Chapter 7

Conclusions and Future Research

This thesis investigated how selective-search methods can improve the performance of a game program for a given domain. This led to the formulation of our problem statement in Section 1.3.

> **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 game, 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 (e.g., a player, chance or imperfect information is added). The domains consisted of deterministic one-, two- and multi-player games with perfect information, and two-player non-deterministic or imperfect-information games. We have posed four research questions that should be answered before we could address the problem statement.

In this chapter, we present the conclusions of the thesis. In Section 7.1 we answer the four research questions one by one. We formulate an answer to the problem statement in Section 7.2. Finally, in Section 7.3 we provide promising directions of future research.

7.1 Conclusions on the Research Questions

The four research questions stated in Chapter 1 concern different classes of games, each with a different level of complexity, i.e., (1) one-player games, (2) two-player games, (3) two-player games with non-deterministic and imperfect-information, and (4) multi-player games. They are dealt with in the following subsections, respectively.

7.1.1 One-Player Games

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 type 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. 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?*

To answer the first research question, we proposed a new MCTS variant called Single-Player Monte-Carlo Tree Search (SP-MCTS). We adapted MCTS by two modifications resulting in SP-MCTS. The modifications concern (1) the selection strategy and (2) the backpropagation strategy. For testing SP-MCTS, we have chosen the puzzle SameGame 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 Search (Cazenave, 2009) and Heuristically Guided Swarm Tree Search (Edelkamp *et al.*, 2010), 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.

7.1.2 Two-Player Games

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. A search method specially designed as mate-solver is Proof-Number (PN) search. PN search is efficient in searching game trees with a non-uniform branching factor. However, PN search has to expand nodes until the end of the game is reached. Moreover, for quite some games, endgame databases played a substantial role in solving. When using endgame databases, branches entering the database can be pruned. 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?*

We examine 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 size 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 the game of Fanorona (5×9) has been weakly solved and is drawn when both players play optimally, adding Fanorona to the list of solved games (cf. Van den Herik *et al.*, 2002). This result was achieved by a well-chosen combination of the PN-search variant PN^2 and all endgame databases up to 7 pieces.

Finally, we remark that we were the first to compute and analyze these endgame databases for Fanorona. From these experiments we offer two additional conclusions. The first conclusion we draw is that the optimal endgame-database size for Fanorona (5×9) is 6 or 7 pieces. The time required for solving Fanorona with 7 pieces was less than the time required for creating the 7-piece endgame databases. The second conclusion we draw is that White is able to force a win on board sizes with one side equal to 3. We conjecture that for boards where both sides have at least size 5, White does not have this advantage for the majority of cases.

7.1.3 Two-Player Games with Non-Determinism and Imperfect Information

In 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?*

For answering the third research question, we have proposed 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 (Buro, 1995). 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 a good 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 was able to win 54.4% on 4,000 games.

Thus, answering research question 3, the proposed forward-pruning technique ChanceProbCut improves the search performance in non-deterministic games and games with imperfect information.

7.1.4 Multi-Player 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. This has led us to the fourth research question.

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

For answering the fourth research question, we proposed 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 was 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.

Thus, answering research question 4, the proposed search algorithm BRS is able to significantly outperform both established algorithms, the $\max^n$ and paranoid algorithm.

7.2 Conclusion on the Problem Statement

After answering all four research questions, we are now able to provide an answer to the problem statement.

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

Taking the answers to the research questions above into account we see that there are several ways to improve selective-search methods. We can summarize these in four points. 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, was able to prove that the game-theoretic value of the two-player game Fanorona is a draw. Third, ChanceProbCut is able to forward prune at chance nodes 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.

7.3 Recommendations for Future Research

The research presented in this thesis indicates the following areas of future research.

1. **Improving SP-MCTS.** We mention three possible enhancements in SP-MCTS. (1) Knowledge can be included in the selection mechanism with RAVE (Gelly and Silver, 2007) or progressive widening (Coulom, 2007a; Chaslot *et al.*, 2008d). (2) We demonstrated that combining small searches can achieve better scores than one large search. However, there is no information shared between the searches. This can be achieved by using a transposition table, which is not cleared at the end of a small search. (3) Regular root parallelization should be investigated to take advantage of multi-processor architectures.

2. **Improving PN-Search.** The time for solving may be reduced significantly by using Evaluation-Function Based Proof-Number Search (EF-PN) (Winands and Schadd, 2011). EF-PN is a general framework for employing a traditional evaluation function in PN search. The search is directed to branches where the evaluation function indicates a promising situation. For Fanorona, the material on the board may be used as an evaluation function.

3. **Improving ChanceProbCut.** We propose three directions for improving ChanceProbCut. (1) ChanceProbCut uses a number of linear regression models to predict the value at depth d using a value of depth $d-R$. For improving the effectiveness of ChanceProbCut, additional linear regression models for different depths may be used. (2) The regression parameters and cut-threshold t

can be bootstrapped according to the game phase. (3) A successor of ProbCut exists, called Multi-ProbCut (Buro, 2000). This technique could also be adapted for chance nodes (i.e., Multi-ChanceProbCut).

4. **Testing and improving BRS.** Sturtevant (2008a) showed that MCTS outperforms $\max^n$ and paranoid in Chinese Checkers, when given enough time. For further testing BRS, an interesting experiment would be to compare the playing strength of BRS against an MCTS program in Chinese Checkers. Furthermore, the BRS principle can be used in MCTS programs as well. This might lead to an improvement in playing strength.

   For improving BRS, a possible direction is variable-depth search (Marsland and Björnsson, 2001). The most prominent forward-pruning techniques are null moves (Beal, 1989; Goetsch and Campell, 1990), ProbCut (Buro, 1995) and Multi-Cut (Björnsson and Marsland, 2001). Using these techniques, an even larger lookahead would be possible. Because of the direct succession of MAX and MIN nodes in BRS, we expect that forward-pruning techniques are more effective in BRS than in paranoid. This could give BRS an edge over paranoid.

   We provide three further approaches to improve BRS. (1) Searching illegal positions is not necessary for BRS. Instead, the opponents who are not selected for the counter move could be allowed to play the first move from the static move ordering. This may make BRS applicable to games like Hearts. (2) Because lookahead is important, it would be interesting to test how an algorithm performs that only searches moves by the root player (making it a one-player game, as in the evaluation function for Chinese Checkers by Sturtevant, 2003a). (3) The MP-Mixed algorithm chooses a search method based on the current situation of the game (Zuckerman *et al.*, 2009). BRS may be able to improve the strength of this technique as well.

5. **Application to other domains.** All our proposed enhancements and algorithms can be tested in other domains. 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.