Go Fish and Probabilities

Scott Munro, Andrew Gordineer

Austin, TX, United States

Abstract

Many people enjoy playing the card game Go Fish. Generally, the game is played by randomly guessing what cards a players opponents may or may not have. We set out to develop a system that uses probability to help players make better educated guesses. Using these educated guesses, we propose that a player's average win rate can be increased.

1. Important Resources and Sources

Project Source Code:
`http://github.com/gordineerandrew/Go-Fish`
Mean and Std Deviation Tools:
`https://www.easycalculation.com/statistics/standard-deviation.php`
Normal Curve Generator:
`http://davidmlane.com/hyperstat/z_table.html`

2. Logistics

This project was developed using the Java programming lanuage and packages defined with the standard Java 7 SDK. Java was used for it's ability to rapidly prototype the GoFish simulation and to quickly develop simple object oriented designs. While other languages such as C can make more efficient use of the processor (which improves performance) and other languages such as Python or R are used in scientific fields, Java was chosen because the simulation is fairly lightweight and the ability to quickly write code outweighed the advantages of other languages.

3. GoFish Rules

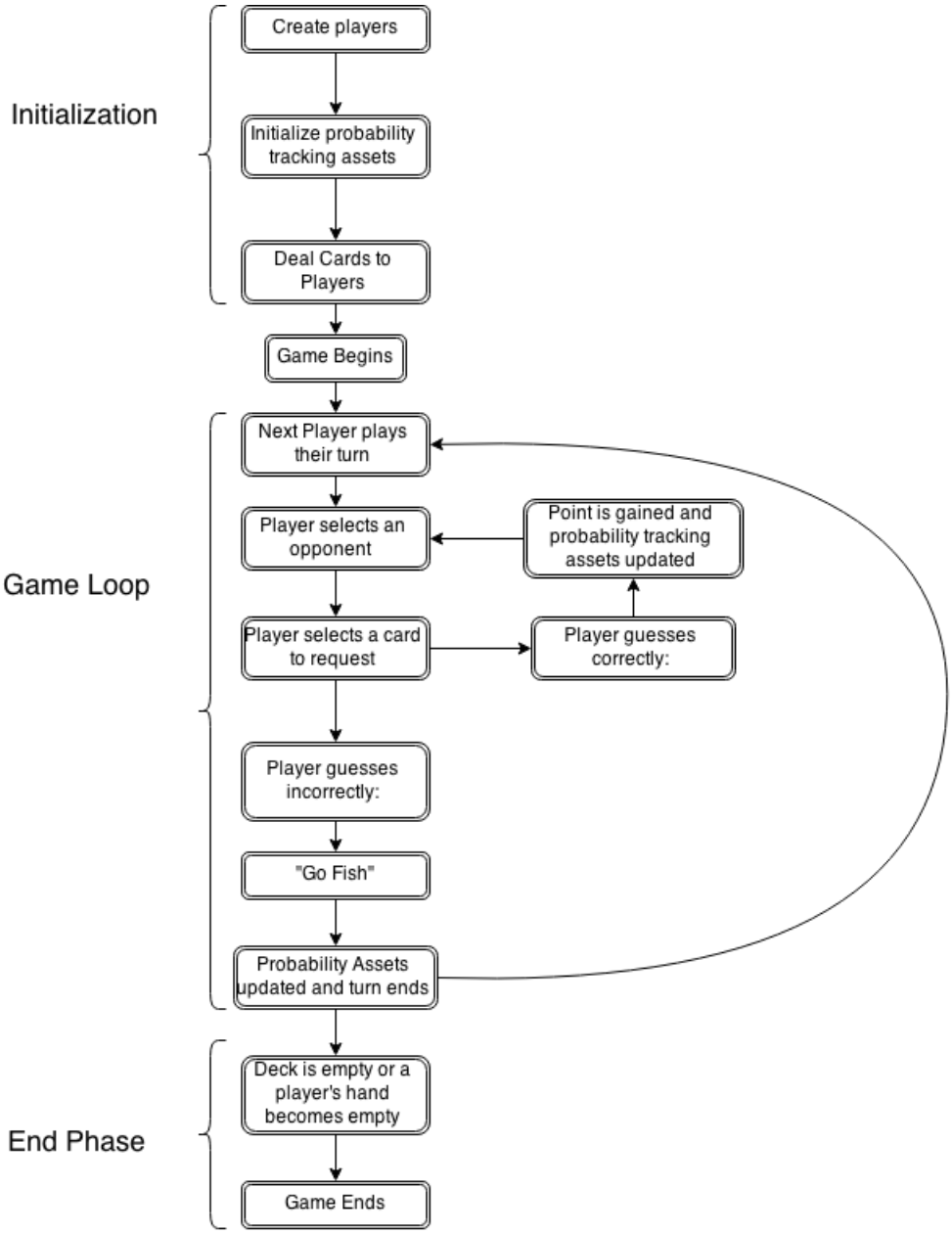
Because Go Fish is not explicitly defined anywhere, we defined a set of guidelines and rules for our particular simulation.

- At the start of the game, each player is dealt seven cards, creating books as needed.
- The game cycles in turns between each player until either a player loses all cards from their hand, or the deck is empty.
- Each turn consists of a player requesting a card in their hand from an opposing player. If that opposing player has the card, then the asking player is given that card to form a book and player gains a point. If the requested player does not have the card, then the asking player must Go Fish! and draw a card, ending their turn.
- Books are comprised of a pair of cards with identical values (i.e. pair of twos, aces etc).
- When the game is over a winner is determined by finding the player with either the maximum amount of books or, in the event of a tie, finding the player with no cards left in their hand.

4. What We Accomplished

We created a program that simulates the game of Go Fish, displaying the probabilities of each value of card for each of the players opponents. These probabilities are constantly updated based on events occurring in the game (i.e. Asking for cards, being asked for a card, handing over a card, making a book, and denying having a value). The AIPlayers play the game by randomly asking one of their opponents for a card that is currently in their hand. This version of AI is random, but representative of how people generally play GoFish, disregarding information that is gained through the games progression. A headless, logging mode was also established so that multiple iterations of the simulation could be run and recorded without needing explicit human input.

4.1. Basic Control Flow Diagram



4.2. How Probability is Tracked

The probability for each value of card is tracked by keeping a running count of the number of cards in each of the players opponent's hands known to

not be that value. This is accomplished by incrementing the count each time a player gains a card (i.e. draws a card), decrementing the count whenever a player loses a card (i.e. makes a book or gives up a card to an opponent), and zeroing a cards value if the player definitely doesnt have that value (i.e. they have just formed a book or denied a request for the value).

4.3. How the Probability is Calculated

The probability is calculated by using a combination of what is known about the unknown cards in a players hand that could be a specified value, the total number of unknown cards in the game that could be the specified value, and the number of cards of that value remaining in the game.

Formula (R = player being asked, C = card being asked for)
X = the number of cards in the R's hand that could be C
Y = total unknown cards in the game that could be C
Z = Number of C remaining in the game and not in your hand

$$\Pr(\text{C in Rs Hand}) = \frac{X}{Y} * Z \tag{1}$$

4.4. Computer Simulated Players

To appropriately simulate GoFish, multiple players are needed. To simplify the development of the simulation environment, the extra players are implemented as AI-Players instead of as other Human Players. Whereas the human player has to manually select a player and a card to ask for, the AI-Players randomly choose one of their opponents and a random card that exists in their hand during each of their turns. Aside from just acting as an extra player, these AI-Players act as a contrast to how the Human Player selects their cards. The Human Player can make educated guesses, but the AI-Player makes naive, random guesses that dont adapt to the new information that is made available to every player as the game progresses.

4.5. Extension

This project could be extended in a few interesting ways. A better user interface could be developed that makes use of graphical components. A variety of difficulties of AI could be created, taking advantage of the probability calculations being available and evolving their behavior based on the current state of the game.

5. Experiment

5.1. Overview

It can be accepted with common sense that having access to more information would positively affect a player's win rate, but it is an interesting challenge to see exactly how much it can benefit a player.

5.2. Procedure

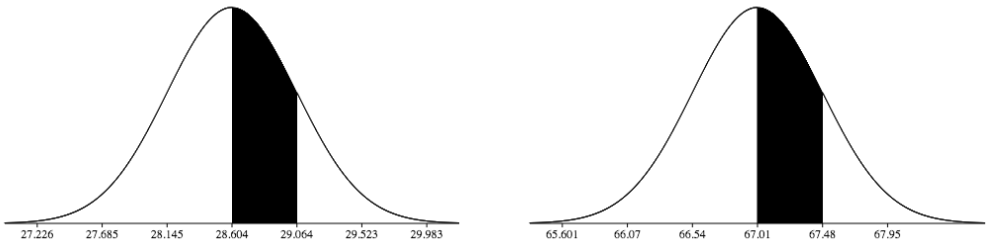
To compare and contrast the naive strategy to our strategy with educated guesses, both needed to be tested in a standard way. The standard test we designed was to run the GoFish simulation with 4 players 10000 times and get the percentage of wins. This was repeated 1000 times and the results of these trials were used to generate a reliable mean and standard deviation. In turn these measures of central tendency can be used to form a normal curve to evaluate each strategy in comparison to each other.

5.3. Data

Naive Strategy	Educated Strategy
$\mu = 28.604$	$\mu = 67.0099$
$\sigma = 0.45955$	$\sigma = 0.4698$

μ = Mean

σ = Standard Deviation



5.4. Results

The results are clear. Using the educated strategy with probability awarness results in winning twice as many rounds of Go Fish as using the naive strategy. Furthermore, the standard deviation of both strategies is so miniscule that there is virtually no chance that the naive strategy will ever deviate anywhere near the results of the educated strategy. The experiment has been proven successful.

6. Some Challenges During Development

6.1. Learning about Go Fish

Go Fish is not explicitly defined and is riddled with house rules, variations, and inconsistencies. To develop a proper simulation, a specific subset of rules had to be established and this set of rules needed to cohesively work with our project.

6.2. Tracking the probability

Giving different portions of our code the correct access to the correct information proved troublesome when we began trying to work from a strictly object oriented programming paradigm. A simpler, more imperative form of object oriented programming was adopted to fit our project.

6.3. Calculating Probability

Determining a method of calculating the probabilities of each of the card values, without having to compute very intense functions proved to be difficult. Issues concerning whether or not the probabilities were to be distributed among opponents and how to consider a multitude of unknown cards made the problem difficult to conceptualize.

6.4. Designing the user interface

Displaying the appropriate information to the user in an organized and readable manner proved difficult within a text-based interface. A lot of time was spent defining formatting standards and input delays so that a player could appropriately digest all of the information that would need to be presented to them.