ISYE 6644 – Summer 2022

Project Report

Team Member Names:

Xiaolu Su

Project Title:

Solitaire Simulation

I. Project Abstract

The final goal of the solitaire game is to consecutively store a total of 52 randomly scattered cards in their corresponding suit. The three major actions of "move", "draw", and "store" have weaving relationships with the purpose of enabling each other. Since all the "face-up" cards have more freedom between locations, as long as all the "face-down" cards are unlocked, the game can immediately be regarded as solved.

The major finding of this project is that it's very unlikely to play the same game set twice, since there can be about 3.96e+73 possible game settings, and each run has no influence on any other run. Therefore, the game can be regarded as pseudo-random with an extremely large cycle. Each game set can be seen as a Bernoulli experiment. Theoretically, it's possible to find superior strategies based on the win vs. loss ratio of different strategies on the same simulation run, as long as the seed is the same. In another word, the results of the game follow Binomial distribution. However, there might not be enough data to support that, because different strategies in this project gave the same results.

II. Background and Descriptions

This project shows interest in this simulation game because this popular game seems to mimic the issues that could be faced in cargo or inventory storing and retrieving management. With the same frustration, there's no definite plan to solve the problems, but we can let the simulation system to help us efficiently choose a relatively better solution to deal with certain situations.

Variables

Here are some definitions of the variables that have been straightforwardly named for the convenience of this project in order to find the most appropriate approaches for them:

- Upcard: a card that faces up; also a card with more freedom of movement.
- Downcard: a card that faces down; also a card that has no freedom unless removing the cards above.
- Table: the major playing area of this game where 7 upcards and 21 downcards are put when a game is initialized.
- Foundation: the area that has 4 slots that only allow cards with the same suit to be stored together.
- Waste: the group of upcards that are ready to be moved to a suitable location as long as you are patient enough to shuffle from the first to last, for how many times you want.
- Sequence: a queue of cards that are linked by having consecutive numbers. One type of sequence is the table sequence, every adjacent card should have a different color than the card itself. Another type of sequence is the Foundation sequence, which only allows the same color and suit to build a queue.
- Front: the first upcard of a sequence.
- End: the last upcard of a sequence.
- Single End: an upcard that is the only upcard in its column, which is either above a downcard or by itself.
- Last: a card that has no downcard beneath.
- Move: a card in movement.
- Hold: a card that takes the Move card, which means its number is x +1.
- Aim: a card that is targeted to move because it can open up a downcard.
- Block: a card that is right above the aim card.
- Need: a card or cards that need to be stored in Foundation in order to remove and store the block card.
- Sub: a card that shares the same color and number.
- Buffer: a card whose only purpose is to hold the block or insignificant cards.
- Spare: a card to get back from Foundation to get the buffer card needed.
- Top: the most currently updated card in a column of Foundation.

- Help: opposite of spare card, it's the card to be stored in Foundation so that the card targeted can also be stored.

III. Major Findings

Strategies and Flowchart

There are three strategies being considered:

Strategy_0: The basic strategy that is considered the foundation of the project. The goal of this strategy is to open as many downcards and store as many cards to the foundation as possible. It has been proven to succeed in some game simulations, but its success rate is not expected to be high.

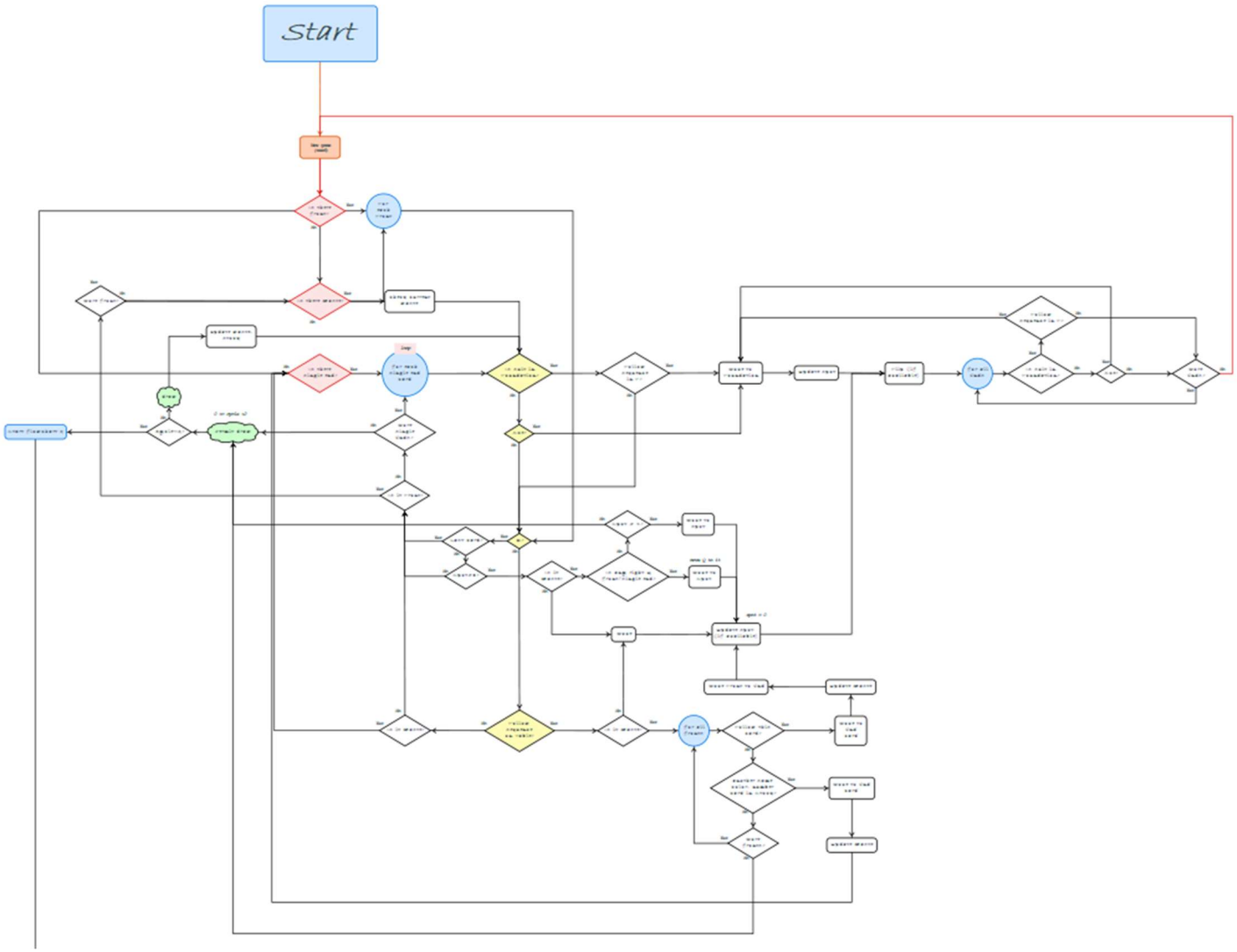

Figure 1 Flowchart: Strategy 0

As soon as a new game starts by random generation, there are a few items to check, and they should follow a certain sequence in order to make it more efficient for future function loops. As Figure 1 Flowchart: Strategy 0 has shown, this basic strategy involves a few steps. First, the orange block right after the starting point is game initialization. Then, the pink diamonds are the main checking points: fronts, wastes, and the single ends. It might not make sense to check the front and waste at the beginning of the game because there would be none, but it will certainly show its purpose in later steps.

The waste is not supposed to be used immediately, so the first item the program is checking is actually the single ends cards because they are the closest to the targeted object: downcards. If they are able to solve some sequence by themselves, it will be the most expected, because sometimes cards in the Waste have duplicate properties(color, number) of the upcards on the table; if we draw any of those cards from the Waste, we might lose an opportunity to open one of the downcards. A blue circle indicates a loop to check each of the available items once identified. After we determine to loop through a certain card type, the yellow diamonds are the key branching points to decide whether or not, or how we should use them. To be specific, we first check if the card can be sent to the Foundation, then we check if it's a King which can only go to an empty spot, then eventually, we check if the card can help create a connection with other cards. All with the purpose to open the downcards. In short, the yellow diamond splitting points comprise the central part of the strategy. Once there's no more move to be made between the Table and Foundation, the program will reach the final resort, the green cloud, to draw cards to help.

The reason that the program check items with such a sequence ("front > current waste card > single end") is that the fronts are the closest to the downcards or empty spots, especially after a few steps into the game. But they are also frustrating due to the cards being locked together, unless they can be hold by a substitute card. So they should be taken care of first. Next is the most current waste card, which we might miss a few good options because we lack some cards that show up right after them. Of course, we don't want to loop over the whole system again to retrieve those cards. In addition, as we proceed far into the game, there are usually not many single-end cards left, and we definitely don't want to waste the step and time to move the cards that are already in a sequence unless they can be stored in the Foundation. Last, if you might notice from the flowchart, at the top right corner, there's a loop that is the common destination no matter which branch we choose, and that loop is to check every end card and store them in the Foundation if available after we finish messing with the other more complicated steps.

Strategy 1: This strategy is the first one that tries to find improvements in the simulations. It needs a transition step as shown in Figure 2 Flowchart: Strategy 1-1. This strategy would take a closer look at the front card. And in order to use this strategy, we need to pass a few requirements as suggested by the decision diamonds colored yellow. First, we need to find if the front is able to find all the cards it needs to pave its way to Foundation. We will name it an aim card as soon as it's verified. Next, we need to look at each of the block cards in its sequence and check if it can find its way to go to the Foundation too. For them, we can be more flexible if they have their substitutes appearing on the Table or ready to be used in the Waste. If these two rules can be satisfied, then we can start using the strategy by sending the cards to Foundation one by one. Along the way, sometimes we need to pull some cards back from the Foundation to store more cards in it.

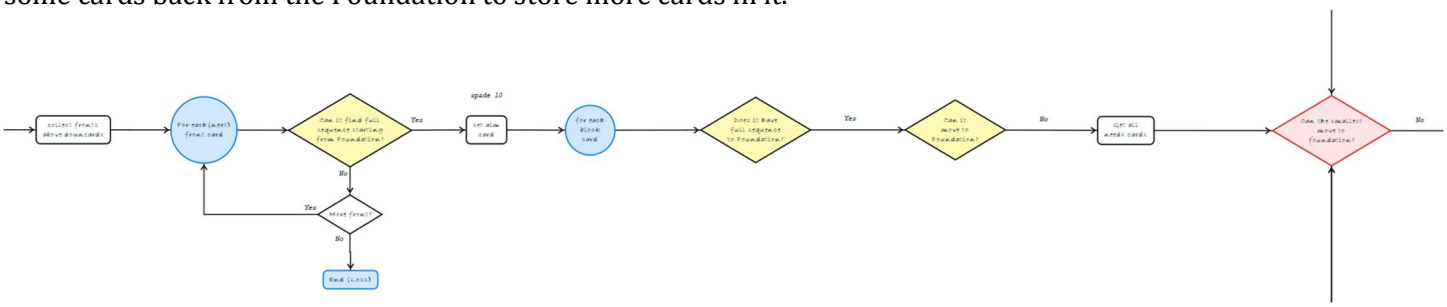

Figure 2 Flowchart: Strategy 1-1

This strategy would suggest looking at the smaller number card first and then moving up. When we are this far into the game, we don't expect the cards would be as easily moved to Foundation as in the earlier process. But to accomplish that, we can find substitutes of cards on the Table, that's where we start to move the cards from the middle of the sequence instead of the front or the end. However, sometimes we need to find a substitute for a card that holds the substitute of another card. Or if necessary, cards in the Foundation should be reused as spare cards too. It involves using seemingly redundant back and forth steps to release the cards we want, as shown in Figure 3 Flowchart: Strategy 1-2.

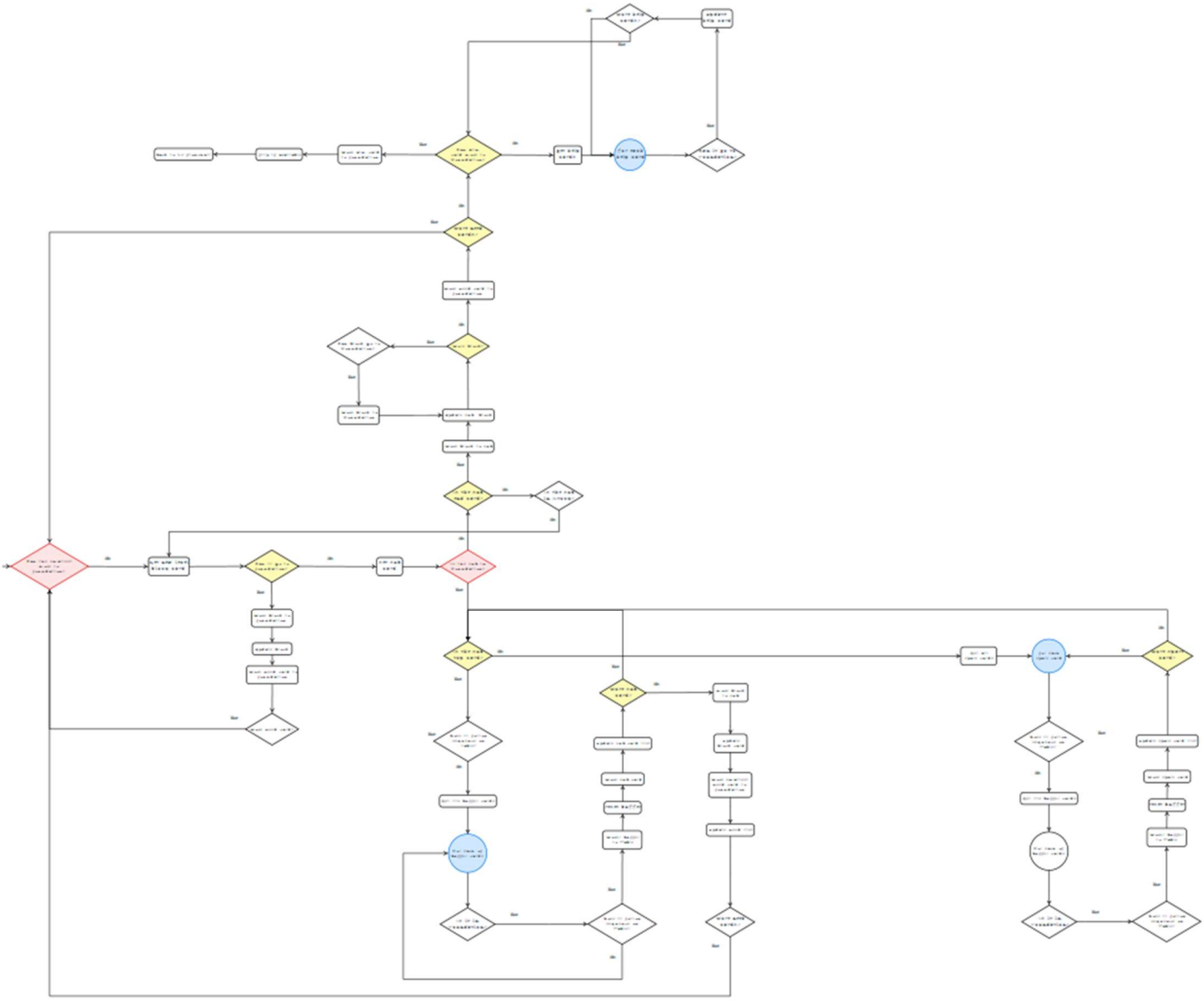

Figure 3 Flowchart: Strategy 1-2

Strategy 2: Another improvement strategy is to memorize the steps as a human would along with some backstepping tricks. It's common to find a card with two different moving options. And due to randomness, there's no fixed rule to rate one over another. When it happens, we can try each of the two

options and see which step would open up more opportunities to release downcards. This method would be implemented on the Strategy 0. We only need to identify certain binary choices when they occur, and keep a copy of the table setting, such as Table, Foundation, and Waste. If we are more likely to get stuck at one option, we simply discard the progress and move back to the memory point to pick the other option.

Parameters

Random game simulation cycle: 3.963542727900181e+73

Calculation explanation:

From total 52 cards, randomly choose 24 cards to be in Waste, where sequence doesn't matter. On the premise of that, from the rest of the 28 cards, randomly choose 21 to be the downcards and 7 to be the upcards, where sequence is essential.

	Success Rate (100 Runs)	Success Rate (5000 Runs)
Strategy 0	0.03093	0.01936
Strategy 1	-	-
Strategy 2	0.03093	0.01936

Figure 4 Table of Parameters

Evaluation and Future Improvement

There are some questions provoked by this project, even though they haven't been able to be answered:

Question 1: Strategy 1 is still under development.

It's basically an effort to mimic human thinking. When no more cards can be moved with the assistance of waste cards, we would try to think of new possibilities the stored cards can create. However, this step involves guessing, and trying, where a large number of back-and-forth steps and variables can be involved. The simulation functions I designed for Strategy 0 might not be sufficient for this next-level Strategy 1, because Strategy 1 is not only collecting data from the ends, it also requires enough information from the middle cards. Therefore, to achieve the ability to collect enough data, instead of a few decision points that are only providing True/False results to identify the unique situations, every deciding point will need to automatically collect all the card data for future use. Undoubtedly, it will require a lot more storage, and it will take more time to run the same number of iterations.

Question 2: Why does Strategy 2 show the same success rate?

This strategy has been developed for the purpose of the only situation where a card can have two moving options. In the Solitaire game, every card shares similar properties with another card, so this might be the reason that a card blocking either one of the two similar cards barely has any effect on the progress. In short, Strategy 2 does not show any significance. Yet, it can still be improved by changing the focus on the flipped card options.

Question 3: Will different card distributions significantly change the difficulty of the game?

This question leads to imagining the increased number of downcards or redesigning the columns' composition to spread the downcards more evenly, assigning certain range of numbers to the downcard, or using the same property of the card to block the card below, which would all assumably add more challenges to the game. It might make it impossible to solve some of game simulations.