Analysing rational agent performance in environments altering from imperfect to perfect information states

Anonymous submission

Vrije Universiteit Amsterdam

Abstract. In this paper, three different rational agents using PIMS (Perfect Information Monte Carlo Sampling), Mini-Max with Alpha-Beta pruning, and machine learning are introduced. The performance of the agents is examined in a digital version of the Austrian card game 'Schnapsen', also known as the game of '66'.

The research is focused on analysing the performance of the agents in environments altering from imperfect to perfect information states. We wanted to examine how combining different algorithms in the first and second phase of Schnapsen will affect the performance of the agent. The experiments consisted of tournaments between bots that combine Monte Carlo Sampling (first phase of the game) and Mini-Max with Alpha-Beta pruning algorithm (second phase of the game), and bots combining Monte Carlo Sampling (first phase of the game) with machine learning (first and second phase of the game).

We evaluated the win ratio of different combinations of algorithms. Based on the experiment data, no significant improvement in the performance of the agents was found. Future research is required in order to obtain better a understanding of altering rational agent performance by combining algorithms.

Keywords: Perfect Information Monte Carlo Sampling · Mini-Max Algorithm · Alpha-Beta Pruning · Intelligent Systems · Artificial Intelligence · Adversarial Search · Machine Learning · Neural Networks

1 Introduction

Rational agents Rational agents and intelligent systems are able to act upon available information retrieved from the environment. In some situations, agents deal with states of imperfect information, requiring the agent to make an assumption.

Intelligent system objective The purpose of a rational agent is to decide to take an action that maximises the expected value of the performance measure. To do so, the agent relies on the information retrieved from the environment. In

a state of imperfect information, the agent relies on an assumption about the environment. In addition to this information, the agent can make use of acquired prior knowledge regarding the environment.

Schnapsen The Austrian card game 'Schnapsen' is highly suitable for testing intelligent system performance in situations which alter from imperfect to perfect information states due to the game's characteristics[2].

Schnapsen characteristics:

1. Adversarial agent setting:
   - The opponent interferes with the agent's objectives.
2. Imperfect information state in the first phase of the game:
   - The first phase of Schnapsen is characterised by its enormous uncertainty (billions of possible situations)[1].
3. Perfect information state in the second phase of the game:
   - Once the stock has been exhausted it is known which cards have been played in the game, which cards are in possession of the agent and therefore which cards are in the opponent's hand. The environment is completely observable from that point.
4. Schnapsen is suitable to apply many different techniques:
   - A wide variation of algorithms can be used to develop playing strategies [3].
5. Repeatable experiments:
   - The digital version of Schnapsen is highly suitable for repeating (a high number of) experiments. In addition, a seed can be used to generate a specific state which allows for experiments to be run with the same starting state of the game.
6. Playing speed:
   - The playing speed of the agent is only important if chosen to be, for instance when setting a maximum to the time which can be used for making a move.
7. Relative simplicity:
   - Schnapsen is a relative simple game, however, the applicable techniques have a very widespread application.

Schnapsen agents Three Schnapsen playing bots have been built for this research.

Table 1. Schnapsen bots used for research.

Bot	First phase	Second phase
1	Monte Carlo Sampling	Mini-Max with Alpha-Beta pruning
2	Monte Carlo Sampling	Machine learning
3	Machine Learning	Machine learning

The bots have been evaluated based on their performance. The goal of the bots is to perform 'rationally'. To acquire rational agent behaviour, the following methods were used:

- Systematic exploration of the state-spaces with relation to the possible moves
- Knowledge representation
- Automated learning of game playing strategies

The decision of the agent is based on the following variables:

- The expected value of the performance measure (the heuristics)
- The possible moves (search space)
- Acquired knowledge (machine learning)
- Prior environment knowledge (knowledge representation)

2 Background information

Schnapsen combinatorics Schnapsen is a point-trick turn-based card game. For an impression of the state space of the game in the first phase (in which the agent is dealing with imperfect information) the following table has been included.

Table 2. Possible deals in the first phase of Schnapsen.

Beginning of trick	Calculation	Possible deals
5	6×1	6
4	$8 \times 7 \times 6$	336
3	$10 \times 9 \times 8 \times 7 \times 6$	30.240
2	$12 \times 11 \times 10 \times 9 \times 8 \times 7 \times 6$	3.991.680
The whole game	$\frac{14!}{5!}$	726.485.760

Perfect information Monte Carlo Sampling (PIMS) We define $P(s)$ as the probability of a deal. Therefore, the best action is the move with the highest probability of winning the game. The following function uses the Mini-Max algorithm for all possible deals and chooses the one with the highest probability of winning. This computation is infeasible as it requires summing over 726.485.760 possible states (see Table. 2).

$$\text{argmax}_a \sum_s P(s) Minimax(Result(s, a))$$

Using the Mini-Max algorithm, the maximum amount of nodes in each game is:

$$5! \times 5! \times 5^{10} = 140.625.000.00$$

Therefore the total size of the search for the winning strategy using the Mini-Max algorithm would be:

$$140.625.000.00 \times 726.485.760 = 102.162 \times 10^{18}$$

Using Monte Carlo Sampling[4] we can repeat a random sampling N times, thereby playing N random games. This results in the following function:

$$\text{argmax}_a \frac{1}{N} \sum_{i=1}^{N} Minimax(Result(s_i, a))$$

The computation has improved but is still expensive, the amount of nodes to be explored is:

$$N \times 5! \times 5! \times 5^{10}$$

The Monte Carlo Sampling algorithm which is used in bot 1 and 2 (see Table. 1) for the first phase of the game, samples over random rollouts and returns the maximum value using the following function:

$$\text{argmax}_a \frac{1}{N} \sum_{i=1}^{N} randomrollouts(s_i, a)$$

3 Research question

How does the addition of machine learning or the Mini-Max algorithm with Alpha-Beta pruning to PIMS affect rational agent performance in environments altering from imperfect to perfect information states?

Approach The initial phase of the research consisted of mapping the digital Schnapsen environment in which the research question would be examined.

Table 3. Digital Schnapsen environment used for research.

Characteristics	First phase	Second phase
Observable	Not completely	Completely
Deterministic	Yes	Yes
Static	Yes	Yes
Discrete	Yes	Yes
Agent	Multi	Multi

The difference in the observability of the environment between the first and the second phase raised questions regarding optimal rational agent behaviour. Based on this discrepancy the possibilities regarding the use of different algorithms in the first and second phase of Schnapsen were analysed[6].

3.1 Suitable algorithms

PIMS in the first phase of Schnapsen As described in the background information (see section 2), the PIMS algorithm samples over random rollouts[9]. This algorithm is highly suitable and performs very well in situations of imperfect information[7]. However, when the game alters from an imperfect information state to a perfect information state, the PIMS algorithm continues to rely on random rollouts with relation to the environment. Therefore applying a different, more suitable, algorithm for the second phase of the game could potentially improve the agent's performance[5].

Mini-Max algorithm with Alpha-Beta pruning in the second phase of Schnapsen The second phase of the game is characterised by perfect information. The Mini-Max algorithm is characterised by its' goal to maximise the minimum value it can achieve in a search tree[12]. The Mini-Max algorithm requires perfect information to be able to analyse the search tree. Depending on the depth to which the Mini-Max algorithm is allowed to search the tree and the actual presence of a winning strategy in the search tree, the Mini-Max algorithm will always return a winning strategy (with emphasis on the fact that this only happens when there is a possible winning strategy in the search tree). In addition to the Mini-Max algorithm, Alpha-Beta pruning can be implemented in order to prevent analysing sections of the search tree which would not affect the outcome of the game[10]. The implementation of Alpha-Beta pruning in the Mini-Max algorithm is characterised by:

Alpha-Beta pruning characteristics[11]

- No effect on final results of the search
- Entire subtrees can be pruned
- Good move ordering improves the effectiveness of pruning
- In situations of 'perfect ordering' the time complexity is $O(b^{\frac{m}{2}})$
- Mini-Max with Alpha-Beta pruning can look twice as far as Mini-Max without Alpha-Beta pruning in the same amount of time

Based on these algorithm features, the Mini-Max algorithm with Alpha-Beta pruning is highly suitable for implementation in the second phase of Schnapsen.

Machine learning in Schnapsen Potential improvements in the agent's performance could be obtained through the implementation of machine learning[8]. A neural network was trained on games played by the Rdeep bot (Rdeep uses PIMS in both the first and the second phase of the game). The following feature set was used for the training of the neural network:

Table 4. Machine learning feature set.

Player 1	Player 2	Game	Opponent
Points	Pending points	Trump suit	Cards played
Points	Pending points	Phase	
		Stock size	
		Leader of the game	
		Whose turn	

Table 5. Possible outcome research setup.

Bot	Phase 1	Phase 2	Potential outcome
1	PIMS	Mini-Max with Alpha-beta pruning	No significant improvement
1	PIMS	Mini-Max with Alpha-beta pruning	Significant improvement
2	PIMS	Machine learning	No significant improvement
2	PIMS	Machine learning	Significant improvement
3	Machine learning	Machine learning	No significant improvement
3	Machine learning	Machine learning	Significant improvement

4 Experimental setup

To analyse the performance of the bots, the Rdeep bot (which uses PIMS in both the first and the second phase of the game) was set as the baseline for comparison. This choice was made because the focus of this research is on optimising rational agent performance in situations altering from imperfect to perfect information states. Since Rdeep performs extremely well in imperfect information phases, setting this as the baseline will enable insight into performance improvement regarding an alteration in the handling of the information available to the agent [14].

4.1 Implementation

Bot 1 versus Rdeep As described in the introduction (see section 1) three different bots have been built. First bot 1, which uses PIMS in the first phase of the game and Mini-Max with Alpha-Beta pruning in the second phase of the game (see table 1) played 100, 75 and 50 random generated seed games against the Rdeep bot.

Training the machine learning bot for the second phase The machine learning bot was trained on games only consisting of the second phase played by the Rdeep bot. A new bot was created using PIMS in the first phase of the game and the acquired machine learning model for the second phase of the game. This bot played 100, 75 and 50 random generated seed games against the Rdeep bot.

Training the machine learning bot for the first and the second phase A new machine learning bot was trained on games consisting of both the first and the second phase played by the Rdeep bot. This bot played according to the machine learning model which was trained on games from both game phases. The bot again played 100, 75 and 50 random generated seed games against the Rdeep bot.

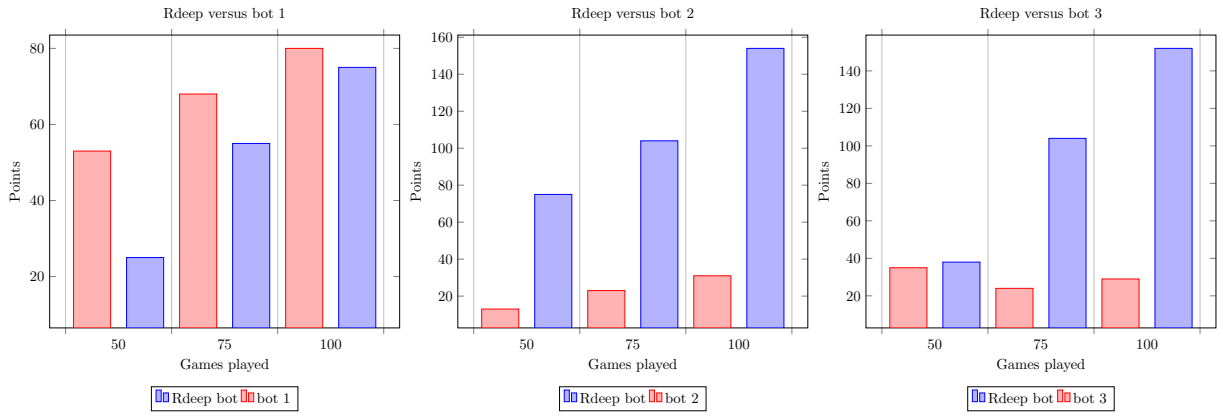
Comparing different methods Each bot was tested in three tournaments of 100, 75 and 50 random generated seed games respectively, with Rdeep as the adversarial agent. Tournaments of 100, 75 and 50 games were chosen to collect more data on the performance of the bots.

Dependent variable The amount of points won by each of the bots in the tournaments was measured as the dependent variable.

Hypotheses Three independent hypotheses have been set (one for each bot respectively):

1. Bot 1 performs better than the Rdeep bot (the probability of winning a game with Rdeep as adversarial agent for bot 1 is > 0.5)
2. Bot 2 performs better than the Rdeep bot (the probability of winning a game with Rdeep as adversarial agent for bot 2 is > 0.5)
3. Bot 3 performs better than the Rdeep bot (the probability of winning a game with Rdeep as adversarial agent for bot 3 is > 0.5)

5 Results



Result overview To analyse the performance of bot 1, 2, 3 and the Rdeep bot, all points won in the three tournaments (100, 75 and 50 games respectively) were added and divided by the number of games.

Table 6. Experiment results bot 1, 2 and 3.

Bot	50 games tournament	75 games tournament	100 games tournament	total points	Games won*
1	53 points	68 points	80 points	201	$\frac{201}{7} = 28.7$
2	13 points	23 points	31 points	67	$\frac{67}{7} = 9.6$
3	35 points	24 points	29 points	88	$\frac{88}{7} = 12.6$

*In Schnapsen the goal is to reach 7 points, therefore the total number of points was divided by 7.

Table 7. Experiment results Rdeep bot.

Opponent bot	50 games tournament	75 games tournament	100 games tournament	total points	Games won
1	25 points	55 points	75 points	155	$\frac{155}{7} = 22.1$
2	75 points	104 points	154 points	333	$\frac{333}{7} = 47.6$
3	38 points	104 points	152 points	294	$\frac{294}{7} = 42$

Table 8. Relative and binomial experiment results.

Bot	Games won	Games lost	Total games	Win-ratio	Lose-ratio	Probability value
1	28.7	22.1	50.8	0.56	0.44	0.216177
2	9.6	47.6	57.2	0.17	0.83	1
3	12.6	42	54.6	0.23	0.77	0.999988

6 Findings and analysis

Interpretation and explanation of results bot 1 The first bot achieved an average win-ratio of 0.56 in the three tournaments played by the bot with the Rdeep bot as the adversarial agent. The binomial test performed on the experiment data, resulted in a probability value of 0.216177. Therefore, the null hypothesis was not rejected, meaning that bot 1 does not perform better than the Rdeep bot and that the probability of winning a game with Rdeep as the adversarial agent is not higher than 0.5. In the experiment data, the Mini-Max with Alpha-Beta pruning algorithm does not significantly outperform the heuristics function of the Rdeep bot. Generalising the experiment results is highly unreliable due to the specific heuristics used by the Rdeep bot and the specific (simulated) environment in which the research has been conducted. Potential future research could be focused on improving the heuristics of the Rdeep bot and performing similar (improved) research experiments.

Table 9. Experiment results bot 1.

Tournament	Points won	Points lost	Total points	Won points ratio	Lost points ratio
50 games	53 points	25 points	78 points	$\frac{53}{78} = 0.68$	$\frac{25}{78} = 0.32$
75 games	68 points	55 points	123 points	$\frac{68}{123} = 0.55$	$\frac{55}{123} = 0.45$
100 games	80 points	75 points	155 points	$\frac{80}{155} = 0.52$	$\frac{75}{155} = 0.48$
Average				0.58	0.42
Standard deviation				0.085	0.085
Coefficient of variation				0.147	0.202

Interpretation and explanation of results bot 2 The second bot achieved an average win-ratio of 0.17 in the three tournaments played by the bot with the Rdeep bot as the adversarial agent. The binomial test performed on the experiment data, resulted in a probability value of 1. Therefore, the null hypothesis was not rejected, meaning that bot 2 does not perform better than the Rdeep bot and that the probability of winning a game with Rdeep as the adversarial agent is not higher than 0.5. In the experiment data, the implementation of machine learning to the second phase of Schnapsen does not significantly outperform the heuristics function of the Rdeep bot. Generalising the experiment results is highly unreliable due to the specific feature set used for training the machine learning model, the specific heuristics used by the Rdeep bot and the specific (simulated) environment in which the research has been conducted. Potential future research could be focused on finding optimal feature sets with relation to training neural networks in environments altering between imperfect and perfect information states.

Table 10. Experiment results bot 2.

Tournament	Points won	Points lost	Total points	Won points ratio	Lost points ratio
50 games	13 points	75 points	88 points	$\frac{13}{88} = 0.15$	$\frac{75}{88} = 0.85$
75 games	23 points	104 points	127 points	$\frac{23}{127} = 0.18$	$\frac{104}{127} = 0.82$
100 games	31 points	154 points	185 points	$\frac{31}{185} = 0.17$	$\frac{154}{185} = 0.83$
Average				0.16	0.84
Standard deviation				0.015	0.015
Coefficient of variation				0.094	0.018

Interpretation and explanation of results bot 3 The third bot achieved an average win-ratio of 0.23 in the three tournaments played by the bot with the Rdeep bot as the adversarial agent. The binomial test performed on the experiment data, resulted in a probability value of 0.999988. Therefore, the null hypothesis was not rejected, meaning that bot 3 does not perform better than the Rdeep bot and that the probability of winning a game with Rdeep as the adversarial agent is not higher than 0.5. In the experiment data, the implementation of machine learning to both the first and the second phase of Schnapsen does

not significantly outperform the heuristics function of the Rdeep bot. Generalising the experiment results is highly unreliable due to the specific feature set used for training the machine learning model, the specific heuristics used by the Rdeep bot and the specific (simulated) environment in which the research has been conducted. Potential hypotheses for future research could be focused on finding optimal feature sets with relation to training neural networks in environments altering between imperfect and perfect information states. In addition, future research could be focused on equilibrium analysis with relation to the use of machine learning and Monte Carlo Sampling in environments altering from imperfect to perfect information states.

Table 11. Experiment results bot 3.

Tournament	Points won	Points lost	Total points	Won points ratio	Lost points ratio
50 games	35 points	38 points	73 points	$\frac{35}{73} = 0.48$	$\frac{38}{73} = 0.52$
75 games	24 points	104 points	128 points	$\frac{24}{128} = 0.19$	$\frac{104}{128} = 0.81$
100 games	29 points	152 points	181 points	$\frac{29}{181} = 0.16$	$\frac{152}{181} = 0.84$
Average				0.28	0.72
Standard deviation				0.177	0.177
Coefficient of variation				0.632	0.246

7 Future research

The experimentation and findings in this research are rather concise. Further research will have to be conducted. Based on the findings in this research, the research group encourages future research into the following topics:

- Combining algorithms in environments which alter multiple times from imperfect to perfect information or from perfect to imperfect information
- Machine learning with relation to environments altering from imperfect to perfect information
- Monte Carlo Sampling in environments altering from imperfect to perfect information states
- Optimising heuristics in Monte Carlo Sampling algorithms
- Optimising feature sets for neural networks used in environments altering from imperfect to perfect information states and vice versa
- Machine learning in combination with Monte Carlo Sampling algorithms

8 Conclusions

The scope of the research in this paper was focused on analysing the performance of rational agents in environments altering from imperfect to perfect information states. For the research, three different bots were built. The performance of the bots was measured in a digital version of the trick-based card game Schnapsen.

The digital Schnapsen environment is highly suitable for the conducted experiments due to the game's characteristics, specifically the difference in the information which is available to the agent in phase 1 and phase 2 of the game (see section 1).

The performance of the Rdeep bot was set as the baseline for comparison in the experiments. The Rdeep bot performs exceptionally well in environments of imperfect information (first phase of Schnapsen). As an addition to the PIMS algorithm used by the Rdeep bot, the Mini-Max with Alpha-Beta pruning algorithm was added in bot 1. In bot 2 a combination of PIMS for the first phase of the game and machine learning for the second phase of the game was implemented. Bot 3 used machine learning in both phases of the game.

Three independent hypotheses were stated (one for each bot respectively). The hypotheses were directed on each bot's win probability with the Rdeep bot as the adversarial agent. Each bot played three tournaments of 50, 75 and 100 games respectively with Rdeep as the adversarial agent. Based on the results acquired in the experiment the binomial test was performed for each of the bots and the corresponding hypothesis. In neither of the bots a significant improvement was found, therefore, none of the null hypotheses were rejected.

References

1. Parker, Austin, Dana Nau, and V. S. Subrahmanian: "Game-tree search with combinatorially large belief states." IJCAI. 2005.
2. Scofield, J.A. and Garner: "Schnapsen and Sixty-Six Rules Variants", http://psellos.com/schnapsen/rules-background.html.
3. Scofield, J.A. and Garner: "Winning Strategy for Schnapsen or Sixty-Six": http://psellos.com/schnapsen/strategy.html
4. Long, Jeffrey and Sturtevant, Nathan and Buro, Michael and Furtak, Timothy: "Understanding the success of perfect information monte carlo sampling in game tree search", AAAI Conference on Artificial Intelligence, vol 24, 2010
5. Noam Brown, Anton Bakhtin, Adam Lerer Qucheng Gong: "Combining Deep Reinforcement Learning and Search for Imperfect-Information Games", 2020
6. Sjoerd Druiven: "Knowledge Development in Games of Imperfect Information". 2002.
7. Florian Wisser: "An Expert-Level Card Playing Agent Based on a Variant of Perfect Information Monte Carlo Sampling", Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)
8. Christopher Solinas and Douglas Rebstock and Michael Buro": "Improving Search with Supervised Learning in Trick-Based Card Games". 2019.
9. Matthew L. Ginsberg: "GIB: Imperfect Information in a Computationally Challenging Game". Journal of Artificial Intelligence Research 14 (2001) 303–358. 2001.
10. Fuller, Samuel H and Gaschnig, John G and Gillogly, JJ and others: "Analysis of the alpha-beta pruning algorithm", 1973, Department of Computer Science, Carnegie-Mellon University
11. Knuth, Donald E and Moore, Ronald W: "An analysis of alpha-beta pruning", Artificial intelligence, pages293-326, 1975

12. Rivest, Ronald L : "Game tree searching by min/max approximation", Artificial Intelligence, vol 34, nr.1, pp. 77–96, 1987
13. Neal, Radford M: "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification", 1997
14. Chaslot, Guillaume and Bakkes, Sander and Szita, Istvan and Spronck, Pieter: "Monte-Carlo Tree Search: A New Framework for Game AI.", AIIDE, 2008