(1)

Using Knowledge about the Opponent in Game-Tree Search

Peter J. Jansen

September 1992
CMU-CS-92-192

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at Carnegie Mellon University.

This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. Government.

Keywords: Artificial Intelligence, Cognitive Science, Game Theory, Game-Tree Search, Structure of Games, Speculative Play

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Using Knowledge About the Opponent in Game-Tree Search

PETER JANSEN

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

Herbert A Simon — September 21, 1992
THESIS COMMITTEE CHAIR — DATE

[signature] — 9/28/92
DEPARTMENT HEAD — DATE

APPROVED:

R. Rej — Sept 28, 1992
DEAN — DATE

Abstract

Ever since Shannon proposed his view on programming a computer to play chess, computer chess researchers have respected his basic assumptions. This thesis relaxes some of these assumptions, in particular the tenet that the two players should be assumed identical or similar.

During at least part of any interesting game, one of the players (the 'agent') attempts to achieve a result larger than the game-theoretic value (e.g. tries to convert a drawn position into a win, or to salvage a loss). This is only possible if the opponent makes a game-theoretic mistake the agent can exploit. Hence the agent will try, based on knowledge of the opponent, to bring about a situation in which the likelihood that the latter commits an error is increased ('speculative play').

An error by the opponent may be due to (1) factors pertaining to the domain, (2) factors pertaining to the opponent, and (3) factors pertaining to the interaction of the opponent with the domain. This thesis investigates factors contributing to error both in the structure of game domains and in properties of the opponent's search strategy.

The first part of the thesis considers properties of *random games*, as defined by their game graphs. The execution of the *retrograde analysis* algorithm for the solution of games imposes an implicit *structure* on the game. From this structure the thesis derives several interesting properties of random games, including difficulty-related characteristics. These properties are investigated both in an empirical analysis of actual random games and through a mathematical analysis of an approximate statistical model of random games. Several properties exhibited by both analyses are shown to be relevant, in fact explanatory, of properties of actual games.

In the second part of the thesis, some properties defined in part I are discussed for the chess endgame of King and Queen vs. King and Rook (KQKR). Human play in KQKR is analyzed, and common heuristics evaluated. This information is used to construct a 'heuristic program' for database play, which is shown to induce an increased average error in human subjects in comparison to a standard database program. This result demonstrates feasibility and utility of speculative strategies in actual game playing situations.

Acknowledgements

Even a solitary player of the thesis game cannot avoid incurring a staggering debt to a large network of co-players in his research environment.

I would like to thank my thesis committee, who provided excellent feedback on a difficult draft in a very short time. First and foremost, I am indebted to my advisor, Herb Simon, for his seemingly infinite patience, understanding and interest. That I made it, and that I now perhaps understand a little more what science is all about, is mainly his doing. Jonathan Schaeffer's stimulating enthusiasm and dynamism spurred me on when I seemed to be stuck in perpetual check. Though I interacted with Jaime Carbonell only for a relatively short time, his careful revisions and sharp questions and insights made a big difference. I am grateful to Danny Sleator, who challenged and complemented some of the theoretical arguments presented in this thesis, and whose *playful* ways made the last few months much more enjoyable than they would have been otherwise.

Apart from the members of my thesis committee, many friends and colleagues in the SCS environment at CMU contributed in small or not so small ways to this thesis. I would especially like to thank Fernand Gobet, who spent hours participating in my experiments and carefully scrutinizing my drafts, and often provided me with a sounding board for my ideas. Many other people helped me out in many ways, and I cannot do justice to them all. I feel especially indebted to Hans Berliner, Jerry Burch, Murray Campbell, Bert Enderton, Gordon Goetsch, Kathie Porsche, Indira Subramanian, and Raul Valdes-Perez.

My thanks also go out to the other members of the Deep Thought team: Murray Campbell, Thomas Anantharaman, Feng-Hsiung Hsu, Andreas Nowatzyk and Mike Browne. Being a part of the DT experience was one of the highlights of my years at CMU.

I would also like to thank the people whose job in SCS it is to be there, but who manage to do it with this unexpected extra edge of enthusiasm. My special thanks to Sharon Burks, without whom everything would not have been quite the same.

Then there are all those who helped making life tolerable, and occasionally much more than that. Here I am thinking especially of Martha and Nico Habermann and Irene and Dana Scott, Peter and Betty, Marko and Eva, Veronik, Bev and Gene, Pierre, Fang and XiaoGang, Sabine and Consuelo. To them, and all the others I had to omit here, my sincere thanks.

Finally, all of this would not have been possible without the understanding of my parents and other family members, who kept up their wholehearted support even when it seemed to them that the endgame was taking me a little bit too long...

Contents

List of Tables

List of Figures

Prelude

Chapter 1

Introduction

This thesis investigates some aspects of how the availability of knowledge about a fallible opponent may affect game-tree search and the choice of move.

Errors on the part of an opponent may be more or less likely depending on parameters pertaining to the domain. The structure of a game determines how many correct and how many incorrect moves are available to a player, in a given position (local) or as an average over the domain (global). The first part of this thesis investigates the structure of random games with respect to some difficulty-related parameters.

Errors may also result from erroneous knowledge (e.g. in the form of heuristics) or from limitations on the searching capabilities of a player. If some of these limitations are known, a model can be constructed, which can then be used in move decisions. Depending on how such a model is used, it is possible that the new strategy proposes moves that would not be optimal against an identical opponent. Can such a strategy outperform an optimal strategy? The second part of this thesis investigates this possibility in the actual game domain of King and Queen vs. King and Rook (KRKN), a subgame of the game of chess. The strategies investigated estimate the likelihood of error by the opponent on the basis of common heuristics, as well as certain parameters of the KQKR domain.

In section 1.1, we see how computer chess research has closely followed the directions set out by Shannon in 1950.

Section 1.2 discusses what happens if some of Shannon's assumptions are relaxed, to be more consistent with human game playing.

Section 1.3 reviews some related work.

Section 1.4 takes on one of the main questions of this thesis, but for a highly abstracted model of game playing: under which conditions could it be beneficial to make non-optimal moves? This question can be answered mathematically – in the simple abstract model – and leads to a set of rules which have some general validity, notably in the KQKR endgame discussed in Part II.

This introductory chapter conludes with a statement of the goals of this thesis and a thesis overview.

1.1 AI, Search and Games.

Game-Tree Search, a field of study generally classified under Search, a subfield of Artificial Intelligence, is concerned with methods of finding the best possible decisions in situations occurring in game playing, usually based on look-ahead. In research, as in real life, games serve as simplified models of more complicated social and economic interactions. In this sense, the study of games may teach us something about the basic principles of typical human behavior.

The game that has received the most attention throughout the history of Artificial Intelligence is the game of chess, the "Drosophyla of AI"[39]. This game is an excellent paradigm: its rules can be stated simply and concisely, it is deterministic, results can be measured precisely in wins and losses, chess is popular with many, and it is generally viewed as an activity requiring superior intelligence for competitive performance. In other words, the game of chess presents an ideal challenge to computer programmers who want to prove that they can capture certain aspects of intelligent behavior in their programs.

In a rather strong sense, the overall strategy of computer chess research has been laid out by Claude Shannon's seminal work[62]. Shannon defined the concept of *evaluation function*, the principles of the *Minimax* backup procedure, the idea of building an approximate evaluation function based on *features* of the position such as material, mobility etc., and even the concept of an endgame database ("dictionary"). In addition, his "type B strategy" foreshadows quiescence search and selective search such as singular extension search[4] ("forceful variations").

During recent years, researchers have extended the limits of several of Shannon's design decisions, bringing chess computers up to the level of the top 500 human players in the process. The main components of this progress include (1) the discovery of the alpha-beta algorithm[25], and (2) rapid improvements in technology (speed and memory capacity of computers; special purpose hardware) which enabled programs to search ever deeper[5, 10, 21]. In addition, several researchers investigated and evaluated features of positions, and looked for better ways of combining them into evaluation functions[9]. Because the interactions of the evaluation function parameters with the search are complex, these parameters were (are) sometimes learned automatically, based on available data on human play[5].

Other researchers studied theoretical phenomena in uniform trees, in which certain properties of a game were posited as parameters[45]. The limits and expected performance of various algorithms were analyzed in great detail, based again on forward search with artificially imposed parameters. Nau and others discovered the phenomenon of *search pathology* (i.e. a situation where looking deeper leads to less accurate decisions), as a consequence of accumulating errors[46, 60, 51].

Still others constructed *endgame databases* for small games, or subgames of larger games[7, 26, 14, 15, 59, 17]. Such an endgame database, once constructed, needs only to be loaded in memory to enable a computer to play optimally. Databases were also used to study problems in knowledge extraction from data, for example how one can represent the database in the smallest number of human-understandable rules [41].

Writers of game programs implemented many 'tricks' related to tournament play, such as time control strategies, and various *ad hoc* ways to improve performance.

Recently, many new algorithms were proposed as alternatives to alpha-beta search. Some researchers looked at the influence of the fallibility of the players[53, 42]. Others criticized the statistical incorrectness of the minimax rule and are investigating new algorithms based on information-theory[8]. Many stepped away from a basic uniform-depth approach, and proposed possibly more meaningful non-uniform search surfaces (singular extensions[4], conspiracy numbers[38], equi-potential search[6] etc.).

Most of Shannon's assumptions, either explicitly stated or tacitly assumed, have been accepted uncritically ever since by researchers in the field. The next section takes a critique of some of these assumptions as a starting point for the research described in this thesis.

1.2 Game-Tree Search vs. Game Playing

This section identifies three of Shannon's implicit assumptions and considers the implications of relaxing them. A good argument can be made that these assumptions are inconsistent with actual human game playing. Depending on the relative state of knowledge of both players, a game can be divided in several phases. Finally, we can classify game playing situations depending on the knowledge the agent has about his opponent and the game domain.

1.2.1 Relaxing Shannon's assumptions

As a starting point for this thesis, we could take a critical look at the following three assumptions, made implicitly by Shannon in his paper.

1. The game-playing situation is symmetric (*symmetry assumption*).
2. The opponent has the same evaluation function and search strategy (*identity assumption*).
3. A game is a sequence of moves, in each of which the goal is to find the best possible move (*optimality assumption*).

Let us first consider the identity assumption. As long as chess programs performed at a level far below the average human player, it was a good objective to try to imitate human play in a computer. But this situation no longer holds, and if computers don't consistently beat the best human players yet, they do already perform much better in certain classes of positions. Though imitating (modeling) human play is still interesting for Cognitive Psychology, it is no longer useful as a way to achieve improved move selection. In addition, imitation leads to play towards positions with which the human is familiar, and has therefore a relative advantage. Finally, even if it were possible to spell out the human 'algorithm', it is unlikely that this procedure would be a good algorithm for a totally different architecture.

In an asymmetric situation, one player (the *agent*) has information (about the value of positions or about the opponent) not available to or computable by the opponent, and can use this information in his move decision (i.e. *speculative play*).

The optimality assumption implies that the agent searches, at each move, for an *'optimal move'*, i.e. a move that leads, within the agent's search horizon, to the largest possible value against best play[1]. But given that information is available about the opponent, this strategy may not lead to the best possible results (expressed in the proportion of games won). What is best against a player who thinks alike, may not be good against a totally different opponent[2].

---

[1]However, note that 'value' here really means the predicted value if the opponent were identical to the agent. In most cases there is no guarantee that this value is any closer to the game-theoretic value than the value computed for speculative play.

[2]For instance, a human playing a computer may prefer to keep the position closed and non-tactical, which would not give best results against another human player.

1.2.2 Game Playing vs. Problem Solving

The computer chess research community has viewed a game of chess[3] basically as a *series of problems* to be solved consecutively, in each of which the goal is to find the 'best move' in some sense. Although this is a good first-order approximation of human game playing, there are several important differences.

Some of these differences, namely those that are related to the Shannon assumptions under discussion, are given below as propositions. They follow from the premise that *people try to obtain the highest possible score*, i.e. they want to win more games than other contenders.

Proposition 1: *In any real game between human players, at least one of the players is unable to compute an optimal strategy during part of the game .*

If both players could compute an optimal strategy from the beginning, the result would be predictable and the game uninteresting. For example, the game of Tic Tac Toe remains interesting, as a game, only until such time as both players know how to avoid a loss consistently.

Proposition 2: *In any real game, at least one of the players attempts to obtain a better result than the game-theoretic value.*

If both players are satisfied with the (estimated) game-theoretic value, there is no need to play the game. For example, during a chess game at least one player is trying to convert a drawn position into a win, or to save a lost position.

Proposition 3: *A strategy for increasing the game-theoretic value necessarily incorporates a model of the opponent.*

The game-theoretic value is the value obtainable by best possible play. It is not possible to obtain a better result, unless the opponent makes a move that reduces his game-theoretic value, i.e. unless the opponent makes a *mistake*. To win against some opponent from a drawn position, or to draw or win from a lost position, the agent must bring about a position in which the opponent makes a mistake the agent can exploit[4]. Implicitly or explicitly, this always involves some model of the limitations of both the self and the opponent, since the agent must evaluate the opponent's probability of error. For example, when an opponent has little time available, it is a common strategy to create 'complications'. The purpose of this strategy is to increase the likelihood of error. The assumption behind it is that the opponent is more likely to make an incorrect decision in 'complicated' positions than in 'simple' positions when he has little time at his disposal to analyze the position. This assumption is only valid for a certain class of opponents.

---

[3]Note that, in this and the following subsection "game" refers to a specific series of successive positions, leading from a starting position to some final position. In most of the remainder of this thesis, however, the word "game" will refer to any collection of positions on which transition rules and terminating conditions are defined (see section 2.1). The intended meaning should follow naturally from the context.

[4]The agent need not be 'conscious' of this fact. Human chess players often refer to a position as 'better' because the opponent has (or is perceived to have) more chances to go wrong. In other words, the possibility of making mistakes may form an integral part of the evaluation function.

1.2.3 Phases of game playing

In the beginning of a game, at least one player (the opponent) is incapable of working out a winning strategy reliably. During this phase, one player (the agent) tries to induce his opponent to make a mistake he can exploit. Any strategy to accomplish this must be able to *predict* which move(s) the opponent is likely to make, or at least be able to evaluate in which positions the opponent is likely to make a mistake, i.e. which positions are *difficult*. This phase can be called the *speculative phase* of a game.

At some (later) point during the game, a position will be reached in which one player (say the agent) is now able to calculate a win (or forced draw). This phase can be called the *technical phase* of a game. In this phase, only the structure of the game is important to the agent. No model of the opponent is needed to achieve the win. Play may continue, because this situation could still be unclear or speculative to the opponent.

Finally, at some stage, both players are capable of working out the solution *and* are convinced that the other player can do so too. There is no more point in playing on, and at this point the practical game usually ends, by resignation of one of the players, or by a mutual agreement to a tie.

1.2.4 Classifying game-playing situations

An analysis of strategies for speculative play could be divided according to the quality of knowledge the agent has about the game and the opponent. The following is one possible taxonomy.

1. Game and opponent are completely known (e.g. endgame database; opponent's evaluation function known).

2. The game is completely known. The opponent is only partly known (e.g. endgame database; human opponent).

3. The opponent is completely known but the game is not (e.g. the full game of chess; opponent's evaluation function known).

4. Neither game nor opponent are completely known (e.g. the full game of chess; human opponent). A further distinction can be made depending on the estimated strength of the opponent.

In this thesis, we focus mainly on the second of these situations, namely when the domain is known completely (as when the endgame database is available) and the opponent needs to be estimated or predicted. The condition of interest is when the agent (computer program) has a losing position, and can use speculative play to obtain a draw.

1.3 Related work

Whereas much effort has been spent in the directions of Shannon's framework, little research is available on the issue of fallibility of an opponent, and on including such information in search or in the move decision process.

- To whomever is familiar with actual human game playing it should come as no surprise that the idea of speculative play (in a theoretical setting) is fairly old (consider, for instance, Emmanuel Lasker's views on chess around the turn of the century). Interesting is a discussion of the issue in Luce and Raiffa's standard work on game theory [36] (p. 77).

- Many apparent paradoxes arose in computer chess, simply because in the programming no allowance was made for glaring weaknesses on the part of an opponent. For instance, several cases are known when a computer chess program gives up material for no apparent reason, only to avoid a mate which could not possibly have been found by the program's opponent. This phenomenon led several programmers to include 'hacks' in their programs which would result in more 'human-like' play on the part of their programs, or would lead to positions more likely to be favorable. For example, certain asymmetries were introduced in top-level programs (such as Deep Thought) which would reduce its tendency to trade material (and thus simplify the position) and would increase the probability of obtaining more complicated positions. Finally, programmers *often gear the opening book of their programs towards difficult and trappy lines.*

- Some studies were made of human play in database domains (see, for example, Kopec for KRKN[27]). The main idea, however, was not to exploit weaknesses, but to learn how to formulate concise rules for play in the endgame, allowing more compact programming in computers. For more of this kind of knowledge extraction, see work by Michie and others[41].

- Some attempts were made to include the opponent in the search, and to explain evaluative comments which have no game-theoretic counterparts. Some researchers propose to model the opponent as a random player[34, 53]; others propose a search based on an involved system of values and utilities[42].

- This author investigated dynamic features of a standard search algorithm to asses the difficulty of a chess position. These dynamic features can also be used to classify human errors into several types of mistakes[23].

- Korf investigated some issues in search with two evaluation functions. He showed that no cutoffs comparable to alpha-beta are possible. However, he looks only at a symmetric situation with perfect knowledge on the part of both players, whereas in this thesis it is essential that the situation is asymmetric, and at least one player does not have complete knowledge.

- Issues of difficulty have been a topic of investigation in the domain of cognitive psychology[32, 33]. The domain of the cited references is puzzle solving, however, and focusses mainly on issues of representation.

- Several researchers appear to feel uncomfortable with the 'optimality' assumption, and speculate that an 'optimal' move may not always be the best move against a given opponent [51, 12].

1.4 When is making mistakes beneficial?

The following chapters will analyze game domains to determine parameters relevant to error behavior, and investigate the use of predictions of human behavior in endgame database play. Before setting out on this task, let us first convince ourselves, on the basis of a simplified model, that it is indeed possible that a non-optimal strategy in the game-theoretic sense would outperform an optimal strategy under certain conditions.

The simplified model of game playing considered is as follows. One player is assumed to have perfect knowledge of the game and its game-theoretically optimal strategies, as well as some knowledge about the opponent (enough to predict, with some degree of reliability, whether he would make a mistake in a given position). The opponent is fallible, and his performance can be characterized by a few simple statistical parameters.

We can now compare two different strategies for the first player: an 'optimal' strategy, limited to optimal moves in the game-theoretic sense, and a 'speculative' strategy, which bases its decisions on the moves it predicts for the opponent. A comparison of the expected value of these two strategies can be expressed as a *decision surface* in the parameter space, separating the regions in which each of the strategies is better.

An analysis of this decision surface leads to some rules about which strategy should be preferred, and some of these rules can be extrapolated – with some caution – to more realistic situations.

First, let us define this model, and the relevant parameters which will form the dimensions of the parameter space.

1.4.1 A simple model of game playing

Consider two players, **MAX** (the agent) and **MIN** (the opponent), playing a 2-player zero-sum game with a branching factor $b = 2$. Every terminal position (leaf node) i in the tree has associated with it a value V_i. Player **MAX** knows this value (explicitly or implicitly). Player **MIN** doesn't know the V_i, but can compute an estimate based on some features of the position (not further specified here).

In each position the player to move has two choices: the correct move or a 'mistake'. Let the (expected) "cost" of a mistake be a constant value δ. The value of the position resulting from the mistake is δ less (for **MAX**) or more (for **MIN**) than the value of the position after the optimal move. As we shall see, it is important that δ is different for the two players, and we shall need to make a distinction between δ_{MAX} and δ_{MIN}. Further assume that successive full moves[5] are independent of each other, so we only need consider a 2-ply decision tree. This tree, with the values after one full move, is depicted in Figure 1.1. In the figure, transitions corresponding to mistakes are labeled with question marks, and transitions corresponding to correct moves with exclamation marks. In the root node **R**, it is **MAX**'s move, and he has a choice between nodes **A** and **B**, in both of which it is **MIN**'s

[5] *Full move* = one move by each player.

move. The second choice would, at least game-theoretically, be a mistake, as **B** has a lower value than **A**, in the amount of δ_{MAX}. **MIN** has a similar choice in both of these positions, eventually leading to any of the positions **C**, **D**, **E**, or **F**. In this case, mistakes lead to a higher value for **MAX**.

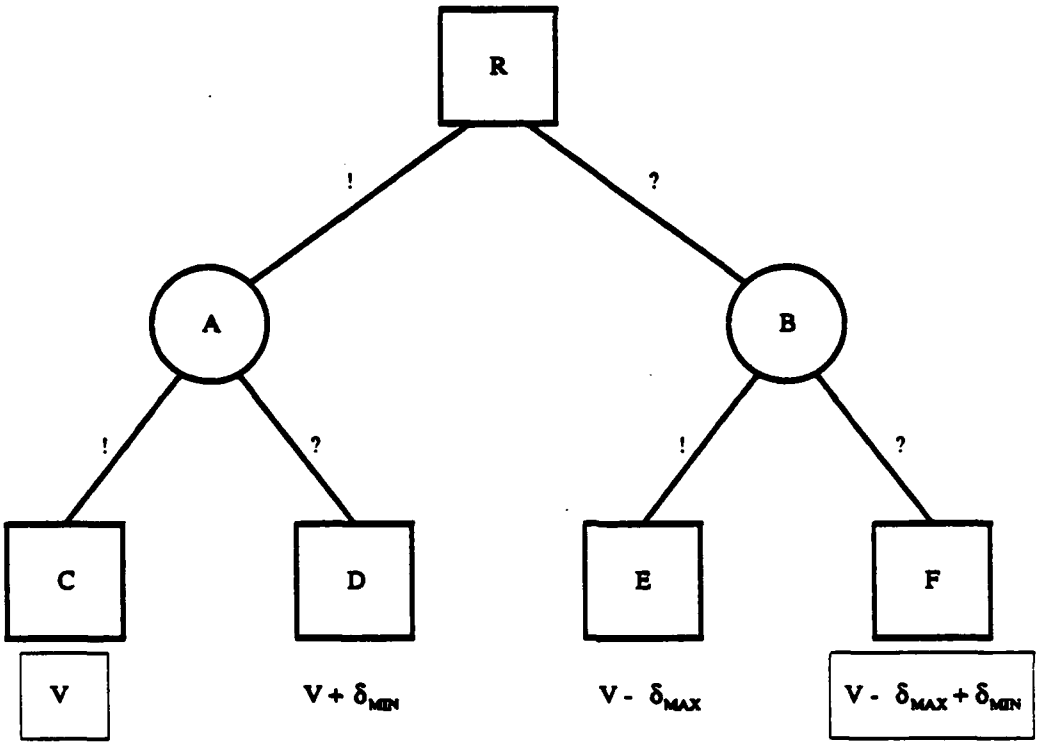

Figure 1.1: Simple game-playing model: two-ply decision tree. Mistakes are indicated with a '?', and good moves with a '!'.

As player **MAX** is omniscient, and knows the exact game-theoretic values of all nodes, he in principle can always avoid mistakes. Player **MIN** is fallible, and makes mistakes with an a priori probability p. In addition, **MAX** has some knowledge about **MIN** (the exact form of which we shall ignore here – he may, for example, be aware of some of the features **MIN** uses while searching). Based on this information, **MAX** can make a *prediction* for **MIN**'s move at each node, and use this prediction in his own move selection. The relevant statistical parameters here are the *prediction errors* ε_1 and ε_0, indicating the probability **MAX** predicts a mistake when **MIN** actually makes the correct move, and **MAX** predicts a correct move when **MIN** actually makes a mistake, respectively. Let the value 0 stand for a mistake by **MIN** and 1 for a correct move. Let $P(x|y)$ denote the probability that **MAX** predicts that **MIN** will play x in a situation where **MIN** will actually play y. We can then write the prediction errors as

$$\varepsilon_0 = P\{1|0\}$$

$$\varepsilon_1 = P\{0|1\}$$

For the sake of simplicity, we will in a first analysis assume that both errors are of equal magnitude, or $\varepsilon_1 = \varepsilon_0 = \varepsilon$

Finally, since the only way in which the error magnitudes influence the result is in terms of their ratio, we define the *relative error magnitude* $\Delta = \delta_{\text{MIN}}/\delta_{\text{MAX}}$.

1.4.2 Two basic strategies for MAX.

Now let us consider two basic strategies for player **MAX**. In strategy 1, an *optimal strategy*, **MAX** always picks the objectively best move, independent of his predictions of **MIN**'s actions. In strategy 2, a *speculative strategy*, **MAX** picks the move with the *highest value, given his predictions*, in spite of the fact that this could lead to a position of lower value, namely in those cases when the predictions call for **MAX** to make a mistake.

Just from inspecting Figure 1.1, we can draw one conclusion right away. Suppose that $\delta_{\text{MAX}} \geq \delta_{\text{MIN}}$. In this case, both of the nodes in the left part of the tree (under **A**) have values which are larger than or equal to both of the nodes in the right part (under **B**). Therefore it is never to **MAX**'s advantage to move to **A**, i.e. to make a mistake. We can put this as a first Rule:

> **Rule 1**
> A speculative strategy for **MAX** cannot be better than an optimal strategy *unless* $\delta_{\text{MAX}} < \delta_{\text{MIN}}$, i.e. $\Delta > 1$.

This rather obvious rule states that it is not useful to try speculative strategies unless the error magnitude can be kept smaller than that of the opponent, on the average. But this is not a sufficient condition, so let us expand this analysis further, and compute the *decision surface* between the two strategies, i.e. the set of points in the parameter space where both strategies are expected to perform equally well.

1.4.3 Computing the decision surface.

A complete analysis would consider all possible combinations of **MIN**'s actual strategies with **MIN**'s strategies *as predicted by* **MAX** (16 cases). However, both strategies will recommend the same decision (the optimal move) except when the prediction is that **MIN** will make a mistake from position **B** and the correct move from position **A**. We can therefore limit ourselves to this case.

If the predictions turn out to be correct, the optimal strategy obtains a value of V, whereas the speculative strategy obtains $V - \delta_{\text{MAX}} + \delta_{\text{MIN}}$ (both highlighted by boxes in Figure 1.1). As, according to rule 1, the second value is larger than the first, speculative play is better than optimal play under conditions of *perfect prediction*. However, if the prediction is wrong, strategy 1 will get a higher value than expected, and/or strategy 2 a lower one, cancelling out some or all of the expected benefit.

Let us denote the four possible strategies for **MIN** as ordered pairs (m_0, m_1), where $m_0 = 0$ if **MIN** responds to mistake by **MAX** with a mistake of his own, and $m_0 = 1$ if he replies to a mistake with a correct move, and similarly for m_1. From the previous paragraph, we know that the only case we need consider for each of these four strategies is the case where **MAX** predicts the $(1,0)$ strategy for **MIN**.

Table 1.1 gives the probabilities that MIN adopts each strategy and MAX makes the appropriate predictions, as well as the expected *gain* for MAX's two strategies. The gains G_1 and G_2 are defined as the amounts by which the values actually achieved by the respective strategies exceed the 'optimal' value V. For example, in case 2, MIN's strategy consists of making an error after a correct move, and a correct move after an error. This gives strategy 1 an unexpected bonus of δ_{MIN}, and penalizes strategy 2 in the additional amount of δ_{MAX}. The probability of this situation occurring is dependent on the *a priori* probability that MIN makes a mistake (in this case, MIN makes a mistake in A, with probability p, and the correct move in B, with probability $(1-p)$). It also depends on the prediction errors (in this case, both predictions are wrong, which occurs with probability $\varepsilon_0\varepsilon_1$). From this follows the table entry for the probability. Case 3 is the only situation which is to the advantage of

Case	MIN's strategy	G_1	G_2	Probability
1	(0,0)	δ_{MIN}	$\delta_{\text{MIN}} - \delta_{\text{MAX}}$	$p^2(1-\varepsilon_0)\varepsilon_0$
2	(0,1)	δ_{MIN}	$-\delta_{\text{MAX}}$	$p(1-p)\varepsilon_0\varepsilon_1$
3	(1,0)	0	$\delta_{\text{MIN}} - \delta_{\text{MAX}}$	$p(1-p)(1-\varepsilon_0)(1-\varepsilon_1)$
4	(1,1)	0	$-\delta_{\text{MAX}}$	$(1-p)^2(1-\varepsilon_1)\varepsilon_1$

Table 1.1: Marginal probability and gain for each of MIN's strategies.

a speculative strategy, but it occurs most frequently when the prediction errors are small. Case 2 is a worst case scenario, but this occurs only when both predictions were wrong. In both other cases, strategy 2 simply loses the total amount of the error, δ_{MAX}.

From the table we can calculate the expected value for both of MAX's strategies. The decision surface between the two strategies is given by the condition that these two values are equal. If we put together the four parts of the table, equate the expected values, and group terms, we get the following equation.

$$\delta_{\text{MAX}}\left((1-\varepsilon_1)\varepsilon_1(1-p)^2 + (1-\varepsilon_0)(1-\varepsilon_1)(1-p)p + \varepsilon_0\varepsilon_1(1-p)p + (1-\varepsilon_0)\varepsilon_0 p^2\right) = \delta_{\text{MIN}}(1-\varepsilon_0-\varepsilon_1)(1-p)p$$

Working out, simplifying and letting $\varepsilon_1 = \varepsilon_0 = \varepsilon$ yields for the decision surface

$$\Delta = \frac{\delta_{\text{MIN}}}{\delta_{\text{MAX}}} = \frac{(p - 2\varepsilon p + \varepsilon)(1 - \varepsilon - p + 2\varepsilon p)}{(1-2\varepsilon)(1-p)p} \tag{1.1}$$

This equation expresses the required relative magnitude of error Δ in terms of p and ε.

We can now state a more precise rule than Rule 1:

Rule 2
Strategy 2 is better than strategy 1 if, and only if,

$$\Delta = \frac{\delta_{\text{MIN}}}{\delta_{\text{MAX}}} > \frac{(p - 2\varepsilon p + \varepsilon)(1 - \varepsilon - p + 2\varepsilon p)}{(1 - 2\varepsilon)(1 - p)p}$$

It is easy to see that there are three situations when the required Δ approaches infinity, which, as MIN's mistakes are assumed to be finite, corresponds to $\delta_{\text{MAX}} = 0$. We can express these degenerate cases by the following rule.

Rule 3
A speculative strategy is *never useful* when

1. $\varepsilon = \frac{1}{2}$: the prediction gives no information.
2. $p = 0$: MIN never makes mistakes.
3. $p = 1$: MIN never makes the correct move.

It is also possible to show that the right hand side of Rule 2 is always ≥ 1 when the parameters are in their allowable range ($0 \leq p \leq 1$ and $0 \leq \varepsilon \leq \frac{1}{2}$), in agreement with Rule 1.

The decision surface of equation (1.1) is plotted in Figure 1.2. The X-axis is the prediction error ε, the Y-axis the opponent's error probability p, and the Z-axis the relative error magnitude Δ. Values were cut off at $\Delta = 15$. The error probability p varies between about 0.05 and 0.95. We see that the decision surface is at $\Delta = 1$ when the prediction error $\varepsilon = 0$. The decision surface curves upward towards ∞ when the error probabilitiy p approaches 0 or 1, or when the prediction error ε approaches $\frac{1}{2}$. We can also see that the surface slopes up more gradually for intermediate values of p.

Based on the decision surface, several special cases can be considered. When the error probability, p, is kept constant, one can compute the required relative error magnitude, Δ, from the prediction error, ε. For example, when $p = \frac{1}{2}$ we get

$$\Delta = \frac{1}{1 - 2\varepsilon}.$$

This function is plotted in Figure 1.3. It represents a vertical cross section of the decision surface parallel to the X-axis.

Another possibility is to keep Δ constant. For example, one might know the opponent's average error magnitude δ_{MIN}, and wish to determine under which conditions of prediction error, ε, and error probability, p, speculative play is indicated. The decision criterion between the two strategies is then given by the the *horizontal contours* of the decision surface, parametrized by the value of Δ. Some of these contours are plotted in Figure 1.4. In this figure, the X-axis points to increasing error in prediction, ε, the Y-axis in the direction of increasing error by the opponent, p. Strategy 2 is better than strategy 1, or speculative play

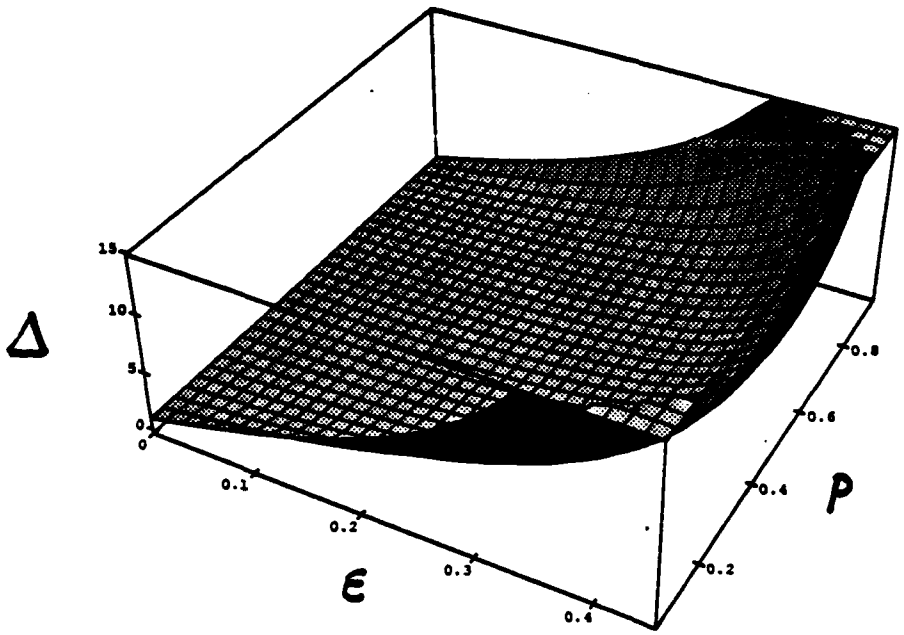

Figure 1.2: Decision surface between Strategies 1 and 2.

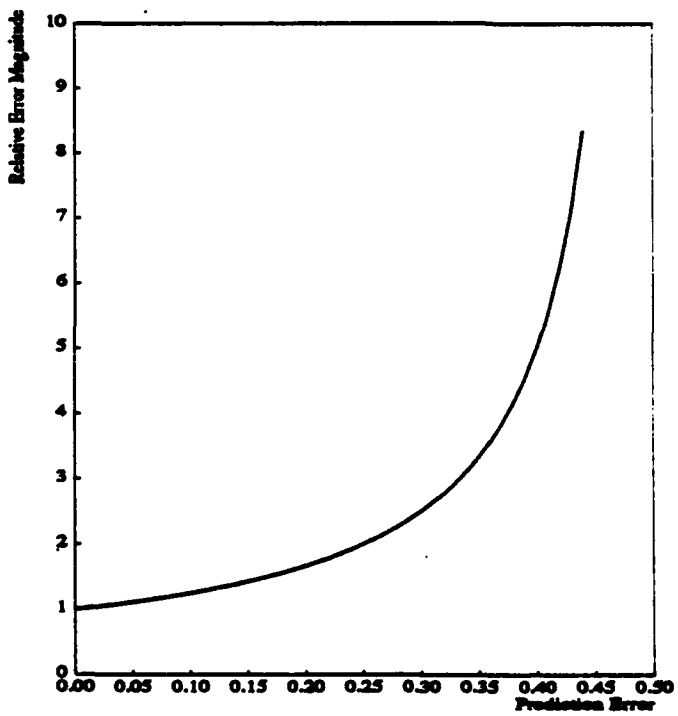

Figure 1.3: Relation between rel. error magnitude and prediction error for $p = 0.5$.

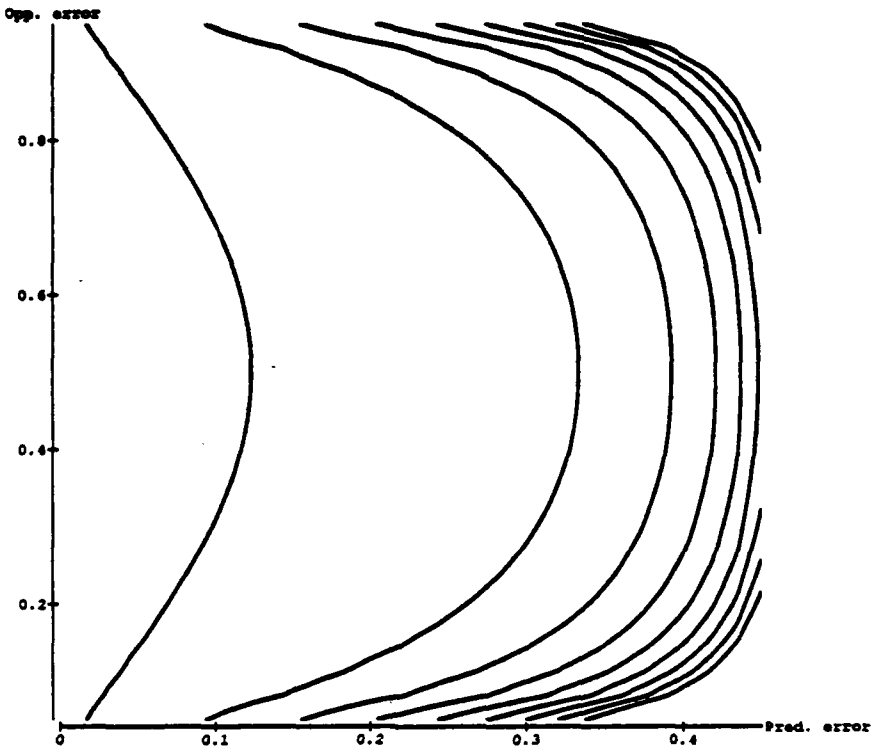

Figure 1.4: Contours of Figure 1.2 indicate decision criteria between strategies.

leads to better results, if the parameter point is to the *left* of the appropriate contour.

Figure 1.4 allows us to rephrase Rule 2, in the case of constant relative magnitude of error:

Rule 4
For a given relative magnitude of error, strategy 2 is more likely to be better than strategy 1 when

1. The opponent is of intermediate strength, (his error probability p is neither very small or very large), and
2. MAX's prediction of MIN's moves is more accurate.

1.4.4 Application: KQKR database play

The model under discussion corresponds very well to the situation studied in section 7 (treating the King and Queen vs. King and Rook endgame), where MAX is the player with the Rook trying to draw the game by stalling MIN's win. The value V is the number of moves to win with optimal play (i.e. it is equal to the depth $D()$), and the error magnitudes δ_{MAX} and δ_{MIN} corresponds to the number of moves lost (on average) in case of a mistake.

Of course, not all the features of the actual KQKR play are represented by the simplified

model. For instance, the error probability and magnitude are in fact dependent on the distance to win, D (see section 6.1), and the branching factor is much larger than 2. To bring this situation in line with the model discussed here, we could lump together all the optimal moves, as well as all the 'mistakes', but then neither the error probability nor the error magnitude can be expected to remain constant over the domain.

Nevertheless, we can still try to interpret some of the observations made in the KQKR eperiments based on the model. From the analysis of 24 Master/Grandmaster games described in section 6.1.3 we find that, on the average, people make one mistake every three moves[6], i.e. $p = 33\%$, and when they do, the expected magnitude is $\delta_{\text{MIN}} = 2.24$. In the experiments conducted in the course of this thesis[7], subjects ranging from A-class player to International Master made errors in 30% to 60% of the cases, with magnitude between about 1.5 and 4.0.

As far as the prediction error ε is concerned, a rough estimate is that it ranges from about 15% to 40%. As the humans' error magnitude δ_{MIN} is not large, speculative play can only be successful if the program's error magnitude, δ_{MAX} is also small. This observation may explain why the initial experiments (where the heuristic program was allowed to drop up to 3 points) did not perform better than random choice among optimal moves. In other words, to make speculative play work in the KQKR setting,

1. δ_{MAX} should not be much larger than 1, and
2. it should not be used against very strong or very weak players, or against players who are not well-known (not predictable enough).

Another feature of KQKR which seems to be ignored by the model is the possibility of *ties*, i.e. when there is a choice between two moves that are both optimal. But in fact, this is nothing other than the situation where the error magnitude $\delta_{\text{MAX}} = 0$. As we saw, speculative play *always* works in this case (if, at least, the prediction error is smaller than 0.5). Similarly, we can consider ties for the opponent ($\delta_{\text{MIN}} = 0$). In this case, speculative play *never* works. We can phrase the first case as another rule.

Rule 5
Strategy 2 is always better than strategy 1 in the case of ties for MAX, provided $\varepsilon < 0.5$

(Compare the discussion for Rule 3).

1.4.5 Discussion

Limitations of the model

The model considered in this section leaves out or keeps implicit certain factors relevant to practical game playing. One is that the *search* is made completely implicit beyond the

[6]For more detailed information, see section 6.1.3.

[7]See section 7.

second ply. In practical situations some search may be required both to assess the objective values of all nodes, as well as to predict the opponent's behavior.

Another is that *limitations on resources* (time or memory) are completely ignored. The principal player is supposed to be able to make a complete and correct assessment of the situation instantaneously. In practice there will be a trade-off between the resources used and the various error measures. For instance, the heuristic programs used in the experiments with the KQKR endgame use a certain amount of searching to predict the (human) opponent's behavior, and the prediction error ϵ decreases with the amount of search. There is also the trade-off between the amount of time spent on assessing the objective values and the time spent on prediction. If predictive accuracy is improved at the expense of evaluation correctness, it is quite possible that the benefits gained will be completely nullified.

Also implicit is the *structure of the domain*, local or global. Peculiarities of the domain may influence both the predictive accuracy ϵ and the probability p that MIN will make a mistake.

Furthermore, it is probably not completely realistic to assume *independence* between p and δ_{MIN}. Other things being equal, a player who makes mistakes more often will probably also make mistakes that are worse. This may change the precise shape of the contours of Figure 1.4 somewhat, but will not alter the qualitative conclusions.

Another dependency is between the *playing strength* of the opponent and his *predictability*. If a 'constant' strategy is to be used, it may be good to gear it towards the high range of the playing strength.

Finally, the assumption that the *size of* MIN*'s mistake*, δ_{MIN}, is the same in both branches of the tree may not be realistic. This does not seem to present any conceptual difficulties, however.

Relevance of the model

In spite of these simplifications, there is apparently a good agreement between the model and the results from experiments with the KQKR endgame. For instance, speculative play is *not beneficial against weaker players* (in the context of KQKR this means chess players below expert strength), as they are bound to make mistakes most of the time. It is also *not useful against the strongest players* (in KQKR this means grandmasters and masters who are sufficiently prepared), as they make very few and small mistakes, and can only benefit from mistakes by the program. Hence, as discussed with rule 4, speculative play is the most useful against players of 'intermediate' strength (ranging from Expert to IM). As even the strongest players still make a significant number of mistakes, increasing the predictive accuracy might allow some improvement against world class players as well[8].

[8] A factor not considered here with potential practical significance is that some mistakes by the program may lead to positions that are not normally encountered (either when playing other humans or in optimal lines extracted from an endgame database). This unfamiliarity may also contribute to a larger number of mistakes.

An important consequence of rule 5, as already stated in the previous subsection, is that *speculative strategies are always beneficial to resolve ties.* Although this opportunity may not arise frequently, it is absolutely risk free, and could lead to a sizeable benefit, even when this should occur only few times per game.

Summary of rules

The results of this section, as presented in the various 'rules' is summarized in table 1.2.

Rule 1:	*only if*	$\Delta > 1$	(basic condition)
Rule 2:	*if, and only if*	$\Delta >$ [Eq. 1.1]	(decision surface)
Rule 3:	*only if not*	$\varepsilon = \frac{1}{2}$ or $p = 0$ or $p = 1$	(exceptions)
Rule 4:	*probably only if*	$0 << p << 1$ and/or $\varepsilon << \frac{1}{2}$	(most likely)
Rule 5:	*always if*	$\delta_{MAX} = 0$ and $\varepsilon < \frac{1}{2}$	(ties)

Table 1.2: When does a speculative strategy outperform an optimal strategy?

1.5 Goals of the thesis

The issues discussed in this chapter open up a whole new field in Game-Tree Search, and one thesis cannot hope to explore all of them in depth.

This thesis will focus on two points.

1. Show that certain properties observed in actual games can be derived from a random-game model. Develop insight into the structure of games, in particular concerning properties related to the difficulty of games.

2. Exhibit an existence proof, in an actual game domain and with actual human subjects, of a non-optimal strategy that performs at least as well as an optimal strategy.

1.6 Thesis Overview

This thesis examines two issues in game-tree search that are related to the presence of a fallible opponent.

Part I is concerned with issues related to the game-playing domain. Chapter 2 defines the structure of a game in terms of its Optimal-Solution Graph, and elaborates on the construction of this graph by means of the retrograde-analysis algorithm. Chapter 3 discusses Random Games, and describes a series of experiments designed to discover the relation between the basic parameters of a game (size, branching factor and the number of terminal positions) and the game's emergent properties. Chapter 4 proposes a mathematical model of an 'average' random game, and proves, within certain assumptions, certain interesting properties of random games. These are then compared with the result of the previous chapter, and – qualitatively – with properties of actual games.

Part II is focussed on issues related to characteristics of the opponent as problem solver. Chapter 5 describes the structure of the chess endgame of King and Queen against King and Rook. A connection with the previous part is made, in particular regarding factors governing how difficult the endgame is for human players. Chapter 6 proposes several criteria of quality of play and uses these criteria to analyze human games. Chapter 7 analyzes the quality of common human heuristics with respect to the depth of a KQKR position. Chapter 8 describes an experiment in which the opponent is modeled by a predictor function built up out of these heuristics. A comparison is made between the performance of a strategy using this model and a standard strategy limited to optimal play.

Both of these parts are components in the study of errors in problem solving behavior in game playing situations, and in how a problem solver interacts with the domain, and are as such on the same level. The different parts of this thesis can also be viewed as logical components of the main claim of this thesis, namely that there exist actual situations in which using knowledge of the opponent in the search is appropriate and benficial. Figure 1.5 summarizes the thesis as a dependency tree in support of this argument.

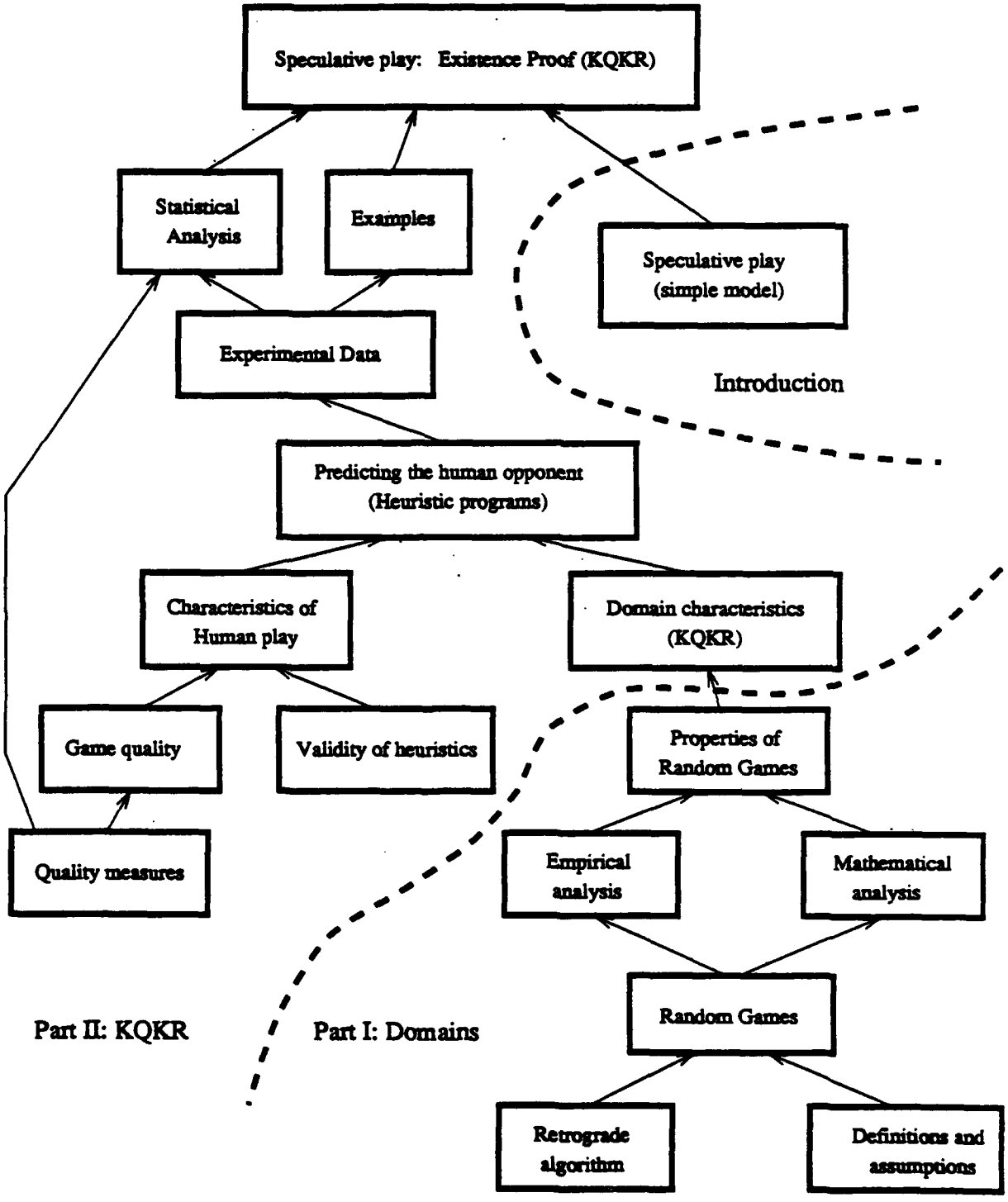

Figure 1.5: Logical overview of the thesis

Part I

Game Domains

How well a player (problem solver) will perform in a certain game domain depends on factors that pertain to the player, his opponent and the domain, as well as the interaction between these three components. This part of the thesis explores the *structure* of a game domain, as engendered by the rules of the game.

In chapter 2, games are represented by their Game Graphs. For two-player games in which players alternate moves, the Game Graph is a directed bipartite graph. The common retrograde analysis algorithm for the solution of games can be phrased in Problem Space terminology, and the resulting structure examined in terms of the problem solving structure.

Chapter 3 defines random games, and discusses an experiment to discover empirically the relation between the basic parameters of the game, such as the size of the game space, the branching factor and the number of terminal positions, and the game's expected emergent properties, such as the number of winning positions in the domain for a player, the expected maximum number of moves required to win, as well as the expected depth distribution.

Chapter 4 tries to get insight into the same emergent properties for random games in general, by means of a mathematical analysis of a simplified *'game equation'* — actually a set of differential equations governing the depth distribution. This chapter discusses several characteristics of games that follow from an analysis of this equation, and compares these characteristics with observations for randomly generated games as well as small chess endgames.

Chapter 2

The structure of two-player games: Optimal-Solution Graphs

How well a player (problem solver) will perform in a given game domain depends on factors that pertain to the player, his opponent and the domain, as well as the interaction between these three components. In this chapter we explore the statistical structure of a game domain, as it follows from the rules of the game. We pay specific attention to those parameters of the domain that are relevant to a player's (expected) performance.

A game is usually looked at in the direction it is played ("forward"). In most computer game models since Shannon, search starts with the current position and gradually expands nodes until some criterion of time or depth is reached. The positions considered in the search are positions *within a certain radius of the initial position.* This is the most natural way of looking at a game, as it corresponds to the usual situation of a game in progress, and in many cases, when the domain is large, it is also the only practical method. As a result, most game-playing algorithms are based on forward search ("data-driven"), with alpha-beta search as a typical example.

In a forward-search framework, however, it is hard to answer some important questions about any particular game or games in general. How many winning positions are there? Can one determine from the rules whether a game will be interesting? Are some positions more useful to remember than others? How can one determine which positions are winning? Also, although a tree-search model may be apt for problem solving and game playing, it can hardly be construed as an appropriate model for the games themselves[1].

In this chapter we shall look at games from the opposite point of view, namely *starting with the terminal winning positions of the game* ("goal-driven"). This *retrograde analysis* approach will allow us to make a better attempt at answering questions like the above, and come up with more reliable predictions about a game's structure.

The idea of solving games backwards is of course not new, and dates back at least 60

[1]For instance, the uniform-tree model of a game with branching factor b would imply that a fraction $\frac{b-1}{b}$ of the positions are terminal positions at the same distance from the initial position, an unrealistic assumption in most cases.

years. In the more recent past, retrograde analysis has been applied in chess and checkers in the construction of *endgame databases* for certain endgames in which only a few pieces remain (up to six pieces for chess [70], or up to 8 pieces for checkers [59]). However, it appears that no attempts have been made to learn more about the structure and properties of game domains based on this approach, and the research described in this chapter may be a first step in that direction.

In the context of this chapter, we consider games with the following characteristics, rather similar to those put forth by Berlekamp et al.[12] (p.16).

1. The game is a two-player zero-sum game with complete information (i.e. both players can uniquely identify the current position (state) of the domain).
2. players alternate moves
3. moves are deterministic (no chance element) and the set of legal moves is completely specified (by the rules of the game)
4. certain positions are defined to be terminal positions (by the rules of the game)

Unlike Berlekamp's model, the present model does not require a specific starting position, and allows repetition as a possibility (draw). The above characteristics allow us to represent the game by a *directed bipartite graph*, or *game graph*, with a single value at each node. Any other characteristics of the game (e.g. shape of the board, etc.) are abstracted away, and only implicitly represented in the game graph. One single game graph may therefore correspond to more than one game. All the games corresponding to one game graph can be said to have identical *structure*, as the positions of any two such games can be put into exact one-one correspondence [2].

Section 2.1 gives a formal definition of a game, explicitly or implicitly corresponding to the definition given above. Section 2.2 shows how 'winning positions' can be computed, and how they can collectively be viewed as a construct (optimal game graph) that determines the structure of the game.

[2]This need not mean that two such games will also have the same level of difficulty for a human player. Indeed, the representation of the domain may make mental manipulation harder or easier, as investigated, for example, by Kotovsky [32]. Example games with identical structure are Tic Tac Toe, Magic Fifteen and others[12].

2.1 What is a Game?

Let us first phrase the definition of a game in problem-space terminology [48]. In the following, imagine two players, called White and Black, who alternately 'have the move'. A game, then, consists of the following elements.

1. a *game space*, G, which is a set of uniquely identifiable states p, called *positions*. Note that the state also identifies the player to move.

2. a *successor function*, $S(p)$ which is the set of permissible successor positions of a position p (i.e. all the positions resulting from a legal move by the player to move). The successor function of a set of positions, A, is then the union of the successor sets of each of the positions in A: $S(A) = \bigcup_{p \in A} S(p)$. The *inverse successor function* $S^{-1}(p)$ gives the set of positions from which a position p can be reached in one move. As for $S()$ the inverse successor function of a set A, is the union of the successor sets of each of the positions in A: $S^{-1}(A) = \bigcup_{p \in A} S^{-1}(p)$. The inverse successor function of a set of positions, A, is the union of the inverse successor sets of each of the positions in A: $S^{-1}(A) = \bigcup_{p \in A} S^{-1}(p)$.

   To impose the assumption that the two players alternate moves, we require that the resulting directed graph be *bipartite*, or $G = G_W \cup G_B$ such that $S(G_W) \subseteq G_B$ and $S(G_B) \subseteq G_W$. G_W denotes the set of positions in which White has the move (WTM), and G_B the corresponding set for Black (BTM).

3. a *set of terminal white wins*, T_W. These positions are *immediate* wins for White, as defined by the rules of the game. Without loss of generality we may assume terminal positions to be positions with white to move, i.e. $T_W \subseteq G_W$ (for example, White can mate in one move). Indeed, suppose the rules of the game define a position $p_b \in G_B$ to be a terminal loss (e.g. Black to move is mated). We can then either omit this p_b from G_B and add the predecessor positions $S^{-1}(p_b)$ to T_W (mate in one), or, alternatively, extend the game with successor positions of p_b such that $S(p_b) \in T_W$ (s e.g. White can take the Black King).

   The size of this set will be denoted by $t_0 = |T_W|$.

4. a *set of terminal black wins*, T_B, corresponding to positions in which Black wins according to the rules of the game.

5. a *set of terminal draws*, $T_=$, corresponding to positions in which neither side can win according to the rules of the game (for example stalemate). For the sake of convenience, assume that this and the previous set are subsets of G_B.

In what follows, we look at the game from the point of view of one of the players, say White. We can assume that only *wins* matter, we shall lump together draws and losses (for

White) into a set of *non-wins*, of which $N_W \subseteq G_W$ and $N_B \subseteq G_B$[3]. This can be done without loss of generality, since the analysis of the situation from Black's point of view is completely symmetrical, and both views together are sufficient to compose the complete solution of the game.

Now what is a *winning position*? It is a position from which White to move can force the transition to a position in the set of terminal wins T_W in a bounded number of moves, no matter what his opponent does. Conversely, a *losing position* for Black is a position from which he cannot avoid being forced to a position in T_W in a bounded number of moves, i.e. from which all moves lead to winning positions for White.

We can now classify all positions into one of the following six mutually disjoint classes (with $\setminus$ denoting set difference).

1. The set of terminal wins T_W (WTM)
2. The set of terminal non-wins $T_B \cup T_=$ (BTM)
3. The set of non-terminal forced white wins $W_W \setminus T_W$ (WTM)
4. The set of forced black losses L_B (BTM)
5. The set of white non-wins N_W (WTM)
6. The set of non-terminal black non-losses[4] $N_B \setminus (T_B \cup T_=)$ (BTM)

These six sets are illustrated schematically in Figure 2.1. In the figure, possible transitions (moves) are indicated with arrows. As White does not have a forced win from N_W, all moves must lead to positions in N_B. Similarly, as Black cannot avoid a loss from L_B, all moves must lead to W_W. From any position in $W_W \setminus T_W$, White must have at least one move leading to L_B, as he must have a forced win. But he might also have moves leading to N_B (mistakes), indicated by a question mark '?'. Similarly, from N_B Black must have at least move that does not lead to a forced loss (leading to N_W), but he might have other moves to W_W as well. We shall come back to the subject of mistakes, a central concept in this thesis, shortly.

---

[3]In other words, N_W is the set of positions from which White to move cannot win, adn N_B is the set of positions from which Black to move can avoid losing.

[4]To avoid the cumbersome use of 'non-win' and 'non-loss', we shall call both of these 'draw', ignoring the possibility that the other side may win. We must still distinguish between N_W and N_B, since positions in these classes have a different player to move.

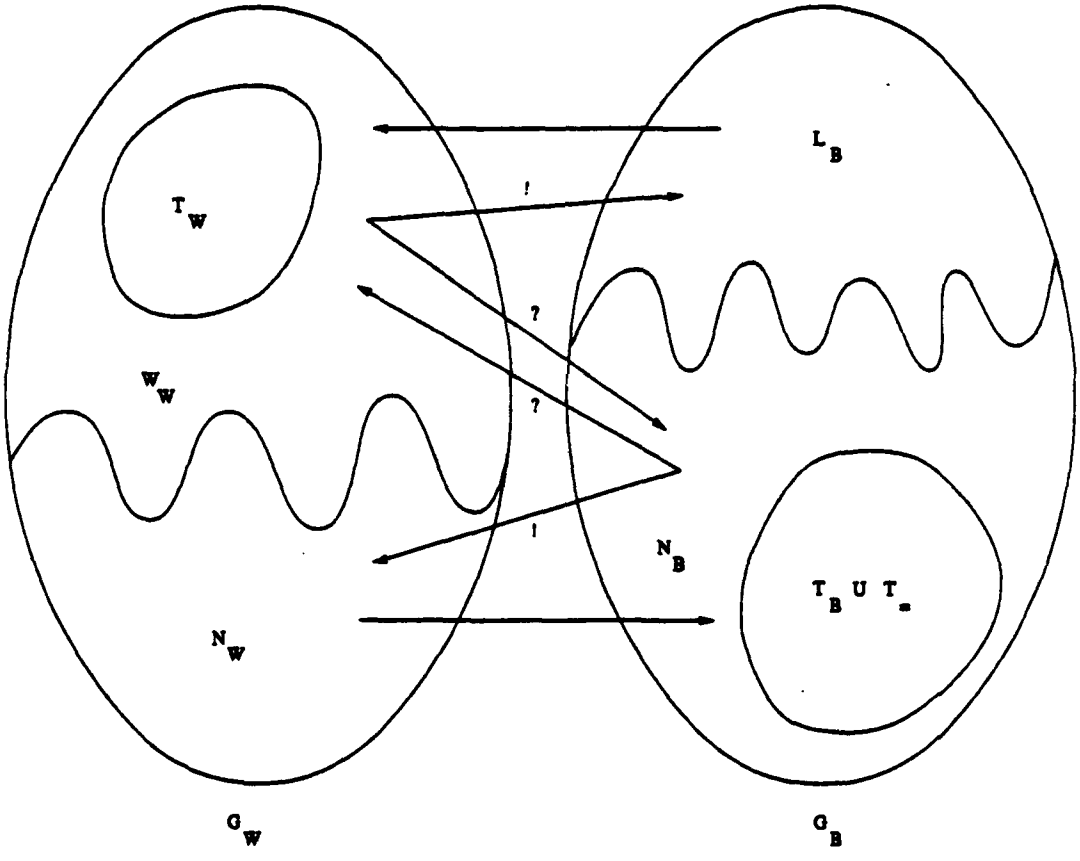

Figure 2.1: Schematic representation of the game graph. G_W is the set of positions with White to move, G_B the set of positions with Black to move. W_W is the set of winning positions with White to move, of which the set of terminal winning positions, T_W, is a subset. L_B is the set of losing positions with Black to move. N_W and N_B are the sets of positions from which a win cannot be forced. The set of terminal non-wins (composed of the set of positions that are terminal draws and positions that are terminal wins for Black) is a subset of N_B.

2.2 Computing the Optimal-Solution Graph

In this section, first an algorithm is given to solve a game (in principle). The intermediate stages of this algorithm suggest an 'algorithmic' representation of a game, corresponding to the stage of the algorithm positions are added to the sets of positions known to be wins (WTM) or losses (BTM). This algorithmic representation corresponds to the game's *structure*, which we could call its *optimal-solution graph*. Finally, we find that *mistakes* and *difficulty* can be meaningfully defined with respect to the optimal-solution graph.

2.2.1 Solving a game by retrograde analysis

The procedure to solve a game is implicitly specified by the recursive definition for winning positions given in the previous section. Let us first formalize this definition.

1. terminal wins are winning positions, i.e $T_W \subseteq W_W$
2. positions in G_B are losing positions if *all the successors* are winning positions, i.e. $S(p_B) \subseteq W_W \Rightarrow p_B \in L_B$
3. positions in G_W are winning positions if *at least one successor* is a losing position, i.e. $S(p_W) \cap L_B \neq \phi \Rightarrow p_W \in W_W$

This definition recursively maps functions of positions onto other functions of positions, and what we need is a *fixed point* of this mapping. To make it correspond exactly with the definition given in the previous section, we need to look at the *smallest fixed point*, containing all the positions which can be reduced to positions in T_W in a finite number of steps. What this means is that we exclude positions which would only be wins because one player can force the other player back into such a position after a finite number of moves (i.e. we exclude forced repetitions from W_W[5]).

Given the above, the following algorithm approximates the set W_W and L_B iteratively, and is guaranteed to terminate with sets corresponding to the smallest fixed point set for any finite game.

Call W_i the ith approximation of the set W_W, and L_i the ith approximation of L_B. The complement $\overline{A}$ of a set A is taken with respect to the appropriate subgraph (G_W or G_B) of which A is a subset. Then the basic algorithm is given in Fig. 2.2 in set notation.

At each stage positions are added as losses when all of their successors are elements of the updated set of wins. Positions are added as wins when at least one of their successors is a member of the updated set of losses. The procedure terminates when no more positions can be marked as losses, i.e. when both sets are unchanged during one full step of the procedure

[5] Actually, the rules of a game could prevent repetition altogether, or make it losing. In principle it is always possible to distinguish all states based on how many times they have occurred before, and labeling the third (nth) occurrence as a terminal state. For the purpose of this chapter we call (forced) repetition a draw, as no player can force a transition to terminal states.

1. $i \leftarrow 0$
2. $W_0 \leftarrow T_W$; $L_0 \leftarrow \phi$;
3. **repeat**
   (a) $i \leftarrow i + 1$
   (b) $L_i \leftarrow \overline{S^{-1}(\overline{W_{i-1}})}$
   (c) $W_i \leftarrow S^{-1}(L_i)$

   until $L_i = L_{i-1}$ or $W_i = W_{i-1}$

Figure 2.2: Conceptual algorithm for retrograde analysis

(i.e. when a fixed point has been obtained). Notice how steps 2, 3(b) and 3(c) correspond with items 1, 2 and 3 of the definition given earlier.

Various implementations of this algorithm have been used to construct endgame databases for chess (and other games). Because of the fact that all the positions in the domain need to be considered and stored, the applicability of the algorithm is limited to small domains, such as endgames with few pieces (at most 6 in chess and 8 in checkers with the current computational capacity). A discussion of detailed programming techniques is outside the scope of this thesis. Several publications on the construction and interpretation of such databases have appeared in the computer chess literature[75, 15].

2.2.2 Optimal-Solution Graphs and the "distance to win" parameter

The above algorithm, by means of the iteration parameter i, implicitly defines the distance to win $D(p)$ of every position $p \in W_W$ (also called 'optimal depth' or shortly 'depth' in what follows).

Observation

Let δ_{Wi} be the subset of W_W added at stage i:

$$\delta_{Wi} = W_i \setminus W_{i-1}$$

and δ_{Li} the subset of L_B added at stage i:

$$\delta_{Li} = L_i \setminus L_{i-1}$$

then a position $p_W \in W_W$ which requires $D(p_W) = i$ full moves to win is an element of δ_{Wi}, and a position $p_B \in L_B$ which allows a defense of at most $D(p) = i$ full moves is an element of δ_{Li}.

This result is easily proven by a straightforward induction argument.

To understand this better, consider the momentary state of the sets G_W and G_B as they vary over time when the algorithm is executed. Fig 2.3 shows this 'algorithmic' representation of a game[6]. Notice that, because of the stage i(b) in the algorithm, positions newly added to L_B (positions in δ_{Li}) must have at least one successor in the positions newly added to W_W (those in $\delta_{W(i-1)}$), as they would otherwise already have been added during a previous iteration. Also, no successor can be in $G_W \setminus W_i$, because then the position would not be added at the present stage. In intuitive terms: Black has at least one move that forces White to still make $i-1$ moves, but none that force him to play longer. An almost symmetrical reasoning holds for White: because of step i(c), positions newly added to W_W (positions in δ_{Wi}) must have at least one successor in the positions newly added to L_B (those in δ_{Li}), as they could otherwise not be proven at this stage to lead to a win. Also, no successor can be in L_i, because in that case the position would have been added during a previous stage. Intuitively, if White has a win in i moves, he can place Black in a position from which he must move to a position which can be won in at most $i-1$ moves.

We see, therefore, that optimal play for both players proceeds along the path indicated by fat arrows in the figure, and moves from a position in δ_{Wi} to a position in δ_{Li} to a position in δ_{Wi-1} etc. Hence, $D(p)$ decreases by 1 after each full move, as expected.

In chess, the 'depth' $D(p)$ could be the number of moves necessary to achieve mate (the actual end of the game). With suitable redefinition of terminal positions, the depth could also denote the number of moves it takes to win heavy material, or to obtain a position in which the outcome is known to be a win (i.e. an "effectively' terminal position).

The depth will play a large role in the following sections. One reason is that it provides an *operational evaluation function*, as opposed to the actual *value* of the game, which can take on only two values (or three, if we differentiate between draws and losses), and therefore doesn't allow discrimination between longer and shorter wins[7]. $D(p)$ is also an important component of the *difficulty* of a position, as in general a searcher will need to spend time

[6]The δs are indicated by Ds in the figure.

[7]In fact, it is necessary to add this notion to practical game playing programs, to avoid that they would cease to make progress when they come within visibility range of a certain win ('mesa effect'). A spectacular instance of this effect occured in a game COKO III - GENIE, at the 1971 ACM computer chess tournament.

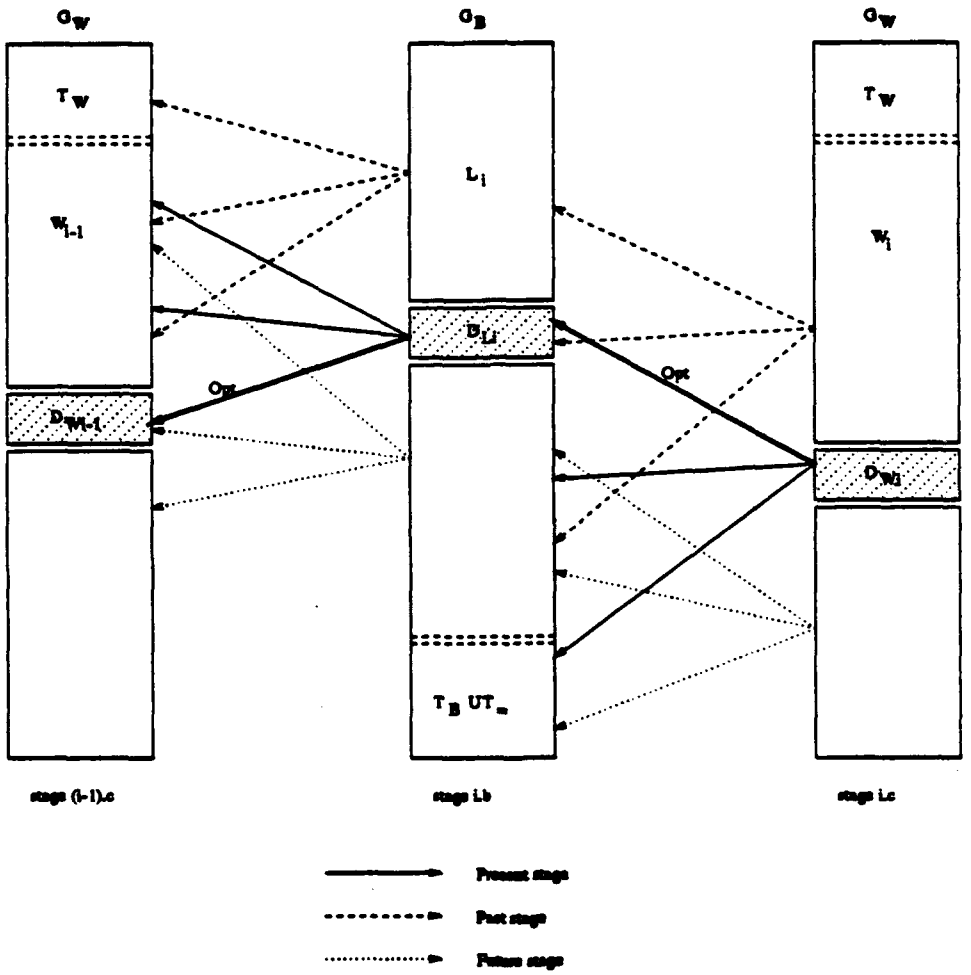

Figure 2.3: 'Algorithmic' representation of a game. Rectangles denote the sets as they appear with the algorithm in progress. Arrows denote moves that are possible from positions in various sets. The algorithm proceeds left-to-right, adding new positions to the winning set, W_i, and the losing set, L_i. Shaded areas are the sets of newly added positions. Blank areas are positions not marked as yet.

exponential in $D(p)$ to prove a win, and, if a player is fallible, the probability that he succeeds in playing out the win correctly (against optimal play) is an exponentially decreasing function of $D(p)$.

This remark brings us to the topic of *mistakes* and their significance in theoretical models of games.

2.2.3 Mistakes

So far we have only paid attention to optimal play, as defined by the optimal-solution graph implicitly constructed by the back-up algorithm. However, in a realistic situation it will in general not be possible for a player to understand the game well enough to make an optimal move every time. So, depending on their way of thinking, both players may at some point diverge from an optimal solution.

Based on the definitions, we can distinguish two types of mistakes for either player: *fatal mistakes*, which change the game-theoretic value, and *inaccuracies* or *non-fatal mistakes*, which only change the distance to win with respect to the optimal move(s)[8]. Note that, as a consequence of our asymmetric view of the game, these mistakes are *not symmetrical*: White can make both types of mistakes only in W_W, but Black can make a fatal mistake only in N_W, and an inaccuracy only in L_B. Fatal errors lead either from W_W to N_B , or from N_B to W_W. (Refer back to Fig 2.1, where they are indicated with a '?'.) Inaccuracies only change the distance to win $D(p)$: to larger distances for a White inaccuracy ("up"), and smaller for a Black inaccuracy ("down"). To indicate the possibility of a mistake in a certain position, we could indicate a correct move from this position with a '!', as in Fig. 2.1.

We now have the tools to analyze in which way the domain itself is responsible for increasing the probability or severity of mistakes or inaccuracies, in other words, why some positions are more intrinsically *difficult* than others. All other things being equal, a position can be considered more difficult[9] if, for example,

- there are relatively fewer correct moves
- there are relatively more fatal mistakes
- the magnitude of inaccuracies is larger
- the position has a larger depth

We can now also estimate to which extent these criteria are likely to vary with the depth $D(p)$, in other words how the structure of the domain changes when play proceeds towards

---

COKO III lost the game, in spite of having had the opportunity to deliver mate in 1 move for 7 moves in a row[35].

[8]We ignore for now the second order inaccuracies, which would merely make the position more difficult (or easier for the opponent), but do not make a difference in the distance to win.

[9]There are of course other components of difficulty that cannot be expressed in terms of the game graph, but are related to representational features of the game. These components will here be ignored.

positions closer to the win. An analysis of some components of difficulty will be carried out in part II, in the context of the KQKR endgame.

For now, let us turn to an empirical analysis of small random games.

Chapter 3

An empirical analysis of random games.

To gain insight into what governs the structure of the optimal solution graph, this chapter describes an experiment with some relatively small *random games*, as defined by their graph G.

The first section defines random games, as instances of a class of games with fixed parameters, in which edges between nodes are assigned randomly. In this definition, an attempt is made to preserve several common characteristics of actual games. Section 2 gives gives an informal discussion of difficulty on the basis of the optimal-solution graph of random games. Section 3 describes an statistical analysis of a sample of actual random games. It discusses several important properties of games and how they are dependent on the number of terminal positions. Section 4 recapitulates the results of this chapter.

3.1 Random games

3.1.1 What are random games?

A *random game* is a game selected, according to some distribution, from a certain class of games, consistent with the assumptions and definitions given in the previous chapter, and in addition parametrized by

1. the total number of positions N_W and N_B in G_W and G_B, respectively (for simplicity we shall often assume that $N_W = N_B = N$),
2. the number of terminal winning positions, t_0, and
3. the branching factor b (possibly different for Black and White, and in that case distinguished by b_B or b_W respectively).

We can view a random game with these parameters as a random assignment of the successor function $S(p)$, in which for each position b nodes are chosen as successors in the appropriate

(other) half-graph (G_W or G_B), in combination with a random selection of t_0 positions as terminal positions.

Note that we have ignored the terminal non-wins ($t_=$) in this model. Also, instead of using a fixed branching factor b, it would have been possible to use a constant *probability* for each node to be a successor for a given node. The fixed b was preferred because of simplicity and good correspondence with actual games. Similarly, the number of terminal positions was considered constant, whereas it would also have been possible to assign a fixed probability for each node.

3.1.2 Generating random games

The simplest way to generate random games computationally is to choose all the successors of each position independently. This means that for each position, b positions are chosen independently of the successors of other positions. This can be done simply by choosing a random subset of size b from the appropriate half-game for each position.

However, in many games the *unmove branching factor*[1] is roughly equal to the branching factor b: any position with White to move can be reached from about b_B different positions with Black to move, and vice versa[2]. Therefore, in most of the experiments "conservation of flow" was enforced, by limiting the number of predecessors of each position to b (or b_B). This second model can be implemented simply (though perhaps not optimally) as a random permutation of a vector of size $n \times b$.

3.1.3 An example game.

Here is an example of a random game, which may also clarify some of the concepts and procedures described in earlier sections.

The game's parameters were $N = 10$, $b = 2$, and $t_0 = 4$. The successor function $S(i)$ was randomly generated in such a way that the unmove branching factor would also be equal to two. As the successor function is undefined for terminal nodes, the specification of $S(i)$ for positions $i \in T_W$ is arbitrary, and is given in parentheses in the table. The complete solution of this 'raw' game was then found by computing the depths $D(i)$ for all nodes i. The result is shown in tabular form in Table 3.1. Note that depths are here given in *ply* (half-moves). Nodes from which a forced win is impossible are marked by a '–' in the $D(i)$ column.

We see that with White to move 7 out of 10 positions are won. If only non-immediate wins are considered, 3 out of 6 positions are 'non-trivial' wins. All these wins are 2 ply deep (1 move) which is also the maximin depth (longest forced win). Black to move loses in 6 out

[1] *Unmoves* are operators leading to predecessor positions of the current position, i.e. the elements of $S^{-1}(p)$. The unmove branching factor is the number of positions from which the current position can be reached.

[2] This is approximately true in chess, for example, as long as captures are excluded, as captures greatly increase the unmove branching factor in some regions of the domain.

Node # (i)	White to move $S_W(i)$	$D_W(i)$	Black to move $S_B(i)$	$D_B(i)$
0	(5) (5)	0	2 1	1
1	(2) (6)	0	2 9	3
2	(1) (8)	0	4 8	–
3	(1) (3)	0	0 1	1
4	8 7	–	5 6	–
5	7 2	–	9 8	3
6	4 4	–	3 3	1
7	0 9	2	5 4	–
8	0 3	2	7 6	–
9	6 9	2	7 0	3

Table 3.1: Tabular representation of an example random game.

of 10 positions. Half of these losses take 3 ply (1.5 moves), which is the deepest loss for this game.

If positions with identical node numbers correspond to the same 'position', but with different players to move, we can define the concept of *Zugzwang* for random games. *Absolute Zugzwang* is a situation where neither player would want to move if he was given the option to 'pass'. This happens when the value (or depth) with Black to move is *smaller* than the value (or depth) with White to move. In the example, positions 5 and 6 are Zugzwang, since they are drawn if it is White's move, and lost, in 3 or 1 moves respectively if it is Black's move.

This game could be represented in bipartite graph form (see Figure 2.1), but a more meaningful representation is one in which the nodes are arranged in order of their depth $D(i)$, as in Figure 2.3. This has been worked out completely in Figure 3.1 below. Like the table, this figure gives a complete representation of the game. In addition, it is immediately clear for every move whether it is optimal or not (optimal moves make one step 'down' and are represented by solid arrows in the figure).

3.2 Difficulty and mistakes in random games

As we see in the example game of Figure 3.1, both Black (◯) and White (□) occasionally have the opportunity to make non-optimal moves. As a first approximation of the concept of difficulty, we could call positions *difficult* from which such mistakes are possible (as opposed to positions where there are only optimal moves). A deeper approach is to impose an *ordering* on positions to reflect how often a player will get an opportunity to go wrong on the way to the win (or draw).

Consider again Figure 3.1. Mistakes by White (boxes) correspond to arrows leading upward (delaying the win). The figure shows two instances of this type of mistakes: from

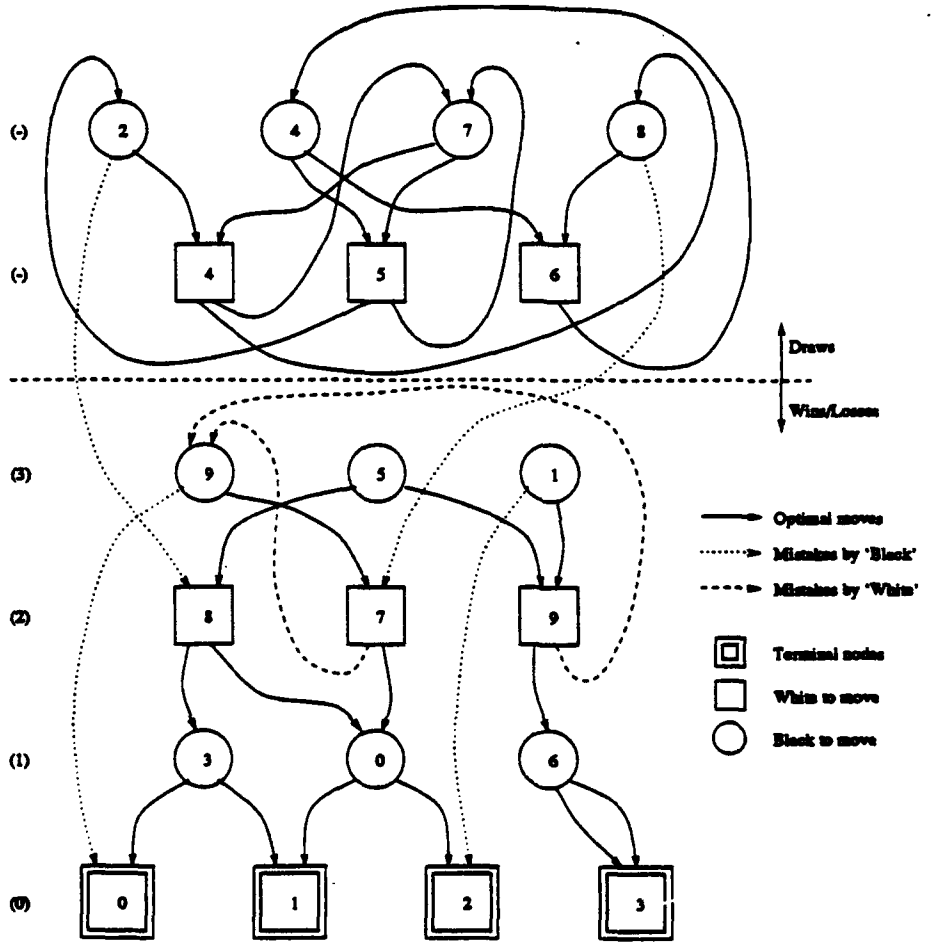

Figure 3.1: Optimal Solution Graph of the example random game. Numerical labels denote the number of half-moves required for a win. The upper portion of the graph (draws) consists of positions from which no forced winning strategy exists.

(white) nodes [7] and [9] to (Black) node (9). Both mistakes are non-fatal, i.e. White can still win afterwards. In this particular game, there are no instances of fatal errors for White, i.e. moves that would throw away the win.

Mistakes by Black are represented by dotted arrows that point downward by more than one level at the time. In the example, there are two instances of inaccuracies (from node (9) to [0] and from node (1) to [2]), which only make the win shorter. There are also two instances of fatal mistakes (from (2) to [8] and from (8) to [7]), which transform a draw into a loss.

Clearly, node (8) is more difficult for Black than node (7), as he has the opportunity to make a mistake. Consequently, White, when playing from his node [4], would do better to play to node (8). This implies a *partial ordering* of drawn positions depending on the likelihood that an opponent will make a mistake.

In a similar vein, we can try to impose an ordering on White's winning nodes at each level. We can say, for example, that, although nodes [7], [8] and [9] are on the same level (two ply away from the win), nodes [7] and [9] are *more difficult* for White than node [8], since White has an opportunity to make the winning process longer. Hence, Black's best move from his node (5) would be to White's node [9] as it is more difficult than the alternative node [8].

It is an interesting question whether it is possible to define an ordering in a meaningful way for every random game, corresponding to the difficulty to win or draw in each position. Although difficulty depends on the opponent and/or features of the game, it is possible to study this concept purely on random games, for example by assuming a certain probability for certain types of mistakes (one can for example assume all moves equi-probable). Several approaches suggest themselves, but were not further elaborated in the context of random games in this thesis. However, the ordering of moves based on their difficulty was used implicitly in the heuristic programs used in the experiments of Chapter 7 (via the probability vector for legal moves).

3.3 An empirical analysis of small random games.

To get some insight into how the parameters influence the structure of the game, an obvious first step is to analyze a sample of small random games with given parameters.

Some emergent properties we are interested in are

- the expected number of won (lost) positions for a game, and the distribution of this variable,
- the expected maximum depth ('maxdepth') of a game,
- the overall class of the game (won, drawn, etc.), and
- the distribution of positions over depth.

Although it is not clear to what extent these properties will have well behaved distributions, it can be hoped that the results will be indicative. There are other interesting properties that can be derived from the structure of random games. Some of these are analyzed in the context of the KQKR endgame (such as the expected number of optimal moves in a position, for example as a function of depth); others need to be referred to future work (see section 9.1).

3.3.1 An experiment

The remainder of this section describes an experiment with random games with size $N = |G_W| = |G_B| = 20$, and branching factor $b = 2$. The number of terminal positions, t_0, varied from 2 (the minimum value required to have losing positions) to 12. For each value of t_0, 100 random games were generated, solved by means of the retrograde algorithm, and analyzed.

The choice of the game parameters was in part opportunistic: as the programs were all written in Common Lisp, running time was a factor (to obtain a reasonable sample). But it also turned out that even games this small exhibit many of the characteristics of larger random games, making the following analysis qualitatively representative.

3.3.2 Expected number of winning positions

Given the total number of states, the branching factor and the number of terminal wins, how many positions can one expect to be won with White to move or lost with Black to move?

The total number of positions in the non-drawn portion of the game graph was computed for 100 games for each value of t_0. The *average value* over this 100 game sample is given in Figure 3.2 (note that no values are given for $t_0 < 2$ because there can be no losing positions unless $t_0 \geq b$). We see that the average value gradually increases from 5% when $t_0 = 2$ to 95% when $t_0 = 9$. More than 50% of the domain is won or lost when t_0 is 6 or greater (indicated in the figure).

From the plot in Figure 3.2 one could get the impression that the number of winning positions is likely to increase gradually with the number of terminal positions for most of the random games in the sample. This turns out not to be the case, however. Instead, the increase in average reflects a change of relative proportion of certain classes of games in the sample. As an example, consider the distribution (histogram) of the number of winning positions for $t_0 = 6$, which is given in Figure 3.3. We see that, in this 'middle region' of t_0, a game is either *trivially drawn*[3] (there are 6 winning positions, which are exactly the t_0 terminal positions), mostly drawn, (less than about half the positions are won or lost), or *totally won* (all but a few positions are won). We could call the mostly drawn games *interesting draws*. There is a lot of 'play': the win may take many moves, and both players

[3]With the conventions adopted in this paper, the game could still fall into any of the categories from Black's point of view.

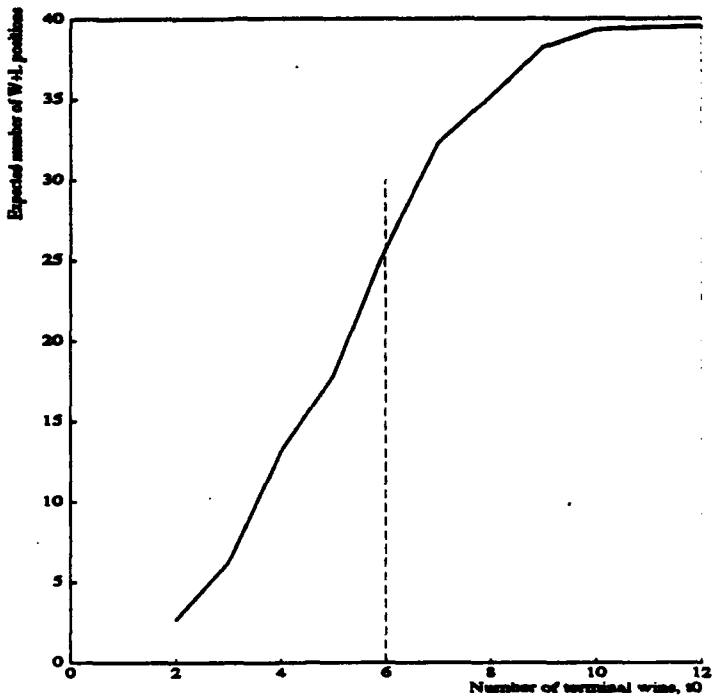

Figure 3.2: Expected number of winning positions in a game with $b = 2$ and $N = 20$, as a function of t_0.

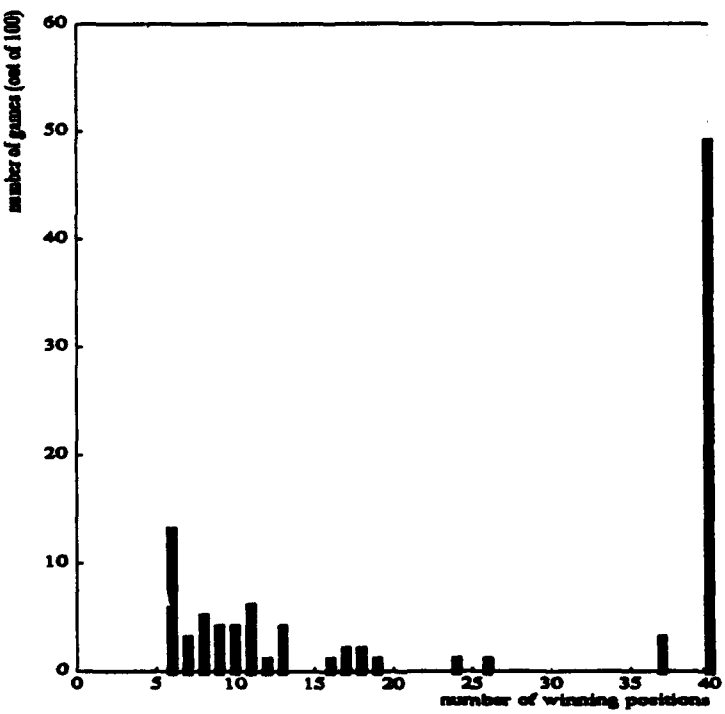

Figure 3.3: Distribution of the number of winning positions for games with $b = 2$,$N = 20$, and $t_0 = 6$.

have opportunities to make mistakes, both fatal or non-fatal. It is interesting to observe that there are virtually no games 'in between' the last two categories.

When t_0 increases, the proportion of 'total wins' increases at the expense of (mainly) the proportion of trivial draws. Any single game, however, falls within one of the categories, and has correspondingly a small or very large number of winning positions.

3.3.3 Expected category of a random game

Let us now look at the probability that a random game can be classified into one of the above three categories. To get an impression of how many games of each kind are possible with certain parameters, Figure 3.4 shows their proportion for our experimental class of games ($N = 20$ positions for each player to move, and a branching factor $b = 2$, with t_0 variable). In the figure, the absence of shading indicates the trivial draws, the black area corresponds to the total wins, and the gray area in between to the 'interesting' games.

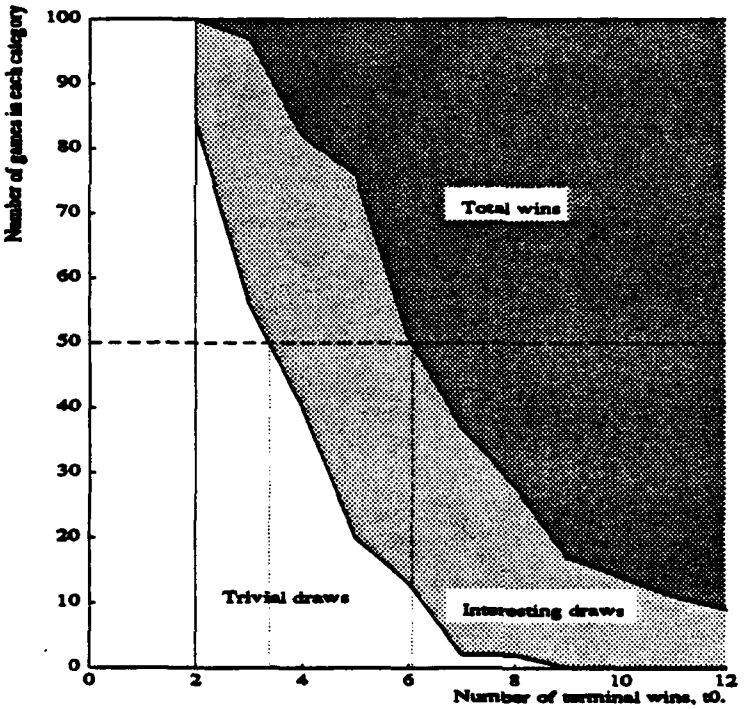

Figure 3.4: Proportions of categories of games, for $N = 20$ and $b = 2$, as a function of t_0.

We see that most games with t_0 lower than a certain value (about 3) will be trivial draws, and most games will be total wins when t_0 is larger than a 'critical value' of about 6.

3.3.4 Maximum depth to win

Another interesting emergent parameter of a game is how long the win takes in the worst case. How this largest depth ('maxdepth') is distributed for random games with given parameters

is illustrated in Figure 3.5. Represented in the figure are the first and 99th percentile of the sample distribution of the maxdepth, as well as its average value. We see that the depth

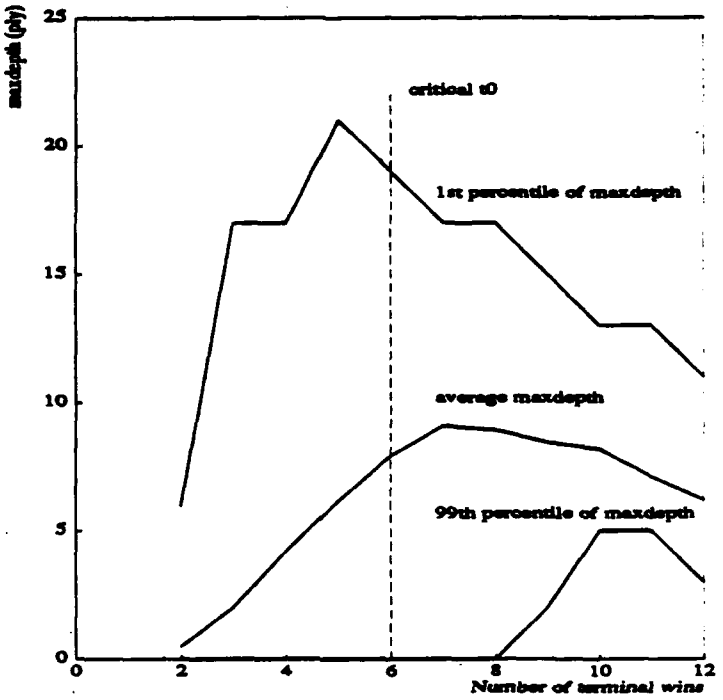

Figure 3.5: Largest (1%), smallest (99%) and average maximum depth.

gradually increases with t_0, reaches a maximum just after the critical value $t_0 = 6$ and then decreases with increasing t_0. If the maximum depth is an indication of how 'interesting' a game is, then values of t_0 close to the critical value lead to more interesting games.

3.3.5 Distribution of depths

In addition to the maximum depth, it is interesting to know the overall shape of the distribution of depths over positions. How many positions are shallow, and how many are deep?

With games of the size under consideration here (only 20 positions for each side to move), one cannot expect to obtain reliable distributions: the impact of random fluctuations on the end result is too large. Nevertheless the main properties of the distributions are striking, and occur in larger random games as well. In further sections, we shall see that the same properties also follow from a mathematical model of random games, and hold for many actual games as well.

The following figures show some typical distributions of positions over depth, both taken from the sample of games with parameters $b = 2$, $N = 20$ and $t_0 = 6$. Figure 3.6 shows an example of a total win, and Figure 3.7 shows an example of an 'interesting draw'. The two distributions start off very much alike, as can be expected, and both show a decrease in the

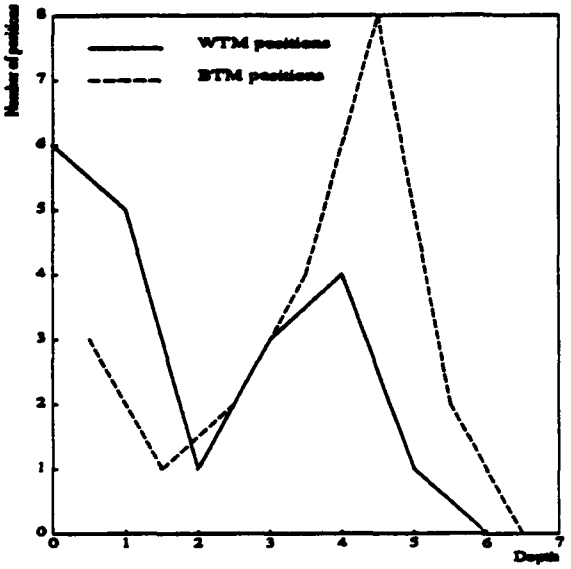

Figure 3.6: Typical distribution of depth for a 'won' game ($b = 2, N = 20, t_0 = 6$)

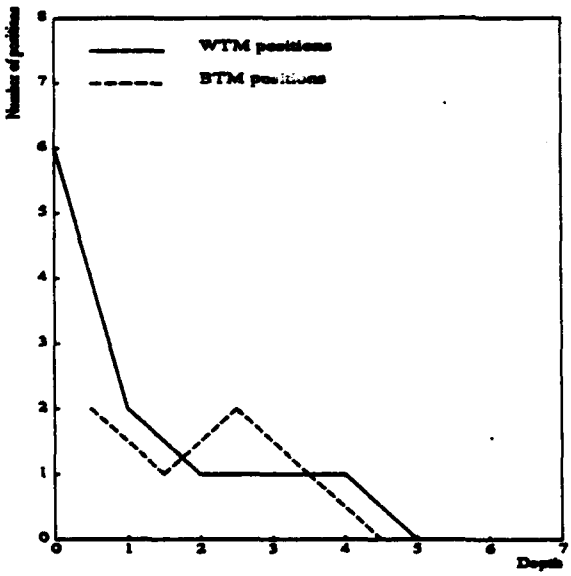

Figure 3.7: Typical distribution of depth for a 'drawn' game ($b = 2, N = 20, t_0 = 6$)

number of positions with depth. But, after the 'bottleneck' at depth 2, the distribution for the 'won game' increases again to a second maximum, followed by a final decline, whereas the distribution for the drawn game hovers, and then dies out.

3.4 Observations

Several intriguing observations can be made based on the data described above. Most of these occur also in larger random games, and, qualitatively, in real games (see the following chapter).

1. Games can be classified into *three classes*: total wins, interesting draws and trivial draws. Although the total number of winning positions in an 'interesting draw' varies considerably, there are virtually no games that are 'mostly wins', i.e. there are always a sizeable number of drawn positions.

2. The *absence of 'mostly won' games* will be substantiated mathematically in the next chapter. Intuitively, once a *'critical mass'* of winning positions is reached, the probability that any unlabeled position can be 'forced' into this set of wins will become large enough to ensure additional losses, and hence additional winning positions, until the domain 'fills up' (snowball effect!).

3. Most 'won' games have a *bimodal depth distribution*. The intuitive explanation is very similar to the one in the previous point. Again the next chapter will provide mathematical support.

4. The character of a random game is strongly determined by *the number of terminal positions* t_0 (the influence of the branching factor was not explored here). Small changes in t_0 around a critical threshold (about 6 in the example) have a profound effect on the expected structure of the game.

5. 'Deep games' (i.e. games with a large value for the maximum depth) occur mainly *near the critical threshold* on t_0. For larger values, not only does the number of available positions (non-terminal) decrease, but the probability that any position will be marked as win/loss at a shallow depth becomes larger rapidly. For smaller values of t_0, the depth decreases because the probability soon becomes too small to support the addition of new losing positions.

Chapter 4

A mathematical analysis of the properties of random games

This chapter presents mathematical analysis concerning the properties of random games. A set of equations is worked out to model the approximate distribution of the distance to win for an average random game with given parameters. This system of equations is solved numerically for certain cases. Further analysis substantiates the observed dichotomy between wins and draws, as well as some other properties of games.

In section 1 a set of equations is proposed as a simplified model of the number of positions which are included in the set of wins at each stage of the retrograde algorithm. This set of equations has as parameters the branching factor, the number of terminal positions, and the size of the domain. Section 2 discusses one simple aspect of this equation, namely the minimum number of terminal positions required for a game to have any non-trivial winning positions. In section 3 this set of equations is reduced to one highly non-linear difference equation. Some numerical results for small branching factor are presented. Section 4 exhibits proof that there are essentially two classes of non-trivial games, non-wins and wins, and that the dividing line between these two classes is a very sharp one. Section 5 compares the actual distributions of actual random games with the predictions made by the (numerical) solution of the game equations. Section 6 makes a (qualitative) comparison between the results obtained in this and the previous chapter with the properties of actual chess endgames, as computed on the basis of their endgame databases.

4.1 A mathematical model

This section develops and analyzes a difference equation to model the distribution of positions over depth. We model the expected size of the relevant sets in the course of the execution of the retrograde-analysis algorithm, and make the following simplifying assumptions and approximations to make the analysis tractable.

1. The expected number of positions is a *real number* (i.e. ignore rounding or truncating). This assumption makes most of the analysis independent of the domain size, N, and makes an analytical study of the game equation possible.

2. During each step of the algorithm, all the positions of the appropriate half-graph (i.e. G_W or G_B) are equally likely as successors for a candidate position for L_i and W_i respectively.

3. The probabilities that a node at some level has successors in a certain portion of the graph is the same for all successive nodes considered at this level (i.e. we approximate the probabilities as if we were sampling *with replacement*).

4. The probability that a node is a win at level at most i is independent of the probability that a node is a win at level $i-1$ (i.e. it only depends on the number of losses on level i or less).

5. There are no terminal Black wins or draws ($T_= \cup T_B = \phi$) (i.e. there are no Black winning positions or immediate draws, such as stalemate).

Section 8.1 will discuss the impact of these assumptions.

Let w_i be the expected number of winning positions and l_i the expected number of losing positions at depth i or less[1]. Then the assumptions allow us to write the difference equations governing the number of positions at each level as

$$\begin{aligned} w_0 &= t_0 \\ l_i &= N \times \left(\frac{w_{i-1}}{N}\right)^{b_B} \\ w_i &= w_0 + (N - w_0) \times (1 - (1 - \frac{l_i}{N})^{b_W}) \end{aligned} \tag{4.1}$$

The independent parameters of this model are, as for the corresponding random games, the size N, the branching factors b_B and b_W, and number of terminal wins t_0.

In the remainder of this section, we explore the properties of this statistical model. Even without relying on the assumptions it is possible to obtain an accurate[2] analytical lower bound on the t_0 required in order to have at least one losing position at level 1 ($B_1 \neq \phi$). This is given in the first subsection.

---

[1] The quantities w and l are written in lower case to emphasize the fact that they are meant to be statistical averages, rather than sets or exact numbers.

[2] "Accurate" should here be interpreted as "not dependent on the assumptions".

Then a numerical solution is given for a simple case (branching factor $b = 2$), and its behavior analyzed. The observations we can make based on the numerical solution are further mathematically substantiated in the next subsection. In the last part of this section the impact of the various assumptions is discussed. It turns out that the model, in spite of these simplifications, provides an excellent fit for the empirical results of the previous section.

The basic questions this section tries to answer are (1) under which conditions (which region in the parameter-space) a game is mostly won or mostly drawn, and (2) what the distribution of winning positions over depth-to-win looks like. But first, let us take a look at an easier question, namely, under what conditions a game is likely to contain any (non-trivial) winning positions at all.

4.2 A necessary condition for a game to be interesting.

Clearly a game that has no non-trivial winning positions is not interesting. Let us estimate the number, t_0, of terminal winning nodes we need in order to have at least one non-trivial winning position, i.e. $W_W \neq T_W$.

Theorem 1 *A necessary condition for the existence, with a greater than even probability, of non-trivial winning positions, is that*

$$\frac{t_0}{N} \geq \sqrt[b_B]{1 - \sqrt[N]{\frac{1}{2}}} \tag{4.2}$$

Proof: In order to have a non-trivial winning position, there must be at least one losing position in L_B. This means that there must be at least one position of which all the successors are in T_W. The probability of this event is the complement of the probability that there are no such positions, i.e. that all positions in G_B have at least one successor outside of T_W. The probability that one specific position $p \in G_B$ has all of its successors in T_W is $\left(\frac{t_0}{N}\right)^{b_B}$.

Now we can write the condition that we have a 50% probability that at least one losing position exists as

$$1 - \left(1 - \left(\frac{t_0}{N}\right)^{b_B}\right)^N \geq \frac{1}{2}$$

This can be rewritten as equation (4.2). ∎

When N is large (as in most interesting cases) we can approximate $\sqrt[N]{\frac{1}{2}}$ by $(1 - \frac{1}{N}\ln 2)$. This allows us to simplify equation (4.2) to

$$t_0 \geq N^{\frac{b_B - 1}{b_B}} (\ln 2)^{\frac{1}{b_B}} \tag{4.3}$$

Figure 4.1 shows how the required value of t_0 changes according to equation (4.3), for a domain size of 2 million (this is about the size of a 4-piece chess endgame without pawns). The figure shows that for a branching factor of about 20 (as in KQKR) *half* of the domain needs to consist of terminal winning positions in order to have any non-trivial wins (dotted line). It may seem surprising that this result is dependent on the size of the domain, even

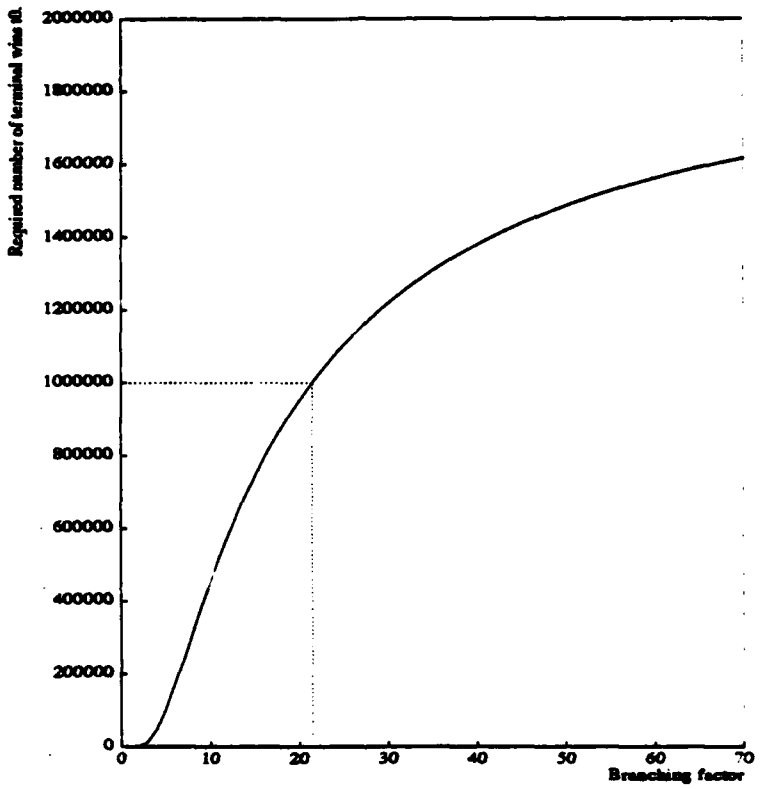

Figure 4.1: Number of terminal wins, t_0, required, as a function of b, for a game with $N = 2 \times 10^6$ to be probably interesting.

though the equations (4.1) contain only proportions. The reason is that we require here that a *fixed number* of positions be losing positions (i.c. 1), and that any fixed number forms a smaller or larger portion of the domain depending on N. The possible effects of round-off errors in real domains will be discussed further later.

As a numerical example of the use of equation (4.3), consider the endgame of King and a-pawn against King. The size of the domain is about $6 \times 60^2 = 21600$ (six squares for the Pawn, and approximately the whole board for the Kings). The branching factor for Black is roughly $b_B = 5$. Using equation (4.3), we calculate that we need about $21600^{\frac{1}{5}}(\ln 2)^{\frac{1}{5}} = 2700$ terminal wins. But the number of immediate wins (either promotion or capture of the Black King) sums up to about 6,500 positions, so we can expect non-trivial wins to exist.

As another example, consider random games with parameters $b = 2$ and $N = 20$, which are the parameters used in the empirical analysis of section 3. Plugging these values into

equation (4.2) gives us

$$t_0 \geq 20 \times \sqrt{1 - \sqrt[20]{\frac{1}{2}}} = 3.7$$

This is in perfect agreement with the experiments, as can best be seen on Figure 3.4.

4.3 A numerical solution of the game equations

The set of equations (4.1) can easily be solved numerically for various values of the parameters[3].

Let us first rewrite the equations. Let $b_B = b_W = b$, $\alpha = t_0/N$, $\xi_i = w_i/N$, and $\eta_i = l_i/N$. Then ξ_i is the proportion of G_W which can be won in i moves or less, η_i is the proportion of G_B which is lost in i moves or less, and α is the proportion of the domain which is won immediately. After also eliminating η_i from the second equation, we obtain

$$\begin{aligned} \xi_0 &= \alpha \\ \xi_i &= 1 - (1-\alpha)(1-\xi_{i-1}^b)^b \\ \eta_i &= \xi_{i-1}^b \end{aligned} \tag{4.4}$$

These equations allow us to compute the values ξ_i and η_i iteratively for any value of α and b. The values of ξ_i are plotted in figure 4.2, for $\alpha = t_0/N$ ranging from 0.08 to 0.40 (the numbering on this axis is arbitrary), and i from 0 to 30. From the figure we clearly see that, if α is large enough, the whole domain consists of winning positions. If α is smaller, less than half of the positions are winning positions. The threshold (T_α) that separates these two cases is sharply defined (as we shall further substantiate below).

Perhaps a more usual form of expressing the same information is by plotting the *densities* $\delta\xi_i = \xi_{i+1} - \xi_i$, which indicate the proportion positions that are wins in *precisely* i moves. They are plotted in figure 4.3, again for $\alpha = t_0/N$ ranging from 0.08 to 0.40 (the numbering on this axis is again arbitrary), and i from 0 to 30 (from a different point of view to show more clearly the relevant parts of the figure). In the figure, consider the shape of the distribution for $\alpha = \frac{t_0}{N}$ varying from smaller to larger values (0 through 30 in the arbitrary coordinates in the figure). At first, we have a typical drawn distribution: one first peak (t_0) and a rapid decrease to zero. When α increases, the shape remains identical, but the number of positions at each depth gradually increases (leading also to an increase in maximum depth). At $\alpha = T_\alpha$ a sudden transition to won games occurs. The density function now shows a second maximum, and a 'bottleneck' region in between the two maxima. Initially, the second maximum occurs at large depths (in fact at infinity for $\alpha = T_\alpha$). When α increases further, the 'critical mass' of positions needed for the second maximum is reached sooner, leading to a gradual shortening of the bottleneck. Observe, however, that the shape of the second maximum does not vary much. Eventually, the maximum depth becomes so small that the two maxima merge and the bottleneck disappears. Clearly, the maximum depth of a game

[3]In this thesis, this was done by means of the Mathematica package, which was also used to generate most of the 3-D plots.

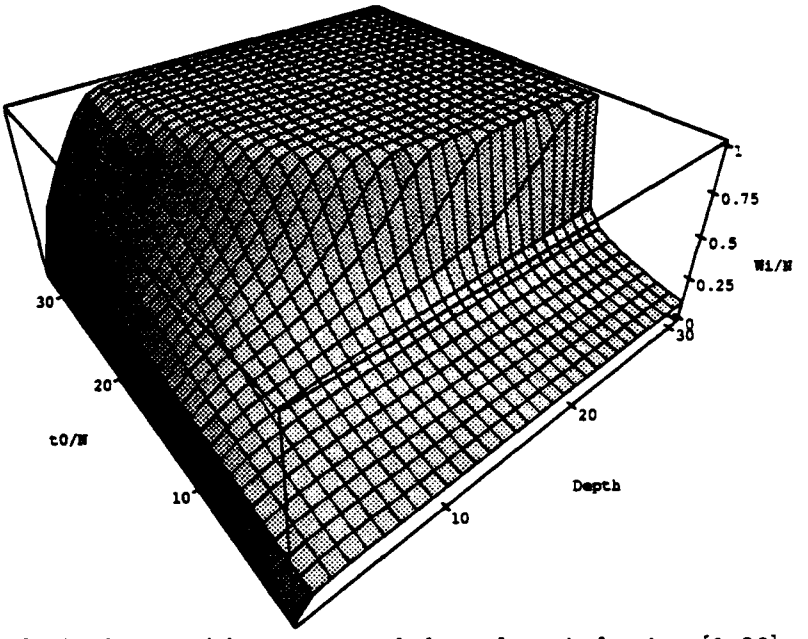

Figure 4.2: Number of winning positions at depth less than i, for $i \in [0, 30]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 2$).

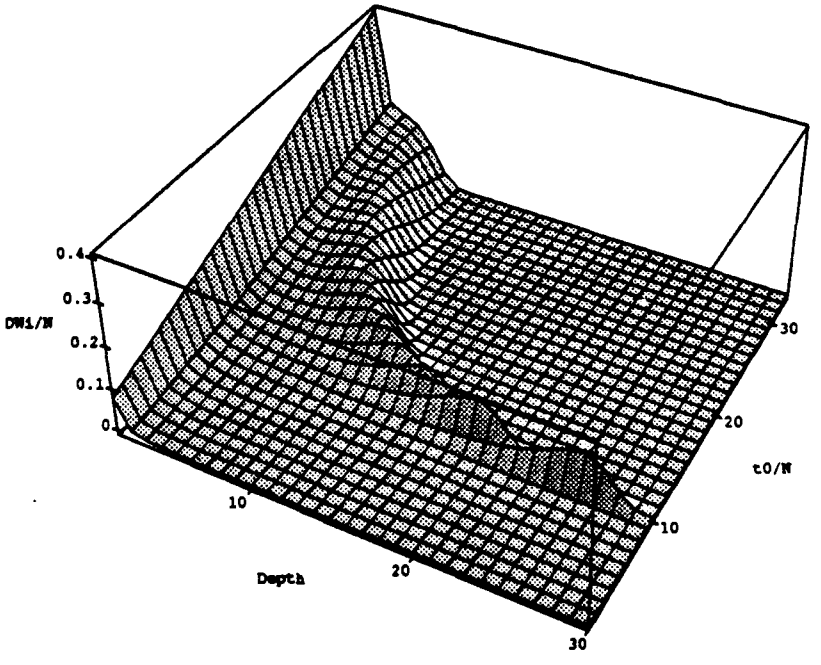

Figure 4.3: Number of winning positions at depth i, for $i \in [0, 30]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 2$).

increases when α approaches T_α. But when $\alpha > T_\alpha$ all the positions in the domain are winning, whereas when $\alpha < T_\alpha$ a significant portion of the domain consists of draws.

From the proportion of winning positions for White ξ_i, we can compute the distribution of losing positions for Black by means of the equation $\eta_{i+1} = \xi_i^b$. These values are plotted in figure 4.4, with the same range as above (except for $i = 0$).) As before, we are sometimes

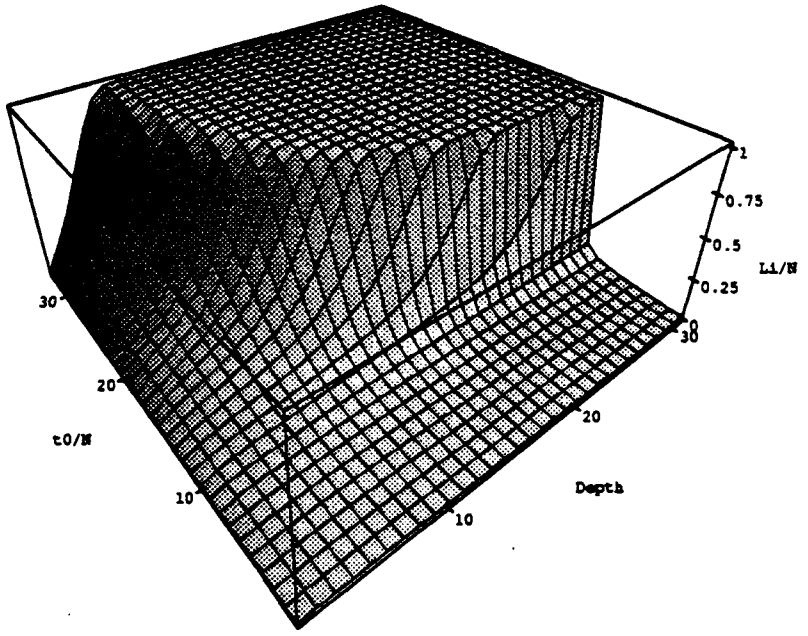

Figure 4.4: Number of losing positions at depth less than i, for $i \in [1, 29]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 2$).

more interested in the density function $\delta\eta_i$, which is plotted in figure 4.5.

In comparison, figure 4.6 plots the cumulative values ξ_i, and figure 4.6 plots the densities $\delta\xi_i$ for a branching factor $b = 3$. Notice that the threshold value for α has increased, as we could expect, and that the interesting region has 'flattened' out, so to speak, but that the essential qualitative features remain precisely the same.

4.4 The threshold T_α: when is a game a total win?

Let us try to find a formal explanation for the above observation. Why and how does the class of a game depend on the parameter α (or, equivalently, t_0). We shall see that, for any branching factor b, there will be a *threshold* value $T_\alpha(b)$ that determines whether the game is won or drawn[4]. We shall also find that the basic shapes of the distributions are always the same. Finally, we can indicate several other properties of T_α and the distributions, and define a 'critical mass' of positions necessary for a win.

[4]The probability that a random game is a total win could be called a *threshold function* on t_0, and the theorem is similar to what is called a *Zero-One Law* in Graph Theory[69]. However, here the properties are a function neither of the edge density (i.e. b) or of the total number of nodes (i.e. N). Instead the critical parameter is an extra parameter α, denoting a density of terminal positions.

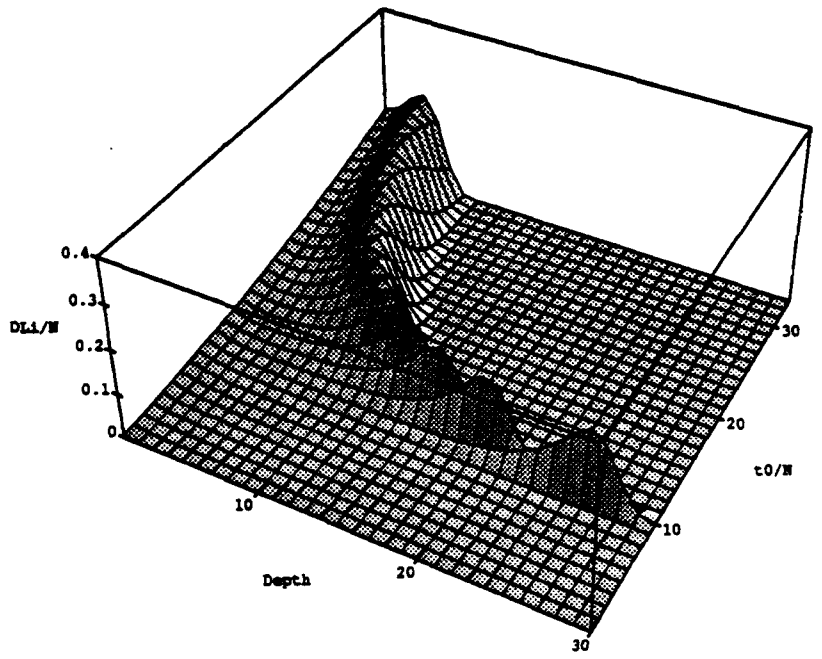

Figure 4.5: Number of losing positions at depth i, for $i \in [1,29]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 2$).

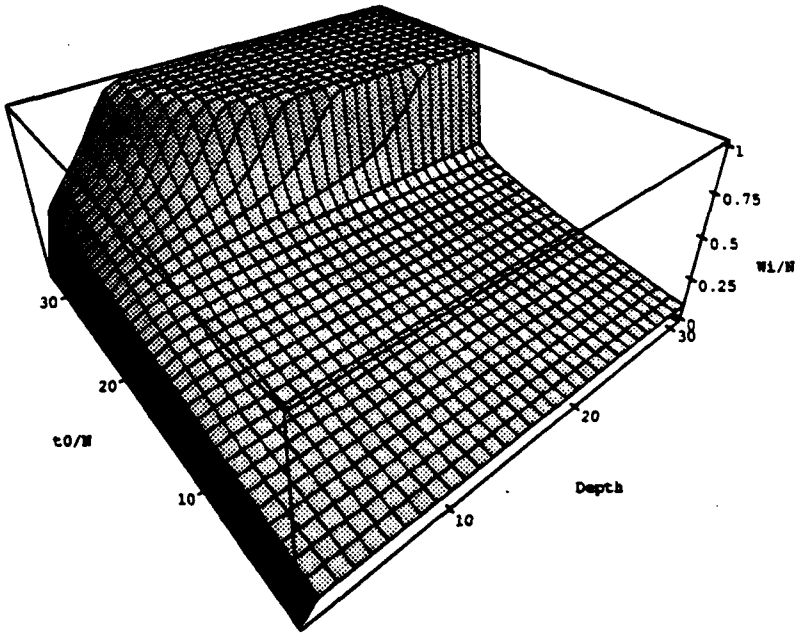

Figure 4.6: Number of winning positions at depth less than i, for $i \in [1,29]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 3$).

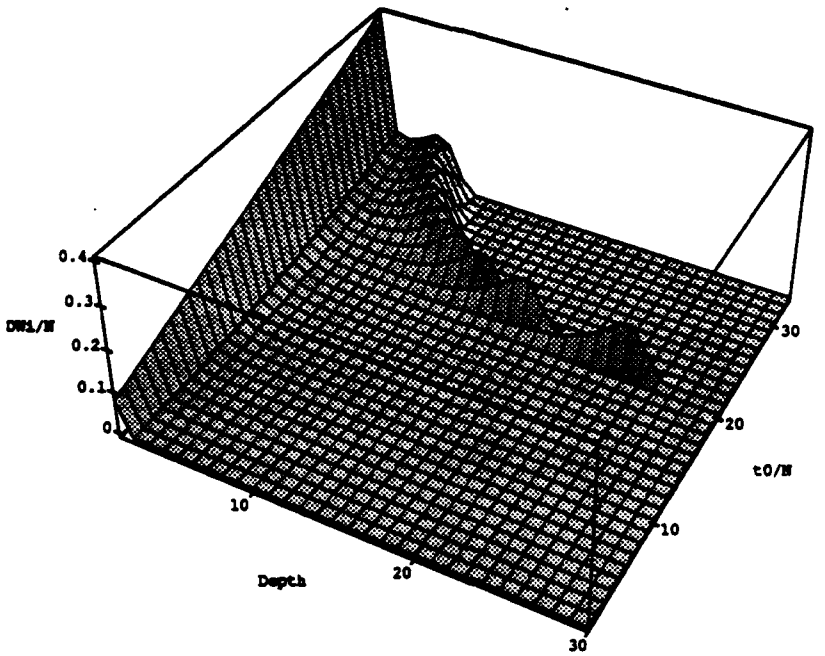

Figure 4.7: Number of winning positions at depth i, for $i \in [1, 29]$ and for $\alpha \in [0.08, 0.40]$ (branching factor $b = 3$).

Theorem 2 *For each branching factor $b \geq 2$, there exists a threshold $T_\alpha(b)$ such that, if $\alpha > T_\alpha$ then $\lim_{i\to\infty} \xi_i = 1$, and if $\alpha < T_\alpha$ then $\lim_{i\to\infty} \xi_i << 1$.*

Proof:

For a won game, we require that $\lim_{i\to\infty} \xi_i = 1$. This is only possible if the ξ_i are monotonically increasing when smaller than 1, or

$$\xi_i < 1 \Rightarrow \xi_{i+1} > \xi_i$$

If, therefore, at any point we have that $\xi_{i+1} \leq \xi_i$, we cannot have a totally won game. Plugging this requirement into equation (4.4), we get, after slight rearrangement and substituting x for both ξ_i and ξ_{i-1},

$$\frac{(1-x)}{(1-\alpha)} = (1-x^b)^b$$

or, rearranging some more,

$$f(x) := \frac{(1-x^b)^b}{(1-x)} - \frac{1}{1-\alpha} = 0 \tag{4.5}$$

Note that $f(x)$ is a polynomial, as $(1-x^b)$ is divisible by $(1-x)$. It is directly related to the $\delta\xi_i$, but with a negative coefficient:

$$\delta\xi_i = -(1-\alpha)(1-\xi_i)f(\xi_i).$$

Hence, $f(\xi_i) < 0$ corresponds to $\xi_{i+1} > \xi_i$. If $f(x)$ has zeroes in $]0,1[$, then $\lim_{i\to\infty} \xi_i < 1$. Conversely, if $f(x)$ has no zeroes in $]0,1[$, it must be negative in this whole interval (since $f(0) = -a < 0$). Accordingly, the ξ_i must continue increasing with i until $\xi_i = 1$.

To find the boundary condition, then, we try to find the value of α for which $f(x)$ has a double real root. This value is the T_a we seek. The function $f(x)$ has a double root when this root corresponds to its maximum value. In the following lemmas, we ascertain that $f(x)$ has one, and only one, maximum in $]0,1[$ independent of α. As a consequence, we can always solve for a unique value T_a.

Lemma 3 *If $0 < \alpha < 1$, then the $f(x)$ in equation (4.5) has at most one extremum in $]0,1[$. This extremum is a maximum.*

Proof: To analyze the extrema of $f(x)$, we find the zeroes of its derivative

$$\frac{\partial}{\partial x} f(x) = \frac{(1-x^b)^{b-1}}{(1-x)} \left(\frac{1-x^b}{1-x} - b^2 x^{b-1} \right) = 0$$

The left part of $\frac{\partial}{\partial x} f(x)$ can only have zeroes of unit absolute value. We can therefore limit ourselves to finding the zeroes of the other part. Define

$$g(x) = \frac{1-x^b}{1-x} - b^2 x^{b-1}$$

This can also be written as

$$g(x) = 1 + x + x^2 + \cdots + x^{b-2} + x^{b-1} - b^2 x^{b-1}$$

or

$$g(x) = \frac{1-x^{b-1}}{1-x} - (b^2-1)x^{b-1}$$

As $g(0) = 1 > 0$ and $g(1) = -b(b-1) < 0$, $g(x)$ must have an odd number of zeroes in $]0,1[$. Let x_g be a zero of g in $]0,1[$, or $g(x_g) = 0$. If we can show that $g(x) < 0$ for $x > x_g$, then we can conclude there cannot be a second zero in $]0,1[$.

We can write the condition $g(x_g) = 0$ as follows.

$$g(x_g) = (1 - C_0 x_g^{b-1}) + (x_g - C_1 x_g^{b-1}) + \cdots + (x_g^{b-2} - C_{b-2} x_g^{b-1}) = 0$$

in which each individual term can be made 0 by a proper choice of the C_i, namely

$$C_i = \frac{1}{x_g^{b-1-i}}.$$

For $x > x_g$, the ith term becomes

$$(x^i - \frac{x^{b-1}}{x_g^{b-1-i}} = x^i \left(1 - \left(\frac{x}{x_g} \right)^{b-1-i} \right).$$

By inspection, this is < 0 when $i < b - 1$. Hence, all terms are negative, and so is the sum.

∎

The value T_α can now be computed as

$$T_\alpha(b) = 1 - \frac{(1 - x_g)}{(1 - x_g^b)^b}$$

Lemma 4 *If $0 < \alpha < 1$, then the $f(x)$ in equation (4.5) has two zeroes, $x_1(\alpha)$ and $x_2(\alpha)$, that have their real parts in $[0, 1[$.*

This follows from the previous lemma.

Lemma 5 *There exists a $T_\alpha \in]0, 1[$ s.t. $x_1(\alpha)$ and $x_2(\alpha)$ are real for $\alpha < T_\alpha$ and complex for $\alpha > T_\alpha$*

This follows from the previous lemma and the prologue of the proof.

This lemma proves Theorem 2. ∎

It is interesting to consider the significance of the single root of $g(x)$ (i.e. the value, which when plugged into $f(x)$ allows us to compute the critical value for α. We could call this root (which we called x_g before) the *critical mass* required for a total win, as the ξ_i can never exceed this value when $\alpha < T_\alpha$.

As an illustration, consider the case $b = 2$. Equation 4.5 becomes

$$f(x)|_{b=2} = (1 + x - x^2 - x^3) - \frac{1}{1 - \alpha} = 0 \tag{4.6}$$

This function has one negative root (at about -1.6) and two positive roots between 0 and 1. The real part of these two roots is plotted in Fig. 4.8. To find the critical value T_α, compute the derivative of $f(x)$.

$$g(x)|_{b=2} = 1 - 2x - 3x^2 = (1 + x)(1 - 3x)$$

The critical mass $x_g = 1/3$. Setting $x = x_g$ in equation (4.6), we find

$$\frac{32}{27} - \frac{1}{1 - T_\alpha} = 0$$

or $T_\alpha = \frac{5}{32}$. In Figure 4.8, the (real) roots indeed converge at $\alpha = 0.15625$ at which point they become conjugate complex numbers. As can be seen in Figure 4.2 and Figure 4.3, this is precisely the discrimination point between wins and draws. Intuitively, if the proportion of terminal positions, α, is less than 5/32, the game will be a draw, and the number of winning positions can be at most equal to the critical mass ($\frac{N}{3}$) of positions. If α is larger than 5/32, the game is a total win, and all positions are winning positions.

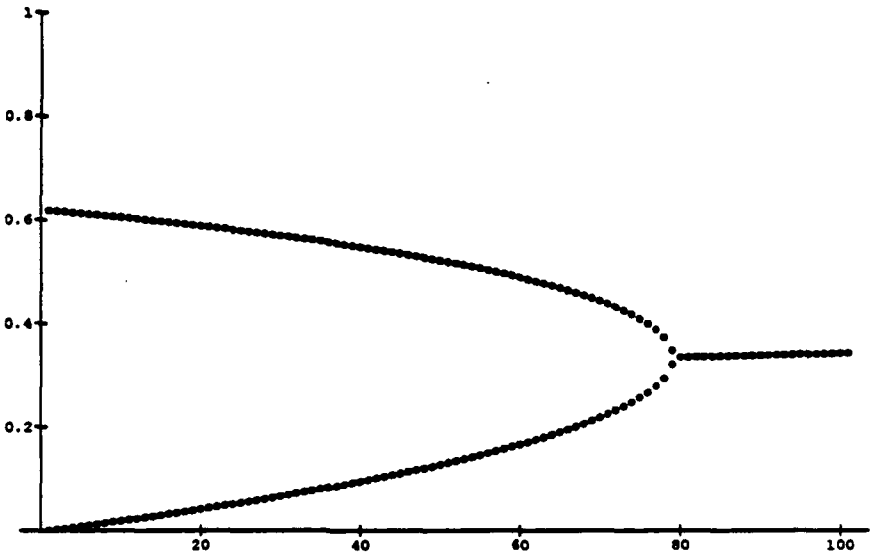

Figure 4.8: Real part of the roots of Eq. 4.5, as a function of α (scaled $\times 500$).

4.5 Comparison with actual random games

The theoretical model given above can be used to compute the expected number of positions at each depth iteratively. As an indication of the value of the model (at least where the purely random games are concerned which it was supposed to model), Figure 4.9 gives three different random games with the same parameters (each consisting of 1000 positions ($\times 2$), 400 terminal wins, and a branching factor $b = 4$). Also represented, in thick lines, are the values predicted by the statistical model. The solid lines indicate the number of positions with White to move (w_i), the dashed lines positions with Black to move (l_i).

We see that all three games are here of the same type (total wins), which corresponds to the prediction by the model. Interestingly, one of the games (game 1) shows an almost exactly the same distribution as predicted by the model (th1). The only difference between the three random games is that the bottleneck is longer or shorter. The reason for this is twofold. First, the number of positions added at each step in the algorithm is dependent on the total number of positions already marked and on the number of positions marked during the last step, but not on the number of steps already carried out. Second, the number of positions in this bottleneck region is very small, and small statistical fluctuations may have a large effective impact on the result.

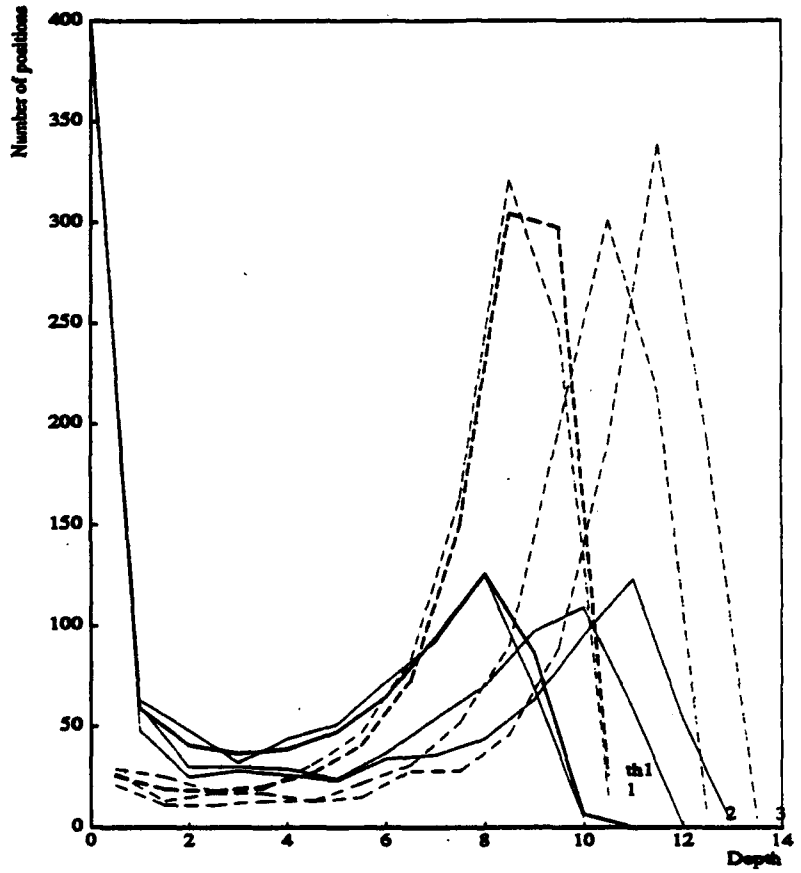

Figure 4.9: Comparison of three instances of random games with the model (parameters $N = 1000$, $b = 4$, $t_0 = 400$). Solid lines give the distribution of positions with White to move, dashed lines the corresponding distributions for Black to move. Narrow lines indicate the three 'real' random games, thicker lines give the distributions predicted by the statistical model.

4.6 Comparison with 'real' domains.

In this section the statistics of random games are compared with corresponding statistics of 'real' domains, in particular simple chess endgames, as the database allows us to compute these statistics exactly. We shall see that there is a surprising correspondence, suggesting that much of the qualitative characteristics are adequately modeled by the random game model. Certain differences are also apparent, and an attempt is made to explain a number of these.

4.6.1 Games with an excellent fit

Let us first look at two games which show an excellent fit with the model, namely the endgames of KRKN and KQKR. That this is so can be explained by the fact that the branching factor is fairly uniform, and that the pieces have a large action radius, hiding geometric effects (e.g. the shape of the board, or particularities in the way pieces move).

King and Rook vs. King and Knight

In Figure 4.10 the (actual) distribution is given of the number of winning positions (solid line) and losing positions (dashed line) for each optimal depth of win for the KRKN endgame. The general shape is like the 'interesting draw' category. Although more than 50% of the positions with White to move are won, by far the most positions with Black to move are drawn. It is appropriate to call such a game a draw, since many of the wins are trivial or very short. Note how well the distribution of this 'real example' fits the distribution predicted by the theoretical model.

King and Queen vs. King and Rook

In Figure 4.11 the (actual) distribution is given of the number of winning positions (solid line) and losing positions (dashed line) for each optimal depth of win for the KQKR endgame. The general shape is like the 'total win' category. Essentially, all positions are won (WTM) or lost (BTM). Although many wins are very short, most losses are long. This means that in any given position, if Black is to move, he is very likely to have at least one move that brings him to a position which will take White a long time to win. Purely statistically then, we can expect KQKR to be very difficult for White to actually win optimally. Again, the distribution of this "real example" corresponds qualitatively rather well with the theoretical model, although in this case the distribution for larger depths (secondary maximum) is lower and more stretched out. Part of the reason for this is that Kings can move only one square at the time, and may need 7 moves to reach to other side of the board. Another reason is the violation of assumptions 2 (equal successor-probability) and 3 (sampling with replacement) in actual games (which is the case for random games as well).

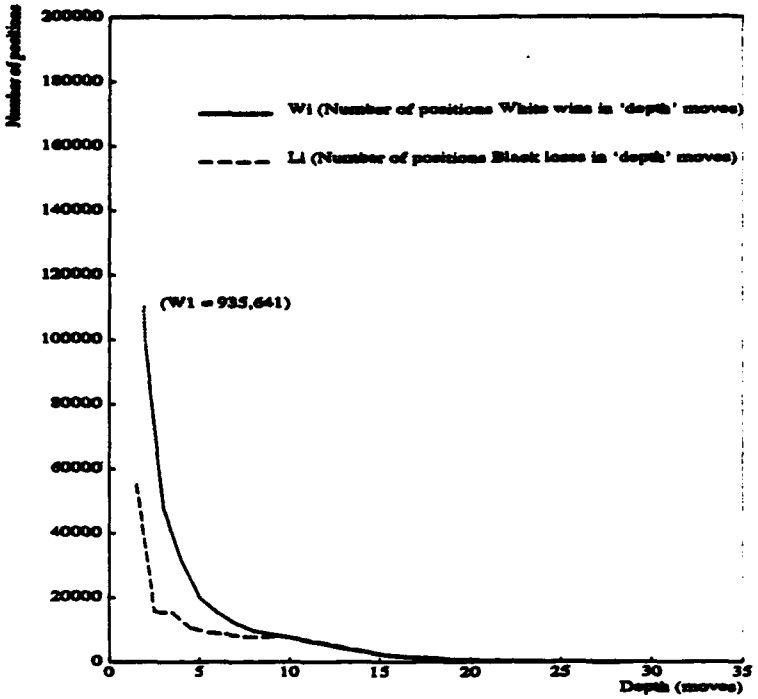

Figure 4.10: Distribution over depth of won positions in KRKN.

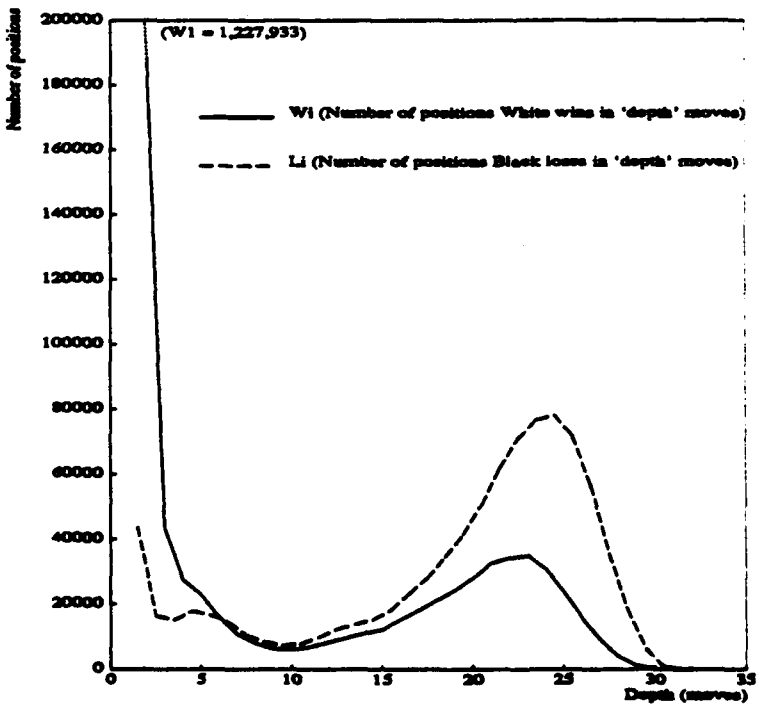

Figure 4.11: Distribution over depth of won positions in KQKR.

4.6.2 Games that don't fit so well

Some other endgames show a much less striking correspondence. Most often this is due to the fact that some pieces move only slowly, introducing localization ('close' positions will have similar features). Some unusual rules may also cause pathological phenomena. In most instances it is the Pawn, a piece full of exceptions, which is the culprit.

King and Pawn vs. King

One of the simplest chess endgames is the endgame where White has a King and Pawn left against Black's King (or the reverse) (KPK). As King and Pawn cannot mate, White needs to force promotion of his pawn, converting into an easily won endgame of King and Queen (or Rook) vs. King (KQK or KRK).

As the pawn can only move straight ahead, and can therefore not change its horizontal position (file), the KPK endgame can be viewed as a composition of four essentially unrelated separate games (rook pawn, knight pawn, bishop pawn and queen/king pawn). The database for KPK used here was indeed constructed in this way (by the author of this thesis). Terminal positions were those positions from which White can either safely promote or capture the Black king.

The distribution of positions over depth for KPK is given in Fig. 4.12. In this figure, positions with White to move (WTM) are indicated with a solid line, and positions with Black to move with a dashed line.

The corresponding distribution for White to move, individually for each of the different pawn positions is given in Fig. 4.13.

The shape of all four distributions looks more like a 'drawn game' distribution than that of a win. Nevertheless, only the endgame of K+a-pawn against K can properly be called a draw. That the second maximum is only barely visible for the other distributions, can be explained by the geometric constraints on the movements of both the P and the K. Note that the spatial constraints only play a role because all three pieces move relatively slowly, and need between 5 and 7 moves to get to the other side of the board.

In addition, there is the surprising peak in the distribution at depth 4, and the larger-than-expected number of positions at depths 1–3. This anomaly is the result of the special rule allowing a pawn to move either one or two squares when it is still on its initial square on the second row. As most wins in 4 moves are positions in which the pawn promotes (in 5 moves, i.e. 4 moves until an immediate win) without the Black King being able to stop it, and a pawn can promote in 5 moves from either the 2nd or 3rd rank, this explains the peak at this point in the graph.

4.6.3 The endgame of two Knights against Pawn

The distribution for the endgame of King and two Knights vs. King and Pawn, given in Figure 4.14 (WTM only), seems unusual. However, as in the previous example, this endgame

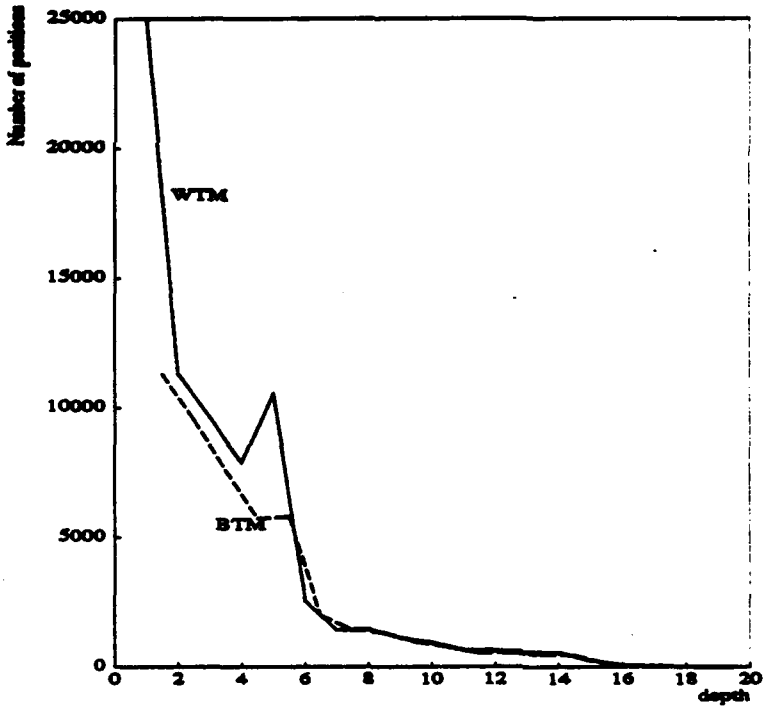

Figure 4.12: Distribution of wins over depth for KPK.

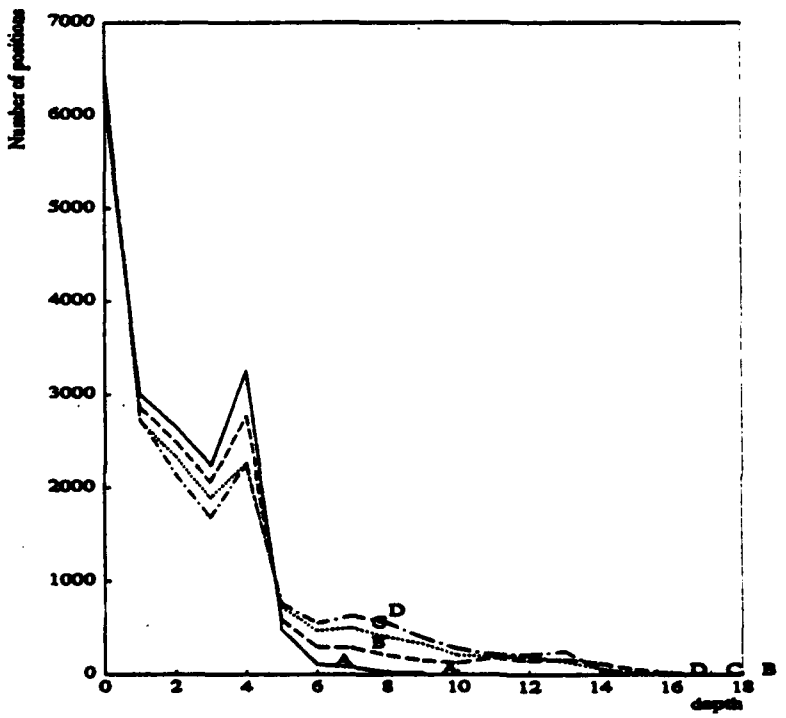

Figure 4.13: Distribution of wins for KPK (each pawn separate).

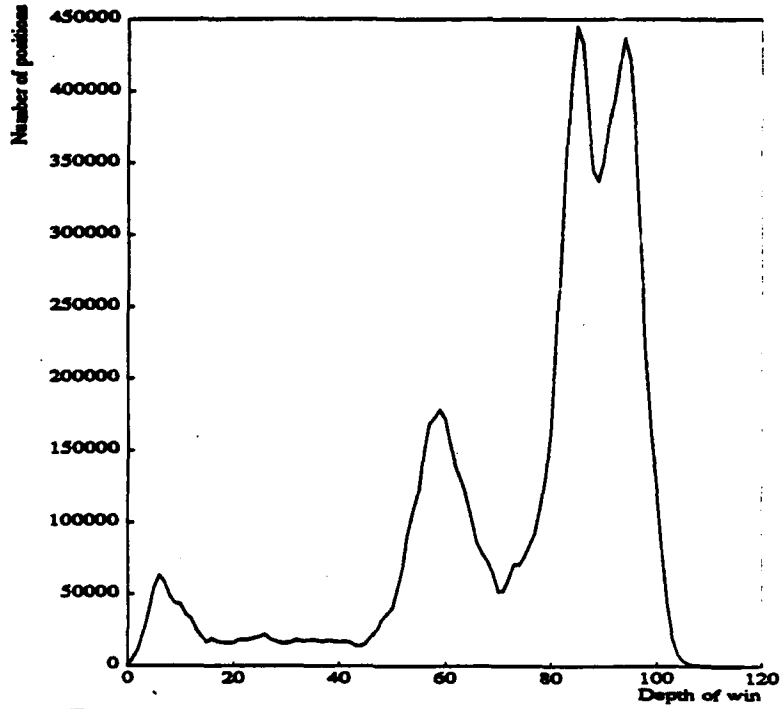

Figure 4.14: Distribution of wins over depth for KNNKP.

is a composite of 4 totally separate games, one with the pawn on each row (one could even argue that it is appropriate to consider different games for all 24 different pawn positions). The four different peaks in the distribution, then, precisely correspond to the four different subgames.

The non-appearance of the initial peak is the consequence of the legality definition used for the construction of the database (the actual chess definition, under which all win-in-1 positions are illegal).

4.6.4 Comment

That 'real games' (like chess endgames) show apparently random characteristics is in part the consequence of the strong 'non-linearity' of the rules of the game, and their interaction with boundary conditions (limitations of the playing terrain). Although the rules themselves can be expressed as fairly simple predicates, their combination with the back-up rule does not allow much simplification and we should therefore expect the complexity (size) of higher-order predicates ('win-in-n') to increase exponentially in n. So the apparent randomness of a predicate expressing a 'win' in terms of simple board features is in fact the result of opaque complexity.

Actual games *do* have some additional structure not captured by the random-game model. Where there exists a strong correlation between some features of the board and the distance to win $D(p)$, the observed properties of the game may deviate significantly from their pre-

dicted values. Also, as the number of pieces in the simple examples considered does not change, the branching factor will be fairly uniform. This cannot be expected to be true of, say, the complete game of chess, as different trades and pawn moves along different branches may make the branching factor vary considerably. In addition, the complete game may be considered built up out of a large number of sub-domains with little or no potential interaction (consider the endgame KPK as an example, in which the four subgames are completely disjoint). Finally, some board features change only slowly from move to move, for instance, when pieces are extremely limited in their movement. This is especially true for pawns, and, to a lesser extent, for Kings. The result is that some of the distributions are 'smoothed out'.

Part II

Experiments with the endgame of King and Queen vs. King and Rook.

In this part of the thesis, several elements of the foregoing chapters are combined and tested in the context of one four-piece chess endgame, the endgame of King and Queen vs. King and Rook (KQKR). We look at the structure of KQKR, observe how well human chess players play this endgame, discuss the validity of certain plausible heuristics, and test the hypothesis that a computer program can exploit knowledge both about the domain and about human heuristics in increasing the difficulty of the winning procedure for human chess players, thanks to, or in spite of, occasional non-optimal moves (see section 1.4).

A computer program that has access to an endgame database is capable of optimal play. When playing from a winning position, such a program will win in the least number of moves possible against another optimal player. When playing from a losing position, it will hold out the longest possible.

The issue is different, however, when the program plays a fallible opponent, who will at times make non-optimal moves. With the program playing the Queen the situation is a technical win[5]: optimal play always wins. With the program playing the Rook, however, the situation is interesting. In most positions, even optimal play on the part of the program loses against a perfect opponent. But against a fallible opponent, it may be possible obtain a better than game-theoretic result if this opponent makes errors. Whether a program (or other player) succeeds in this may depend on whether it is able to increase the likelihood of mistakes on the part of the opponent, i.e. whether it is able to choose moves that lead to positions which are more difficult for the opponent. This part of the thesis will focus on this situation: where a computer program plays the Rook against a human opponent, starting from a position which is game-theoretically lost for the program. The program attempts to delay the loss long enough for the game to be drawn according to the 50-move rule.

In chapter 5, the structure of the KQKR endgame is described. Some difficulty-related statistics, as explored in Part I of this thesis, are now presented for KQKR. The main starting point in this discussion is the distribution of positions over depth to win. In the KQKR endgame, this distribution corresponds qualitatively well with the distributions of random games. Some parameters, such as the likelihood that a random move is optimal or win-preserving, which were not discussed in much detail for random games, are here considered more extensively for the special case of KQKR.

Chapter 6 treats human play in KQKR. Several measures are presented and discussed to assess the quality of play for non-optimal players, based on a complete knowledge of the domain. Some of these measures are used to analyze the performance of human chess players in games and game analyses found in the chess literature. Then a number of plausible heuristics are discussed and evaluated both objectively and subjectively in the KQKR domain.

Chapter 7 describes experiments to test the hypothesis that there are situations in which an optimal strategy is not the most effective strategy against a fallible opponent, and that KQKR is an example of a 'real' domain in which such situations occur. Some of the heuristics of the previous chapter are combined in a predictor function, which estimates the depth after

[5]This is only if the depth of the position is not larger than the number of moves allowed for the win. If the depth is too large - which is not possible for KQKR - the initial position must be considered a draw, and knowledge of the opponent may be useful to achieve a win.

one (or more) moves by the fallible opponent. Several heuristic programs are described that use different versions of this predictor function to decide on the choice of move, and are not bound to strict optimality. Experiments were run to compare the performance of these heuristic programs with the performance of programs restricted to optimal play.

In sum, this part is directed toward exhibiting the potential utility of occasionally non-optimal strategies that use some knowledge about a human opponent, in an actual game (KQKR).

Chapter 5

The KQKR domain.

This chapter discusses the structure of the KQKR domain, what makes it difficult, relevant and interesting, and how well it corresponds with the findings on optimal-game graphs for random games. After some general information about KQKR (section 5.1), some properties of the KQKR domain and their computation from the endgame database are discussed (section 5.2). Section 5.3 lists some factors in the difficulty of the KQKR domain for human players. Finally, section 5.4 motivates the choice of the KQKR domain rather than the alternative endgame of King and Rook vs. King and Knight (KRKN).

5.1 The KQKR endgame

The KQKR endgame occurs fairly frequently relative to other 4-piece endgames, at least if one counts the number of individual *positions* encountered. It often results from a Rook endgame with Pawns, in which one side sacrifices its Rook to force promotion of its Pawn to a Queen. The ChessBase database turns up 24 games for the last 20 years, corresponding to 486 positions total (see section 6.1.3). Although this number may seem small, it must be kept in mind that many more KQKR positions will be encountered in the course of a player's analysis (in his search tree) than in the actual games.

Until the relatively recent construction of chess endgame databases, the KQKR endgame was considered a fairly easy win for the Queen in almost all positions. The noted grandmaster and endgame expert, Ludek Pachman, states in his 1977 endgame book that " ... [W]e see that it is simple enough to win this ending, provided that we know the correct method." [49]

However, about 20 years ago, the cost of computer memory and cycles had decreased so much that it had become feasible to calculate the exact number of moves required to win *for all the positions in the domain*, provided the number of pieces was 4 or less [1]. After building several such databases for 4-piece endgames, Ken Thompson decided to put one of the more

[1] The first endgame databases, for some 3-piece and 4-endings (including KQKR), date back to T. Ströhlein's effort in 1970 [71]. Recently databases for endgames with 6 pieces were constructed by L. Stiller on the Connection Machine [70].

interesting ones (KQKR) to the test against one of the best US chess players at the time. To the surprise of the chess world, grandmaster Walter Browne, many-times US champion, was incapable of winning the position given to him by Thompson in the 50 moves allotted to him. Even after a week of intensive preparation, Browne only barely managed to win his second game in precisely 50 moves (see section 6.1). Since then, many more masters and grandmasters have tried and failed to win winning positions, among them no less than the current and former world champions Garry Kasparov and Anatoly Karpov[16]. Clearly, the KQKR endgame is much more difficult than its reputation!

What is the reason for the apparent discrepancy between people's perception (an 'easy win') and the actual difficulty of winning against optimal play? The explanation must be that KQKR *is* indeed an easy win, *when played against other human chess players.* The analysis of section 6.1.3 will show that people make about the same average amount of error[2] with both the Rook and the Queen. As a result, the game is won in roughly the number of moves required with optimal play on both sides.

For the purpose of this chapter, it is important to note that KQKR, though simple and completely analyzable by computer, contains the full range of difficulty from trivial to virtually impossible for human chess players. The target subset of the KQKR domain is the intermediate region, where it may be possible to turn a near-win into a borderline draw by the use of appropriate playing strategies.

5.2 Properties of KQKR

How does the KQKR endgame appear to the chess player? In general, the endgame of KQKR is a *win*, i.e. the player with the Queen ('White') can almost always force a win, even against the best defense by the player with the Rook ('Black'). There are roughly 3 types of positions in which Black can force a draw (or even a win).

1. Black forces a win or trade of the Queen or mates White. Black can force a trade or win of the Queen by means of an "X-ray" check (check the King and then take the Queen behind it), a "fork" (simulataneous attak on King and Queen), or a "pin" (attack on the Queen, which cannot move without exposing the King).

2. Black forces a *stalemate*. With the Black King on the border, the situation may arise that the King has no flight squares left. He can then 'sacrifice' the Rook, leading to a stalemate if White accepts the offer.

3. Repetition of moves (perpetual check). This happens when the only way to avoid the stalemate draw (or the trade) is to allow continued checks by the Black Rook.

Except for beginners, no player will overlook a possibility corresponding to case 1. Case 2 is slightly more surprising, but, except for rather weak players, no chess player will actually allow the stalemate. What *does* happen is that a player overlooks this possibility in planning

[2]The average amount of error is the average number of moves lost per move with respect to optimal play. See section 6.1 for definitions.

ahead, and needs to retreat (losing valuable time) to avoid the stalemate. Case 3 is the most difficult to foresee[3], and some intermediate players (and occasionally masters) fall for this trap.

All in all, though, the defending side should not expect to be able to obtain a game-theoretical draw or win. His approach should be to stall progress as long as possible (long enough to obtain a draw according to the rule that limits the maximum number of moves allowed).

5.2.1 Some general statistics of KQKR

The array representing the endgame database contains about 2 million entries. This excludes most of the symmetrically equivalent positions (the database is stored in normalized form, allowing only squares in one octant of the board for the White Queen[4]). On the other hand, many illegal positions are stored as these are used for initialization of the database (e.g. more than one piece on one square). Each entry contains the *optimal depth* for both White to move and Black to move from the position corresponding to that entry. For White, this is the minimum number of moves needed to win against best defense by Black. This value is 1 if White can, with the best move, take the Black Rook (without stalemate) or the Black King. For Black it is one more than the largest number of moves White still needs after a Black move from the position. For example, mate is considered to have a value of 2, as White can take the Black King on the next move (which is a win in 1)[5].

The database was generated by a standard back-up procedure (retrograde analysis) as described in chapter 2. Some positions are initialized as illegal, others as immediate losses. Then a two-step procedure finds first all positions that can be won in i moves, and then all the positions which lead necessarily to wins in i or less (and are marked as a loss in $i+1$ if not already marked). The algorithm stops as soon as no new positions are marked.

To understand the structure of the KQKR domain, it is important to have an idea of how many positions require how many moves to win with optimal play on both sides. The distribution of these depths, for both Black to move (BTM) and White to move (WTM), is given in Table 5.1. [6]

---

[3]In fact, detecting this situation automatically with forward search may require more than 30 moves of search before repetition is detected! That people assess these situations so quickly is because they easily latch on to the pattern.

[4]The symmetry is 8-fold, as the value of a chess position without pawns is invariant to both rotation and reflection.

[5]This is a slightly different convention from the one used in generating the databases described in the literature, where a mate-in-1 has value 1, as does capture-in-1. The difference can be considered negligible for the purpose of this chapter.

[6]Most of the drawn positions are included under 'draw', although some (stalemate, and when the Queen is captured) are included in the table under the 'illegal' entry, because of initialisation. On the other hand, the depth-1 positions with Black to move are either illegal (King has just been taken) or no longer part of the domain (Rook has just been taken), and hence should not be considered in the analysis. Finally, WTM positions with depth 1 may be legal (can take the Rook) but may also be illegal under the standard chess rules (can take the King).

d	BTM	WTM
1	(146566)	1227933
2	43568	187658
3	16392	43368
4	15174	27579
5	17927	23019
6	16658	15928
7	14537	10652
8	10397	7902
9	8447	6272
10	7316	6098
11	7924	6917
12	9881	8324
13	12367	9678
14	13825	11109
15	15347	12300
16	18299	15242
17	23249	18027
18	27969	21138
19	34759	24306
20	41555	27990
21	50803	32647
22	61796	34226
23	70764	34969
24	76686	30852
25	78063	23991
26	71593	15758
27	56296	8835
28	35498	3886
29	18391	1098
30	6234	131
31	857	4
32	23	0
draw	1023113	11105
illegal	43854	187186
total	2096128	2096128

Table 5.1: Distribution of depths of KQKR positions, BTM and WTM. (Explanation and comments: see text).

From the table we learn the following.

- The *longest forced win* takes 31 moves (WTM).
- About 99% of legal WTM positions are wins. In other words, almost all positions are won for White, if he has the move.
- However, most of the WTM wins are immediate wins (i.e. take K or R). If we only count positions which require 2 or more moves, then 25.3% of the domain (legal positions) are *interesting* wins (482,246 positions), about twice as much as in KRKN.
- About 46.3% of the (legal) BTM positions in KQKR are lost[7].
- If we discount the positions with $d = 2$, then about 44.0% of the domain is non-trivially lost for Black (882,595 positions), about 7.5 times as much as in KRKN.
- We can also get an approximate idea about the number of positions in the domain that are *potentially difficult*, i.e. cannot be searched reliably. A rough, but probably conservative, estimate counts the number of positions for which a search of more than 5 moves (9 ply) is necessary to find the win. Some more shallow but counter-intuitive positions might be hard as well, but on the other hand some deeper but constrained or 'logical' positions might be easier. For Black to move, this leaves about 790,000 positions, for White to move about 390,000. The total amounts to about 1,180,000, about 8 times as much as in KRKN.
- We can say that the domain is generally won, even though a significant portion (with BTM) are draws. in this respect, KQKR falls perfectly within the "win" category described in chapter 3.

Let us now take a look at some of the less obvious structure of KQKR.

5.2.2 Distributions – White to move

We can represent the distributions and average values of various parameters pictorially. Consistent with the discussion of important properties in random games (see Part I), we can look at the following relevant parameters.

- the optimal distance to win (*depth*),
- the legal *branching factor*[8],
- the number of *good moves* (i.e. White moves that preserve the win, even though they may delay it significantly), and
- the number of *optimal moves* (i.e. those that are part of a shortest path to win against optimal counterplay).

---

[7] BTM losses in 1 move are initialised losses, and do not technically belong to the domain.

[8] This excludes moves which lose (or draw) immediately. When those moves (which, for example, put or leave one's own King in check), are included, the branching factor is much more nearly a constant.

The latter three parameters will also be computed as a function of the first (the depth). Based on these statistics we can predict how well a *random player*[9] would perform in different positions of the domain. Again, we shall see that heuristics and/or search are necessary in KQKR to obtain the same performance as humans.

First, consider the distribution of the depth over the positions of the domain. This distribution is given in Table 5.1, and represented graphically in Figure 5.1. The solid line represents the number of winning positions with White to move at each depth, the dashed line the number of losing positions with BTM. This is a typical 'win' distribution, including the *bottleneck* between the depths of about 7 and 15 (see Chapter 3). For the sake of resolution, the number of positions in which White can win immediately (W_1) is not represented on the figure.

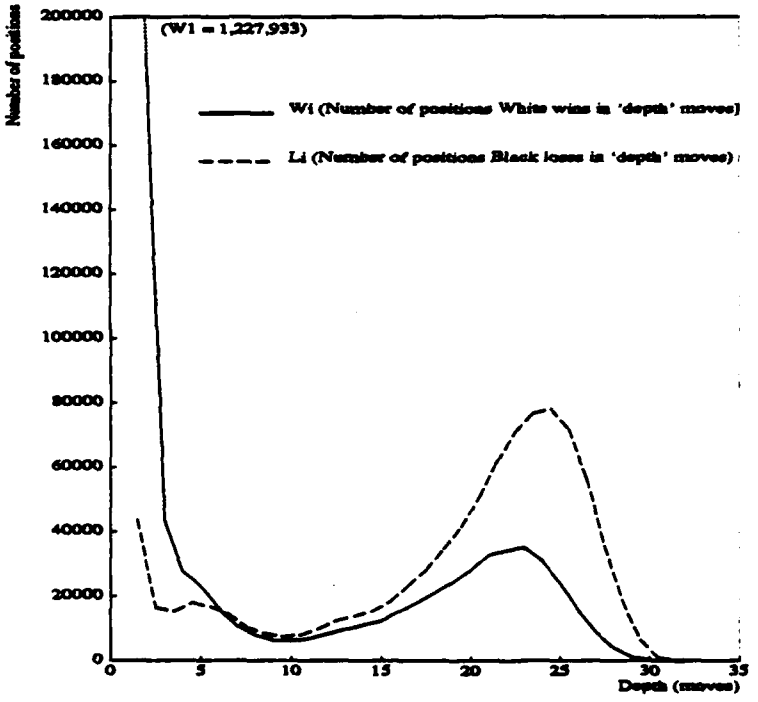

Figure 5.1: Distribution of depths, BTM and WTM, for KQKR

Figure 5.2 shows the distributions of the branching factor, the number of good moves, and the number of optimal moves for winning positions (WTM). The distribution of the branching factor gives, for each possible value in the range, the number of positions in the KQKR domain for which the branching factor has this value. The distributions of the number of good moves and of the number of optimal moves are defined similarly. In the figure we see that in most of the positions there is only one optimal move. The number of good (or win-preserving) moves is, as expected, much larger, mostly around 15–25. Finally, the branching factor peaks around 22–30 moves. Note however that several positions have

[9]With *random player* is meant a player who selects a move by a uniformly random choice among all the legal alternatives, not a player with random characteristics.

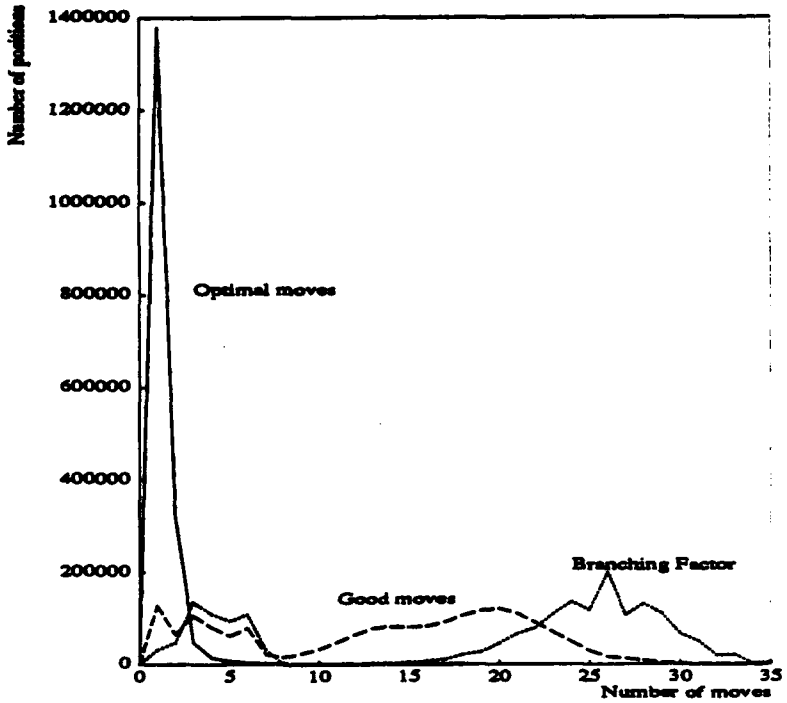

Figure 5.2: Distribution of branching factor, 'good moves' and optimal moves in KQKR (WTM).

a branching factor around 5 (White is in check).

Figure 5.3 shows how these quantities vary as a function of depth. For each depth, the graphs represent the average values of the corresponding quantity over all the positions of that depth. From the figure, we see that

- The (White) branching factor is essentially more or less constant, independent of the depth to win, except for the positions close to a win (depth ≤ 7). That the average branching factor increases closer to the win does not mean that, in individual games, we should expect White's branching factor to increase as the depth decreases. The explanation is, rather, that if White is in check he has a small probability of being close to the win. Hence, positions in which White *does* have a quick win will include fewer and fewer positions in which White is in check (small branching factor), so that the average branching factor for those depths increases. This is illustrated in Fig. 5.4, showing the joint density plot of the branching factor (BF) and the depth (D). The plot shows how many positions at each depth D have a WTM branching factor BF. The BF density given in Fig. 5.2 is the summation (over depth) of this joint density. Of course, summation in the other direction (over BF) yields again the depth distribution (WTM) of Fig. 5.1. We see that the distribution is bi-modal for all depths, the lower 'mode' corresponding to positions where White is in check (and hence has few moves), and the other mode to most of the other positions. The general shape remains about the same independent of depth, but the relative importance of low-BF positions decreases for lower depths, resulting in an increase in average value.

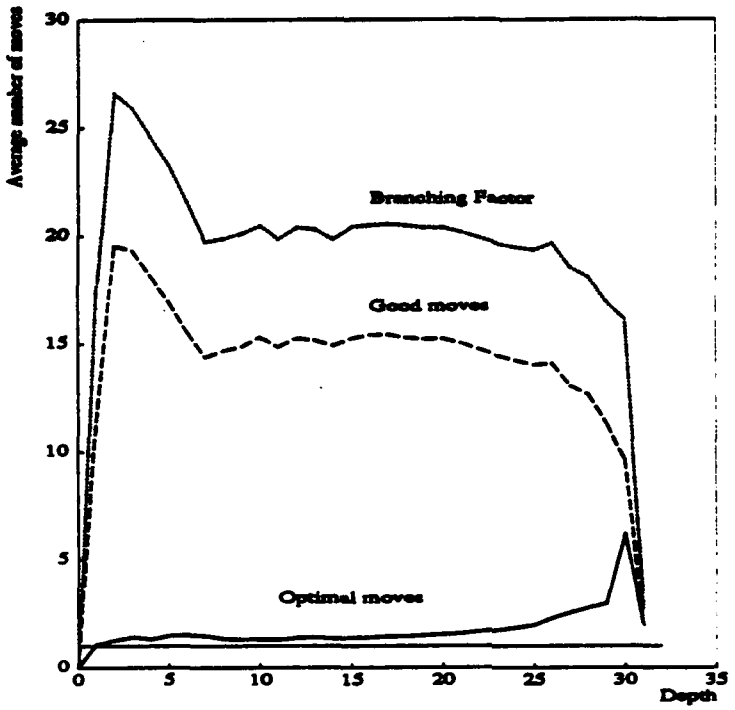

Figure 5.3: Average branching factor, good and optimal moves as a function of depth (WTM).

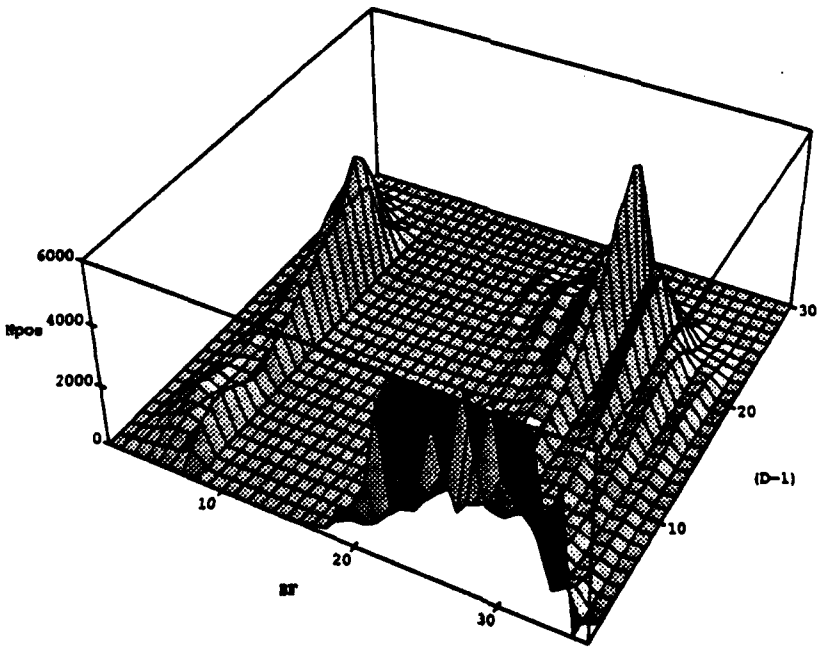

Figure 5.4: Joint distribution of Branching Factor with Depth for KQKR (WTM).

That the average branching factor is lower for the deepest positions is caused by the fact that the White King is in check in most of the (few) positions at depth 30 and 31.

- The number of *optimal moves* is also more or less constant and, as in KRKN, just slightly larger than 1, apart from perhaps a slight dip between the depths of 7 and 14 (the bottleneck!) and a strong increase at larger depths (≥ 25). This means that, independent of how deep the win is, there is usually only one optimal move (this is consistent with the predictions about random games in chapter 4) (given a maximin depth of 31).

- As can be expected for a won endgame, the number of *good* moves is much larger than the number of optimal moves. That it is not equal to the branching factor is due to the fact that this endgame contains (terminal) draws. The difference is an approximately constant 6–8, suggesting that White has on the average about 6–8 moves that would throw away the win (for example by putting the Queen *en prise*.)

5.2.3 Distributions – Black to move

We can compute the same statistics with the Rook player ('Black') to move. The distribution of positions over the number of moves to loss was given earlier in Fig. 5.1. The distribution is quite consistent with what would be expected based on the the analysis of random games (chapter 4). There are much fewer BTM than WTM positions for small depths, the bottleneck occurs at about the same depth, and the second maximum is more pronounced and occurs at a slightly larger depth than for the WTM positions.

By definition, in a losing position all moves lose, so the concept of 'good' move has lost its meaning. But instead, we can consider the number of *long lines*, defined (more or less arbitrarily) as the number of moves after which White still needs at least 4 moves to win (i.e. mate in 3 or capture of the Rook in 4). As most human players are (presumably) capable of finding the win in positions closer to the win than 4 moves, this number gives an upper bound on the range on Black's choice of reasonable moves. Figure 5.5, then, shows the distribution of the branching factor, the number of long lines, and the number of optimal moves for losing positions (BTM). As with White to move, there is usually only one optimal move. The number of long lines peaks around 5. Note that the distribution has a value ≥ 0 for $i = 0$, since there are positions from which there are no long lines (namely those which are closer than 4 moves to the win). Again, the distribution of the branching factor is bimodal, with peaks around 5 (in check) and 18 (otherwise).

Figure 5.6 shows the distribution of the branching factor, the number of long lines, and the number of optimal moves, as a function of depth. In the figure we see that there is only one optimal move for most depths, although this number comes closer to two for deep positions (25–30). The number of *long lines* is (by definition) equal to the number of optimal moves for $D = 5$, and increasingly larger for larger depths. Interestingly, the *branching factor* is strongly correlated with depth for $D \leq 10$, and more or less constant when D is larger. In the section on heuristics we shall see that this corresponds well with the efficiency of the *check* heuristic. Intuitively, as for smaller depths a larger percentage of WTM positions

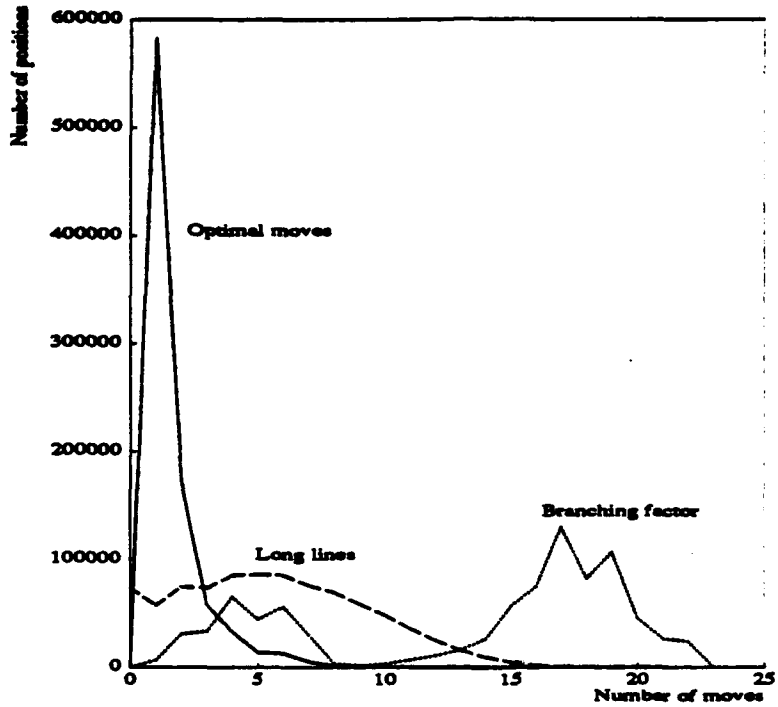

Figure 5.5: Distribution of branching factor, long lines and optimal moves in KQKR (BTM).

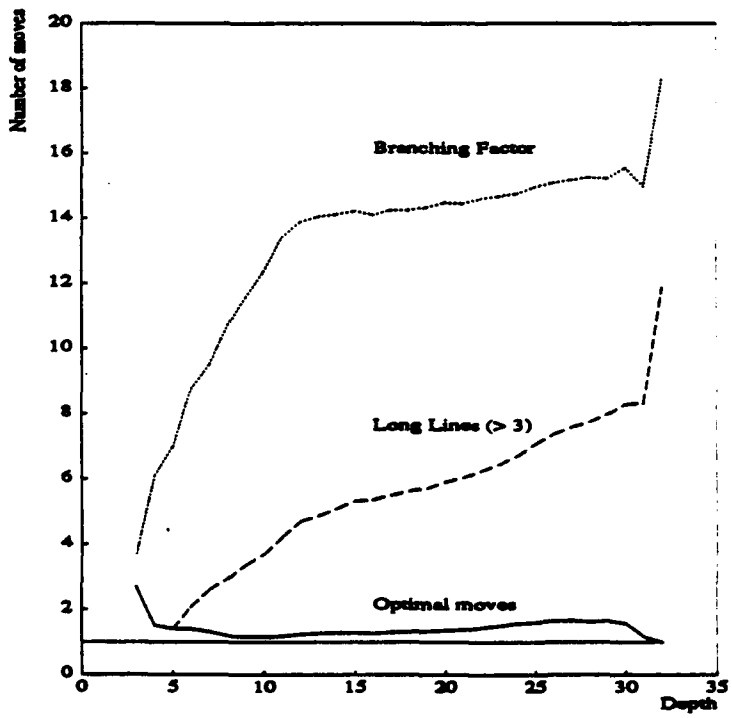

Figure 5.6: Average branching factor, number of 'long lines' and number of optimal moves as a function of depth (BTM).

requires a check for optimality, relatively more BTM positions at these depths must have the Black King in check. This decreases the average (legal) Black branching factor.

Another observation we can make is that the number of optimal moves is close to one for depths between 7 and 12, but comes closer to two for depths of 25–30. In addition, Black's flexibility (number of long lines) increases as well. This leads to the expectation that Black will have more leeway in selecting a strategy for larger depths (25–30), and that it is in that part of the domain that a program could make the most use of any extra knowledge. On the other hand, we saw in Figure 5.2 that White too has an easier time in the 25–30 move region. Furthermore, the expected punishment for a mistake (magnitude of error) becomes larger for shallower depths (see next section). It is therefore to be expected that the critical region for performance comparison between different Black players will be somewhere in between, say for $15 \leq D \leq 20$. In this 'region of interest', Black has a choice among at most 5–6 reasonable moves, on the average.

5.3 Elements of difficulty in KQKR

What makes the KQKR endgame, simple and small though it seems, so difficult that even the best chess players in the world cannot avoid mistakes?

First, there is the (legal) **branching factor**. For White, this is about 20–25 on the average for most depths; for Black, about 14–16. This makes exhaustive search impossible in most cases, even when it is possible to exclude several trivial lines (for Black there are still some 4–8 long lines, for White some 15–20 win-preserving moves).

The second relevant parameter is the **maximin depth**, i.e. the depth of the longest optimal line in the endgame. For KQKR this is 31, clearly too deep to allow searching, even if we could restrict ourselves to optimal moves only[10]. The maximin depth is not necessarily a reliable measure of the average difficulty of an endgame (it only applies to one position or a small set of positions). The distribution (as in Figure 5.1 or Table 5.1) provides a better indication of how many positions are deep. For example, from the table we can compute that more than half of the 'interesting positions' (i.e. depth $D \geq 2$) are 19 moves deep or more.

As expected from the analysis of random games (chapter 4), the distribution exhibits a **bottleneck**[11] — here between depths of 7–12 moves. In a purely random graph, we could expect the number of optimal moves to be the smallest in this bottleneck (which is true, though the effect seems small). Once past the bottleneck, the probability of making an optimal move increases. It would appear, then, that it is this region which is most critical. Another characteristic of the bottleneck region is that the positions in it occur on many optimal paths to the win (viz. from all the nodes of higher depths). In terms of mastering

[10]KQKR is the 'deepest' endgame in this sense among 4-piece endgames. Of course, many 5-piece and 6-piece endgames are much more difficult in this respect.

[11]Recall that the "bottleneck" is the region between the two maxima of the distribution for which the position density is the smallest. This region is important, since all optimal paths starting from deeper positions must necessarily pass through the much smaller number of positions in the bottleneck region.

the KQKR endgame, it is useful to learn these frequently relevant positions first. In fact, many positions given in endgame books are precisely from this region.

Another aspect of the distribution is that, statistically, **the expected depth after a random move is nearly a constant for low depths** (see Fig. 5.9 below). A 'random mistake' will therefore tend to be much more costly when D is small. For example, whereas it takes about three mistakes at depth 20 to reach a drawn position according to the 50-move rule, one single mistake from a position at depth 5 (after some 25 moves of play) is enough.

How big is such a 'random mistake' in KQKR ? The plot of Fig. 5.7 shows the average distribution, over all positions of a given depth (WTM), of the depth (then BTM) of a position after a random White move. For each depth with WTM, the distribution starts with a peak (relatively larger or smaller depending on depth) at the optimal depth. There are, of course, no positions with lower values, as no move can win faster than an optimal move, by definition.

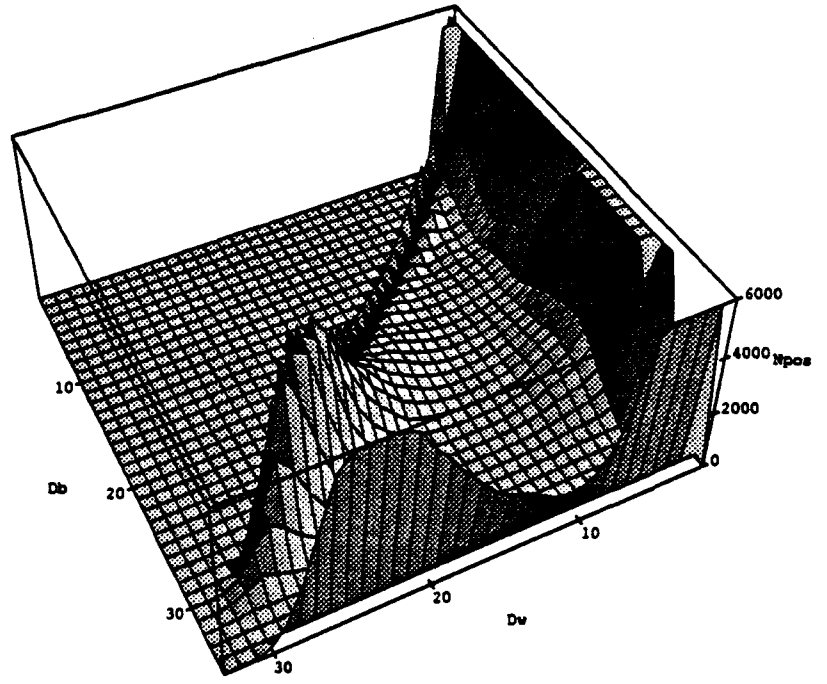

Figure 5.7: Joint distribution of Branching Factor with Depth for KQKR (WTM).

A cross section of this graph where the initial depth D_W is kept constant represents the distribution of depths after a random move from such a position. Such a cross section, for $D_W = 16$, is given in Fig. 5.8. The initial peak represents optimal moves, the middle part of the distribution all the possible win-preserving moves ('good moves'), and the final peak the fatal errors, leading to a drawn game. A simplified analysis assumes that moves will reach all positions with 'legal' values with the same probability (i.e. no values lower than D_W). Moves can therefore be expected to be distributed with the same density as the general BTM distribution, conditioned on $D_B >= D_W$, i.e. only the part for values larger than D_W, suitably normalized. There must also be a proportion of at least $1/b$ optimal moves (as we need at least one optimal move from each winnig position). This proportion is in general larger than what we could expect from a random distribution, resulting in the peak.

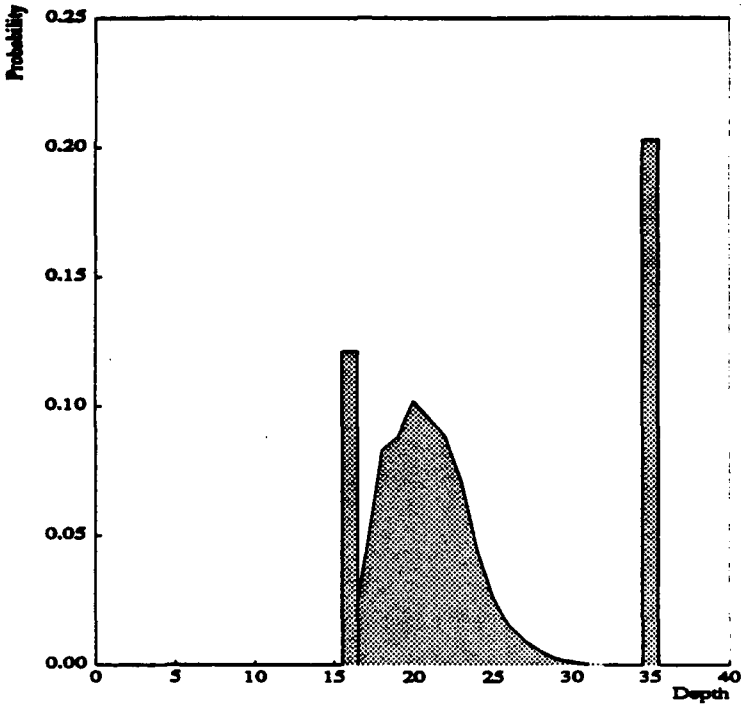

Figure 5.8: Average distribution of values for a random move, $D_W = 16$. $D_B = 35$ represents game-theoretical drawing moves.

To discover the expected magnitude of a mistake at each depth, we can compute the average of the density given in Fig. 5.7 for each depth. This average, ignoring fatal mistakes[12], is given in Fig. 5.9. We see that, starting from the right, the expected depth remains at first fairly close to the optimal depth, i.e. the expected mistake (which is just the difference between the two curves) remains fairly small, though increasing with decreasing D_W. Below a depth of $D_W \approx 12$, however, the expected depth remains more or less constant, resulting in a much stronger increase of expected error. From the foregoing, we can conclude that *the random player*, even if he is prevented from making fatal mistakes, *cannot win a won KQKR position*. To reduce the large expected error at small D_W, *search* will be needed. When D_W is larger, *heuristics* can improve the probability of selecting an optimal move, or at least to reduce the expected magnitude of error. We shall be able to compare this with the actual performance of human players in section 6.1.3.

As in KRKN, the **irregularity of the domain** makes learning by humans difficult (no easy compression into rules, or *chunks* is possible). Although the patterns (for winning positions) are more regular than corresponding patterns for KRKN, they can still not always be easily explained. An example is given in Fig. 5.10, called a 'depth chart' by Roycroft

[12]Fatal mistakes are hard to quantify (in other parts of this chapter, numerical values in between 40 and 100 have been used). Ignoring them altogether leads to an underestimate of the average error, but corresponds well with actual human play, as master level players rarely make fatal mistakes at all.

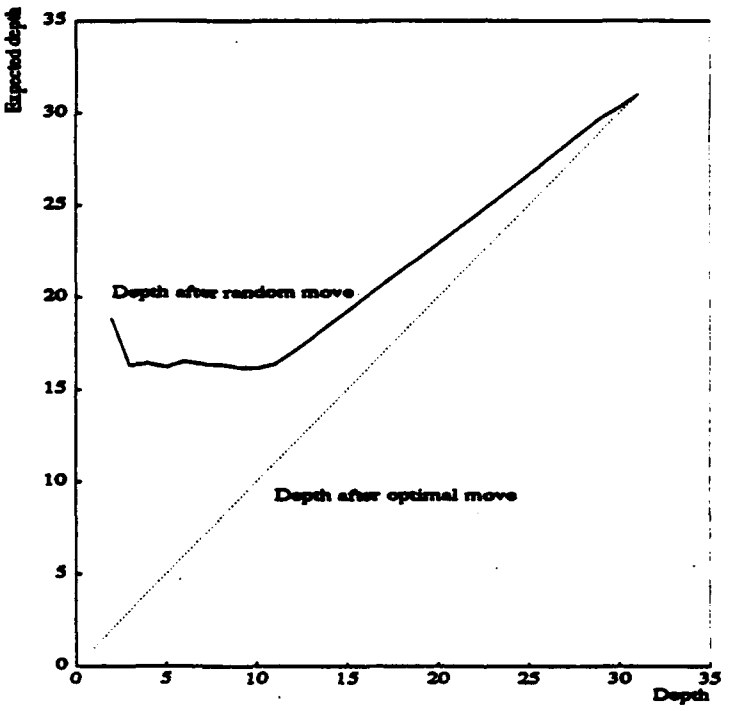

Figure 5.9: Expected depth (D_B) after a random move from a position with depth D_W (compared with the depth of an optimal move).

and Beal[55][13]. In it, on each square is indicated the value (number of moves to win) of the position with Black to move if the White Queen were placed on that square. The absence of a number indicates a drawn (or lost, or illegal) position. Although some patterns are

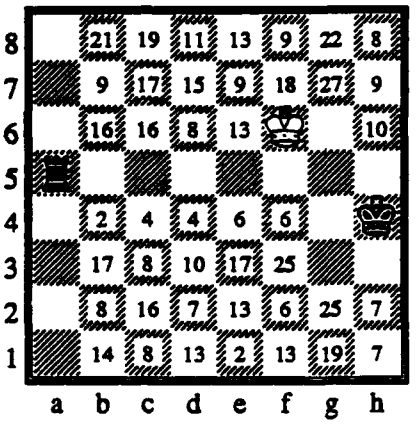

Figure 5.10: Example position for pattern irregularities.

clearly visible, such as the fact that the depth is much lower in general when Black is in check, most of the values seem to have a fairly random character. Some of the variations can be explained concisely (such as the difficulty of avoiding Rook checks with the Queen on g7), but for many there does not seem to be an explanation short of presentation of a large portion of the search tree.

Finally, as we shall see in section 6.2, some heuristics commonly used by people are actually good **only for certain values of the depth.** Using such heuristics may be beneficial in the appropriate part of the domain, but may lead to mistakes in other positions.

The reason that the KQKR endgame is hard, even for chess masters, is a combination of the above factors. Since many of these factors are properties of a general statistical nature, it is reasonable to assume that a similar analysis applies to other games as well.

5.4 Comparison of the KQKR and KRKN domains

In the work leading to this thesis, both KQKR and KRKN were used as test domains. Both are small, allowing for construction of a perfect endgame database, and 'clean' in the sense that the branching factor is (or can be made to be) fairly constant. Both domains thus correspond well with the random game model developed in chapter 3.

According to the classification developed in chapter 3, KQKR is just a 'win' (but not by much), and KRKN is just a 'draw'. Because both endgames are so close to the critical threshold (T_α: see Chapter 4), their maximin depth is large (31 and 27 respectively), and both are therefore relatively interesting domains. Although it would seem that either would

[13]Developed independently by the author of this thesis some 4 years ago.

be an equally good choice for experiments with human subjects, this is not entirely true (see below). Furthermore, it is not obvious that a program that has access to the database should necessarily follow the same approach in both cases. Indeed, whereas the losing side in KRKN mainly tries to get the winning side to make a *fatal mistake* to obtain a draw[14], this will in general not be possible in KQKR (as almost all positions are theoretical wins). In KQKR, however, the goal will be to delay the win as much as possible, with the aim of making the opponent exceed the total allotted number of moves. The general strategies for both cases may be similar, but this is not *a priori* clear.

The differences in KQKR and KRKN have a large impact in the way the experiments should be run, as well as in the information that can be expected to be obtained from them. It would seem that it has certain significant advantages to use KQKR rather than KRKN.

1. There are more winning positions in KQKR than in KRKN (as almost all positions are wins). This indicates a larger variety of positions, features and strategies, as well as more different levels of difficulty.

2. There are more choice points for the defender (i.e. positions where the defender has a choice between several equi-optimal, or near equi-optimal moves). This gives the defender more possibilities to select a more difficult alternative.

3. Human subjects are more easaily motivated to play KQKR than KRKN. Indeed, it is possible to play *'games'* against a database program that approximate play in tournament conditions, and the subject never, or at least infrequently, loses the possibility to win the game eventually, even after a mistake[15]. For KRKN, the same experimental setup (playing games) makes for many short games and frequent interruptions (most often to provide negative feedback) by the experimenter. This is the main reason the KRKN experiments (not further described in this thesis) were presented as a 60-position 'test', in which subjects had to indicate their preferred move, rather than in the form of games.

As a result, it was possible to obtain about three times more data[16] on KQKR than KRKN. On the minus side, most of the positions for which the subjects had to decide on a move were different from each other (all the games were different!), making a comparison between subjects and/or positions inherently more difficult. This is because different starting positions were chosen to avoid learning effects[17], and because games played by different subjects quickly diverged from each other, even from the same starting positions.

A deeper comparison of KQKR and KRKN along several dimensions would be interesting, but falls outside the scope of this thesis.

---

[14]Of course, stalling is a valid objective in KRKN as well, but there it is usually secondary to trying to obtain a game-theoretic draw.

[15]Some subjects find it depressing that they never (or hardly ever) manage to win, however, and in the long run motivation is a serious problem.

[16]One 'unit' of data is here one position with the choice of move by one subject

[17]Only in the experiments with subject C2 (ChipTest) was it possible to use the same starting position multiple times.

Chapter 6

Human play in KQKR

This chapter discusses some issues related to how human chess players play the KQKR endgame: when, how often, and how badly do people make mistakes? When do the heuristics they use lead to non-optimal play? This information will be considered in the light of the next chapter, which explores its usefulness in KQKR play.

In section 6.1 tools are proposed for analysing the quality of games by non-optimal players, based on complete knowledge of the domain (by the evaluator). Some of these measures — the probability of error, the expected magnitude of error, and the expected error overall — are then used to analyze the performance of human chess masters in a collection of games, and of human analysts in endgame book analysis.

When humans play the KQKR endgame, they make use of certain heuristics to guide their planning and choice of moves. Section 6.2 discusses some plausible heuristics, how beneficial it is to use them, as a function of the depth of a position, compared to a random move choice, and whether they can be used to predict moves of human chess players.

6.1 Measuring the quality of games in KQKR. A study of master level play.

In this section, various ways are discussed of measuring the quality of play in games like the KQKR endgame, in which the exact distance to win is known in all positions. These criteria are then applied to examples from play, positions from endgame books and a set of 24 master games. Finally, in a further analysis of this set of master games, some of the quality criteria are computed as a function of depth, and the results are compared with the same parameters for a 'random player'.

Recall that in KQKR fatal mistakes occur only infrequently in human play. In the following discussion, we shall therefore restrict ourselves to non-fatal mistakes (inaccuracies).

6.1.1 Measuring the quality of play in KQKR

There are various ways one can try to define the notion of *quality of play*, based on the mistakes one detects in the course of a certain player's games. In general (say, for the complete game of chess) the issue is complicated by the fact that mistakes and inaccuracies cannot reliably be detected and evaluated because a complete analysis is not feasible. But when the game is small enough, and an analysis can be based on perfect knowledge, it is possible to quantify exactly how well a player's performance matches optimal play.

The space of possible games

Basically, a quality function imposes an *ordering on the space of possible games*, so let us first discuss the characteristics of this space. If we ignore the possibility of a game-theoretic draw or loss (for instance as a result of a blunder) and a draw by threefold repetition, a game ends either (1) because the player of the Queen wins, or (2) because he exceeds the total number of moves allowed. Fig. 6.1 plots the (optimal) depth of the game position as a function of the number of moves played, where the initial position has depth D_0. The limit on the number of moves is given by the common 50-move rule. Each game traces some *trajectory* (path) in the game space, which is delimited, for the current example, by the polygon $< (0, D_0), (1, 31), (A, 31), (50, 0), (D_0, 0) >$. The game can be considered to end as soon as the trajectory crosses one of its boundaries. Obviously if it hits the line $< (50, 0), (D0, 0) >$ the game has ended in a win, which terminates the game. The second possibility is when the line $< (A, 31), (50, 0) >$ is crossed. From that point it is no longer possible to win in 50 moves or less, even with optimal play.

In the figure, several possible games are sketched as the trajectories D_0–A, D_0–B, ..., D_0–G (in reality, all the trajectories are of course discrete paths). Region **X** is a stylized representation of a collection of paths all ending up in point A, and which cannot be distinguished from each other based on the end points D_0 and A. Region **Y** is a region in which no game trajectories can fall when the Rook player plays optimally (by definition, the most progress that can be made against optimal play is 1 move per move). The fastest possible win, against optimal play, is a game with trajectory D_0–G.

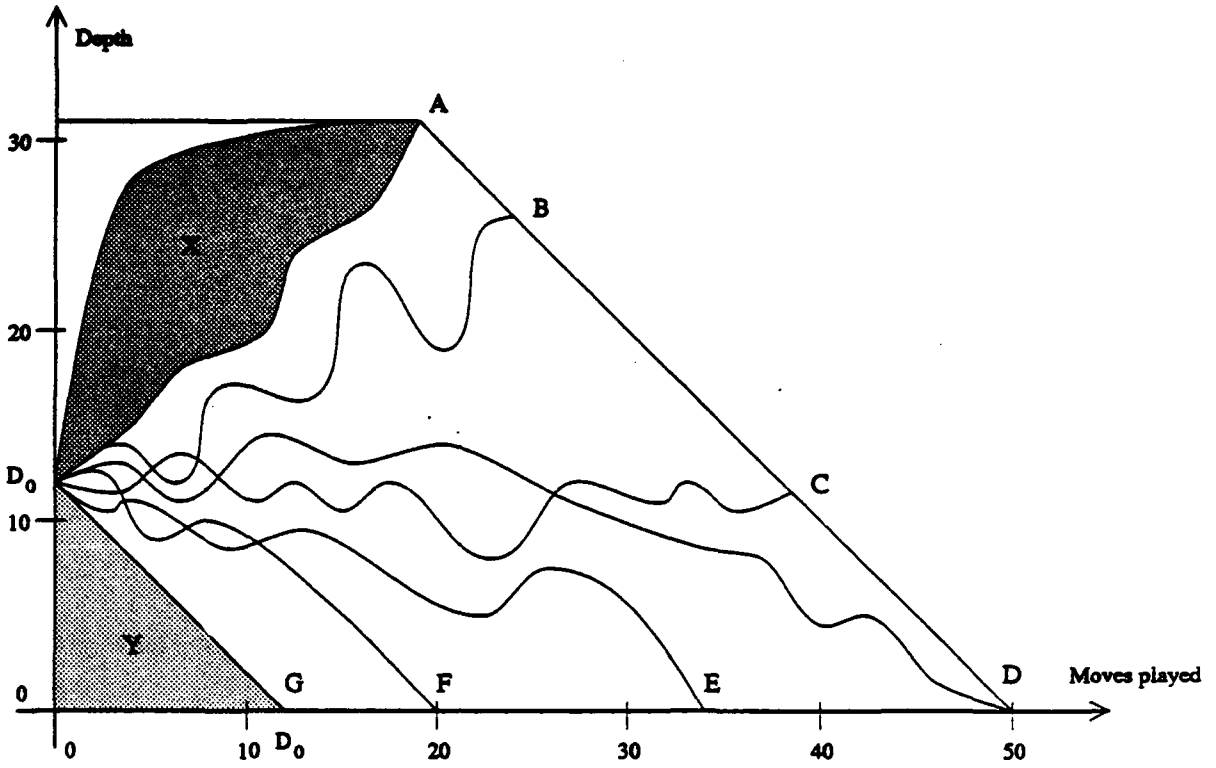

Figure 6.1: Space of possible games in KQKR, for given initial depth D_0, and a maximum of 50 moves allowed: depth as a function of number of moves played. A ... G represent possible games.

Of course, by only looking at the value (depth) of a position, some structure is lost, and many different actual games may correspond to any given trajectory. In particular, one game could be considerably more difficult than another game but still have the same depth trajectory. There is no way to tell such games apart. Furthermore, between any two points there could be many different trajectories. A measure of quality that restricts itself to the endpoints (D_0 and the boundary crossing) loses in addition any information on how many mistakes were made, and their magnitude (this is why it is useful to consider other measures as well). Also, the structure of the game itself may prevent discrimination at the worst end of the scale, as the maximum depth of the game is limited. In particular, many paths could lead to point A (roughly indicated by region **X**), and there is no way to compare them. All in all, however, 'endpoint' criteria provide a useful summary of how well a game compares with optimal play, and many of the quality measures in the following section are of this type.

Quality of games based on their trajectories

How well did White play (assuming optimal play by Black)? As a first approximation, the quality of a game can be derived from the *boundary points of the game trajectory*. If White wins, his quality of play is assumed to be better if he *wins faster*[1]. If White draws (or loses), we could say that the quality of play is better if he *keeps his winning chances longer*. In other words, the sooner the game crosses the drawing boundary (can no longer be won), the

[1] A good case can be made that this need not always be true for an imperfect player, however[61].

worse the play is. Hence, in Fig. 6.1, the possible endpoints A through G can be ordered according to quality as $G > F > E > D > C > B > A$.

One way of rating game quality then, is to simply take the *move number* at which the boundary is crossed, and subtracting it from 101 moves if the game ends up being a draw (this way the scale is continuous, as draws yield values from 51 through 81, depending on how long the player was able to remain in the winning region). Let $M(g)$ be the number of moves played in a game g, and $Q_1(g)$ its quality according to this metric. Further, call a game that ends in a win g_w and a game that ends in a draw g_d. Finally, for the sake of generality, call the maximum number of moves allowed N and the maximin depth of the domain D_{max} (in the concrete case under consideration, these are $N = 50$, and $D_{max} = 31$ respectively). Then we have

$$\begin{aligned} Q_1(g_w) &= M \\ Q_1(g_d) &= 2N + 1 - M \end{aligned}$$

For this function, the best games are placed at the lowest end of the scale, so it is rather a measure of badness than of quality. But the metric Q_1 has a few more fundamental problems: (1) it does not compare games of different starting depths naturally (e.g. optimally played games from different initial positions receive different values), (2) it does not extend to games that are interrupted before (or after) they reach the boundary, (3) it does not take into account the number and magnitude of any mistakes made, and (4) it only gives a relative measure of quality, namely the *difference* of the playing quality of both players. The following modifications will try to remedy one or more of these problems. Note, however, that there may be situations where Q_1, simple as it is, will perform adequately.

The first objection can be taken care of in part by assigning the same value (e.g. 0) to optimal play from any starting depth, and counting from there. This leads to a second quality measure, Q_2:

$$Q_2(g) = D_0 - Q_1(g)$$

(or any equivalent measure obtained by adding a suitable constant). Metric Q_2 yields 0 for optimal play, and negative values for non-optimal games. The value can be positive only in the case of mistakes by the Rook player.

Now, as Q_2 corresponds well to (minus) the *total amount of error* committed by the player (in fact, it is exactly that for wins, and it is a lower bound for drawn games), we can almost equivalently use this total error as a metric. The advantage of using the total error, however, is that it can be computed for games of any length, independently of the outcome (except for when the player makes a fatal mistake). Again, with M the total number of moves made and D_M the depth at that point,

$$Q_3(g) = M + D_M - D_0.$$

Note that, in the case of a win, $D_M = 0$, and Q_3 corresponds exactly with Q_1 (with changed sign). However, Q_3 is not a good discriminator for drawn games played out to full length. Indeed for a draw, we have that $D_M \gtrsim N - M$, and therefore $Q_3(g) \gtrsim N - D_0$, which depends only on the initial depth.

To solve this problem, we can *normalize* the error with respect to the number of moves made. This yields

$$Q_4(g) = \frac{M + D_M - D_0}{M} = 1 - \frac{D_0 - D_M}{M}.$$

Measure Q_4 gives the *net average error per move*, and will be used frequently in the remainder of this chapter.

Equivalent to this, of course, is the *average progress per move*, which is simply

$$Q_5(g) = \frac{D_0 - D_M}{M}.$$

With optimal play, measure Q_5 has a value of 1. All other games have a lower value, possibly negative. This measure has been used in the literature (and was called "path efficiency" by Michie and Bratko[41]) in the special case that the game is carried out all the way to the win ($D_M = 0$), and Black provided optimal counterplay.

If Black does not always play optimally, we can split up the above metric(s) on a move-by move basis. Let $D_{W,i}$ be the depth in a WTM position after Black's ith move, and $D_{B,i}$ the depth in a BTM position after White's ith move. Then the D_i used in the above expressions corresponds exactly with $D_{W,i}$. A White move leads from a position with depth $D_{W,(i-1)}$ to a position with depth $D_{B,i}$, and a Black move from a position with depth $D_{B,i}$ to a position with depth $D_{W,i}$. With optimal play, we have[2] $D_{B,i} = D_{W,(i-1)}$ and $D_{W,i} = D_{B,i} - 1$. If White makes a mistake (inaccuracy) then $D_{B,i} > D_{W,(i-1)}$, and the magnitude of his error at move i, $\varepsilon_{W,i}$, can be expressed as the difference:

$$\varepsilon_{W,i} = D_{B,i} - D_{W,(i-1)}.$$

Similarly, if Black makes a mistake then $D_{W,i} < D_{B,i} - 1$, and the (magnitude of the) error can be expressed as the difference:

$$\varepsilon_{B,(i-1)} = D_{B,i} - 1 - D_{W,i}.$$

The *average error* for White is then

$$Q_6(g) = \frac{\sum_{i=1}^{M} \varepsilon_{W,i}}{M}.$$

Measure Q_6 is similar to what Althöfer called this the *normalized noise* of a game[3], except that Q_6 only looks at the performance of one player (and not both combined). It provides us with an easily interpretable and computable metric to judge the quality of games. When Black plays optimally, Q_6 reduces to Q_5. Otherwise, Q_6 still expresses White's quality of play, whereas Q_5 measures the *difference* in quality between the two players.

The last two measures discussed provide an important summary of the overall course of a game. Q_5 gives an indication of whether one player can make progress against another player (and can, or cannot, eventually win the game). Q_6 gives an absolute measure of quality for

[2]This is actually only one possible convention, which was adopted here because it is the same convention used in the construction of the KQKR database.

one player (White), and can be used to judge, either the difficulty of the positions in a specific game (if the player's strength is supposed to be constant), or the change of a player's playing strength over time. Hence both measures can be equally useful in assessing games between two players. In the analysis of a set of master games (section 6.1.3), and in the analysis of the experiments (section 7), Q_6 will be called d, and Q_5 "*net*".

Interesting related criteria are the probability and expected magnitude of mistakes. Using the above terminology, the probability of error p can be expressed as

$$p = Q_7(g) = \frac{\sum_{i=1}^{M} \text{sign}(\varepsilon_{w,i})}{M}$$

and the expected magnitude (i.e. total error divided by number of mistakes) as

$$\delta = Q_8(g) = \frac{\sum_{i=1}^{M} \varepsilon_{w,i}}{\sum_{i=1}^{M} \text{sign}(\varepsilon_{w,i})} = \frac{Q_7(g)}{Q_6(g)}$$

As we saw, both p and δ play a role in whether Black can gain a benefit from occasional mistakes to reach more difficult positions (see section 1.4).

Remarks

The range of most of the quality measures discussed above is dependent on both the *length* of a game, and on the *initial depth*. Indeed, shorter games allow for a larger average error, as do games from an initial position with a smaller depth. Ideally, then, quality comparisons should be made only between games of equal length, starting from the same initial depth. This is however not easy to stipulate in the case of a relatively small set of actually played games (see section 6.1.3), and turned out to be hard or unnatural to enforce in the experiments described in section 7.

It is now also possible to give a more concrete meaning to the concept of *difficulty*. What is intuitively understood by the "difficulty" of a certain position or class of positions for a certain player or class of players, can be appropriately translated as *the probability that such a player cannot win such a position within the allotted number of moves*. If the quality criteria discussed above are considered to be intrinsic properties of a player, one can compute, if not actual values for difficulty, at least a set of positions which would on the average require N moves for a given player. This set can be considered the *boundary* between hard and easy positions for that player. Borderline wins last 50 moves (N moves), or $D_N = 0$. Hence, the allowable average error is $d = Q_4(g) = 1 - \frac{D_0}{N}$. If we know d, then borderline positions will have depth $D_0 = (1 - d)N$ (provided that difficulty is assumed to be only dependent on depth).

Alternatively, this difficulty measure partitions the domain in regions of 'minimal required playing strength'. If the error value d characterizes a player, then a 'playing strength' of $d = 1 - \frac{D_0}{N}$ or better (i.e. less) is required for a position of initial depth D_0 (given the same assumption as before).

Finally, assuming a player and the initial depth to be constants, we also can judge the *performance of a program that plays the Rook* by the quality of play of the (human)

opponent, i.e. by his apparent playing strength, or, equivalently the apparent difficulty of the positions in the course of the game. This is essentially the criterion used in the analysis of the experiments (section 7).

6.1.2 Using the quality criteria on human play and analysis of KQKR

The Browne games

In 1977, Ken Thompson challenged grandmaster Walter Browne to a game against his computer chess program Belle, starting from a position in the KQKR domain[35, 73]. As Belle could load the KQKR endgame database, it played only optimal moves. Browne was given the White side (Queen) in a maximin position (one of the positions requiring the largest number of moves to win – 31 – against optimal play). He was allowed a maximum of 50 moves and 2 hours of thinking time to actually win the game. Browne was not able to win the first game (he might have won in 54 moves, had he continued the game optimally from the final position on move 45). After one week of "intensive preparation", Browne tried again (from virtually the same position, but rotated), and this time succeeded in winning in exactly 50 moves. Figure 6.2 shows, for both games, the depth of each position as a function of the number of moves made by Browne. The dotted lines delimit the game space as explained with Fig. 6.1. Note that game 1 was continued beyond move 33, even though it could then no longer be won in less than 50 moves total. The figure shows little improvement

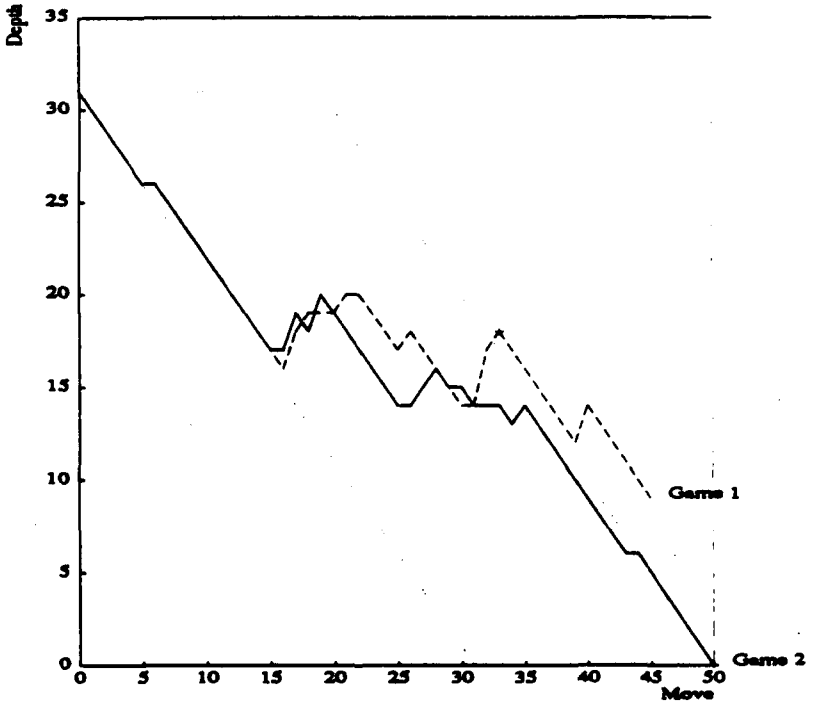

Figure 6.2: Depth as a function of number of moves played: Walter Browne in 2 games from KQKR maximin positions. The dotted line delimits the game space.

between games 1 and 2: until move 31 both games run a similar course. Also, from the game score for game 2, one gets the impression that Browne was perhaps slightly lucky in being able to bring about a repeating, and here also optimal, pattern, which led him straight down to the win.

Let us analyze the two games in terms of the criteria described above, in particular the average error, the probability of error and the error magnitude.

- In game 1, Browne made 45 moves, 12 of which were mistakes, with a total error of 23 moves. This means his error probability was $p = 0.27$, his error magnitude $\delta = 1.92$ moves per mistake, and his average error is $d = 0.51$ moves per move. If we only count the path within the game space, the errors are larger (as expected for a shorter game). He made 33 moves, 11 of which were mistakes, with a total error of 20. This means his error probability was $p = 0.33$, his error magnitude $\delta = 1.82$ moves per mistake, and his average error is $d = 0.61$ moves per move.

- In game 2, Browne made 50 moves, 12 of which were mistakes, with a total error of 19 moves. This means his error probability was $p = 0.24$, his error magnitude $\delta = 1.58$ moves per mistake, and his average error $d = 0.38$ moves per move.

The comparison would tend to indicate that, by his preparation, Browne was able to slightly reduce both the average magnitude δ and his probability of error p. However, the shape of the trajectory for both games is similar and seems random enough between moves 15–35 to cast some doubts on how much progress was really made between the times of the two games.

In any case, the plots show that most of the mistakes occurred within a 'critical region' of positions with depths from 14 - 20, and only a handful for deeper or shallower positions. This confirms the estimate in chapter 5 of the region most likely to cause difficulty for humans. The low end of this region (14) may indicate that Browne is capable of deeper searches than the average master player in the set of games considered below.

In comparison with the subjects used for the experiments in this thesis (see section 7), Browne's error probability is rather low (one every four moves). Apart from Browne being the better chess player, an explanation can be found in that the positions used in the experiments were positions in the critical range, and hence 'more difficult'. For instance, if Browne were considered to have started from the position after move 15, his error probability would be around $p = 0.3$ for both games.

Endgame book analysis

The measures discussed in the previous subsection were also used to evaluate the analysis given with most of the KQKR positions in the endgame books by Hooper[22] and Pachman[49].

Most of the shallow positions (depth less than 10) were analysed correctly, except for occasional small mistake. The lines given for the deeper positions[3] were rather flawed,

[3]Only two positions were deeper than about 12 moves: a position by Grigoriev and one by Euwe.

however, and the average error d was about 0.5 for both sides (for the length of the variations given).

Apparently, people are quite capable of analyzing KQKR up till a depth of about 10, given sufficient time, but not deeper, or at least not without serious error. We shall see that this corresponds well to the results of the analysis of errors as a function of depth in section 6.1.3.

Other human players against the database

In a few preliminary trials (in preparation of the experiments discussed in section 7), two master level chess players played the KQKR database, without prior study or training. The starting positions were randomly selected positions, 12 moves deep. The quality was computed when the game could no longer be won. Both players averaged an error of $d = 1.5$, i.e. negative progress of about 0.5 move every move.

Let us now turn to a more deeper analysis of a set of master games, which provide us with a more significant dataset on top-level human play.

6.1.3 An analysis of a set of 24 master games

An analysis was made, by means of the KQKR database, of a set of 24 master and grandmaster games that ended in the endgame of King and Queen vs. King and Rook[4]. All the games entered the KQKR domain as wins for White [5], and eventually duly ended in Black's resignation.

First, we look at the overall quality of play, then at how quality depends on the depth of a position.

Overall analysis

The 24 games consist of a total of 245 White moves and 241 Black moves from winning or losing positions, respectively. Neither player made *fatal mistakes* at any time. In particular, the Queen player never allowed his opponent to force a draw. However, both sides committed numerous *inaccuracies.*

- The player of the Queen made 79 errors (an error rate $p = 0.322$), totalling a loss of 177 moves. The error magnitude was therefore $\delta = 2.24$ moves per mistake, or an average of 0.72 moves per move.

---

[4]My thanks go to Murray Campbell for extracting this set of games from the DT2 project's ChessBase database, containing at that time about 100,000 master games.

[5]We keep here the convention that the side with the Queen is called 'White' and the side with the Rook 'Black', even though in the actual games the colors may have been reversed.

- The player of the Rook made 67 errors (an error rate of 0.278), totalling a loss of 200 moves. The error magnitude was therefore $\delta = 2.99$ moves per mistake, or an average of 0.83 moves per move.

As has been pointed out several times already, these results show that the KQKR endgame is certainly difficult. For instance, against optimal play, White could be expected to take about *four times as long as necessary* to win positions similar to the ones in the set studied (an error of 0.75 means a progress of 0.25 moves/move, i.e. one fourth of the optimal progress). In other words, one could expect the 'average human master' to only be capable to win positions 14 moves deep or less within the 50 moves allowed. Deeper positions would be 'too difficult'.

However, the endgame is hard for the Rook too, who would hold out only about half as long as possible against optimal play (as the Rook drops 0.83 moves per move, the total progress is 1.83 moves/move). As the errors more or less cancel each other out, the net result is that the Q wins slightly faster than with optimal play by both. This is probably what led to the belief (before the advent of endgame databases) that KQKR was an easy win for the Queen.

As far as fatal mistakes are concerned, although a sizeable proportion of moves lead to a draw, the draws themselves appear to be easy to recognize and are in general not very deep. Hence all the Queen players were able to avoid them. However, several of the mistakes really were caused by the fact that the player overlooked a stalemate defense deeper in the tree. This phenomenon also occurred frequently in the experiments discussed later (section 7).

Quality as a function of depth

In this section, the quality of play in the master games is calculated with the depth D as a parameter, and compared to the expected quality of play for a random player (i.e. a player who makes random 'good moves'). This will give us more insight in the conditions by which 'risky' play is more likely to be successful.

The *expected error*, d, for humans is given in Fig. 6.3, together with the expected error for the random player (excluding fatal mistakes). For the top portion of the domain, the general trend of the two error curves is similar, with the human masters committing about 25% of the error made by the random player. Though small, this error is still sufficient to prevent progress, on the average, in the class of positions between (roughly) 7–15 moves deep. At depths of 8 or less, however, the expected error drops sharply (in contrast to the error by the random player). This drop is due mainly to a drop in the probability of error, but also a drop (or non-increase) of the average magnitude of error (see below).

The *probability of error*, p, for humans[6] is given as a function of depth in Fig. 6.4, together with the error rate for the random player (excluding fatal mistakes). Again a sharp decline in error rate is noticeable for lower depths ($D \leq 10$). Surprisingly, perhaps, is that the error rate also declines for larger depths, faster than can be ascribed to the expected random error.

[6]The curve was smoothed by taking the 3-point moving average.

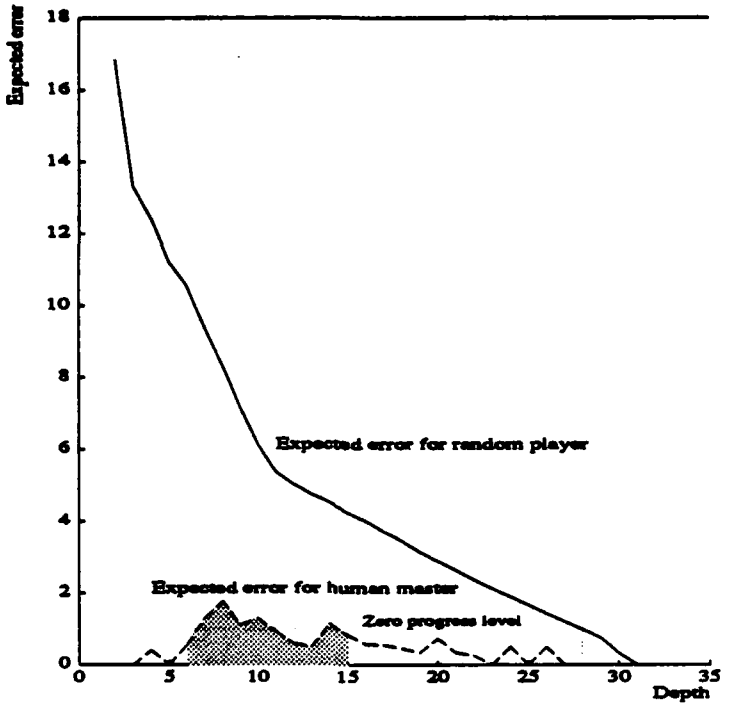

Figure 6.3: Expected error d for human masters, as compared to the random player.

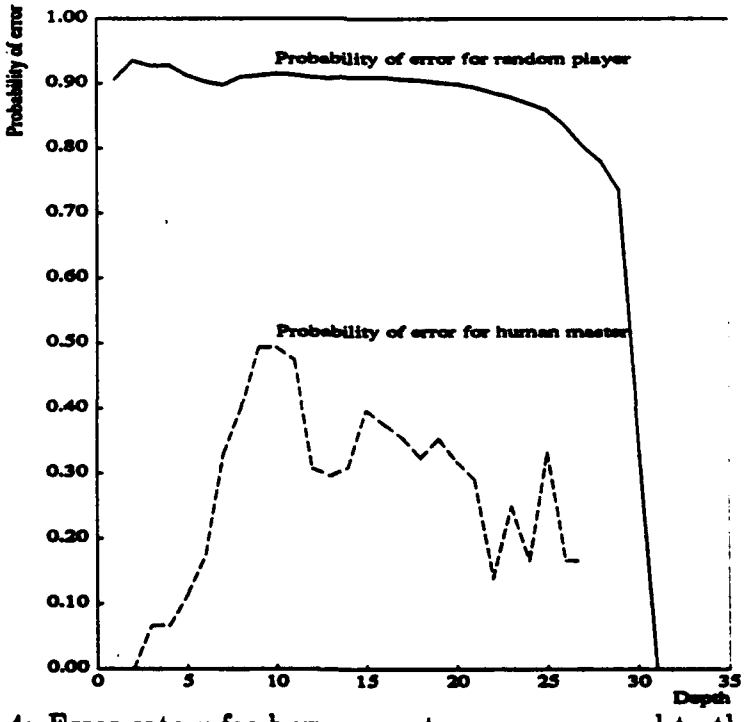

Figure 6.4: Error rate p for human masters, as compared to the random player.

The *expected error magnitude*, δ, for humans is given as a function of depth in Fig. 6.5. One of the conclusions of section 1.4 was that the allowable 'risk' should be (much) smaller

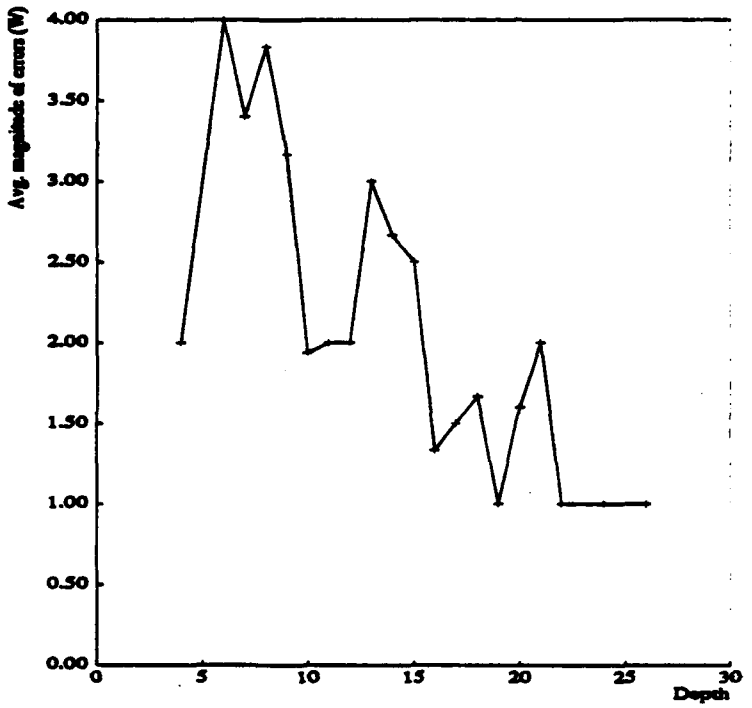

Figure 6.5: Expected error magnitude for human masters as a function of depth.

than the error magnitude of errors made by an opponent. Apparently, people make large errors only in the range of depths between 4–15.

Interpretation

We can propose the following partial explanation[7] for some of the phenomena observed in the figures above.

- That the error rate is smaller for shallow positions, is probably due to the fact that (1) in this type of position, people are capable of solving the position by *searching*, (2) *heuristics* (for example the check heuristic) are efficient and (3) most positions presented in *endgame books* are in this category.

- That, perhaps paradoxically, the error rate is also smaller for deeper positions can be explained by the fact that (1) there are relatively more optimal moves available, i.e. it is in the *nature of the domain*, and (2) *heuristics*, for instance getting the White King away from the border, are efficient here as well.

[7] In real human play other elements play a role too. For instance, human masters may grow tired in the course of a game, and this is most likely to affect the more shallow positions. These factors were ignored in the context of this thesis.

- Conversely, the larger error rate and amount in the intermediate class of positions can be explained by the complement of the above: (1) heuristics are not so efficient in such positions (see the next section on heuristics), (2) no book positions are available for learning, (3) search is impractical, and (4) the expected error is larger.

Consider the shaded region in Fig. 6.3. In this region, for depths between about 7–15, the expected error d is larger than or about equal to 1, so that on the average no progress is made from such depths. Below a depth of 15, the 'statistical pull' would tend to increase the depth towards 15; at depths larger than 15, a gradually diminishing progress would tend to also direct the depth towards 15. In other words, a depth of about 15 can be likened to a *stable equilibrium depth* for human masters in KQKR. An unstable equilibrium is a depth $D = 7$, below which the master's searching capabilities makes rapid progress possible.

Finally, when is it appropriate to make mistakes (with the Rook) to reach a position more difficult for the human opponent? The results on the magnitude of error, δ, tend to indicate that mistakes can probably only be useful for positions in between 5 – 15 moves deep, or perhaps a little deeper (up till 20) if the 'risk' (maximum error allowed for Black) is reduced to its smallest possible non-zero value, 1. The results on probability of error, p, indicate that, since p is much smaller for $D < 9$, increased difficulty is unlikely to compensate for loss of optimality for small D. In sum then, though it is always good to select a more difficult alternative to resolve ties, it is seldom worth it to make mistakes larger than 1, and these only in order to reach positions with a depth of about 10 – 15 (– 20), i.e. from positions with depths of about 12 – 17 (– 22).

6.2 Heuristics for KQKR: objective utility and use for predicting human play

In this section, we look at what heuristics are, and how they can be evaluated. If it is possible to estimate how much weight people give to certain heuristics, it is possible to predict which moves people are more likely to make, or in which positions they are more likely to make mistakes.

6.2.1 Heuristics

Pearl defines heuristics as "[...] criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal [52]". Given a number of alternatives to choose from, then, a heuristic can be regarded as a criterion or method for imposing an *ordering* on these alternatives, ranking first the ones which are, in some sense, the most promising. In this chapter we will assume a slightly more restrictive definition. Except where noted, a heuristic will be considered to be a *binary predicate* on moves. In other words, a heuristic defines a set of moves with a certain characteristic, and sets those moves apart from the others.

Instead of regarding heuristics as properties of *moves* it will frequently be easier to look at the resulting *positions*. In this case, a heuristic maps a set of successor positions of a given position onto a subset of this set. For example, the *check* heuristic selects from a list of moves only the moves which give check to the opponent. This heuristic is equally easily expressed in terms of positions: it selects from a list of positions those positions in which the opponent is in check. However, because a move is more specific than a position (it contains the precursor position), some heuristics cannot be expressed as the property of a position alone. This is true, for example, for the **ApproachK–K** heuristic (see below) which is based on the distance of the Kings in both the initial and final positions.

6.2.2 Evaluating the utility of heuristics by means of an endgame database

How good are heuristics? If a heuristic is to be used to control the search, it saves time by eliminating alternatives unlikely to lead to a solution, or by allowing the more promising alternatives to be searched first. For a heuristic to be useful, it must at least allow a better than purely random choice among alternative moves to be made. When the value of every position in the domain is known, as in KQKR, it is possible (at least in principle) to compute the contribution of any heuristic exactly, and this we shall do for three different criteria.

Let a heuristic be defined as a binary predicate $H(p_0, p_1)$ on two positions, the initial position, p_0, and the final position, p_1. Predicate $H(p_0, p_1)$ returns `True` if the move defined by p_0 and p_1 exhibits a certain property, and `False` otherwise. A heuristic can be used as a *move preselector*, and is as such an operator from sets of moves to sets of moves. With $S(p)$ the successor set of p, and $S_H(p)$ the preselected successor set, this can be written as

$$S_H(p) = \{p_i \in S(p) | H(p_i)\}$$

Whether a move is *optimal* depends of course on the initial position as much as on the final position. Let the goal be to *minimize* a given value function on positions, $V(p)$ [8]. Optimality can then be defined by the following predicate.

$$O(p_0, p_1) \equiv (p_1 \in S(p_0) \wedge \forall p_i \in S(p_0) : V(p_1) \leq V(p_i))$$

In words, the move from p_0 to p_1 is optimal if it is legal, and if no other move has a lower value.

In the following, the initial position is left out to simplify the notation, but, in principle, all of the following are to be imagined as being calculated for some specific initial position, p_0. Let the number of legal moves from position p_0 be equal to the branching factor, b. The number of optimal moves is

$$o = |S_O(p_0)| = |\{p_i \in S(p_0) | O(p_0, p_i)\}|$$

where the function $|.|$ takes the cardinality of a set. Similarly, the number of moves that conform to the heuristic, called *heuristic moves*, is

$$h = |S_H(p_0)| = |\{p_i \in S(p_0) | H(p_0, p_i)\}|$$

The number of moves which are both optimal and heuristic is

$$ho = |S_{HO}(p_0)| = |S_H(p_0) \cap S_O(p_0)| = |\{p_i \in S(p_0) | O(p_0, p_i) \wedge H(p_0, p_i)\}|$$

Let q be the probability that a random move is optimal and q_H the probability that a heuristic move is optimal. For position p_0 we then have that $q = o/b$ and $q_H = ho/h$. We can then define the *power* or *gain*, g, of a heuristic as $g = q_H/q$.

The following are three criteria that can be used to judge the goodness of a heuristic.

1. A first measure is the *fraction of positions within the domain of a heuristic for which a heuristic move is more likely to be optimal than a random move.* In the above terminology, this is the fraction of positions for which $\frac{ho}{h} > \frac{o}{b}$, or $g \geq 1$. This indicates how often using the heuristic has beneficial effects in terms of the probability of selecting an optimal move.

2. Another measure is the *average gain* g obtained by using the heuristic. Although this is clearly related to the previous measure, it can be expected to be more optimistic. The reason for this is that the average taken is an arithmetic mean, and the distribution of the gain, g, is likely to be highly asymmetrical with respect to the threshold value 1. Indeed, whereas g can never be lower than 0, it can be as large as the branching factor b when there is one optimal move which is also the only heuristic move.

3. Another way in which a heuristic can be useful is that it may select 'generally good' moves. For instance, perhaps not all the heuristic moves are optimal, but if most of them are close to optimal, the expected error can still be small. It makes sense, therefore, to compute the *expected depth after making a heuristic move* and compare it to the expected depth after a random move.

---

[8] With this convention, the situation becomes similar to the KQKR play discussed in this chapter: White tries to reduce the depth of the position with each move.

6.2.3 Some heuristics for KQKR

An observation of human play seems to indicate that the following heuristics may be important – in a positive or negative way – in the move selection of human players. This list is by no means intended to be exhaustive or complete, but provides a good idea of the range of possibilities.

Check In p_1, the Black King is in check.

Attack In p_1, the Black Rook is attacked (and insufficiently defended).

Threat If it were White's move again in p_1, he could force a quick win. A 'quick win' is defined as a position from which White can win in 2–4 moves. This excludes 'threat in 1', which is already represented by the two previous heuristics.

LongLines A 'long line' is a (WTM) position that requires at least another 4 moves to win with optimal play. This heuristic is true for a (BTM) position if there are one or two long lines from this position, i.e. if the Rook player is highly constrained in his choice of (reasonable) moves. See below for more on this interesting heuristic.

Approach(K,Q)–(K,R) Approach the K or Q to the opponent's K or R (4 different heuristics). As in a mating position the two Kings are necessarily close to each other[9], and the King may help drive the opponent's King to the border, it seems *a priori* useful to approach the King to the opponent's King as much as possible, at least when there is no concern with a higher priority. The other approaching heuristics may reduce the mibility of the piece being approached.

Centralize(K,Q) Bring the K or Q to a square closer to the center of the board. In the center, both pieces have a higher mobility, and the K can reach any square of the board quickly.

Zugzwang White puts Black in *Zugzwang* if the value of the position, if White had to move again, would be *larger* than after any Black move. in other words, if given the possibility, Black would prefer to make no move at all (for more, see below).

DecentralizeBK The Black King can only be mated on the border of the board, hence a move may be good if it forces Black to move his K in that direction.

Prevent-Check Everything else being equal, checks by the opponent cause a delay in the winning procedure, as the WK must move out of check.

These heuristics are not independent from each other. For instance, **Check** is a common way of reducing the number of long lines (in fact, the number of all lines). Some heuristics have a symmetric counterpart in the (assumed) heuristics for the opponent, from which 'second-order' heuristics can be deduced. For example, if giving check is a good strategy

[9]This excludes one not very relevant case in which the Black King is in the corner and its Rook next to it. In this case the presence of the WK is not required for the mate.

for Black as well, then *avoiding* check must be a good heuristic for White (for instance by covering with the Queen the square on which the Black Rook could check the White King). If centralization of the K is good for White, then avoiding or reducing centralization of the Black King (i.e. driving it to the border) is probably a good strategy too.

6.2.4 An 'objective' analysis of some heuristics

In this section, a few of the heuristics described above will be analyzed with respect to the three proposed measures. This will give us some insight on how good or bad those heuristics actually are, and how this depends on the depth of the position under consideration.

The Check heuristic

The **Check** heuristic is strongly present in human play, and with good reason. A plausible psychological explanation may be that a check limits the opponent's possibilities, and allows one to look deeper into the tree. From an adequately shallow root position, a player can reach terminal positions in his search (if these are reachable via checks), and the **Check** heuristic is in that case greatly beneficial. There may be other reasons for a human chess player's predilection for checks as well, for instance the feeling of control resulting from being able to restrict the opponent's movements.

Figure 6.6 shows the result of our first measure (fraction of positions in which the heuristic is likely to be beneficial) for this heuristic. In the figure, the dotted line indicates the *applicability* of the **Check** heuristic, i.e. the proportion of positions in which at least one move is check. The solid line indicates the proportion of the latter positions in which check is beneficial. The applicability is fairly high: 75.3% overall (taken over positions with a depth of 3 or more). As a function of depth, the applicability is a more or less constant 70%, except for shallow positions (depth $D \leq 6$)[10]. Overall, the check heuristic is useful in 58.2% of the positions with depths 3 or larger in which check is possible. This might seem to indicate that giving check is usually good. However, when we plot this measure as a function of depth, we see that giving check is useful only for positions that are either shallow (depth $D < 10$) or deep ($D > 28$). When the depth is in the region between, check is, more often than not, not beneficial. Therefore, when adopted as a general heuristic, check, though beneficial in some parts of the domain, may lead to errors in others.

The expected gain in probability of selecting an optimal move is graphically represented in Fig. 6.7. According to this measure, **Check** would be useful everywhere in the domain, although its utility diminishes at larger depths.

Finally, in Fig. 6.8, the expected depth after a checking move is compared with the expected depth after a random move (also averaged over positions from which a check is possible). This figure too shows that **Check** is always useful, in the sense that the expected

[10]That there are relatively more positions for shallow depths in which White can give check can be ascribed to the fact that, when White is unable to give check, this is most often because Black has just checked White first, and as we saw in section 5 the proportion of low-branching factor positions for White becomes much smaller for low depths.

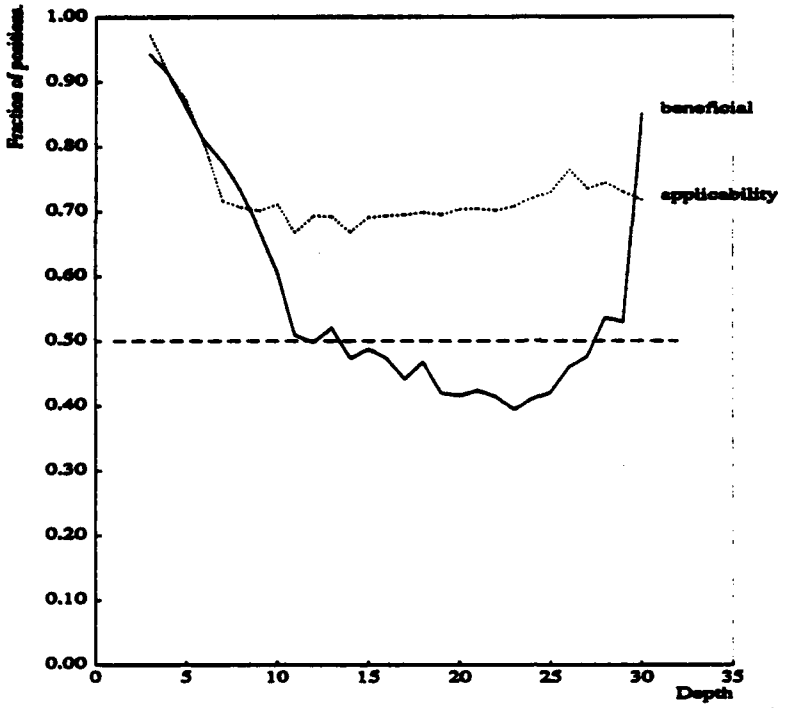

Figure 6.6: Proportion of positions where the **Check** heuristic is beneficial, as a function of depth.

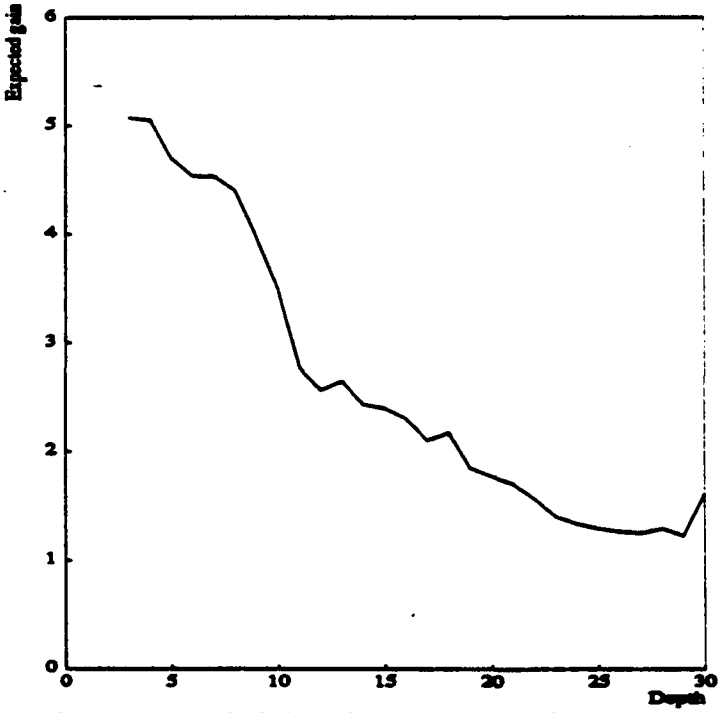

Figure 6.7: Expected gain in probability from the **Check** heuristic, as a function of depth.

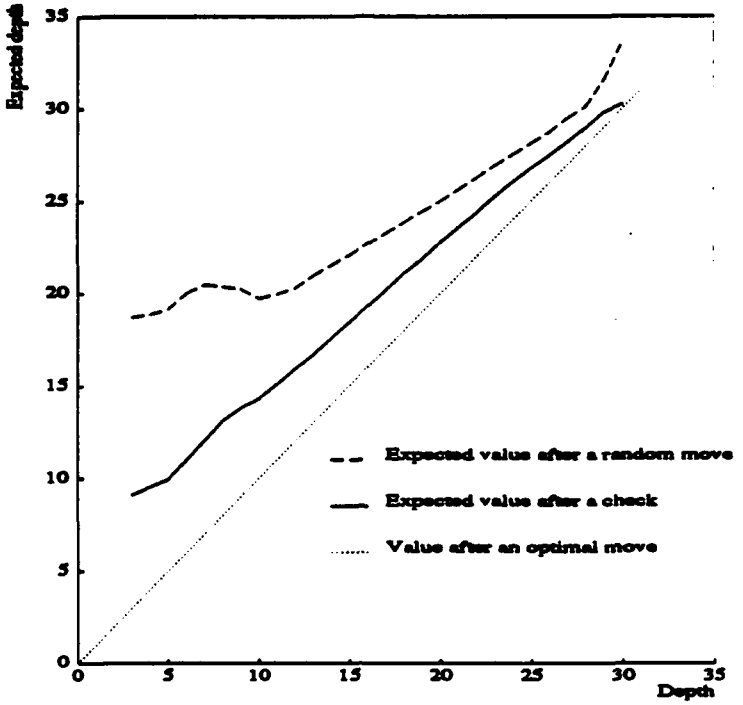

Figure 6.8: Expected depth after a **Check** move as compared to a random move.

value after a check move (solid line) is always lower than the expected value after any random move (dashed line) (both are of course larger than the depth of an optimal move, represented by the dotted line). This difference is most pronounced for shallow depths: the expected depth after check is then less than 1/2 of the overall expected depth, and is about 3 times closer to the best possible value (i.e. after an optimal move).

The LongLines heuristic

Another heuristic, with an almost identical justification for human players, is the *number of long lines* the opponent has after the move under consideration, i.e. the number of variations Black can choose from without making the win trivial. What is trivial is not so easy to determine. By lack of a better criterion, triviality was, more less arbitrarily, equated to a depth of 3 moves or less. The number of long lines is a variable that can take on multiple values. To make it consistent with the binary view of heuristics, **LongLines** was considered true for positions in which Black has 1 or 2 long lines available. In addition, to make the heuristic independent from the **Check** heuristic, only 'LongLines' moves that were *not also checks* were considered.

The proportion of positions in which this 'orthogonalized' heuristic is beneficial is represented in Fig. 6.9. Because of the restriction on checks, the applicability of this heuristic (dotted line) is small (17.8% overall, for positions with $D \geq 5$), especially for larger depths. Perhaps surprisingly, this heuristic is beneficial in less than half of these positions (38.9% overall), with the exception of large depths. This means that, in almost two thirds of the po-

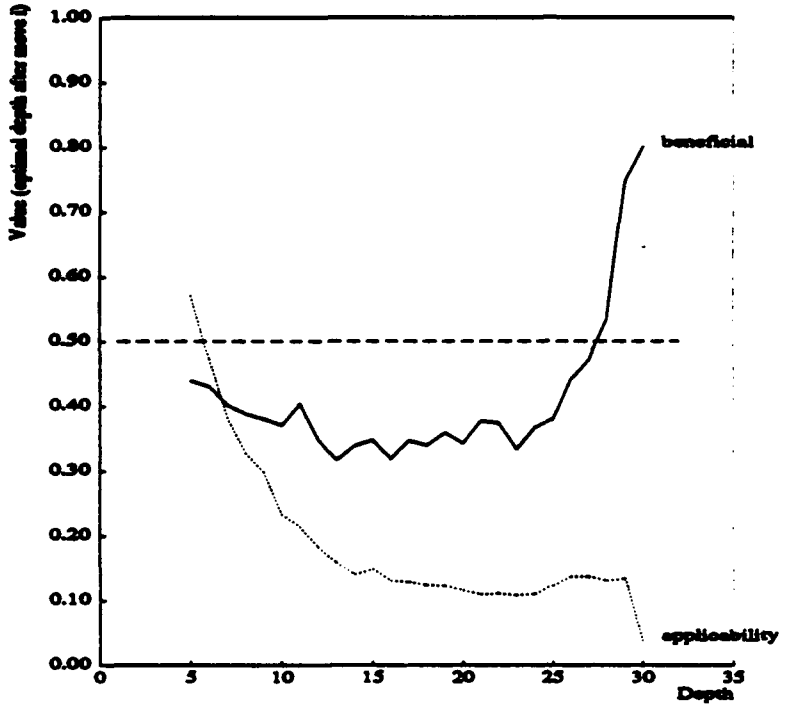

Figure 6.9: Proportion of positions where the 'LongLines' heuristic is beneficial, as a function of depth.

sitions, adopting the long-lines heuristic leads to a less-than-random probability of selecting an optimal move!

On the other hand, the expected gain in probability of selecting an optimal move is still fairly large (about a factor of 3 over the whole domain), as can be seen in Fig. 6.10. This indicates that, in those positions where the Longlines heuristic is useful, it offers a large gain, whereas when it isn't, it doesn't perform much worse than random in this respect.

The expected depth after a 'long lines' move, as compared to a random move, is given in Fig. 6.11. We see that LongLines is always useful in this respect, but much less so than Check in the lower half of the domain. Whereas Check reduces the expected error by more than 1/2, LongLines only reduces it by about 1/3. It outperforms Check slightly in the upper half of the domain, however.

The position of Black's King

As an example of a second-order heuristic, consider the position of the Black King (BK). If it is to Black's advantage to centralize his King, then it is to White's advantage to try to prevent this. Computing this heuristic requires an additional ply of search since, obviously, the BK position will not change during a White move.

Figure 6.12 gives, for each depth, the relative probability of the presence of the Black

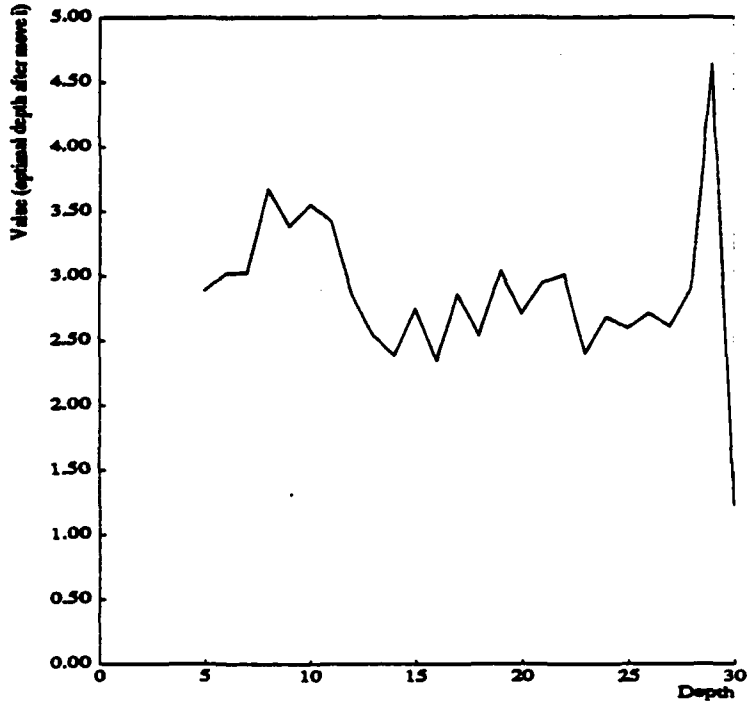

Figure 6.10: Expected gain in probability from the 'LongLines' heuristic, as a function of depth.

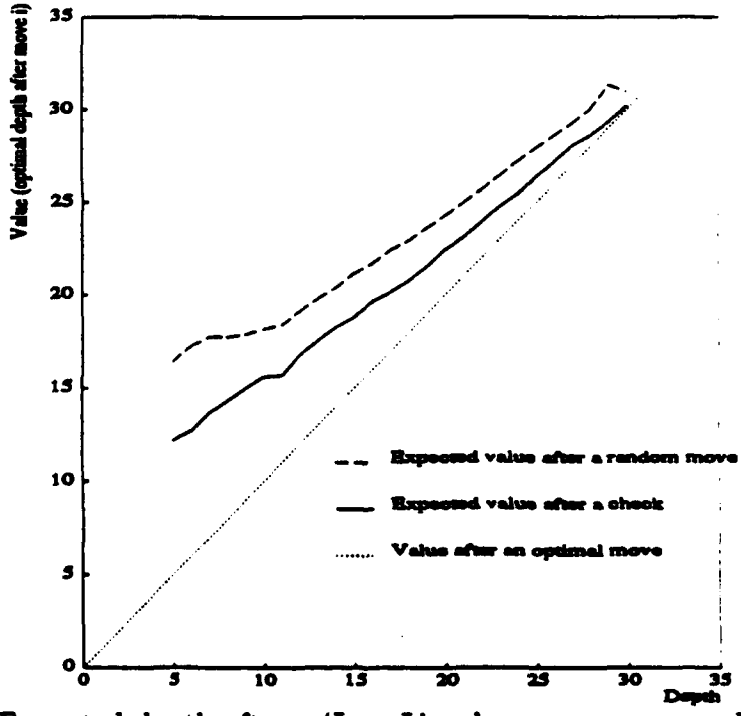

Figure 6.11: Expected depth after a 'LongLines' move as compared to a random move.

King on each of the 10 different squares[11]. For example, at depth 30 the BK is on c3 (or an equivalent square) in slightly over 50% of the positions, on d4 in about 15%, etc.

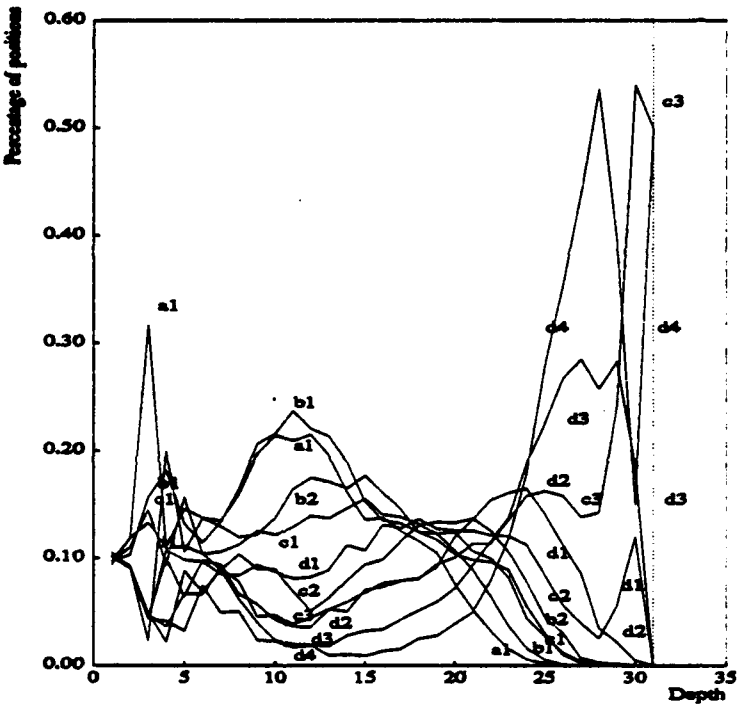

Figure 6.12: Relative frequency of the position of the BK for each of the 10 different squares, as a function of depth.

It is interesting to observe that there are basically three regions of depth in which the relative proportions of the different squares separate out, i.e. where some squares are much more likely for the Black King than other squares. A first such 'region' is at depths 3 and 4, where by far the most prominent square is the corner square **a1**, followed by the other squares on the *border* of the board, **b1**, **c1** and **d1**. This is perfectly in accordance to the general strategy taught in chess books, namely to drive the opponent's King to the border of the chess board, and preferably into the corner.

A second separating region is at a depth of roughly 8–16. Here the most prominent squares are, in order, **b1**, **a1**, **b2**. and **c1**. These are basically squares in the *corner region* of the board. Why do the *center squares* occur almost equally often when the depth is in between the two regions just mentioned (depths 5–8)? One explanation for this could be that White uses the confinement of the Black King to get the Rook to move to a bad square. After this the Rook could be successfully chased by a series of checks, during which the BK is released from the border. However, a probably more accurate explanation is that many low-depth positions *do not have optimal precursors*, i.e. can only be reached through mistakes (recall that this is the region of depths where the number of positions at each

[11] As was mentioned in Chapter 5, all other squares are equivalent, by reflection or rotation, to the 10 squares in the bottom lower left octant of the chess board.

depth decreases with depth, resulting in a relatively smaller probability that a position at this depth has a precursor at the next level). If this is the case, the distribution of the squares may tend to correspond to a mixture of the distribution for larger depths, in which the center squares are more common squares for the Black King.

Finally, a third 'separating region' is between depths of 25–30 when, as was to be expected, the BK will most often be found on the *center squares* **d4**, **c3** and **d3**.

These observations show that it is possible, to some extent, to use the placement of the BK as a feature of the position correlated with the depth to win. Note that positioning on the corner squares implies a smaller branching factor ($b \approx [3..5]$) than positioning on the center squares ($b_B \approx 8$), and that this might account for some of the correlation.

A player can benefit from forcing the enemy King to the border of the board, though in some cases this strategy may lead to a mistake. For instance, with the Black King in the corner (**a1**), various stalemate combinations are possible.

Zugzwang

When the value of the position if White had to move again would be *larger* than after any Black move, we say that White puts Black in *Zugzwang* (see also sections 3.1 and 9.1.3). If given the possibility, Black would prefer to make no move at all (Black strives to increase the value as much as possible). Conversely, when the value of the position if Black had to move again would be *less* than after any White move, Black puts White in Zugzwang. In this case White would prefer not to move.

The *threat value* of a position is the difference between the value a player can obtain if he were allowed to move again and the value he can actually obtain after the best move by his opponent. Most frequently, the threat value is positive, and the positions contains a *threat* (i.e. it is a 'hot' position[12]). Zugzwang corresponds to a *negative threat value* (i.e. the position is 'cold').

We can compute the same criteria as before for the **Zugzwang** heuristic. The proportion of positions in which putting Black in Zugzwang is beneficial is represented in Fig. 6.13. Because of the low frequency of occurrence of Zugzwang positions, the applicability of this heuristic is small (only 3.2% overall). However, when applicable, its utility is high.

The expected gain as a function of depth is given in Fig. 6.14. This gain is large indeed for most of the domain. An explanation could be that, when a zugzwang move is good, it is (almost) always the only optimal move, and in such cases the gain is obviously large.

Also in terms of expected depth, Zugzwang is exceptionally good, as can be seen in Fig. 6.15.

It is important to remark here that the Zugzwang heuristic is not a heuristic in the usual sense. Indeed, as it relies on knowledge of the exact depths of a position, Zugzwang is a *non-operational* heuristic (people cannot make use of it directly). Furthermore, the applicability region is *biased*, which is clearly visible for example in the plot of the expected depth for a random move from such a position (see Fig. 6.15).

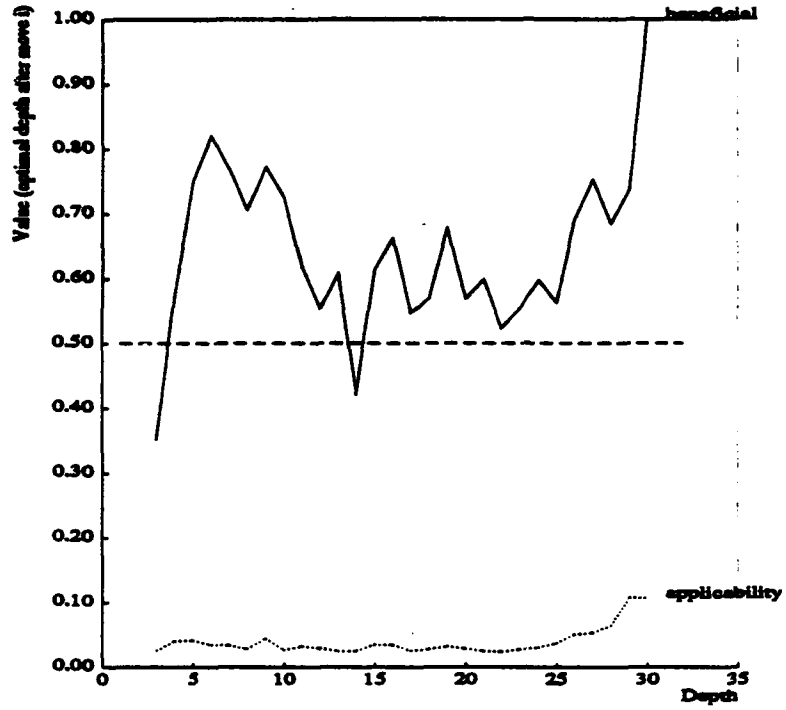

Figure 6.13: Proportion of positions where the 'Zugzwang' heuristic is beneficial, as a function of depth.

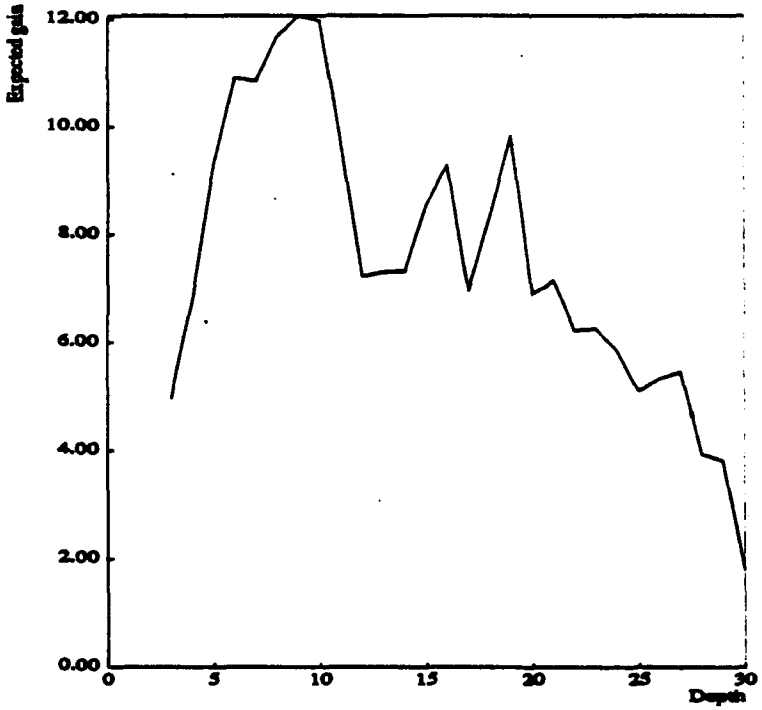

Figure 6.14: Expected gain in probability from the 'Zugzwang' heuristic, as a function of depth.

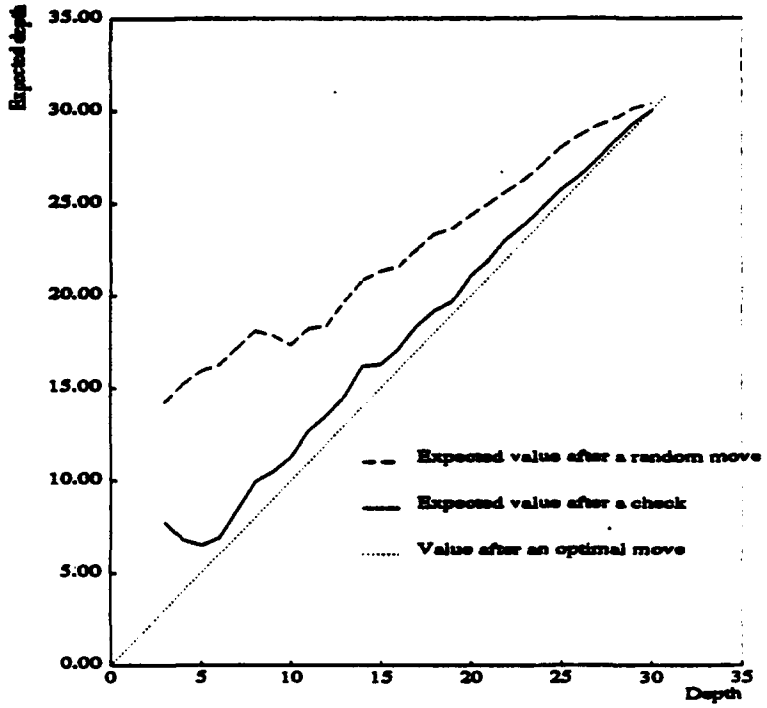

Figure 6.15: Expected depth after a 'Zugzwang' move as compared to a random move.

However, even though people cannot compute this heuristic directly, the program playing them can if it has access to the endgame database. If a program assumes that the human is less likely to make a Zugzwang move (which seems to be confirmed by the data), then it can still use this heuristic to help it predict the human's move.

Other heuristics

Other heuristics can be analyzed in the same way as the heuristics studied above. Here is a brief summary of the results for some other examples.

- **Attack** (of the Rook) is never useful (about 1/3 of the positions if the depth is larger than 10, and much less below that). This is contrary to the apparent strategy used by people[12].
- Threat (of win in 2 - 4) is even less useful (only about 25% of the positions)[13].
- Approaching the White King to the Black Rook is useful when the depth is large enough (≥ 16). Even so, the expected depth is not much better than random.

[12]An explanation for this is that people are used to playing other people, and that certain assumptions of fallibility are 'built into' the heuristics acquired from experience!

[13]The same comment holds as given in the previous note. People work with threats, mainly because other human players are likely to overlook them.

- As expected, approaching the White King to the Black king has high utility, but only for depths from 10–28. Even so, the expected depth is not much better than random, even worse for some depths (presumably because of the increased risk for a stalemate!). Of all the heuristics considered, this is the one that is the most often applicable.

- Although also highly applicable, approaching the White Queen to the Black Rook is only useful in the limited range of depths (from 5–10). Only slightly better is approaching the White Queen to the Black King, and this only for the lowest depths (3–5).

All in all, it appears that many heuristics frequently used by human players can lead to mistakes. Naturally, the above analysis is only a first-order approximation of a single-minded use of a single heuristic at the time. Human players combine multiple rules in different combinations depending on characteristics of the position, something which has been ignored in this section. Even so, the analysis may help explain some types of mistakes by human players, and the heuristics can, even when combined in fairly simple ways, help predict certain characteristics of human moves.

6.2.5 Heuristics and human play

From the human point of view, heuristics are rules of thumb that guide the decision making process when it is impossible or impractical to look at all the relevant possibilities. Based on patterns recognized on the chess board, some moves or strategies 'suggest themselves', and can be chosen directly or used to guide the search.

Heuristics are in fact condensed bits of generalized knowledge, rules that approximate the essence of the domain, rounded down to fit the size of human comprehension. These rules are synthesized from experience, either directly by playing the game, or indirectly by acquiring the knowledge from other players or from books.

Because they are based on observation and experience, heuristics may have a definite objective validity, in the sense that blindly following the rule is better than following no rule at all. But, being approximate, heuristics may, and *will* sometimes fail. Some of the reasons for failure are

1. *Variations in the domain*: If the structure of the domain varies (according to some parameter, be it visible or invisible), heuristics valid in one part of the domain may be invalid elsewhere. If human players, because of certain idiosyncratic ways of playing the game, have been limited in their experience to only a certain portion of the domain, it is quite possible that the heuristics they develop are not valid in other regions of the domain[14]. This is what we learned from the analysis in the previous subsection.

[14]As an example, chess middlegame strategies or rules may vary significantly depending on the opening played. That many of the older rules are now considered obsolete is perhaps mainly due to the fact that the older openings form only a small fraction of all the openings played by chess masters today. In computer game playing, much work went into designing different evaluation functions for different parts of the domain, while avoiding abrupt transition phenomena (see, for example, Berliner's work on SNAC coefficients [9]).

2. *Insufficient resolution*: If the power of the pattern language is not large enough to allow perfect discrimination, positions that differ in essential ways may match the same pattern, resulting in apparent randomness in the class described by the pattern.

3. *Bias in the learning experience*: People learn a game by playing it with other people. If their comprehension is not close to completely accurate, the subjective structure of the game may be entirely different from the actual structure. Heuristics that are perfectly valid when playing other human players may be worthless when playing another type of player, say an endgame database, and vice versa.

In chapter 7 these factors are considered only implicitly. A more complete analysis would try to predict human errors based, at least in part, on the situations where heuristics break down.

6.2.6 A subjective analysis of heuristics in KQKR

Although the list of heuristics considered may match some of the strategies people use in selecting a move, it is certainly far from complete, and probably inaccurate. But even without making any claim of plausibility of the model as a cognitive model, we can use heuristics as *features* to help predict which moves people will be more likely to make from a given position.

A heuristic function can be constructed by learning from a collection of datapoints taken from human play. This was not completed in the context if this thesis, except in a rudimentary sense. By an informal study of the set of master games (see section 6.1.3), as well as the experiments described below, it was found that people (playing the Queen)

- give check frequently
- reduce the number of 'reasonable moves' for the Rook side as much as possible (long lines),
- try to set up shallow threats, and
- make few, if any, Zugzwang moves.

These heuristics were incorporated in the heuristic function as described below. Some other heuristics seemed less important or more difficult to implement, and were not incorporated in the predictor function. Given the results of the experiments (see the next section), it may be worthwile to set up another experiment with an improved predictor function, including perhaps some of these other heuristics.

Learning from human play need not necessarily involve human-like heuristics. It is possible to use any reasonable features of the position to try to identify the 'human equation'. This was not attempted in this thesis, and only heuristics and information about the structure of the domain were used in the heuristic programs.

6.2.7 Constructing a predictor function

There are various ways of conceiving predictor functions to model human play, depending on attempted degree of accuracy, and on the actual quantity one is trying to predict. One or more heuristics may be combined linearly, or polynomially, or in any non-linear way. A predictor function may offer a prediction of the actual move a player is most likely to make. Alternatively, it may estimate the likelihood of an error, without reliably indicating a likely move. Yet another possibility is that it ranks positions in order of expected error, without providing any measure as to the probability of error or the chosen move.

Finding a good predictor is a problem that falls outside the scope of this thesis. When setting up the experiments (section 7) the goal was not to construct a heuristic function that would provide the best predicition for human play (neither was it to construct the best possible heuristic player). Instead, the objective was to construct a *minimal* (in the sense of least complicated and/or sophisticated) predictor function that would illustrate the possibility of 'better-than-optimal' speculative play against human players. To this end, just a few heuristics (but empirically important ones) were used in a simple multiplicative combination to provide an ordering over successor positions.

Intuitively, the value computed for each position was intended to correspond to the *expected depth* after n half-moves of play from this position against a human player. This expected depth could also be considered a measure of *difficulty*. Even if these predictions are not accurate, they may still allow us to pick the most difficult successor position, provided that the ordering is at least approximate.

As a further simplification, the influence of the various heuristics was assumed to be *independent* of the others, constant over the domain, and be characterizable by a simple geometric increase in the probability of a certain move.

The next chapter explains how such predictor functions were used in heuristic programs, and tested in play against human chess players.

Chapter 7

Making KQKR harder to win: an experiment.

This chapter discusses experiments in which the performance of human subjects against several heuristic programs is compared with their performance against control programs, which were restricted to optimal play. Recall that the main aim of these experiments was to show that it is possible to obtain better results in game playing when some knowledge about the opponent is used, even if this knowledge is rather coarse, and even if this requires the use of non-optimal strategies.

The knowledge about a human opponent is modeled by a small set of general heuristics (see chapter 6). These heuristics are combined into a predictor function, which estimates the expected depth after 4 ply. This predictor function was then used by a heuristic program to determine the best move against a human opponent, while removing the restriction that the selected move must be optimal.

Section 7.1 discusses the various programs used in the experiments, optimal and heuristic. Section 7.2 describes the experimental setup. Section 7.3 then presents the results of the experiments. These results are most naturally described in two parts, each corresponding to one particular heuristic program. Finally, section 7.4 discusses an illustrative example, taken from one of the experimental games. This example provides an additional argument in favor of the hypothesis of this chapter.

7.1 Description of the programs used

In the course of the experiments described in this section, several different programs were used to play the Rook side. All programs had access to the KQKR endgame database, and were able to determine which moves were *optimal*, by a simple 1-ply search and table look-up.

The following two subsections describe first the reference programs, which are standard endgame database programs, and then the *heuristic programs*, which make use of knowledge about human heuristics to select the most difficult move possible.

7.1.1 The reference programs

Two reference programs were used in the experiments. One program, further denoted as **R** ('Random optimal'), makes a simple uniform random selection among equi-optimal alternatives. The second program, further called **B** ('Best optimal'), performs a 2-ply search, and selects that optimal move which allows the *smallest relative number* of optimal replies by the opponent.

Let **B** have a choice among n optimal moves. Let b_i represent the opponent's branching factor after move i, and o_i the number of optimal moves. Program **B** will then select a move j for which the relative proportion of optimal moves is the smallest, or

$$\frac{o_j}{b_j} = \min_{1 \le i \le n} \left(\frac{o_i}{b_i} \right).$$

It is to be expected that program **B** will outperform program **R** to the extent that the (human) opponent behaves like a random player, i.e. if his likelihood of making an optimal move is roughly proportional to the relative number of optimal moves among all the choices.

7.1.2 The heuristic programs

The heuristic programs used in the experiments have the same basic description, and share a similar algorithm. The main differences are in the parameters, and the list of heuristics that are used.

Basic characteristics

Essentially, the *heuristic programs* used in the experiments originated as extensions of program **B**, with the following differences.

1. the program is allowed to make a move which is **non-optimal** by at most some prespecified amount, δ (see section 1.4). This amount can properly be called *risk*, as it is exactly what the program stands to lose if the opponent proceeds optimally.

2. the opponent is still modeled roughly as a random player. Instead of using a uniform probability vector (every move supposed equi-probable), the **probabilities** are now **weighted** according to whether the corresponding move conforms to certain heuristics or not.

3. the program is allowed to perform a **search**, and can select the move with the highest expected value after d ply. In practice this value was set to $d = 4$, leading to a computation time of between 2 and 14 seconds. Longer times would have made it difficult to maintain the subjects' interest.

Description of the algorithm

The search procedure can be summarized as follows. At odd levels (Max node – the player of the Rook), only those positions with a value within δ of the optimal value are considered as successors. Backup is a simple MAX step. Let $V^{(i)}$ be the backed up value at level i ($i = 0$ being the root). Then we have

$$V^{(2k)}(p) = \max_{1 \le i \le r} V^{(2k+1)}(p_i)$$

where the maximum is over the selected set of r low-risk successors. At even levels (Min node – the human opponent, playing the Queen), all moves are considered, and now the *expected value* is backed up. Let w_i be the estimated probability that the opponent will pick move i (see below), and b the branching factor. We then have

$$V^{(2k-1)}(p) = \sum_{1 \le i \le b} w_i \times V^{(2k)}(p_i).$$

In sum, the search is a 'probi-max' search, maximizing the values at choice points for the program, and computing the expected value at choice points for the opponent.

Parameters used

Following is a specific description of the parameters of the three heuristic programs used in the experiments, **HP0**, **HP1**, and **HP2**.

Program **HP0**.

The components in program **HP0** are as follows.

- The risk, or maximum allowable error, is set to $\delta = 4$.
- The depth of search is set to $d = 4$ ply. In other words, the move is chosen that results in the largest expected value after two full moves.
- As discussed before, the predictive step is a simple weighted averaging step of the backed-up values after each White move. The weights are obtained by a modification of a uniform probability vector over all legal moves, depending on whether the moves are consistent or not with the heuristics. The modifying factors were chosen in an *ad hoc* manner, but loosely based on apparent importance of the various heuristics in human play (as judged from, for example, the set of master games in section 6.1.3). The heuristics used in program **HP0** were, in order of importance,
  - Check
  - Attack
  - Zugzwang (reduces the weight)

Finally, the weights are also adjusted based on the *actual dephts* of the resulting positions. This 'catchall' heuristic is basically intended to capture the (presumably beneficial) effect of all the heuristics people use that were not incorporated into the model. It has the effect of biasing the expected values slightly in the direction of the actual values, thus resulting in 'safer' estimates.

Program **HP1**.

This program is identical to program **HP0**, except for the following points.

- The risk is reduced to $\delta = 3$.
- Two new heuristics are included[1]:
  - Threats: when White would have a short winning line (depth ≤ 4) if he were allowed to move again. The weight is adjusted in a way inversely proportional to the depth of the threat.
  - Long Lines: the number of 'reasonable' Black responses (depth ≥ 4). The weight adjustment is again inversely proportional to the number of long lines (people are more likely to select lines that reduce the number of options for the opponent).

Program **HP2**

This program is identical to program **HP1**, except that

- the risk is further reduced to $\delta = 1$, i.e. only mistakes of the smallest possible magnitude are allowed. The reason for this change was the theoretical analysis of section 1.4, and the observation that, for human players, the expected error magnitude is around 2. This means that an error larger than 1 by the program cannot on the average lead to any gain, and must be to the program's disadvantage.

7.2 Experimental setup

7.2.1 Subjects

The subjects for the experiments described here ranged from class A or B (or perhaps below) to International Master. One subject, C1, is a tournament chess computer program (ChipTest at a nominal search depth of 9 ply). Subjects A1, A2 and A3 are unrated players below expert class whose actual playing strength is not easy to determine. It is perhaps relevant that subject A2 is a very strong Go player (comparable to at least Master level for chess). Subjects E1 and E2 are expert class chess players (2000 - 2200 USCF rating) and have had a fair to extensive tournament experience. Finally, subject M1 is a National Master and M2 an International Master[2].

[1]Note that both heuristics deviate slightly from the definition given in section 6.2, in the sense that multiple values are allowed.

[2]Other subjects performed the experiments, but their results could not be included because of bias (the author), because of extremely informal (noisy) conditions, because the games played were too short to be

7.2.2 Design

For each session, a subject was presented with two or more positions (one per program) from the endgame KQKR. The subject was asked to play the side with the Queen, and to try to win the position, or at least make as much progress as possible. Positions were randomly selected from among all positions with a pre-specified depth. A program was selected at random from among the various programs (optimal or heuristic) without the knowledge of the subject.

The experiments can be divided up into two sets, as follows:

- In the first series,
  1. all interaction between the subject and the programs was mediated by the experimenter, who carried out the moves made by the program on the board, and typed the moves made by the subject into the program.
  2. the positions were either 14 or 18 moves deep
  3. only two programs were compared: programs **B** (best optimal) and **HP0** (see above for description).
  4. depending on the session, the subject was expected to make about 10 or about 20 moves in each game, but this number was not strictly enforced.
- In the second series,
  1. A computer account was set up to provide automated access by the subject to the database programs. The main program selects positions, keeps track of the number of moves made, and checks if the position has become a draw, either by value or by virtue of the 50-move rule. The program also provides the user with an ASCII representation of the board.
  2. The depth of the position to be played is selected from a range of possibilities determined by a "difficulty level" selected by the user. Again, the actual position is chosen randomly among positions of that depth.
  3. Four programs were compared here: programs **B** and **R** are restricted to optimal play, programs 1 and 2 are two versions of the heuristic program, as described above.
  4. Length of the sessions is basically left up to the subject, but a game is terminated in case of a draw[3].

7.3 Results

This section first presents an overview of all the results. Then the two main series are considered individually. Finally, some secondary results are given regarding the influence of

statistically meaningful or because they did not allow a comparison to be made (played only against one category of programs).

[3]This includes the 50-move rule, but there is no check for repetition.

chess playing strength and the effect of learning.

7.3.1 General overview of the results

A summary description of all the results (both series combined) is given in Table 7.1. As mentioned above, this list does not include some games too short to be meaningfully analyzed statistically, as well as the games played by the author.

Most subjects performed the experiments without direct preparation for the endgame. Subject M2, however, performed some experiments after some specific study of KQKR. To make a distinction, subject M2 is thereafter named M2+ in the table.

#	S	P	D	NH	EH	TH	pH	delH	dH	NC	EC	TC	pC	delC	dC	Net	DMAX
1	A1	0	14	23	14	42	0.61	3.00	1.83	23	6	12	0.26	2.00	0.52	1.31	[10]
2	A1	B	14	19	11	30	0.58	2.73	1.58	-	-	-	-	-	-	1.58	[10]
3	A2	0	14[19‡]	10	6	13	0.60	2.17	1.30	10	1	2	0.10	2.00	0.20	1.10	[10]
4	A2	B	14	10	4	6	0.40	1.50	0.60	-	-	-	-	-	-	0.60	26
5	A2	0	14	12	7	21	0.58	3.00	1.75	13	0	0	0.00	??	0.00	1.75	[10]
6	A2	B	14[26‡]	16	8	14	0.50	1.75	0.88	-	-	-	-	-	-	0.88	15
7	A3	2	8[19‡]	5	3	10	0.60	3.33	2.0	5	0	0	0	??	0	2.00	[10]
8	A3	2	10[17‡]	7	4	9	0.57	2.25	1.29	7	2	2	0.29	1.00	0.29	1.00	[10]
9	A3	R	7	8	6	8	0.75	1.33	1.00	-	-	-	-	-	-	1.00	[10]
10	E1	2	10[16‡]	33	19	69†	0.57	3.63	2.09	33	5	5	0.15	1.00	0.15	1.94	[10]
11	E1	R	7	48	22	61	0.46	2.78	1.27	-	-	-	-	-	-	1.27	[10]
12	E2	B	15	34	15	36	0.44	2.40	1.06	-	-	-	-	-	-	1.06	[10]
13	E2	1	15	48	22	64	0.46	2.90	1.33	48	8	14	0.17	1.75	0.29	1.04	[10]
14	M1	0	18	22	11	33	0.50	3.00	1.50	22	6	11	0.27	1.83	0.50	1.00	[10]
15	M1	B	18	23	7	28	0.30	4.00	1.20	-	-	-	-	-	-	1.20	[10]
16	M1	R	14	22	8	21	0.36	2.63	0.95	-	-	-	-	-	-	0.95	12
17	M1	1	14	24	12	35	0.50	3.00	1.50	24	7	10	0.29	1.43	0.42	1.08	[10]
18	M1	1	18	20	11	29	0.55	2.63	1.45	20	3	3	0.15	1.00	0.15	1.30	[10]
19	M1	R	18	25	5	10	0.20	2.00	0.40	-	-	-	-	-	-	0.40	[all]
20	M2	0	14	8	4	9	0.50	2.25	1.13	8	3	10	0.38	3.33	1.25	-0.12	[all]
21	M2	B	14[27‡]	12	4	6	0.30	1.50	0.50	-	-	-	-	-	-	0.50	30
22	M2	B	18	12	7	11	0.58	1.57	0.92	-	-	-	-	-	-	0.92	13
23	M2	0	18[23‡]	24	10	24	0.42	2.40	1.00	24	3	3	0.13	1.00	0.13	0.87	15
24	M2	1	14	21	12	30	0.57	2.50	1.43	21	3	5	0.14	1.67	0.24	1.19	[10]
25	M2	R	14	24	14	34	0.58	2.43	1.42	-	-	-	-	-	-	1.42	[10]
26	M2	1	18	20	9	28	0.45	3.10	1.40	10	5	6	0.25	1.20	0.30	1.10	[10]
27	M2	R	18	20	9	17	0.45	1.90	0.85	-	-	-	-	-	-	0.85	16
28	M2+	1	11	18	9	21	0.50	2.33	1.17	18	1	2	0.06	2.00	0.11	1.06	[10]
29	M2+	1	8[23‡]	26	12	38	0.46	3.17	1.46	26	10	19	0.38	1.90	0.73	0.73	21
30	M2+	2	10[17‡]	31	15	30	0.48	2.00	0.97	32	1	1	0.03	1.00	0.03	0.94	12
31	M2+	R	10[15‡]	35	17	37	0.49	2.18	1.06	-	-	-	-	-	-	1.06	[10]
32	M2+	R	18[22‡]	9	2	7	0.22	3.50	0.78	-	-	-	-	-	-	0.78	19
33	M2+	B	16	39	25	38	0.64	1.52	0.97	-	-	-	-	-	-	0.97	11
34	M2+	1	14	9	4	8	0.44	2.00	0.89	9	1	1	0.11	1.00	0.11	0.78	19
35	M2+	1	15	15	1	1	0.07	1.00	0.07	15	1	1	0.07	1.00	0.07	0.00	[all]
36	M2+	2	17	32	10	25	0.31	2.50	0.78	33	5	5	0.15	1.00	0.15	0.63	25
37	C1	0	18	10	3	9	0.30	3.00	0.90	10	2	6	0.20	3.00	0.60	0.30	[all]
38	C1	B	18	24	8	16	0.33	2.00	0.67	-	-	-	-	-	-	0.67	23
39	C1	1	18	32	14	29	0.44	2.07	0.91	32	4	6	0.13	1.50	0.19	0.72	21
40	C1	R	18	19	6	11	0.32	1.83	0.58	-	-	-	-	-	-	0.58	27
41	C1	R	14	15	6	11	0.40	1.83	0.73	-	-	-	-	-	-	0.73	20
42	C1	B	14	31	16	26	0.52	1.63	0.84	-	-	-	-	-	-	0.83	17
43	C1	1	14	30	20	45	0.67	2.25	1.50	30	5	8	0.17	1.60	0.27	1.23	[10]

†Draw included (value 50).
‡Large error from initial position (not included in statistics).

Table 7.1: Overview of KQKR experiments: Summary of statistics for each game. (For description, see text).

Represented in the table are the following quantities.

An ID number for each game played, chronologically per subject.

S Subject code number (see above).

P Database program used (with 0 standing for program **HP0** etc.).

D Depth of the initial position. In some instances, the subject made a large error on the very first move (before the program had any opportunity of influencing the game). This is indicated by a '†' in the table.

NH The total number of (relevant) moves played by the subject in this game. A game ended after a fatal mistake, after a pre-specified maximum number of moves, or on initiative of the subject. Not included is the move from the initial position.

EH The number of errors committed by the subject in this game

TH Total amount of the errors (sum of the number of moves lost with respect to the optimal move for each error).

pH The proportion of erroneous moves. This is simply $\mathrm{pH} = \frac{\mathrm{EH}}{\mathrm{NH}}$, and corresponds to the probability p in the model of section 1.4.

delH The average amount of error per mistake. This is given by $\mathrm{delH} = \frac{\mathrm{TH}}{\mathrm{EH}}$, and corresponds to the magnitude of error δ_{MIN} in the model of section 1.4.

dH The average amount of error per move. This is given by $\mathrm{dH} = \frac{\mathrm{TH}}{\mathrm{NH}}$. This corresponds to the measure of quality we use mainly in the experiments, and indicates how non-optimal a subject plays.

NC The total number of (relevant) moves played by the program (C for Computer) in this game. Excluded is the last move, i.e. made right before the game is ended.

EC The number of errors committed by the program in this game.

TC Total amount of the errors by the program. Both TC and EC are = 0 for the reference programs, as these make only optimal moves (zero error).

pC The proportion of erroneous moves, i.e. $\mathrm{pC} = \frac{\mathrm{EC}}{\mathrm{NC}}$ For the reference programs, this is again = 0.

delC The average amount of error per mistake. This is $\mathrm{delC} = \frac{\mathrm{TC}}{\mathrm{EC}}$, and corresponds to the magnitude of error δ_{MAX} in the model of section 1.4. This is undefined ('??' in the table) for the reference programs. It is equal to 1 for program **HP2**, since the only mistakes the program is allowed to make are those of magnitude 1.

dC The average amount of error per move for the program, i.e. This is given by $\mathrm{dC} = \frac{\mathrm{TC}}{\mathrm{NC}}$. Again, this is = 0 for the reference programs.

Net The net average number of moves lost by the subject per move, i.e. Net = dH − dC. If this quantity is smaller than 1, the subject makes progress. If larger than 1, the depth increases at each move, and the subject would not be expected to win, even if given an infinite number of moves. This is the most important measure of the relative quality of the two opponents.

DMAX the maximum depth of position the subject is expected to be able to win in 50 moves, if he played depths 10 or less perfectly. Essentially, this is another way of phrasing the previous measure.

We see that, contrary to the traditional opinion of chess theory, the KQKR endgame is indeed far from easy. Only two games were won by the human subjects collectively (games 19 and 35) (C1 won 6 out of 7 games, however). Of the human players, only the master

strength players, and A2, succeeded in making some progress in some cases, and only M2 was capable of doing this in most cases.

7.3.2 First series of experiments: HP0 vs. B

Table 7.2 is the subset of the previous table that describes the first series of experiments, in which two versions of the database program were compared. Program B always picks an optimal move, and in addition picks the optimal move with the smallest fraction of optimal moves for the opponent in case of a tie. Program 0 selects a move within 4 moves from optimal, in such a way that the predicted value after 4 ply (2 moves) is the highest possible.

#	S	P	D	NH	EH	TH	pH	delH	dH	NC	EC	TC	pC	delC	dC	Net	DMAX
1	A1	0	14	23	14	42	0.61	3.00	1.83	23	6	12	0.26	2.00	0.52	1.31	[10]
2	A1	B	14	19	11	30	0.58	2.73	1.58	-	-	-	-	-	-	1.58	[10]
3	A2	0	14[19‡]	10	6	13	0.60	2.17	1.30	10	1	2	0.10	2.00	0.20	1.10	[10]
4	A2	B	14	10	4	6	0.40	1.50	0.60	-	-	-	-	-	-	0.60	26
5	A2	0	14	12	7	21	0.58	3.00	1.75	13	0	0	0.00	??	0.00	1.75	[10]
6	A2	B	14[26‡]	16	8	14	0.50	1.75	0.88	-	-	-	-	-	-	0.88	15
14	M1	0	18	22	11	33	0.50	3.00	1.50	22	6	11	0.27	1.83	0.50	1.00	[10]
15	M1	B	18	23	7	28	0.30	4.00	1.20	-	-	-	-	-	-	1.20	[10]
20	M2	0	14	8	4	9	0.50	2.25	1.13	8	3	10	0.38	3.33	1.25	-0.12	[all]
21	M2	B	14[27‡]	12	4	6	0.30	1.50	0.50	-	-	-	-	-	-	0.50	30
22	M2	B	18	12	7	11	0.58	1.57	0.92	-	-	-	-	-	-	0.92	13
23	M2	0	18[23‡]	24	10	24	0.42	2.40	1.00	24	3	3	0.13	1.00	0.13	0.87	15
37	C1	0	18	10	3	9	0.30	3.00	0.90	10	2	6	0.20	3.00	0.60	0.30	[all]
38	C1	B	18	24	8	16	0.33	2.00	0.67	-	-	-	-	-	-	0.67	23

‡Large error from initial position (not included in statistics).

Table 7.2: Overview of first series of KQKR experiments.

The relevant error parameters, d_H and *Net*, are given in Table 7.3, averaged by subject. In addition, the table contains the explicit comparison of these parameters between the two programs **HP0** and **B**. From this table we can conclude the following:

Subject	d_H(B)	d_H(HP0)	$d_H(HP0) - d_H(B)$	Net(HP0)	Net(HP0) - d_H(B)
A1	1.58	1.83	**+0.25**	1.31	**-0.27**
A2	0.77*	1.55	**+0.78**	1.45	**+0.68**
M1	1.20	1.50	**+0.30**	1.00	**-0.20**
M2	0.71*	1.03	**+0.32**	0.63*	**-0.08**
C1	0.67*	0.90*	**+0.23**	0.30*	**-0.37**
Avg	0.99	1.36	**+0.38**	0.94	**-0.05**

Table 7.3: Main results of first series: average human error against program B vs. average human error and net average error against program HP0. A '*' indicates that the subject makes progress, in this setting.

1. The average size of mistakes made by the subjects was *larger* when playing against program **HP0** than against program **B**. The difference (0.38 on the average) is signif-

icantly larger than 0. In other words, program **HP0** succeeded in selecting positions that were *more difficult* for the subjects (see column *dH*).

2. On the other hand, because program **HP0** was allowed to drop up to 4 moves when making a mistake, the error made by the heuristic program more than nullified the extra amount of error by the the subjects, except in one single case (subject A2).

3. The stronger chess players among the subjects make fewer and smaller mistakes than the weaker chess players (for more on this, see below).

To improve the performance of the heuristic program, certain weights were altered and new heuristics introduced (most importantly the number of long lines). Also, instead of the program **B**[4] a second reference program **R** was used, which picks any optimal move at random.

7.3.3 Second series of experiments: HP1 vs. R and B

Table 7.4 extracts from Table 7.2 the data pertaining to a comparison between programs 1, **R**, and **B**. Note that, as subject C1 was a computer program, it was possible to use the

#	S	P	D	NH	EH	TH	pH	delH	dH	NC	EC	TC	pC	delC	dC	Net	DMAX
12	E2	B	15	34	15	36	0.44	2.40	1.06	-	-	-	-	-	-	1.06	[10]
13	E2	1	15	48	22	64	0.46	2.90	1.33	48	8	14	0.17	1.75	0.29	1.04	[10]
16	M1	R	14	22	6	21	0.36	2.63	0.95	-	-	-	-	-	-	0.95	12
19	M1	R	18	25	5	10	0.20	2.00	0.40	-	-	-	-	-	-	0.40	[all]
15	M1	B	18	23	7	28	0.30	4.00	1.20	-	-	-	-	-	-	1.20	[10]
17	M1	1	14	24	12	35	0.50	3.00	1.50	24	7	10	0.29	1.43	0.42	1.08	[10]
18	M1	1	18	20	11	29	0.55	2.63	1.45	20	3	3	0.15	1.00	0.15	1.30	[10]
25	M2	R	14	24	14	34	0.58	2.43	1.42	-	-	-	-	-	-	1.42	[10]
27	M2	R	18	20	9	17	0.45	1.90	0.85	-	-	-	-	-	-	0.85	16
24	M2	1	14	21	12	30	0.57	2.50	1.43	21	3	5	0.14	1.67	0.24	1.19	[10]
26	M2	1	18	20	9	28	0.45	3.10	1.40	10	5	6	0.25	1.20	0.30	1.10	[10]
31	M2+	R	10[15‡]	35	17	37	0.49	2.18	1.06	-	-	-	-	-	-	1.06	[10]
32	M2+	R	18[22‡]	9	2	7	0.22	3.50	0.78	-	-	-	-	-	-	0.78	19
33	M2+	B	16	39	25	38	0.64	1.52	0.97	-	-	-	-	-	-	0.97	11
28	M2+	1	11	18	9	21	0.50	2.33	1.17	18	1	2	0.06	2.00	0.11	1.06	[10]
29	M2+	1	8[23‡]	26	12	38	0.46	3.17	1.46	26	10	19	0.38	1.90	0.73	0.73	21
34	M2+	1	14	9	4	8	0.44	2.00	0.89	9	1	1	0.11	1.00	0.11	0.78	19
35	M2+	1	15	15	1	1	0.07	1.00	0.07	15	1	1	0.07	1.00	0.07	0.00	[all]
40	C1	R	18	19	6	11	0.32	1.83	0.58	-	-	-	-	-	-	0.58	27
41	C1	R	14	15	6	11	0.40	1.83	0.73	-	-	-	-	-	-	0.73	20
38	C1	B	18	24	8	16	0.33	2.00	0.67	-	-	-	-	-	-	0.67	23
42	C1	B	14	31	16	26	0.52	1.63	0.84	-	-	-	-	-	-	0.83	17
39	C1	1	18	32	14	29	0.44	2.07	0.91	32	4	6	0.13	1.50	0.19	0.72	21
43	C1	1	14	30	20	45	0.67	2.25	1.50	30	5	8	0.17	1.60	0.27	1.23	[10]

‡Large error from initial position (not included in statistics).

Table 7.4: Overview of second series of KQKR experiments.

same position three times. Programs **HP1**, **B**, and **R** were compared on two positions, one 14, the other 18 moves deep. We find that **HP1** outperformed **B**, which in turn does better than **R** for both positions tested, even after subtracting the program's error (**Net**). Also,

[4]Recall that program **B** selects an optimal move deterministically, such as to minimise the probability that a random opponent will play an optimal response.

both **dH** and **Net** are larger for the shallower position (see the discussion in section 6.1 for an explanation of this at first sight paradoxical result).

As for the first series of games, the relevant error parameters from the table are given in Table 7.5, as averaged by subject, with an explicit comparison between these parameters for the different programs. From this table we can conclude the following:

Subject	d_H(B,R)	d_H(HP1)	$d_H(HP1) - d_H(B,R)$	Net(HP1)	Net(HP1) - d_H(B,R)
E2	1.06	1.33	**+0.27**	1.04	**-0.02**
M1	0.84*	1.45	**+0.61**	1.16	**+0.31**
M2	1.16	1.41	**+0.25**	1.14	**-0.02**
M2+	0.99*	1.00	**+0.01**	0.66*	**-0.33**
C1	0.72*	1.19	**+0.47**	0.97*	**+0.25**
Avg	0.95	1.28	**+0.32**	0.99	**+0.04**

Table 7.5: Main results of second series: average human error against programs R and B vs. average human error and net average error against program HP1. A '*' indicates that the subject makes progress, in this setting.

1. For subjects E2, M1, M2 and C1 (i.e. 4 out of 5 subjects), program **HP1** outperforms program **R** where **dH** is concerned, i.e. *program* **HP1** *induces people to make more and/or bigger mistakes than the random optimal reference version.* For M2+ (i.e. M2 after training), there is no difference. However, note game 35, in which M2+ was virtually *forced*, by a series of Black checks, to make the correct moves until a position was reached from which the win was easily accomplished by a simple *macro* strategy (repeating pattern)[5]. In other words, this was an unusually easy position[6].

2. As to the net gain, program **HP1** obtains equal or slightly better performance with subjects E2, M1, M2 and C1, but worse with subject M2+. As such, program **HP1** fails to show an *overall improvement* with respect to the optimal reference programs. However, note that, on the average, **HP1** also does not perform any *worse* than the reference programs, in spite of the mistakes made by the program, which were rather too large because of a generous risk factor δ.

7.3.4 Program HP2 vs. HP1 and R

For a comparison of program **HP2** with **R**, the relevant games are given in Table 7.6. Program **HP2** performs considerably better than **R**, even where **Net** is concerned, for

[5]In fact this was the same repeating pattern Walter Browne employed in his second game against the database.

[6]Game 35 went as follows. The initial position selected was *White: Ka7, Qd1; Black: Kb4 Rh7* (White is in check). Play proceeded: 1. Kb6 Rh6+ 2. Kb7 Rh7+ 3. Kc6 Rh6+ 4. Kd5 (so far all is forced and optimal) 4. ... Rb6 5. Qb1+ (5.Qd2 was 1 move faster, but Qb1+ leads to the standard pattern) Ka5 6. Qa2+ Kb5 7. Qa3 and now a repetition of a 4-move pattern 7. ... Rb7 8. Qb3+ Ka6 9. Qa4+ Kb6 10. Kd6 etc. with a standard book win

subjects A3 and E1. For subject M2+, d is about the same for the two programs, and Net is slightly worse for program **HP2**.

#	S	P	D	NH	EH	TH	pH	delH	dH	NC	EC	TC	pC	delC	dC	Net	DMAX
7	A3	2	8[19‡]	5	3	10	0.60	3.33	2.0	5	0	0	0	??	0	2.00	[10]
8	A3	2	10[17‡]	7	4	9	0.57	2.25	1.29	7	2	2	0.29	1.00	0.29	1.00	[10]
9	A3	R	7	8	6	8	0.75	1.33	1.00	-	-	-	-	-	-	1.00	[10]
10	E1	2	10[16‡]	33	19	69†	0.57	3.63	2.09	33	5	5	0.15	1.00	0.15	1.94	[10]
11	E1	R	7	48	22	61	0.46	2.78	1.27	-	-	-	-	-	-	1.27	[10]
28	M2+	1	11	18	9	21	0.50	2.33	1.17	18	1	2	0.06	2.00	0.11	1.06	[10]
29	M2+	1	8[23‡]	26	12	38	0.46	3.17	1.46	26	10	19	0.38	1.90	0.73	0.73	21
30	M2+	2	10[17‡]	31	15	30	0.48	2.00	0.97	32	1	1	0.03	1.00	0.03	0.94	12
31	M2+	R	10[15‡]	35	17	37	0.49	2.18	1.06	-	-	-	-	-	-	1.06	[10]
32	M2+	R	18[22‡]	9	2	7	0.22	3.50	0.78	-	-	-	-	-	-	0.78	19
34	M2+	1	14	9	4	8	0.44	2.00	0.89	9	1	1	0.11	1.00	0.11	0.78	19
35	M2+	1	15	15	1	1	0.07	1.00	0.07	15	1	1	0.07	1.00	0.07	0.00	[all]
36	M2+	2	17	32	10	25	0.31	2.50	0.78	33	5	5	0.15	1.00	0.15	0.63	25

‡Large error from initial position (not included in statistics).

Table 7.6: Comparison of program HP2 with other programs.

Only one subject, M2+, played both **HP1** and **HP2** (games 28–30 and 34–36). The average error induced is about the same for both programs (though it would seem that program **HP1** was better in most cases). However, program **HP1** made a larger average error (as expected), with the result that program **HP2** does better than **HP1** overall.

7.3.5 Secondary results

Although incidental to the goal of the experiment, the table also gives us some information about the influence of playing strength and the influence of training. The results are not striking, but seemed interesting enough to deserve mention here.

Influence of general chess playing strength

Perhaps not too surprising is that stronger chess players (in general) also perform better in KQKR. The averages of the error parameters over all the games played by players in each category are given in table 7.7. We see that stronger players (M and E) make mistakes

Class	p	σ_p	δ	σ_δ	d	σ_d
A	0.58	0.03	2.34	0.23	1.36	0.15
E	0.48	0.03	2.58†	0.14	1.22	0.05
M	0.44	0.03	2.31	0.11	1.03	0.08
C	0.45	0.06	2.08	0.15	0.88	0.11

†Draw move excluded for the purpose of measuring magnitude.

Table 7.7: Error as a function of general chess playing strength.

less often (lower p) than class players. This difference is significant (1%). The distinction

between Masters and Experts, however, is perhaps indicative but not statistically significant. ChipTest makes statistically the same number of mistakes as Masters (or Experts). There is no significant difference in error magnitude for the human players of various classes. In other words, all players make about the same size of mistake when they do make a mistake. ChipTest does a bit better in this respect, but this result is not significant at the 5% level. Finally, there is a clear trend for lower average error with increasing playing strength. Only the difference between class players and Masters obtains a 5% significance, however. ChipTest has a significantly lower average error than class players or Experts. The difference with Masters is suggestive but not significant.

In sum, a better general skill and knowledge about the game of chess seems to help in playing one of its subdomains. Perhaps this is only due to some very general heuristics (centralize, check, ...) and skills (searching) acquired during the early steps of learning how to play the game, and less to the more specific knowledge acquired after that stage. This could explain why the difference between Masters and Experts was less than the difference between Experts and A-players.

The effects of training

At some point, more or less halfway the experiments, subject M2 spent some time learning the KQKR endgame from sections on this subject in a few endgame books, such as Pachman's and Hooper's [22, 49].

Although the experiment was not designed to measure the amount of improvement obtainable by book learning, it seemed worthwile to analyze this aspect on the basis of the available data. A comparison between M2 and M2+ (= M2 after learning) is given in Table 7.8. The differences are not large and, though indicative of a better performance for M2+,

Subject	p	σ_p	δ	σ_δ	d	σ_d
M2	0.48	0.03	2.21	0.18	1.08	0.11
M2+	0.40	0.05	2.24	0.24	0.91	0.12

Table 7.8: Error parameters before and after book learning.

not significant (the difference in p is barely significant at the 10% level). However, M2+ did perform considerably better against program HP1 than M2, which could indicate that the speculative programs are most affected by this slight improvement in play. Finally, note that the differences are not unlike what was observed for the Browne games (section 6.1).

7.4 An illustrative example

Though it is true that the experiments do not prove, in a statistically conclusive way, that better net results can be obtained by a program that occasionally makes non-optimal moves, many *specific examples* can be found in the experimental games of positions in which the

benefit of speculative play seems obvious, and where this is in fact supported by the move chosen by the subject.

The diagram in Fig. 7.1 is a case in point. It is taken from game nr. 23 (see Table 7.1), played by subject M2, against Program **HP0**. In this position, Black is in check. He cannot

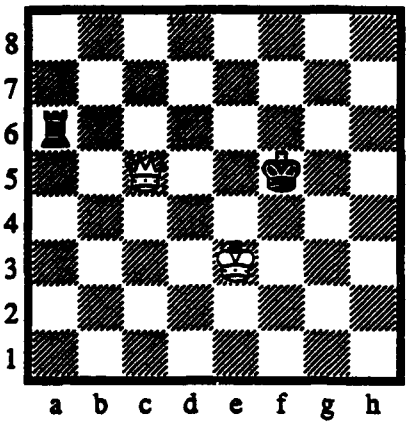

Figure 7.1: Illustrative example of speculative play.

parry the check by interposing his Rook, so he must move his King. The King has two reasonable moves, *1. Kf6* and *1. Kg6*, because after the other K-moves White wins the Rook immediately by means of a fork (double attack on K and R). Which of these is best? As the values of *1. Kf6* and *1. Kg6* are 18 and 19, respectively, a simple database program will pick *1. Kg6* (the only optimal move). Program **HP0** chose to play *1. Kf6*, however. This cost one move, but apparently induced the subject to respond with a 4-move error.

The following is a sorted list of responses after the two moves under consideration, as printed out by the database. In this list, v is the value (depth), m the branching factor, and o the number of optimal moves. The list gives the depth after each 'reasonable' response, as well as, in parentheses, the 'threat value' (optimal depth if White was allowed to move again). For example, after *1. Kg6*, the move *Qc8* leads to a win in 19, and threatens a win in 1 move (*Qc8* attacks the Rook).

```
kg6: v19 m30 o5
        Qc8:(1)19 Qd5:(2)19 Qd4:(2)19 Qc4:(1)19 Qc3:(2)19
        Ke4:(15)20 Kd4:(15)20 Qc2:(1)20 Qb5:(1)20
        Qc7:(18)21 Kf3:(17)21 Qe7:(16)21 Qf8:(17)21 Qb4:(17)21 Kf4:(13)21
        Qc1:(17)22 Kd3:(19)22 Ke2:(20)22
        Kd2:(20)23 Kf2:(20)23
        Qe5:(17)24
kf6: v18 m30 o1
        Kf3:(17)18
        Ke4:(15)19 Kf4:(13)19 Qc3:(1)19
        Qd4:(1)20 Qc4:(1)20 Qf8:(1)20 Qc2:(18)20 Qc1:(2)20
        Qd5:(15)21 Qc7:(15)21 Kd4:(17)21 Qb4:(16)21
        Ke2:(18)22 Kd3:(19)22 Kf2:(18)22 Qc8:(1)22 Qb5:(1)22
```

```
Kd2:(20)24
Qh5:(2)26
```

After the optimal *1. Kg6*, White has no less than five optimal responses, all of which involve short threats. Moreover, the (only) check, Qc2+, drops only one move. After the 'trick move' Kf6!?, White has only *one* optimal response, which has no threat (in fact it is close to being Zugzwang), is not check, and allows check (it does approach the K, but not as much as the moves Ke4 or Kf4). The checks Qc3+, Qd4+ and Qf8+ lose one or two moves. Furthermore, similar threat moves have a (slightly) higher value after Kf6 than after Kg6. In fact, it was one of those moves (*1. ... Qb5)* that was chosen by M2, dropping 4 moves with respect to the optimal move Kf3, 3 moves with respect to the global optimal line (if both players had played optimally) and 2 moves with respect to the same threat move after *1. Kg6.*

Program **HP0** chose the move, because its 'expected value' after *1. Kf6* (22.70) was larger than its expected value after *1. Kg6* (22.00), a clear apparent advantage for the move *1. Kf6.*

In sum, the 'trick move' is here justifiably best against a human opponent. But, how do specific positions like this help support the argument that occasional mistakes can be beneficial against fallible opponents? Consider the limiting case of a program that plays optimally, except in the position given above, or a small set of similar positions, in which it is highly probable that speculative play will lead to an improvement. Then this program will, with high probability, never do worse than the standard optimal program, and occasionally better. Even if such an occasion arises only once in a 50-move game (i.e. a 2% probability), this could cause an otherwise lost game to be drawn!

Finale

Chapter 8

Discussion

This chapter first looks back to the two main parts of this thesis, and discusses what we can learn from the analysis and experiments described there.

Section 8.3 tries to put the results on speculative play obtained in this thesis in a wider perspective. Finally, section 8.4 discusses the contributions of this thesis.

8.1 Lessons from Random Games

In Part I, we have seen that several intriguing and intricate properties of games can be explained just on the basis of their game graph. The simple fact that two opposing forces operate in alternation on a bipartite graph with random edges is suficient to derive, qualitatively, certain interesting characteristics of games.

In fact, the random-game model can be considered a *theory* of games, capable of explaining several intriguing questions about games suggested by an examination of some of the apparent peculiarities of completely solved games (e.g. endgame databases). Though only approximate (especially for larger depths), this theory predicts qualitatively certain properties of actual games. Let us consider some of these in more detail.

8.1.1 What can we learn from the mathematical model?

A main result of chapter 4 is that it is possible to model the structure of games with a simple set of difference equations, describing the behavior of the retrograde analysis algorithm. This provides us with a powerful tool for the analysis of properties of games.

This model allows us to answer questions which did not fit so easily in a standard forward-search framework. In Part II, we have seen that it allows us to make predictions, based on the rules of a game, on

1. whether the game is won or drawn (according to the theory: when the proportion of terminal wins is above or below the threshold T_α, respectively),
2. whether the game is interesting or trivial, (proportion of terminal wins above or below another threshold, which may be domain-size dependent), and
3. whether the game is likely to be deep (according to the theory: close to T_α on either side, at least when the domain size is large enough).

The model also *explains, in a qualitative way, the depth distributions observed in real games*, most notably the 'bi-modal' distribution of total wins. This result is not easily explained intuitively, but follows in a rather straightforward manner from an inspection of the game equations.

An analysis of the difference equation also explains why there are so few games in between draws and total wins, i.e. *why the dichotomy between wins and draws is so sharp.* Counter-intuitive at first is the fact that the *number of terminal wins*, t_0, required for interesting games is such a large portion of the domain. We tend to think that whether a game can be won depends on whether we can 'force' our opponent in the direction of a possibly very small set of winning positions. However, the only way such forcing is possible, is when either (1) Black's mobility decreases significantly close to the win or (2) there are many immediate wins which Black needs to avoid along the way[1]. However, it is easy to overlook the many

[1]These two conditions can in fact be considered equivalent, since a game in which one of them is true can always be mapped onto an equivalent game in which some extra positions have been added or deleted.

positions that are built in into the rules of the game. They are easy to recognize, and hence need not be remembered or studied.

The random-game model also allows us to investigate other properties of games in a statistical way.

- For instance, one can investigate *difficulty* and probability of error in random games.
- One could analyze empirically or mathematically how many and which positions are likely to be *important* in the sense that many optimal paths pass through these positions.
- One could try to predict how certain properties are likely to *change with the depth* $D(p)$, for example, the number of optimal moves, or the number of 'good moves' (win-preserving moves).
- One can analyze the phenomenon of *zugzwang* and *threat value* based on random games, by randomly combining one BTM position with each WTM position.

These, and other, possibilities were not pursued in depth in the context of this thesis, though some (difficulty, and how certain properties vary with depth) were analyzed in some detail in the particular context of the KQKR endgame. See also chapter 5 and section 9.1.

8.1.2 A critical look at the assumptions and approximations

In order to make the analysis of a game tractable, we have assumed that the game is *random* with a fixed branching factor. In addition, to make the model tractable, we have made several simplifying approximations. Because of these assumptions and approximations, as well as the inherent randomness of the graph, some predictions of the model (e.g. whether a certain game is a win, or what its maximum or average depth is, or at which point the 'bottleneck' in the distribution occurs, if any) can not be expected to be very accurate.

Without going into great detail, it is still relevant to speculate what the effect is when these assumptions and/or approximations are not satisfied.

1. *Variable branching factor*: For most games, the branching factor is not uniform, but changes depending on characteristics of the position (an exception is the game of NIM, where both players always have three moves). One observation is that it is *Black's* branching factor b_B that is most determinative in the properties of the game. Indeed, whereas, for shallow depths, the number of winning positions at the next level will be *proportional* to b_W and the number of losing positions at the previous level, the number of losing positions at the next level depends in a roughly (negative) *exponential* way on b_B. This means that, the fewer moves Black has 'close' to terminal winning positions, the larger the probability that the domain will be a total win for a given number of terminal wins. This may be one argument in favor of limiting an opponent's mobility. Suppose the terminal positions are distributed roughly evenly over the domain, but the domain can be roughly divided into regions of different branching factor (the K in the middle vs. the K in the corner, say). Then the chance that one such region is a total win (and hence the specific position is winning) is larger if b_B is smaller.

2. *The presence of terminal Black wins or draws* (see assumption 5): When there are Black terminal non-losses (say $t_=$), the second equation of equations 4.1 becomes

$$l_i = (N - t_=) \times \left(\frac{w_{i-1}}{N}\right)^{b_B} \tag{8.1}$$

Hence, the number of Black losses at each depth will be lower than the values predicted by the model. by a certain fixed fraction. If the other parameters the same, this results in a lower probability of having a total win, and a 'flattening' of the second mode of the distribution (for a total win). In other words, the threshold T_a will be higher. A second consequence is that the number of winning positions no longer approaches the whole size of the domain. Taking the third equation of equations 4.1 and working out, we get that

$$w_i < N - (N - w_0) \times (\frac{t_=}{N})^{b_W} \tag{8.2}$$

This is very close to N when the branching factor is large (as in KQKR, for example), but may be significantly lower if not.

3. *Positions are discrete* (see assumption 1): Since the difference equations are *not dependent on the depth* (i), random changes at lower depths may mean that the remainder of the distribution is shifted to higher or lower depths (this can be seen very clearly on Figure 4.9, where three different random games with the same parameters are compared). The sensitivity to random fluctuations is of course the largest in the *bottleneck* region of the distribution. If the bottleneck is narrow, and/or the total domain size is small, it is probable that a game that would be a total win according to its parameters, will be a draw because of truncation. This *increases the effective value of* T_a and also makes it *dependent on the domain size* N for some values of the parameters. On the other hand, random fluctuations may also push a game that should be a draw just enough over the critical mass to turn it into a win. As this phenomenon is strongest for small games, it can clearly be seen in Figure 3.4. For larger games, the transition at the threshold value can be expected to be much sharper.

4. *Sampling without replacement* (see assumption 3): Every node marked as a win at some level, 'occupies receptors' on certain lost positions and vice versa. Hence, the probability of the next node is *dependent* on whether the previous node was marked a win or not (smaller if the position was marked). The effect of this is that fewer positions will be marked wins than expected when the number of wins and losses is already large, resulting in a *flattening* of the second maximum of the distribution.

5. *History dependence*[2] (see assumption 2): According to the assumption, we can treat all wins and losses generated at previous stages in a statistically uniform way when generating the next stage. This is clearly not the case. It is not easy to tell what the influence of this assumption actually is, but it would seem that a violation of this

[2]In this context, 'history' means all that went before in the course of executing the retrograde analysis algorithm. History independence means that at each point only the total set size counts and not the number of positions added at the previous step.

assumption contributes also to a flattening of the actual distribution. In addition, the maximum is pushed somewhat to larger depths. Intuitively, the reason is that the number of wins or losses at each next level is limited by the number on the previous level, which is not taken into account in the model.

8.1.3 Further comments

Applicability to other domains

The retrograde algorithm can also be used for much larger domains, at least for some part of these, provided that a subgame can be factored out which does not depend (or depends only to a limited extent) on other parts of the domain. One example of an *infinite* domain which proves analyzable is the endgame of King and Rook versus King on an infinite board. The reason is that the range of types of positions resulting from any given initial position is limited, and that positions outside those bounds can thus a priori be marked as terminal 'draws'.

Comparison with solutions for puzzles

It is interesting to compare the dynamics of optimal game graphs with the dynamics of optimal puzzle graphs (1-player games). It is possible to apply the same iterative technique described in section 3 to puzzles (1-player games). The algorithm will actually be simpler since only the step corresponding to finding new positions in W_W will have to be used.

Since only the amplifying step is retained, we can expect the following properties for the optimal game graph.

- The distribution increases more or less exponentially, limited only by exhaustion of the whole domain.
- The maximum depth of solutions is likely to be less than the same parameter for a 2-player game for the same size of domain, unless the number of terminal positions is very small, as is the case for the 8-puzzle.

As an example, Fig 8.1 gives the distribution of positions over wins for the 8-puzzle.

What happens if the game graph is a tree?

Treating the game graph as a tree implies the assumption that the indegree of every node can be taken to be equal to 1, as opposed to the outdegree (equal to the actual branching factor of the game).

In this case, the forward-search approach discussed in the literature applies[52], and the proportion of terminal nodes, together with the branching factor and the depth of the tree, determines the probability that the root node is a win or a draw. Note that part of

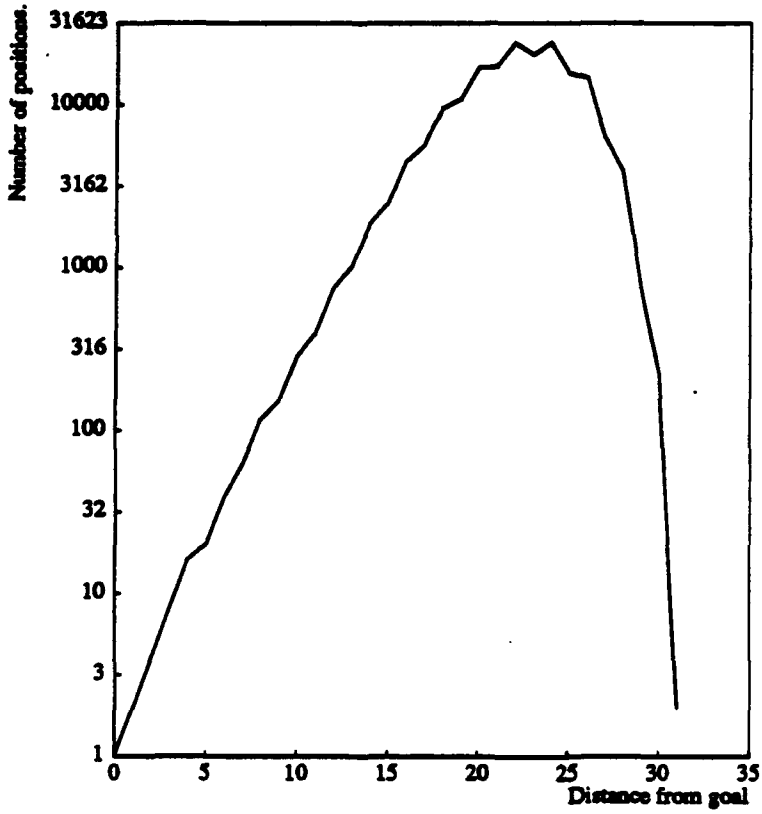

Figure 8.1: Distribution (log) of nodes over distance for the 8 puzzle.

the structure has been fixed in advance by assuming uniform depth, so that the potential structure of a game is limited to a very small set of possibilities.

The assumption that the game graph is a tree is false for most games. Only a few artificial games (e.g. Pearl's game) satisfy it exactly. Nevertheless, there do exist games which can be approximated by a game tree because their maximum length is fixed and fairly small. Examples of such games are Tic Tac Toe and Othello. Even in these games the assumption is not completely accurate as the same position can be reached via different paths.

8.2 Lessons from KQKR

8.2.1 Do the experiments show a potential for speculative play?

The goal of the experiments was to show that the KQKR endgame is a 'real' example of a game in which speculative play may lead to better performance than optimal play.

To meet the conditions derived from the simple model in section 1.4 (see figure 1.4), we need a situation where

1. the fallible opponent makes mistakes with a probability close to 50%,
2. the size of mistakes made by the opponent is (much) larger than the size of the mistakes made by the program (risk), and
3. the program can predict with reasonable accuracy whether the opponent will make a mistake.

Conditions 1 and 2 are fulfilled by human masters playing the Queen in a large portion (the intermediate region of depth) of the KQKR domain (see Section 6.1). The weak link in the experiments was the quality of the predictor functions used, in which only a few relevant heuristics were combined in a rather ad hoc fashion. On the other hand, unlike in the model, the heuristic programs will often help resolve ties between optimal moves. In such a case even a fairly bad predictor may yield good results, which may warrant its use in spite of occasional misses.

In the experiments, the heuristic programs were capable of selecting *more difficult positions* in their games against human players than the control programs. This is evidenced by the – statistically significant – increase in average error for humans when playing a heuristic program as opposed to a control program. To some extent this may be the consequence of the statistical 'pull' towards the middle region of depth (i.e. a mistake by the program is a move to a position in which the human, because of the nature of the domain, is likely to make a slightly larger error), but this effect seems insufficient to adequately explain the observed differences. The follow-up experiments proposed in the next section would, at least for the most part, exclude this bias.

To achieve increased difficulty, the heuristic programs make occasional mistakes themselves, limited by a 'risk' value. These mistakes bring down the net performance of the program, and none of the heuristic programs show a statistically significant net improvement with respect to the control programs[3]. In part, this is due to the fact that the number of data points obtained was relatively small, and in part because the heuristic programs are far from the best possible, both in terms of predictive accuracy, as in terms of the size of the mistakes (risk). To make a stronger case, additional experiments would be required. A better predictor function is also essential in a clear demonstration, as was shown in section 1.4.

[3]However, note that the heuristic programs did, on the average, not perform worse either, in spite of the mistakes.

In sum, then, the experiments strongly hint at the potential of speculative play in KQKR, if not in a statistically conclusive way. The experiments also provided a number of intuitively convincing examples of specific positions that would be particularly appropriate for the method (and indeed were used to good effect in the experimental games).

8.2.2 How relevant is KQKR as a domain?

The KQKR domain is a small subgame of the game of chess, and it has nowhere near its complexity. How can a toy domain help understand general issues?

One of the usual motivations for using toy domains is to make the experiments and measurements computationally manageable. It is possible to do a complete analysis of certain parameters over the whole domain. This is clearly the case with KQKR, a 4-piece endgame, which has a total size of about 4 million positions (2 million with each player to move). Although some of the analysis would still be possible with 5-piece endgames, larger endgames are out of the question with current technology.

In addition, taking a toy domain allows the researcher to focus on one particular issue at the time. In this thesis, the goal was to investigate how certain knowledge of an opponent could permit more effective strategies against such an opponent. Having perfect knowledge of the whole structure of the KQKR domain makes it possible to concentrate on the issues related to playing an opponent without the noise introduced by uncertainties about the domain and the evaluation by the agent.

In spite of its size, the domain proves to be of sufficient complexity to contain positions ranging from trivially simple, to hard enough to confound even the best human chess players. There are, for each chess player, a sizeable number of positions on the borderline of the possible that might be pushed into the impossible by even small changes in play. In other words, though the domain is small, the problems it presents to human chess players are not.

Even as it is, any strategies discovered to improve play against human players could immediately be incorporated in an actual tournament chess program. Indeed, when such a program ends up on the losing side of a KQKR position, it has nothing to lose, and a draw to gain with higher probability, by employing a speculative strategy.

In addition, the analysis of random games suggests that many of the statistical properties of the KQKR endgame will be similar to those of other 'won' games, with the necessary modifications for changes in branching factor, etc. KQKR is typical for a large class of 2-player zero-sum games with alternating moves and approximately constant branching factor. Other won games will also exhibit the 'bottleneck' and a second 'mode' at large depths. They will also have slightly more than 1 optimal move from each position, increasing towards larger depths, and the random player will be increasingly confused when closer to a win. For larger games, the fraction that can be searched by a human player will be smaller, and many more positions will be incomprehensible by his search and heuristics[4]. But the issues regarding

[4]This is clearly seen in most of the 5-piece endings, such as KBBKN, and even more so in 6-piece endings, such as KBRKNN.

what makes positions easier or harder for a human player are likely to be similar for larger domains.

In this sense, the present analysis of the KQKR domain is relevant to similar issues in other games as well, in that it provides an example of how relevant parameters in a domain can be measured, and how they affect a problem solver trying to find the best move in a certain position. Some results can be extrapolated to domains where an analysis like the one given is impossible.

The algorithm itself, though not particularly original or optimized, is directly applicable to other domains where estimated human heuristics are known (or any other way of computing the likelihood of moves), and a fairly accurate evaluation function is available.

8.2.3 Applicability of speculative play

One of the reasons the number of data points turned out to be insufficient may have been that the opportunity for speculative play arises relatively infrequently. Let us consider this point in a little more depth. How often can running a risk make a difference in the KQKR endgame? Fig. 8.2 shows how the number of 'considerable moves' increases as a function of the risk δ the program is willing to take. The figure plots, as a function of depth, how many

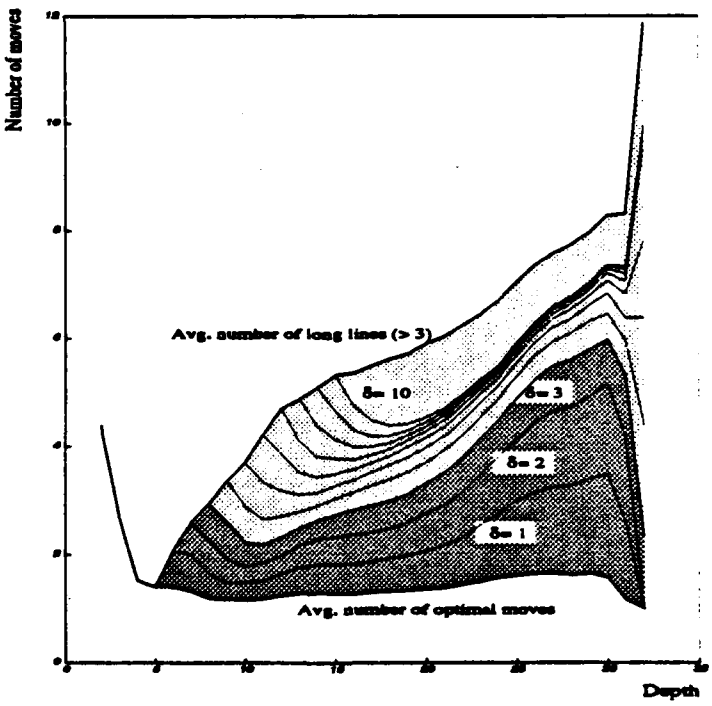

Figure 8.2: Average number of long lines available with risk limited to δ. Optimal moves have $\delta = 0$, the total number of long lines provides the upper limit.

moves are available to a player with a risk of δ or less. The total shaded area corresponds to the maximum possible number of speculative options (on the average). The dark shaded

area corresponds to the example of $\delta = 3$. In the region of interest (say depths 12–17), the number of speculative moves is about equal to the number of optimal moves, for larger depths, it is about 2.5 times as large. If the risk is limited to $\delta = 1$, the number of speculative moves is 1/3 to 1 times the number of optimal moves in these respective situations. In this last case, the option of making a speculative move arises once every 2 or 3 moves. Even when it does, the predictor function will predict the speculative move to be better in about (at most) 20% of the cases (estimated based on the model in section 1.4). All in all this means that the speculative program would, by its 'mistakes', be distinguishable from an optimal program in about 10% of the positions[5]. Hence, the result would be noticeable only in long games, and even in the course of a 50-move game, the heuristic program would make only about 5 mistakes in all. This means that we *cannot expect* statistically significant results from fairly short experiments.

8.2.4 The interaction between a player and the domain

Both the analysis of master games and the experimental results show that the quality of human play is affected by the local structure of the domain, as well as by some more or less general knowledge (heuristics) about the domain.

In various respects, a human player's behavior is similar to that of a random player. The general shape of the graphical representation as a function of depth of the probability of mistake, the expected error and the error magnitude for a human player are similar to the same parameters for a random player. There are however the following main differences:

- The error parameters are much smaller than for the random player (for all depths).
- The shape of the error curves breaks down abruptly for lower depths, and errors become small, unlike for the random player where they keep increasing towards smaller depths.

The reasons for this difference are to be found in the fact that

1. human chess players carry out a *search*. This mainly helps for shallow depths (which is the reason for the large differences with a random player at such depths), but may be significant for deeper positions as well (as it helps weed out short sidelines).
2. people use *heuristics* to guide the search. The benefit of heuristics depends on the depth of the domain (and indeed, people make the largest errors in a region where the common Check heuristic is less useful). Even so, a refined combination of heuristics may provide people with a relatively accurate evaluation function, keeping the error low in the whole domain.
3. *book knowledge* about the endgame is mostly restricted to shallow positions, with the exception of one or two lines of play (which are not analyzed completely accurately).

To successfully predict human behavior, then, it is useful to include elements of all of the above (local domain structure, depth of search, heuristics and book knowledge). These elements were to some degree incorporated in the heuristic programs, either explicitly or implicitly.

[5] In the experiments with program HP2, for which the maximum value of δ was set to 1, the probability was about 15%.

- The structure of the domain, or at least the local branching factor, and distributions of depths over successor moves, was represented implicitly: in the generation of the initial weight vector (uniform distribution over all moves) before modification by heuristics, and in the backing up of the depths for all successor positions. If there are fewer optimal moves in a position, this will reflect in a higher backed-up value, corresponding to a larger predicted error.

- Some search parameters are implicitly present as well, though they are perhaps not reducible to actual human search parameters. Some parameters that can be reduced to implicit assumptions are: the definition of 'long lines' as lines requiring a search depth of at least 4 moves (7 ply), and the 4-ply search depth performed by the heuristic programs.

- The only 'knowledge' of human play (however approximate) that was taken into account explicitly was the use of various heuristics. The interaction of the heuristics with the domain is indirect (via the predicted values).

- Book knowledge on KQKR has been mostly ignored, as not much has been written about it, and then mostly about fairly shallow positions (less than 10 moves deep). The only implicit (and indirect) bow to potential book knowledge of the human opponent, has been to reject speculative moves when the depth is less than 10.

8.3 General discussion

This section tries to provide an answer to possible objections to and questions about the thesis.

8.3.1 How convincing are the KQKR results?

It follows conclusively from the experiments that the heuristic programs are capable of selecting positions that are more difficult for human (and even computer) opponents. In addition, the experiments show that a heuristic program does not perform worse overall, in spite of the 'mistakes' it makes. However, there was no *net improvement* for the heuristic programs, and one could question the utility of speculative play in this context. Why can this result be considered a positive result?

Note that the heuristic programs used in the experiments were clearly not the best possible: only a few heuristics were used in a predictor function constructed in an *ad hoc* manner. Also, the same strategy was followed in (almost) the whole domain, contrary to what the findings in section 6.1.3 would suggest.

There are various ways in which the predictor function can be improved. The predictor function can be trained automatically, based on the master games and other data. The search depth can be extended. The parameters of the heuristic function could be made dependent on the current situation on the board (notably the current depth).

In addition, more datapoints are needed to make conclusive claims, since the opportunity to make a speculative move arises only infrequently.

Finally, there is the fact that not all human players are alike. Better predictions, and therefore better speculative results, can be obtained by specific preparation for an individual opponent.

In sum, it seems clear that further research has good prospects of demonstrating the validity of the claim that speculative play can obtain better overall results against human players.

8.3.2 The influence of uncertainty

Another question one might ask is if speculative play may still be useful in a more general context. In the previous chapters we only considered the situation where the program (agent) has complete knowledge of the domain but not of the opponent, in a deterministic game. We could consider two possible deviations from the present case: the domain may not be completely known (such as in the complete game of chess), or the moves themselves may be subject to chance (such as in the game of Backgammon).

In both cases, uncertainty is introduced, in the latter case about the potential future positions, and in the former about the value of the positions obtained. Both uncertainties have an effect on speculative play similar to the prediction error analyzed in section 1.4: the

larger the uncertainty, the more stringent the requirements on position and opponent for speculative play. As a result, it will be much less frequently advisable to make speculative mistakes.

Even so, however, a speculative strategy may always be used to decide between ties, i.e. moves that appear to have the same value. This includes, for example, situations when the agent decides that the game is lost in all variations, in which every losing line can be considered 'equi-optimal' in some sense.

8.3.3 Applying speculative play to the complete game of Chess

A last question discussed in this section is whether the technique used for the KQKR experiments in chapter 7 is applicable to more general situations.

In the complete game of chess against a human opponent, the exact values are not known, nor can the opponent be predicted accurately. In addition, incorporating the predictions into the search makes cutoffs impossible and greatly increases the required search time (or, conversely, the search depth that can be reached in a given amount of time).

However, one approximate technique, consisting of modeling a human opponent as *a computer program that carries out an identical but more shallow search*, may benefit a chess program in tactical situations[6].

As in the KQKR experiments we can try to assign probabilities to each of a set of candidate moves from a given starting positions. These probabilities can be computed on the basis of the evaluation a computer program calculates for this candidate move at a shallow depth, and are estimates of the likelihood that a human player would choose this move. The "true" values can be estimated by a deeper search (instead of a table lookup). The *speculative value* of the position, i.e. expected actual value after a human move from this position, is then the weighted average of the true values for all the candidate moves with the probabilities as described above.

The following example, one of many similar ones, may serve to illustrate this point. The diagram in Figure 8.3 is taken from a recent candidate match for the chess world championship (Ivanchuk–Yusupov, 10th match game, August 1991). It is the position after Black's 27. ... Re6.

In this position White (Ivanchuk) went on with 28. Qb7, which is a mistake ('?') and should have lead to a drawn outcome. Instead, a better move is 28. Nce7 ('!'), which would (probably) have won. Figure 8.4 shows the evaluations of eight candidate moves by the *Deep Thought* chess program (DT), as a function of search depth (1 pawn = 128 points). In the figure, we see that the move chosen by Ivanchuk is the move that DT gives the highest value at search depths of 6 or 7 ply (about a Queen advantage for White). For deeper searches, DT's value drops to a draw value. The actual best move is ranked 5th by a 5-ply search, and

[6]This is only one possible approach. The appeal stems from the possibility of using existing chess programs directly. Some work in this direction was described by this author elsewhere[23], and is part of an ongoing project.

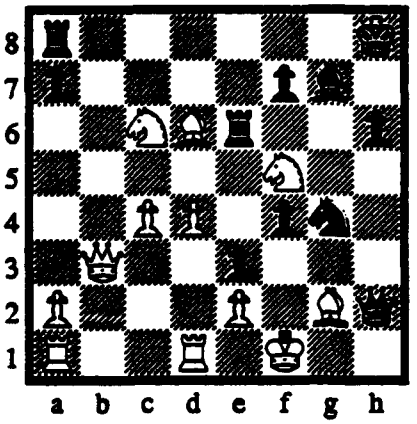

Figure 8.3: Position after 27. ... Re6 in Ivanchuk–Yusupov (10), 8/91.

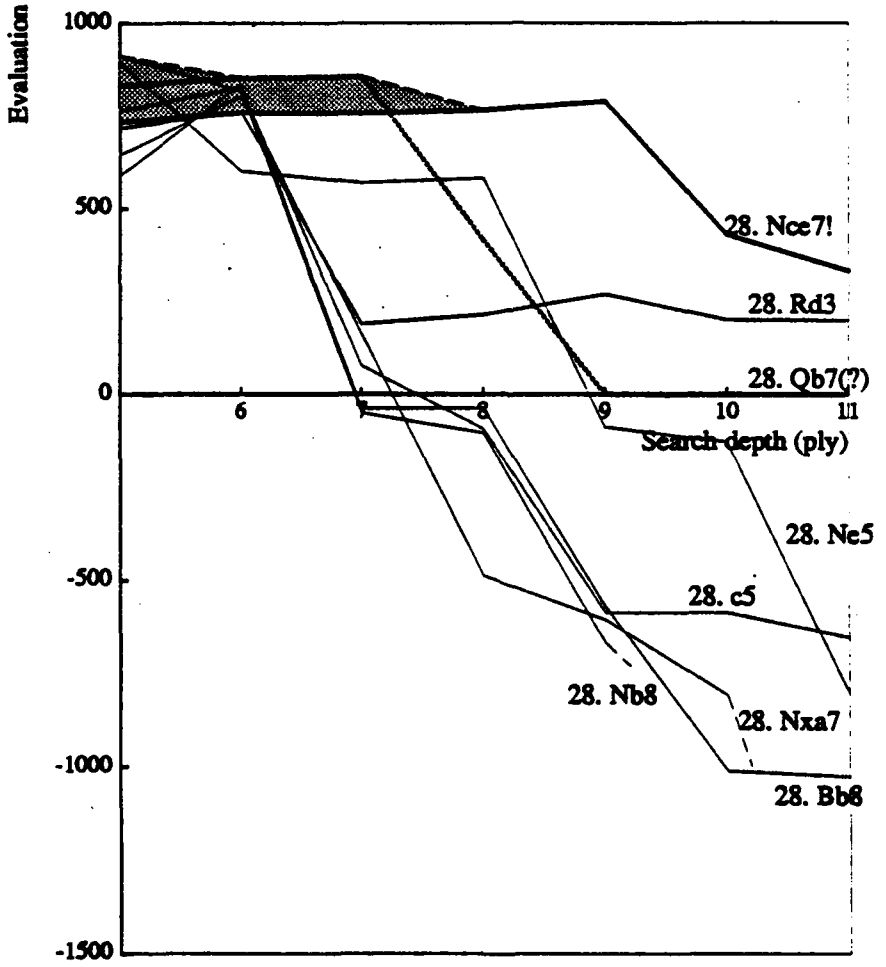

Figure 8.4: Evaluation as a function of search depth for several candidate moves on move 28 for the game Ivanchuk–Yusupov (game 10 of their 1991 match). The thick solid line indicates the best move, the dotted line the move actually chosen. The dashed line is the current best value in the course of the search. The shaded area indicates the confusability of the best move with other moves.

is rated first only from depth 8 on. The shaded area gives an indication of 'confusability' of the best move with other moves.

In this case, the minimax value of the position is positive for White (about +300). Let us say that the probabilities are computed based on the values of each of the candidate moves after a 6-ply search, and that the true values are 11-ply estimates. Depending on the exact computation of the probabilities, the speculative value for this positions could be zero or negative, i.e. could be advantageous for *Black*.

Whether Black should move to this position (from the position before the last move, i.c. 27. ... Re6), depends on the alternative moves available to him at that point. By comparing the 11-ply minimax values and the speculative values of these positions, he can assess the 'risk' and the speculative value of each move, and base his decision on this information.

In sum, similar techniques to the ones used for the KQKR endgame would be applicable to the complete game of chess, given suitable modifications. Of course, speculative play may not be applicable as often as for KQKR, and it is possible that better methods are available. This is an interesting area of future research.

8.4 Contributions of the thesis

The contributions of this thesis are mainly in the theory of games, and computer game playing.

The primary contributions are the following.

1. The thesis proposes **a new area of research on game-tree search**, namely the study of search algorithms and techniques in the presence of some knowledge of the opponent.

2. The focus of this thesis was on a **demonstration of the potential utility of such techniques, in an actual game-playing situation against human players.** The experiments in the KQKR domain provide strongly suggestive evidence towards this claim, although more experimental evidence and an improved experimental design would be necessary for a statistically conclusive proof.

3. The thesis also presents the foundations of **a novel theory of sequential, 2-player, zero-sum games**, based on a random-game model. This theory provides an explanation of several interesting properties and characteristics of actual games, for which no such explanation has been proposed before. In addition, it will allow a mathematical analysis of up till now rather elusive concepts, such as the intrinsic difficulty or 'importance' of positions in a game domain.

Other contributions of this thesis serve to provide supporting evidence to the main hypotheses of this thesis.

1. An analysis of the utility of common human heuristics, as a function of the distance to win. This analysis shows that the utility of heuristics strongly depends on the depth in the KQKR endgame. Some common heuristics are only beneficial in positions close to the win.

2. A demonstration and characterisation of situations in which a non-optimal (i.e. "speculative") strategy is better against certain opponents than a game-theoretically optimal one, in the context of a simplified model of game playing. This analysis shows that the 'best move' is context-dependent.

3. A database interface program for KQKR, easily extendable to other domains. The program allows the selection of best moves according to several criteria, provides info on estimated difficulty, and allows many useful database manipulation operators.

4. Criteria for measuring the quality of non-optimal play in a domain for which perfect knowledge is available. These criteria were used to analyse the quality of human play in games from the literature, as well as in the experiments conducted for this thesis.

5. An analysis of human play in the KQKR endgame, and a convincing demonstration that even this small 4-piece endgame is very difficult for people.

Chapter 9

Further work

The results described in this thesis are only a small step towards understanding the factors that contribute to errors in game playing, and in when and how knowledge of an opponent can or should be used in game playing or game-tree search.

The two main parts of this thesis each suggest a multitude of potential extensions and follow-ups. In this chapter, some of these will be given, in section 9.1 for the theoretical analysis of random games, and in section 9.2 for work with endgame databases.

In this thesis, many related issues and problems had to remain on the sidelines. Some of these are given in section 9.3

9.1 Random games

Based on the Optimal-Solution graph construct, one can try to answer several interesting questions by a mathematical or statistical analysis. Following are a few possibilities.

9.1.1 "Important" positions

When the domain is too big to be completely stored in memory, but also too irregular and too deep to be solved with forward search, it is interesting to ask what can be obtained with a combination of both. Presumably the first researcher to consider this possibility was Samuel [Samuel 1958] when treating rote-learning in checkers. The fact that selected positions are stored in a database with some (possibly approximate value) allows the search both to go deeper (in the same time) and to obtain more accurate results. The problem is that it is not clear which positions are most useful to store.

In the context of this paper, I would like to put forward a few possible characteristics of such "important" positions. One criterion is the number of optimal paths that pass through a position, as a position which is a key point of many winning lines may be a worthwile subgoal to work towards, and is more likely to show up in many lines of a forward search (provided that it starts from a winning positions to begin with). Another criterion is the expected difficulty of a position. Indeed it is not necessary to store positions that conform to the general rule, but only those that are exceptions to it (see for example the relevance of endgame studies in chess).

9.1.2 Difficulty in random games

Many properties can be derived from the depth distribution of winning positions, including several characteristics related to the difficulty of playing the position (for instance, the expected number of optimal moves s a function of depth). Since random games appear to provide good models of actual games, a study of difficulty in the relatively clean environment of random games may lead to insights into elements of difficulty in actual games as well.

Intuitively, we require the following characteristics of a function indicating the level of difficulty[1].

1. It is dependent on the *depth of the win*, i.e. a position which takes (with optimal play) a longer time to win is more difficult than a position that takes a shorter time.

2. It depends on the *difficulty of the successor states*, i.e. all other things being equal, a position with difficult successors is more difficult than a position with easy successors.

3. It depends on the *values of the successor states* and their distribution. For example, a position with only one winning move is harder than a position with many winning

[1] In this, potential factors in "real" games, such as specific board features, have again been excluded.

moves. Or, in a drawn position, a position with only one way to draw (other moves losing) is harder than a position with many drawing moves.

These three characteristics are essentially related to (1) how much time will be required to compute the exact decision in a given position, and (2) how likely it is that a random decision is a mistake either in the starting position or in the subsequent play.

Some other features may be included when more specific information is known about the exact nature of the opponent (his search engine, heuristics or knowledge). For example, we could have a good estimate of the depth of search an opponent can attain. Unless further specified, I shall limit myself to the three criteria given above.

There are two ways a formula could be put to use. One is to provide a ranking of positions according to the likelihood that a certain opponent will make a mistake. The other is to compute a different value for any given position in a game, based on the *expected outcome*, given the difficulty function.

Some of these ideas were carried out in the particular context of the KQKR endgame. It will be interesting to see to what extent they compare to a general analysis, and an analysis in the context of a game of a different class ('draw'), such as KRKN.

9.1.3 Zugzwang

Another topic that lends itself well to an investigation in random games is the topic of it Zugzwang (see also Section 3.1).

"Zugzwang" is a German word, meaning 'compulsion to move'. In particular, it denotes a situation where the player to move would rather pass (every move he has makes the position worse). In the abstract model of a game I have used so far, there is no such thing as Zugzwang, since positions with Black to move and positions with White to move are completely distinct, and 'passing', if legal, would just be the transition to another state (both players passing leads to a 2-cycle).

There are two reasons why Zugzwang is important to people, however. First, in almost all games, the position will *look* the same, no matter who has the move. The *features* (chunks) of the position are the same, and it will call forth very much the same information in both cases. Second, zugzwang positions are likely to contribute to the difficulty of a position (whether the zugzwang occurs at the root or deeper in the tree). The reason for this is that a correct evaluation of a position in which zugzwang plays a role disallows the use of planning techniques in which the two players are considered independently (e.g. "If I move A then I threaten B and then C. So he has to...").

How can we say something about zugzwang in the frame of random games? In order to introduce it we need a 1-1 correspondence of positions in the two component sets of G. The easiest way to do this is just to match the positions in each set in order (as the graph was generated randomly, this essentially a random mapping). We can then compute the optimal game graph, and count how many positions are lost with Black to move when the corresponding position with White to move is only a draw or wins more slowly.

As Zugzwang is important as a component of difficulty, it is worthwile understanding the occurence of this phenomenon. This may be one area, however, where random games match less well with actuall games, in which there are, after all, some geometric features that do influence the value of positions. As a case in point, the number of Zugzwang positions in the KQKR domain are only about 10% of the number predicted by an analysis of random games.

9.2 Human play and endgame databases.

Clearly, the analysis and experiments of the foregoing sections are only a small step on the way to a better understanding of games. The experiments indicate the potential of the new playing strategies described, but do not show this in the most forceful manner possible.

As an immediate follow-up to the work done in this thesis, multiple refinements and additions are possible, leading to better predictors, and more effective speculative play. A follow-up experiment, using some of these improvements, could more unambiguously demonstrate the validity of the strategies for speculative play.

9.2.1 Fine-tuning the heuristic programs

Selective application

The local structure of the KQKR domain is strongly dependent on the optimal depth to win, mainly because of the domain's statistical properties. Indirectly, the validity of certain heuristics, when people are more likely to make mistakes, and how often the opportunity for speculative play presents itself are also a function of the depth. It seems clear then, that the apropriateness of speculative play will also be depth dependent.

Selective application of speculative play (SP) would probably

- turn off SP for shallow depths, as people can then use search, and mistakes by the program would greatly accelerate the win.
- turn off SP for large depths, as one can expect people to make few and small errors (statistically and empirically).
- decrease the risk factor towards larger depths, as the expected magnitude of error is a decreasing function of depth.

Instead of a rigid depth dependence, a combination of difficulty and depth could be used.

In the programs used in Section 7, selectivity was used in a rudimentary fashion: only optimal moves were allowed for depths $D \leq 9$.

Expanding the predictor function

Only a few heuristics were used so far, in a fairly simplistic combination. Many obvious extensions are possible in terms of the kind and number of the heuristics and the way they are combined.

Little attention was paid to possible cutoffs and improvements in efficiency (in terms of machine cycles) in the heuristic programs. A more efficient algorithm might lead to deeper search and improved predictions.

Approximating the human evaluation function

To achieve the best possible predictions, one could use learning to construct an evaluation function for human play, for example by fitting a set of heuristics (features) with the games discussed in Section 6.1.3. As indicated in Section 1.4, an improved predictive ability would increase the expected utility of speculative play.

9.2.2 A follow-up experiment.

Motivation

One problem with the way the experiments were run is that it is hard to compare the performance of different subjects and the performance of the same subject against different programs. The reason is that positions vary widely in their characteristics, and some may be considerably easier than others. To make a more direct comparison possible both (1) between different subjects, and (2) between different programs, a new experiment could be carried out, as described below.

Relevant positions

Suppose we try to compare two programs, a control program (say **R**) and a heuristic program **HP**. In many positions, the move played by these two programs would be the same, in spite of the differences between them. Clearly, such positions do not contribute to our assessment of the relative merit of the two programs. It stands to reason, therefore, only to consider positions in which the decision is different. As **R** could pick any optimal move, we can limit ourselves to the situation where **HP** makes a non-optimal move, a mistake. Whether **HP** will perform better against human players is determined by whether the program succeeds in inducing human players to make, on the average, a mistake more than one move larger than the mistake after an optimal move would have been.

Description

1. **Subjects:** Preferably at least 5 or 6 subjects of Expert strength or better.

2. **Programs:** One reference program (possibly **R**), and one (or more) heuristic programs **HP**. It would be desirable, and seems entirely possible, to rather significantly improve the heuristic programs described in the previous section.

3. **Design:** *N relevant* KQKR positions (BTM) are selected randomly from specified depths. From these N positions, a test of $2N$ positions is derived by taking the positions resulting from the moves made by **R** and **HP** in the N selected positions[2].

[2]To diminish a possible learning effect in the subject, positions may be randomly rotated, or presented in a different order for different subjects.

The subjects are then presented with a (random) mix of these test positions, and asked to indicate, in each position, which move they would play, if this were a position in an actual game.

4. **Data:** For each position, the subject's move.

Results

An analysis of the experiment

1. finds the value of all the moves by each subject (i.e. the depth after their move).
2. computes the average of this value over subjects
3. compares the values for each of the two sets, to determine (1) in which case subjects make a larger mistake, (2) in which case the average depth is larger.

The analysis can be viewed from different viewpoints: globally, as a function of depth or as a function of characteristics of the subject.

9.2.3 Other work

After extending the analysis on KQKR, the obvious next step is to perform a similar experimental analysis for the KRKN endgame, and compare the results with KQKR. Larger endgames (say 5-piece chess endgames) may reveal some additional twists not apparent in the small endgames discussed here.

Another plausible direction would be a more theoretical analysis of speculative play and error behavior in endgame databases, and to try to extend this analysis as far as possible to the situation where the value is not known exactly for each position. Some ideas for the complete game of chess were given in section 8.3.

9.3 General directions

The area of research considered in this thesis, concerning the use of knowledge about an opponent in the search, opens up many new exciting avenues for further work. Here is a non-inclusive list of possibilities.

- **Asymmetric search algorithms.** The opponent is modeled as an entity with different characteristics. This may mean that the opponent is assumed to use a different evaluation function, or is expected to search less deeply. When suitable approximations are made, this area lends itself well to theoretical investigation. There are various issues in this direction, concerning
  - algorithms,
  - complexity,
  - comparison, in terms of efficiency and utility, with standard search algorithms, and
  - approximate algorithms, which, while achieving results close to the optimal, significantly increase efficiency.
- Issues of **predictive search.** Depending on the degree with which an opponent's move can be predicted a whole range of algorithms is possible between the two extremes of minimax and completely predictive search, which degenerates into puzzle search (1-player search).
- Using **dynamic search features** in existing chess programs. Often, the difficulty of a position, or the likelihood a chess player will be able to find the best move, can be predicted on the basis of dynamic features of the search process[23].
- Issues related to **identification of an opponent** (i.e. learning). How can one obtain crucial features of a chess players knowledge and 'algorithm' on the basis of his games? What different types of information are there? Which are most useful in the search?
- **History-dependent play.** In human chess playing, it is not sufficient to model the opponent's state of mind, it is also necessary to keep track of *changes* in this state. For instance, if I successfully employ one opening variation in one game against a certain opponent, this same line will be much less likely to succeed the next time.
- A further study of the components of **difficulty** of games and positions in those games for human players.

Bibliography

[1] B. Abramson, *The expected-outcome model of two-player games*, Ph.D. Thesis, Columbia University, 1987.

[2] Y.S. Abu-Mostafa, Random problems, *Journal of Complexity*, March 1988.

[3] I. Althöfer, Retrograde analysis and two computerizable definitions of the quality of chess games, *The ICCA Journal*, Vol. 12, no.2, June 1989.

[4] T.S. Anantharaman, M.S. Campbell and F.-H. Hsu, Singular extensions: adding selectivity to brute-force searching, *Artificial Intelligence*, Vol. 43, pp. 99–109, 1990.

[5] T.S. Anantharaman, M.S. Campbell, F.-H. Hsu and A. Nowatzyk, Deep Thought, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 55–78, Springer Verlag, 1990.

[6] T.S. Anantharaman, *A statistical study of selective min-max search in computer chess*, Technical Report CMU-CS-90-173, Ph.D. Thesis, Carnegie Mellon University, May 1990.

[7] V.L. Arlazarov and A.L. Futer, Computer analysis of a rook endgame, in *Machine Intelligence 9*, Hayes, Michie and Mikulich eds. Ellis Horwood Ltd., 1979.

[8] E. B. Baum, Minimax is not optimal for imperfect game players, submitted for publication, 1991.

[9] H.J. Berliner, On the construction of evaluation functions for large domains, *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pp. 53-55, 1979.

[10] H.J. Berliner and C. Ebeling, Hitech, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 79–110, Springer Verlag, 1990.

[11] W.G. Chase and H.A. Simon, The mind's eye in chess, In *Visual Information Processing*, Academic Press, Inc., 1973.

[12] E.R. Berlekamp, J.H. Conway and R.K. Guy, *Winning ways for your mathematical plays*, Academic Press inc., London 1982.

[13] D.F. Dansereau, *An information processing analysis of mental multiplication*, Master's Thesis, Carnegie Institute of Technology, June 1966.

[14] S.T. Dekker, H.J. van den Herik and I.S. Herschberg, Complexity starts at five, *The ICCA Journal*, Vol. 10, no. 3, September 1987.

[15] S.T. Dekker, H.J. v.d. Herik and I.S. Herschberg, Perfect Knowledge and Beyond, in *Advances in Computer Chess 5*, D.F. Beal, ed., pp. 295–312. North-Holland, 1989.

[16] F. Friedel, personal communication, 1/1992.

[17] R. Gasser, Applying retrograde analysis to Nine Men's Morris, In *Heuristic programming in Artificial Intelligence: the 2nd computer olympiad*, D. Levy and D.F. Beal, eds., 1990.

[18] J.A. Gaschnig, *Performance measurement and analysis of certain search algorithms*, Ph.D. Thesis, Carnegie Mellon University, 1979.

[19] G.J. Goetsch and M.S. Campbell, Experiments with the null-move heuristic in chess, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 159–168, Springer Verlag, 1990.

[20] D. Hartmann, Notions of evaluation functions tested against grandmaster games, In *Advances in Computer Chess 5*, D.F. Beal, Editor, pp. 91–142, North-Holland, 1989.

[21] R.M. Hyatt, A.E. Gower and H.L. Nelson, Cray Blitz, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 111–130, Springer Verlag, 1990.

[22] D. Hooper, *A pocket guide to chess endgames*, G.Bell & Sons Ltd., 1970; B.T.Batsford Ltd., London 1986.

[23] P. Jansen, Problematic positions and speculative play, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 169–181, Springer Verlag, 1990.

[24] P. Jansen and J. Schaeffer, Seconding a grandmaster, *The ICCA Journal*, Vol. 13, no. 1, March 1990.

[25] D.E. Knuth and R.W. Moore, An analysis of alpha-beta pruning, *Artificial Intelligence*, Vol. 6, no. 4, pp. 293–326, 1975.

[26] E.A. Komissarchik and A.L. Futer, Computer analysis of a queen endgame, *The ICCA Journal*, Vol. 9, no. 4, December 1986.

[27] D. Kopec and T. Niblett, How hard is the play of the King-Rook-King-Knight ending, In *Advances in Computer Chess 2*, M.R.B. Clarke, ed., pp. 57–81, Edinburgh University Press, 1980.

[28] D. Kopec, M. Newborn and W. Yu, Experiments in chess cognition, In *Advances in Computer Chess 4*, D.F. Beal, ed., pp. 59–79, Pergamon Press, 1986.

[29] R.E. Korf, Real-time single-agent search: first results, *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 1987.

[30] R.E. Korf, Real-time single-agent search: new results, *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, 1988.

[31] R.E. Korf, Generalized game-trees. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-89)*, 1989.

[32] K. Kotovsky, J.R. Hayes and H.A. Simon, Why are some problems hard? Evidence from the tower of Hanoi, *Cognitive Psychology*, Vol. 17, pp. 248–294, 1985.

[33] K. Kotovsky and H.A. Simon, What makes some problems really hard: explorations in the problem space of difficulty. *Cognitive Psychology*, Vol. 22, pp. 143–183, 1990.

[34] D. Levy, Improving the performance of endgame databases, *The ICCA Journal*, Vol. 10 no. 4, December 1987.

[35] D. Levy and M. Newborn, *How computers play chess*, W. H. Freeman and co., New York, 1991.

[36] R.D. Luce and H. Raiffa, *Games and decisions*, John Wiley & Sons, Inc., 1958.

[37] T.A. Marsland, Computer chess methods, In *Encyclopedia of Artificial Intelligence*, Wiley & Sons, Inc., 1987.

[38] D.A. McAllester, Conspiracy numbers for min-max search, *Artificial Intelligence*, Vol. 35, pp. 287–310, 1988.

[39] J. McCarthy, Chess as the drosophyla of AI, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 227–237, Springer Verlag, 1990.

[40] D. Michie, Information and complexity in chess, In *Advances in Computer Chess 3*, M.R.B. Clarke, ed., pp. 1–16, Pergamon Press, 1982.

[41] D. Michie and I. Bratko, Ideas on knowledge synthesis stemming from the KBBKN endgame, *The ICCA Journal*, Vol. 9, no. 1, March 1986.

[42] D. Michie, Evaluative comments in chess, Chapter 4 in *On Machine Intelligence*, pp. 44–60, Ellis Horwood, Ltd., 1986.

[43] D. Michie, Computable sub-games of chess, Chapter 5 in *On Machine Intelligence*, pp. 61–76, Ellis Horwood, Ltd., 1986.

[44] D. Michie, Brute force in chess and science, In *Computers, Chess and Cognition*, T.A. Marsland and J. Schaeffer, eds., pp. 239–257, Springer Verlag, 1990.

[45] D.S. Nau, Decision quality as a function of search depth on game trees, *Journal of the ACM*, Vol. 30, no. 4, pp. 687–708, October 1983.

[46] D.S. Nau, Pathology on game trees revisited, and an alternative to minimaxing, *Artificial Intelligence*, Vol. 21, pp. 221–244, 1983.

[47] A. Newell, J.C. Shaw, and H.A. Simon, Chess-playing programs and the problem of complexity, *IBM Journal of Research and Development*, Vol. 2, pp. 320–335, 1959. Reprinted in *Computers and Thought* by Feigenbaum and Feldman.

[48] A. Newell and H.A. Simon, *Human Problem Solving*, Prentice Hall Inc., Englewood Cliffs, N.J., 1972.

[49] L. Pachman, *Chess endings for the practical player*, (Otto Hardy, transl.), (John Littlewood, ed.), Routledge & Kegan Paul plc, London 1983 (translation), Wilhelm Heyne Verlag, 1977 (original).

[50] A.J. Palay, *Searching with Probabilities*, Ph.D. Thesis, Carnegie Mellon University, 1983.

[51] J. Pearl, On the nature of pathology in game searching, *Artificial Intelligence*, Vol. 20, pp. 427–453, 1983.

[52] J. Pearl, *Heuristics*, Addison-Wesley Publishing Co., 1984.

[53] A.L.Reibman and B.W. Ballard, Non-minimax strategies for use against fallible opponents, *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, 1983.

[54] J. Rosenschein, *Rational interaction: cooperation among intelligent agents*, Ph.D. Thesis, Stanford University, 1985.

[55] J. Roycroft and D.F. Beal, To make dumb endgame databases speak, in *Advances in Computer Chess 6*, D.F. Beal, ed., pp. 149–159, Ellis Horwood Ltd., 1991.

[56] J. Roycroft, Expert against Oracle, in *Machine Intelligence 11*, Hayes, Michie and Richards eds., Clarendon Press, Oxford 1988.

[57] S. Russell and E. Wefald, Decision-theoretic control of reasoning: general theory and application to game-playing, Technical Report no. UCB/CSD 88/435, University of California / Berkeley, October 1988.

[58] T.A. Marsland, and J. Schaeffer, eds., *Computers, Chess, and Cognition* Springer Verlag, New York, 1990.

[59] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu and D. Szafron, A world championship caliber checkers program, *Artificial Intelligence*, Vol. 53, pp. 273–289, 1992.

[60] G. Schrüfer, Presence and absence of pathology on game trees, in *Advances in Computer Chess 4*, D.F. Beal, ed., pp. 101–112, Pergamon Press, 1986.

[61] R. Seidel, What constitutes optimal play?, *The ICCA Journal*, Vol. 9, no. 1, March 1986.

[62] C.E. Shannon, Programming a computer for playing chess, *Philosophical Magazine*, Vol. 41, no. 7, pp. 256–275, 1950.

[63] H.A. Simon and P.A. Simon, Trial and error search in solving difficult problems: evidence from the game of chess, *Behavioral Science*, Vol. 7, no. 4, October 1962.

[64] H.A. Simon, Style in design, In *Proceedings of the 2nd Annual Design Research Association Conference*, 1971.

[65] H.A. Simon and W.G. Chase, Skill in chess, *American Scientist*, Vol. 61, no. 4, pp. 394–403, July-August 1973.

[66] H.A. Simon, The psychological concept of "losing move" in a game of perfect information, *Proc. Nat. Acad. Sci. USA*, Vol. 71, no. 6, pp. 2276–2279, June 1974.

[67] H.A. Simon and J.B. Kadane, Optimal problem-solving search: all-or-none solutions, *Aritificial Intelligence*, Vol. 6, pp. 235–247, 1975.

[68] H.A. Simon and J. Schaeffer, The game of chess, in print.

[69] J. Spencer, Ten lectures on the probabilistic method, CBMS-NSF Regional Conference Series in Applied Mathematics, no. 52, Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[70] L. Stiller, Group graphs and computational symmetry on massively parallel architecture, *The Journal of Supercomputing*, Vol. 5, pp. 99–117, 1991.

[71] T. Ströhlein, Untersuchungen über kombinatorische Spiele, Dissertation, Fakultät für Allgemeine Wissenschaften der Technischen Hochschule München, 1970.

[72] K. Thompson, Retrograde analysis of certain endgames. *The ICCA Journal*, Vol. 9, no. 3, September 1986.

[73] K. Thompson, personal communication 1/1992.

[74] A.M. Turing and C.S. Strachey, Digital computers applied to games, in *Faster than Thought*, B.V. Bowden, ed., Sir Isaac Pitman, London, 1953.

[75] H.J. van den Herik and I.S. Herschberg, A database on databases, *The ICCA Journal*, Vol. 9, no. 1, March 1986.

[76] J. von Neumann, and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[77] D.E. Wilkins, Using knowledge to contol tree searching, *Artificial Intelligence*, Vol. 18, pp. 1–51, 1982.

[78] Zermelo, E., On the applications of the theory of sets to the theory of chess games, in *Matrix Games*, Moscow, Fizmatgiz, 1961, pp 167–172 (as quoted in [26]).