Universiteit Leiden
The Netherlands

Opleiding Informatica

Theoretical and Practical Aspects of FREECELL

Marten Klaver

Supervisors:
dr. Walter Kosters & dr. Jeannette de Graaf

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

August 2, 2022

Abstract

This thesis covers the SOLITAIRE game FREECELL, looking at what makes games solvable and unsolvable, mainly focusing on the unsolvable games since the win rate for FREECELL is very high. There are multiple patterns found in unsolvable games. These patterns are however not enough to predict from the start if a game is solvable or not. Different aspects of FREECELL are also looked at to see how they affect the win rate, i.e., number of *freecells*, number of suits, *Worrying Back* and more. This thesis also looks at the highest possible scores for unsolvable games.

Contents

1 Introduction **1**
- 1.1 Thesis Goals . . . 1
- 1.2 Thesis Overview . . . 2

2 Rules **3**

3 Related Work **6**

4 Methods **8**

5 Experiments **12**
- 5.1 Different Configurations . . . 12
- 5.2 One Colour . . . 15
- 5.3 Worrying Back . . . 15
- 5.4 Highest Target Reached and Maximal Scores For Lost Games . . . 15
- 5.5 Patterns . . . 21

6 Conclusions and Further Research **23**
- 6.1 Unsolvable Patterns . . . 23
- 6.2 Predicting Unsolvable Games . . . 23
- 6.3 Highest Target Reached Problem . . . 24
- 6.4 Same Stack Length . . . 24
- 6.5 Worrying Back . . . 24
- 6.6 Further Research . . . 25

References **26**

1 Introduction

This thesis is about the Patience/Solitaire card game Freecell. Like in Klondike (commonly known as Patience/Solitaire), the goal of Freecell is to move all the cards from the field to target stacks. The rules for Freecell are explained in Section 2. It is a relatively new game, invented in the 1970s by Paul Alfille, compared to for instance Klondike which was already played in the 19th century. It was based on Baker's Game, where stacks can only be built using cards from the same suit. Baker's Game in turn was based on the game Eight Off, a game that has eight freecells, eight stacks of six cards, with the remaining cards going to four of the freecells. Empty stacks can only be filled with Kings in this game, and stacks can only be built with cards from the same suit. The game was popularised by Jim Horne, who wrote an implementation for DOS and later Windows. The game was included in the default programs of the operating system [Kel22].

In the game of Freecell you spread a standard deck (without jokers) over eight stacks. The goal is to move all the cards from the stacks to target stacks, one for every suit. You first move an Ace to a target stack, and build from there in numerical order. You can move cards onto other cards if they are one higher and a different suit colour. You can move cards to empty freecells. There are 4 freecells in total.

1.1 Thesis Goals

The goal of this thesis is to research different aspects of the game, mainly focusing on unsolvable games, see Figure 2. This includes trying to find patterns in games that cannot be solved, seeing if there are ways to predict if a game is unsolvable from the start, see Figure 1, and trying to find the unsolvable games with the highest number of cards that have been moved to the target stacks. Techniques like Monte Carlo Search and Depth-First-Search are among the techniques that will be used to try to solve games of Freecell.

♠3	♠10	♡4	♡12	♣5	♢1	♣1	♢3
♠8	♠12	♣8	♡10	♡3	♡5	♢5	♣6
♢4	♣10	♡11	♡2	♡1	♢7	♡7	♠4
♢8	♠7	♣9	♣7	♠13	♢6	♢12	♠5
♡8	♠6	♠9	♣2	♠1	♢11	♢9	♣4
♣13	♡6	♣3	♣11	♢13	♡9	♢10	♠11
♠2	♣12	♡13	♢2				

Figure 1: An unsolvable game of Freecell, can we see this from the start?

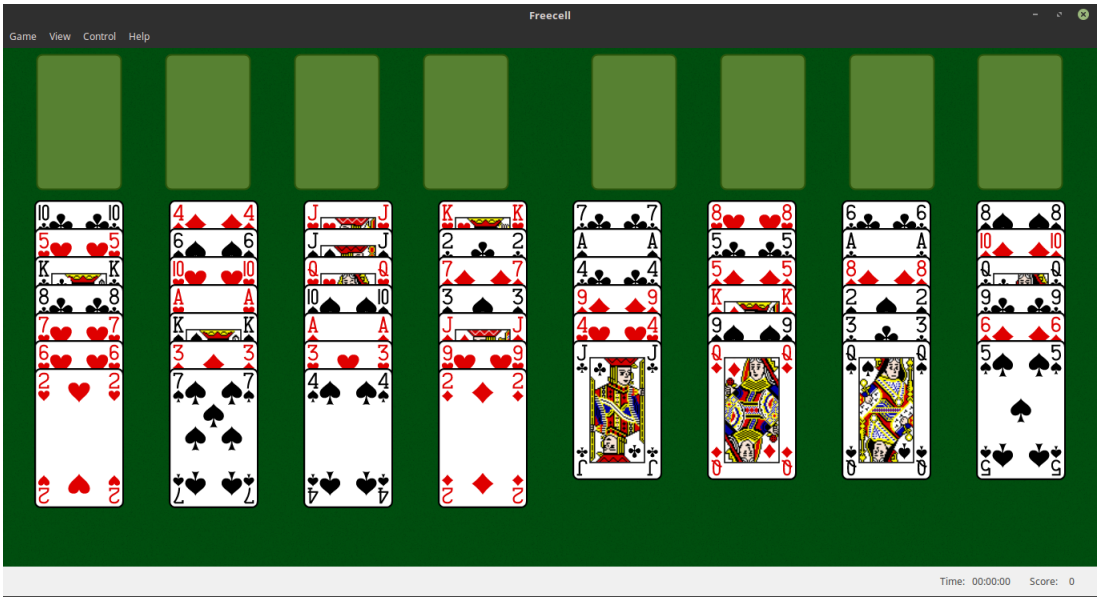

Figure 2: A random game of FREECELL, is this game solvable? Implementation by Aisleriot [GNO20].

1.2 Thesis Overview

This thesis will cover the rules of FREECELL in Section 2, related work is discussed in Section 3. Section 4 covers the implementation and algorithms used for this project. Section 5 describes the experiments and their outcome and Section 6 draws conclusions from those experiments and concludes this thesis. This bachelor thesis for LIACS was supervised by dr. Walter Kosters and dr. Jeannette de Graaf.

2 Rules

The game FREECELL is played with a deck of 52 playing cards, Ace to King, with four suits. The starting layout has eight *stacks* with all the cards facing up, and hence visible. Four of these stacks have seven cards, the other four have six cards. There are four *freecells* and also four *foundations / target stacks*. See Figure 3 for a visual example of the game.

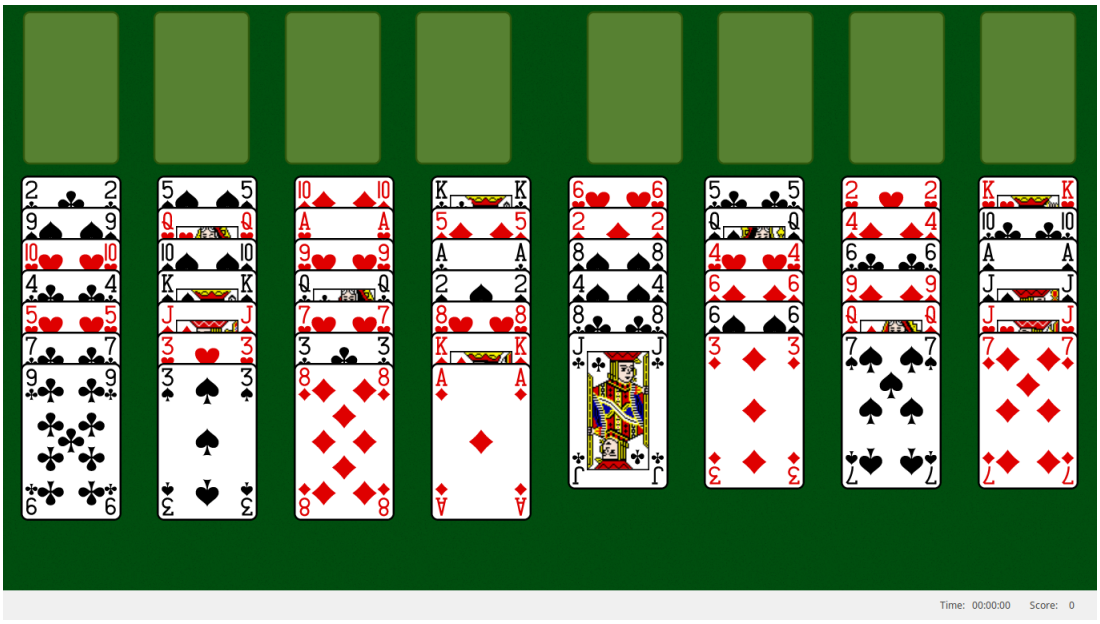

Figure 3: Starting layout of a random FREECELL game, with four freecells in the top left and the four target stacks in the top right. Implementation by Aisleriot [GNO20].

The goal of the game is to bring all the cards to the target stacks, starting with the Aces and ending with the Kings. Each suit has its own target stack that needs to be completed. Any card can be moved to an empty freecell, but at most one card is allowed per freecell.

Cards from the top of a stack, from a freecell, or from a target stack can be moved onto another card on one of the stacks if the card that is being moved and the target card have a different *suit colour* (red/black). E.g., a spades card can be moved onto a diamonds card, and if the target card's numerical value is one higher than the value of the card that is being moved. An example move is shown in Figure 4.

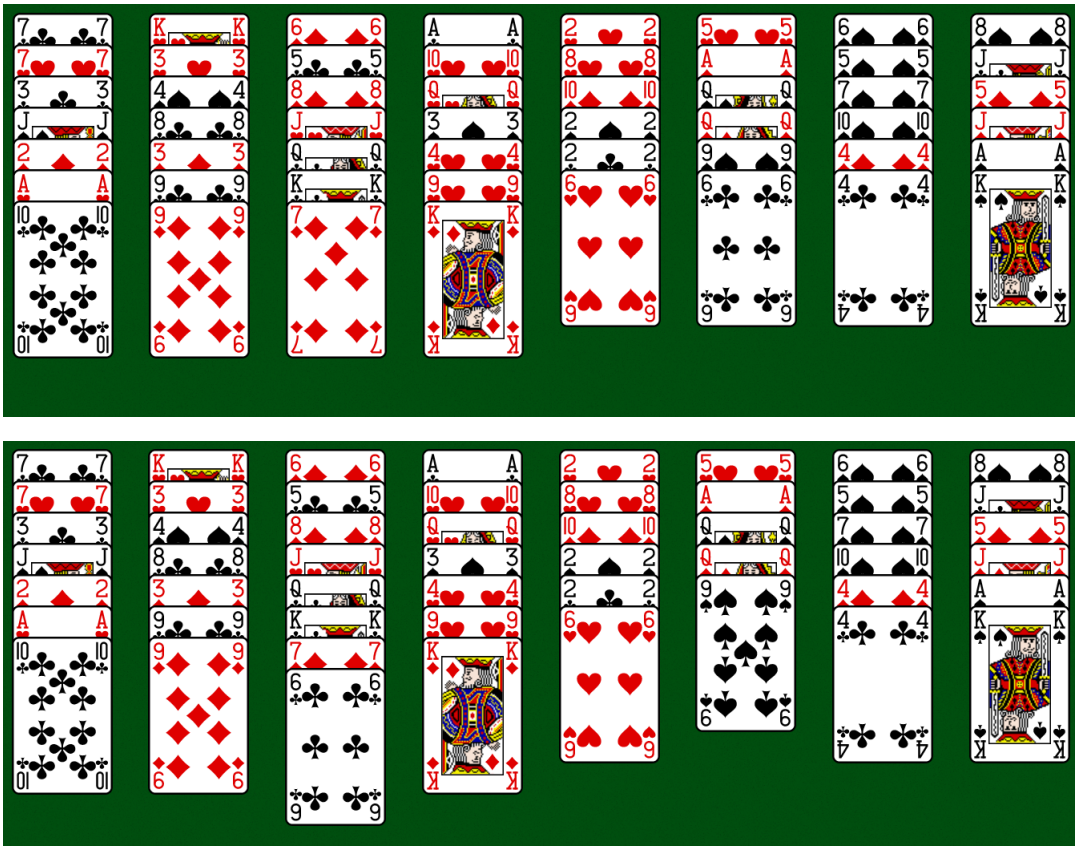

Figure 4: The 6 of clubs has been moved onto the 7 of diamonds.

When all the cards from a stack have been moved away (to other stacks, freecells or the target stacks), any card can be moved onto the resulting empty stack, not just a King [Kel09].
Some variants of the game allow cards that have been moved to the target stacks to be moved back onto the stacks or onto a freecell. This is called *Worrying Back*, and there are variants that do not allow this [Kel22].

In many implementations (Windows, Aisleriot [GNO20] and more) we have the concept of *Supermoves*. These are moves where multiple cards from a stack are moved at the same time. These cards have to be in order of ascending numerical value and alternating *suit colour*, see Figure 5. These Supermoves are not actually a separate way of moving cards, but rather an abstraction of moving cards utilizing empty stacks/freecells, see Figure 6. This means that there is a limit on the number of cards in a Supermove, being $(2^{openspaces})(openfreecells + 1)$, where *openspaces* is the number of empty stacks and freecells is the number of unoccupied freecells [Lup19].

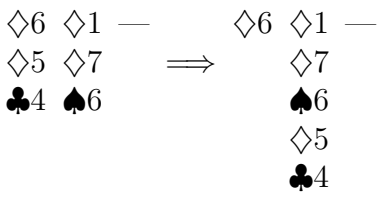

Figure 5: An example of a Supermove, with — as an empty stack.

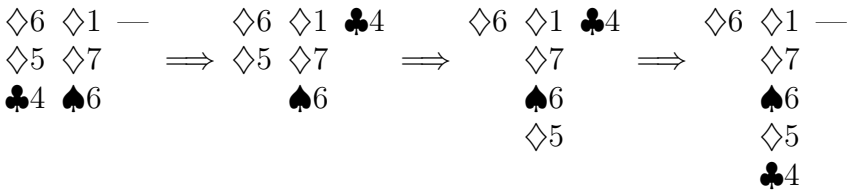

Figure 6: Previous Supermove as normal moves, with — as an empty stack.

3 Related Work

Patience/Solitaire games have existed for a long time, and much research has been done on this subject [Par91]. A popular Solitaire game is Klondike/Canfield, see Figure 7, which has been extensively studied already, including at Liacs. Rob Reitenbach [Rei21] looked at the success rate and tried to find patterns in unsolvable ones. Johan de Ruiter [dR12] researched into counting the number of unsolvable games. Pieter Bas Donkersteeg [Don10] tried solving the game using the Monte Carlo method. Marieke Kortsmit [Kor15] analyzed three different strategies for playing Klondike. Unlike Freecell, Klondike has a relative low probability of winning [BG19] with an around 82% probability at most. In Freecell all cards are visible from the start, leading to more informed decisions. Freecell also has freecells unlike other Solitaire games.

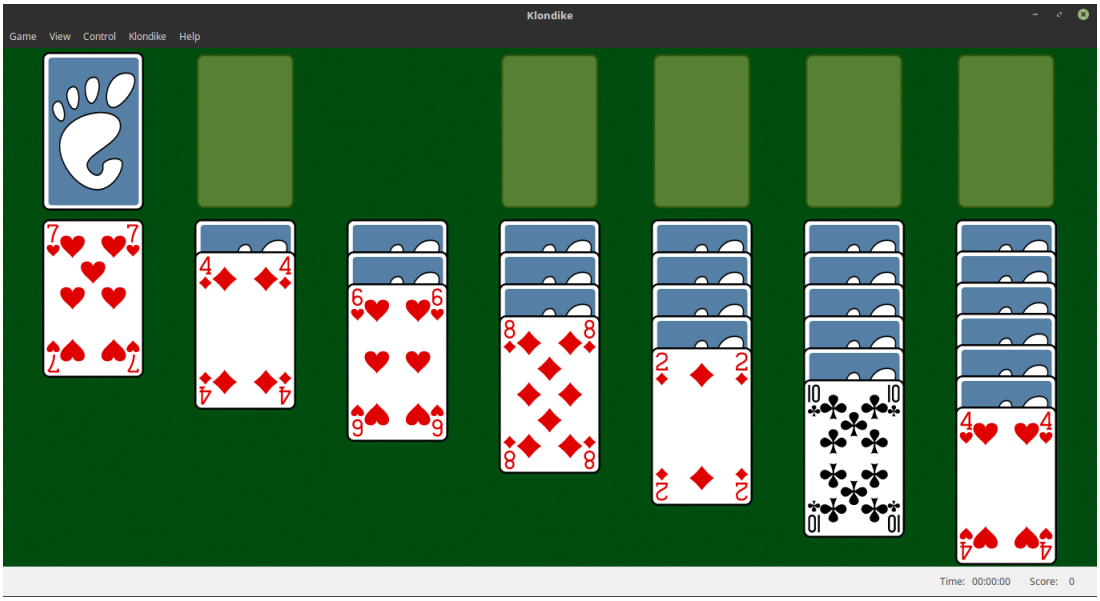

Figure 7: A random Klondike game, implementation by Aisleriot [GNO20].

Much research has been done into Freecell by enthusiasts at the solitaire laboratory website [Kel22]. A variant of Freecell was made for Microsoft in 1992, which was later included in its Windows 95 operating systems. This variant had 32,000 standard possible deals, where only one was unsolvable: game 11982, see Figure 12. Later versions had more standard deals, with the Windows XP variant having a million deals, with eight of these being unsolvable. Looking at a hundred million random games 1282 were found to be unsolvable, which gives a win rate of about 99.999% [Kel22].

Adding more freecells to unsolvable games helps with solving, for self explanatory reasons. Game 11982 becomes solvable by adding just one extra freecell. The highest number of freecells currently found to be needed is eight, for the *Magic 8* game, see Figure 8 [Kel22]. The upper bound for the number of freecells needed is at most 44, since with 44 freecells all cards can be made reachable without moving any card onto another card or one of the target stacks.

♠1	♢1	♣1	♡1	♠12	♢12	♠10	♢10
♠5	♢5	♠3	♢3	♣12	♡12	♣10	♡10
♣5	♡5	♣3	♡3	♠8	♢8	♠6	♢6
♠9	♢9	♠7	♢7	♣8	♡8	♣6	♡6
♣9	♡9	♣7	♡7	♠4	♢4	♠2	♢2
♠13	♢13	♠11	♢11	♣4	♡4	♣2	♡2
♣13	♡13	♣11	♡11				

Figure 8: A FREECELL game needing eight freecells to solve; discovered by David A. Miller [Kel22].

There exits $52! \approx 8 \cdot 10^{67}$ possible starting layouts for FREECELL games, since there are 52 playing cards in a deck. However, some of these layouts are equivalent to other layouts. For instance, if you were to swap all the clubs and all the spades, the way to solve the game would stay exactly the same. The same goes for swapping all the red cards and all the black cards. Moving columns around will also result in layouts that are effectively the same. Taking this into account, there are at most roughly $1.75 \cdot 10^{64}$ different deals [Kel22].

Research has been done into finding efficient solvers for FREECELL and other SOLITAIRE games. This includes using A* solvers [RN20, Chapter 3], which gives optimal solutions for FREECELL games. The underlying ideas could also be used for similar problems that have deadlocks [PH16]. Research has also been done into evolutionary algorithms, which have given solvers that outperform most other solvers and top human players [EHS11].

4 Methods

The implementation of FREECELL used was written in C++ from scratch as to not plagiarize other people, and to make it easier to modify for testing. The implementation does not allow for Supermoves, since they do not affect the ability of a FREECELL game to be solved (a unsolved game cannot become solvable by allowing Supermoves) and would do redundant work for exhaustive searches at the cost of computation time. Also Worrying Back is not allowed in this implementation. The program takes different parameters to determine the size of the FREECELL games:

- The number of suits there are in play, at least one suit and at most twenty-six suits.
- The number of cards in a suit.
- The number of stacks used.
- The number of available freecells.
- If target moves should always be done automatically or not (this setting is used in random games and Monte Carlo games, automatically doing all target moves would get in the way of doing an exhaustive search).
- Which game mode should be played, the game modes are mentioned below.
- If a file should be read in, and if so the file name.
- Optionally a seed to use in generating the game and playing random/Monte Carlo games.
- Optionally an offset for the seed.

Starting layouts for games (so only the stacks, since the freecells and target stacks are empty) can be read in from a text file using a format where the cards are represented by letters starting with 'a' and a number starting from '1', where the 'a' and 'c' represent the black cards, and the 'b' and 'd' the red cards. This continues for the other letters in the English alphabet, with alternating letters representing black and red cards. '1' represents an Ace, and '13' a King. The cards are separated from each other with a ',', see Figure 9 for how this looks.

a2,c2,b2,d2,a1,c1,b1,d1	♠2	♣2	♢2	♡2	♠1	♣1	♢1	♡1
a8,c8,b8,d8,a7,c7,b7,d7	♠8	♣8	♢8	♡8	♠7	♣7	♢7	♡7
a13,c13,b13,d13,b6,d6,a6,c6	♠13	♣13	♢13	♡13	♢6	♡6	♠6	♣6
b12,d12,a12,c12,a5,c5,b5,d5	♢12	♡12	♠12	♣12	♠5	♣5	♢5	♡5
a11,c11,b11,d11,b4,d4,a4,c4	♠11	♣11	♢11	♡11	♢4	♡4	♠4	♣4
b10,d10,a10,c10,a3,c3,b3,d3	♢10	♡10	♠10	♣10	♠3	♣3	♢3	♡3
a9,c9,b9,d9	♠9	♣9	♢9	♡9				

Figure 9: Text format and normal format for an unsolvable game by Hans Bodlaender [Bod96].

The implementation in general does not allow Worrying Back since it does not cause a big increase in number of games lost, and is easier to calculate. There is a variant that allows Worrying Back.

This can be used to find unsolvable games instead of the variant without Worrying Back, or can be used on already found unsolvable games to see if they are truly unsolvable.

The program automatically does all the target moves that can be done for free, the moves that can be done without causing fewer options to become available later on. For example moving an Ace to the target stack should always be done, since there are no cases where having an Ace on the field would be beneficial, Aces cannot be used to start a stack. A card can be moved to the target for free if the lowest number on the target stacks for the cards from the colours opposite to the colour of the card that is being checked if it can be moved for free is at most two lower than the card to be checked. So if you want to move a 5 of diamonds to the target, the clubs and spades have to have at least up to and including 3 on the target, so that removing the 5 of diamonds from play will not prevent you from moving the 4 of clubs and 4 of spades around. The lowest number on the target stacks for the suits of the same suit colour as the card has to be three lower. In the case of 5 of diamonds, the 2 of hearts has to be on the target stack, since if it is not on there the 5 of diamonds is potentially needed to place the 4 of spades on, which could be needed to move the 3 of hearts; if the 2 of hearts was on the target stack already the 3 of hearts could be moved to the target, and moving the 5 of diamonds to the target would not prevent one from moving the 2 of hearts. All the other target moves cannot be done for free. These moves are done automatically in random and Monte Carlo games if the setting is turned on.

There are multiple game modes in the implementation:

- Random.
- Monte Carlo.
- Exhaustive.
- Manual/player controlled.

The *manual* game mode is mainly for doing some manual testing to see how and if everything works, and not meant for recreational play. The *random* game algorithm generates a list of possible moves and randomly selects one of these moves to execute, until a game is won or until a certain number of moves is done. Depending on the given parameters it either automatically does all the possible moves to the target stacks first, or makes no distinction between target moves and non target moves for the random selection. Each move to an available freecell is counted as a separate move. If there are three empty freecells, there will be three possible moves to the freecells to select from for the random selection. All moves are equally likely. The maximal number of moves that are done in a random game is set to 5000, since FREECELL games tend to not go over 1000 moves.
The *Monte Carlo* game mode uses the pure Monto Carlo method [RN20, Chapter 5] to try to win the game. The Monte Carlo algorithm makes a temporary copy of the game state for every possible move, and does this possible move on it. After this move is done, the random algorithm is applied to the resulting game. The Monte Carlo algorithm does this multiple times for each possible move, and calculates the average resulting score for each possible move. The move with the highest score is done on the actual game and then the Monte Carlo algorithm is applied again on the resulting game. If there is a tie between multiple moves, the one that is encountered first is used. If a winning move is found it stops the Monte Carlo method prematurely and returns the winning moves chain.

The score is based on the number of moves done and if those moves are moves to the target stack or not, a move to the target gives 1000 points and any other takes away 1 point.

The *Exhaustive* algorithm calculates all the possible moves to determine definitively if a game is solvable or not. If one loses a random or Monte Carlo game it could be due to bad luck, which is not the case with exhaustive games. The exhaustive games are implemented in three different ways:

- Depth-First-Search.
- Iterative-Depth-First-Search.
- Breadth-First-Search.

The Exhaustive algorithm has to keep track of which situations have been encountered before to prevent infinite loops, see Figure 10 for an example of this. Keeping track of moves also prevents duplicate work from being done in cases where a game state can be reached from multiple different paths. The algorithm keeps track of which moves have been done already by having the current state of the game (stacks, freecells and target stacks) turned into a string and saved into an associative container. When a new move is done the resulting state is compared to the states in the container and if the state is already there no further action is taken with that move.

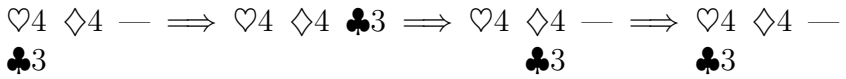

Figure 10: Example of an infinite loop, with — as an empty stack.

Depth-First-Search [RN20, Chapter 3] takes the intuitive approach to Exhaustive algorithms by recursively doing all the moves it can. If a move is done the possible moves on the resulting game are calculated and the first of these calculated moves is done. This is the fastest of the three approaches. This is probably because one rarely runs into dead ends in FREECELL. A drawback is that the winning game it finds tends to not have the fewest number of moves done to win.
The *Iterative-Depth-First-Search* [RN20, Chapter 3] algorithm proceeds in a similar fashion to the Depth-First-Search algorithm but stops after a certain number of moves have been done. This limit is increased every time the Depth-First-Search algorithm is completed, where all the moves do not go further than the limit, until a solution is found. This limit is increased by adding one to a counter every time the limit is reached without finding a solution. In effect this method is actually a version of a Breadth-First-Search. The benefit of this method is that it finds the shortest possible path to victory, however it does much work it has already done before again.
Generally *Breadth-First-Search* [RN20, Chapter 3] does a move and puts the resulting moves at the back of a queue and does the next move in the queue instead of one of the resulting moves like in Depth-First-Search. The Breadth-First-Search algorithm used in the program uses multiple queues based on the number of cards in the target stacks. It does this by counting the total number of cards in the target stacks after a move has been done. For instance, if after the move there are seven cards in the target stacks, the resulting situation is placed in the eighth queue. The

situations are played out in FIFO order, and the queues are played out from low to high. This limits the number of states that need to be saved in the state container, since it can clear the state container when it goes to the next queue, because one cannot lower the number of cards in the target stacks. See Figure 11 to see what the queues look like. Breadth-First-Search gives a more complete picture of what can be done in a game than in a game using Depth-First-Search and does fewer redundant calculations than Iterative-Depth-First-Search. It also gives shorter winning paths than Depth-First-Search, but it does take much more time than Depth-First-Search.

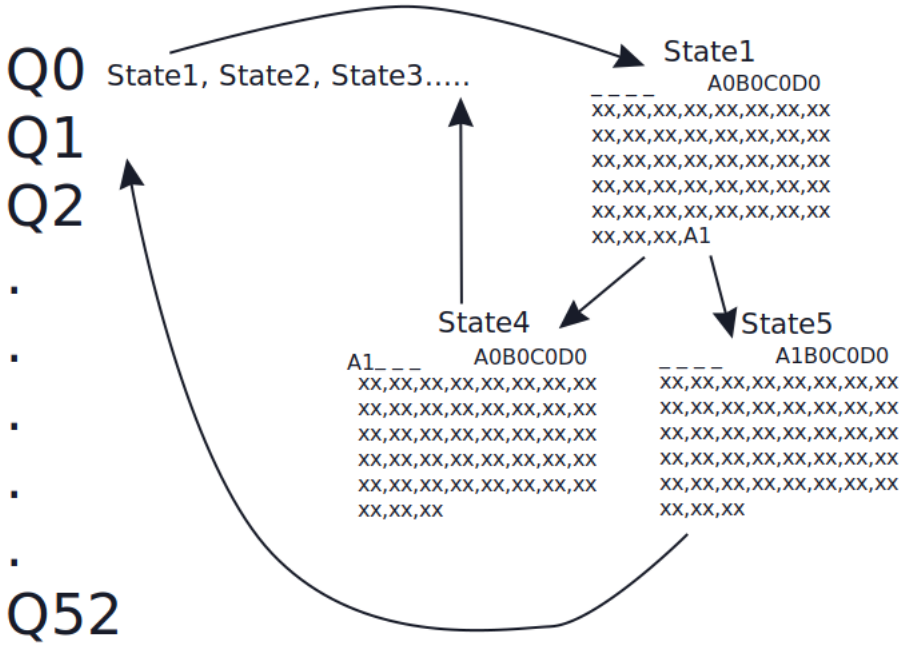

Figure 11: Example of queue usage in Breadth-First-Search.

The program only allows cards to be moved to the first available freecell if there are multiple available, since the effect on the game is the same, and calculating everything again with the card in the second freecell instead of the first one would give much wasted effort. The cards in the freecells are automatically moved left if one is removed. Similarly, it only allows a card to be placed on the first open stack if there are multiple available, for the same reason.
A card is disallowed from moving to an open stack if a freecell is available or if it is already in a freecell if it cannot be used to further build any stacks. For instance, a 2 can never have anything put on top of it, since Aces always are moved to the target stacks. So moving a 2 towards an open stack from a freecell would not give any new possibilities, and disallowing this reduces the amount of work that needs to be done.

5 Experiments

There are games of FREECELL that cannot be solved. Out of eleven million random games, 136 were unsolvable. This gives a loss rate of 0.00124%. Some of these will be used in the experiments. These games include, but are not limited to, the Bodlaender game, see Figure 9 [Bod96], Magic 8, see Figure 8 [Kel22], 11982, see Figure 12 [Kel22], and Unmoveable, see Figure 13. In the Unmoveable game the cards have been laid out to prevent any movement from one stack to another. The cards can only be moved to the freecells, and cannot be moved back.

♠1	♡1	♠4	♢1	♣2	♡6	♡10	♡11
♣3	♠3	♡12	♢12	♡8	♠7	♣1	♡13
♣13	♠6	♡5	♣4	♠9	♠11	♡9	♢3
♢11	♣5	♢5	♢8	♣9	♣10	♠13	♢7
♢6	♢2	♠10	♠12	♣6	♢10	♡4	♡7
♣11	♣7	♠8	♢9	♠2	♣12	♢4	♠5
♢13	♣8	♡2	♡3				

Figure 12: Microsoft's 11982 [Kel22].

♠10	♡10	♣10	♢10	♠1	♡1	♣1	♢1
♠12	♡12	♣12	♢12	♠8	♡8	♣8	♢8
♠6	♡6	♣6	♢6	♠4	♡4	♣4	♢4
♠2	♡2	♣2	♢2	♠5	♡5	♣5	♢5
♠3	♡3	♣3	♢3	♠9	♡9	♣9	♢9
♠7	♡7	♣7	♢7	♠13	♡13	♣13	♢13
♠11	♡11	♣11	♢11				

Figure 13: An unsolvable FREECELL game with no possible movements besides to the freecells.

5.1 Different Configurations

The game has parameters to play the game with different number of cards, different number of suits, different number of stacks and different number of freecells. These different configurations have different win rates. This is helpful since the default configuration only has about twelve unsolvable games per million, making it difficult to study large numbers of unsolvable games, see for instance Section 5.4. Smaller variants are easier to use in order to process many games quickly, and also use less memory which was a hindrance with the default game size. These different configurations can also be used to address questions about FREECELL, like how many freecells are needed to solve unsolvable games, or questions about how games with only three suits would function.

Games using two suits have a much higher loss rate compared to games using the default four suits when using comparable configurations, i.e., using twenty-six cards per suit to have the same

number of cards as the default game, or using only four stacks to have the same stack length as normal games using less cards. See Figures 14 and 15 for the loss rates; the default loss rate is 0.00124%.

Default	4 stacks, 13 cards	5 stacks, 13 cards	8 stacks, 26 cards
0.00124%	34.7%	0.0206%	65.7%

Figure 14: Loss rate for different configurations using two suits and four freecells, compared with the default configuration of eight stacks, thirteen cards per suit, four suits and four freecells.

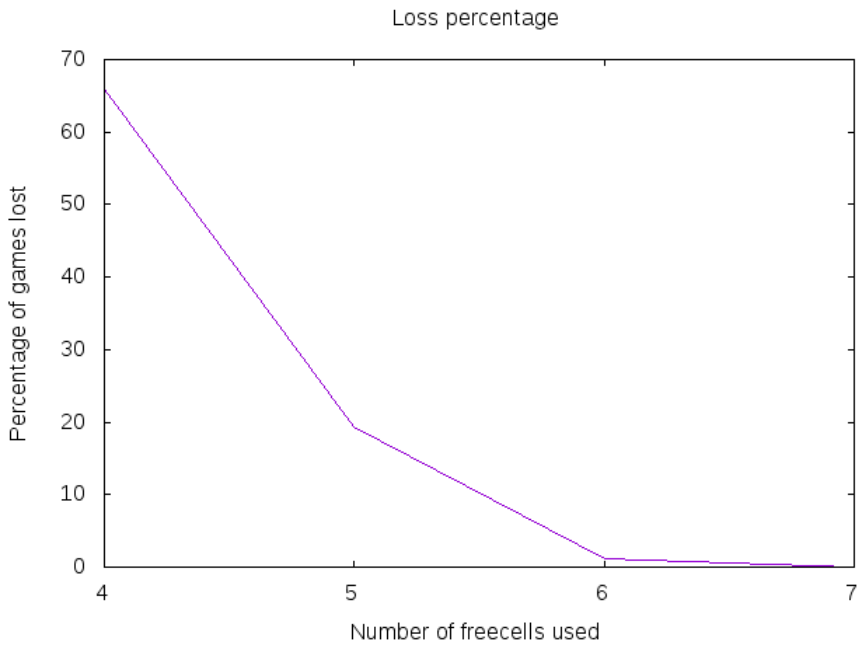

Figure 15: Loss rate for configurations using a different number of freecells and two suits, eight stacks and twenty-six cards per stack.

Games using only three suits of cards have a higher win rate than the games using the standard four suits, when the same number of stacks is used, with two out of a million unsolvable in contrast to the default of about twelve. This can be explained by the fact that with the same number of stacks and freecells, but fewer cards, the size of the stacks is much smaller, and therefore the games are easier to solve. Using only six stacks instead of eight results in stacks of equal size to the stacks in the default game, stacks of size six and seven, and gives a rate of 12,300 unsolved games out of 100,000 random games. This is lower than the default game by a factor of about ten thousand. The fact that this is lower can be explained by the fact that having only three suits available limits the moves that can be done extremely: if there is only one black suit, the red cards can only move to one other card, and both the red suits have to compete with each other for the available black cards. If the 5 of diamonds is on top of the 6 of spades, the 5 of hearts can never be moved onto another card (without moving the 5 of diamonds somewhere else).

Games using four suits will likely give the closest possible comparisons to the standard game since that also uses four suits. This means that if one wants to use different smaller configurations to learn something about FREECELL using four suits will be the most prudent. For instance looking at the question of what effect using a number of cards that gives stacks that are all the same length has, one could look at the default game size $\pm$ 1, or one could look at smaller games that are much faster to compute. See Figure 16. This could also be useful when looking for patterns in unsolvable games.
Another thing that is looked at using four suits is the effect of the number of freecells in use, see Figure 17.

Stacks	Cards Per Stack	Freecells	Loss Percentage
8	12	4	0%
8	13	4	0.00124%
8	14	4	0.0483%
6	8	2	0.800%
6	9	2	7.80%

Figure 16: Loss rate comparing games with stacks that are all the same length and games with variable stack length like the standard game has.

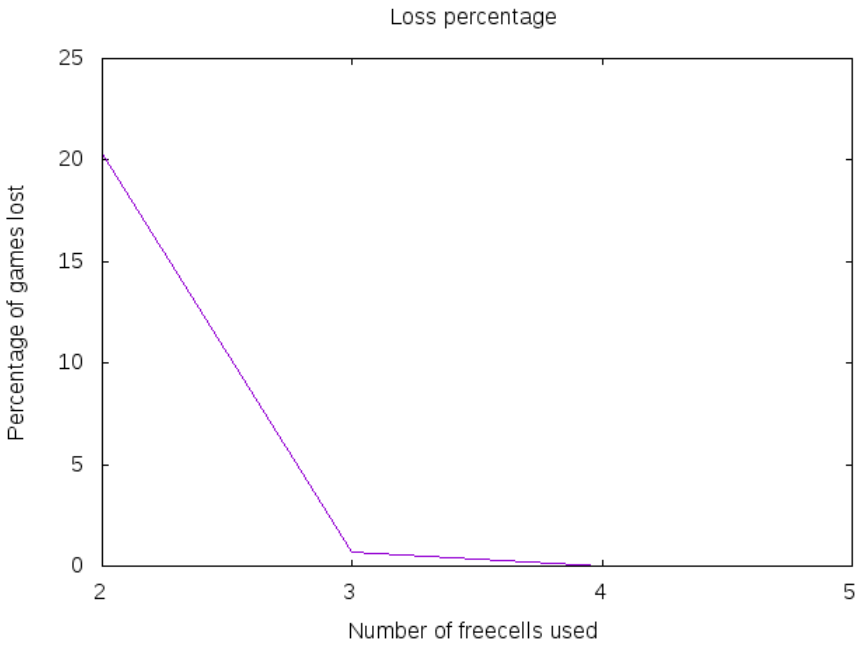

Figure 17: Loss rate for configurations using a different number of freecells, with the other variables having the default value.

5.2 One Colour

A question one might ask is "What if all suits could move onto each other, not just Red on Black, and Black on Red?". This would presumably lead to some unsolvable games to become solvable since there are more possibilities for moves, and less chance to get stuck. Out of 733 unsolvable games with the configuration of five stacks, six cards per suit, four suits and two freecells, 732 became solvable with this method. 303 unsolvable games out of 303 became solvable with the configuration of four stacks, five cards per stack, three suits and two freecells. One example of a game that is still unsolvable is depicted in Figure 13.

5.3 Worrying Back

Allowing Worrying Back (returning cards from the target stacks into play) could potentially increase the number of games that are winnable. This will increase the complexity of a game of Freecell, since it opens up many possible moves, so by default it is not allowed. It was however not too complicated to modify the program to allow Worrying Back. Since unsolvable games are pretty rare and tend to be quickly found to be unsolvable, it is feasible to run the Worrying Back variant on already found lost games. The results of doing this are shown in Figure 18.

Stacks	Cards Per Stack	Suits	Freecells	Worry Back Win Percentage
4	7	2	2	9.76%
4	5	3	1	3.51%
4	5	3	2	12.87%
4	5	4	2	7.25%
4	6	4	3	7.49%
8	24	2	4	2.87%
8	26	2	4	1.36%
8	28	2	4	0.37%
8	26	2	5	2.98%
8	26	2	6	10.20%
8	13	4	4	1.47%

Figure 18: Percentage of games that become solvable when Worrying Back is allowed.

5.4 Highest Target Reached and Maximal Scores For Lost Games

One aspect of unsolvable games we can look at is what the highest number of cards that can be moved to the target stacks is. Looking at the standard Freecell game, we only need to check up to forty cards on the target stacks, since if there are twelve cards left in play, the game is always winnable. There exists a layout of thirteen cards that is unsolvable, which means thirty-nine cards have been moved to the target stacks, see Figure 19. This scenario cannot be won, because if you want to move the cards to the target stacks the Ace of clubs (1) has to be

freed, but it cannot be freed since the King of clubs (13) is on top of it, and the King of clubs cannot be moved because all the empty stacks and freecells are in use. Of course the question is if the aforementioned scenario can be reached from a starting layout using the default configuration. A game of FREECELL is a *Maximal Lost Game* when there exists no possible games that have more cards in the target stacks whilst still being unsolvable. The upper bound for the number of cards in a Maximal Lost Games is (the number of suits · the number of cards per suit) – (number of stacks + number of freecells), since at least one card has to be blocked by another card to prevent it from moving to the target stacks. For the standard configuration this is 39.

Figure 19: Unsolvable game state with thirteen cards left, four of the cards represent freecells, eight represent the stacks.

See Figure 20 for the number of times each total was the highest total reached in unsolvable games using the standard layout. This can also be done for game configurations with fewer cards in total, and more unsolvable games, to get larger sample sizes. Using a configuration with five stacks, seven cards per suit, four different suit colours and two freecells we get the results in Figure 21. Similarly to the default scenario the possible highest number of cards in the target stacks does not exceed 21 for unsolvable games, since anything higher than that is always solvable. The highest number of cards that went to the target stacks, while still being unsolvable was 17.

Since the card configuration in Figure 19 is unsolvable, it is interesting to see if starting with a similar configuration, putting all the cards from one suit as deep as possible with the King covering the Ace, would have more unsolvable games than the default one. The opposite is true however, with zero out of a million games being unsolvable, contrasted with about twelve games out of a million being unsolvable with the default layout. A similar configuration to start out with, is to have all the suits of one colour as deep as possible. One suit is placed down in order, with the King above the Ace. Then the other suit of the same colour is placed in order on top of those cards, with the King above the Ace. This configuration has more unsolvable games than the default layout, eighty-five out of a million, however none of those unsolvable games reached a higher target number than the highest from the default games.

One way of finding Maximal Lost Games is to try to make them by hand. This is trivial for configurations using only one stack and (number of cards per suit – 2) freecells or using (number of cards per suit – 1) stacks and zero freecells. For more complicated configurations another way of finding Maximal Lost Games is to use a Simulated Annealing like method, starting with an unsolvable game and then exchanging two cards in that game to see if the number of cards that can be moved to the target stacks increases. This is done for most of the possible two card swaps in the unsolved game, leaving the top row of cards that all have the same suit, and the King on top of the Ace intact. From these swaps the one that gives the best score is the most likely to be picked, but there is still a small chance that a swap with a worse score is picked, to prevent getting stuck in local maxima. The picked swap is then done, and the algorithm con-

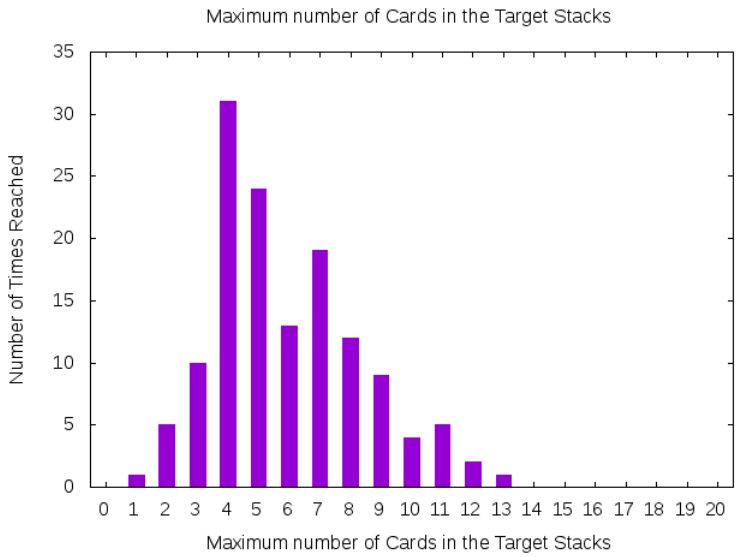

Figure 20: Number of times the highest number of cards on the target stacks occurred using the default configuration, out of 136 unsolvable games.

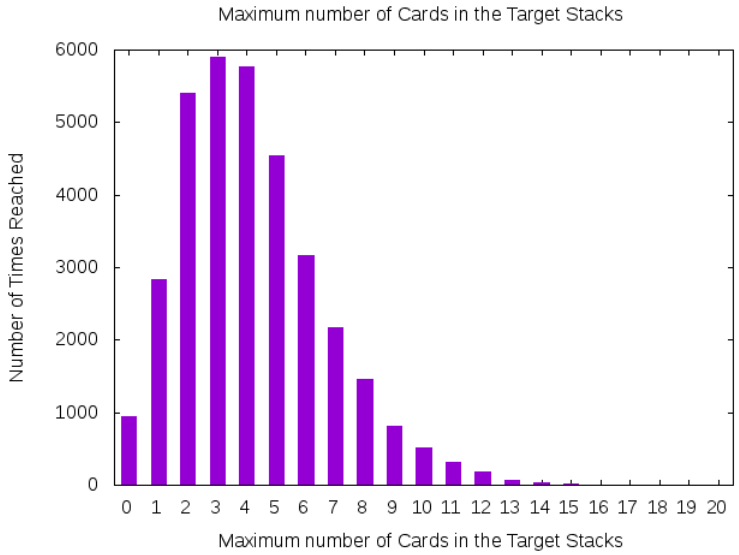

Figure 21: Number of times the highest number of cards on the target stacks was reached for a 5-7-4-2 configuration, out of 34,133 unsolvable games.

tinues. Using this algorithm multiple Maximal Lost Games have been found for configurations with four suits and ten cards per suit. See Figures 22, 23 and 24 for examples of Maximal Lost Games.

This Simulated Annealing like method can also be used to refine the found Maximal Lost Games to get the shortest possible games, the games with the fewest moves done to completely exhaustively

♠1	♠2	♠3	♠4	♠5	♠6	♠7	♠8
♠10	♣2	♣9	♣4	♣5	♣6	♣7	♣8
♡10	♣1	♣3	♣10	♡1	♢9	♡2	♢2
♢10	♡3	♡4	♡5	♡6	♡7	♡8	♡9
♠9	♢3	♢4	♢5	♢6	♢7	♢8	♢1

Figure 22: Maximal Lost Game using eight stacks, one freecell and ten cards per suit.

♠1	♠2	♠3	♠4	♠5	♠6
♠10	♣3	♢3	♢9	♣5	♣8
♠9	♠7	♢6	♢5	♡7	♠8
♣10	♡3	♡6	♡5	♡10	♣6
♡1	♣9	♢8	♢1	♣1	♣2
♣4	♡8	♢2	♢4	♢10	♣7
♢7	♡9	♡2	♡4		

Figure 23: Maximal Lost Game using six stacks, three freecells and ten cards per suit.

♠1	♠9	♠3
♠10	♡9	♠6
♠7	♢2	♣2
♠8	♡7	♢6
♣8	♡10	♣4
♠4	♡5	♣5
♢10	♡1	♣1
♠2	♣9	♡3
♢8	♡8	♢3
♢1	♣7	♣3
♢7	♡6	♡2
♠5	♢9	♣6
♢4	♡4	♣10
♢5		

Figure 24: Maximal Lost Game using three stacks, six freecells and ten cards per suit.

search it. This is done by swapping cards and seeing which swap results in the least number of exhaustive moves and going forward with that swap. The game in Figure 25 can only do 136 exhaustive moves before the game is finished, compared to the 2,464,221 from Figure 22 it was refined from. See Figure 26 for how the number of possible moves that have a certain number of cards in the target stacks compare.

♠1	♠2	♠3	♠4	♠5	♠6	♠7	♠8
♠10	♣2	♣6	♣4	♣5	♣1	♣7	♣8
♣10	♡1	♣3	♢7	♡5	♡10	♡8	♢2
♢10	♢9	♡4	♢3	♢8	♣9	♡6	♡9
♠9	♢6	♢4	♡7	♡3	♡2	♢5	♢1

Figure 25: Maximal Lost Game using eight stacks, one freecell and ten cards per suit, with a small number of possible moves: 136.

Finding a Maximal Lost Game for the default configuration, eight stacks, thirteen cards per suit,

Cards	Moves	Cards	Moves
0	0	16	5
1	9	17	1
2	2	18	21
3	1	19	0
4	1	20	0
5	1	21	0
6	1	22	2
7	3	23	2
8	1	24	2
9	0	25	2
10	0	26	2
11	2	27	1
12	2	28	23
13	2	29	8
14	2	30	0
15	5		

Cards	Moves	Cards	Moves
0	0	16	85946
1	17	17	31113
2	4	18	12519
3	4	19	2512
4	4	20	514
5	4	21	89
6	4	22	23
7	4	23	4
8	4	24	4
9	678	25	4
10	916	26	4
11	66174	27	4
12	66039	28	16
13	17646	29	22
14	95205	30	0
15	142953		

Figure 26: Number of different possible moves per total number of cards in the target stacks for two Maximal Lost Games using a 8-10-4-1 configuration.

four suits and four freecells, could theoretically be possible, see Figure 19. However, this desired game was not found while working on this thesis. Different methods were tried to find it, including trying to construct games by hand that look like they might potentially be Maximal Lost Games. This includes putting all the black cards on top and red cards on the bottom, looking at other games for inspiration like Figure 13 and the Maximal Lost Games for smaller configurations like in Figure 22. After these constructed games turned out to not be Maximal Lost Games, but still unsolvable, they were used in the aforementioned Simulated Annealing like method to try to find Maximal Lost Games, see Figure 27 and Figure 28 for the constructed games.

♠1	♠3	♠6	♠4	♠2	♠7	♠5	♠8
♠13	♣8	♡1	♣6	♡11	♢11	♣10	♡10
♢13	♣1	♣13	♡13	♡8	♢8	♣12	♢10
♢6	♠9	♢1	♡6	♡4	♢4	♣4	♡12
♢2	♠10	♣2	♡2	♡5	♢5	♣5	♢12
♢3	♠11	♣3	♡3	♡9	♢9	♣9	♣11
♢7	♠12	♣7	♡7				

Figure 27: Manually constructed game to try to find a Maximal Lost Game for the default configuration. Cards were placed in a way to minimize the number of moves not to the freecells or target stacks.

A midway point was found that resulted in a highest target reached of 39, see Figure 29. This

♠1	♠3	♠6	♠2	♠4	♠5	♠7	♠8
♠13	♣6	♣1	♣2	♣4	♣5	♣7	♣8
♣13	♣3	♡13	♢1	♢10	♢9	♢5	♡5
♢13	♠10	♡4	♠12	♡12	♡10	♡7	♢4
♣12	♡3	♢8	♡2	♡1	♢12	♡9	♡6
♡11	♡8	♢11	♠9	♣9	♢6	♣11	♠11
♣10	♢2	♢3	♢7				

Figure 28: Manually constructed game to try to find a Maximal Lost Game for the default configuration. Cards were places to have as many stacks of the same number as possible, while still trying to minimize the number of moves not to the freecells or target stacsk.

midway point was then filled with the cards from the target stacks and freecells in that midway situation, and the Simulated Annealing like method was applied to this. After this failed to find a Maximal Lost Game, a more exhaustive approach was taken. All possible previous states for the midway point were checked if they also qualified as a midway point, still reached a highest target of 39. The previous states were checked by looking at all the cards in play and putting them back in all possible spots the cards could have come from, and then checking if the resulting previous state had a highest target reached of 39. For example a card in the target stack could have come from any of the non target stacks, or any of the freecells. If the midway point can be found from an actual game, there has to be a path from the start of the game to the midway point that consistently has a target reached of 39. However none of the previous states had a highest target reached of 39, proving this midway point to not be a reachable midway point. After this new midway points were found using the Simulated Annealing Like method, none of these proved to be valid midway points either. Taking an even more exhaustive approach, this method of checking previous states was then applied to the midway point in Figure 19. Using a Depth-First approach there was a possibility of finding a Maximal Lost Game quickly. This proved to be an unfeasible way of finding a Maximal Lost Game because of the number of possible previous states is too high to compute in a reasonable time frame.

♠9	♠10	♠11	♠12	♠0	♡10	♣0	♢10
♠1	♠3	♠6	♠2	♠4	♠5	♠7	♠8
a13	♣8	♣3	♣10	♣2	♣13	♣9	♣12
♣6	♣5	♢11	♡11	♢12	♣11	♣7	♣4
♣1							
♡13							
♢13							
♡12							

Figure 29: A midway point that results in a Maximal Lost Game for the default configuration.

The highest target reached for the default game was 22, not the desired 39, see Figure 30. Finding the Maximal Lost Game in the future could work by continuing with the current method, and

hoping that random chance works out. Another theoretical, but not practical, option would be to run all possible FREECELL games, and see if any of them reach 39. Yet another theoretical, but not practical, option would be to check all previous state from Figure 19.

♠1	♠3	♠6	♠2	♠4	♠5	♠7	♠8
♠13	♣6	♣2	♣4	♣3	♣10	♣12	♣13
♣11	♣5	♢11	♢1	♢4	♠12	♣8	♡5
♣1	♠11	♢2	♣7	♡1	♢5	♢10	♣9
♡10	♠10	♡6	♡11	♡12	♢3	♢6	♠9
♡8	♢8	♢7	♢12	♡9	♡4	♢9	♢13
♡2	♡13	♡3	♡7				

Figure 30: The game with a highest target reached of 22 cards on the target stacks, the highest currently found for the default layout.

5.5 Patterns

One goal is to find patterns in unsolvable games. This is done by first looking at a small number (4) of smaller games (four stacks, six cards per suit, four suits and four freecells) to find initial patterns, and then looking at a larger number of default games (20) to see if the patterns hold and if some more patterns can be found. Four patterns were found:

- At the start of the game unsolvable games tend to not allow any moves besides moves to the freecells to be played. In all 4 out of the 4 small games this is the case. In 16 out of the 20 default games this is also the case, and in the 4 games where this is not the case there are no subsequent moves that are unlocked by doing these initial moves, still forcing cards to be played to the freecells early.
- Another commonality is that the Aces are high up in the stacks (covered by many cards), often in the top row. All 4 small games have all their Aces in the top half of the stacks. For the default layout all 20 have at least two Aces in the top half, with many games having in the top.
- All games using both layouts have multiple occurrences of the same number being stacked on itself, i.e.,

  ♠5
  ♣5
  ♡5

  Sometimes there are stacks where the same number is being stacked on itself, but with another card in between. Depending on what card is in between the stacks would be equivalent or less solvable to stacks where the order of cards would be swapped, for example ♡10, ♠1, ♣10 is less solvable than ♡10, ♣10, ♠1.

- Many games using the default layout have high cards low in the stacks (few or no cards covering them). Only 2 out of the 20 games do not have a King on the lowest row, and even then there is a King on the second lowest row. Many games even have multiple Kings on the lowest row.

6 Conclusions and Further Research

In this thesis the Solitaire game Freecell is studied. This game consists of trying to move the cards from a standard deck to target stacks (one for each suit), from eight different stacks they have been spread out over. The cards are moved by putting them on cards that are a different colour and one higher, or utilizing freecells. Unsolvable games were mainly looked at, and how far those unsolvable games were able to go regarding the number of cards moved to the target stacks.

6.1 Unsolvable Patterns

One of the main goals of this thesis was to find patterns in unsolvable games of Freecell. Four patterns were found:

- At the start of the game unsolvable games tend to not allow any moves besides moves to the freecells to be played.
- Aces are high up in the stacks (covered by many cards), often in the top row.
- Cards are stacked on other cards with the same number, i.e., a ♡5 on a ♣5 on a ♠5.
- High cards are low in the stacks (few or no cards covering them).

In general it looks like that having to fill the freecells early on in the game makes the game harder to solve, and games with fewer possible move options are more likely to be unsolvable.

6.2 Predicting Unsolvable Games

Another question asked in this thesis was if there is a way to predict if a game is unsolvable from the start. Looking at the aforementioned patterns however is not enough to see if a game is unsolvable, and no way of predicting has been found. In Figure 31 one can see an unsolvable game, and all four of the patterns hold in this game. However, if the ♣7 and ♡7 are swapped the game becomes solvable, even though all four patterns still hold, see Figure 32. Looking at both of these games it is very hard, if not impossible, to see why one of them is solvable, but the other is not.

♣7	♢4	♠8	♠10	♡2	♠2	♣12	♡9
♡10	♠1	♡1	♣1	♢2	♣6	♣11	♠7
♠12	♡4	♡6	♣10	♢12	♢5	♠13	♣2
♢3	♢8	♡8	♠4	♠11	♢1	♣4	♢10
♡3	♠3	♢6	♢7	♠6	♣5	♣8	♣9
♡12	♣13	♡5	♢11	♠9	♣3	♢13	♡11
♡13	♠5	♢9	♡7				

Figure 31: An unsolvable game Freecell, satisfying multiple patterns in unsolvable games.

♡7	♢4	♠8	♠10	♡2	♠2	♣12	♡9
♡10	♠1	♡1	♣1	♢2	♣6	♣11	♠7
♠12	♡4	♡6	♣10	♢12	♢5	♠13	♣2
♢3	♢8	♡8	♠4	♠11	♢1	♣4	♢10
♡3	♠3	♢6	♢7	♠6	♣5	♣8	♣9
♡12	♣13	♡5	♢11	♠9	♣3	♢13	♡11
♡13	♠5	♢9	♣7				

Figure 32: The game from Figure 31, made solvable after swapping ♣7 and ♡7, still satisfying the same patterns.

6.3 Highest Target Reached Problem

Looking at the Highest Target Reached Problem for unsolvable games, most unsolvable games can only move a few cards to the target stacks at the same time at most. This takes the shape of a bell curve skewed to having few cards in the target stacks, see Figure 20 and Figure 21. This result makes sense since in most unsolvable games one can at least free an Ace or two if one focuses hard on doing that, even if it does not lead to solving the game. On the other side of the curve it makes sense that there are fewer unsolvable games that have a high number of cards in the target stacks, because the more the target stacks get filled, the fewer cards there are in play, having less chance of cards blocking other cards, and more chance of having more empty stacks and empty freecells. At a certain point, if enough cards are in the target stacks, it is impossible to lose a game. The moment in the game just before this point is reached is looked at with Maximal Lost Games. These games have been proven to exist for some configurations, see Figure 22 for example, however an example of this has not yet been found for the default configuration.

6.4 Same Stack Length

One thing that was looked at was how the win rate of games that have beginning stacks that are all the same length compared to games that have stacks of different lengths. This, however, gave inconclusive results, see Figure 16, since the difference in win rate can also be explained by the fact that having fewer cards in total makes games easier to solve. If for instance having fourteen cards had a higher win rate than using thirteen cards, a possible explanation could have been the stack length, but with the actual situation on hand nothing meaningful can be said.

6.5 Worrying Back

We can see that Worrying Back does in fact solve some unsolvable games. This was expected since it gives more freedom to make moves, which tends to increase the win rate. For the default configuration this increase in win rate was minor, with only 1.47% of unsolvable games becoming solvable. However, this is not the case for all configurations, e.g., using four stacks, five cards per stack, three suits and two freecells, the rate of unsolvable games that become solvable is 12.87%. There is no single variable that clearly correlates with win rate increasing when Worrying Back is allowed, see Figure 18. For some configurations it seems that Worrying Back is not enough to increase the win rate by large amounts, e.g., using eight stacks, twenty-six cards per stack, two

suits, the Worrying Back win rate actually increases with the number of freecells used, which indicates that in this case easier games have higher increases. A possible conclusion from this is that Worrying Back is more useful for configurations where more unsolvable games are closer to being solved.

6.6 Further Research

Further research can be done by looking into unsolvable games for Solitaire games similar to Freecell, like for instance Baker's Game, Eight Off, Seahaven Towers, Penguin and Stalactites or even into other Patience/Solitaire games like Klondike. More research could also be done into finding Maximal Lost Games for Freecell. Maximal Lost Games could potentially also be defined and found for other games similar to Freecell, and other Solitaire games like Klondike. Research could also be done in predicting losing games from the start for other Solitaire games, maybe it will be more feasible for those.

References

[BG19] Charlie Blake and Ian P. Gent. The Winnability of Klondike Solitaire and Many Other Patience Games. *arXiv*, 2019.

[Bod96] Hans L. Bodlaender. An Unsolvable Instance of Freecell: A Game from Windows'95. https://webspace.science.uu.nl/~bodla101/d.freecell/freecellhtml.html, 1996. Last accessed 7 June 2022.

[Don10] Pieter Bas Donkersteeg. Klondike Strategies using Monte Carlo Techniques. Bachelor thesis, LIACS, Leiden University, 2010.

[dR12] Johan de Ruiter. Counting Classes of Klondike Solitaire Configurations. Bachelor thesis, LIACS, Leiden University, 2012.

[EHS11] Achiya Elyasaf, Ami Hauptman, and Moshe Sipper. GA-FreeCell: Evolving Solvers for the Game of FreeCell. In *GECCO '11: Proceedings of the 13th Annual Conference on Genetic and evolutionary computation*, pages 1931–1938. Association for Computing Machinery, 2011.

[GNO20] GNOME. Apps/Aisleriot - GNOME Wiki. https://wiki.gnome.org/Apps/Aisleriot, 2020. Last accessed 27 July 2022.

[Kel09] Michael Keller. Freecell tutorial for beginning players. https://solitairelaboratory.com/tutorial.html, 2009. Last accessed 7 June 2022.

[Kel22] Michael Keller. Freecell FAQ and Links. https://solitairelaboratory.com/fcfaq.html, 2022. Last accessed 7 June 2022.

[Kor15] Marieke Kortsmit. Strategies for Klondike Solitaire. Bachelor thesis, LIACS, Leiden University, 2015.

[Lup19] Arcanist Lupus. FreeCell: How Many Cards can be Moved at Once? https://boardgames.stackexchange.com/questions/45155/freecell-how-many-cards-can-be-moved-at-once/45157#45157, 2019. Last accessed 7 June 2022.

[Par91] David Parlett. *A History of Card Games*. Oxford University Press, 1991.

[PH16] Gerald Paul and Malte Helmert. Optimal Solitaire Game Solutions Using A* Search and Deadlock Analysis. *AAAI Publications, Ninth Annual Symposium on Combinatorial Search*, pages 135–136, 2016.

[Rei21] Rob Reitenbach. Determining the Success Rate for the Game of Solitaire. Bachelor thesis, LIACS, Leiden University, 2021.

[RN20] Stuart Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Pearson, 4th edition, 2020.