Summary

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.

In Chapter 1 we provide a brief introduction on games research and discuss selective search for games. 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. 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) Monte-Carlo Tree Search (MCTS), (2) Proof-Number (PN) search, (3) expectimax, and (4) multi-player search.

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 MCTS and its enhancements.

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 the selective-search method MCTS does not require an admissible heuristic, it may be an interesting alternative. This has led us to the first research question.

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

Chapter 3 answers the first research question by proposing a new MCTS variant called Single-Player Monte-Carlo Tree Search (SP-MCTS). MCTS is adapted by two modifications resulting in SP-MCTS. The modifications concern (1) the selection strategy and (2) the backpropagation strategy. The selection strategy is adapted to take the standard deviation of the scores for a move into account and to combine the average score with the top score. The backpropagation strategy is modified to provide the selection strategy with all required information. For evaluating SP-MCTS, the puzzle SameGame is chosen as test domain. So far, there does not exist a good admissible heuristic evaluation function for this game.

On the standardized test set of 20 SameGame positions, the manually tuned SP-MCTS method, which invests all search time at the initial position, scored 73,998 points. This was the highest score on the test set at that point of time (2008). The main contribution is therefore that we successfully adapted MCTS for a one-player game. Inspired by our approach, two other Monte-Carlo-based approaches, Nested Monte-Carlo and Heuristically Guided Swarm Tree Search, broke our record subsequently. At the time of publishing this thesis SP-MCTS, with parameters tuned by the Cross-Entropy Method and with time equally distributed over the consecutive positions, scored 78,012 points on the test set, which is currently the third highest score (2010). Thus, answering research question 1, we have shown that MCTS is applicable to a one-player deterministic perfect-information game. Our variant, SP-MCTS, is able to achieve good results in the game of SameGame. SP-MCTS is a worthy alternative for puzzles where a good admissible estimator cannot be found.

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. A selective-search method specially designed for solving (end)games is Proof-Number (PN) search. PN search is efficient in searching game trees with a non-uniform branching factor. Moreover, for quite some games, endgame databases played a role in solving. This has led us to the second research question.

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

Chapter 4 answers the second research question by examining the tradeoff between time spent on PN search and time spent on creating endgame databases when solving the game of Fanorona. This game has a material-based theme with a state-space complexity similar to checkers. Endgame-database statistics show that (1) the player to move has an advantage and (2) that a draw can often be achieved in

spite of having fewer pieces than the opponent. The optimal endgame-database sizes for the 3×9, 5×5, and 7×5 Fanorona variants are 3, 4, and 5 pieces, respectively. We conclude that the optimal database size is located at the point where the time required for database construction and the time required for solving by PN search are of the same order. Our main result is that standard Fanorona (5×9) has been weakly solved and is drawn when both players play optimally, adding Fanorona to the list of solved games. This result was achieved by combining the PN-search variant PN^2 with all endgame databases up to 7 pieces.

Another form of searching selectively in two-player deterministic games with perfect information is variable-depth search. 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. For two-player games with non-determinism or imperfect information expectimax may be used. Expectimax adds chance nodes to the search tree. There are, however, no forward-pruning techniques available for chance nodes. This has led us to the third research question.

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

Chapter 5 answers the third research question by proposing the forward-pruning technique ChanceProbCut for expectimax. This technique is the first in its kind to forward prune at chance nodes. ChanceProbCut is inspired by the $\alpha\beta$ forward-pruning technique ProbCut. ChanceProbCut estimates values of chance events based on shallow searches. Based on the correlation between evaluations obtained from searches at different depths, ChanceProbCut prunes chance events in advance if the result of the chance node probably falls outside the search window. Two non-deterministic games (Dice and ChanceBreakthrough) and a game of imperfect information (Stratego) served as test domains. Experiments revealed that ChanceProbCut is able to reduce the size of the game tree significantly without a loss of decision quality in Stratego, Dice, and ChanceBreakthrough. A safe node reduction of between 30% and 85% is achieved across all games. Thus, ChanceProbCut finds the best move faster in the expectimax framework, while not affecting the playing strength. The gained time may be invested in a deeper search. Selfplay experiments in Stratego and Dice showed that there is a small but relevant improvement in playing strength. In ChanceBreakthrough, though, a significant increase in performance was measured. ChanceProbCut is able to win 54.4% on 4,000 games.

In deterministic two-player games with perfect information, the majority of research focused on the $\alpha\beta$ algorithm. For deterministic multi-player games with perfect information, the choice of algorithm is not as straightforward. The two main algorithms are called $\max^n$ and paranoid, 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. Furthermore, the underlying assumption

of $\max^n$ may be unrealistic, resulting in $\max^n$ to be too optimistic. When searching deep with the paranoid algorithm, the other players may dominate the root player, resulting in paranoid to be too pessimistic. This has led us to the fourth research question.

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

Chapter 6 answers the fourth research question by proposing a new search algorithm, called Best-Reply Search (BRS), for deterministic non-cooperative multi-player games with perfect information. This algorithm allows only one opponent to play a counter move. This opponent is the one with the strongest move against the root player. The other players have to pass their turn. Using this approach, more turns of the root player can be searched, resulting in long-term planning. At the same time, some sort of cautiousness is preserved by searching the strongest opponent move.

We have chosen three deterministic multi-player games of perfect information, i.e., Chinese Checkers, Focus, and Rolit. BRS is able to significantly outperform $\max^n$ in these games, with a win ratio of between 65% and 95%. Against paranoid, BRS is significantly stronger in Chinese Checkers and Focus, with win ratios of between 57% and 71%. In Rolit, BRS and paranoid are on equal footing. When playing different kind of opponents at the same time, BRS is the strongest algorithm in Chinese Checkers and Focus. In Rolit, BRS is somewhat behind paranoid. Increasing the search time generally does not have a negative effect on the performance of BRS. This implies that searching illegal positions, which are generated by forcing opponents to pass, does not have a large influence. The possible negative effect is outbalanced by the larger lookahead.

Chapter 7 concludes the thesis and gives an outlook on open questions and directions for future research. Taking the answers to the research questions above into account we see that there are four ways to improve selective-search methods. First, Single-Player Monte-Carlo Tree Search balances exploitation and exploration such that it is a worthy alternative for one-player games where a good admissible estimator cannot be found. Second, PN search with endgame databases, which prefers narrow subtrees above wide ones, is able to prove that the game-theoretic value of the two-player game Fanorona is a draw. Third, ChanceProbCut can forward prune chance events in two-player games with non-determinism or imperfect information. Fourth, in non-cooperative deterministic multi-player games with perfect information, Best-Reply Search achieves long-term planning by assuming that only one opponent is allowed to play a counter move.

While this thesis shows that selective-search methods are successful in quite some domains, all our proposed enhancements and algorithms can be tested in other domains, as well. These domains include classes of games with corresponding complexity levels, but also game classes which have not been covered in this research. We mention one-player games with non-determinism, two-player games with non-determinism and imperfect information, and multi-player games with non-determinism and/or imperfect information.