Risk-Taking vs. Laid-Back Strategies in the card game Schnapsen

Jasper Meijerink and Skick Bakker

Vrije Universiteit, De Boelelaan 1105, 1081 HV Amsterdam

Abstract. This study investigates the effectiveness of risk-taking versus laid-back strategies in the two-player card game Schnapsen by developing deterministic bots representing each approach. The risk-taking bot prioritizes aggressive tactics, such as early trump exchanges and high-value card plays, while the laid-back bot keeps their resources for late-game dominance. Through 100,000 simulated matches under controlled conditions, we analyze win rates and performance against baseline opponents (RandBot and RdeepBot). Results reveal that while the risk-taking bot achieves a marginally higher win rate (50.4%) against the laid-back bot (49.6%), its victories are decisively dominant, securing maximum points (3 per win) compared to the laid-back bot's narrow wins (1 per win). This difference highlights that aggressive play gives higher-impact victories, even with near-equal win rates. However, against the advanced RdeepBot who uses the Monte Carlo Algorithm, the laid-back bot significantly outperforms its counterpart (42.6% vs. 19.7% win rate), showing better gameplay against foresight-driven strategies. Both strategies dominate the random baseline (RandBot), displaying the superior strategy of structured play. The study emphasizes the importance of adaptability in Schnapsen and suggests hybrid strategies as a promising idea for future research.

Keywords: Schnapsen Bot · Passive vs Aggressive · Risk-taking vs Laid-back

1 Introduction

"Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it." [1] This foundational saying from the 1956 Dartmouth Conference Proposal begins our investigation into strategic decision making in the card game Schnapsen. By making two opposing human-like strategies, risk-taking and laid-back, into deterministic bots, we try to empirically determine which approach achieves better performance. Specifically, we ask: In Schnapsen, does an aggressive strategy prioritizing early dominance outperform a passive strategy focused on endgame control, or do both strategies get comparable results?

Schnapsen, a two-player trick-taking game, involves collecting 66 points through tactical card play, trump suits, and marriages (King-Queen pairs). The structure of the game, divided into an imperfect information phase and a perfect information phase, creates distinct strategic challenges for bot's to tackle.

To evaluate these approaches, we developed two rule-based bots:

- Risk-taking bot: Prioritizes trump exchanges, marriages, and high-value cards to secure quick victories.
- Laid-back bot: Conserves high-value cards and trumps, minimizing early risks to dominate the late game.

This paper is structured as follows: Section 2 talks about Schnapsen's rules and prior work; Section 3 outlines the research question; Sections 4–5 describe the experimental methodology and results; Section 6 discusses implications; and Sections 7–8 propose future research directions and conclusions.

Strategy	**Strengths**	**Weaknesses**
Risk-taking	Early dominance	Vulnerable to counterplay
Laid-back	Late-game control	Susceptible to early aggression

2 Background

2.1 Explantion of Schnapsen

Schnapsen is a two-player card game where the goal is to gather 66 points by winning tricks or declaring marriages. The game uses a lower amount of cards than a normal deck. The deck consists only of Aces, 10s, Kings, Queens, and Jacks, thus leaving 20 cards in the deck. Each player is given five cards, and a card is flipped over to determine the trump suit. This trump suit wins over all other suits regardless of the card's rank. Players take turns playing cards, trying to win tricks by playing higher cards of the same suit or using trump cards. A player can declare a marriage if they have both the King and Queen of the same suit. If a player holds the Jack of the trump suit, they can exchange it for the face-up trump card. If a player reaches 66 points and their opponent has not won a single trick, the winning player scores 3 game points; if the opponent has won at least one trick but has not earned over 33 points, the winner scores 2

game points; and if the opponent has more than 33 points, the winner scores 1 game point. The game consists of two phases, the first phase and the second phase. In the first phase, it is an imperfect information game, meaning that you do not know which cards your opponent has in his hands, and you also do not know what the next card will be that you draw. The second phase can be seen as a perfect information game, this phase begins when the deck is drawn empty, this is because if you remember the cards that have been played, you can figure out what the cards are in your opponents hand.

The amounts of value each card holds are as follows:

Card	**Points**
Ace	11
10	10
King	4
Queen	3
Jack	2

The rules of Schnapsen described here are based on the book "Winning Schnapsen: From Card Play Basics to Expert Strategy" by Tompa and D. M. T. P. [2] and the rules outlined on the Pagat website [3].

2.2 Schnapsen Engine

For this project we used a python schnapsen game engine, which was made for easy customizability and the integration of custom bots [4]. This platform also includes already existing bots like RandBot and RdeepBot, which we used to test the effectiveness of different strategies.

2.3 RandBot

This bot makes completely random moves using Python's built-in `random` module [5]. The bot is started with a deterministic random number generator (`random.Random`), making sure it can be reproduced when a fixed seed is provided. For every valid move (whether leading a trick or responding to one), it chooses a random action from the list of available moves. The simplicity of RandBot makes it a reliable baseline for measuring more advanced strategies.

2.4 RdeepBot

This bot uses a Monte Carlo sampling strategy [6] to simulate game outcomes. Started with a deterministic random number generator (`random.Random`), it evaluates moves by generating `num_samples` random games (up to a specified `depth`) and calculating their average heuristic score. The heuristic prioritizes moves where the bot's points dominate the total points in simulated end states. This computationally heavy strategy provides a good benchmark for evaluating the performance of other bots.

2.5 Randomness

In this project we use randomness to setup the games. Every game is dependent on the random setup, the cards in the talon and the cards in the hands are given randomly. To be able to recreate the same game, we can not use total randomness, therefore we use pseudo-randomness. Pseudo-randomness works by setting a seed, from this seed random numbers can be generated. With the same seed the random numbers will always be the same, even in the same order, making it possible to recreate the same game as many times as needed. [7]

3 Research Question

For our project we wanted to research two different strategies that were the opposite of each other, from this we went with the idea of whether a laid-back or risk-taking strategy is better for the card game schnapsen. Our research question is: In the trick-taking card game schnapsen, which strategy, risk-taking or laid-back, achieves a higher win rate against diverse opponents, and under what game conditions does each strategy excel? We chose these opposite strategies because they show two common approaches that human players often use when playing card games. This makes our research relevant not just for computer bots, but also for understanding human gameplay strategies.

We hope to see if there is a clear winner among the strategies. We predict that one of the strategies would outperform the other, this is because the strategies have a completely different approach, and thus we expect that the advantages of one of the bots will excel and lead it to victory. Thus either the the risk-taking bot will win by closing out games quickly, if it has good cards in his hand, or the laid-back bot will win by holding onto the good cards and closing the game out in the second phase, where it will win most tricks.

To make our research complete, we will test these bots not only against each other but also against existing bots like the RandBot and RdeepBot. This will help us understand how well these strategies work against different types of opponents/algorithms.

4 Experimental Setup

To test whether a laid-back or a risk-taking strategy wins most often, we coded the strategies in bots, so we can make them play each other and maybe even other bots. Thus, we had to code two bots, one with the laid-back strategy and the other with the risk-taking strategy.

4.1 Defining Risk-taking

The definition of the risk-taking bot is that, if he has the first move, he will always play the highest valued move first, this would be in order: trump exchange, marriage, trump card, high scoring cards. If the bot does not have the first

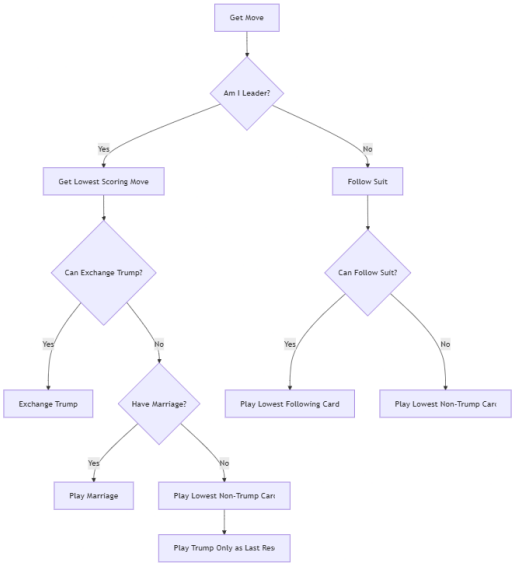

Fig. 1. Flow chart demonstrating the process of the Risk-Taking Bot. The bot uses high-scoring moves and uses aggressive play.

move, he should play a card that wins the trick, so a card of the same suit that has been played, but with a higher value, or a trump card. If he does not have such cards, he should play the lowest card he can, to refrain the opponent from getting to much points.

4.2 Coding Risk-taking Strategy into a Bot

The first thing the bot does is identifying who plays first, he or his opponent. If he plays first, he should check within all his possible moves, which one would score the highest points. It checks whether it can play a trump exchange, then it checks whether it can play a marriage, then it checks whether it can play a trump card, and if it can't play either of these moves, it plays the card which has the highest score, so if it has an A's, a 10 and a Jack in its hand, it will play the A's first. If the bot does not have the first move, but has to follow on the opponent, he will check if he has the cards to win the trick and play those if he can, these are cards of the same suit to the card that has been played, that also has a higher score. If he does not have the cards to win the trick in his hand, he would play the lowest card in his hand. This decision-making process is illustrated in Figure 1.

4.3 Defining Laid-back

The laid-back bot uses a passive strategy where it tries to minimize risk by playing lower valued cards first. This strategy tries to save the high scoring cards

for later in the game. If the bot has the first move, it will prioritize moves in this order: trump exchange, marriage, and then the lowest scoring non-trump card available. The bot tries to avoid playing trump cards unless necessary, saving them for the end.

4.4 Coding Laid-back Strategy into a Bot

The laid-back bot first checks whether it's making the first move or responding to the opponent. When making the first move (leader move), a decision process is followed: first checking for possible trump exchanges or marriages (since these give guaranteed points), then looking for the lowest scoring regular card to play. It specifically tries to avoid playing trump cards early in the game unless it has no other choice. When responding to an opponent's move, the bot tries to follow suit with the minimum necessary card. If it can win the trick, it will use the lowest-scoring card that can still win. If it cannot win the trick, it plays its lowest-scoring card to minimize point loss. Only when it cannot follow suit will it consider playing a trump card, and even then, it will use the lowest-scoring trump card possible. The complete strategy of this conservative strategy is shown in Figure 2.

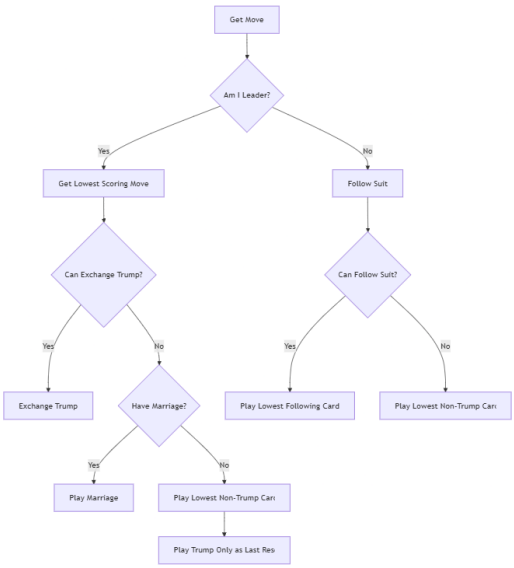

Fig. 2. Flow chart showing the process of the Laid-Back Bot. The bot uses a passive strategy.

4.5 Allow the Bots to Play each other

To test how well the bots perform, they need to be able to play each other. For this we created a separate python file, which allows the user to input the bots they would like to see play each other and the number of times the bots play each other. It even allows the input of more than 2 bots, so they can play a tournament against each other. In the tournament the bots play all other bots the given number of times. The script allows also the usage of a random seed, this number will set the basis for every number that would be generated randomly, thus meaning that the same seed will always result in the same outcomes.

For the tournament we use 2 bots besides our bot, the Randbot and the RdeepBot. The Rdeepbot is set at a depth of 10 and has 10 number of random rollouts. These 2 bots will be playing against both the risk-taking and laid-back bots.

4.6 Reproducibility

All experiments were designed for reproducibility. A fixed random seed (155 etc.) ensures consistent card distributions and bot decisions across runs. The full codebase, including bot implementations and tournament scripts, is publicly available [8].

5 Results

5.1 Summary of Bot Matchup Results

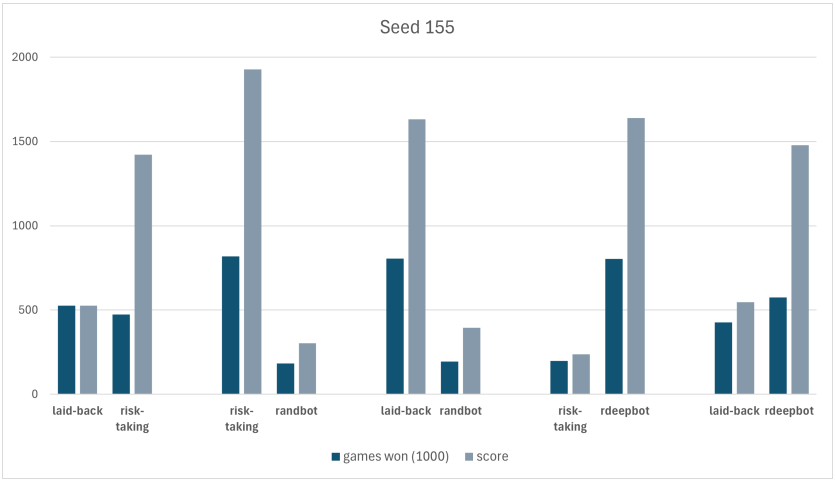

Fig. 3. Graph of the results from the matches that the bots have played.

We have let the both the laid-back and the risk-taking bot play 1000 games against each other and against both the RandBot and RdeepBot. In figure 3 we can see the results from these matches. In the graph both the games won and the scores can be seen. Per game of Schnapsen a total of max 3 points can be given to the winner, given the games outcome. The gray bars display the score of the bots per match, and the dark blue bars display the games the bots won per match. In the graph, the matches played are divided, thus the matches are displayed as the two bots that are close to each other, then there is a space between each of the matches

Table 1. Summary of Bot Matchup Results (1,000 Games, Seed 155).

Note: Results from 1,000 games. Smaller seeds (122, 132, 148) showed similar trends.

Matchup	Bot	Wins	Win Rate (%)	Total Score	Avg. Score
Laid-Back vs Risk-Taking	Laid-Back	526	52.6	526	1.00
	Risk-Taking	474	47.4	1422	3.00
Risk-Taking vs RandBot	Risk-Taking	818	81.8	1928	2.36
	RandBot	182	18.2	302	1.66
Laid-Back vs RandBot	Laid-Back	805	80.5	1632	2.03
	RandBot	195	19.5	395	2.03
Risk-Taking vs RdeepBot	Risk-Taking	197	19.7	237	1.20
	RdeepBot	803	80.3	1640	2.04
Laid-Back vs RdeepBot	Laid-Back	426	42.6	547	1.28
	RdeepBot	574	57.4	1478	2.57

As shown in Table 1, the risk-taking bot narrowly outperformed the laid-back bot (52.6% vs 47.4% win rate). However, the laid-back strategy showed greater effort against the future seeing RdeepBot, achieving a 42.6% win rate compared to the risk-taking bot's 19.7%. Both strategies dominated the RandBot, confirming that thought out play wins over randomness.

5.2 Statistical Analysis

To determine if the observed differences in **win rates** were meaningful, we used a chi-square test [9]. This test compares actual results to what we would expect if no difference existed (the *null hypothesis*). To prove the significance, we performed 100,000 games between the risk-taking and laid-back bots (*seed 999*).

$$H_0 : P_{\text{Risk-Taking}} = P_{\text{Laid-Back}}$$

$$H_1 : P_{\text{Risk-Taking}} \neq P_{\text{Laid-Back}}$$

Risk-Taking vs Laid-Back The null hypothesis states that both strategies perform equally. In 100,000 games, the risk-taking bot won 50,401 times, while the laid-back bot won 49,599 times.

The chi-square statistic is calculated as:

$$\chi^2 = \frac{(50401 - 50000)^2}{50000} + \frac{(49599 - 50000)^2}{50000} = 3.2 + 3.2 = 6.4$$

where 50,000 is the expected number of wins for each bot under the null hypothesis.

With 1 degree of freedom, the resulting p-value is 0.011. Since this value is less than the significance level of 0.05, we reject the null hypothesis. The risk-taking bot's slight edge (50.4% vs 49.6%) is statistically significant.

Summary of Findings

- *Risk-Taking vs Laid-Back*: The 50.4% to 49.6% win rate difference is statistically significant ($p = 0.011$), but practically small.

6 Findings

6.1 Laid-back vs Risk-taking

What we found surprising is that when the risk-taking bot wins, it often secures games with a high point margin. This outcome comes from the aggressive playstyle of the risk-taking bot, which allows it to dominate games fast, leaving the opponent with little opportunity to win tricks at the end of the game. This phenomenon is nicely shown in matches against the laid-back bot, where the risk-taking bot frequently wins without losing a single trick. In those cases, the risk-taking bot is given the full 3 points per game. When you look at it the other way around, when the laid-back bot wins, it often earns only 1 point, indicating that the risk-taking bot has secured enough tricks to go over the 33-point threshold in those games.

6.2 Performance Against Other Bots

The risk-taking and laid-back bots are nearly evenly matched, with the risk-taking bot getting a slightly higher win rate. Both bots perform much better against the RandBot, showing that structured strategies outperform random moves. But both struggle against the RdeepBot, showing that while these strategies are better than randomness, they are less effective than advanced algorithms that use foresight and calculation.

The laid-back bot performs substantially better against RdeepBot than the risk-taking bot, achieving nearly double the win rate and double the points. This shows that when faced with a highly strategic opponent, a laid-back approach may be more effective than an aggressive one. In such scenarios, holding resources back appears to be a more viable strategy than attempting to dominate the game early.

6.3 Key Takeaways

Our experiments revealed two important insights about strategies. The risk-taking bot won 50.4% of 100,000 head-to-head games against the laid-back bot. Although statistically significant, this 0.8% difference is too small to matter in practice, but the amount of points earned by the risk-taking bot still far exceeds the laid-back bot by a large margin. Against RdeepBot, the laid-back bot won 42.2% of the games, more than double the 19.5% of the risk-taking bot. This difference is both statistically and practically meaningful, showing that conservative play counters foresight using opponents better. In summary, risk taking works better against a passive approach, but laid-back strategies excel against opponents who plan ahead.

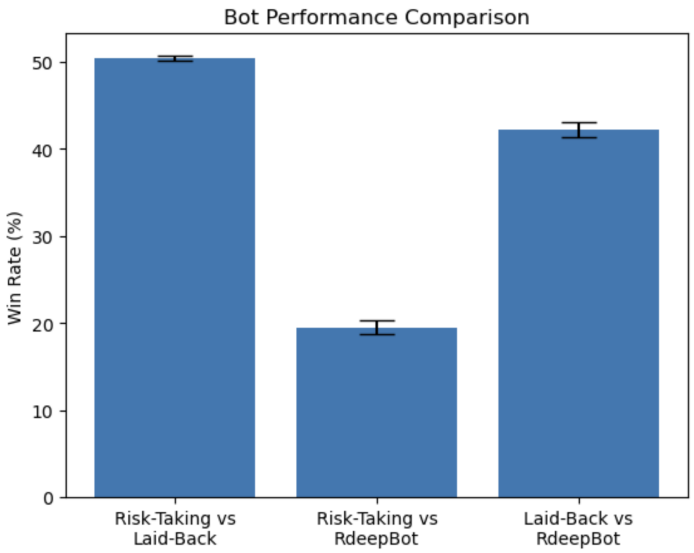

Fig. 4. Bar chart demonstrating the win rate for different matches.

7 Future Work

With the bots we have coded, there is not nearly enough evidence to really support whether a risk-taking or a laid-back strategy is truly better for the card game Schnapsen or trick taking games in general. We have only touched the basics of both strategies. There can be done so much more to improve and refine these strategies.

7.1 Studying Human Strategies

One thing that can be done is studying the playing style of a real player which plays either using a risk-taking or a laid-back strategy. By studying real players, the strategies can be more refined and thus be better implemented in code. For example, watching how human players decide when to hold back or go all-in during certain moments could help to find out if the strategy is applicable in real world scenarios.

7.2 Hybrid Strategies and Advanced AI

Exploring hybrid strategies that dynamically combine elements of both approaches, like switching to risk-taking when ahead in points, could also be an interesting thing to research. Hybrid Intelligence (HI) combines human and artificial intelligence to solve problems more effectively than either could alone. According to Dellermann, HI uses human strengths like creativity and adaptability alongside with AI's ability to process data and recognize patterns [10]. These mixed strategies could be tested in phases, such as playing safe early but taking calculated risks once the deck is smaller. A hybrid bot might start with a laid-back style to conserve strong cards early, then switch to aggressive plays once it gains a points advantage or identifies weaknesses in the opponent's hand in the second game phase. Testing rules like `''go risky if holding two trump cards''` or `''play passive if the opponent has already used high-value cards''` could reveal optimal performance for the second game phase. By experimenting with these combinations, we might discover that flexibility is better than sticking to one strategy, which will lead to better overall performance.

8 Conclusion

In this study, we investigated the effectiveness of risk taking versus laid-back strategies in the card game Schnapsen by developing two rule-based bots that represent these two opposite approaches. Using extensive simulations like 100,000 head-to-head matches and additional games against benchmark bots (RandBot and RdeepBot), we evaluated their performance.

The key findings reveal that while the risk-taking bot achieved a marginal higher win rate (50.4% vs. 49.6%) in direct competition, this higher winrate was statistically significant but practically not noticeable. The risk-taking strategy secured their victories by often earning maximum points (3 per game) by dominating early phases, whilst the laid-back strategy often scored a low amount of points per victory. In contrast, the laid-back bot demonstrated resilience against advanced opponents, outperforming the risk-taking bot significantly when faced with RdeepBot (42. 6% vs 19. 7% win rate). This suggests that passive play, counters foresight-driven (Monte Carlo algorithm) strategies more effectively. Both strategies outperformed the random baseline (RandBot).

These results highlight that optimal strategy depends on context: risk-taking does better in general play by taking use of aggressive early moves, while laid-back tactics thrive against opponents using long-term planning. For human players, this implies flexibility; adapting strategies based on the opponent's style can turn out to give the best results. Future work could explore hybrid approaches or adaptations inspired by humans to further improve our findings.

In summary, while risk-taking proves more effective against passive opponents and laid-back strategies excel against those who plan ahead, neither strategy is universally dominant. Our analysis emphasizes that situational awareness and strategic adaptability are critical for success in Schnapsen and similar trick-taking games.

References

1. McCarthy, J. (1956). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.
2. Tompa, M. and D.M.T.P. (2015) *Winning Schnapsen: From Card Play Basics to Expert Strategy*. CreateSpace.
3. McLeod, J. *Schnapsen - card game rules*. https://www.pagat.com/marriage/schnaps.html.
4. Cochez, M. et al. (2025) *Schnapsen*. GitHub. https://github.com/intelligent-systems-course/schnapsen.
5. Python Software Foundation: *random — Generate pseudo-random numbers*. Python Documentation. Retrieved from https://docs.python.org/3/library/random.html.
6. Shapiro, A. (2003): Monte Carlo Sampling Methods. In: Handbooks in Operations Research and Management Science, vol. 10, pp. 353–425. Elsevier. https://doi.org/10.1016/S0927-0507(03)10006-0.
7. Compagner, A. (1991) Definitions of randomness. American Journal of Physics, 59(8), 700-705.
8. (2025) *GitHub - jasp-nerd/schnapsen: Intelligent Systems Project*. GitHub. https://github.com/jasp-nerd/schnapsen.
9. Turhan, N. S. (2020). Karl Pearson's chi-square tests. *Educational Research and Reviews*, 15(9), 575–580. https://doi.org/10.5897/ERR2019.3817.
10. Dellermann, D., Ebel, P., Söllner, M., & Leimeister, J. M. (2019). Hybrid intelligence. *Business & Information Systems Engineering*, 61(5), 637–643. https://doi.org/10.1007/s12599-019-00595-2.