Maverick Solitaire

Saul Spatz

September 15, 2022

Abstract

In Maverick solitaire, 25 cards are dealt at random, and the player attempts to partition them into five pat poker hands. I have computed the exact probability of winning, or rather, of the existence of a solution. The exact probability is ????. In the remainder of this article I describe how the background of the problem, and how the computation was performed.

1 Introduction

On January 19, 1958, when I was 11 years old, the popular television program *Maverick* aired an episode titled "Rope of Cards." Bret Maverick, a gambler played by James Garner, bets that he can separate 25 randomly dealt cards from the ordinary 52-card French deck into five pat poker hands, where a pat hand is a straight, a flush, or a full house. (Straight flushes and royal flushes are considered special kinds of flushes in this game.) He wins the bet, and later states that the game can be won "practically every time." My personal recollection is that he said "49 times out of 50," but I cannot find any support for this, so I believe he must have said it in a later episode.

The story goes that the following day, novelty shops all over the United States sold out of playing cards, as people tried the proposition for themselves. In my home, card-playing was a popular recreation, so we tried it out immediately after the show. I have been fascinated by this game ever since.

Sometime time in the mid 1990's I did a statistical study of the game, and found with a 99% confidence level that the probability of winning (with perfect play,) is indeeed a bit more than 98%. I've always wanted to know the exact probability, and this article describes how I've computed it.

2 The Rules

Poker is played with a 52-card deck, with four suits (Clubs, Diamonds, Hearts, and Spades) of 13 cards each. The cards in each suit have ranks

2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King ,Ace.

In what follows, Jack, Queen, King, and Ace will be abbrevaited as J, Q, K, A, respectively.

A poker hand contains five cards, and the hands have certain values, of which only three concern us. Five cards of consecutive ranks constitute a *straight*. While the Ace is normally the highest card of a suit, for this purpose it can also be considered the lowest card, so that 10 J Q K A and A 2 3 4 5 are straights. Five cards of the same suit constitute a *flush*. In poker a hand that is both a flush and a straight, that is five consecutive ranks of the same suit is called a *straight flush*, and is a very good hand indeed, but for our purposes it's just another flush. The third type of "pat hand" is called a *full house*, and consists of three cards of one rank and two cards of another rank. These are presumably called "pat hands" because in draw poker, a player holding one of these hands must normally "stand pat", holding all five cards, for to draw a card is to destroy the value of the hand.

In Maverick poker, the player deals 25 cards at random, and attempts to separate them into five disjoint pat hands. Of course, sometimes this is possible, but the solution is elusive, and the player may not find it. We shall assume however, that the player plays perfectly, always solving the problem if a solution exists. Alternatively, we may just ask for the probability that a solution exists.

3 Equivalence Classes

There are

$$\binom{52}{25} = 477,551,179,875,952$$

possible deals of 25 cards. The suits don't really matter. If we permute the suits in a deal, the resulting deal has a solution if and only if the original deal did. There are usually $4! = 24$ ways to permute the suits, though sometimes there are fewer. If two suits have exactly the same ranks, then interchanging those two suits will not affect the deal at all. In this case, there are only 12 equivalent deals, and if there are three identical suits, there are only four equivalent deal, since the only choice is which suit is the inequivalent one.

The plan is to generate one deal from each equivalence class, and determine whether it has a solution and how many deals it represents. Then we need only consider about one tweny-fourth of the deals. I did this by deciding that there would always be at least as many Spades as Hearts, at least as many Hearts as Diamonds, and at least as many Diamonds as Clubs. Furthermore, if two of the suits had the same number of cards, then the suit that would normally be longer will have the higher cards. In poker, hand are compared lexicographically. The highest cards in each hand are compared; the higher one belongs to the better had. If the highest cards are the same, then the second-highest cards are compared, and so on. Only in the case where we have a tie all the way down the line do we have to adjust the number ofhands represented.

This is a substantial reduction, but we can do even better. Besides the symmetry of suits, there is a symmetry of ranks. Suppose we alter a hand by replacing every 2 by the K of the same suit, and every K by the 2 of the same suit; replacing every 3 by the Q of the same suit, and every Q by the 4 of the same suit, and similarly exchanging 4 and J, 5 and 10, 6 and 9, and 7 and 8. Aces are unchanged. Clearly, the new deal has a solution if and only if the original deal has one. The transformation takes flushes to flushes, full houses to full houses, and straights to straights (because the Ace can be high or low.) Now we need only consider about half the possible Spade suits. We don't need to consider both a suit and its "mirror image." If the suit has a King but not a 2, we accept it; if it has a 2 but not a K, we reject it; if it has neitehr or both, we look to the 3 and Q to decide, and so on. Again, it is possible that there are ties all the way down the line, and then we don't get a doubling.

Unfortunately, the rank and suit symmetries conflict, because transforming the ranks by reflection may change the ordering of the suits. After relection the suit that originally had the lowest low card will now have the highest high card, and the low cards probably had no role in determining which of the original suits was better. So, for example, if the Spades and Hearts have the same length, there is no way to combine the rule that the Spades are at least as good as the Hearts with the rank symmetry. Still when all the suits have different lengths, we can use the rank symmetry. This occurs in about 30% of the deals, so we get a very worthwhile reduction. Suit symmetries alone produced about 19.9 trillion equivalence classes. Applying the rank symmetry reduces this number to 17,023,704,173,138. Still a formidable number, but 460 trillion less than we started with.

4 Exhaustive Search

In mathematical terms, this is a "set exact cover" problem. We are given a set (the 25 cards of the deal,) and a family of subsets of the set (all the pat hands that can be constructed from those cards). The problem is to find a subfamily that will exactly "cover" the set, that is such that each element of the set belongs to exacly one member of the subfamily. In Maverick solitaire, we must select five pat hands so that each card occurs in exactly one of the pat hands.

Problem like this are usually solve by "backtracking." Pick one of the cards, and make a list of all the pat nahs that include it. Choose one of them, and then choose a card that is not covered by that hand. Now make a list of all the pat hands that include the second card, but no card in the first hand. Choose one of of these, and then choose another card that hasn't be covered yet, and make a list of all the pat hands that cover this third card, and so on. If we find five hands, we have a solution, but it's likely that at some point in the process, we will find it impossible to proceed. Then we back up and try again. For example, if there is no pat hand available to cover the third card, we back up and try the the next hand on the list we created for the second card. If that list been exhausted, we back up again, and try the next hand on the first list we made. If all the hands on that list have been tried, there is no solution; every hand that covers the first card fails.

This can be a time-comsuming process. We might hit on the solution on the very first try, or we might keep making unfortuante choices, and take a long time to hit on the answer. It is a proverb in computer science that one should keep one's search trees "narrow at the top," that is, one should attempt to fulfill the most stringent conditions earliest. In this case, it means that we should first try to cover the card that is hardest to cover, the one that occurs in the fewest pat hands. Once we've covered the first card with some hand, all pat hands contatining any card in that hand are eliminated, so the second card we try to cover should be the uncovered card that occurs in the fewest of the eligible pat hands that remain.

The famous computer scientist Donald Knuth has developed an efficient algorithm along these lines called DLX and described his paper "Dancing Links", and that is the method I've used. Efficient though this method is, it can still take an aprecialble time. Before we can run the algorithm, we have to find all the pathand in the deal. Since there can be 2,000 or more pat hands in a deal, this takes a while.

5 Avoiding Exhaustive Search

It would be better to avoid doing the exhaustive search at all. At first it seems that we can never be sure there is no solution without doing an exhaustive search, but this is not so. Consider the following deal:

♠AKQJ10987
♡AKQJ10987
♢AKQJ10987
♣3

This has no solution, because there is no pat hand that covers the 3 of Clubs. Of course, DLX would discover this immediately, but in the setup for DLX, we have to construct the $1,208$ pat hands in the deal, and tabulate how many pat hands each of the 25 cards appears in. The first thing the program does is to check whether some card is not contained in any pat hand. Of course, any card in suit of at least five cards is contained in a flush, so this check is only performed on cards that occur in a suit Of fewer than five cards.

It may also be possible to avoid exhaustive search by using heuristics. A heursistic is a sort of rule of thumb that oftem leads to the right answer, though there is no guarantee. In this case, we may try to mimic the approach that a human player might take to attack the problem, or we might try to find a method peculiarly suited to computer calculation. If the heuristic fails to find a solution, we must fall back to exhaustive search.

There is another way to avoid exhaustive seach that is peculiar to this problem. The deals are generated in order; we can think of it like the wheels on an odometer, with Spades on the left, then Hearts, then Diamonds, and then Clubs. The Clubs cycle fastest and the Spades slowest. Suppose some deal has a five-card Club suit, and that we finds a solution that contains a Club flush. Then until the Diamonds are incremented, every deal has a solution, because only the Clubs change. We can do exactly the same things with the Spades, Hearts, and Diamonds, and just replace the Club flush a another one. So, we don't have to generate those hands at all; we simply record the results, andchange the setting on the odometer. The same approach applies to hands with exactly five Diamonds and no Clubs, but there are many fewer of these.

6 The Program

Here I describe the overall structure of the program, and the data structures used.

We can classify the structure of the various deals like rhyme schemes. *abcd* means that there are four suits of different lengths; *abb* means that there are three suits, with the Hearts and Diamonds having the same length and the Spades a different length. There are ten lengths:

abcd

abc

ab

aabc

abbc

abcc

abb

aab

aaab

abbb