Chapter 1

Introduction

This thesis investigates how selective-search methods can improve the performance of a game program for a given domain. Selective-search methods aim to explore only the profitable parts of the state space, but take the risk to overlook the best move. We propose several selective-search methods and test them in a large number of game domains. The domains consist of deterministic one-, two- and multi-player games with perfect information, and two-player non-deterministic or imperfect-information games.

In this chapter we provide a brief introduction on games research (Section 1.1) and discuss selective search for games in Section 1.2. Next, we formulate the problem statement which guides our research, together with four research questions (Section 1.3). Finally, we provide an overview of this thesis in Section 1.4.

1.1 Games

Since thousands of years, games are a phenomenon across human cultures, where people display intelligence, interaction and competition (Huizinga, 1955; Bell, 1980). But games are also an important theoretic paradigm in logic, AI, computer science, linguistics, economics, biology, and increasingly also in social sciences and social psychology. Games can be classified according to different dimensions. Five classifications are (1) the number of players (one-, two-, multi-player), (2) whether chance is involved (deterministic or non-deterministic), (3) how much information a player has (perfect or imperfect information), (4) which time system is used (turn-based or real-time), and (5) the decision space (discrete or continuous). Dimensions (4) and (5) are relevant for video games which are beyond the scope of the thesis. For the remainder of this section, games are turn-based and discrete, if not mentioned otherwise.

What makes games of particular interest is their hybrid character. On the one hand, the rules of the game are well-defined, the states are easy to represent and the possible actions are known, but on the other hand, games allow complex behavior and reasoning. One example of a simple game which still forms a challenge is the L-Game (De Bono, 1968), where players are moving an L-shaped piece on a 4×4

board, trying to block the opponent from moving. Therefore, games are an ideal AI testbed for better understanding intelligent human behavior including decision-making, intentions, strategies, and interaction. Study of these topics has a rich tradition in logic and computer science. Here, cognition meets computation: games challenge humans by their difficulty, and the study of machines by humans highlights basic issues of complexity.

The outline of this section is as follows. In Subsection 1.1.1 we describe some research performed on games from the mathematical and logical viewpoint. Afterwards, in Subsection 1.1.2, the computational approach to games is discussed.

1.1.1 Game Theory

Game theory can be defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers (cf. Myerson, 1997; Myerson, 1999; Peters, 2008). Mathematical game theory provides general techniques for analyzing game-like situations in which two or more players make decisions that will influence one's another's welfare. It has a wide range of applications, but is most prominently seen in economics. Pioneering work has been done by Zermelo (1913), Borel (1921) and Von Neumann and Morgenstern (1944). In this field, several research efforts were successful to the point of giving rise to research areas of their own, of which we want to introduce two. (1) Game-theoretic semantics for logics (Lorenz and Lorenzen, 1978) aim at understanding a game on a conceptual level. Meaning is given to the components of the game, such as the number of players, the goal of a player and the interaction between the players. Depending on the game, different types of logic may be used, such as modal logic, epistemic logic and dynamic-epistemic logic (Van Benthem, 2001). (2) Combinatorial game theory (CGT) is a mathematical theory that studies finite two-player deterministic games with perfect information, such as chess, checkers, Go, Domineering, Dots-and Boxes, Nim, and many others (Nowakowski, 2009). Berlekamp, Conway, and Guy (1982) use a stricter definition of a combinatorial game, which adds that a player unable to move loses the game (*normal play*) and that no draws exist because of move repetition (*ending condition*) (Guy, 1996). The aim of CGT is to find the algebraic structure of the game such that the optimal play for both players can be determined. Additionally, this field of research is concerned with the complexity of games. A game falls into a certain complexity class, depending on how fast the problem grows when the input size is varied. Two of these classes are called PSPACE-complete and EXPTIME-complete. Many games are proven to be PSPACE-complete (Papadimitriou, 1994; Van Emde Boas, 2002), such as Go-Moku (Reisch, 1980), Hex (Reisch, 1981), Othello (Iwata and Kasai, 1994), Scotland Yard (Sevenster, 2006), Connect6 (Hsieh and Tsai, 2007) and Amazons (Hearn, 2009). Games that are proven to be EXPTIME-complete include Chinese Checkers (Kasai, Adachi, and Iwata, 1979), chess (Fraenkel and Lichtenstein, 1981), Go (Robson, 1983), checkers (Robson, 1984) and Shogi (Adachi, Kamekawa, and Iwata, 1987). For an overview of a large number of games and their complexity classes we refer to Demaine and Hearn (2001).

1.1.2 Games and AI

Since the dawn of computer technology, AI researchers have let computers play games as a testing ground for computational intelligence. The AI research in games obtained an impulse in 1944 when Von Neumann republished his article about the minimax algorithm (Von Neumann, 1928) together with Morgenstern in the book "Theory of Games and Economic Behavior" (Von Neumann and Morgenstern, 1944). These ideas were continued by Shannon (1950), Turing (1953), and Newell, Shaw, and Simon (1958) who wrote the first articles about how computers could play chess as intelligently as possible.

A popular research area in the field of games and AI concerns two-player deterministic perfect-information games. A classic example of these games is chess. A great deal of effort has been devoted in the past to construct a world-champion-level computer chess player. The most prominent success so far in this area was the result when Deep Blue defeated the world chess champion Garry Kasparov in 1997 (Hsu, 2002). The next step would be to solve the game (i.e., knowing the game-theoretic value), but Chess will not be solvable in a foreseeable future according to Levy and Newborn (1991). Not long ago, another breakthrough was achieved by Schaeffer *et al.* (2007), who solved the game of checkers. A forward search enhanced with endgame databases proved that if both players play optimally the game ends in a draw. For the game of Go, programs are not on grandmaster level yet. However, in 2009 a 9-dan Go professional was beaten by the program MoGo with "only" a 7-stone advantage.[1]

In *non-deterministic* games[2] an element of chance is involved. A well studied non-deterministic two-player game is Backgammon, in which computers are stronger than humans since the 1990's (Tesauro, 1994). One of the most famous programs, TD-Gammon by Gerald Tesauro, employs neural networks (Fukushima, 1975) and temporal-difference learning (Sutton, 1988) to achieve a high level of play. *Imperfect-information* games hide information from the players. Examples of imperfect-information games are the chess-variant Kriegspiel and the Go-variant Phantom Go, in which the pieces of the opponent are hidden. In Kriegspiel, the strongest programs use metapositions during the search, but they cannot beat the best humans (Ciancarini and Favini, 2007). In Phantom Go, the strongest programs are based on Monte-Carlo methods, and are able to play on an experienced level (Cazenave, 2006; Borsboom *et al.*, 2007). Scrabble[3] is a game with both non-determinism and imperfect information. The program Maven uses four different search engines for each phase of the game (i.e., opening, middle game, pre-endgame and endgame) with a selective move generator, a well-tuned evaluation function and Monte-Carlo simulations to beat all human Scrabble experts (Sheppard, 2002a; Sheppard, 2002b).

One-player games are essentially optimization problems, of which two famous examples are the 15-puzzle (Korf, 1985) and the Rubik's Cube (Korf, 1997). These

[1] The game was played at the Taiwan Open 2009 and can be downloaded from: http://go.nutn.edu.tw/2009/English/result_eng.htm

[2] These games may also be called stochastic games, or games with chance.

[3] Scrabble® is a registered trademark. All intellectual property rights in and to the game are owned in the U.S.A and Canada by Hasbro Inc., and throughout the rest of the world by J.W. Spear & Sons Limited of Maidenhead, Berkshire, England, a subsidiary of Mattel Inc. Mattel and Spear are not affiliated with Hasbro.

games are typically tackled with the A* algorithm (Hart, Nielson, and Raphael, 1968). One-player games may also be non-deterministic, such as Tetris (Demaine, Hohenberger, and Liben-Nowell, 2003). In Tetris, an evaluation function tuned by the Cross-Entropy Method (Rubinstein, 2003) is able to remove more than 35 million lines on average (Thiery and Scherrer, 2009). Per definition, deterministic one-player games with imperfect information do not exist. In order to have imperfect information either an element of chance should be present (e.g., shuffling of cards) and/or a second player is required (e.g., choosing a hidden card of the second player), which would contradict the notion of a one-player deterministic game. A well-known one-player non-deterministic game with imperfect information is Klondike Solitaire.[4] Here, most research focuses on the solvability of Klondike Solitaire with an initial setup of cards. Bjarnason, Tadepalli, and Fern (2007) demonstrate empirically that no less than 82% and no more than 91.44% of Klondike Solitaire games have winning solutions.

As the name already indicates, in multi-player games more than two players participate in the game. Chinese Checkers is a well-known deterministic perfect-information game in this category, which has received quite some attention from researchers (Sturtevant, 2003a). Chinese Checkers has been used as test domain for tree-search methods with coalition forming (Sturtevant and Korf, 2000) as well as for Monte-Carlo methods (Sturtevant, 2008a). Examples of multi-player games with chance are Monopoly[5] and Ludo (Carter, 2007). Visiting each square on the Monopoly board may be modeled as a Markov chain (Ash and Bishop, 1972) and evolutionary algorithms can be used to learn strategies (Frayn, 2005). Siguo (Xia *et al.*, 2005; Lu and Xia, 2008) and Scotland Yard (Sevenster, 2006) are multi-player games with imperfect information. For Siguo, Bayesian networks may be used in which beliefs are propagated (Xia, Zhu, and Lu, 2007). In Scotland Yard, Doberkat, Hasselbring, and Pahl (1996) use a multi-agent system to establish cooperation between the agents. Many multi-player card games have both non-determinism and imperfect information, such as Poker (Billings *et al.*, 1998a; Billings *et al.*, 1999) and Bridge (Ginsberg, 1999). Here, opponent modeling plays a central role (Jansen, 1992; Carmel and Markovitch, 1993; Iida *et al.*, 1994; Billings *et al.*, 1998b; Donkers, 2003).

Table 1.1 gives examples of games for each classification. We remark that a large number of multi-player games may also be played by only two players. Depending on the class of a game, different methods are necessary to create a strong AI. Taking the AI perspective into account, it is possible to use different classes. An example of an alternative classification is the theme of the game, which has a large influence for constructing an evaluation function. The theme of a game can be material (chess), territory (Go), connection (Hex), or racing (Chinese Checkers).

Finally, we mention that the field of modern video games has become increasingly prominent in games research over the last decades (cf. Bakkes, 2010). These games typically are of synchronous and continuous nature which means that a fast decision has to be made in real-time for an infinite[6] action space. This makes the creation

[4] It is just called Solitaire on the Microsoft Windows operating system.
[5] Monopoly® is a registered trademark of Hasbro, Inc. All rights reserved.
[6] or several magnitudes larger than in abstract games

Table 1.1: Games of different classifications.

	perfect information	
Players	deterministic	non-deterministic
1	15-puzzle	Tetris
2	chess, Go	Backgammon
>2	Chinese Checkers	Ludo, Monopoly
	imperfect information	
Players	deterministic	non-deterministic
1	-	Klondike Solitaire
2	Kriegspiel, Phantom Go	Two-Player Scrabble
>2	Siguo, Scotland Yard	Poker, Bridge

of video game AI a challenging task. In modern games, the intelligent behavior of game characters is often established on the basis of hacks and cheats (Millington, 2006). When observing this kind of behavior, human players often assume intelligence where none exists (Crawford, 1984; Lidén, 2004). They appreciate if the game AI maintains the illusion of being intelligent (Scott, 2002). The research on video game AI has moved from rule-based systems (Nareyek, 2000; Tozour, 2002), to goal-oriented behavior (Millington, 2006) and adaptive game AI (Spronck, 2005; Bakkes, 2010).

1.2 Selective Search

AI researchers have proposed a large number of computational methods for game playing. Applying search methods to turn-based games has been successful for many years. The most prominent search method is the *minimax* algorithm (Von Neumann, 1928), enhanced with $\alpha\beta$ *pruning* (Knuth and Moore, 1975). This algorithm searches through the state space of the game until a fixed depth is reached and evaluates the desirability of game states on the search horizon. $\alpha\beta$ pruning detects states which do not have an influence on the value of the root and prunes these states safely, increasing the search speed. $\alpha\beta$ is a form of backward pruning because a refutation is found after a move is searched. Minimax often forms the basis for other algorithms as well. Examples are *expectimax* (Michie, 1966), which introduces chance nodes to model chance events in non-deterministic games, and the *paranoid* algorithm (Sturtevant and Korf, 2000) for multi-player games.

When a computer program has to decide which move to play, it is generally limited by a time constraint. Thus, the program should utilize the given time as well as possible. In a traditional $\alpha\beta$ program with iterative deepening, the search depth is iteratively increased until time runs out. This is an iterative *depth-first search*. If domain knowledge is available, there often exists an indication whether a move is strong or weak (e.g., capturing or losing a queen in chess). Using this knowledge, *variable-depth search* can be performed (Marsland and Björnsson, 2001) that uses $\alpha\beta$ search as foundation. Variable-depth search consists of two parts: (1)

forward pruning and (2) search extensions. If a move is regarded as unlikely based on some criteria, it can be pruned. This is called forward pruning, because the decision to skip a move is made in advance. In contrast, moves which seem promising can be granted additional search depth, which is called a search extension.

While variable-depth search uses more time for strong moves and considerable less time for weak moves, the nominal search depth still has to be finished before a deeper search starts. Other approaches try to spend their time on the most interesting part of the search tree at any point of time. If nodes on the search horizon are ordered so that the one with the best evaluation is expanded first, the resulting strategy is called *best-first search* (Russell and Norvig, 2003). The search tree created by best-first search is typically narrow and deep. An example is *Proof-Number (PN) search* (Allis, Van der Meulen, and Van den Herik, 1994). PN search is a (mate-)solver which tries to prove or disprove a certain goal for the root node. This algorithm repeatedly selects the most-proving node for expansion. Therefore, PN search is always spending its time on the most interesting part of the tree. A disadvantage of PN search is that it can only be used for solving. The played move is weak if it is chosen before the position has been solved.

*SSS** (based on state-space search) is another form of best-first search of which it is proven that it expands fewer nodes than $\alpha\beta$ (Stockman, 1979). SSS* is an effective technique if no good move ordering is available because multiple solution trees are investigated at the same time. A disadvantage is, however, that a list of nodes has to be sorted and kept in memory. The additional memory requirements and sorting effort may be an overhead if a good move ordering is available.

A different type of best-first search that has become increasingly popular for letting computers play games, is *Monte-Carlo Tree Search* (MCTS) (Kocsis and Szepesvári, 2006; Chaslot *et al.*, 2006b; Coulom, 2007a). Instead of using a heuristic evaluation function to guide the search, MCTS utilizes Monte-Carlo simulations. MCTS balances exploration and exploitation to spend most effort on the best move, while making sure that alternatives are not overlooked. Recently, this technique has revolutionized computer Go (Coulom, 2007a; Gelly and Silver, 2007; Chaslot *et al.*, 2008d). MoGo and Crazy Stone, the two pioneering MCTS programs, managed to defeat professional players with less than a 10-stone handicap for the first time in history (Chaslot *et al.*, 2008a). MCTS has also been applied successfully in quite some other two-player games such as Amazons (Lorentz, 2008; Kloetzer, Iida, and Bouzy, 2009), Hex (Cazenave and Saffidine, 2009; Arneson, Hayward, and Henderson, 2010) and Lines of Action (LOA) (Winands and Björnsson, 2010).

Thus, current research is not focused on the traditional iterative depth-first approach anymore, but on approaches which spend the majority of thinking time on the parts of the tree which are the most profitable. If the search method selectively investigates the state space of a game with a risk of overlooking the best move, we call it a *selective search* (cf. Marsland, 1986; Buro, 1995; Björnsson, 2002).

1.3 Problem Statement and Research Questions

In the previous section we discussed the relevance of selective search to improve computer game play. That is precisely the topic of this thesis. The following problem statement guides our research.

> **Problem statement**: *How can we improve selective-search methods in such a way that programs increase their performance in domains of different complexity?*

Rather than testing selective-search methods on one class of games, we chose different classes of games, which all have to be addressed differently. Each class of games represents a level of complexity. Between every level there exists a *complexity jump*. With a complexity jump the complexity of the game increases significantly because the mechanism of the game is changed (Van Emde Boas, 2003) (e.g., a player, chance, or imperfect information is added). We have chosen five different levels of games, resulting in four complexity jumps. (1) One-player games, or puzzles, involve no opponent and are a testbed for planning algorithms. (2) Two-player games are the classic testbed for search methods. We use them for investigating mate-solvers. For testing search with chance nodes, (3) non-deterministic and (4) imperfect-information games may be used. (5) Multi-player games are a testbed for dealing with coalition forming.

We formulate four research questions to guide our research. Each one deals with search for a different class of games and a different selective-search method. The four research questions address (1) MCTS, (2) PN search, (3) expectimax, and (4) multi-player search.

> **Research question 1**: *How can we adapt Monte-Carlo Tree Search for a one-player game?*

The traditional approach to deterministic one-player games with perfect information is applying A* or IDA*. These methods have been quite successful in coping with this class of games. The disadvantage of these methods is that they require an admissible heuristic evaluation function. The construction of such a function can be difficult. Since Monte-Carlo Tree Search (MCTS) does not require an admissible heuristic, it may be an interesting alternative. Therefore, we investigate which modifications are required to apply MCTS successfully to a deterministic one-player game with perfect information. We call the resulting variant *Single-Player Monte-Carlo Tree Search* (SP-MCTS) and test it in the puzzle SameGame.

> **Research question 2**: *How can we solve a two-player game by using Proof-Number search in combination with endgame databases?*

Ideally, a search method is able to prove that a move is the optimal one for a given game. The game is solved if this is achieved. In the last years quite some deterministic two-player games with perfect information have been solved (e.g., Connect-Four (Allis, 1988), Nine Men's Morris (Gasser, 1996) and checkers (Schaeffer *et al.*, 2007)). A selective-search method specially designed as mate-solver is Proof-Number (PN)

search (Allis *et al.*, 1994). PN search is efficient in searching game trees with a non-uniform branching factor. Endgame databases (Ströhlein, 1970; Van den Herik and Herschberg, 1985) proved to be vital to the strength of computer programs in quite a number of games (Schaeffer, 1997; Heinz, 1999). Schaeffer *et al.* (2007) used a combination of proof-tree manager, the $\alpha\beta$ based program CHINOOK, the PN-search variant Df-pn (Nagai, 2002) and endgame databases to solve the game of checkers. We investigate to which degree endgame databases are able to enhance the PN-search variant PN^2 for the deterministic two-player perfect-information game Fanorona. The complexity of Fanorona is similar to the complexity of checkers.

> **Research question 3**: *How can we perform forward pruning at chance nodes in the expectimax framework?*

Another form of searching selectively in two-player deterministic games with perfect information is variable-depth search (Marsland and Björnsson, 2001). Branches can be pruned if they seem unpromising (forward pruning), or extended if the branches are promising (search extensions). There exist several successful forward-pruning techniques for the $\alpha\beta$ algorithm (Beal, 1989; Buro, 1995; Buro, 2000; Björnsson and Marsland, 2001). For two-player games with non-determinism or imperfect information expectimax may be used (Michie, 1966). Expectimax adds chance nodes to the search tree. There are, however, no forward-pruning techniques for chance nodes. We propose a new forward-pruning technique based on *ProbCut* (Buro, 1995), called *ChanceProbCut*, and test it in two-player games with either non-determinism or imperfect information.

> **Research question 4**: *How can we improve search for multi-player games?*

In deterministic two-player games with perfect information, the majority of research focused on the $\alpha\beta$ algorithm (Knuth and Moore, 1975). For deterministic multi-player games with perfect information, the choice of algorithm is not as straightforward. The two main algorithms are called *max*^{n} (Luckhardt and Irani, 1986) and *paranoid* (Sturtevant and Korf, 2000), both approaching the problem from a different angle. Max^n assumes that every player tries to maximize the own score, while paranoid assumes that all opponents form a coalition against the root player. However, these assumptions have drawbacks. Due to the lack of safe pruning in max^n only a limited lookahead is possible (Sturtevant, 2003b). Furthermore, the underlying assumption of max^n may be unrealistic, resulting in max^n to be too optimistic (Zuckerman, Felner, and Kraus, 2009). When searching deep with the paranoid algorithm, the other players may dominate the root player (Saito and Winands, 2010), resulting in paranoid to be too pessimistic. We propose a new search method, called *Best-Reply Search* (BRS) that avoids these drawbacks. This method does not allow every opponent to make a move, but only the one with the strongest counter move.

1.4 Thesis Overview

The thesis structure is as follows. Chapter 1 provides an introduction to the research presented in this thesis.

Chapter 2 is a general introduction to search methods for games. It explains the minimax algorithm and the well-known $\alpha\beta$ search. Standard techniques for enhancing the $\alpha\beta$ search are discussed as well. We furthermore explain Monte-Carlo Tree Search (MCTS) and its enhancements.

In Chapter 3 we answer the first research question. The chapter explains the rules of the test domain, the puzzle SameGame, and thereafter discusses the modifications made to MCTS for deterministic one-player games with perfect information. The new algorithm is called Single-Player Monte-Carlo Tree Search (SP-MCTS). We provide experiments on tuning the parameters and show that randomized restarts can improve the score. Experiments reveal that SP-MCTS is able to achieve a high score on the standardized test set.

Chapter 4 starts with explaining the rules of Fanorona, a deterministic perfect-information two-player game. Next, an analysis of this game is presented. Subsequently, the construction of omniscient endgame databases is described. The mate-solver Proof-Number (PN) search and its two-level variant PN^2 are explained thereafter. The chapter concludes with the experiments that resulted in solving Fanorona and its board variants. This chapter answers the second research question.

Next, in Chapter 5, expectimax and its pruning techniques, Star1 and Star2, are explained. We give an introduction to forward pruning and propose a new forward-pruning method for chance nodes, ChanceProbCut. We describe three two-player test domains, the games of Stratego (with imperfect information), Dice (non-deterministic) and ChanceBreakthrough (non-deterministic) and finish with experiments in each of these games. Chapter 5 gives an answer to the third research question.

In Chapter 6 we first give an introduction to the two main algorithms for deterministic multi-player games with perfect information, $\max^n$ and paranoid. We present a new algorithm for deterministic multi-player games with perfect information, called Best-Reply Search (BRS). Thereafter, Chinese Checkers, Focus, and Rolit are introduced as test domains. Next, experiments and a discussion are given. This chapter answers the fourth research question.

Chapter 7 summarizes the answers to the research questions and answers the problem statement. We also give directions for future research.