in turn functions of even simpler concepts. Thus, Go4^{++} does not have an elaborate rule-based expert system at its core. A positive consequence of this approach is that Reiss' relatively weak ranking (2 kyu) compared with other top Go programmers is not a deficit. It also means that Go4^{++} usually responds well to the strange moves (by human standards) that are quite often played by Go programs and which are difficult to foresee when designing a rule-based system. The major drawback of this approach is that it is computationally intensive and requires a powerful computer.

At the foundation of Go4^{++} is a massive amount of connectivity data from which almost everything else is eventually derived. Go4^{++} selects a move by first generating about 50 candidate moves that are analysed by an evaluation function that estimates territory. The move scoring the highest evaluation is played as the next move. Thus, Go4^{++} uses one ply global lookahead, although tactical lookahead and search was used in evaluating each move.

4.2.1 Candidate Move Generation

Candidate moves are generated by a process of pattern matching to determine the best 50 moves to examine in depth. Go4^{++} has approximately 15 high-level patterns that include terms for the probability of an eye, safety, and territorial value. Whenever a match is made, the board point(s) suggested as good places to play have an appropriate value added to them. The 50 highest scoring board points are then examined by the evaluation function. The board point that receives the highest score from the evaluation function is played as the next move.

9. Go4^{++} is the program's tournament name; it has been released commercially as Go Professional.
10. This description is a time stamp of Go4^{++} in 1996.

In some circumstances, Go4^{++} effectively evaluates more than 50 candidate moves per turn. Pattern matching is not applied to regions of the board where the stone safety (see section 4.2.2.4) and territory (see section 4.2.2.6) do not differ from those of the previous move. Thus after evaluating the 50 highest scoring board points after pattern matching, the next move is selected from the combined set of evaluation scores of the 50 board points just considered and the evaluation scores of any board points in the undisturbed regions that were considered during the selection of the previous move.

4.2.2 Evaluation Function

The evaluation function is a six step process. First, a connectivity probability map is generated. For each black and white stone on the board, the probability of connecting it to a friendly stone (real or hypothetical) at the 32 connectable points[11] is computed (generating a large amount of data). Determining connectivity involves tactical search. Second, groups are determined from the connectivity map and tactical search. Third, eyes are determined (using patterns) from the connectivity and group data. Fourth, a group's safety is determined based on the number of eyes it has. Fifth, the safety of each stone is radiated in proportion to its connectivity probability map and summed over all stones. Sixth, Black and White territory is estimated from the radiated values. The difference between Black and White territory is returned as the evaluation for a given move.

4.2.2.1 Connectivity Probability Map

Each time a hypothetical move is played on the board, a connectivity probability map for all friendly and opponent stones (including the hypothetical stone) is generated and stored. The connectivity probability map for a stone contains the probability of connecting to nearby friendly stones already on the board or to an empty point if it was occupied by a friendly stone. Stones and empty points that can be reached directly (nobi), by a diagonal move (kosumi link), a one point jump (ikken-tobi link), a two point jump (nikken-tobi link), a small knight’s move (kogeima link), or a large knight’s move (ogeima link) are examined (total of 32 possible points).

The probability of connecting two stones is calculated empirically by playing out the sequences

11. There are 32 possible points to which a stone can be connected using the six types of links defined in the Go literature (see Figure 2.9).

of moves needed to determine whether the stones can be connected or whether the opponent can cut them. This process requires the use of lookahead, particularly when ladders are generated by cutting moves (e.g., cutting a kogeima link generates two ladders, one for each cutting point). If all possible cutting stones are captured, then the probability of connection is 100%. However, if the cutting stones can not be captured, a hand-tuned algorithm estimates the connection probability.

A massive amount of data is generated in calculating the probability map for each stone on the board for every hypothetical stone being evaluated. The process is very computationally expensive and requires a powerful PC.

4.2.2.2 Group Determination

Stones and strings are catalogued into groups: any stones that are 100% connected are considered groups. A tactical search on all strings with less than four liberties is used to determine which strings are definitely dead. Dead strings are not included as part of any group.

4.2.2.3 Eye Identification

Identifying eyes is achieved in a two step process: firstly, eye space is determined, and secondly, individual eyes are identified from the eye space.

A floodfill type algorithm is used to identify contiguous regions in which opponent connectivity probability is low. Points in those regions whose eight neighbouring points are mostly clear of opponent connectivity probability are considered to be eye space.

Individual eyes within a group’s eye space are identified by shape. There are several grades of eyes with the grade being a function of the opponent’s connectivity map. Eyes with a grade above a certain level (see next section) are considered to be true eyes.

4.2.2.4 Stone Safety

The safety of each group is determined by the number of true eyes it has. The safety process is repeated five times; each successive pass is more pessimistic about which points will eventually become true eyes (i.e., the threshold is raised for selecting which grade of eyes will eventually become true eyes). The final safety value for a group is the average of its safety scores on each of the five passes.

4.2.2.5 Radiation Function

The safety value for each stone is radiated to the nearby empty points in proportion to the connectivity probability map for that stone. The inverse of a stone's safety (i.e., 100% - safety) is radiated in the same manner and contributes to the opponents radiated value, that is, dead stones radiate full opponent safety. Black and white radiation (including inverse radiation) is summed at each empty point.

4.2.2.6 Territory

Black and white territory is estimated from the radiation values of the empty points. The difference between black and white territory is returned as the evaluation of the hypothetical move for which it was generated.

4.2.3 Performance and Timeline

Reiss' Go skill has progressed from 10 kyu in 1987 to 2 kyu in 1996. Reiss has programmed full-time on Go4^{++} in 1995 and also programmed full-time for one and a half years around 1986. Other than for these two periods since starting Go programming in 1983, Reiss has programmed part-time with the most effort usually expended in the two months leading up to a competition.

There have been many commercial releases of Reiss' programs by Oxford Softworks with another soon to follow. Notable releases have been Go Player Professional and an updated Windows version, Go Professional II, released in Japan.

Reiss characterized Go4^{++} as the weakest of the current strong programs at tactical situations requiring life and death analysis although it is good at creating moyos and gaining territory in the middle and endgame.

According to Reiss, other weaknesses are the absence of global lookahead, lack of a concept of "shape", and the reluctance of Go4^{++} to invade. Since the evaluation function pessimistically considers that the opponent moves next during tactical searches, Go4^{++} does not usually evaluate the likelihood of invading stones forming eyes as being favourable. To compensate for not invading well, Go4^{++} always plays its first move at the 3,3 point and also tries to live in as many places as possible to stop its opponent creating moyos early.

Reiss uses many empirically hand-tuned algorithms within Go4^{++} although no learning algorithms are used. Carefully crafted changes to these algorithms can improve the strength of Go4^{++}. For example, by making the connection algorithm more accurate, eye and territory evaluation also become more accurate, resulting in an increase in playing strength. Before the 2nd FOST Cup in 1996, Reiss tuned the performance of Go4^{++} by playing 700 test games against Handtalk, resulting in Handtalk's only defeat during that competition.

4.3 Game-Tree Search and AI Techniques in Competitive Go Programs

This section reports details of several competitive Go programs: HandTalk (*HT*), Go Professional (*GP*), Many Faces of Go, and Go Intellect (*GI*) (Chen, 1989; 1990; 1992), Explorer (*EX*) (Müller, 1995). Many of the details were communicated to me personally by the programmers[12] by email and at FOST'96 (Burmeister & Wiles, 1999). The programs are described with respect to state representation, move generation, goal states, evaluation, tactical search, and influence functions.

4.3.1 Position Representation

All programs represent stones and strings (unproblematic since they are deterministic). They also all represent groups, typically using pattern-based heuristics to determine connectivity between strings in order to determine the strings that belong to individual groups. Armies are also represented in EX and GI. The heuristics used to determine armies include influence (GI), and zones of territory (EX).

Important attributes of the objects represented on the board (e.g., groups, armies) include their life and death status (also referred to as *safety* or *strength*), territory value, number of eyes, and influence. The values of attributes in some cases are determined by tactical search (e.g., life and death).

Some components of MFG's representation are controlled by the evaluation function (e.g.,

12. I gratefully acknowledge the assistance of Ken Chen (GI), Zhixing Chen (HT), Martin Müller (Explorer), David Fotland (MFG), Mick Reiss (Go4^{++}/GP) in collating these details.

groups, connections or links, eyes, territory, and influence). GP's representation of the board is simply the representation formed by the evaluation function (e.g., groups, eyes, safety, territory).

4.3.2 Candidate Move Generation

In all programs, candidate moves are heuristically generated by a rule-based expert system and/or by pattern matching. Board points suggested as good moves are assigned a value. At the end of the move generation process, all the suggested values are summed (linearly or by weighted sum) for each board point. Typically the highest scoring board points are selected as moves for full-board evaluation.

The number of candidate moves that are evaluated at a full-board level varies between the programs: 4 in HT; between 1 and 12 in GI; up to 10 in MFG; and 50 in GP. The number of rules contained in a program varies: approximately 20 for GI; approximately 100 for EX; approximately 200 for MFG.

Patterns usually contain both low-level and high-level information. The low-level information pertains to the position of the black and white stones, which points must be empty, and the points whose occupancy were not of interest. The high-level information pertains to such things as number of liberties, safety, eyes, and territory. Pattern matching consists of matching not only the stone configurations but also any high-level requirements on stones or strings within the pattern. Due to the symmetry possible on a Go board, pattern matching a large number of patterns can become computationally expensive.

The number of patterns varies greatly between the programs: approximately 15 in GP; approximately 1,200 in MFG; and approximately 3,000 in EX. Some programs also contain a database of opening move patterns (joseki) (e.g., MFG contains approximately 45,000 joseki patterns).

4.3.3 Goals

Although heuristic estimations of territory are possible, territory is not always the best indicator of the merit of a position. In the early part of the game, possession of a large amount of territory may indicate an over-concentrated position that may be detrimental with respect to securing territory later in the game. Maximising influence rather than territory at the beginning of the game generally results in acquiring more territory at the end of the game. Influence is an example of a sub-goal that may be useful in determining the merit of a position.

There seems to be no consensus amongst Computer Go programmers about which sub-goals are the best indicator of a position's merit. Typically life and death status (safety) of groups and territory are included as goals or sub-goals. In HT, the emphasis is on tactical fighting as in MFG which also concentrates on connectivity, eyes, and group strength. GP concentrates almost exclusively on connectivity; almost everything is eventually derived (directly or indirectly) from the connectivity probability map.

4.3.4 The Evaluation Process

Evaluating Go positions is a slow process (e.g., MFG evaluates less than 10,000 full-board positions for the entire game at less than 10 evaluations per second, compared to 10,000-100,000 evaluations per second for chess programs). Limits are typically placed on the number of full board evaluations carried out due to tournament time constraints (e.g., MFG performs no more than 100 full-board evaluations in selecting its next move).

In GP, each of the 50 candidate moves are evaluated. In GI, evaluation is used during global search. If the value of one of the candidate moves is significantly better than the values of any of the others, it is played as the next move. However, if several of the candidate moves are approximately equivalent, global search, facilitated by evaluation, is used to determine which one to select. During evaluation, the safety of armies is used as a basis for assigning a value to each board point between -64 and +64 indicating the degree of Black and White control. The values are summed for all points to return the evaluation value. The global search does not examine any more than about six to seven moves and does not search any deeper than six ply.

4.3.5 Tactical Search

Tactical search is selective goal-oriented search and is used for a variety of purposes including determining whether strings are dead or alive (GP, MFG, EX, GI), whether connections are safe or can be cut (GP, MFG), whether eyes can be formed (MFG), candidate move generation (GI), and determining the life and death of groups (MFG, EX). Just as at the full-board level, tactical search also requires move generation and evaluation. A difference exists between full-board and tactical evaluation: evaluation within tactical search is with respect to the goals of the search (e.g., the number of liberties of a string). Due to time constraints, tactical search is typically limited with respect to the number of nodes, branching factor, and ply depth. MFG’s tactician provides a good illustration of the operation of tactical search and is described more fully in 4.1.1.2.

4.3.6 Influence Functions

Influence provides an indication of the potential control exerted by the stones of each player on empty points. By ensuring that the placement of stones is not over-concentrated at the beginning, a player maximises the chance of acquiring territory later in the game.

Influence is computationally modelled by influence functions as described earlier (Zobrist, 1970; Ryder, 1971; Chen, 1989). Typically, stones radiate influence (black and white stones radiating negatively opposite values) to surrounding board points with the radiated influence values from all the stones being summed at each board point (Black and White influence is kept separate in MFG). The influence radiated from stones decays as a function of distance: $1/2^{\text{distance}}$ in HT and GI; 1/distance in MFG. Programs that rely too heavily on influence do not play well (EX and GP no longer use influence although GP's radiation function is similar to an influence function).

The heuristic use of influence includes determining connectivity and armies (GI) and determining territory (MFG). In MFG, the initial value radiated from a group depends on its strength, that is, strong groups radiate more influence than groups that are weak or almost dead. This also means that dead stones radiate negative influence, that is, to the benefit of their opponent. In both MFG and GI, influence does not radiate through stones and in MFG it also does not radiate through enemy links.

4.4 Summary

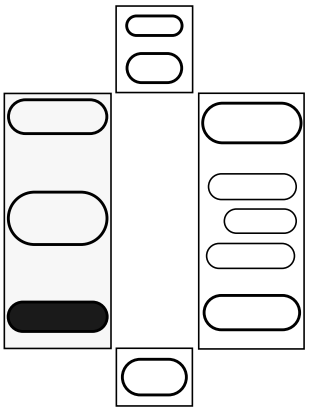
In summary, the competitive Go programs detailed in this chapter are similar in that they do not use full-width brute-force search, instead using some form of candidate move generation, forward pruning and local/tactical search. They differ in the basic model used and the emphasis placed on various aspects of Go, such as life and death analysis.

Chapter 5

Research Issues in Computer Go

Introduction

This chapter concludes Section II in which Go programs have been described in detail. The issues arising from Chapters 3 and 4 regarding knowledge, search and their interaction are discussed in section 5.1 and the conclusion in section 5.3 is that based on having successfully addressed the criteria AI1 - 4, Go provides a good research domain for AI.



5.1 Discussion

The traditional AI approach to game programming is brute-force search, in which the game-tree is searched as deeply as possible. The game positions are evaluated using a static evaluation function and as much of the game-tree as possible is pruned based on those evaluations. The brute-force search approach can be characterised as “knowledge-poor” because the emphasis is on evaluating as many nodes as possible and applying as little knowledge to each node as possible. Most effort is expended on making the search efficient rather than on gathering or applying more knowledge.

Prerequisites for successful brute-force search include an effective means of pruning the game-tree, that is, a static evaluation function. An effective static evaluation function is one for which the correlation between the intermediate positions and the game result is strong. For example, in chess, taking a major piece (without losing one in exchange) correlates strongly with winning.

Brute-force search is not used in the Computer Go field. There are two main reasons for its failure: the branching factor is very large, and static evaluation fails. Of these two problems, the difficulty associated with static evaluation is the most debilitating. This point is best illustrated by comparing the state-of-the-art of 9x9 and 19x19 Go programs. In general, Go programs are not significantly better at 9x9 Go than at 19x19 Go, even though the average branching factor is much smaller and closer to the branching factor in chess (around 55 compared with around 200).

Static evaluation fails in Go because of the necessity of acquiring information that is unavailable without search. In particular, life and death, and ko threats cannot be statically evaluated. Evaluating Go positions ultimately devolves to estimating territory. Other factors may be traded off against territory, but the accurate estimation of territory is essential for effective evaluation because ultimately securing territory is the goal of the game. Accurate estimation of territory requires accurate information regarding the life and death status of the strings and groups on the board. Accurate life and death information can not be gained statically —it requires search.

A further complication is that strings and groups can be alive in ko, that is, alive only if a ko is won. Safe strings or groups can be killed if a large enough ko threat can be found, that is, safe strings and groups may be sacrificed if larger strings or groups are the subject of a ko threat. Thus, the number and size of ko threats affects the determination of territory. However, determining the number and size of ko threats is tantamount to being able to completely evaluate the entire position - a classic “chicken-and-egg” problem.

Go programs rely more heavily on knowledge than chess programs. Since search is local or tactical, knowledge must be applied to forward prune the game-tree. Thus, search in Go can be characterised as knowledge-guided search. That is, Go knowledge is used to prune the game-tree rather than positional knowledge as is typically used to prune the game-tree in the brute-force search approach. The distinction here is between Game knowledge (or game theory) and game-specific knowledge (i.e., knowledge about the current game). In chess, the static evaluation function encodes Game (chess) knowledge that is used to acquire game-specific knowledge about the current game which is then used to prune the game-tree and ultimately to select the next move. Go requires the possession and use of more Game (Go) knowledge than chess programs since the game-tree is pruned prior to acquiring game-specific knowledge.

5.1.1 The Interaction of Knowledge and Search in Go Programs

Knowledge

Current Go programs use many different types of knowledge, and different methods for representing and utilising the knowledge they possess. The lack of an effective static evaluation function combined with the combinatorially large search space has required Go programs to address the use of knowledge in search. The following examples illustrate, but do not exhaust, the types of knowledge and techniques currently used:

- Influence functions were an early attempt to succinctly represent issues of territory.
- Joseki represent known optimal play in restricted aspects of the opening game. The skill in using joseki for both humans and programs lies in selecting an appropriate joseki for a given situation.
- Procedures to detect ladders reflect knowledge of the special circumstances that arise in cutting/connecting (e.g., kogeima links). A ladder can require reading up to 60 plies ahead, but each ply is constrained (by the ladder) to a single candidate move.
- Pattern databases use bitmaps, shape, an expert system or hand-coded decision trees to encode knowledge about significant features. The potential number of significant patterns in Go is combinatorially large and each program deals with pattern knowledge in an individual way.
- Life and death and ko are the most complicated aspects of Go and require the use of knowledge to forward prune the search tree. Ko is still a major challenge to Go programs: there are techniques that work reasonably well in closed local situations (Müller, 1997) but their efficacy in open local situations is a matter of ongoing research.

Search

The key task of a Go program is to select the next move. A small number of candidate moves are typically generated and then evaluated (e.g., Go4^{++} generates 50, MFG selects from 10, as described in section 4.3.2). From the perspective of move generation, the major search issues involve evaluation of the generated positions rather than full-board search. Search in Go programs is tactical or local, that is, search is used to determine answers to local questions rather than to global questions. The uses of tactical search include life and death analysis, cutting/connecting, and determining eyes.

Interaction

The evaluation of candidate moves requires a search process but that search process itself interacts heavily with the knowledge available to the Go programs.

An example that illustrates the interaction between knowledge and search is provided by life and death searching in MFG which is goal directed (e.g., to kill a group). Moves are generated at each node, that is, knowledge is used to forward prune the search-tree. The static life and death evaluator (which uses the tactician) is used at each node to determine whether the goal has been achieved. Thus, tactical search is used to obtain the knowledge needed to guide the life and death search. During the life and death search process, the life and death status of a group may change even if it is not actually examined by the life and death search engine. Such a change is implemented by the static life and death evaluator, that is, the static life and death evaluator may change the life and death status of a group as a side-effect of determining whether the life and death search has reached its goal. Thus, the state of the MFG's knowledge is altered as a result of life and death searching.

From the above illustration, it can be seen that knowledge and search interact with each other to a large degree in at least some Go programs.

5.1.2 Other Approaches Used in the Computer Go Field

Besides the programs described in Chapter 4, there are other approaches to Go programming that merit attention.

5.1.2.1 Combinatorial Game Theory

Combinatorial game theory (*CGT*) is a mathematical formalism that provides a divide-and-conquer method of analysing games in terms of smaller, independent sub-games. CGT has been applied theoretically to Go (Berlekamp & Wolfe, 1994) and also practically in Explorer (Müller, 1995, 1997). Müller has extended his work into the analysis of ko (Müller, 1997). One drawback of CGT is that it makes certain assumptions that cannot be satisfied during most of the game and thus it is applicable mainly in the endgame. Current research involves looking at ways to use heuristic conditions to relax the assumptions and make the approach useful earlier in the game.

5.1.2.2 Learning in the Computer Go Field

There are three programs that are active on the Computer Go Ladder[1] which employ learning algorithms.

The most successful learning program is Gogol by Cazenave (1996a, 1996b, 1996c; Nigro and Cazenave, 1996) which is currently 5th on the 9x9 ladder and 6th on the 19x19 ladder. Gogol is a rule-based program that uses 1st order logic and an extended combinatorial game theory to represent Go knowledge. The rules are a list of premises and a list of 1 or more conclusions and are learned by a symbolic learning algorithm (as opposed to a sub-symbolic algorithm such as a neural network). Gogol is written in C++ and consists of 40,000 lines of code and several thousand rules.

Gogol uses an expanded form of combinatorial game theory in which the win (G) and lose (P) values are augmented by an uncertain value (I). If an accurate determination of the combinatorial value of a sub-game is impractical due to the intractability of searching the sub-game, its combinatorial value becomes I. Each combinatorial game has a goal associated with it (e.g., connect, capture, make eye) and its combinatorial value reflects whether it can be achieved (G), not achieved (P), or whether it is not possible to determine (I).

A deductive learning algorithm is used to learn the rules and consists of four phases: problem solving, regression, explanation, and generalisation:

- Problem solving: The problem solving phase is a deductive process in which all rules are fired to deduce new facts. The problem solving phase finishes when no more facts can be deduced;
- Regression: Knowledge about the unknown combinatorial value (I) is contained in regression rules that may allow further facts to be deduced;
- Explanation: Facts generated about the position after the move (post-move facts) are explained (i.e., a rule is generated) in terms of facts about the position before the move (pre-move facts) and the fact of the move itself; and
- Generalisation: The rules generated in the explanation phase are generalised so that they will match more than the specific case that gave rise to them. Generalization is achieved by

1. The Computer Go Ladder is an informal online competition which is for Go programmers to have fun competing against each other and also encourage progress in the Computer Go field. As at September 1999, the ladder can be found at http://www.cgl.ucsf.edu/go/ladder.html

replacing constants (e.g., specific board locations) with variables in the explanation rules.

The premises of the rules are compiled into an efficient order using metaknowledge about the number of times each premise has been matched. Speedups of 14,000 times have been achieved on a 9x9 board using the compilation technique. The logic language is compiled into C++, resulting in a speedup of 60 times when matching rules.

The two other learning programs active on the Computer Go Ladder are:

- Gobble (Brügmann, 1993) which uses simulated annealing and contains no Go knowledge beyond the rules; and
- NeuroGoII (Enzenberger, 1996) is a backpropagation neural network which uses a temporal difference algorithm TD(0) and integrates expert Go knowledge into the program along with the network.

5.1.3 Go as a Domain for AI Problems

Go has been used as a research domain for investigating AI issues other than game-playing. That is, the objective of the research is not necessarily to produce a competitive Go program but to solve AI problems that also occur in Go.

5.1.3.1 Pattern Recognition

Go is a visual game: the Go literature is full of references to patterns, shape, and visual imagery. According to Saito and Yoshikawa (1997):

- perception is needed to segment the board into meaningful chunks;
- patterns have to be recognised from any perspective since, unlike chess, Go does not impose any orientation on the board;
- minor displacements in stones may cause radical changes in the meaning of a pattern;
- both language level inference and pattern recognition play an important part in Go; and
- Go programs separate recognition and inference (e.g., lookahead) whereas human players tend to integrate them.

Another aspect to pattern matching in Go is that humans do not match simply on point occupancy (i.e., black, white, empty), but also incorporate high-level aspects such as strength, and thickness into the pattern recognition process.

Most Go programs have at least a rudimentary pattern recognition capability. Pattern recognition research in the Computer Go field includes Zobrist (1970) (see section 3.1.1); Reitman and Wilcox (1975, 1978), Wilcox (1988) and Reitman, Nado, Wilcox (1978) (see section 3.1.3).

5.1.3.2 Planning

Planning is a natural part of adversarial games. According to Saito and Yoshikawa (1997), the unique characteristics of Go with respect to planning are:

- the extensiveness and complexity of plan interaction;
- plans are closely related to stone configuration and chunks; and
- plan conception is affected by changes in how stone configurations are viewed.

There have been several research projects encompassing planning in Go:

- Lehner (1981, 1983) used a representative search to provide information about many potential move sequences that satisfy a strategic plan by examining only one of those sequences. Lehner's results suggested that representative lookahead was not alone sufficient to evaluate strategic plans, however, it could quickly reject strategically unsatisfactory plans. He saw the use of representative search as providing a means to reduce the number of moves that would be passed on to a more precise tactical lookahead mechanism.

- Reitman (Reitman, Kerwin, Nado, Reitman, & Wilcox, 1974) saw Go as a domain in which it was possible to extend the AI conceptions of planning and goals to ill-structured, dynamic, multi-person, resource bound and highly interactive problems.

- An adversarial planning architecture based on hierarchical task network has been used to provide a framework for goal-directed play in Go (Willmott, 1997; Willmott, Richardson, Bundy & Levine, 1998).

5.1.3.3 Problem-Solving

Games are full of problem solving activity. Saito and Yoshikawa (1997) note that problem solving in Go is found at many levels: life and death, strategic, territory versus influence.

Reitman and Wilcox (1979) modelled a skilled Go player's tactical problem solving ability in a component of their INTERIM.2 program, PROBE (described in section 3.1.3.3). Selective lookahead was used to create a best-first tactical problem solving strategy which emulated human selective search.

5.1.3.4 Opponent Modelling

There seems to be nothing in the literature reporting opponent modelling in Go. However, it is a commonly observed fact that humans adjust to the play of a program opponent whilst programs do not adjust to the play of a human opponent. This disparity makes it hard to accurately rank a program's ability since after the first few games against a program, a human opponent usually takes advantage of its faults with an apparent decline in its ranking. The issue of opponent modelling is important in chess and Shogi[2], and can be expected to be seen in Go programs in the future.

5.1.3.5 Learning in AI

Go has been used as a research domain for various machine learning studies:

- Temporal difference learning (TD) and TD(λ) has been used to train neural networks to evaluate Go positions (Schraudolph, Dayan & Sejnowski, 1994; Chan, King, & Lui, 1996) and TD(0) has been used to integrate expert Go knowledge into the program along with the network (Enzenberger, 1996).

- The Golem Go program (Enderton, 1991) uses a neural network to recognise professional Go players' moves based on the local context around the moves.

- The issue of whether a program should play the move that provides the best chance of winning the game or the move that provides the most useful information for learning was examined by Pell (1991). The results of his investigation into exploratory learning in Go showed that it is best for a learning program to play the move providing the most information.

- Go was used as a domain for researching machine learning by Stoutamire (1991). A simple error function was developed which, rather than incorporating tactical or strategic Go

2. Japanese chess.

knowledge, was derived from a database of expert games. Stoutamire then developed a classification technique called pattern preference which automated the process of deriving patterns that were representative of good moves. Due to exponential growth of the pattern database, a hashing function was developed that had the nice property of graceful degradation as memory requirements increased.

5.2 Future Directions

The acquisition, integration and use of knowledge is critical to the progress of the Computer Go field. Since human players are the best source of Go knowledge, the possession of knowledge associated with human competency in certain aspects of Go would be beneficial for Go programs. In particular, the knowledge associated with the following Go skills would be helpful:

- *Pattern matching:* Go programs typically perform pattern matching based on physical stone location. Human pattern matching involves both physical stone location and also high-level concepts (e.g., thickness). Incorporation of high-level knowledge into pattern matching would be of tremendous benefit to Go programs.

- *Integration of josekis:* Go programs typically choose joseki by pattern matching and/or evaluation at their end points. The choice of joseki by human players is influenced by the need to integrate them with the rest of the board position. Such knowledge would enable Go programs to select joseki which further the program's local, global, short- and long-term goals.

- *Ko fights:* Current Go programs are not good at ko fights with their ko threats being typically limited to atari (capture) threats. Research into improving ko fighting ability is still at a preliminary stage (Müller, 1997). Good Go players try to simultaneously maximise their potential ko threats and minimise the potential ko threats available to their opponent. Once a ko fight has started, good players play ko threats that contribute both to winning the ko and advancing their overall game plan. Without the knowledge to identify and capitalise on ko threats, Go programs will be susceptible to losing control of the game when confronted with a ko fight.

- *Candidate move generation:* Go programs typically employ rule-based expert systems and/or pattern matching to generate candidate moves to evaluate. It is common for the best moves to be forward pruned before being considered as candidate moves. Good human players exhibit phenomenal skill in considering only the 2 or 3 best moves before selecting their next move. Since Go programs by necessity must employ forward pruning, knowledge that allows them to concentrate on the very best candidate moves would be extremely useful.

Cognitive Psychological investigations of Go players could prove very useful in eliciting knowledge and techniques that may be useful to Go programmers.

5.3 Conclusions

The assessment of Go as a research domain for AI conducted in Section III was with respect to the criteria AI1 - 4 which are repeated below:

AI1 *To what extent does the domain of Computer Go facilitate the investigation of knowledge, search and their interaction?*

AI2 *What type of AI research issues can be investigated in the domain of Computer Go?*

AI3 *Do interesting research questions arise from investigations of search, knowledge, and their interaction within the domain of Computer Go?*

AI4 *Do generalisations useful to AI arise from research conducted within the domain of Computer Go? What types of generalisations can be made beyond the domain of Computer Go?*

AI1

The survey of early work in the Computer Go field, the analysis of the state-of-the-art of contemporary programs, and the case studies of Many Faces of Go and Go4^{++} demonstrate that the issues of search, knowledge and their interaction are core issues in Computer Go. Thus, criterion AI1 has been successfully addressed.

AI2

Section 5.1.3 discussed the use of Go to investigate AI issues other than game-playing such as pattern recognition, planning, problem-solving, and opponent modelling. Thus, criterion AI2 has been addressed.

AI3

Criterion AI3 was addressed in section 5.1.2 by illustrating some of the new approaches which have been developed in order to overcome the difficulties of managing the interaction between search and knowledge in Computer Go.

AI4

Criterion AI4 is addressed above in section 5.1.3 by demonstrating that generalisations from within the research domain impact outside the domain itself, and that the issues of the research domain exist elsewhere in AI. The generalisations from within the domain of Go have the potential to impact outside the domain, and that the issues within the domain of Go exist elsewhere in AI. Thus, generalisations useful to AI do arise from research conducted within the domain of Go and there are potentially many cognitive studies that could be conducted using Go as a research domain which may similarly give rise to useful generalisations. Thus criterion AI4 has been successfully addressed.

Go Provides a Good Research Domain for AI

In summary, having successfully addressed the criteria AI1 - 4, it can be said that the work reported in Section II demonstrates the game of Go provides a good research domain for AI.

Section III

Studies in Memory and Learning for Go Sequences

Meta-level Rationale

Section III is concerned with assessing Go as a research domain for Cognitive Psychology by addressing the assessment questions CP1 - 4 from 1.2.2, which are repeated below:

CP1 *To what extent does the domain of Go facilitate the investigation of memory for moves? To what extent does the domain of Go facilitate the investigation of the impact of inference on memory performance?*

CP2 *What type of Cognitive Psychology research issues can be investigated in the domain of Go?*

CP3 *Do interesting research questions arise from investigations of memory for moves within the domain of Go? Do interesting research questions arise from investigations of the impact of inference on memory performance within the domain of Go?*

CP4 *Do generalisations useful to Cognitive Psychology arise from research conducted within the domain of Go? What types of generalisations can be made beyond the domain of Go?*

Overview of Section III

Addressing question CP1 involves selecting specific issues to investigate. In Chapter 6, following reviews of the seminal memory studies in chess and cognitive studies in Go, the initial research questions of the project are established as:

- what factors affect memory performance for Go moves?; and
- to what extent does inference impact on memory performance in Go?

Two experimental memory studies (reported in Chapters 7 and 8) were conducted on participants ranging from beginners to masters to investigate these initial research questions.

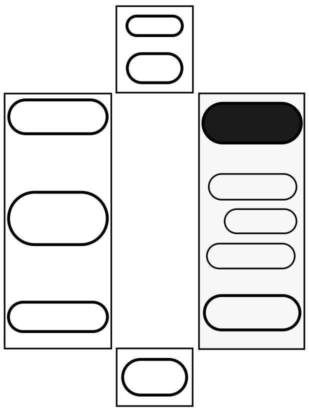
During the studies reported in Chapters 7 and 8, an interesting phenomenon was observed, namely implicit learning by the participants. The implicit learning that was observed was pursued in a further experimental study reported in Chapter 9, enabling the question CP3 to be addressed.

Chapter 10 completes the Section by concluding that Go provides a good research domain for Cognitive Psychology. Within the discussion in that chapter, the questions CP3 and CP4 are addressed.

Chapter 6

The Cognitive Psychology of Go: Literature Review and Experimental Paradigms

Introduction



The relatively small body of literature concerning Go research in Cognitive Psychology (see section 6.2) is best appreciated in the context of the seminal memory studies in chess conducted by De Groot, and by Chase and Simon (see section 6.1). Indeed, a natural line of inquiry at the outset of an exploration of the Cognitive Psychology of Go is the similarities and differences between Go and chess, a theme that is addressed in sections 6.2 and 6.3. Investigating the differences between Go and chess was the initial motivation for unpublished pilot studies conducted at the beginning of this project, however, other interesting issues arose that changed the direction of the project.

In the seminal chess studies, much more attention was paid to memory for game positions than to memory for game moves (i.e., the sequence of moves comprising a game). The neglect of memory for game moves seems somewhat surprising considering that game players (e.g., chess players) spend so much time studying and memorising lines of play (e.g., opening gambits) and past games. Thus, in the unpublished pilot studies conducted at the beginning of this project, both memory for Go positions and Go moves were investigated.

As a result of the pilot studies, it became apparent that generalisation[1] impacted much more strongly on the reproduction of Go positions than had been suggested in the chess literature. Although it was unclear how the process of generalisation was achieved, it was surmised to be

1. The term generalisation is used in this thesis in the behavioural sense in that knowledge gained in one situation is used to perform better in subsequent situations.

some form of inference. Thus, both the seminal chess literature and the Cognitive Psychology Go literature was examined to investigate the issue. This chapter concludes with discussion in section 6.3 of the literature surveys provided in sections 6.1 and 6.2 including a justification of the initial research questions which are pursued in the experiments comprising Section III.

6.1 Seminal Cognitive Studies in Chess

The seminal cognitive studies using chess as a research domain were conducted by De Groot (1965[2]) and by Chase and Simon (1973a), and have been described as “citation classics” by Charness (1992). Related studies include later work by De Groot (1966) and De Groot and Gobet (1996[3]), and associated work by Chase and Simon (1973b). These studies have been of enormous significance to memory research within Cognitive Psychology:

> *If the game of chess is, as Charness (1989) proposes, “the drosophila of psychology” (p. 184), then the memory paradigm pioneered by de Groot (1966) and popularized by Chase and Simon (1973a, 1973b) and others is its “microscope”.* Cooke, Atlas, Lane & Berger (1993, p. 321)

The survey that follows focuses on the issue of inference which may in the first instance be regarded to be a high-level cognitive action such as logical deduction or problem solving. This view will be refined in light of the results of the experimental studies to also encompass low-level generalisation carried out within the memory system (see Chapter 8). The survey also pays attention to memory for moves, and since memory tasks are the necessary vehicle for investigations within the memory paradigm, special attention will also be paid to the memory tasks used.

6.1.1 Perception and Memory: De Groot

De Groot conducted extensive investigations of chess players’ thought processes, collecting a large number of verbal protocols. Two of the main conclusions from De Groot’s work on chess players in the 1930s and 1940s was that the difference between master chess players and weaker players was in their perceptual abilities and that the basis of chess mastership was largely a matter of memory (De Groot, 1965). De Groot was interested in how chess perception worked,

2. English translation of De Groot’s PhD thesis, *Het denken van den schaker,* published in 1946 (Second Edition 1978).
3. Together with Gobet, De Groot completed and reanalysed his earlier work reported in 1966.

the extent to which it was instrumental in chess achievement, how chess memory was acquired, and the structure and function of chess memory (De Groot, 1966). The latter project remained dormant for several decades and was completed with Gobet in the 1990s (De Groot and Gobet, 1996).

6.1.1.1 Perception

According to De Groot (1966), domain skill (e.g., chess skill) makes information normally obtained by abstraction or inference available perceptually. That is, abstraction and inference is replaced by perception. Therefore, a “given” problem would be perceived differently by experts and novices and could not thus be described as “given”. De Groot’s explanation for the difference in perceptual abilities was that the master players possessed a different (and better) coding system. As evidence, he cited results from an early Russian study[4] which showed that master chess players do not generally perform better than novice chess players on visual memory tasks outside the domain of chess.

De Groot (1966) made calculations of how much information was being coded by his master participants. The upper limit of information necessary to encode any arbitrary chess position was calculated to require 143 bits to encode. De Groot generated the prototypical master chess position at the 21st move from 192 games and used this result together with other calculations to reach an estimate that 40 bits would be needed to encode a prototypical master chess position at the 21st move (the reduction from 143 to around 40 bits resulted from various factors such as the typical reduction in pieces during the first 20 moves). From this result, De Groot concluded that the master’s perceptual abilities were not due to a general knowledge of chess probabilities.

6.1.1.2 Memory

De Groot (1966; De Groot and Gobet, 1996) performed a perception experiment on 8 master and 8 weak chess players who had two tasks: position reproduction and position construction[5]. In the position reproduction task, the participants were shown a chess position for 5 seconds (De Groot, 1966, p. 43) and then given as long as they liked to concentrate on the position they had just seen. They were then given the chess pieces they had just seen and asked to reproduce the position. After their first attempt, the incorrect pieces were removed and, without again seeing the original position, they were asked to place the pieces on the board. This process was

4. Djakow, I. N., Petrowski, N. W. and Rudik, P. A. (1927) *Psychologie des Schachspiels*. de Gruyter, Berlin as cited in De Groot (1966).
5. In his earlier work, De Groot referred to this task as “blind” position guessing (De Groot, 1966).

repeated until all the pieces were correctly placed or until 12 attempts had been made. In the position construction task, the participants were not shown the position, but were given chess pieces and asked to place them where they thought they should be. The wrongly placed pieces were given back to them for replacement as in the position reproduction task. The participants were exposed to 5 boards in each task (the boards were from the 192 games described in the previous paragraph).

The results were consistent with De Groot’s (1965) earlier findings that master players exhibited better performance than the weaker players on both tasks. The master players performed as well on the construction task as the weaker players did on the reproduction task, with the weaker players gaining little advantage from the 5 second exposure to the position. The marked difference between the masters’ performance on the two tasks led De Groot to conclude that prior chess knowledge contributed little to the master players’ ability to reproduce chess positions (De Groot, 1966). De Groot further claimed that his results gave evidence that the specific perceptual achievement of chess masters was due to their superior coding system.

The difference between perception and abstraction (inference) were also briefly discussed by De Groot (1966). He speculated on the existence of “dynamic perception” - the ability to “see” a move possibility. De Groot suggested the possibility that the dynamics of a position could be either “seen” or “inferred”.

6.1.1.3 Performance Issues

De Groot and his co-author, Gobet, disagree on the role of problem solving (inference) in chess memory (De Groot & Gobet, 1996). De Groot describes the process of perceiving, retaining, and reproducing a position as a steered problem solving process that uses memory. Gobet, although recognising the role of procedural knowledge, places more emphasis on automatic recognition mechanisms, a view that is in accordance with those espoused by Chase and Simon (1973a, 1973b).

In De Groot’s protocols (De Groot & Gobet, 1996), the master participants reported sometimes employing both “pure” guessing and “educated” guessing (e.g., using information provided by perceptual images). De Groot suggests that “pure guessing” is a relative concept since it seems evident that the master participants use their first-order and higher-order location probability knowledge to help with the placement of pieces about which they are uncertain.

Further to the issue of guessing, De Groot (De Groot & Gobet, 1996) suggested that in the construction task, both the master and weak participants employ guessing with common sense, although at different levels. In the reproduction task, De Groot suggested that the weak participants performed in a similar manner to how they performed in the construction task, that is, they used common sense guessing in the absence of any informative mnemonic basis for the placement of pieces. De Groot appealed to the notion of common sense informally, indeed, common sense continues to resist adequate definition and implementation.

In summary, according to De Groot, chess skill involves perception and memory. Although of the view that chess skill makes information once obtained by abstraction (inference) available perceptually, De Groot also reported that intelligent/educated guessing (which may be interpreted as a form of inference) was employed by his participants.

6.1.2 Chunking: Chase and Simon

Building upon De Groot's work, Chase and Simon investigated chess skill and the chunking of chess information in a series of experiments (Chase & Simon, 1973a, 1973b). In so doing, they introduced many new methodological techniques for studying memory and in particular, for studying chunking.

6.1.2.1 Recall of Positions

Chase and Simon hypothesised that chess information was encoded into chunks and conducted an experiment to identify those chunks (Chase and Simon, 1973a). There were two tasks: the memory task was similar to De Groot's reproduction task whilst in the perception task, participants were asked to reconstruct a chess position that was in full view onto an empty chess board. However, whereas De Groot used real chess positions exclusively, Chase and Simon's stimuli contained both real and random chess positions. Their participants consisted of a chess master, a class A player, and a beginner. Their results were consistent with De Groot's earlier findings that the amount of information extracted from a chess board is related to chess skill. They also found that there was no facilitation for chess skill for random[6] chess positions.

6. Chase and Simon mistakenly attributed a similar finding to De Groot (1965) although it was De Groot's student, Jongman, together with Lemmens who made the unpublished finding (Vincente & De Groot, 1990).

Chase and Simon hypothesised that in the memory task, long pauses between the placement of successive pieces would mark between-chunk boundaries and that short pauses between the placement of pieces would indicate that they were within the same chunk. They further hypothesised that in the perception task, pieces placed between glances would be from the same chunk. Five chess relations were identified: attack, defence, proximity, common colour, and same piece. The relations between successive pieces were examined to try and confirm that the pieces in the chunks generated by these hypotheses were actually perceptual chunks. What Chase and Simon expected to see was that the chess relations between successive within-chunk pieces would be similar whereas the relations between successive between-chunk pieces would be dissimilar.

Chase and Simon's results showed that an interval of more than two seconds between piece placement was sufficient to segment pieces into separate chunks. They also found that within-chunk pieces generally had many relations between them and that the relations between successive between-chunk pieces were distributed randomly.

In examining chunk size and memory span, Chase and Simon expected that the number of chunks remembered by the participants would not exceed the standard capacity of short-term memory and that the master's chunks would contain more pieces than the other participants. Their results confirmed that the master's chunks were larger than those of the other participants. However, they were surprised to find that the master also remembered more chunks than the other participants. Overall, chess skill was reflected in the speed with which chunks were perceived in the perception task and the size of the chunks in the memory task with the number of chunks encoded being within the standard capacity of short-term memory (7 ± 2).

Chase and Simon advanced three possible explanations for the master remembering more chunks than the other participants. Firstly, the participants would sometimes determine where pieces should be from their general chess knowledge rather than actually remember where they were. That is, inference or problem solving played a part in their performance. The master's performance was particularly facilitated by this ability which may have contributed to the master remembering more chunks than the other participants. Secondly, the master's long-term memory is structured so that information associated with chunks would cue the retrieval of other chunks. Thirdly, chunks are not of uniform size. Highly overlearned information structures may tax short-term memory less than information structures that are not so well learned.

6.1.2.2 Reconstructing Positions: Pennies-guessing Task

To show how reconstructing a position from partial information is relatively easy, Chase and Simon referred to the pennies-guessing task in which all the pieces on a chess board are replaced with pennies and the participants must substitute chess pieces for the pennies to "reconstruct" the position (Chase & Simon, 1973b). Although it is unclear from the text[7] whose data they are citing or what the experimental design for the pennies-guessing experiment was, they claim that the master performed almost perfectly and the class A player performed at over 90%.

6.1.2.3 Recall of Moves

Chase and Simon also conducted relatively small studies into long-term memory for games and immediate recall of moves. Of particular interest is the latter study in which their participants were given 30 seconds to study an initial position (from around Move 20). Ten moves (20 plies) were read out in standard chess notation at a rate of one move every 5 seconds. The participants played the moves on the board as they heard them. The stimuli included 10 real and 10 random sequences (the initial position was always from a real game). Five seconds after the last move was read out, the board was removed and a board containing the initial position was placed before the participant (the intervening interval being around 10 seconds). The participants then replayed the sequence of moves, and were only supplied with correct moves if they made a mistake or were unable to remember the next move after 10 seconds.

The results showed that, as for recall of positions, recall of move sequences was related to chess skill. Surprisingly, this result pertained to both real and random sequences although recall of real sequences was superior to recall of random sequences. The master recalled the sequences almost perfectly. Chase and Simon observed that performance in the move recall experiment relied less on short-term memory than performance in the position recall experiment because the participants had almost 2 minutes to view the position as it was read out.

Chase and Simon surmised that move sequences are probably organised into little episodes related to a goal and also suggested the possibility that the episodes were organised hierarchically. They hypothesised that chess players store games as a series of quiet positions[8]

7. The text appears in a footnote on page 238 of Chase and Simon (1973b).
8. Positions in which no exchange of pieces is taking place.

together with information that enables transition between positions. The information needed to transform positions into subsequent positions would be stored in chunks in long-term memory. Thus, they concluded that it is not necessary to explicitly store intermediate positions since they can be regenerated from plausible moves stored with the quiet positions.

6.1.2.4 Chess Expertise

Chase and Simon's explanation for chess skill was that the labels of chess patterns held in long-term memory (estimated to be between 50,000 and 100,000) are placed in short-term memory by a salient pattern detector (i.e., skilled chess processing takes place at the perceptual level), and that this process also underlies chess memory performance (Chase & Simon, 1973b). They concluded that there was evidence that much of the skilled chess processing takes place at the perceptual level. However, it could be argued that the behaviour they observed resulted from the organisation of the information at recall. They tested this hypothesis by manipulating the input and output stimuli: the input and output stimuli were either real chess pieces or diagrammatic symbols for pieces. They found that there was no difference for output stimuli. That is, the participants' performance was qualitatively similar regardless of whether they placed real pieces or drew diagrammatic pieces on paper. They did observe an initial difference when the input stimuli was diagrammatic, however, this effect disappeared after approximately an hour of practice which they interpreted as resulting from perceptual learning. They interpreted the absence of an effect for the output stimuli together with the initial effect for the input stimuli as evidence that skilled chess processing takes place at the perceptual level. Thus, their view that inference plays only a minor role in chess memory performance was supported. The implication is that memory performance is inextricably linked to perceptual activities such as pattern recognition and that inference plays only a minor role (if it plays one at all) in chess memory performance.

6.2 Cognitive Studies Using Go as a Research Domain

The seminal studies surveyed in the previous section were but the first of a large number of cognitive studies using chess as a research domain. In contrast to chess, relatively few studies have been conducted in Cognitive Psychology using Go as a research domain. Eisenstadt and Kareev (1975, 1977) investigated the use of internal representations in problem solving (reviewed in section 6.2.1). Judith Reitman (1976) investigated chunking in Go by replicating some of Chase and Simon's studies in Go (reviewed in section 6.2.2). Saito and Yoshikawa have conducted Cognitive Science investigations into Go since 1993, publishing their later work in English (Saito & Yoshikawa, 1995, 1996, 1997; Yoshikawa & Saito, 1997a, 1997b). Their investigations involve collecting verbal protocols and eye tracking data to model Go players' thinking (reviewed in section 6.2.3).

Each of these studies is essentially independent since, as yet, no critical mass of research using Go as a Cognitive Psychology research domain has emerged. As in the survey of the seminal chess literature in the preceding section, the survey will focus on the issue of inference.

6.2.1 Internal Representations: Eisenstadt and Kareev

Eisenstadt and Kareev (1975, 1977) used Go and Go-moku (a much simpler variant of Go) in a study of how internal memory representations are used during problem solving and board scanning. Their findings indicated that the internal representations formed by their participants depended on which game they were playing (Go or Go-moku) and incorporated both perceptual information and abstract information that resulted from problem solving processes. They hypothesised that familiar patterns of stones are stored internally as active procedural[9] representations. From observation of their participants' search behaviour, they concluded that players conduct searches for familiar patterns and also that searches were triggered by new patterns that they observed. This led them to suggest that human game playing search is a convergence of top-down hypothesis driven search and bottom-up data driven search. They also offered an explanation for the "progressive deepening" search observed in human problem solving: imaginary moves considered in short term memory cannot be easily erased which leads to a re-initiation of the search rather than the retraction of the last step.

In summary, it would appear from Eisenstadt and Kareev's work that the issue of representation is a fundamental one for understanding problem solving behaviour, particularly with respect to search. Their work would also seem to indicate that models of human problem solving should not rely solely on static representations but should also incorporate the ability to form active procedural representations.

9. Eisenstadt and Kareev use this term in the computer science sense (i.e., a procedure or function containing computer code) rather than in the psychological sense.

6.2.2 Chunking: Reitman

Judith Reitman (1976) replicated Chase and Simon’s (1973a; 1973b) work on chunks (see section 6.1.2) using Go to examine the partition of recall and reproduction data into chunks on the basis of inter-response times. Reitman’s experiment was conducted on a Go master and a Go beginner, and the memory and perception tasks were similar to Chase and Simon’s except that the stimuli consisted of Go positions on a Go board. There were two types of stimuli: meaningful patterns were taken from a Japanese Go journal and random patterns were generated so that they consisted of clusters of similar size and composition to those used in the meaningful patterns. The patterns themselves were presented in the lower right quadrant of a 19x19 board.

Reitman’s results were consistent with Chase and Simon’s in that the master’s performance for meaningful stimuli was superior to the beginner’s and the performance of both participants on the random stimuli was comparable. However, Reitman also found that chunks in Go are not organised linearly or as nested hierarchies as in chess, but as overlapping clusters.

Reitman concluded that inter-response times are not a stable diagnostic tool for determining chunks in Go (and perhaps in other domains) since the assumptions associated with their use are violated. In trying to explain why a single inter-response time could not be found to accurately segregate chunks in both the memory and perceptual data, Reitman discussed the issue of whether the two tasks elicited different chunking performance from the participants. In the perception task, Reitman postulated that the participants encoded chunks at the lowest level since the stimuli were always present, whereas in the memory task, the participants encoded chunks at the highest level to aid memory in the reconstruction phase.

Reitman also made the point that during the memory task, the participants would be more likely to guess than during the perception task and that the master’s guesses were more likely to be correct than the beginner’s. Reitman further surmised that some of the inter-response times may have indicated delineation between problem solving activity (e.g., figuring out the composition of a pattern using the initial stones as clues) rather than chunk boundaries. Since positions in Go change incrementally rather than by displacement as in chess, Reitman concluded that memory performance in Go was more likely to be influenced by guessing than in chess. Reitman suggested that inter-response times are not suitable for determining chunks in tasks that do not minimise the participants’ ability to make intelligent guessing.

In summary, Reitman’s research shows that the chunking theory proposed by Chase and Simon in chess does not necessarily generalise outside of the chess domain. Thus, Chase and Simon’s view that pattern detection forms the basis of chess memory performance may not be adequate for explaining Go memory performance. In particular, intelligent guessing (which may be a form of inference) may contribute more towards Go memory performance than chess memory performance.

6.2.3 Modelling Go Problem Solving and Perception: Saito and Yoshikawa

Saito and Yoshikawa have conducted Cognitive Science investigations into Go since 1993 (Saito & Yoshikawa, 1995, 1996, 1997; Yoshikawa & Saito, 1997a, 1997b). Saito and Yoshikawa have worked on developing a model of Go players’ thinking by investigating both their problem solving behaviour and their perception. The long-term goal of their research is the production of a Go program based on their model and Yoshikawa is currently working on a Go program with another colleague (personal communication[10]).

Their initial working hypothesis was a “two box” model that comprised one or more iterations of candidate move generation followed by candidate move evaluation before a move was played. Their investigations involve collecting and analysing both verbal protocol and eye movement data: candidate move evaluation was investigated using verbal protocols; candidate move generation was investigated using eye movement tracking.

6.2.3.1 Candidate Move Evaluation and Selection (Verbal Protocols)

Verbal protocols were taken from amateur Go players as they played Soudan-Go[11] (Saito & Yoshikawa, 1995). Analysis of the verbal protocols revealed that the use of language plays an important part in the selection of moves and that although verbalisation causes players to play more slowly, it does not have a detrimental affect on their performance. Language was commonly used by the participants to name (i.e., to provide a Go specific term for) their own candidate moves, their opponents’ move, and board situations.

10. I was fortunate enough to spend 3 months in their research group at the Basic Research Laboratories of the Nippon Telegraph and Telephone (NTT) Corporation in Atsugi, Japan, in 1996.

11. Two teams of two players in separate rooms compete via a network so that the players within a team are free to verbalise as they play.

Saito and Yoshikawa observed four types of thinking phases during move selection. Firstly, quick and inevitable response (e.g., response to a forcing move or continuation of a joseki[12]) which may not contain language-level inferences (described below). Secondly, quick evaluation and selection from a small number of candidates making extensive use of language and Go terms. Thirdly, elaborate lookahead and decision during which players generally remain silent, later providing retrospective protocols after playing a move. Fourthly, thinking from the opponent's viewpoint.

In later work (Saito & Yoshikawa, 1996), the frequency of naming the purpose of a move (both the participants' and the opponents') was observed to be relatively high. In about 80% of cases, strong players correctly identified the purpose of their opponents' move and when such an understanding was not present, the result was a loss of local advantage possibly leading to the loss of the game.

Other findings reported by Saito and Yoshikawa include:

- 80% of evaluations were quick pattern-based evaluation without lookahead and 20% were slow evaluations using lookahead. They also observed that evaluation could be categorised into two cases: pattern-based, and language-based;
- Static patterns (i.e., static configurations of stones) were not the only patterns observed in the protocols. Patterns comprised of sequences of moves (e.g., joseki) were also observed. Saito and Yoshikawa described such patterns as sequential patterns (Saito & Yoshikawa, 1997) and as being quite different from static patterns;
- Go players play quickly (average 63 seconds per move; 4 seconds per move was the most frequently occurring latency), examine few candidate moves (usually 1 or 2 and rarely more than 5), make relatively little use of lookahead (average depth of about 4; most frequent depth is 2), and their lookahead is relatively free of branching; and
- From the protocols that they collected, Saito and Yoshikawa found that an issue affecting how well moves could or could not be remembered was whether or not a plausible meaning could be associated with a move. They describe such semantic information as being quite different from patterns.

12. Sequence of moves used in the opening of the game.

The most important function of language for Go players according to Saito and Yoshikawa is search-space reduction. From their observations, it appears that Go players frequently use what they call “language label guided search”. In such a search, sub-spaces of the search-space are labelled and candidate moves are restricted by a description of their purpose, effectively pruning large parts of the search-space. Saito and Yoshikawa conclude that pattern knowledge, language-level knowledge, and language-level inferences play equally important roles in Go. Saito and Yoshikawa use the term “language-level inference” to mean inferences made at a conscious level by employing language (e.g., a Go proverb). It is possible to draw the conclusion from their work that a language-level inference could be explained to another person whereas an inference made a pattern-recognition level may be made at an unconscious level and may possibly be difficult to explain to another person.

6.2.3.2 Candidate Move Generation (Eye Tracking)

A variety of eye tracking experiments were performed to investigate the issue of candidate move generation (Yoshikawa & Saito, 1997a, 1997b), and in particular, the ability to generate candidate moves extremely quickly. These investigations led them to conclude that expert Go players use hybrid pattern knowledge (described below).

Early investigations using eye tracking showed that the players eyes fixate between stones not on them. Players look at only a small portion of the board prior to making a move but examine a wider area of the board after making a move.

Tsume-Go problems are local life and death problems (similar to chess mate-in-x problems) and are traditionally viewed as exercises in lookahead. Yoshikawa and Saito had their two participants (2 kyu and 6 dan) solve Tsume-Go problems under time pressure so that there was not sufficient time to employ lookahead. Thus, it was possible to examine the generation of candidate moves. The participants wore an eye camera that recorded their eye movements. The problems were displayed on a monitor for 4 seconds, during which the participants had to indicate their answer using a mouse. The stimuli included 300 problems ranging in difficulty between 5 kyu and 5 dan. Ten of the problems were presented twice.

General results indicated that when the stronger player was correct, the fixation time was very rapid (between 200 and 660 milliseconds). On repeated problems, the participants did not always fixate on the same places, however, the strong player responded with the same answer.

The knowledge used by the participants (particularly the strong participant) was obtained by interview. From that knowledge it was concluded that strong players possess a special type of pattern knowledge that is employed, at least, in solving Tsume-Go problems. Hybrid pattern knowledge is a mixture of concrete and abstract knowledge. In the context of solving Tsume-Go problems, the information contained in a hybrid pattern includes a description of the situation, a static configuration of stones, suggestions for solving the problem, candidate moves and their priorities, importance of stones, and key stone(s) to solve the problem. Thus, hybrid patterns are comprised of both perceptual information (e.g., a static configuration of stones) and language-level descriptions (e.g., a description of the situation).

Yoshikawa and Saito investigated whether the use of hybrid pattern knowledge could account for the strong player’s performance on the Tsume-Go problems. The 300 Tsume-Go problems were separated into 2 sets according to those that could be answered using the hybrid pattern knowledge obtained by interview from the strong player and those that could not. They found that the stronger player’s performance on problems that could be answered using the hybrid pattern knowledge was around 60% whereas performance on the other set of problems was around 20%.

To try and determine when a player begins to possess and use hybrid pattern knowledge, another two participants (1 dan and 3 dan) were added to the study. Yoshikawa and Saito, acknowledging the small number of participants in their study, claim that their results show that Go players begin to use hybrid pattern knowledge after they progress beyond about 1 dan. Furthermore, the weaker the player, the more they rely on search to identify candidate moves. The patterns used by weaker players tend to be just the configuration of the stones, that is, simple patterns. As players becomes stronger, they rely more on their knowledge to identify candidates and use search to validate their choices.

6.2.3.3 Comparison of Go and Chess

Saito and Yoshikawa identified some similarities and differences between Go and chess that have a bearing on the Cognitive Psychology of Go. The main similarities noted between Go and chess was that the progressive deepening search reported in the chess literature is also apparent in Go, and that pattern knowledge plays a part in Go expertise.

The main differences noted between Go and chess by Saito and Yoshikawa were primarily related to the perceptual nature of Go. In chess, the pieces are clearly distinguishable from each other and have clearly defined roles (e.g., how they move). In Go, players have to perceptually segment stones (into groups etc.) to create meaningful chunks or structures for themselves. The process of segmenting stones into chunks is all the more difficult since, as Reitman (1976) showed, patterns in Go are organised as overlapping clusters. Thus, players may need to entertain multiple perceptual organisations and settle on a final one dynamically. This difficulty is also exacerbated by the size of the board (19x19) which means that it may be too difficult for players to maintain an internal representation of the board, and that they may therefore have to rely on an external representation of the board (i.e., the stones themselves).

Another difference related to perception noted by Saito and Yoshikawa was spatial recognition ability. In chess, there is a fixed orientation to the board and the pieces that encourages a particular view of the board (i.e., along an axis from player to player). In Go, there is no fixed orientation of the board which can be equivalently viewed from any direction. Thus, a pattern of stones must be recognised regardless of the players' viewpoint or the colour of the stones. Advice often given to amateur Go players is to avoid playing on the far side of the board (the opponents' near side) to help minimise the mental load associated with recognition.

In summary, the use of language and language-level inference plays an important role in Go. Skilled Go players appear to use hybrid patterns that contain language-level information as well as pattern (i.e., stone configuration) information.

6.3 Discussion

In light of the preceding reviews, the issues of inference, memory for moves, and memory tasks are discussed in turn. This section concludes by stating the initial research questions to be investigated in the experiments comprising Section III.

6.3.1 Inference

De Groot suggested that perception replaces abstraction or inference in chess problem solving as chess skill improves. If this is the case, it may mean that the process of abstraction or inference becomes so automatic that it is below the threshold of consciousness for skilled players. That is, the process of abstraction or inference seems to be perception because the players are unable to introspect about it. The “guessing with common sense” and intelligent/ educated guessing described in De Groot’s protocols could well be descriptions of a form of inference.

Chase and Simon explained chess expertise in terms of chess knowledge, or more particularly, chess pattern knowledge. They mostly ignored inference as a factor in chess memory, especially memory for positions. However, although they do not explicitly suggest that inference plays a role in memory for moves, the regeneration of the intermediate positions (i.e., the moves) between quiet positions can be construed to be a form of inference.

Reitman suggested that there was the possibility that the participants’ performance had a component of guessing. Furthermore, it was suggested that the stronger participant was more likely to be correct when guessing than the weaker player, and was also more likely to employ some form of problem solving using partially completed patterns as clues. In this case, the problem solving activity may be construed to be a form of inference.

Saito and Yoshikawa discussed the importance of language-level inference. Such use of language would also be indicative of the underlying performance of Go players, that is, language-level inference would indicate that inference is an integral component of Go skill.

In summary, the impact of inference on memory performance has received relatively little direct attention in the Go and chess literature. It would also seem that inference itself may be of many different forms, that is, inference may be a continuum from low level (e.g., akin to perception) to high level (e.g., logical deduction). The issue of the impact of inference on memory performance is worth examining because it has the potential to shed light on both the nature of memory performance, and the role played by inferential processes in memory performance.

6.3.2 Memory for Moves

Memory for moves has not received much attention in either the Go or chess literature. Chase and Simon as well as Saito and Yoshikawa suggest that move sequences are more than just simple patterns, that is, more than just the pieces or the stones on the board. Other information is stored along with the move sequences (e.g., plausible meanings in Go; transition information in chess). Such information would appear to open up the possibility of inferential processes playing a part in memory for moves.

The issue of memory for moves is interesting because it enables recall memory for specific sequences of moves to be investigated. In this thesis, this form of recall memory will be referred to as "episodic memory", meaning memory for a specific sequence of moves learned in a particular episode. Go is a good domain in which to test memory for moves since the final position results from the cumulative addition of stones. By careful selection of stimuli from the opening stage of professional games, it is possible to avoid captures and thus each element in the stimuli (i.e., each stone) is associated with exactly one move.

6.3.3 Memory Tasks

To investigate the issues of inference and memory for moves, it is necessary to either design new memory tasks or to modify existing memory tasks to facilitate such investigations. The memory tasks would need to enable memory for moves to be tested, as well as enable some differentiation to be made between memory performance and inference. Thus, the memory tasks would need to have two conditions: episodic and inferential. The criteria for the two conditions would be:

- *episodic*: to present stones from a Go game in a sequential manner and to test the participants' memory for those specific moves, that is, to test their memory for moves from a specific observed sequence; and
- *inferential*: to test the participants' ability to infer the order of play without first having seen the sequence of moves, and thus enabling some differentiation to be made between the participants' episodic memory and their ability to make inferences.

Two tasks described in section 6.1 appear to be suitable for modification to meet the above criteria, namely: the pennies guessing task described by Chase and Simon (section 6.1.2), and De Groot's construction task (section 6.1.1). The modifications are described in Chapters 7 and 8 respectively.

6.3.4 Research Questions

In light of the literature surveys contained in sections 6.1 and 6.2, the initial research questions investigated in the series of exploratory studies were:

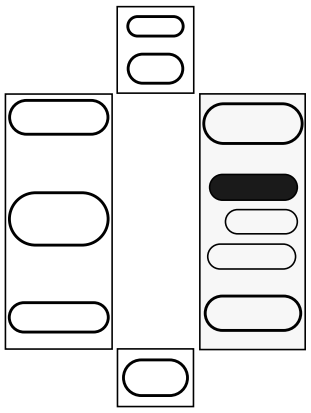
- what factors affect memory performance for moves in Go?, and
- to what extent does inference impact on memory performance in Go?

Several memory experiments were conducted on participants ranging from beginners to masters to investigate these research questions as described in Chapters 7 and 8.

Chapter 7

Study 1: Sequential Pennies-guessing Task

Introduction



The rationale for Study 1, as identified in section 6.3.3, is to begin answering the initial research questions of the project:

- what factors affect memory performance for moves in Go?, and
- to what extent does inference impact on memory performance in Go?

The study in this chapter reports an experiment (section 7.2) using beginner and experienced Go players as participants which investigates the factors affecting memory performance for moves in Go, and the extent to which inference impacts on memory performance in Go. The sequential pennies-guessing task (section 7.1), a modified version of the pennies-guessing task described by Chase and Simon (see section 6.1.2.2), was used as the memory task[1]. The results presented in section 7.3 show that performance on the sequential pennies-guessing task is related to Go skill. Discussion of the results and their bearing on memory for moves and the impact of inference on memory performance closes the chapter in section 7.4.

7.1 The Sequential Pennies-guessing Task

In chess there are six different types of pieces of two different colours. In Go, the playing pieces are homogenous with respect to shape, that is, there are simply stones of two different colours. Thus the pennies-guessing task would devolve into guessing the colour of any given “penny-marked” stone. However, although the pennies-guessing task would not be of much value as a memory task in Go, it is possible to modify the pennies-guessing task so that it involves

1. Random sequence testing based on Chase and Simon’s testing of random chess stimuli was trialled on a beginner and the results were found to be consistent with their results. Random sequences were not included in the studies in this thesis due to a lack of participant time.

reconstructing the order of presentation of the stones rather than reconstructing the final position. Such a modification would overcome the difficulty identified above, and would also meet the criteria outlined in section 6.3.3. That is, the modified sequential pennies-guessing task would have both an episodic and inferential condition so that the participants' memory for moves could be tested, and the inference and memory components of their performance could be differentiated to some extent.

The sequential pennies-guessing task consists of a presentation and a test phase and is conducted under two conditions: episodic and inferential. The participants' task is the same in both conditions: to indicate the order in which the stones in the stimuli were played. In the presentation phase of the episodic condition, the stones in the stimuli are presented sequentially every 2 seconds. In the presentation phase of the inferential condition, the stones in the stimuli are presented statically in their entirety, that is, no information was provided to the participants regarding the order of play.

During the test phase, the stones comprising the stimuli are visible to the participants (i.e., the positions of the stones act as cues for their sequential positions). The participants indicate the order of play by selecting the stones in sequence, receiving feedback on each successive selection. (The sequential pennies-guessing task is described more fully in section 7.2.1.3.)

In the episodic condition, episodic memory relates to memory for episodes (or events). Although the simplest form of episodes are the placement of each individual stone, much more complicated episodes can be hierarchically constructed, particularly by experienced participants. In the inferential condition, the participants' general Go knowledge together with information gained by feedback during the test phase is used to indicate the sequence of the moves. It is assumed that experienced Go players have access to more general Go knowledge and also profit more from the feedback information than beginner players.

7.2 Experiment 7.1: Sequential Pennies-guessing Task

The hypothesis of Experiment 7.1[1] was that the memory performance on the sequential pennies-guessing task would show an effect for Go skill and that all participants would perform better

1. Experiments will be labelled according to chapter and number (i.e., Experiments 7.1, 8.1, 8.2 and 9.1).

in the episodic condition than in the inferential condition (Burmeister & Wiles, 1996).

7.2.1 Method

7.2.1.1 Participants

The participants were unpaid volunteers who were divided into two groups: experienced and beginners. The four experienced players ranged between approximately 10 and 1 kyu; the four beginners ranged between approximately 25 and 15 kyu[1]. All but one of the participants had university level education. The ages of the participants' were in the range of 25 to 50.

7.2.1.2 Materials

Seven board positions[2] from the opening stage of Shusaku's[3] games were chosen in which there was a relatively even spread of stones around all the corners and edges and in which no stones had been captured. The positions contained between 17 and 35 stones (average 24.6). One board position was used as a practice board; the other six were randomly allocated into two groups of three (groups A and B).

Computer software[4] specially written for the experiment was used to present the board positions to the participants. The participants were tested individually. Throughout the experiment, the participants were seated in front of a monochrome X-window display with the experimenter present. A mouse-driven cursor was used to select stones during the test phase.

7.2.1.3 Procedure

The sequential pennies-guessing task was conducted under two conditions, episodic and inferential, and consisted of a presentation phase and a test phase. All participants participated in both conditions. Verbal reports were taken from the participants after the experiment was completed.

Episodic Condition

Presentation phase: The final position was cumulatively built by adding successive stones to the Go board on the computer monitor every 2 seconds in the order in which they had been played in the actual game (see Figure 7.1). Then the board grid and stones were cleared for 10

1. The participants' rating were self assessed on the basis of performance against stronger players.
2. The terms "board" and "position" may be used interchangeably when referring to experimental stimuli.
3. Shusaku was a famous Japanese player of the mid 19th century (Power, 1982).
4. The software was written by the author with the help of a colleague and is described in Appendix C.

seconds.

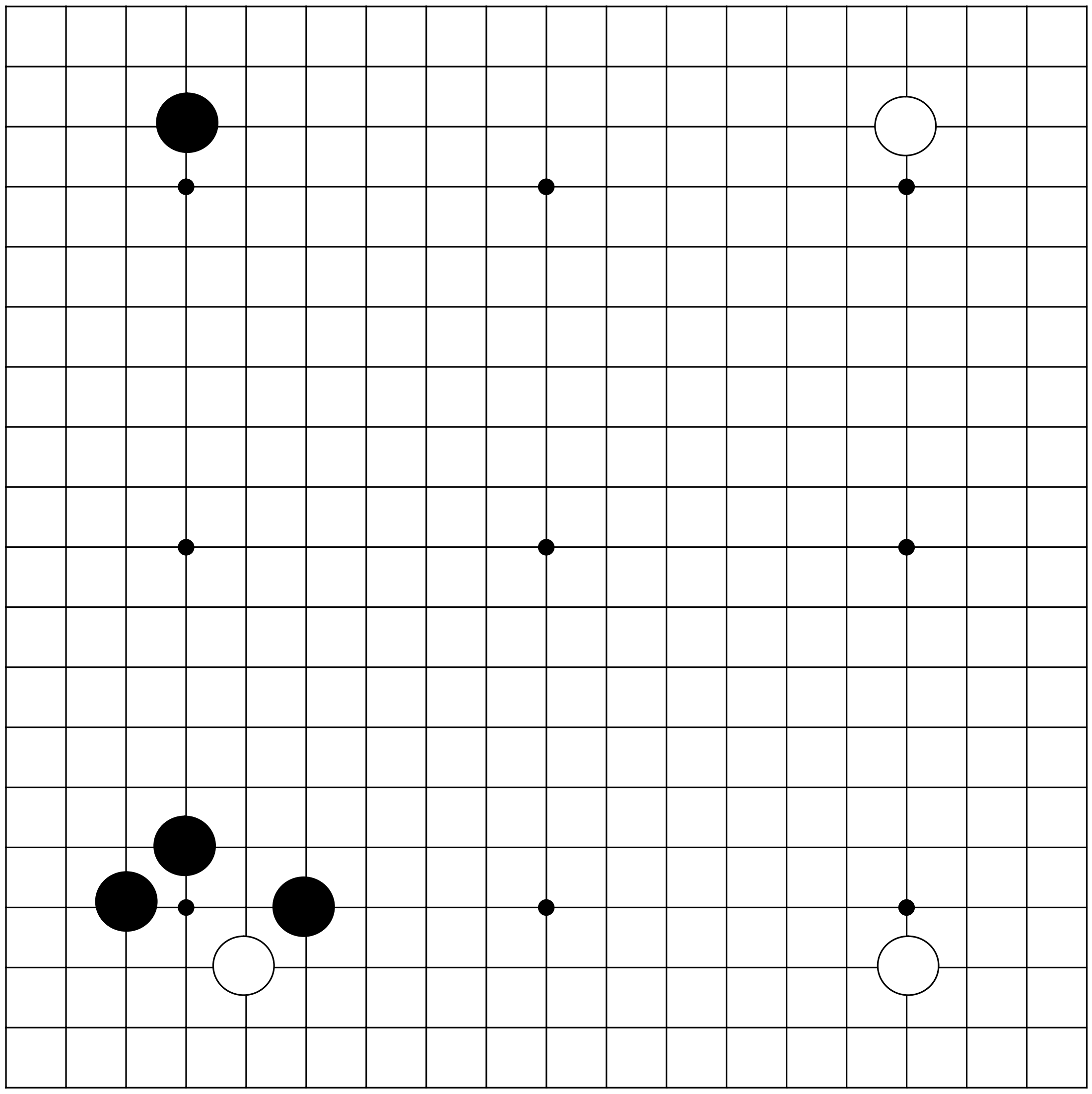


Figure 7.1: Sequential presentation in the episodic condition. Stones were added every 2 seconds: the figure shows the board position 12 seconds after the first stone was presented.

Test phase: The board was redisplayed with all the stones visible, that is, with the entire final board position being visible (see Figure 7.2). Thus, during the test phase, the participants were provided with the stones from the position as cues. The participants were asked to indicate the order in which the stones had been played. When an incorrect stone was selected (i.e., the stone selected was out of sequence or the wrong colour), an ‘x’ was displayed in the middle of the stone (see Figure 7.3). When a correct stone was selected, the number of the move on which it was played was displayed in the middle of the stone and all stones with an ‘x’ in them were

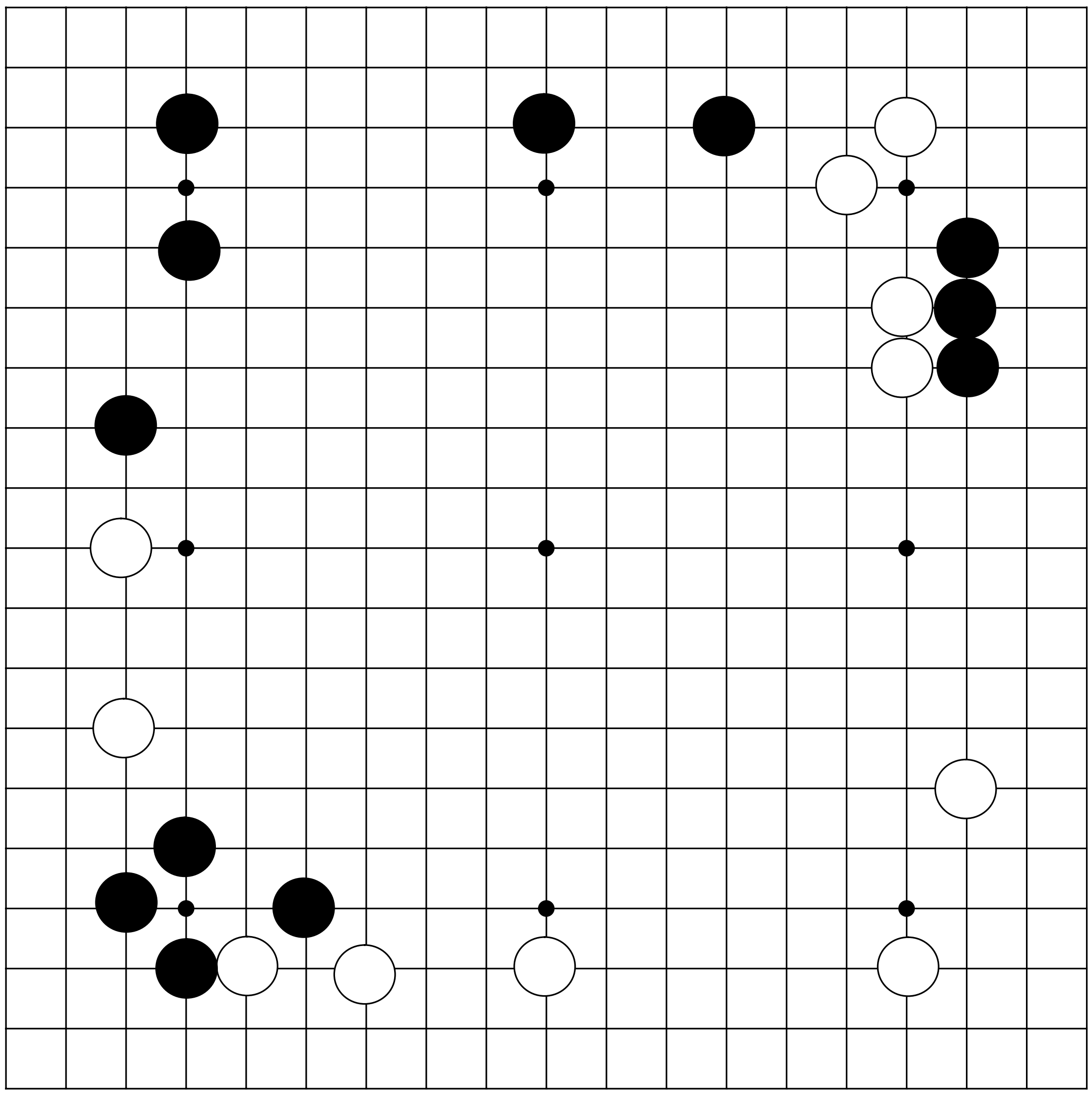


Figure 7.2: The entire position. The test phase of both the episodic and inferential conditions began with all the stones visible.

returned to solid colour (see Figure 7.4). If the participant could not correctly identify the next stone after 10 attempts, a dialog box would appear and upon pushing the 'OK' button, the computer would identify the correct stone and clear the 'x' from the wrongly selected stones. The participant would then continue the test process by trying to indicate which stone was the next one played.

Inferential Condition

Presentation phase: All the stones in the final position were displayed in the presentation phase, that is, all the stones were immediately visible.

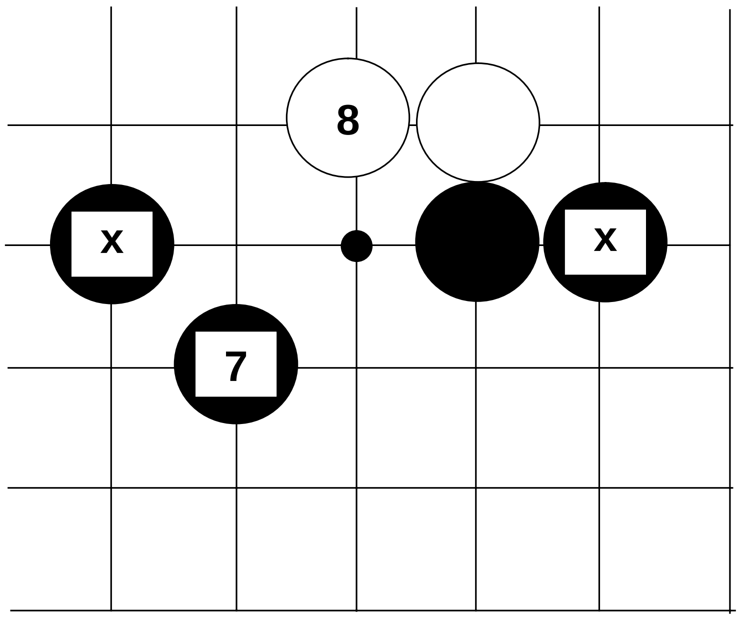


Figure 7.3: Feedback: wrong stone selected. A portion of the board showing the identification of the 9th move with 2 wrong stones selected (labelled ‘x’).

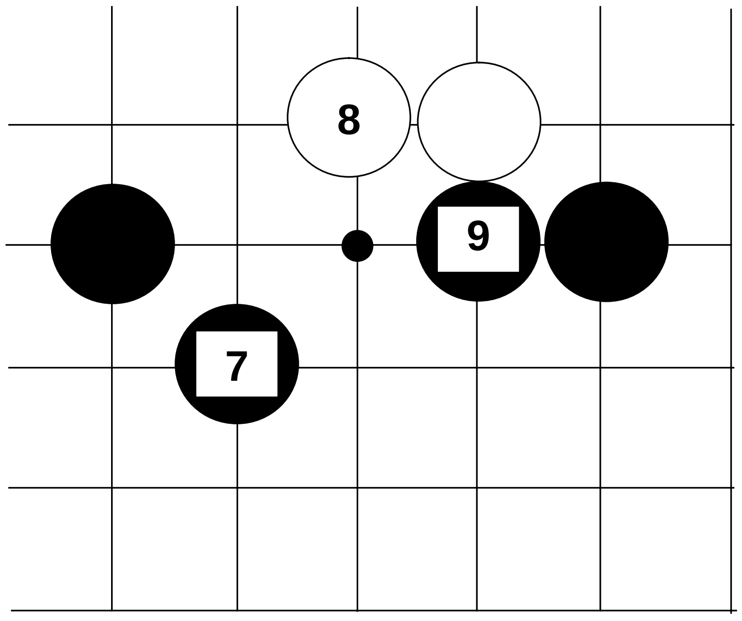


Figure 7.4: Feedback: correct stone selected. A portion of the board showing the ‘x’ labels cleared when the correct stone is selected. The correctly selected stone is labelled with its move number (‘9’).

Test phase: The test phase was the same as for the episodic condition and the mechanics of stone selection and feedback were also the same.

Practice

To enable the participants to become familiar with the use of the mouse and the feedback system employed by the software, they were presented with the practice board prior to being tested in the episodic condition. An explicit practice for the inferential condition was not necessary since the use of the software was the same as for the episodic condition, and the inferential condition always followed the episodic condition. The presentation of the positions from groups A and B were counterbalanced for participants within the experienced and the beginner groups, that is, half the participants in each group received the group A positions in the episodic condition and the group B positions in the inferential condition and the other half received the group A and B positions in the reverse order.

7.3 Results

During the episodic and inferential conditions, the number of attempts to correctly identify the next stone in the sequence was recorded for each stone on each board. The number of stones requiring a given number of attempts (i.e., one, two, three etc. attempts) to be correctly identified[1] was collated and then calculated as a percentage of the total number[2] of stones. The data was plotted as a cumulative curve so that it is possible to determine the percentage of stones requiring any given number of attempts to be correctly identified. Errorbars showing the maximum and minimum performance levels amongst the participants were also plotted. The data for the episodic and inferential conditions for the beginner and experienced participants appears in Figure 7.5 and shows that the performance in the episodic condition was higher than on the inferential condition for all participants and that the experienced participants performed better on the inferential condition than the beginner participants performed on the episodic condition. A clear ceiling effect can be seen and is accentuated by the maximum/ minimum errorbars.

1. Where participants incorrectly selected stones of the wrong colour, the attempt was not counted.
2. Since there was only one valid choice for the last two stones, they were not included in the data.

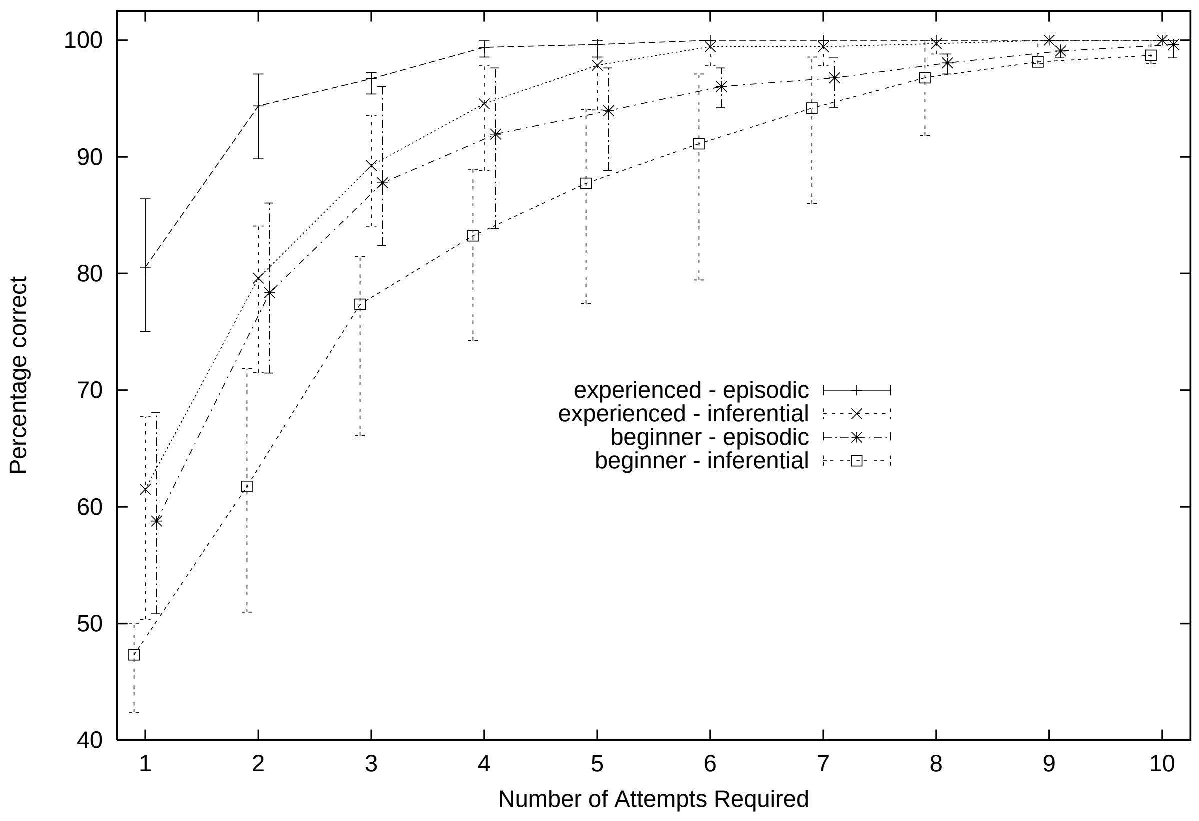


Figure 7.5: Sequential Pennies-guessing Task: experienced and beginner results. The cumulative curves represent the mean percentage of stones requiring a given number of attempts to be correctly identified. The errorbars show the maximum and minimum performance levels amongst the participants. The performance of all participants was better in the episodic condition than in the inferential condition. The experienced participants performed better in the inferential condition than the beginners performed in the episodic condition. (Note that some of the points are offset to make it easier to see the maximum/minimum errorbars and that the y-axis starts at 40%.)

7.4 Discussion

Although it is not possible to directly compare the results from the sequential pennies-guessing task with the results of Chase and Simon's pennies-guessing task (Chase & Simon, 1973b), the results are qualitatively similar with the experienced participants performing at 89.3% within 3 attempts and perfectly within 9 attempts compared to over 90% correct for Chase and Simon's Class A player. The initially low performance of 61.5% on the first attempt may have been due to the participants not detecting tenuki moves[1] on the first attempt but quickly recovering on their second or third attempts. The experienced participants performed almost perfectly after the 3rd attempt which indicates that the task was easy for them.

All participants performed better in the episodic condition, and thus, it may be concluded that the sequential pennies-guessing task is harder in the inferential condition than in the episodic condition. There was an effect for Go skill across the both types of participants with the experienced participants performing at around the same level in the inferential condition as the other participants performed in the episodic condition.

The fixed ordering of the conditions (i.e, the episodic condition always preceded the inferential condition) may be viewed as a possible confound. Any possible benefit of practice on the episodic condition would have raised the level of performance on the inferential condition. However, since performance on the episodic condition was always better than performance on the inferential condition, the order of the conditions does not affect the claim that the inferential condition was harder than the episodic condition.

7.4.1 Memory for Moves

The experienced participants reported a more extensive use of Go knowledge to aid their memory for moves. Surprisingly, Go terms seemed not to have been used, rather they either used their own terms (or spatial cues) or more often, the purpose for the moves. Salient moves were more readily remembered, perhaps suggesting a predictive component to their observations in the presentation phase. One of the beginners and one of the experienced participants indicated that 2 seconds was too short. The experienced participant commented that perhaps 3 or 4 seconds would enable a fuller analysis of each move. Several participants commented that it was sometimes difficult to know which stones had been “played” and which were present as cues.

7.4.2 The Impact of Inference on Performance

The sequential pennies-guessing task facilitates the investigation of inference. If a participant cannot remember the next stone in the order of play on the first attempt, then the only recourse (other than pure guessing) is inference or problem solving. In Experiment 7.1, wrong attempts

1. Moves to another part of the board rather than a local response to the opponent’s last move.

seemed to act as memory prompts for at least one participant, whilst others reported the use of problem solving activity.

It seems reasonable to assume that participants' performance in the episodic condition would have resulted from a combination of memory and inference or problem solving. It is possible to say that the provision of episodic information in the presentation phase results in an increase in performance, although in the current experiment, it is not possible to quantify the proportion of the participants' performance which can be attributed to episodic memory.

In the inferential condition, the participants could not exploit any episodic memory for the move sequence. Once again, in the current experiment, it is not possible to separate the episodic performance into episodic and inferential components. However, since the participants' performance in the inferential condition is around 75% of their performance in the episodic condition for the first attempt, it would appear that at least some component of memory performance on the sequential pennies-guessing task may be attributable to inference or problem solving.

7.4.3 Implicit Learning

An interesting aspect of the verbal reports was that many of the beginner participants reported learning during the experiment and some of them also commented that they felt that the task was a good way to learn Go. In pilot studies on novice Go players, similar learning was observed and reported. This unexpected implicit learning prompted Study 3 and is considered further in Chapter 9.

7.4.4 Need for a Further Study: Study 2

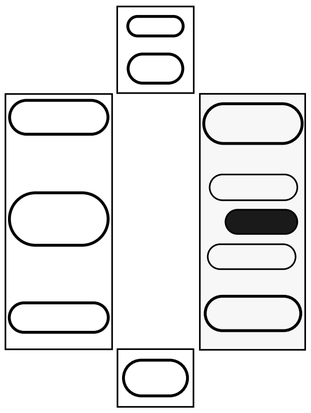
Since a ceiling effect was observed in the results of the sequential pennies-guessing task, a harder task was needed to continue investigation of memory for moves and the impact of inference on memory performance. The sequential construction task (reconstructing the order of play without the stones as prompts) is a harder task than the sequential pennies-guessing task, and the results of various experiments using that task are reported in Study 2 (Chapter 8).

Chapter 8

Study 2: Sequential Construction Task

Introduction

Study 2 built on Study 1 to continue the investigation of memory for moves and the impact of inference on memory performance. The sequential construction task (section 8.1), a modified version of De Groot's construction task (as described in section 6.1.1.2), was used as the memory task and proved to be a more difficult task than the sequential pennies-guessing task used in Study 1.



Study 2 comprises two experiments on beginner and experienced participants, and three case studies on three master Go players. Other than a minor change in stimuli, Experiments 8.1 and 8.2 were essentially the same, differing most notably with regard to participants. In Experiment 8.1 (section 8.2), the participants consisted of Australian beginner and experienced Go players. In Experiment 8.2 (section 8.3), the participants consisted of Japanese beginner and experienced Go players. The results (sections 8.2.2 and 8.3.2) show that performance on the sequential construction task was related to Go skill and that it is a good task for studying the impact of inference on memory performance. Case Studies 1, 2 and 3 (section 8.5) were conducted on three Japanese master Go players (6 to 8 dan amateur). The tasks were similar to those in Experiments 8.1 and 8.2 except that the stones were presented much more quickly in the episodic condition. The master participants exhibited extremely high levels of memory performance even at very fast presentation rates (e.g., 500 milliseconds between stones). Studies 1 and 2 are first steps towards using Go for the investigation of memory for moves and the impact of inference on memory performance, issues which are discussed in section 8.6.

8.1 The Sequential Construction Task

Unlike chess where the opposing players defend “their side” of the board, Go is omnidirectional, that is, there is no fixed sense of ownership of the board and the game could equivalently be presented to non-participants in any of its four orientations. Indeed, the object of the game is to progressively gain control of territory on an initially empty and uncontrolled board. Thus, there is no *a priori* disposition of pieces as there is in chess (e.g., pawn chains). An exact reproduction of De Groot’s construction task in Go would therefore result in an unacceptably high degree of uncertainty and therefore, an unacceptably high error rate. This difficulty would be compounded because the Go board is a much larger playing space (361 versus 64 points) and it would therefore be harder to correctly place a playing piece even if it is in the “right neighbourhood”. Although an unmodified version of De Groot’s construction task would be a difficult task in Go, it is possible to modify the construction task to overcome this difficulty.

The sequential construction task is similar to the pennies-guessing task, consisting of a presentation and a test phase, and is conducted under two conditions: episodic and inferential. The participants’ task was the same in both conditions: to indicate the order of play of the stones in the stimuli. In the presentation phase of the episodic condition, the stones in the stimuli are presented sequentially every 2 seconds. In the presentation phase of the inferential condition, only the first stone of the stimuli is initially presented on the board, that is, no information is provided to the participants regarding the order of play.

During the test phase, the participants are initially presented with a blank Go board (other than the single stone visible in the inferential condition). The participants indicate the order of play by selecting the board point where they think the next stone in the sequence was played. As in the pennies-guessing task, the participants receive feedback as to the correctness of their selection. (The construction task is described more fully in section 8.2.1.3).

One of the main differences between the construction task and the pennies-guessing task used in Experiment 7.1 is that during the test phase, the participants are provided with no cues. In the episodic condition, the participants (re)construct a sequence they have seen, whereas in the inferential condition, the participants are required to infer or problem solve where subsequent stones in the game had been played.

8.2 Experiment 8.1: First Sequential Construction Task Experiment

The sequential construction task was anticipated to be much harder than the pennies-guessing task used in Experiment 7.1 and thus it was expected that the participants' performance levels would be lower than in Experiment 7.1. The main hypothesis was that the memory performance on the sequential construction task would show an effect for Go skill and that all participants would perform better in the episodic condition than in the inferential condition (Burmeister & Wiles, 1996).

8.2.1 Method

8.2.1.1 Participants

All six of the participants in Experiment 8.1 had also participated in Experiment 7.1. The participants were unpaid volunteers who were divided into two groups: experienced and beginners. The three experienced players ranged between approximately 10 and 1 kyu; the three beginners ranged between approximately 25 and 15 kyu. All of the participants had university level education. The ages of the participants' were in the range of 25 to 50.

8.2.1.2 Materials

As for Experiment 7.1, board positions were selected from Shusaku's games (Power, 1982) so that there was a relatively even spread of stones around all the corners and edges and in which no stones had been captured. The eight board positions chosen were randomly allocated into two groups of four (groups A and B). Three positions were used in the episodic condition and contained 25 stones, and three positions were used in the inferential condition and contained 16 stones[1]. Two practice board positions were used; one for each condition.

The participants completed the sequential construction task individually and were observed by

1. A smaller number of stones was used in the inferential condition because it was much harder and there were concerns about the participants' ability to concentrate during the entire task if more stones were used.

the experimenter. The participants used the same software as was used in Experiment 7.1 (see Appendix C). Throughout the experiment, the participants were seated in front of a monochrome X-window display with the experimenter present. The participants selected a point on the board with the mouse and a stone of the correct colour was placed at that point. The mechanics of stone selection and feedback were similar to Experiment 7.1 (as described in section 7.2.1.3).

8.2.1.3 Procedure

The sequential construction task consisted of a presentation and a test phase and was conducted under two conditions: episodic and inferential. All participants participated in both conditions. Verbal reports were taken from the participants after the experiment was completed.

Episodic Condition

Presentation phase: The presentation phase of the episodic condition was the same as for the episodic condition of the sequential pennies-guessing task used in Experiment 7.1 (as described in section 7.2.1.3). The stones were cleared from the board grid for 10 seconds.

Test phase: At the beginning of the test phase, a blank Go board was displayed. During the test phase, the participants were asked to indicate where each stone had been played in the order in which they had been added to the board. Thus, during the test phase, the participants were provided with no cues about the positions of the stones on the board. In the episodic condition, the participants were asked to (re)construct the final position they had seen during the presentation phase by placing successive stones on the board.

When a wrong board point was selected, a stone of the appropriate colour with an ‘x’ was displayed (see Figure 8.2). The wrong stones were cleared either when a correct board point was selected or when the computer placed the correct stone after 10 unsuccessful attempts.

Inferential Condition

Presentation phase: The presentation of the inferential condition consisted of displaying the first stone of a position (i.e., displaying a single stone) on an otherwise blank Go board (see Figure 8.1).

Test phase: The participants were asked to infer or problem solve (or guess) where subsequent stones in the game had been played.

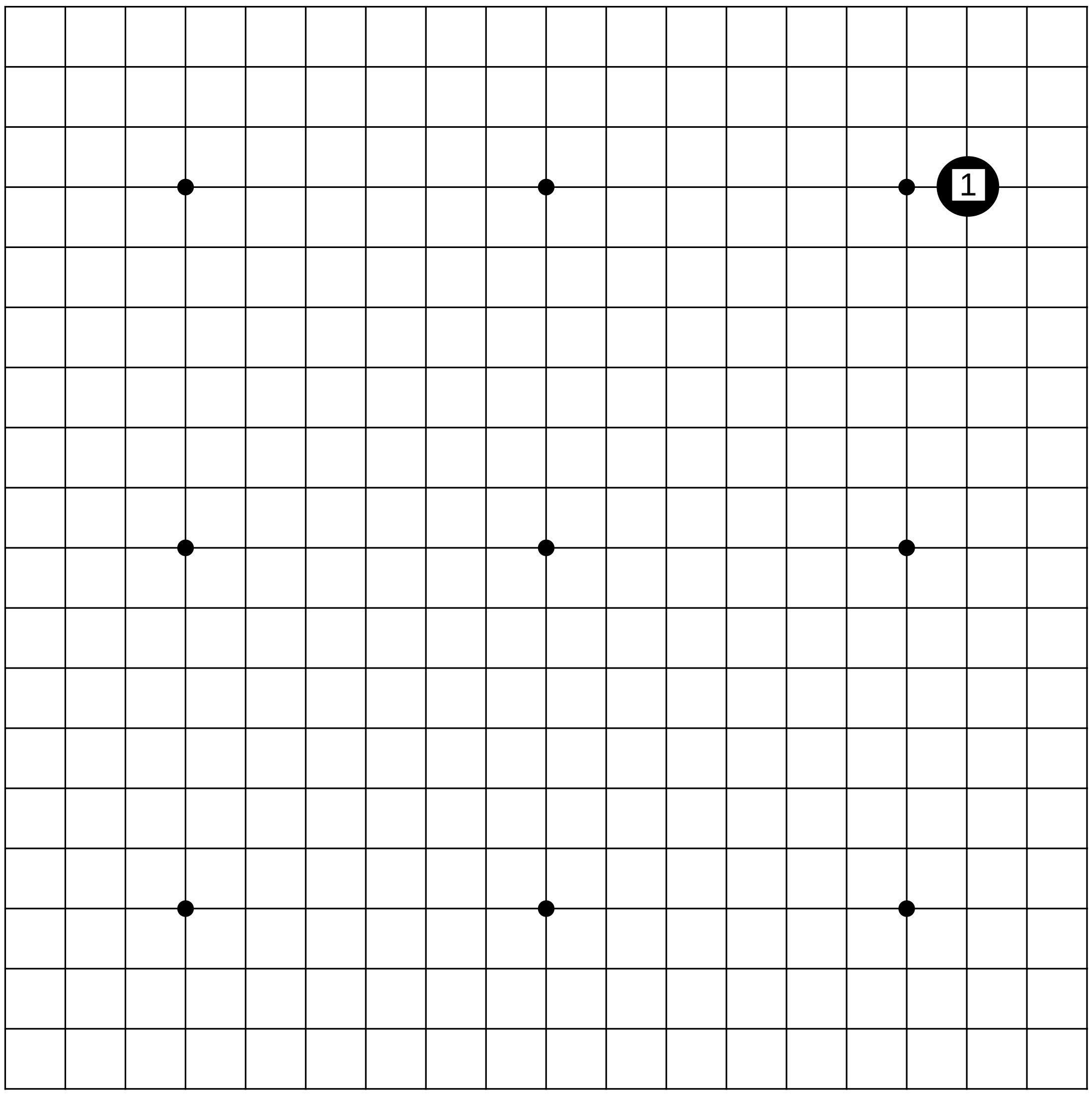


Figure 8.1: Initial stone presented in the inferential condition. Only the first stone was visible at the beginning of the test phase in the inferential condition.

Practice

To enable the participants to become familiar with the use of the mouse and the feedback system employed by the software, they were presented with the practice board prior to being tested in the episodic condition. An explicit practice for the inferential condition was not necessary since the use of the software was the same as for the episodic condition and the inferential condition always followed the episodic condition. Set A boards were used for the inferential condition; set B boards were used for episodic condition.

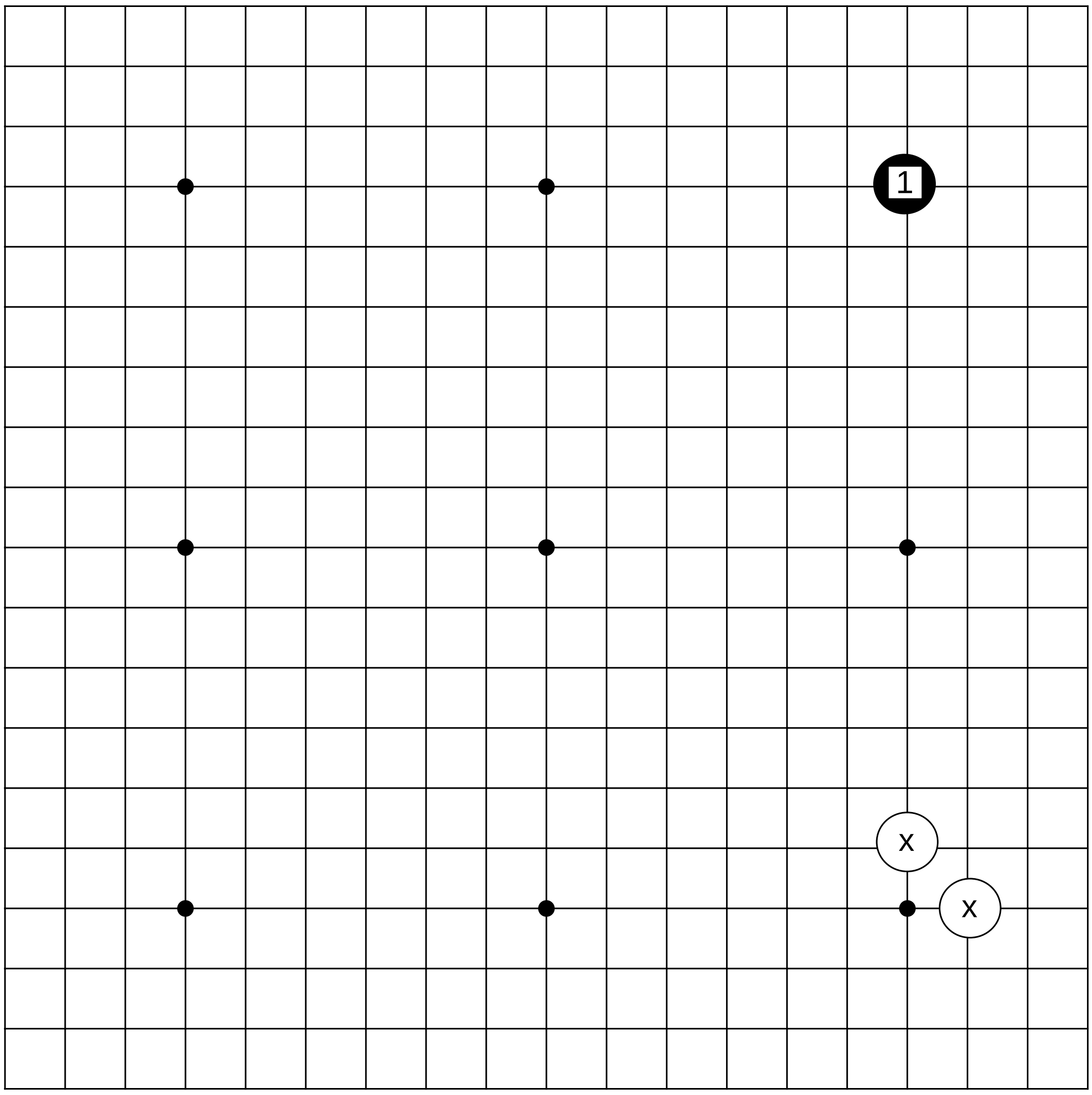


Figure 8.2: Feedback: wrong position selected. An incorrect position for move 2 has been selected. A white stone (labelled ‘x’) is placed on the incorrect position (in this example, two wrong attempts have been made).

8.2.2 Results

During the episodic and inferential conditions, the number of attempts to correctly identify the next stone in the sequence were recorded for each stone on each board. The number of stones requiring a given number of attempts (i.e., one, two, three etc. attempts) to be correctly identified were collated and then calculated as a percentage of the total number of stones. The percentages were calculated from 25 stones for the episodic condition and 15 stones (i.e., excluding the first) for the inferential condition. The data was plotted as a cumulative curve so that it is possible to determine the percentage of stones requiring any given number of attempts to be correctly identified. Errorbars showing the maximum and minimum performance levels amongst the participants were also plotted. The data for the episodic and inferential conditions for the beginner and experienced participants appears in Figure 8.3.

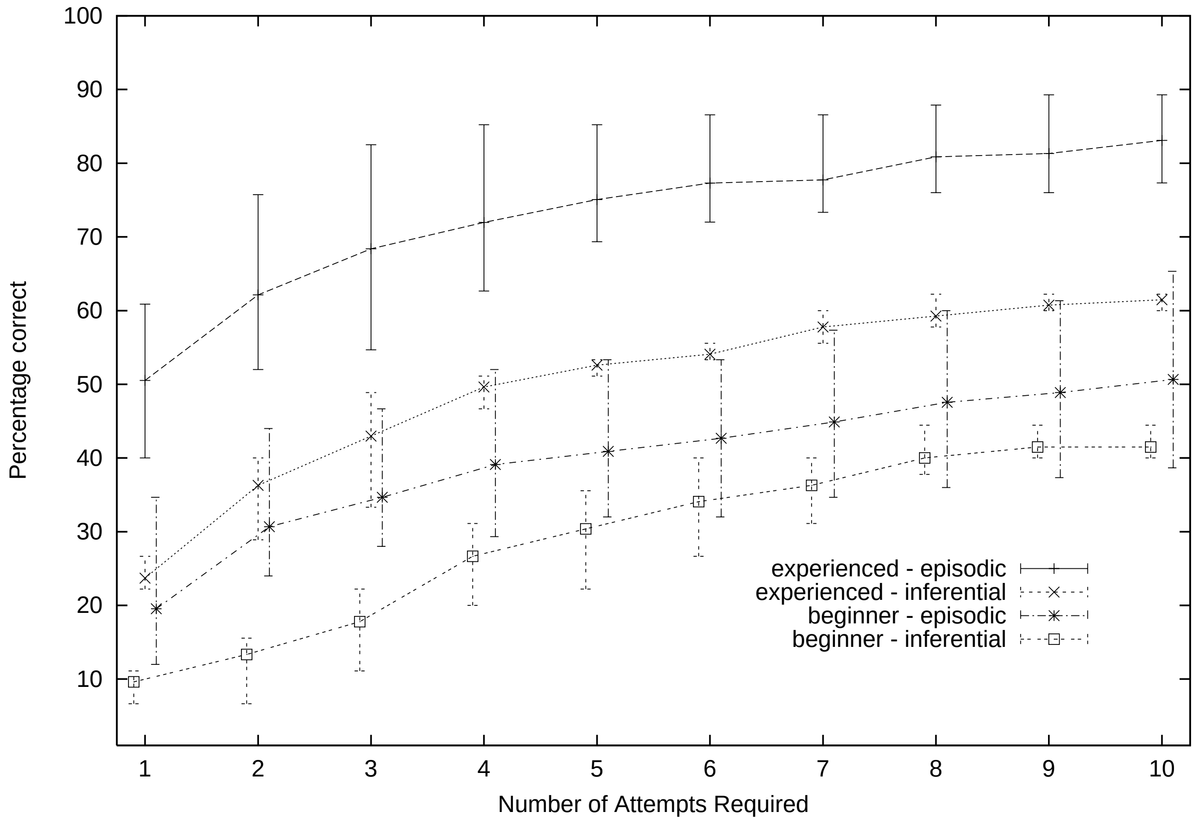


Figure 8.3: Sequential Construction Task: Experiment 8.1 results. The cumulative curves represent the mean percentage of stones requiring a given number of attempts to be correctly identified. The errorbars show the maximum and minimum performance levels amongst the participants. The performance of all participants was better in the episodic condition. (Note that some of the points are offset to make it easier to see the maximum/minimum errorbars.)

The performance in the episodic condition was better than on the inferential condition for all participants (see Figure 8.3). The mean performance of the participants' relative performance was at a level related to their Go skill. However, the errorbars (showing maximum and minimum performance) show that the performance of the experienced participants in the inferential condition was within the bounds of the maximum beginner performance in the episodic condition.

8.2.3 Discussion

All participants performed better in the episodic condition than the inferential condition of the construction task and there was a clear effect for Go skill across both types of participants for both conditions. The results indicate, as expected, that the sequential construction task is harder in the inferential condition than in the episodic condition. Furthermore, unlike the pennies-guessing task of Experiment 7.1, no ceiling effect was evident which supports the expectation that the sequential construction task is a more difficult task than the sequential pennies-guessing task, that is, that it requires more mental effort resulting in correspondingly lower performance levels.

An interesting aspect of the performance of all participants is demonstrated by the relative difference in the variability between the performance in the episodic condition and the inferential condition. The errorbars show that the variability in performance (difference between maximum and minimum performance) was larger in the episodic condition than in the episodic condition. One possible explanation for this is that performance in the episodic condition more accurately reflects a participants Go skill than performance in the inferential condition.

8.2.3.1 Comparison to the Results of De Groot’s Construction Task

Since the tasks are sufficiently different, it is not possible to directly compare the performance of the experienced and beginner participants in the inferential condition of Experiment 8.1 with the performance of the master and weak participants in the position construction condition of De Groot’s experiment (De Groot, 1966, p. 44). However, the performance of De Groot’s participants was superior to the performance of the participants in Experiment 8.1 on a purely numerical basis (starting at around 30% on the 1st trial rising to over 90% on the 10th trial). Acknowledging that a direct comparison is not possible, it is nonetheless instructive to consider there are three possible reasons for the disparity in performance levels.

The first reason is the relative difference in the size of the search space (i.e., 8x8 as compared to 19x19) which meant that the sequential construction task was much more difficult than for

De Groot’s construction task. It was for this reason that the first stone was presented to the participants in the inferential condition: the task would have been more difficult if the first stone also had to be identified.

The second reason is that in De Groot’s experiment, additional pieces could be added that would, on subsequent trials, provide inferential information to aid the replacement of a previously wrongly placed piece (e.g., the incorrect placement of pawn chains would provide information that would facilitate the correct placement of the pawns on subsequent attempts). By contrast, in Experiment 8.1, each stone had to be identified in order which meant that less information was available for inferential use by the participants. This difficulty was perhaps also exacerbated by the fact that there is only one type of piece in Go as opposed to six in chess. The restricted movements of the different types of chess pieces reduces the possible configurations resulting in a reduction of the search space. These difficulties (piece-wise serial reconstruction and homogenous pieces) were perhaps in some way alleviated, but not completely offset, by the participants having far longer to view the stimuli than the 5 seconds given to De Groot’s participants.

The third reason is that De Groot’s master chess players were comparatively better players than the experienced participants used in Experiment 8.1 (i.e., the experienced participants could not be considered to be master Go players). This disparity is addressed in the case studies of master Go players in section 8.5.

8.2.3.2 Verbal Reports

It was expected that the sequential construction task would be harder than the sequential pennies-guessing task of Experiment 7.1 because the search space was larger and the stones were not visible as prompts. Two of the participants (an experienced and a beginner) expressed this view. However, one of the beginner participants commented that they felt that the inferential condition was easier than the episodic condition even though their performance was not as good. Such a conclusion may have been due to feeling less pressure in the inferential condition because an incorrect selection did not indicate a “failure” of memory (i.e., because they were not required to remember the sequence of moves). Thus, the mental load was less in the inferential condition because the sequence of moves did not have to be remembered.

Comments were received regarding the choice of stimuli. One of the experienced participants suggested that more familiar games[2] would facilitate performance because they would contain “standard joseki”. This feedback was taken into account in selecting stimuli for Experiment 8.2.

When asked, several participants including some beginners and some experienced participants commented that they felt they had learned during the experiment, although one beginner said no learning took place. This unintentional learning was unanticipated and was pursued in Study 3 (Chapter 9).

The choice of 2 seconds between stones was based on the results of pilot studies. One of the experienced participants commented that a delay of more than 2 seconds between stones would be required to fully describe the reason for each move. This issue was further explored in case studies conducted on master Go players (see section 8.5) in which the master subjects felt the time pressure at 500 milliseconds).

Aspects of the visual display employed in the experiment caused concerns for at least one of the experienced subjects who expressed the view that it was sometimes difficult to know which stones had been actually “played” and which did not actually “exist” (i.e., which stones were present as feedback to signify for incorrect selections). This problem was rectified in Study 3 (Chapter 9).

8.3 Experiment 8.2: Second Sequential Construction Task Experiment

Experiment 8.2 was conducted in the Basic Research Laboratories of the Nippon Telegraph and Telephone (NTT) Corporation in Atsugi, Japan. Experiment 8.2 was a replication of Experiment 8.1 with two minor differences: slightly different stimuli were used and the participants were Japanese rather than Australian. It was predicted that the results would be similar to those of Experiment 8.1, providing support for the findings of Experiment 8.1.

2. Shusaku’s games are somewhat old fashioned and contain joseki that are unfamiliar and no longer commonly used.

8.3.1 Method

8.3.1.1 Participants

The participants were four male university students and were paid for their participation. They were divided into two groups: experienced and beginner. The two experienced participants were 1 dan and 3 dan; the two beginner participants were both 17 kyu. The participant's ages were in the range of 19 to 23. The participants were native Japanese speakers who spoke sufficient English for the experiments to be conducted in English.

8.3.1.2 Materials

The stimuli used in Experiments 7.1 and 8.1 (Shusaku's games) may appear unusual to experienced Japanese players because they are old fashioned. Therefore, the board positions selected were from Go Seigen's[3] games since they were considered to be more contemporary in style. Eight board positions were selected in which there was a relatively even spread of stones around all the corners and edges and in which no stones had been captured (as for Experiments 7.1 and 8.1). Each position contained 25 stones rather than 16 as for Experiment 8.1 in an attempt to standardise the mental load imposed during the two conditions. Two practice board positions were used (one for the episodic condition and one for the inferential condition); the other 6 were randomly allocated into two sets of three (A and B).

The computer software[4] used was essentially the same as the software used for Experiments 7.1 and 8.1 except for slight modifications that were necessary to adapt it to the NTT computer system. The participants completed the experiment one at a time. Throughout the experiment, the participants were seated in front of a colour X-window display with the experimenter present. A mouse was used to select stones during the test phase.

8.3.1.3 Procedure

The procedure was the same as for Experiment 8.1 (as described in section 8.2.1.3), with an episodic and an inferential condition.

8.3.2 Results

During the episodic and inferential conditions, the number of attempts to correctly identify

3. A contemporary professional player whose career spanned the early to late 20th century.

4. The software is described in Appendix C.

the next stone in the sequence was recorded for each stone on each board. The number of stones requiring a given number of attempts (i.e., one, two, three etc. attempts) to be correctly identified was collated and then calculated as a percentage of the total number of stones. As for Experiment 8.1, the data were plotted as a cumulative curve so that it is possible to determine the percentage of stones requiring any given number of attempts to be correctly identified. Errorbars showing the maximum and minimum performance levels amongst the participants were also plotted. The data for the episodic and inferential conditions for the beginner and experienced participants appears in Figure 8.4.

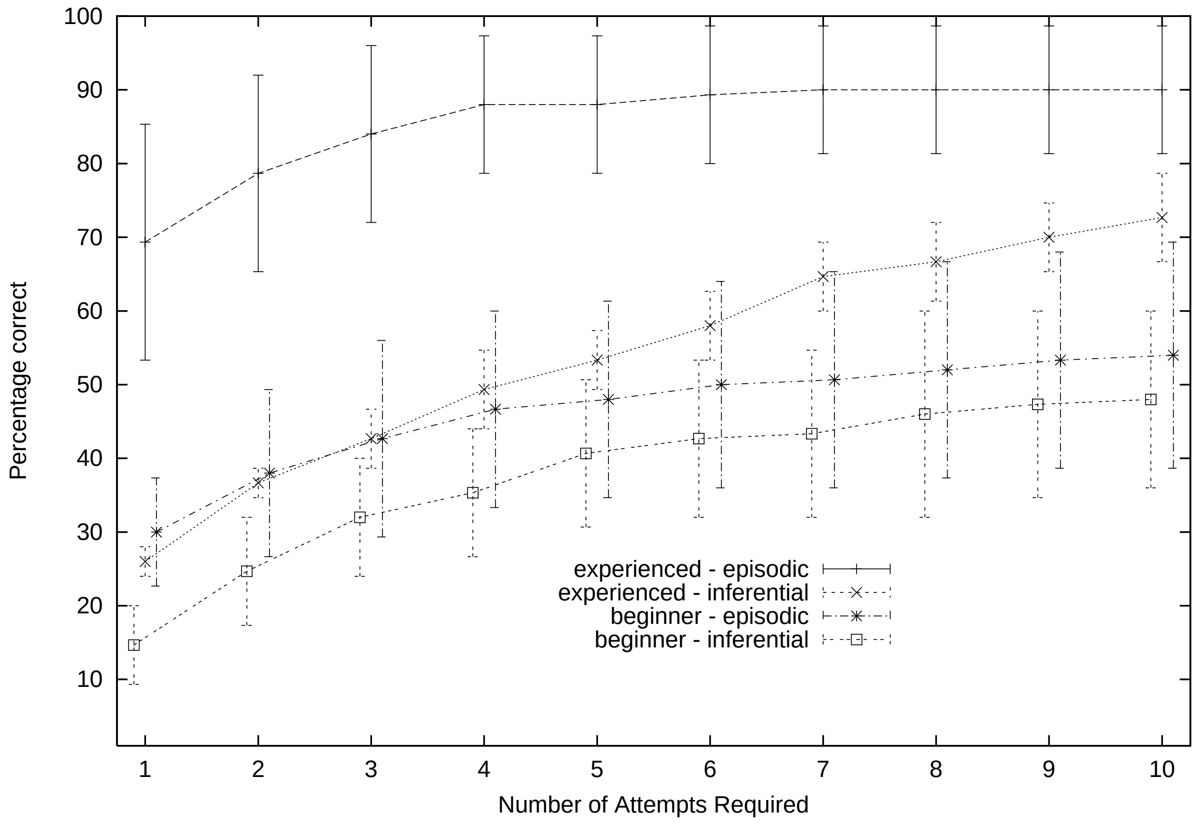


Figure 8.4: Sequential Construction Task: Experiment 8.2 results. The cumulative curves represent the mean percentage of stones requiring a given number of attempts to be correctly identified. The errorbars show the maximum and minimum performance levels amongst the participants. The performance of all participants was better in the episodic condition than in the inferential condition. (Note that some of the points are offset to make it easier to see the maximum/ minimum errorbars.)

The experienced participants' performed at a level related to their Go skill. The performance in

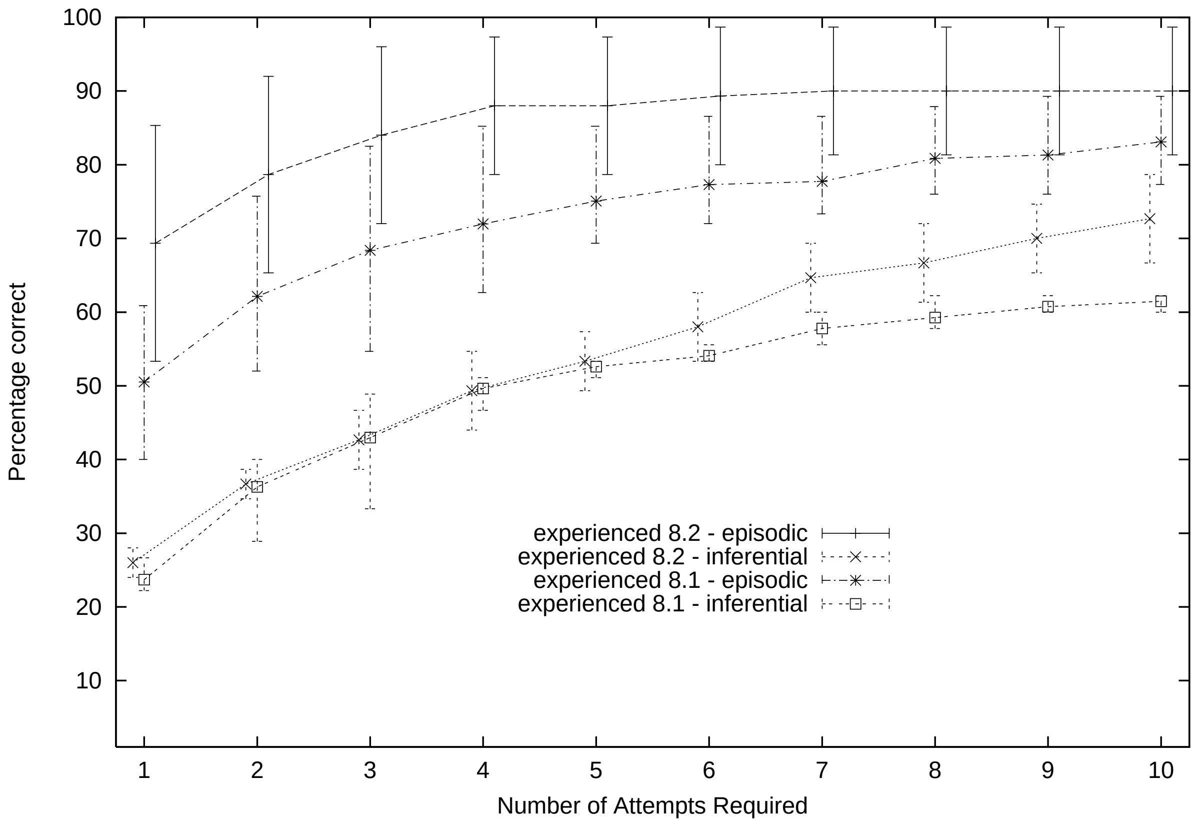


Figure 8.5: Comparison of Experiments 8.1 and 8.2 results: experienced participants. The cumulative curves represent the mean percentage of stones requiring a given number of attempts to be correctly identified. The errorbars show the maximum and minimum performance levels amongst the participants. The participants in Experiment 8.2 performed better than the participants in Experiment 8.1. It should be noted that there were differences in stimuli between the two experiments as described in section 8.3.3.1. (Note that some of the points are offset to make it easier to see the maximum/minimum errorbars.)

the episodic condition was better than in the inferential condition for all participants for stones requiring more than 3 attempts (see Figure 8.4).

8.3.3 Discussion

As for Experiment 8.1, the errorbars show that the variability in performance (difference between maximum and minimum performance) was larger in the episodic condition than in the episodic condition. Once again it is possible to conclude that performance in the episodic condition more accurately reflects a participants Go skill than performance in the inferential condition. Thus, the results of Experiment 8.2 supported the findings of Experiment 8.1 that

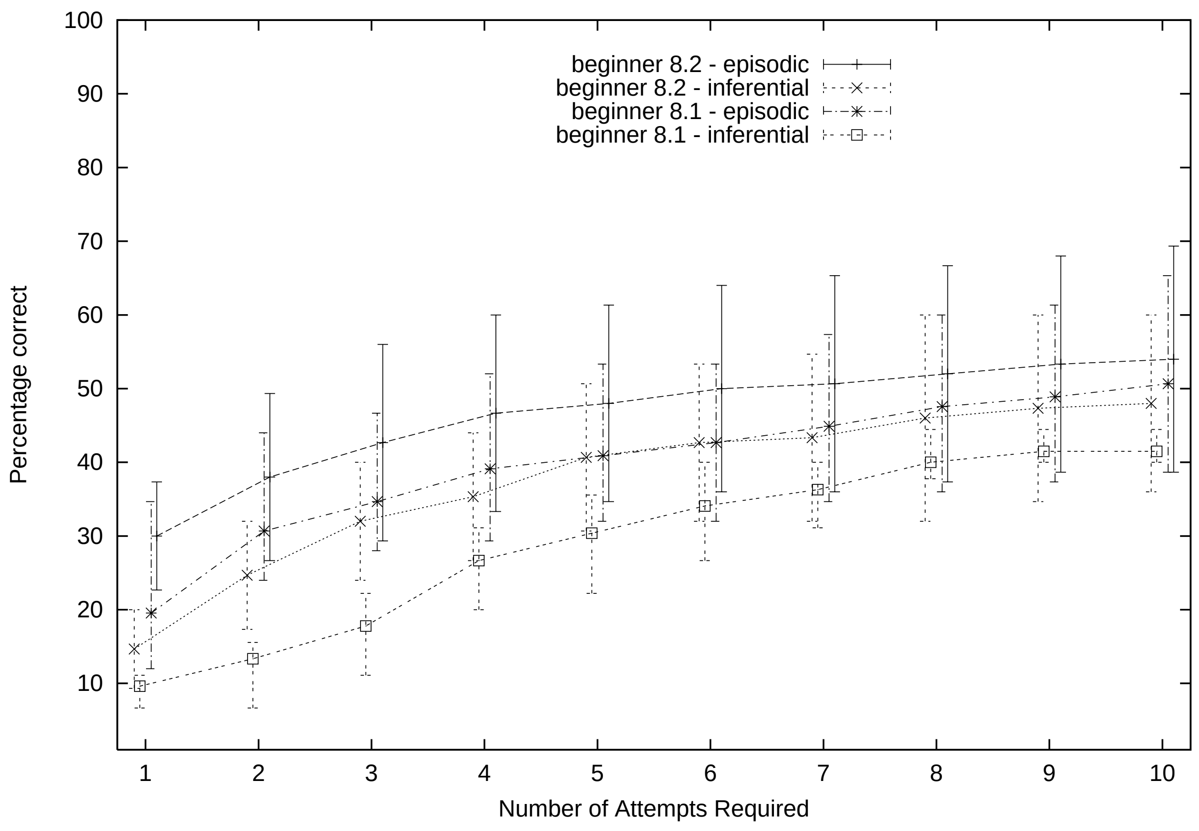


Figure 8.6: Comparison of Experiments 8.1 and 8.2 results: beginner participants. The cumulative curves represent the mean percentage of stones requiring a given number of attempts to be correctly identified. The errorbars show the maximum and minimum performance levels amongst the participants. The participants in Experiment 8.2 performed better than the participants in Experiment 8.1. (Note that some of the points are offset to make it easier to see the maximum/minimum errorbars.)

performance on the construction task is related to Go skill, particularly in the episodic condition. Furthermore, the replication appears to be quite robust since it was obtained not only using a different set of participants, but also using a different set of Go positions (i.e., Go Seigen games rather than Shusaku games).

8.3.3.1 Comparison With Experiment 8.1 Results

The number of participants in Experiments 8.1 and 8.2 was relatively small and the participants were therefore only approximately matched for skill levels. Before comparing the performance of the subjects in the two experiments, it should be noted that there were differences in the stimuli: Go Seigen's games were used rather than Shusaku's games; the stimuli boards used in the inferential condition contained 25 stones rather than 16; the stimuli boards used in the episodic condition contained a uniform 25 stones rather than a variable number (average 24.6 stones). Some of these changes made the task easier and some made the task harder: on the whole the Japanese participants were more skilful and so the tasks were therefore easier for them; Go Seigen's games were more contemporary, therefore making the task easier; the increase in stones from 16 to 25 would have made the task harder. For these reasons it is not possible to directly compare the performance levels of the participants in the two experiments.

In the light of the differences just described, it is illustrative to make a qualitative comparison of the performance levels of both the experienced and beginner participants in the two experiments. That comparison shows that the Japanese participants generally performed better than the Australian participants (see Figures 8.5 and 8.6 respectively). The difference in performance may in part have resulted from the use of contemporary Go Seigen games as stimuli. Comparing the performance of the experienced participants (Figure 8.5) shows that the Japanese participants performed better in the episodic condition across all attempts but only excelled relative to their Australian counterparts in the inferential condition after the 5th attempt. Indeed, for reasons yet to be explained, the performance of experienced Japanese participants increased significantly from the 5th attempt onwards. Comparing the performance of the beginner participants (see Figure 8.6) shows that the Japanese participants performed better in both conditions across all attempts.

8.3.3.2 Comparison With the Results of De Groot's Construction Task

As for Experiment 8.1, it is not possible to directly compare the performance of the participants in Experiment 8.2 with the performance of De Groot's participants. However, on a purely numerical basis, the performance of De Groot's participants is superior although the difference is somewhat smaller. The possible reasons for the disparity advanced for Experiment 8.1 in section 8.2.3.1 also apply for Experiment 8.2.

8.3.3.3 Verbal Reports

When asked, all the participants commented that they thought the task may be useful for learning Go. Both of the experienced participants mentioned recognising josekis and also having difficulties with tenuki moves. One of the beginner participants said that he tried to remember the meaning of the moves but they were too difficult to understand. He suggested games closer to his ability may be helpful in a learning situation.

8.4 Summary: Experiments 8.1 and 8.2

The results of Experiments 8.1 and 8.2 may be summarised as:

- performance on the sequential pennies-guessing task is related to Go skill for all participants;
- the inferential condition is more difficult than the episodic condition for all participants on both tasks (i.e., performance is lower).

The main points resulting from the verbal reports of the participants of Experiments 8.1 and 8.2 were that:

- the sequential construction task facilitated some participants to learn Go, which was not an anticipated outcome of the experiments;
- the appropriateness of the games used as stimuli was important (i.e., modern, familiar games were preferable); and
- the time between stones was linked to the ability to understand the meaning of a move.

It is not possible to make a qualitative comparison to the performance of De Groot's masters since none of the participants tested in Experiments 8.1 and 8.2 could be described as master Go players. Several case studies conducted on master Go players are reported in section 8.5.

8.5 Case Studies: Memory Performance of Master Go Players

This section describes both the performance and the relevant verbal reports resulting from three case studies which investigated the performance of master Japanese Go players on the sequential construction task (Burmeister, Saito, Yoshikawa & Wiles, 1997). Besides investigating memory for moves and the impact of inference on memory performance with master participants, the goal of these case studies were twofold: to provide a comparison to the performance levels of De Groot's master participants, and to investigate the masters' performance on the task as the inter-stone delay was reduced. The methodology for the 3 case studies was similar to that used in Experiments 8.1 and 8.2. Differences (which will be detailed in each case study) were in the delay between the stones (2000, 1500, 1000, and 500 milliseconds) and the boards used as stimuli.

8.5.1 Case Study 1: 8 Dan Player - *8d*

In the first case study, the participant was a 20 year old male university student who received the 1997 university Meijin prize. He was ranked 8 dan and will be referred to as *8d*. The participant was paid for his participation. Although an amateur, 8d played at around the level of a 1 dan professional.

8.5.1.1 Initial Test of Episodic Condition

An initial test was conducted on 8d to gauge the performance of a master participant in the episodic condition of the sequential construction task at several different delay levels. Four 25-stone boards from the previous study were used and 8d was tested on a separate board at 2000, 1500, 1000, and 500 millisecond delays between stones. 8d's performance was at ceiling with a 2000 millisecond delay and did not significantly drop until the delay was reduced to 500 milliseconds (see Figure 8.7). At 500 milliseconds, 8d's performance on the 1st attempt was comparable with the performance of the experienced participants at 2 seconds although no stone required more than 6 attempts to be correctly identified. In general, 8d's performance decreased as the delay decreased although performance at the 1500 millisecond level was lower than at the 1000 millisecond level. This decrease in performance was most likely due to the difference between the individual boards. Due to limitations on the graphic presentation rate of the software used[5], delays lower than 500 milliseconds were not used.

8.5.1.2 Episodic Condition at 1000 Millisecond

8d was tested at 1000 millisecond delay between stones on the set A positions used to test the experienced and beginner participants in Experiment 8.2; the task was practised on 1 position prior to testing. 8d performed nearly perfectly and a clear ceiling effect was observed (see Figure 8.8).

8.5.1.3 Episodic Condition at 500 Milliseconds

8d was tested at 500 millisecond delay between stones on a new set (C) of 3 positions selected from Go Seigen's games in a similar way to the games selected for sets A and B; the task was practised on 1 position prior to testing. Once again, 8d performed nearly perfectly and a clear ceiling effect is present (see Figure 8.9).

5. This aspect of the performance of the software is discussed in Appendix C.

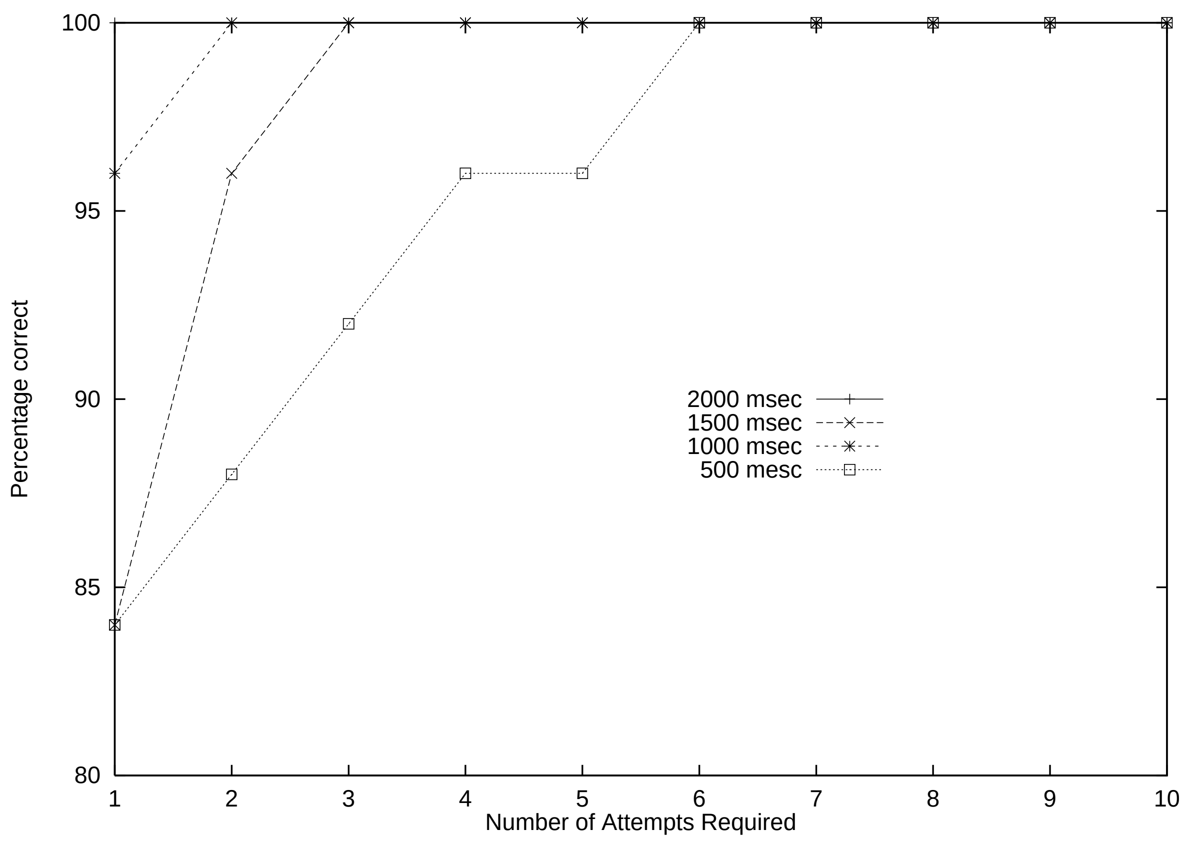


Figure 8.7: *8d* episodic trial results. The cumulative percentage of correctly identified stones for each attempt at 2000, 1500, 1000, and 500 milliseconds. Note that the y-axis starts at 80%.)

8.5.1.4 Inferential Condition

8d was tested in the inferential condition on the set B positions; the task was practised on 1 position prior to testing. 8d's performance was far from ceiling with around 41% correct on the first attempt and 84% correct within 10 attempts (see Figure 8.9).

8.5.1.5 Verbal Reports from 8d

8d reported that during the episodic condition, he created a dialog of the "story" as the moves were added to the board. Meaningful moves were remembered because they made sense in the context of the story. A few moves did not make sense with respect to the story and therefore stood out, making them easier to remember. Although 8d was not explicitly asked, it would seem that the dialog he created indicates that there was an element of prediction associated with the next move. The nature of the prediction is likely to be the type of move (e.g., approach) and a general area (e.g., a small neighbourhood of points at which the move would be expected).

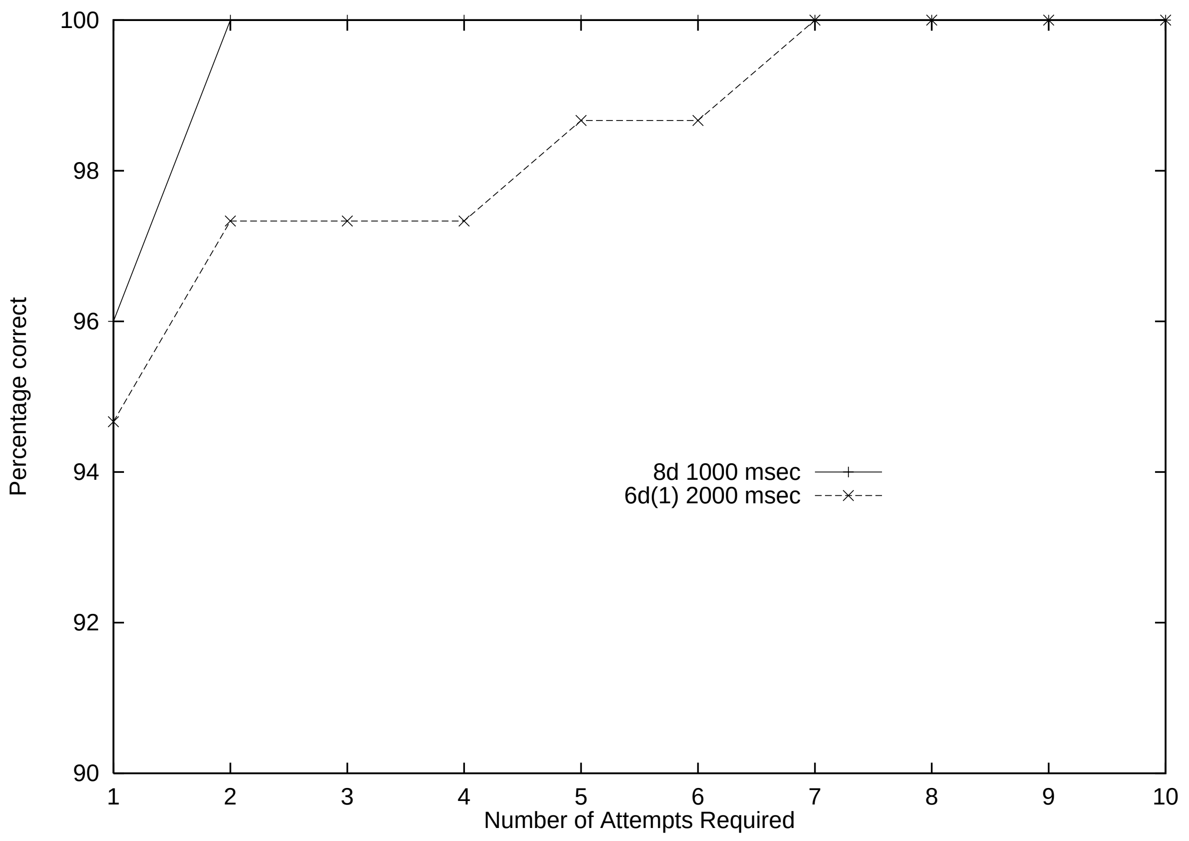


Figure 8.8: *8d* and *6d(1)* episodic results. The cumulative percentage of correctly identified stones for each attempt for *8d* and *6d(1)* tested at 1000 and 2000 milliseconds respectively. (Note that the y-axis starts at 90%).

8d commented that he would probably be able to replay the moves from the positions he was tested on up to 1 week later. Although his recall was not tested, his claim was consistent with his ability to remember entire Go games.

An interesting incident occurred during testing on a position in the episodic condition at the 1 second delay. 8d was surprised by move 18 during the sequential presentation (which is consistent with the prediction hypothesis described above). During reconstruction, he made a mistake in placing move 16 but placed moves 17, 18, and 19 correctly. It seems that the surprise he experienced when move 18 was presented interfered with his encoding of the previous white move (move 16) rather than interfering with the encoding of subsequent moves.

There was a significant delay in the updating of graphic information during the test phase of the

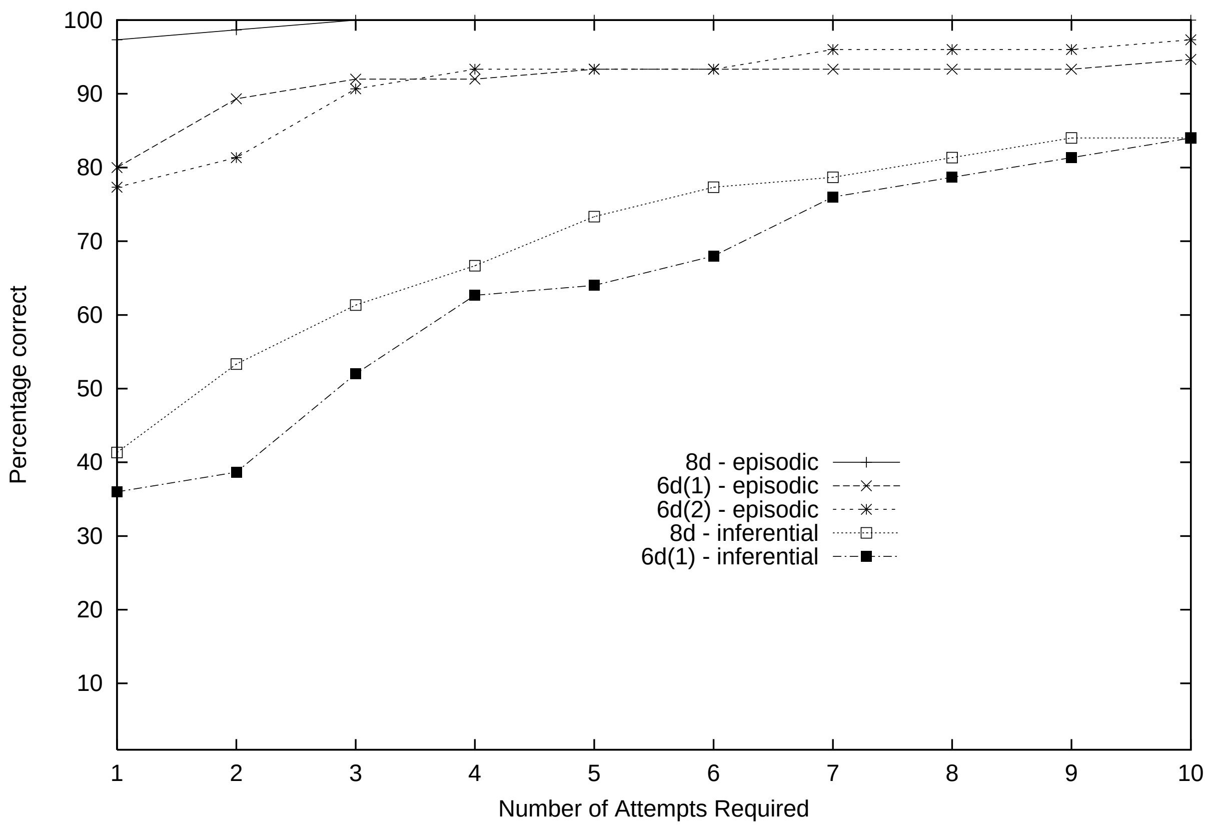


Figure 8.9: Master participants' episodic (500 milliseconds) and inferential results. The cumulative percentage of correctly identified stones for each attempt for master participants in the episodic (500 milliseconds) and inferential conditions.

inferential condition. At times, 8d selected up to 15 moves without any feedback from the software with almost perfect results, indicating not only accuracy, but also confidence in his selections.

During the initial testing in the episodic condition, the participant commented that he could "image" (or see) the whole board between stones delays of 2000, 1500, and 1000 milliseconds but could not do so at the 500 millisecond delay level.

8.5.2 Case Study 2: First 6 Dan Player - *6d(1)*

In the second case study, the participant was a 22 year old female postgraduate university student who was the 1995 University Female champion. The participant was ranked 6 dan and will be referred to as *6d(1)*. The participant was paid for her participation. The 500 millisecond episodic condition was identical to Case Study 1; the A and B sets were counterbalanced with case study 1 for the 2000 millisecond episodic condition and the inferential condition.

8.5.2.1 Episodic Condition at 2000 Milliseconds

6d(1) was tested at 2000 millisecond delay between stones on the set B; the task was practised on 1 position prior to testing. 6d(1)'s performance was slightly below 8d's (see Figure 8.8) but she performed almost perfectly (only 4 stones out of 75 were not correctly identified within 10 attempts).

8.5.2.2 Episodic Condition at 500 Milliseconds

6d(1) was tested at 500 millisecond delay between stones on the set C positions; the task was practised on 1 position prior to testing. 6d(1)'s performance was far from ceiling with 36% correct on the first attempt and 84% correct within 10 attempts (see Figure 8.9).

8.5.2.3 Inferential Condition

6d(1) was tested on the inferential condition on the set A positions; the task was practised on 1 position prior to testing (see Figure 8.9). 6d(1)'s performance on the inferential condition was lower than her performance on the episodic condition.

8.5.2.4 Verbal Reports from 6d(1)

When tested at the 2000 millisecond delay level on the episodic condition, 6d(1) commented that she looked at the whole board between moves. 6d(1) further indicated that joseki patterns were easy to remember and that tenuki moves did not present any problems either. 6d(1) selected many moves without feedback during the reconstruction phase of the inferential condition, which as noted for 8d, indicates not only her accuracy, but also her confidence in her selections.

8.5.3 Case Study 3: Second 6 Dan Player - *6d(2)*

In the third case study, the participant was a male paid university student. The participant was ranked 6 dan and will be referred to as *6d(2)*. The participant was paid for his participation. The 500 millisecond episodic condition was identical to Case Studies 1 and 2.

8.5.3.1 Episodic Condition at 500 Milliseconds

Due to time constraints, 6d(2) was only tested at 500 millisecond delay between stones on the set C positions; the task was practised on 1 position prior to testing. 6d(2)'s performance was comparable to 6d(1)'s (see Figure 8.9) with a clear ceiling effect being observed (only 2 stones out of 75 were not correctly identified within 10 attempts).

8.5.3.2 Verbal Reports from 6d(2)

6d(2) commented that in the episodic condition, he remembered the meaning of the moves (i.e., the intentions of the players). 6d(2) indicated that joseki and shape were used to facilitate memory and that without such a strategy the short time between the stones would make remembering the moves difficult. 6d(2) also reported that he remembered both the joseki and the direction that he described as “economic thinking”.

8.5.4 Summary and Comparison With De Groot’s Construction Task

Even though the master participants’ performance on the episodic condition was affected by the speed of presentation, their performance remained at or near ceiling even at the 500 millisecond level. The master participants performed better in the episodic condition than the inferential condition of the construction task. Their performance was superior to the performance of the experienced participants of Experiments 8.1 and 8.2, both in the inferential condition and at a much shorter delay between stones in the episodic condition (500 milliseconds compared to 2 seconds). Thus, the results of master case studies confirmed the findings of Experiments 8.1 and 8.2 that performance on the construction task is related to Go skill.

From the verbal reports provided by the master players, it seems apparent that the tremendous memory ability of master Go players is possible, at least in part, because they remember moves by their meaning. Indeed, the meaning of the moves is an extremely important factor in the memory performance of master Go players.

Once again, a direct comparison with De Groot’s experiment is not possible. On a purely numeric basis, the performance of the 8d and 6d(1) in the inferential condition was similar to the performance of De Groot’s participants in the position construction condition of his experiment. The initial performance levels for both experiments are similar (36 - 41% for the master Go players compared to 34 - 35% for the chess players). The final performance level for the master Go players was 84% which was slightly lower than the performance levels of the chess players on the 10th trial (92 - 96%). The similarities are marked but may be coincidental. Further research would be required to test whether this was a trend across similar games or purely a coincidence for chess and Go.

8.6 General Discussion: Studies 1 and 2

The results of Studies 1 and 2 may be summarised as:

- performance on both the sequential pennies-guessing task and the sequential construction task is related to Go skill for all participants;
- the inferential condition is more difficult than the episodic condition for all participants on both tasks, that is, performance is lower (although less mental effort may be required);
- the sequential construction task is more difficult than the sequential pennies-guessing task (i.e., performance is lower); and
- master performance on the sequential construction task remains high even at very small delays between stones.

8.6.1 Memory for Moves

The meaning of moves is an important factor in strong Go players' memory for moves. Saito and Yoshikawa (1996) found that the purpose (or meaning) of moves is highly represented in Go players' protocols and that language label guided search dramatically reduces the search space during play. It may also be the case that "memory space" is reduced by the use of a memory analog of language label guided search. If encoding of a sequence of moves is based on the meaning of the moves (with the entire sequence "telling a story"), then the retrieval of the moves from memory would most likely involve the use of the meanings as cues.

It is common after teaching games for Go players to "replay" their game move by move as a way of learning how to play better. During such "replays", alternative sequences are explored so as to investigate "what-if" scenarios that may lead to better play in similar circumstances in future games. During my visit to Japan in 1996, I played against two professional players and asked them to replay our games from my perspective (i.e., rotated 180 degrees from their perspective). Whilst replaying our games, the professionals commented that some of my moves were difficult to remember because they "had no meaning". It may be possible that when the meaning of a move is not available, the search space is more spatial in nature and therefore larger.

When the two professionals replayed our games from a 180 degree rotated view, their recall was put under sufficient strain as to make the whole reconstruction process more amenable to external investigation. Individual moves were recalled from their original view point and then rotated. At some points during the reconstruction, a future position was recalled and the series of moves leading to that position was inferred. The role played by inference was made more obvious by the difficulty associated with the disparity between the mental image and the reconstructed position.

Weaker Go players are less likely to understand (or at best have a shallower understanding of) the meaning of the moves and are therefore more likely to rely on other strategies for remembering moves, including a spatial strategy. A spatial strategy would account for several of the findings from Studies 1 and 2. The sequential pennies-guessing task would tax spatial memory less than the sequential construction task because the stones are presented as cues, and it would therefore be easier. The episodic condition would be easier than the inferential condition in both tasks because spatial information would be conveyed during the sequential presentation of the stones. Further evidence for the spatial encoding of beginner versus experienced players is provided by the time component. The issue of time as a parameter for encoding meaning is illustrated by some of the beginners having difficulty seeing where a stone had been played even with 2000 milliseconds.

Chase and Simon (1973b) suggested that move sequences may be organised into little episodes related to a goal, and that episodes themselves could be organised hierarchically. The verbal reports of the master Go participants indicates that joseki sequences made remembering the sequence of moves easier. Joseki certainly fit the description of little sequences organised around a goal. Similarly, the notion that the participants constructed a “story” as the sequence unfolded is similar to the notion of hierarchically organising the episodes themselves.

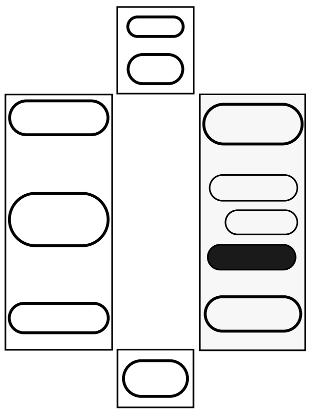
8.6.2 The Impact of Inference on Memory Performance

As for Study 1, it is not possible to separate the participants’ performance into episodic and inferential components, however, it appears that a component of memory performance may be attributable to inference. The results indicate that the utility of the episodic information is much

Chapter 9

Study 3: Learning Go Using an Interactive Memory Task

Introduction



During Studies 1 and 2, some of the participants unexpectedly exhibited and/or reported learning during the experiments. The observed learning phenomenon was interesting both as it related to learning Go specifically and as it related to learning in general. Learning Go is difficult and only a small percentage of people who start to learn Go continue on to become regular players. The drop-out rate, in a large part, appears to be due to two factors: the perceptual noise and confusion experienced by most beginners when they start learning Go, and the complexity of learning both perceptual and high-level concepts at the same time.

The learning observed during previous studies involved learning from perceptual inputs rather than from traditional instruction. Reber (1976), using an experimental paradigm designed to test artificial grammar learning, showed that participants learned to judge the grammaticality of novel strings generated from finite-state grammars more successfully by remembering sample strings of the grammar than by explicitly trying to extract the grammar. Reber's results indicated that an emergent by-product of attending to perceptual inputs is the formation of regularities in those inputs, which Reber in earlier work had called *implicit learning* (Reber, 1967).

The unexpected learning observed during Studies 1 and 2 raised the question of whether it was possible for beginners to improve on a skilled Go task through the implicit learning associated with attending to the perceptual aspects of remembering sequences of Go moves. This question was investigated in Study 3. In order to carry out the investigation, two new experimental tasks were developed.

The first new task, the interactive memory task, was essentially the sequential pennies-guessing task (see section 7.2.1.3) with a few minor changes. A 9x9 board was used instead of the 19x19 board because learning Go is easier on a smaller board. The number of stones presented was 10. In all other respects, the participants' task was the same: reconstruction of a sequence of Go moves with the stones acting as cues.

The interactive memory task is a memory task (remembering a sequence of Go moves) and an interactive reconstruction task (reconstructing the sequence of Go moves with the stones as cues) in which feedback is provided after each attempt. The name "interactive memory task" is used to differentiate the task from the sequential pennies-guessing task because the purpose of the tasks are different. The purpose of the interactive memory task is for the participants to acquire the regularities present in the opening game of Go rather than remembering sequences of Go moves.

The second new task, the prediction task, was once again conducted on a 9x9 board using elements of both the episodic and inferential conditions of the sequential construction task (see section 8.2.1.3). The stones from a position were presented sequentially to the participant whose task was to predict where the next stone in the sequence had been played. Thus, performance on the prediction task was expected to be related to Go skill just as for the inferential condition of the sequential construction task.

Experiment 9.1 comprise two case studies using beginner Go players. The results indicated that the interactive memory task had contributed to the participants' improved performance, that is, the participants had implicitly learned some of the regularities in the Go stimuli presented to them.

9.1 Experiment 9.1: Learning Go

The interactive memory task experimental sessions were conducted on 5 separate days to allow a sufficient amount of learning to occur without overloading the participants on any given day.

The hypothesis of the experiment was that the performance on the prediction task would improve between the pre-test and post-test as a result of the interactive memory task.

9.1.1 Method

9.1.1.1 Participants

The participants were unpaid volunteers:

- S1 was a 35 year old male Masters student who was a beginner Go player. S1 had only a small amount of Go experience (less than 30 games) and had not read any Go books.
- S2 was a 20 year old male undergraduate student who was a beginner Go player. S2 had only a small amount of Go experience (less than 5 games) and had previously participated in one of the unpublished pilot studies.

9.1.1.2 Materials

Professional 9x9 games were used as stimuli[1]. No game was used that contained a capture within the first 10 moves. From a main set of 244 games, two sets of 22 games (A and B) were randomly selected to be used for pre-test and post-tests and five sets of 9 games (1 - 5) were randomly selected to be used for the prediction task (i.e., 45 games in total). The games used for the interactive memory task were selected from the remaining games in the main set.

Computer software[2] specially written for the experiment was used to present the board positions to the participants. The participants were tested individually. Throughout the experiment, the participants were seated in front of a colour PC monitor with the experimenter present. As in the previous studies, a mouse-driven cursor was used to select stones during the test phase.

9.1.1.3 Procedure

Experimental sessions were conducted over 5 days. A pre-test was conducted prior to the experimental session on the first day and a post-test was conducted after the experimental session on the last day. The days for each participants varied: S1 Monday-Friday of one week; S2 Monday, Monday, Tuesday, Thursday, Friday over two consecutive weeks.

1. The games were played by young professional players under severe time restrictions (2 minutes and 30 seconds total time per player) and were broadcast by Yomirui Television in Japan. The game records were made available by Masahiro Okasaki of the Go Culture Study Group who administers the copyright on behalf of the Yomirui Telecasting Corporation.
2. The software was written by the author and is described in Appendix C.

Experimental Sessions

Each experimental session consisted of multiple repetition of two tasks: the interactive memory task, and the prediction task. Each session consisted of two interactive memory tasks (10 games each) interpolated with three prediction tasks (three games each). Thus, a total of 29 games were used per experimental session (20 for the two interactive memory tasks and nine for the three prediction tasks).

The Interactive Memory Task

The interactive memory task was similar to the pennies-guessing task (see 7.2.1.3) and comprised a presentation and a reconstruction phase.

Presentation phase: During the presentation phase, the position (comprising 10 stones) was cumulatively built by adding successive stones to the Go board on the computer monitor every 2 seconds in the order in which they had been played in the actual game. The stones were cleared from the board grid for 10 seconds.

Reconstruction phase: The reconstruction phase began with the board being redisplayed with all the stones visible, that is, with the entire final board position being visible. Thus, during the reconstruction phase, the participants were provided with the stones from the position as cues. The participants were asked to indicate the order in which the stones had been played (i.e., added to the Go board on the monitor). When an incorrect stone was selected (i.e., the stone selected was out of sequence or the wrong colour), an ‘x’ was displayed in the middle of the stone. When a correct stone was selected, the number of the move on which it was played was displayed in the middle of the stone and all stones with an ‘x’ in them were returned to solid colour.

The Prediction Task

The prediction task comprised a presentation and a prediction phase.

Presentation phase: The number of stones presented in presentation phase was between 3 and 7 inclusive, that is, the participants were asked to predict the 4th, 5th, 6th, 7th, or 8th stone of a given sequence. The presentation phase was similar to the presentation phase of the interactive memory task except that the test phase began immediately after the last stone was added (i.e., the stones were not cleared from view prior to the commencement of the test phase).

Prediction phase: The participants were asked to predict where the next stone in the game had been played. The participants selected empty board points using the mouse. If the point selected was the correct location of the next stone, a stone of the correct colour was placed on the point and the participant was invited to start the next board when ready. If the point selected was wrong, an ‘x’ was displayed on the board point[3] (see Figure 9.1). If the participant could not correctly identify the next stone after 10 attempts, a message would appear inviting the participant to start the next board when ready (i.e., the location of the next stone was not revealed to the participant).

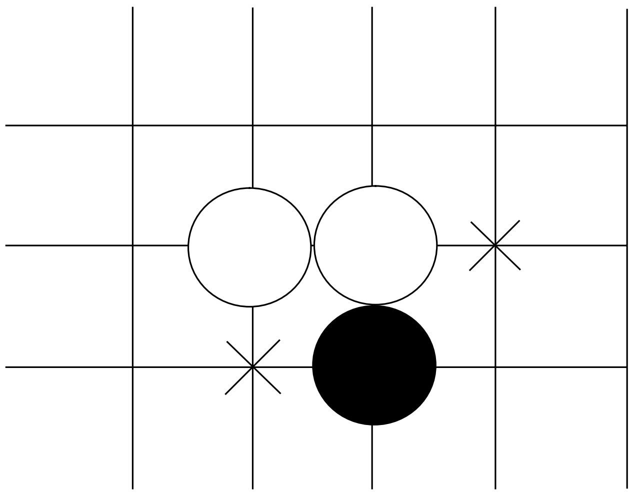


Figure 9.1: Feedback: wrong position selected. A cross (‘x’) was placed on a board position that was incorrectly selected as the position of the next stone in the sequence.

Pre-test and Post-test

The pre-test and post-test were prediction tasks. On the pre-test, the participants were instructed to use whatever Go knowledge they had to try and predict the position of the next stone. The participants were instructed similarly on the post-test, but were additionally instructed to use whatever they had learned during the five experimental sessions to try and predict the position of the next stone.

Each pre-test and post-test consisted of a unique set of 20 games; the participants were shown

3. Instead of placing a stone with an ‘x’ in it on the board point as was done in the sequential construction task, just the ‘x’ was placed on the board point. This arrangement was in response to participants who had commented that the incorrect “virtual” stones interfered with their ability to perform on the sequential construction task.

the task by the experimenter and practised on 2 games prior to testing. Within the 20 games, there were equal numbers of 4, 5, 6, 7, and 8 stone presentations (i.e., 4 games at each level). The presentation order was randomised so that no two consecutive presentations were of the same length. The presentation order was reversed between the pre-test and post-tests.

9.1.2 Results

During the pre-test and post-tests, the number of attempts to correctly identify the location of the correct stone were recorded for each board. The results for the participants S1 and S2 are shown in Figures 9.2 and 9.3 respectively. The number of attempts required to correctly identify the location of the next stone for each board in the pre-test and post-test are shown in rank order. The vertical axis represents the number of attempts taken to correctly identify the location of the next stone, from 1 to 10, with “floor” indicating those boards for which the correct location was not identified within 10 attempts. Each tic on the horizontal axis represents 1 board. The boards are ordered by rank from best performance (1st attempt correct) to worst performance (not successful within 10 attempts). Of particular interest in Figures 9.2 and 9.3 are the number of boards at ceiling[4], the number of boards at floor[5], the median number of attempts, and the relative performance between pre-test and post-test on the interval boards.

S1’s results, presented in Figure 9.2, can be summarised as follows:

- the number of boards at ceiling increased from 4 to 6 boards;
- the number of boards at floor reduced from 6 to 4 boards;
- the total number of attempts reduced from 113 to 97 (with floor being counted as 10 attempts);
- the median number of attempts for the 20 boards reduced from 6 to 4; and
- the performance on the post-test for each rank is equal or better than the performance on the pre-test across all boards.

S2’s results, presented in Figure 9.3, can be summarised as follows:

- the number of boards at ceiling remained constant at 3 boards;
- the number of boards at floor reduced from 8 to 3 boards;
- the total number of attempts reduced from 145 to 110 (with floor being counted as 10

4. Ceiling performance was identifying the correct stone on the first attempt.
5. Floor performance was not correctly identifying the correct stone within 10 attempts.

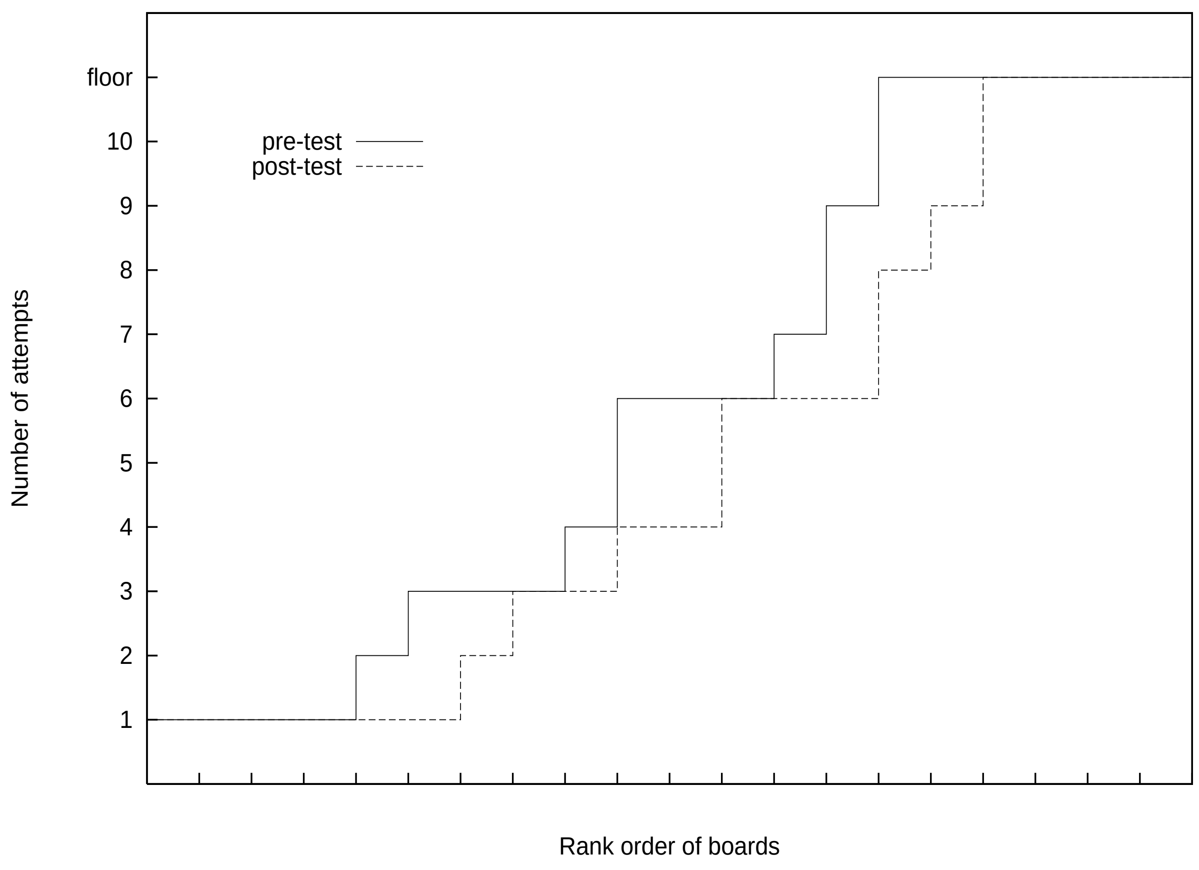


Figure 9.2: Results: S1. The number of attempts required to correctly identify the location of the next stone for each board in the pre-test and post-test in rank order. The performance on the post-test shows improvement compared to the performance on the pre-test.

attempts);

- the median number of attempts for the 20 boards reduced from 10 to 6;
- the performance on the post-test for each rank is equal or better than the performance on the pre-test across all boards.

9.2 Discussion

The performance of both participants was better on the post-test than on the pre-test. At the gross level, the performance on the post-test for each rank was equal to or better than the performance on the pre-test across all boards for both participants. At a finer level, the number of boards at floor, the total number of attempts, and the median number of attempts fell for both

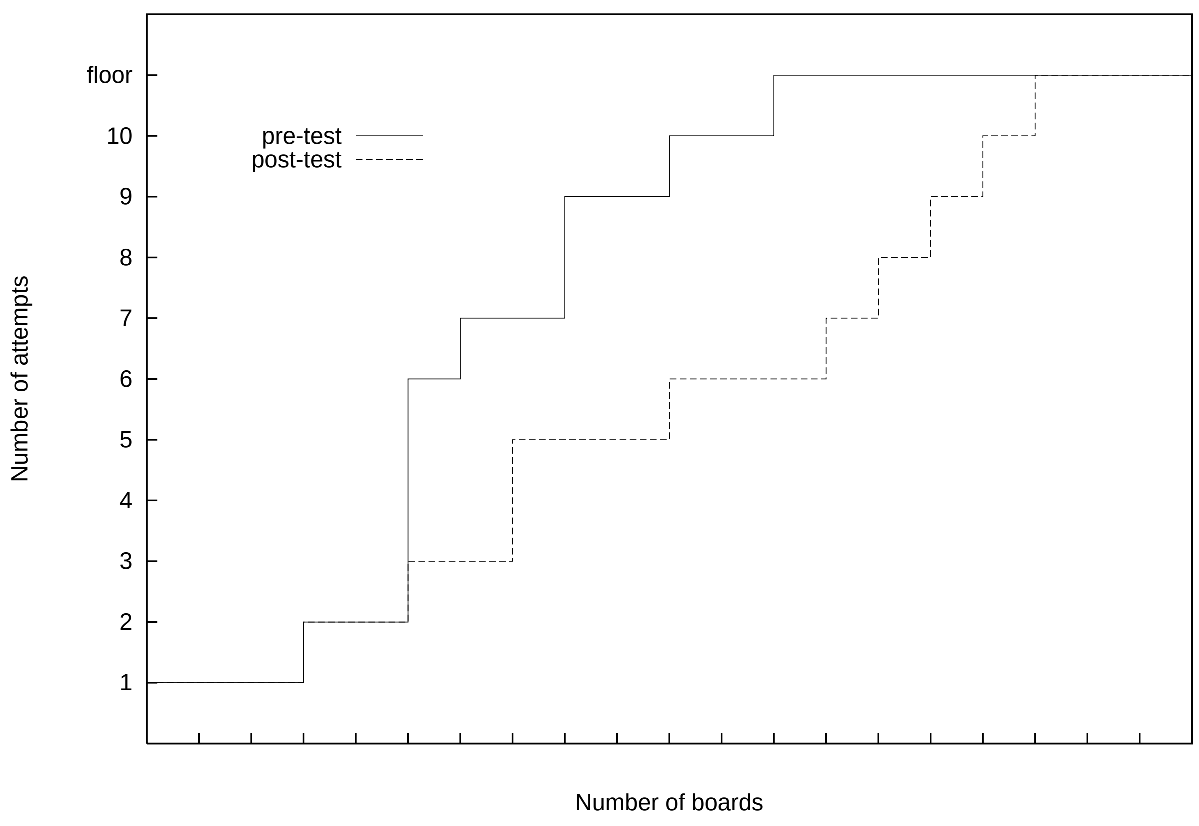


Figure 9.3: Results: S2. The number of attempts required to correctly identify the location of the next stone for each board in the pre-test and post-test in rank order. The performance on the post-test shows improvement compared to the performance on the pre-test.

participants. The number of boards at ceiling rose for S1 and remained constant for S2.

In interviews conducted after the post-test, both participants expressed the view that the experiment had been beneficial in learning Go and said that they had become more aware of familiar sequences during the experimental sessions. The participants also reported that during the prediction task they had made predictions based on their familiarity with the sub-sequences.

These preliminary results are interpreted to indicate that the interactive memory tasks performed during the five experimental sessions between the pre-test and post-test were of benefit to the participants, that is, performing the interactive memory task enhanced the participants' performance.

The interactive memory task is similar in nature to a traditional Japanese learning method called *soufu*. The soufu technique involves reconstructing a Go game from a diagram (with each stone numbered in game order) by placing stones on an empty board as if the learner was playing both sides of the game. The soufu technique is used by the Japanese Go community as an explicit learning task and makes learning Go easier for beginners by removing some of the initial noise and confusion they experience. By showing beginners only good Go sequences, their search space of possible moves is constrained. The learning technique developed in Study 3 is a modification of the traditional soufu task by making the learning implicit rather than explicit, and incorporating the automation of stone placement.

An important aspect to the implicit learning is the regularity within the stimuli. Many beginners learn by playing rather poorly against better players. Thus, they are constantly seeing their own poor play which contains no regularity and “contaminates” the regularity present in their opponents’ play. In Go there is a large amount of both high-level and low-level regularity. An example of high-level regularity is joseki sequences. The participants commented that they had gained familiarity with some of the joseki sequences over the course of the experimental sessions (several joseki sequences were seen repeatedly). In Go there is also regularity at the low level, that is, how and where stones are played exhibits some degree of regularity. An example of low-level regularity is that early in the game (i.e., in the stimuli seen by the participants), Go players do not play on the outside lines of the board. The participants seem to have implicitly learned some of the regularities present in the Go stimuli to which they were exposed.

It was anticipated that interleaving a small number of prediction tasks with the interactive memory task during the daily experimental session would act as a subconscious or implicit incentive for the participant to try and remember the sequence of stones presented in the interactive memory task. Although the impact of interleaving the interactive memory task with the prediction task was not explicitly tested, anecdotal evidence suggests that the interleaving facilitated the acquisition of the regularities by the participants. S2 in particular commented that the prediction tasks made him think more about what he had seen. S2 had previously participated in a pilot study in which there was no prediction tasks during the experimental sessions and he was of the opinion that the current procedure was superior because of the introduction of the prediction task into the experimental sessions. It may be possible in future work to explicitly test the impact of interleaving the interactive memory task with the prediction task.

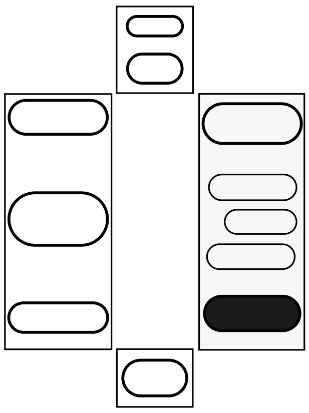
In summary, based on the participants' improved performance and their comments, the interactive memory task was instrumental in their learning some of the regularity present in the stimuli that was presented to them. The learning is similar in nature to that described by Reber as implicit learning (Reber, 1967).

Chapter 10

Go as a Domain for Studying Sequential Aspects of Memory and Learning

Introduction

This chapter concludes Section III in which Go has been assessed as a research domain for Cognitive Psychology. The assessment was conducted with respect to the assessment questions CP1 - 4 from section 1.2.2. An overview of the results of Studies 1, 2 and 3 (Chapters 7, 8 and 9 respectively) is provided in section 10.1 along with a statement regarding the validity of the results. The results are discussed in section 10.2 and the conclusion in section 10.3 is that based on having successfully addressed the assessment questions CP1 - 4, Go provides a good research domain for Cognitive Psychology.



10.1 Summary of Results

The results of the experiments and case studies comprising Studies 1 and 2 can be summarised as follows:

- performance on both the sequential pennies-guessing task and the sequential construction task is related to Go skill for all participants;
- the inferential condition is more difficult than the episodic condition for all participants on both tasks;
- the sequential construction task is more difficult than the sequential pennies-guessing task;
- master performance on the sequential construction task remains high even at very small delays between stones (e.g., 500 milliseconds);
- the utility of the episodic information appears to be much higher in the sequential

construction task than it is in the sequential pennies-guessing task; and

- a proportion of memory performance on the sequential construction task may be attributable to inference or problem solving.

The results of Study 3 indicated that the interactive memory task had contributed to the participants' improved performance, that is, the participants had implicitly learned some of the regularities in the Go stimuli presented to them.

10.1.1 Validity of Results

Throughout Studies 1 to 3, the sample sizes in the experiments and case studies were small but comparable to the sample sizes used by Chase and Simon and by De Groot. Any given experiment or case study in isolation would not be sufficient grounds for the conclusions reached. However, the whole body of data reported plausibly demonstrates that:

- the sequential pennies-guessing task plausibly replicates Chase and Simon's (1973a, 1973b) results with respect to an effect for skill;
- the sequential construction task plausibly replicates De Groot's (1966, 1996) construction task with respect to an effect for skill.

10.2 Discussion

One of the motivating factors behind Studies 1 and 2 was to investigate the impact of inference on memory performance, and in particular, in the context of memory for move sequences. Studies 1 and 2 involved explicit memory tasks and hence, a traditional view would attribute performance to the explicit retrieval of stored memories. However, unlike Chase and Simon (1973a, 1973b) who dismissed inference as having no significant role in memory performance, the results of Studies 1 and 2 have shown that a component of inference is involved. That is, the information retrieved has not necessarily been explicitly retrieved from the memory system.

Inference *in the context of memory performance* may be viewed in two different ways. Inference may be considered to be reconstructed memory, that is, that memory performance consists of both explicit and reconstructed memories. Alternatively, inference may be considered to be a continuum from low-level reconstruction within the memory system through to high-level problem solving. At the high level, the expert would consciously or unconsciously draw on their Go knowledge (e.g., knowledge of joseki sequences) to determine where the next stone was played. The high-level problem solving or inference constrains, or in some way interacts with, the low-level explicit memory storage. The techniques developed and used in Studies 1 and 2 do not enable the participants' performance to be explicitly separated into memory and inference components.

Memory performance for move sequences appears to be related to the participants' ability to create a dialogue or story as the sequence is presented to them. The master players were able to understand the meaning of the moves as they were presented and were therefore able to construct a story as the sequence progressed. Reconstruction of the sequence seems to involve recalling the story, and therefore dramatically constrains the space of possible choices for each successive move. For beginners, however, the strategy was largely spatial since they have no significant Go knowledge to be able to classify moves and create a story. Thus they are forced into trying to remember the move sequences largely based on spatial cues which results in their poor performance. Chase and Simon (1973b) conducted a small study into memory for moves and concluded that move sequences are most likely organised into episodes related to a goal. The results of Studies 1 and 2, and in particular the case studies of the master Go players, support Chase and Simon's observation.

Implicit learning reported by participants during Studies 1 and 2 resulted in the development of the interactive memory task and the prediction task in Study 3. Memorising sequences of moves seems to contribute to implicit learning of the regularities present in Go. Future work is needed to explore this issue more fully.

10.3 Conclusions

The specific assessment questions for assessing Go as a research domain for Cognitive Psychology were provided in section 1.2.2, and are repeated below:

CP1 *To what extent does the domain of Go facilitate the investigation of memory for moves? To what extent does the domain of Go facilitate the investigation of the impact of inference on memory performance?*

CP2 *What type of Cognitive Psychology research issues can be investigated in the* *domain of Go?*

CP3 *Do interesting research questions arise from investigations of memory for moves within the domain of Go? Do interesting research questions arise from investigations of the impact of inference on memory performance within the domain of Go?*

CP4 *Do generalisations useful to Cognitive Psychology arise from research conducted within the domain of Go? What types of generalisations can be made beyond the domain of Go?*

CP1

The sequential pennies-guessing task and the sequential construction task developed for the experiments and case studies conducted in Studies 1 and 2 facilitated the investigation of the impact of inference on memory performance and the investigation of the factors that affect memory for moves. Improvements to the techniques suggested in future work (see section 11.4) may enable a clearer separation of performance in remembering and reconstructing Go sequences into memory and inference. Thus, question CP1 has been successfully addressed.

CP2

The experimental studies presented in this thesis have identified several Cognitive Psychology research issues that can be investigated using Go as a research domain:

- Go provides a good domain for investigating memory for sequences because the complexity of piece movement is not present as it is in chess, and the impact of capture can be managed by careful selection of stimuli.

- Go facilitates the investigation of the impact of inference on memory performance, and in particular, the sequential pennies-guessing and construction tasks have proven useful in this endeavour. Further work is needed to develop experimental techniques that enable inference performance to be clearly separated from purely memory performance.

- Go enables the investigation of expert performance and issues related to expert performance. Besides the performance of master Go players reported in this thesis, Saito and Yoshikawa (Saito & Yoshikawa, 1995, 1996, 1997; Yoshikawa & Saito, 1997a, 1997b) have also investigated the performance of master Go players on tasks such as perception, evaluation, problem solving and the use of language.

- Learning Go may be facilitated through implicit learning. The interactive memory task and the prediction task provide experimental tasks suitable for exploring the issue of implicit learning. Further work is needed to determine the extent to which implicit learning can be employed in Go, and to what extent it can be generalised to other domains.

These issues do not exhaust the possible Cognitive Psychology issues that could be investigated using Go as a research domain.

Concurrently with the work presented here (1995-1999), Saito and Yoshikawa were using Go as a research domain for their Cognitive Science studies. They advocate the use of Go as a testbed for Cognitive Science studies and in so doing, identify issues within Go that exist in Cognitive Psychology (Saito & Yoshikawa, 1997). They identify the characteristic features of Go from a cognitive perspective as perception, problem solving, and knowledge representation which are issues of importance to Cognitive Psychology. Saito and Yoshikawa further identify promising research themes in Cognitive Science using Go as a research domain as: the relation between spatial recognition ability and Go skill; the relation of higher level concepts to board configurations and move sequences; planning, plan modification and plan interaction; pattern recognition versus language level inference; how Go players evaluate Go positions; attention; the extraordinary memory performance of expert Go players; quick recognition of complex stone configurations; very quick generation of a very few good candidate moves; how players build an opponent model; learning; and the role of emotion and meta-cognition. Thus, question CP2 has been successfully addressed.

CP3

Reports of incidental learning observed during Experiments 7.1, 8.1 and 8.2 prompted the investigation of implicit learning conducted in the case studies in Study 3. Thus, investigation of research questions within the domain of Go gave rise to other interesting research questions and question CP3 has been successfully addressed.

CP4

The findings of Studies 1 and 2 extend the findings of cognitive studies in chess (Chase & Simon, 1973a, 1973b; De Groot, 1966; De Groot & Gobet, 1996) from spatial patterns to sequential ones. Similar findings regarding effect for skill and the extraordinary performance of master participants were also observed. Thus, the findings regarding the impact of inference on memory performance and memory for moves contribute to the understanding of the human memory system and human memory performance including expert memory performance just as previous work in chess has done.

Study 3 examined an issue that arose as a result of the nature of Go itself, namely, that the perceptual nature of Go makes it seem chaotic or noisy and therefore difficult to master for beginners. In this respect, Go is similar to many other domains in which mastery of a perceptual interface is gained together with a mastery of the cognitive model of the system.

The generalisations from within the domain of Go have an impact outside the domain itself, and the issues within the domain of Go exist elsewhere in Cognitive Psychology. Thus, generalisations useful to Cognitive Psychology do arise from research conducted within the domain of Go and there are potentially many cognitive studies that could be conducted using Go as a research domain which may similarly give rise to useful generalisations. Thus, question CP4 has been tangentially addressed although obviously much more work remains to be done.

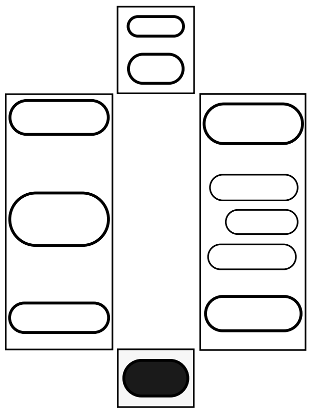
Go Provides a Good Research Domain for Cognitive Psychology

In summary, having successfully addressed the assessment questions CP1 - 4, it can be said that the work reported in Section III demonstrates that the game of Go provides a good research domain for Cognitive Psychology.

Chapter 11

Conclusions, Contributions, and Future Work

Introduction



As stated in Chapter 1, games such as chess have long been accepted as research domains in Cognitive Science because they can be formally specified and provide non-trivial domains without all the problems associated with real world complexity. The over-arching goal of this thesis has been to assess the game of Go as a research domain for Cognitive Science by assessing it as a research domain for AI (Section II) and Cognitive Psychology (Section III).

Building on the conclusions of Sections II and III, this chapter assesses the potential for the exchange of ideas between the AI and Cognitive Psychology disciplines afforded by Go as a research domain (section 11.1). The conclusion of this thesis, namely that the game of Go provides a good research domain for Cognitive Science is argued in section 11.2. The contributions of this thesis are summarised in section 11.3 and future work is suggested in section 11.4.

11.1 Synergy: Exchanging Ideas Between the Computer Go and Cognitive Psychology Fields

The assessment question CS1 from section 1.1.1 (repeated below) is addressed by considering the potential for synergy between Computer Go and Cognitive Psychology fields.

CS1 *In what ways does the domain of Go facilitate an exchange of ideas between disciplines, that is, between AI and Cognitive Psychology?*

A potential synergy exists between AI and Cognitive Psychology research into Go. The Computer Go community can benefit from Cognitive Psychology research in the form of cognitively inspired algorithms (11.1.1) and the Cognitive Psychology community can benefit from Go programs in the form of computationally inspired cognitive models (11.1.2).

11.1.1 Cognitive Inspiration for Programmers

It is common for the programmers of the best Go programs to be strong amateur dan players and for the programs to improve as the programmer's Go skill improves. The knowledge acquisition bottleneck commonly experienced in the expert system field is also found in the Computer Go field. Knowledge of good style from books or experts does not seem to be enough - an introspective analysis of how they play Go is a definite advantage to programmers in being able to incorporate Go knowledge into their programs.

There is interest amongst some Go programmers for cognitive inspiration in designing their algorithms, that is, they are interested in how Go players represent the board, choose moves, evaluate positions etc. Since programming Go is such a knowledge intensive endeavour, Go programs tend to be an implicit expression of a programmer's theory on how people play Go, or more specifically, how the programmer plays Go. Any additional knowledge that psychologists can provide has the potential to lead to improvements in Go algorithms. Three approaches have explicitly taken into account how humans play Go:

- Replicating their own theory of human Go perception and cognition in a Go program was the aim behind Walter Reitman and Wilcox's research (see section 3.1.3). Their research project resulted in the INTERIM.2 program, which was the predecessor to the early commercial Go program Nemesis. The INTERIM.2 program played at around the 27kyu level.

- The interaction of perception and cognition in the solution of complex problems was studied by Friedenbach by using Go as a domain (Friedenbach, 1980). He formalised the notion of abstraction hierarchies which are based on graph theory and then applied them to Go. The choice of Go as a domain, according to Friedenbach, was because concepts in the Go literature are related in a multi-level hierarchy (which is consistent with Judith

Reitman's perceptual studies described in section 6.2.2). Friedenbach described his program as "a very weak player".

- Building a Go program based on their insights into how humans play Go was one of the long-term intentions that Saito and Yoshikawa had for their Cognitive Psychology Go research (see section 6.2.3). Yoshikawa has recently begun competing in Computer Go tournaments with an early prototype of the program he is building although it is not yet competitive at the highest level.

In summary, there is a strong motivation to pursue the understanding of how humans perform tasks such as evaluating life and death, perform pattern matching, learn and use joseki (especially the interaction of joseki with other groups on the board), etc.

11.1.2 Computational Inspiration for Cognitive Modelling

Since the emergence of computers, psychological theories of information processing have drawn inspiration from the data structures and algorithms developed by the Computer Science field. In the Computer Go domain, programmers have had to deal with the combinatorial problems of Go, and the limits of tournaments[1], and necessarily have been required to develop representations and procedures to deal with the interaction between knowledge and search. Although not necessarily initially psychologically motivated, they do provide candidate computational components for developing Psychological theories.

Just as chess was important in the development of theories of chunking, so Go has the potential to be instrumental in the development of theories for sequence processing and other temporal aspects. Go has several characteristics that make it a more attractive domain for these types of investigations than chess:

- Since Go is a game of addition rather than attrition like chess, each subsequent move in a sequence results in relatively little change to the Go board (other than the infrequent situations when strings are captured). This means that if a final position in which no strings have been captured is presented to a participant, all the intermediate positions in the

1. The time limits on Computer Go tournaments are more severe than for Computer Chess tournaments (see point 13 in Table 2.1 of Chapter 2).

sequence leading to the final position are represented in the final position. In chess, due to the mobility of pieces and the frequency of capture, many intermediate positions are not represented in a final position. Thus, although the investigation of the sequence processing performance of master participants would be expected to be of similar efficacy in either Go or chess, the investigation of the sequence processing abilities of weaker players is easier in Go. In sum, the cumulative construction of positions simplifies the research paradigm enormously.

- Ladders are an important aspect of Go[2] and require the ability to "read" the repeating structure of the capture race across the board in order to determine who will win. Since the winner of a ladder can be determined by reading, they rarely occur in high-level games and are therefore not amenable to learning by example. Programs use special purpose procedures to readout ladders. The challenge for good programs (and players) is to assess the interaction between the ladder as it unfolds across the board and ladder-breaker stones. Such an assessment cannot be contained within the ladder procedure and requires the ability to successfully deal with the distinction between local and global information (Burmeister, Wiles & Purchase, 1995a, 1995b, 1999).

- Joseki (set patterns of local play in the opening stages of a game) could be viewed as "sequential chunks". However, there are two challenges presented by josekis: interaction and deviation. Joseki sequences are locally optimal but interact with other stones on the board (e.g., ladder-breakers), and in extreme cases, the interactions can be with stones on the diagonally opposite corner of the board. A deep understanding of a joseki (rather than a surface rote adherence to the joseki sequence) is necessary to enable capitalisation on an opponent's deviation from the locally optimal joseki moves. Go programs are de facto theories of the use of such knowledge and dealing with deviation in such conceptual chunks. Human learners have similar problems, often decreasing two stones in strength temporarily during the stage of development where they learn josekis.

In summary, psychological theories need to incorporate flexible, complex structures for dealing with temporal processes. Computer Go programmers have developed a variety of such flexible

2. According to one ancient Japanese Go proverb, if a player does not understand ladders they should not play Go.

complex structures, many of them based on their own personal knowledge of Go.

11.1.3 Summary

Currently there is a small level of synergy between the Computer Go and Cognitive Psychology Go research fields. However, there is the potential for strong synergy in the future, particularly when compared to the synergy that actually eventuated in the chess field. Unlike chess, a computational shortcut providing expert human-like performance does not seem likely in the short-term which means that the interest in understanding human Go performance will remain.

11.2 Conclusion: Go is a Good Research Domain for Cognitive Science

Good research domains are necessary for long-term research progress in a field: research questions are investigated within research domains, and investigation within research domains generates research questions that extend beyond the original domain. Go is such a domain for investigating high-level perception, and the interaction between knowledge and search, by conducting case studies of Go programs, and empirical studies of inference, memory, and learning in Go.

The primary qualities of good research domains are that they facilitate the investigation of research questions, giving rise to new research questions, and that useful generalisations can be made beyond the domain itself.

Within interdisciplinary fields like Cognitive Science, good research domains enable interaction and exchange between the disciplines. As outlined in section 11.1.3, the synergy between the Computer Go and the Cognitive Psychology Go research fields is still weak but has good potential.

Having reached the conclusions that Go is a good research domain for AI and Cognitive Psychology in Sections II and III respectively, and having discussed their interaction above, it is now possible to conclude that the game of Go provides a good research domain for Cognitive Science, particularly in the areas of the interaction of search and knowledge in combinatorially large domains, and human performance in dealing with such complexity.

11.3 Summary of the Contribution of This Thesis

Contributions to AI

This thesis is unusual in AI in that it involves the study of Go programs rather than the coding of a Go program. It provides the most comprehensive survey of the state-of-the-art in the Computer Go field to date. The survey of the Computer Go field provided in Section II describes:

- an extensive survey of the Computer Go literature including a description of the early Go programs (Chapter 3), case studies of MFG and Go4^{++} (Chapter 4), and a review of current competitive Go programs with respect to state representation, move generation, goal states, evaluation, tactical search, and influence functions (Chapter 4);
- an extension of the literature with new information about current Go programs gained through comprehensive personal interviews with the Go programmers from the Second FOST Cup (1996) in Tokyo and the Third FOST Cup (1997) in Nagoya, and through collaboration via email.

For the Go programmer, the issues addressed in this thesis primarily concern knowledge, search and their interaction in Computer Go, and summarised:

- the issues that Go programming research projects must consider (e.g., life and death, pattern matching and pattern databases);
- the approaches that are current state-of-the-art in Computer Go; and
- the approaches that do not work in Computer Go (e.g., brute-force search, learning based on experiential learning alone).

These descriptions also afford advantages for future Go programming projects by indicating:

- what approaches have been tried unsuccessfully and why they failed;
- the successful approaches; and
- the common issues that must be addressed.

Thus, the case studies provide a starting point and solid foundation of ideas and techniques for future Go programmers.

Contributions to Cognitive Psychology

As indicated in the survey of the Go Cognitive Psychology research field in section 6.2, there have been relatively few studies conducted using Go as a research domain. This is the first thesis to explore the impact of inference on memory performance in Go, factors that affect memory for Go moves, and the use of implicit learning associated with interactive memory tasks as a learning technique for Go. New experimental techniques were developed to enable these investigations to be conducted.

This thesis contributes to the development of Go as an empirical research domain by:

- providing a survey of the Go Cognitive Psychology research field (Chapter 6) summarising the cognitive studies conducted using Go as a research domain;
- the completion of three experimental studies consisting of four experiments and three case studies (Chapters 7 though 9) that investigated the issues of the impact of inference on memory performance and memory for Go moves;
- the development of new experimental paradigms based on the sequential pennies-guessing task (Chapter 7) and the sequential construction task (Chapter 8) for investigating the impact of inference of memory performance and for testing memory performance for sequences of Go moves;
- demonstrating that inference has an impact on memory performance and proposing that the nature of inference is that of a continuum from low-level perception or generalisation to high-level problem solving;
- demonstrating that the performance on both the sequential pennies-guessing task and the sequential construction task is related to Go skill for all participants (Chapters 7 and 8);
- demonstrating that expert memory performance can be investigated in Go, and providing experimental tasks to facilitate such investigations;
- demonstrating that master performance on the sequential construction task remains high even at very small delays between stones (Chapter 8);
- providing and analysing protocols from master participants (Chapter 8); and
- the development of a potential technique for learning Go based on interactive memory tasks (Chapter 9).

11.4 Future Work

In a good research domain, research investigation gives rise to further interesting research questions and issues. Below are issues that build directly on this work:

- This thesis has elucidated characteristics of Go that require interaction between knowledge and search (i.e., ladders, ko, life and death, joseki). Assessing other games that contain some of these characteristics but without all of the complexities of Go will facilitate a better understanding of the individual impact of each characteristic and their collective interaction in Go. Candidate games include Ponnuki Go[3] and Hex:
    1. Ponnuki Go is played on a 9x9 board and is used as an intermediate step in the learning process for new Go players (particularly for children). Ponnuki Go is similar to Go except that the first player to capture a stone or string wins. Life and death is a part of the game, although it is considerably simplified because sacrificing stones is not a winning strategy. Ladders are also a part of the game and are more immediately serious than in Go, since any stone caught in a ladder situation results in the loss of the game. Necessarily, ko is not a part of the game since the first capture ends the game. If passes are forbidden, the game devolves into a struggle for territory.
    2. Hex is similar to Go but is played on a hexagonal grid. The opposing players try to connect the opposing sides of the board with stones (one player connecting the “North-South” sides and the other connecting the “East-West” sides). Stones cannot be captured. Similarities with Go include ladders, ladder-breakers, and links/connections. Hex is primarily a game of tactics rather than a game of long-range strategy.

- Continued elucidation of new and existing algorithms in the Computer Go field so as to facilitate their possible use in Psychological theories.

- Further investigation into the memory and inference performance of master Go players. In particular, testing their ability on very short (e.g., 150 - 250 milliseconds) inter-stone latencies. Such tests require the development of software to enable reliable presentation at very short inter-stone latencies. As the latencies decrease to very short intervals, the facilitation of the presentation of the sequence in the episodic condition over the inferential

3. Also known as First Capture Go, Sudden Death Go or Capture Go.

condition should diminish. A consequence of shorter latencies would be that the difference in performance levels between these two conditions should also diminish. Such conditions would enable a more detailed investigation of the contribution made by inference and memory to performance on sequential memory tasks.

- Further testing to determine the upper limit of sequence lengths that can be reliably reconstructed. Players routinely replay games after their completion where relatively long latencies naturally occur during game play, but it is unclear whether an entire game could be reconstructed after being presented with relatively short inter-stone latencies. Such testing would be useful in investigating whether the dialogue or story constructed by players as a game progresses is primarily linear or whether the meaning of early moves are refined and reinterpreted in the light of later moves.

- Further testing of the interactive memory learning technique and investigation into its possible use with other Go learning techniques may provide an opportunity to explore issues involved with the early acquisition of expertise. Of particular interest is whether the technique, when used in tandem with traditional theory-based teaching techniques, will facilitate the perceptual learning of beginners. This would be of great benefit in reducing the confusing nature of the visual aspect of the game experienced by many beginners.

- Further testing to determine whether the interleaving of the interactive memory task and the prediction task facilitate learning Go.

- Further testing and calibration of the prediction task to facilitate its use in Cognitive Psychology as a potential experimental method for estimating the rank of Go players.

In summary, there are benefits of future work for both the AI and Cognitive Psychology fields. Computational benefits would result from investigating the issues of ladders, ko, life and death, joseki etc. in different games, allowing the issues to be teased apart and possibly studied in isolation and minimising the complications resulting from their interaction. Refinement of the new empirical techniques developed in this thesis would enable further investigation of the nature of inference and its impact on memory performance and the potential use of the interactive memory technique for learning.

References

Allis, L. V., van den Herik, H. J. and Herschberg, I. S. (1991) Which games will survive? In D. N. L. Levy and D. F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2-The Second Computer Olympiad*, pages 232-243. Ellis Horwood.

Bate, J. A. (1997) A beginner's introduction to Go.
Retrieved from http://www.cs.umanitoba.ca/~bate/BIG/BIG.contents.html on 6 October 1999.

Berlekamp, E. and Wolfe, D. (1994) *Mathematical Go: Chilling gets the last point*. A K Peters, Wellesley, Massachusetts.

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

Brügmann, B. Monte Carlo Go. (1993) Monte Carlo Go. Retrieved from
ftp://rzis1.rz.tu-bs.de/pub/go/comp/mcgo.tex.gz on 6 October 1999.

Burmeister, J. and Wiles, J. (1995) The challenge of Go as a domain: A comparison between Go and chess. In *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, pages 181-186, Perth. IEEE Western Australia Section.

Burmeister, J., Wiles, J. and Purchase, H. (1995a) The integration of cognitive knowledge into perceptual representations in Computer Go. In H. Matsubara, editor, *Proceedings of the Second Game Programming Workshop in Japan '95*, pages 85-94, Kanagawa. Computer Shogi Association.

Burmeister, J., Wiles, J. and Purchase, H. (1995b) On relating local and global factors: A case study from the game of Go. In *Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems*, pages 187-192, Perth. IEEE Western Australia Section.

Burmeister, J. and Wiles, J. (1996) The use of inferential information in remembering Go positions. In H. Matsubara, editor, *Proceedings of the Third Game Programming Workshop in Japan '96*, pages 56-65, Kanagawa. Computer Shogi Association.

Burmeister, J., Wiles, J. and Purchase, H. (1999) The integration of cognitive knowledge into a perceptual representation: Lessons from human and Computer Go. In J. Wiles and T. Dartnall, editors, *Perspectives on Cognitive Science: Theories, Experiments and Foundations, Volume II*, pages 239-257. Ablex, Connecticut.

Burmeister, J. and Wiles, J. (1999) AI techniques used in Computer Go. In R. Heath, B. Hayes,

A. Heathcote and C. Hooker, editors, *Proceedings of the Fourth Conference of the Australasian Cognitive Science Society*, University of Newcastle. (Distributed as a CD-ROM.)

Burmeister, J., Saito, Y, Yoshikawa, A. and Wiles, J. (2000) Memory performance of master Go players. In H. J. van den Herik and H. Iida, editors, *Games in AI Research*, pages 271 - 286. Universiteit Maastricht.

Cazenave, T. (1996a) Automatic ordering of predicates by metarules. In *Proceedings of the 5th International Workshop on Metaprogramming and Metareasoning in Logic*, Bonn 1996, post JICSLP workshop.

Cazenave, T. (1996b) Learning to forecast by explaining the consequences of action. In the *Proceedings of the First International Workshop on Machine Learning, Forecasting, and Optimization*, pages 29-38.

Cazenave, T. (1996c) Self fuzzy learning. International Workshop on Logic Programming and Soft Computing, Bonn 1996.

Chan, W. King, I. and Lui, J. (1996) Performance analysis of a new updating rule for TD(λ) learning in feedforward networks for position evaluation in Go game. In *IEEE International Conference on Neural Networks, volume III*, pages 1716-1720, Washington D.C. IEEE Computer Society.

Charness, N. (1992) The impact of chess research on cognitive science. *Psychological Research*, 54(1), pages 4-9.

Chase, W. G. and Simon, H. A. (1973a) Perception in chess. *Cognitive Psychology*, 4, pages 55-81.

Chase, W.G. and Simon, H. A. (1973b) The mind's eye in chess. In W. G. Chase, editor, *Visual Information Processing*, pages 216-281. Academic Press, New York.

Chen, K. (1989) Group identification in computer Go. In D. N. L. Levy and D. F. Beal, editors, *Heuristic Programming in Artificial Intelligence: the First Computer Olympiad*, 195- 210. Ellis Horwood, Chichester.

Chen, K. (1990) The move decision process of Go Intellect. In D. Erbach, editor, *Computer Go*, 14: 9-17.

Chen, K (1992) Attack and defence. In H. J. Van den Herik and L. V. Allis, editors, *Heuristic Programming in Artificial Intelligence 3-The Third Computer Olympiad*, pages 146-156. Ellis Horwood, Chichester.

Eisenstadt, M. and Kareev, Y. (1975) Aspects of human problem solving: the use of internal representations. In D. A. Norman and D. E. Rumelhart, editors, *Explorations in Cognition*, pages 308-346. W. H. Freeman, San Francisco.

Eisenstadt, M. and Kareev, Y. (1977) Perception in game playing: internal representation and scanning of board positions. In P. N. Johnson-Laird and P. C. Wason, editors, *Thinking: Readings in Cognitive Science*. Cambridge University Press, Cambridge, Massachusetts.

Enderton, H. D. (1991) The Golem Go program. Technical report CMU-CS-92-101, Carnegie Mellon University.

Enzenberger, M. (1996) The integration of a priori knowledge into a Go playing neural network. Retrieved from http://home.t-online.de/home/markus.enzenberger/neurogo.html on 6 October 1999.

Fotland, D. (1986) The program G2. *Computer Go,* 1.

Fotland, D. (1993) Knowledge representation in The Many Faces of Go. Retrieved from ftp://rzis1.rz.tu-bs.de/pub/go/comp/mfg.gz on 6 October 1999.

Friedenbach, K. J. (1980) Abstraction hierarchies: A model of perception and cognition in the game of Go. PhD., University of California, Santa Cruz.

Groot de, A. (1965) *Thought and Choice in Chess* (1st edition). Mouton, The Hague.

Groot de, A. (1966) Perception and memory versus thought: Some old ideas and recent findings. In B. Kleinmuntz, editor, *Problem Solving: Research, Method, and Theory*, pages 19-50. John Wiley, New York.

Groot de, A. (1978) *Thought and Choice in Chess* (2nd edition). Mouton, The Hague.

Groot de, A. and Gobet, F. (1996) *Perception and memory in chess*. Van Gorcum, Assen, The Netherlands.

Lichtenstein, D. and Sipser, M. (1980) Go is polynomial-space hard. *Journal of the ACM*, 27(2):393-401.

Lehner, P. E. (1981) Planning in Adversity: A Computational Model of Strategic Planning in the Game of Go. PhD., Department of Psychology, The University of Michigan.

Lehner, P. E. (1983) Strategic planning in Go. In M. A. Bramer, editor, *Computer Game-Playing: Theory and Practice*, pages 167-176, Ellis Horwood, Chichester, West Sussex.

McCarthy, J. (1990) Chess as the drosophila of AI. In T. A. Marsland and J. Schaeffer, editors, *Computers, Chess, and Cognition*, pages 227-237. Springer-Verlag, New York.

Müller, M. (1995) Computer Go as a sum of local games: An application of combinatorial game theory. PhD thesis, Swiss Federal Institute of Technology Zürich.

Müller, M. (1997) Generalized thermography: A new approach to evaluation in Computer Go.

In H. Iida, editor, Proceedings of Using Games as an Experimental Testbed for AI Research (IJCAI workshop), pages 41-49.

Nigro, J. and Cazenave, T. (1996) Constraint-based explanations in games. Information Processing and Management of Uncertainty in Knowledge-Based Systems, Grenade 1996. Proceedings IPMU'96, pages 203-208.

Pell, B. (1991) Exploratory learning in the game of Go. In D. N. L. Levy and D. F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2-The Second Computer Olympiad*, volume 2. Ellis Horwood.

Power, J. (1982) *Invincible: the Games of Shusaku*. Kiseido Publishing Company, Tokyo.

Reber A. S. (1967) Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 5, pages 855-863.

Reber A. S. (1976) Implicit learning of synthetic languages: The role of instructional set. *Journal of Experimental Psychology: Human Learning and Memory*, 2, pages 88-94.

Reitman, J. (1976) Deducing memory structures from inter-response times. *Cognitive Psychology*, 3, pages 336-356.

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 29th National Conference of the ACM*, pages 123-127, San Diego. ACM.

Reitman, W. and Wilcox, B. (1975) Perception and representation of spatial relations in a program for playing Go. In *Proceedings of the 30th National Conference of the ACM*, pages 37-41. ACM.

Reitman, W., Nado, R. and Wilcox, B. (1978) Machine perception: What makes it so hard for computers to see? In C. W. Savage, editor, *Perception and Cognition*. University of Minnesota Press, Minneapolis, Minnesota.

Reitman, W. and Wilcox, B. (1978) Pattern recognition and pattern-directed inference in a program for playing Go. In D. Waterman and F. Hayes-Roth, editors, *Pattern Directed Inference Systems*, pages 503-523. Academic Press, New York.

Reitman, W. and Wilcox, B. (1979a) Modeling tactical analysis and problem solving in Go. In *Proceedings of the Tenth Annual Pittsburg Conference on Modeling and Simulation*, pages 2133-2148, Pittsburg 1979. Instrument Society of America.

Reitman, W. and Wilcox, B. (1979b) The structure and performance of the INTERIM.2 Go program. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 711-719.

Robson, J. M. (1983) The complexity of Go. In R. E. A. Mason, editor, *Proceedings of the IFIP 9th World Computer Congress*, pages 413-417. Elsevier Science Publishers, North-Holland.

Ryder, J. (1971) Heuristic analysis of large trees as generated in the game of Go. PhD thesis, Department of Computer Science, Standford University.

Saito, Y. and Yoshikawa, A. (1995) Do Go players think in words?-interim report of the analysis of Go player's protocol. In H. Matsubara, editor, *Proceedings of the Second Game Programming Workshop in Japan '95*, pages 118-127, Kanagawa. Computer Shogi Association.

Saito, Y. and Yoshikawa, A. (1996) An analysis of strong Go-players' protocols. In H. Matsubara, editor, *Proceedings of the Third Game Programming Workshop in Japan '96*, pages 66-75 Kanagawa. Computer Shogi Association.

Saito, Y. and Yoshikawa, A. (1997) Go as a testbed for cognitive science studies. In H. Iida, editor, Proceedings of Using Games as an Experimental Testbed for AI Research (IJCAI workshop), pages 65-73.

Schraudolph, N. N., Dayan, P. and Sejnowski, T. J. (1994) Temporal difference learning of position evaluation in the game of Go. In J. D. Cowan, G. Tesauro and J. Alspector, editors, *Advances in Neural Information Processing 6*, pages 817-824. Morgan Kaufmann, San Franciso.

Simon, H. A. and Chase, W. G. (1973) Skill in chess. *American Scientist*, 61, pages 394-403.

Stoutamire, D.(1991) Machine learning, game play and Go. Technical Report TR 91-128, Case Western Reserve University, Cleveland, Ohio.

Vincente, K. J. and de Groot, A. (1990) The memory recall paradigm: Straightening out the historical record. *American Psychologist*, 45, pages 285-287.

Wilcox, B. (1985) Reflections on building two Go programs. ACM SIGART Newsletter, pages 29-43.

Wilcox, B. (1988) Computer Go. In D. N. L. Levy, editor, *Computer Games, volume 2*, pages 94- 135. Springer-Verlag, New York.

Willmott, S. (1997) Adversarial planning and the game of Go. Master's thesis, Department of Artificial Intelligence University of Edinburgh, September 1997.

Willmott, S., Richardson, J., Bundy, A. and Levine, J. (1998) An adversarial planning approach to Go. In H. J. van den Herik and H. Iida, editors, *Computers and Games*, pages 93-112. LNCS 1558, Springer, Berlin.

Yoshikawa, A. and Saito, Y. (1997a) Hybrid pattern knowledge-Go players' knowledge representation for solving Tsume-Go problems. In *Proceedings of the 1st International Conference on Cognitive Science in Korea*, pages 134-139.

Yoshikawa, A. and Saito, Y. (1997b) The difference of the knowledge for solving Tsume-Go problem according to the skill. In H. Matsubara, editor, *Proceedings of the Fourth Game Programming Workshop in Japan '97,* pages 87-95, Kanagawa. Computer Shogi Association.

Zobrist, A. (1970) Feature extractions and representation for pattern recognition and the game of Go. PhD thesis, Graduate School of the University of Wisconsin.

Appendix A

Common Go Terms and Their English Equivalents

The following selection of Go terms and their English equivalents is provided as background for the convenience of readers unfamiliar with Go. It is based on a file publicly available on the Internet at http://nngs.cosmic.org/hmkw/stuff/definitions.html (downloaded August 2000).

Credits: Revised November 8, 1985, by Fletch Holmquist; extended November 1991, by Bill Taylor.

Aji (taste)*:* Latent threats or possibilities existing in a situation.

Atari: An immediate threat to capture; a single liberty remains. A verbal warning is often issued when placing an opponent into atari.

Dame (useless): A neutral point, territory for neither; a liberty.

Dan: Advanced grade.

Fuseki: The opening moves of the game where influence and territory outlines are formed. (literally: ‘no stones’)

Gote: Defensive play, loss of initiative. (Literally: ‘lower hand’)

Hane: A diagonal move played in contact with an enemy stone.

Hoshi: (‘star point’), 4-4 point.

Ikken-tobi: One point extension.

Joseki (established stones): Known sequences of moves near the corner which result in near equal positions for white and black.

Keima: Knight’s move extension.

Ko: Repetitive capture. (literally: ‘eternity’)

Ko threat: Intervening move (that one hopes will force a reply) before a ko can be recaptured.

Komi: Score adjustment usually penalizing black for playing first. Often 5.5 points.

Kosumi: A diagonal play next to one’s own stone.

Kyu: Learner grade.

Moyo: Large potential territory.

Nobi (Stretch): An extension away from an opponent’s tsuke, cross-cut, etc.

Ogeima (large knight’s move): Three across and one vertically (or vice versa).

Seki: A situation where neither player may place the other in ate without placing himself in atari.

Semeai: Race to capture.

Sente: Threat forcing direct response, creates initiative. The right to choose where to play next.

Opposite to gote. (Literally: ‘upper hand’.)

Shicho: Ladder play.

Shicho-atari: Ladder breaker. A stone played in the path of a potential shicho, threatening to make it fail.

Tobi: Jump.

Appendix B

The Predecessors to The Many Faces of Go: G2 and Cosmos

The Many Faces of Go (MFG) evolved from two other significant programs: G2 and Cosmos. A fuller appreciation of the long-term evolution of a Go program may be gained in the context of descriptions of G2, Cosmos, and MFG. The material in this appendix has largely been derived from the literature (Fotland, 1986; Stoutamire, 1991) but also includes some personal communication with Fotland.

B.1 G2

Fotland's first Go program determined territory by radiating influence. Unfortunately, it could be beaten by a simple liberty filling algorithm so he discarded it and wrote G2. The internal structure of G2 was described by Fotland in terms of its data structures, its tactical analyser, and its move selector (Fotland, 1986). G2 used full board reading and evaluated all legal moves with an emphasis on strategic moves. After the full board evaluation of all moves, the move that received the highest score was played.

B.1.1 The Data Structures

Most of G2's code was devoted to updating the data structures. Stones occupied points (referred to as squares) and were organized into strings (referred to as groups) and then into groups (referred to as armies). Data was associated with each point on the board and included the distance to the nearest and second nearest edge, a list of adjacent and diagonal neighbours, string number of the stone occupying the point, the number and a list of liberties, colour of stones on neighbouring points (white, black, or mixed), an influence function, and the nearest occupied point in each direction.

Horizontally or vertically adjacent points containing like-coloured stones were organised into strings. String data included colour, number and list of liberties, a list of its connections, a list of neighbouring enemy strings, the army it was part of, and an aliveness value (described below).

The types of connections recognized by G2 included one and two point jumps, diagonal connection and knight’s move. Connection data included the strings it joined, its type, the points involved, and its status.

Groups were comprised of strings that were connected by unbreakable connections or were both adjacent to a dead enemy string. Group data included a list of constituent strings, dead enemy strings used as connections, the number of stones and liberties, the number of eyes, and an aliveness value.

B.1.2 The Tactical Analyser

The tactical analyser was used to evaluate board positions so that a move could be selected. Each group was assigned an aliveness value which was also inherited by its constituent strings. The strength of G2 was primarily determined by the tactical analyser and consumed the largest portion of Fotland’s effort (Fotland, 1986). Since most of the program’s time was spent in the tactical analyser, it had to be both fast and efficient in its pruning of search trees.

Aliveness values resulted from a multi-stage process: strings that could be captured were identified as dead, eyes were found by the eye evaluator, unbreakable connections were identified, connected strings were identified as groups, territory was determined by the territory evaluator, and the life and death evaluator assigned an aliveness value to each group which was then assigned to its constituent strings.

The tactical analyser examined every string with less than five liberties to determine whether it was dead, that is, whether it could be captured; strings that could obtain at least five liberties were assumed to avoid capture. Interesting moves were suggested during lookahead to determine the status of a string. Interesting attacking moves included geta moves for groups with two liberties, liberty filling moves, moves adjacent to an enemy string’s liberties, and liberty gaining moves for friendly strings. Interesting defensive moves included diagonal moves into unoccupied territory, one point jumps, extensions, connections, and liberty filling moves (for adjacent enemy strings).

The interesting moves were heuristically evaluated and the best were selected for further analysis. The best of these moves was hypothetically played and the resulting situation analysed from the opponents point of view. Once again, the best moves were chosen and the best one was hypothetically played. Normal backtracking was used at each stage as well as alpha-beta pruning. Thus, a depth-first tree was generated until the string being examined was captured or gained sufficient liberties to live. The search tree was generated to a maximum of 80 moves which still enabled ladders to be accurately read. The number of “best moves” considered (between 1 and 3) depended on the depth of the current move in the search tree: more moves were considered near the root and less near the leaves. Since the value of the terminal node had only two values (i.e., captured or escaped), alpha-beta pruning was very efficient.

An eye evaluator was used to maintain information about eyes, half eyes (eyes made in gote) and quarter eyes (eyes needing two moves to be made) by evaluating small enclosed eyespace. G2 recognised some basic dead shapes and various types of eyes including three in a row, four in a square, bulky five, pyramid, and cross.

Groups were determined by finding strings that were connected by unbreakable connections. The tactical analyser tried to cut connections to test their strength. Dead opponent strings could also be used as connections between live friendly strings.

Territory was determined by the territory evaluator. As well as completely enclosed territory, G2 also evaluated territory near the board edge that did not need to be completely enclosed. Territory could be made near the edge by being between a stone and the edge or a connection and the edge. However, such territory could be challenged by a stone within four lines along the edge, and closer to the edge (undercutting the edge).

The life and death evaluator analysed every group. An aliveness value was assigned to each group by considering the number of liberties it had, its eyespace, the amount of territory it controlled, whether it could extend along the edge, its size, whether it was in contact with an enemy string, and whether it was completely surrounded. The aliveness value rating itself had 20 different values including “has two eyes”, “has enough space to make two eyes”, "unsettled", "not enough eyespace", and "can't possibly make two eyes".

B.1.3 The Move Selector

A rule-based strategic evaluator with 65 rules was used to select moves by choosing the highest scoring move using a two step process. In the first step, strategic moves were evaluated and in the second step, all legal moves were evaluated. Information from the data structures and the tactical analyser was used to evaluate board positions resulting from moves in both the first and second steps. Fotland felt that this was the weakest component of G2 due to the weakness of the data structures (Fotland, 1986).

The rules suggested strategic moves to evaluate such as playing in an empty corner, make shimari or kakari, a move from the joseki dictionary, extending along an edge, invasions, contact fights, blocking, cutting, connecting, or blocking an invasion on a wall.

In the first step, all the suggested strategic moves were evaluated. A particular strategic move was hypothetically played and the resulting board position was evaluated. The evaluation process consisted of assigning values to the board points and then summing them together to return an overall score. The value for a given board point was determined by the aliveness value of the string that controlled it. If the board position resulting from a strategic move did not satisfy the strategic goals associated with the rule that suggested it, the evaluation of the move was ignored.

In the second step, all legal moves (including the strategic moves considered in the first step) were evaluated. The value assigned to a board position resulting from hypothetically playing a move was based on the aliveness, size, and amount of territory controlled by each string on the board.

The values that resulted for each move from step two were summed with the value they obtained if they were considered in step one. The move with the overall highest score was then chosen as G2's next move.

B.1.4 Size, Performance and Timeline

Between 1981 and 1988, Fotland spent around four years part-time writing G2. He stopped work on G2 for several years because the available computers were not fast enough for the development of strong Go programs. G2 was around 11,000 lines of C code and used around 700k for the code and data. During that time, Fotland's Go skill advanced from 15 kyu to 1 dan. G2 played at about the 25 - 20 kyu level and could beat beginners without much trouble (Fotland, 1986). Fotland had some success at Computer Go competitions with G2 coming 4th in the world Computer Go competition in Taiwan and 1st in the US Computer Go championship in 1987.

B.2 Cosmos

Fotland converted G2 into Cosmos in 1988. The main difference between Cosmos and G2 was that instead of evaluating all legal moves, only suggested moves were evaluated. This transition was achieved by increasing the number of move suggestors to over 250. Quiescent search was added and the connection, eye, and territory evaluations were made more accurate.

B.2.1 Data Structures

The data structures used in Cosmos were basically the same as those used in G2. The eye information included the number of eyes achievable in gote, the number of eyes achievable in sente, the number of eyes achievable if the opponent moved twice, a list of vital points, and eye type. The information contained in the data structures was updated either incrementally (e.g., lists of liberties) or after every move (e.g., influence).

B.2.2 Tactician

Essentially, Cosmos had the same tactician as G2 but better move generation and sorting. In Cosmos, the tactician was only used to determine dead and threatened strings by trying to capture them. Its parameters were maximum liberty count, maximum depth, and maximum search size. The maximum liberty count was 4, and therefore as for G2, strings with more than 4 liberties were assumed to avoid capture. The maximum depth allowed was 100. The search size determined the playing level (Stoutamire, 1991) and since forcing moves were not counted, ladders could be accurately read even at low playing levels.

B.2.3 Influence Function

Territory was determined by radiating positive influence from alive groups and negative influence from dead groups. Radiated influence did not pass through stones or connections and was inversely proportional to distance.

B.2.4 Board Evaluation

Life and death of groups was the primary concern dealt with by the evaluation process. Board positions were evaluated by a similar process to that used in G2 with the addition of quiescent search.

The tactician was used to determine dead and threatened strings (those that could be captured if they moved second) and whether the stones at the diagonal of eyes could be captured. This information was useful in identifying unbreakable connections and forming strings into groups.

Eyes were analysed to determine their potential and then allocated to groups. False eyes were identified by checking the diagonals of eyes, and some dead shapes were also known by Cosmos. Any group with enough eyespace to make two eyes was considered to be alive. Potential eye space could be gained by extensions along the edge, possible connections, being adjacent to a threatened enemy group, and by controlling territory that was not already considered to be an eye.

There were 25 values that described a group’s strength in terms of its life and death status. The values were divided into five main categories which included very alive, alive, unsettled, weak and dead. Determining the strength of a group included considering potential connections, potential eyes, and potential extensions along the edge. Positive influence was radiated from alive groups and negative influence was radiated from dead groups with black and white influence being maintained separately.

Weak groups that were contacted by the influence from friendly alive groups were considered not to be surrounded. Weak groups were further divided into those that could run and fight and those that would almost certainly die.

In general, board points were scored according to the radiated influence values. However, occupied board points, board points adjacent to a stone and board points between a stone and an edge were scored differently. Unsettled groups were scored according to who was to play next.

B.2.5 Move Selection

Cosmos had over 250 rules that suggested moves which included fuseki moves (including shimari, kakari and joseki moves), edge moves, playing in the centre, playing safe when ahead of opponent, “squirming” around when behind opponent, pattern matching, saving weak friendly groups (including making eyes, running, or fighting semeais), killing weak enemy groups, cutting, connecting, contact fights, ko threats, and filling dame. Cosmos had a joseki library that contained around 5,000 suggested joseki moves and a pattern database that had 60 patterns that were 5x5 in size. Whenever a match was made, the rule (code) associated with the pattern was applied (executed) and would then suggest a move. The addition of extra moves to be suggested could easily be accomplished by the addition of new move suggestor rules or the addition of patterns.

The rules would supply a guess value (probable evaluation of the move), bonus value, minimum aliveness value, and would indicate which groups were being attacked or defended if any. The moves were sorted by their guess values and, depending on playing level, a certain number were “played”. Moves could be rejected before being evaluated if they did not “apply”. A move applied if it accomplished what it was intended to do: if a move was intended to defend a group and the group ended up weaker than it started the move was rejected. Surviving moves were awarded a sente bonus if they achieved sente. The position resulting from the move was then evaluated. The evaluation score, the sente bonus and the rule bonus were summed together and the highest scoring move was selected as the next move.

B.2.6 Size, Performance and Timeline

It took Fotland nine months to compress G2 down to 512 k and add a new user interface. In 1988 Cosmos was released by Ishi Press for IBM-PC as Cosmos, The Computer Go Partner and was released a second time in 1989. Due to the amount of time spent on the user interface, Fotland was unable to devote sufficient time to improving Cosmos’ playing level and its performance in various world Computer Go championships was modest coming 8th in 1988 and 7th in 1989.

B.3 The Many Faces of Go

After a rewrite of the user interface and addition of professional graphics and new features, Cosmos was released as The Many Faces of Go in 1990. Additions included a limited full board lookahead capability, consideration of the value of sente, and a strategy function which is used to focus attention on the important areas of the board and to identify urgent moves. The number of patterns in both the joseki and pattern databases were increased. The pattern database was also improved by increasing the pattern size from 5x5 to 8x8, by storing many suggested moves for each pattern in a move tree (rather than just one per pattern as for Cosmos), and by a complete rewrite which encompassed algorithms, code, and data structures and lead to an increase in speed.

Appendix C

The Experimental Software

The experimental studies conducted in Section III used purpose built software which is described in this Appendix.

Studies 1 and 2

The experiments and case studies were conducted using software initially written by my colleague Andrew Hussey and then later modified by me. The experimental software was implemented by Andrew as a case study to investigate the scalability of the Presentation Abstraction Control architecture as part of his Doctoral[1] studies. The software was written in itcl3.0 using Tcl4.0/Tk7.6 and contained approximately 2,500 lines of code. Andrew spent approximately four weeks developing and testing the software. After my initial pilot studies and Andrew's case study were completed, I assumed charge of the software and spent approximately four weeks modifying and testing it in preparation for the experimental studies reported in this thesis. During my stay at NTT in Japan, Takehiko Ohno ported the software to the NTT system.

When the software was originally conceived, the possibility of sub-second latencies was not considered. During the case studies of the master Go players in Japan, it became apparent that it would be possible to accurately control the latencies to 500 milliseconds but no shorter.

Study 3

The experiment conducted in Study 3 used software that I wrote in Java using JDK1.1. It contained approximately 2,000 lines of code and took approximately 6 weeks to develop and test.

1. Hussey, A. (1999) Object-oriented specification and design of user-interfaces. PhD thesis, Department of Computer Science and Electrical Engineering, The University of Queensland, Australia.