HOW MANY GAMES OF SPIDER SOLITAIRE ARE WINNABLE? EXPLORATIONS INTO THE MATHEMATICS UNDERLYING SPIDER SOLITAIRE

by

Mark S. Weisser

Submitted to Wesleyan University in partial fulfillment of the requirements for the degree of Master of Arts in Liberal Studies.

April 2012

Adam Fieldsteel
Wesleyan University
Professor of Mathematics

1. Abstract

The game of Spider Solitaire is analyzed to determine the number of the 104! initial arrangements of cards that have a winning solution. The game is broken down into its essential characteristics. Five parameters describing a Spider Solitaire game card pack and board configuration are formulated allowing variants of the game, notably smaller ones, to be analyzed. A computer program is developed and used to analyze and simulate the play of the various game variants. A class of Spider solitaire games using an infinite number of cards is then examined and used as a basis for constructing a graph of game positions traversed by a random walk. A suggestion for a new type of Spider game based on the graph construction is presented. A definitive answer is not reached. Areas for continued explorations are suggested.

2. Introduction

The richness of mathematics underlying many games has a long history of study. The inspiration for this topic, the analysis of Spider Solitaire, originated when the author viewed a webcast of a lecture given by Persi Diaconis (Diaconis, The Mathematics of Solitaire, 1999). Though the mathematics discussed quickly went beyond the abilities of this viewer the enthusiasm shown for the topic was contagious.

A subsequent search uncovered a website depicting the results of 30,000 random games of Spider Solitaire played by a computer program (Alex at Tranzoa Company, 2005). This is the most thorough data that the author has discovered to date. The results of that analysis, that most games are winnable, is in line with the author's experience in the actual playing of the game. Given enough time and persistence, and more importantly an "undo" or back tracking key, one can eventually find a winning solution to a Spider Solitaire game.

This paper will focus mostly on the decomposition of the game into a format amenable to computer analysis and simulation. But there will be a blend of some of the underlying mathematics found in the game, such as enumerative combinatorics, as well. Data from the computer runs will be collected, summarized and presented. Conjectures based on the data will be offered. Along the way a few detours will be taken as interesting topics arise. What is presented here is a description of a journey of exploration that leads to more questions than answers.

The exploration will proceed as follows. First, in section 3 we provide a description of the standard Spider Solitaire game. The description presented will provide a definition of game characteristics and a decomposition of aspects of the game that will allow us to define game variants that are smaller and more amenable to analysis. In section 4 we define a list of 5 parameters that will be used to describe finite game variants. We also explore some of the smaller game variants. All of the groundwork laid out in sections 3 and 4 leads to the program analysis in section 5. A computer program is described that simulates the play of both the standard Spider Solitaire game and smaller variants. Data is collected from these simulation runs that leads to the formation of conjectures regarding the effect of each of the 5 game parameters on the number of winnable games. This leads to the design of additional experiments. Section 6 introduces the notion of a Spider game variant that uses a potentially infinite number of cards. Here we pose the question as to whether all such infinite games are winnable. A suggestion is presented that outlines how an infinite game variant may be embodied in a form that makes the game playable by a human participant in section 7. Finally, section 8 ties everything that we have learned along the way back to the original Spider Solitaire game and the question posed by this thesis. Further explorations are listed that might lead to a definitive answer.

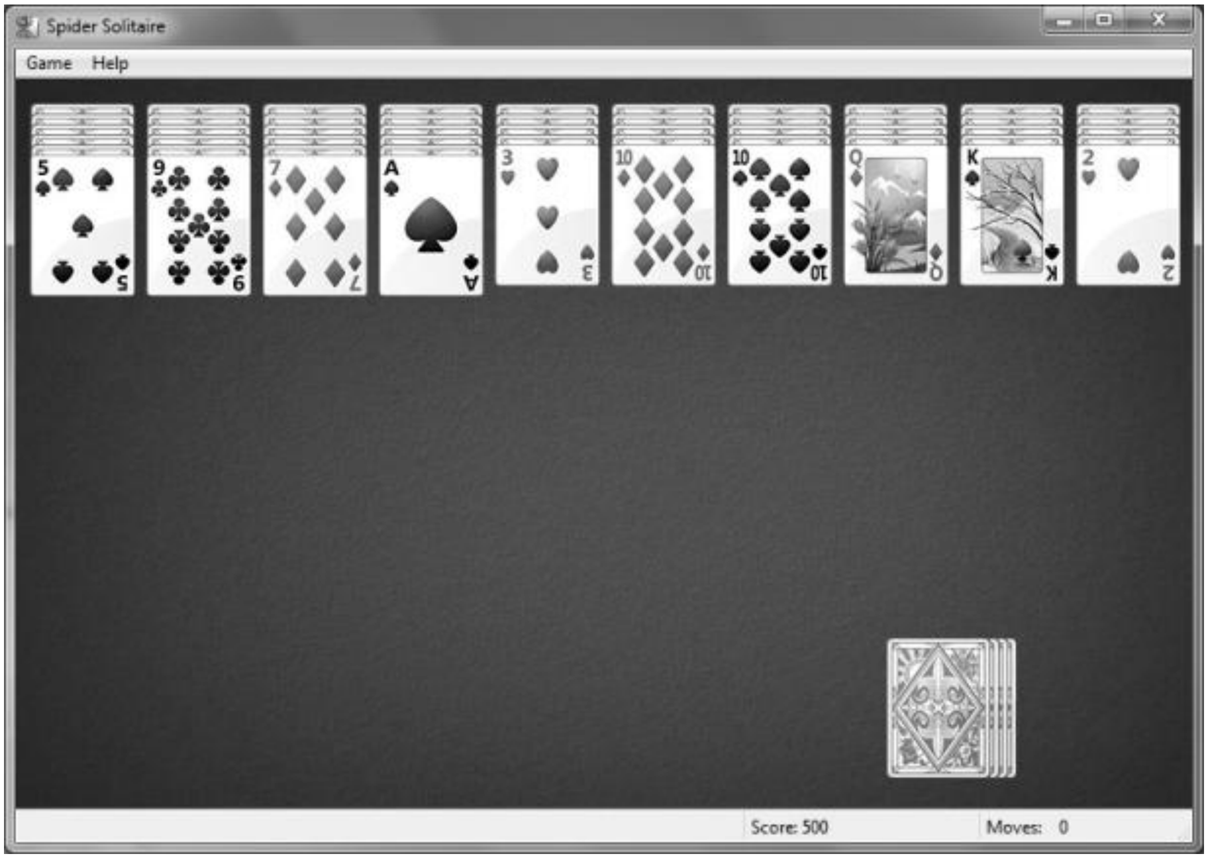
3. The Game of Spider Solitaire

In this section we describe the standard game of Spider Solitaire. Our purpose is to identify the salient elements of the game that will be the underpinnings of the sections that follow.

3.1 Description of the Standard Spider Solitaire Game

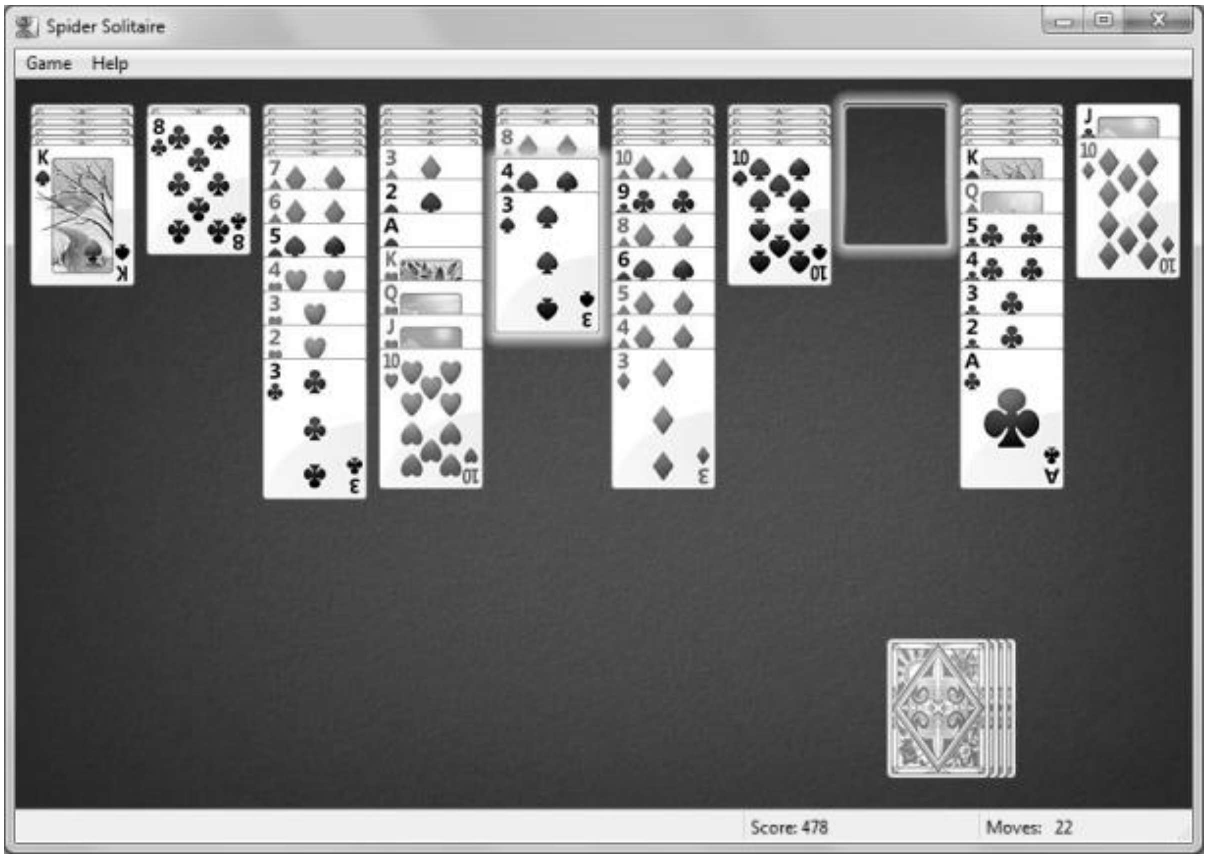
Spider solitaire is a popular game freely available on both the internet (AARP.org, 2012) and many personal computers (Microsoft Corporation). The game is played with 2 standard decks of 52 playing cards consisting of 13 card ranks of 4 suits. The standard Spider Solitaire game is considered by many to be the most challenging of the many solitaire variants.

Throughout this analysis reference will be made to the "standard" game. This is the game that most computer programs such as Microsoft's (Microsoft Corporation) present to a player. The game starts with an initial arrangement of the cards and proceeds through a series of allowable moves until the final desired end state is achieved or no further moves can be made. Cards are rearranged into strictly consecutive descending sequences by rank. Cards of the same suit in a consecutive descending sequence can be moved as a block. When a sequence of 13 cards of the same suit is formed the block of 13 cards is removed from the game. A game is won when all of the cards are removed.



A new game instance of Microsoft's Spider Solitaire

The standard game takes an initial random arrangement of the pack of 104 cards and maps them onto a game board that consists of face down hidden cards, face up active cards, and a stock of cards held in reserve for subsequent deals. At the start of the game the hidden cards are placed on the board in 10 piles arranged in columns. 6 piles have 4 cards and 4 piles have 5 cards to account for the extra 4 cards ($104 \ mod \ 10$). One row of face up cards is placed on top of the 10 piles of hidden cards. These face up cards represent the cards that are initially active, that is, can be moved in accordance with the rules of the game. The remaining 50 cards are held in reserve to be dealt 10 at a time one card per column at any time the player deems appropriate or is required due to lack of any alternative move.

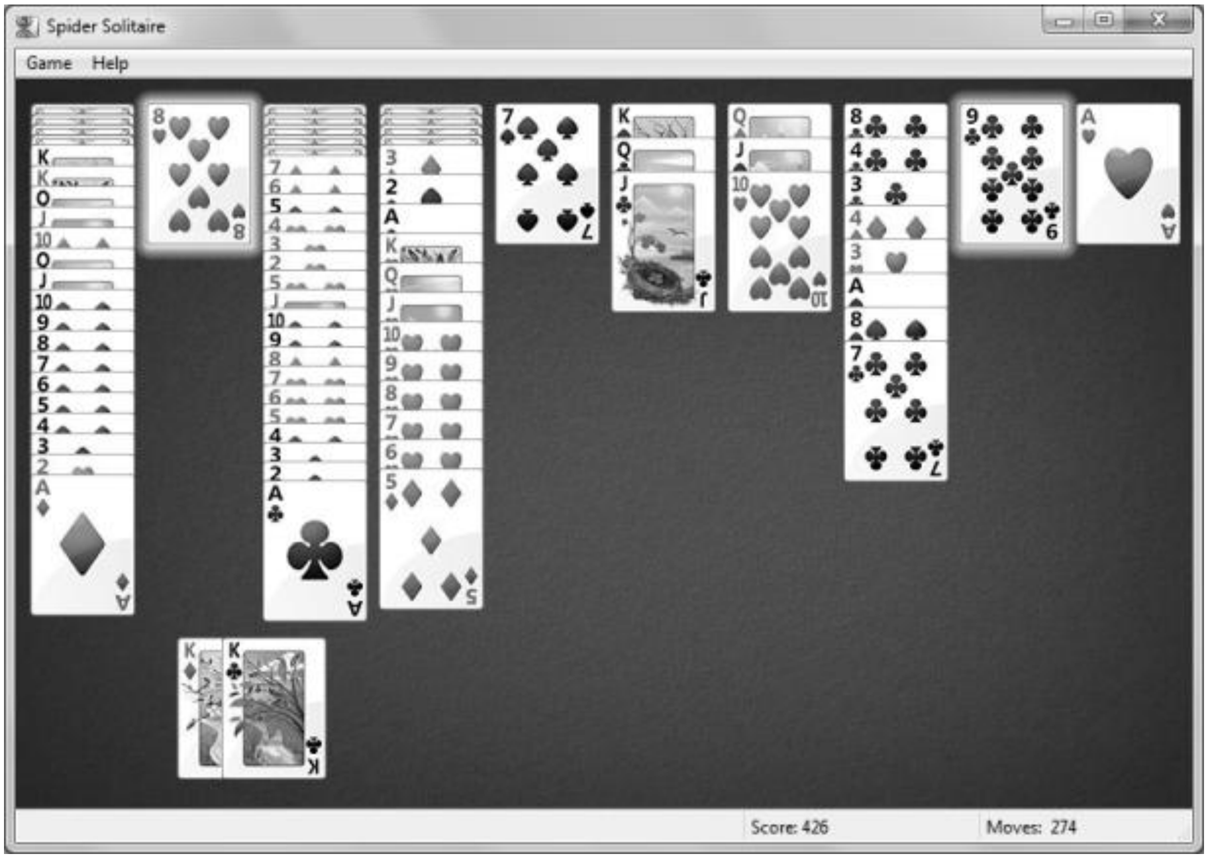


A Spider Solitaire game in progress

3.2 Definition of a Winnable Game

In the normal play of a game by a human participant the game is won by clearing all of the cards from the board. The game is lost if the player's moves lead to a board position where no further moves are possible, no subsequent cards remain in the stock for dealing, and cards remain on the board. But our definition of a winnable game is based solely on the possibility of winning. A player's ability to find the winning solution bears no consequence.

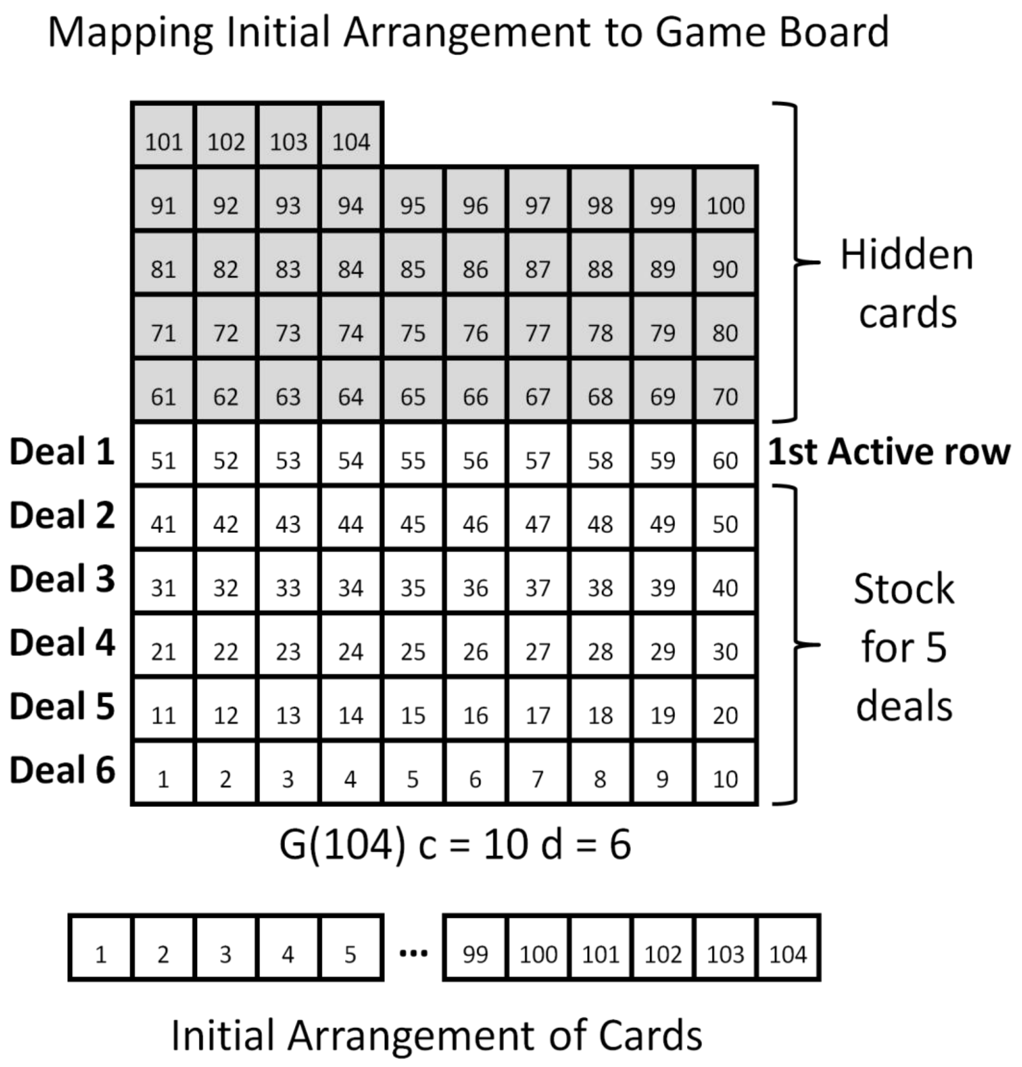
For our purposes if a sequence of moves exists, irrespective of whether a player can find that sequence of moves, the game is deemed winnable. If there is no possible sequence of permissible moves that leads to all of the cards being removed the game is deemed unwinnable. Thus, the 104! initial arrangements of the cards fall into two disjoint sets. The winnable set contains the arrangements that can be changed via a sequence of allowable moves into the desired end state with all cards removed. The unwinnable set contains those initial arrangements that contain a pattern of cards that prevent the initial arrangement from being transformed by the rules of the game into the final winning state.



A game with 2 full blocks removed en route to a win

3.3 Mapping the Initial Card Arrangement to the Game Board

How do we take an initial arrangement of cards and map that arrangement to the game board? There are many ways to map this initial "shuffle" of the cards onto the game board. Applying a consistent and well defined methodology allows us to share instances of a particular game by simply providing the arrangement of cards in either the list form, or preferably, in the abstract board arrangement we depict next. We rely on this mapping for our computer analysis that comes later. The mapping must also accommodate games that we wish to analyze that may have a differing number of cards from the standard game's 104.



Opposite to how a player might mechanically place the cards on the initial board the mapping used in this analysis takes the first 10 cards of an initial arrangement of 104 cards and assigns those 10 to the last deal in sequence, the next 10 to the next to last deal, and so on. The final 4 hidden cards are placed at the start of the topmost row of hidden cards so that the columns with 5 hidden cards appear before the columns with 4 hidden cards. The mapping has advantages when we begin to explore game variants in which the number of columns is varied. The mapping is depicted above. The notation, the details of which follow in the next section, $G(104), c = 10, d = 6$ represents a game of 104 cards arranged onto a board of 10 columns with 6 deals, 1 initial deal and 5 subsequent deals available to the player.

3.4 Unique Standard Spider Solitaire Games

We now take a brief detour to answer a question about the $104!$ possible initial arrangements of cards. How many of these initial arrangement present a unique game challenge? Are there symmetries that make some initial arrangements play identically to others?

The number of possible games of the standard Spider Solitaire game is $104! \cong 1.02290 \cdot 10^{166}$. The number of unique games presented to a player is somewhat less due to symmetries in the arrangements. First, since 2 decks of identical cards are used, there is no way to distinguish a card with a particular rank and suit in one deck from the same card in the other deck. There are 52 such cards that can be arranged in 2! ways that can be factored out of the count of the initial arrangements. The standard game has 10 columns of cards 6 of which have 10 cards and 4 of which have 11 cards. The order of the columns does not affect the play of the game, thus there are 6! arrangements of the columns with 10 cards and 4!

arrangements of columns with 11 cards that can be factored out as well. Finally, the suits themselves do not matter. For example, all of the diamonds could be switched with all of the hearts of equivalent rank without changing the game. There are 4! arrangements of the suits. These three sets of symmetries overlap. To get an exact count the principle of inclusion-exclusion would need to be applied, first subtracting the intersection of each of the pairs of the three sets and then adding back the intersection of all three. For now, we arrive at an approximate answer using the equation:

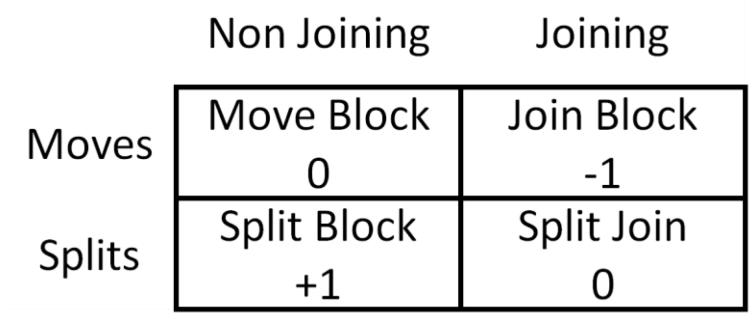
$$\frac{104!}{2!^{52}\,4!\,6!\,4!} \cong 5.51418 \cdot 10^{144}$$

3.5 Standard Rules of Play

What are the permissible rules of movement in the standard game of Spider Solitaire? There are only a few simple rules that are used in the standard game. The initial board as described above is presented. One row of 10 face up cards is shown. A card can be moved onto another card if its rank is one lower forming a consecutive descending sequence. No movement is possible onto the lowest ranking card. A block of cards of the same suit in a consecutive descending sequence may behave as if a single card. Such a block of cards need not move as a unit and can be split at any point within its sequence. Only the bottom card or block can be moved onto the bottom card of another column. When a hidden card is exposed, it is turned face up and becomes active, that is, available for movement. Should a column of cards have all of its hidden cards exposed and the final card moved to another column, the column becomes empty. Any bottom card or block can be moved to an empty column. This is the only way that the highest ranked card can move to another column. At any time, with one exception, when cards remain in the stock a deal from these cards can be made. This deal places one card on each of the 10 columns. A deal from the stock, however, cannot be made if there is an empty column on the board. Thus, a deal will always be made onto active, face up cards in which serendipity may form or add to an existing block. The stock initially holds 50 cards allowing for 5 such deals. Whenever movement results in the formation of a full block, the cards from king descending to ace in the same suit, the full block of 13 cards is removed from the game. This removal is implicit and occurs immediately. The consequence of this game mechanic is that an unwinnable game may occur because it is not possible to fill all columns in order to make a remaining deal. The game is won when all of the cards have been formed into full blocks and removed.

3.6 Classification of the Types of Moves

For the purpose of this analysis and that of the computer program developed to simulate play, the legal moves of Spider Solitaire are further refined. We classify the moves into 5 types. The deal of a new row of cards from the stock is one type. Moving a card or block of cards of the same suit from one column to another make up the remaining 4 types. Here we will think of a single card as a block of cards of the same suit having size 1. Thus, we can speak in terms of moving blocks of cards. So the first of the 4 types of moves is moving a full block of cards from a source column to another target column which is either empty or whose bottom block is of a different suit. The second type joins a full block to a column whose bottom block is the same suit as the block being moved forming a larger block in the target column. The next two types involve splitting a block from the source column and so only apply to blocks of size greater than 1. Blocks are not required to move as a unit and can be split at any point. The third type of move splits off a part of the source block and either moves the lower portion split to an empty column or to a column having a block of a different suit. The fourth and final move type splits off a portion of the source block and moves that to a column having a block of the same suit. The moves types are summarized in the diagram below.



	Non Joining	Joining
Moves	Move Block 0	Join Block -1
Splits	Split Block +1	Split Join 0

Move types and their effect on join operations required to win.

3.7 Join Operations

The joining of blocks which we introduced above is core to the winning of a game. We take a few moments here to explore these join operations. The numbers, $\{-1,0,1\}$, associated with each of the 4 move types represent the effect of the move type on the total number of block joining operations required to arrive at a winning game. Analogous to assembling the pieces of a jigsaw puzzle (Briggs, 2005), or conversely, breaking a chocolate bar into pieces (Winkler, 2010), the number of joining or splitting operations required respectively is equal to 1 less than the total number of pieces involved. A 500 piece jigsaw puzzle requires 499 pieces to be joined together irrespective of the order or clumping of the assembly process. Similarly, to win a game of Spider Solitaire one must assemble full blocks of cards so that the blocks can be removed from the game. Assembling a full block of 13 cards requires 12 join operations.

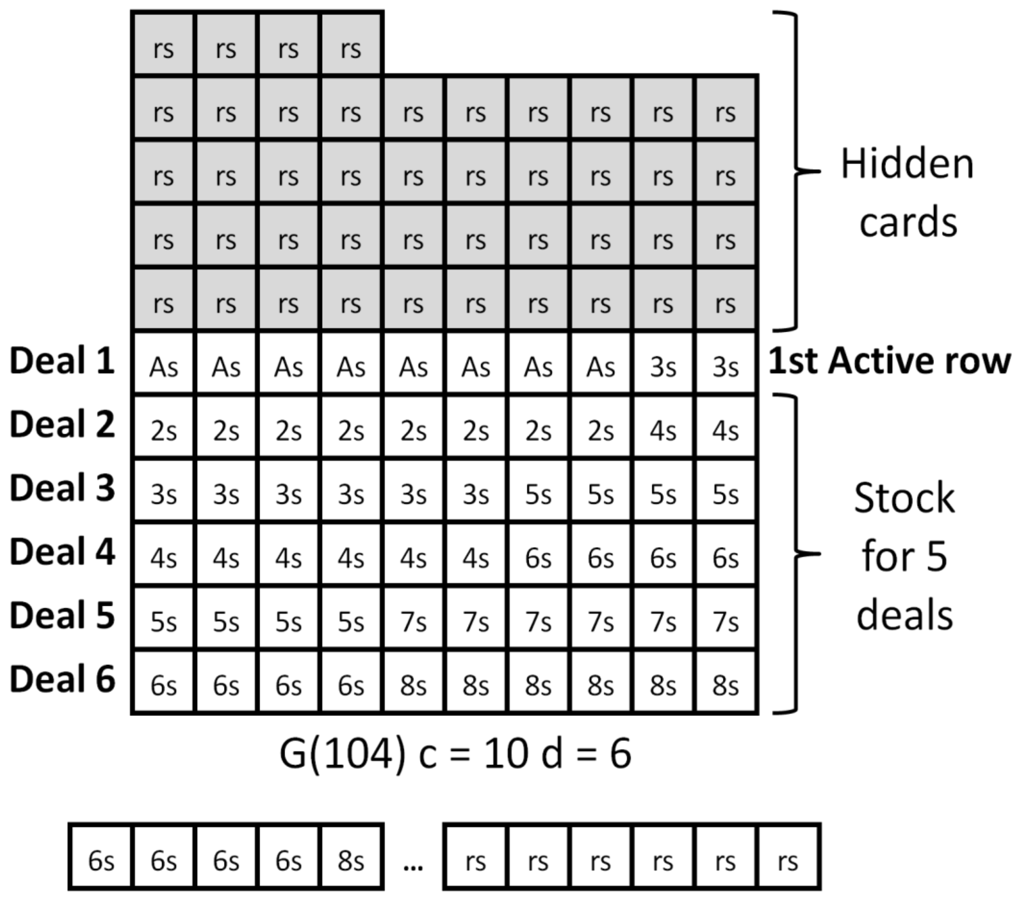
A game of Spider Solitaire is won when there are no blocks left to assemble, that is, when 0 join operations are left remaining. The numbers associated with each move type reflect that joining 2 blocks reduces by one the number of join operations required, splitting a block increases the number by 1 while the remaining two types leave the number of join operations unchanged. Note that in addition to the "Join Block" move type a deal may through serendipity form or extend blocks on the board reducing the number of explicit "Join Block" moves that must be performed to win. There can be from 0 to 10, the number of columns in the game, of these serendipitous joins that occur with each deal.

3.8 Many Unwinnable Games of Standard Spider Solitaire Exist

Our quest is an attempt to determine how many games of standard Spider Solitaire are winnable. We can show that the answer is less than 100% of all games by demonstrating particular games that cannot be won. We use the abstract board configuration previously defined for that purpose.

The following diagram depicts an arrangement of the cards demonstrating unwinnable games in the standard Spider Solitaire game. With this arrangement there is no movement possible other than the dealing of the cards from the stock. In the diagram r represents any card rank and s represents any card suit. Suits have no bearing in the position depicted where no movement is possible after each deal and no deal forms blocks with the previous deal. Thus, the arrangement depicts unwinnable games in the hard, medium, and easy variants of the standard games. We will be addressing these variants shortly.

Unwinnable Game – No Movement Possible



	rs	rs	rs	rs						
	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
Deal 1	As	As	As	As	As	As	As	As	3s	3s
Deal 2	2s	2s	2s	2s	2s	2s	2s	2s	4s	4s
Deal 3	3s	3s	3s	3s	3s	3s	5s	5s	5s	5s
Deal 4	4s	4s	4s	4s	4s	4s	6s	6s	6s	6s
Deal 5	5s	5s	5s	5s	7s	7s	7s	7s	7s	7s
Deal 6	6s	6s	6s	6s	8s	8s	8s	8s	8s	8s

G(104) c = 10 d = 6

Initial Arrangement of Cards

All of the hidden cards in the preceding example can be arranged in any way so that there are $44! \cong 2.65877 \cdot 10^{54}$ unwinnable games represented in this single diagram. Clearly, in the hard, medium, and easy standard variants unwinnable games exist.

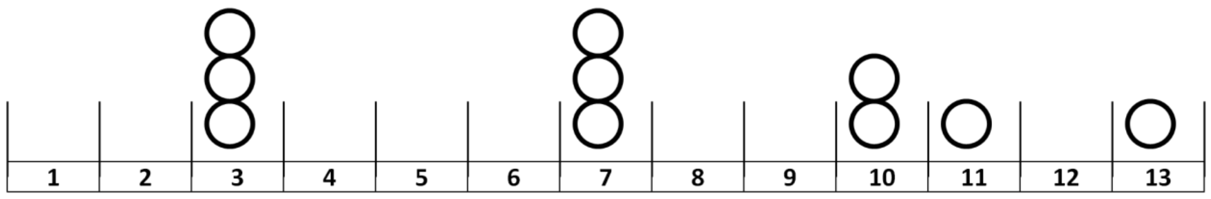
3.9 104 Choose 10

Before continuing with our decomposition of the standard game we take a detour to address a question suggested by the depiction of the unwinnable games above. What is the probability in the standard game that the initial deal, or any deal if we discount the possibility of forming or extending blocks, contains no cards that can be moved? Recall that for card movement to occur there must be a pair of cards having consecutive ranks among the 10 cards dealt. We have $\binom{104}{10} \cong 2.6101 \cdot 10^{13}$. How many of these choices have no movement possible, that is, have no pair of cards with adjacent ranks? Conversely, how many of these choices have at least one pair of cards with adjacent ranks?

We will be taking an indirect approach to reaching our final solution. Here is a brief outline of the roadmap. We define a process. First, we will enumerate the integer partitions of 10 to identify the possible distributions of card ranks. Next we will apply a multinomial coefficient to the distinct parts of the partition to get all permutations of the distribution. Then we will use a binomial coefficient to choose each part from the pack of cards. We formulate a counting function to enumerate the combinations with at least one adjacent pair. Finally, we apply the ratio of adjacent combinations to all the total combinations for each partition. This circuitous route to reach our answer allows the formulation of the problem in a way that a spreadsheet can be used to perform the necessary calculations. The method used calculates along the way the distributions of the ranks in a 10 card deal. As a byproduct we will be able to answer questions such as what is the most likely distribution of ranks in the 10 cards chosen? What is the probability of all 10 cards having different ranks? We can also apply the same process to the distributions of suits or even specific cards and ask questions about those. What is the most common distribution of suits? How often can we expect to see the exact same card appear twice in the 10 cards chosen?

As outlined our approach starts by enumerating the integer partitions of 10. To address the initial question of adjacent pairs we need only consider the rank of each card. If we take the rank of kings as 13, queens as 12, jacks as 11, and aces as 1, each card can have a value taken from the set [13], the integers 1 through 13 inclusive. Each of the 13 values is present 8 times in the 104 cards. In the 10 chosen cards each value may appear 0 to 8 times as long as the total number selected sums to 10. Thus, there might be 3 threes, 3, sevens, 2 tens, 1 eleven and 1 thirteen. (3 + 3 + 2 + 1 + 1 = 10). This is the integer partition of 10 with parts less than or equal to 8, $p(10|\, parts \leq 8)$ (Andrews & Eriksson, 2004). There are 42 integer partitions of 10, p(10) = 42, 40 of which have parts less than or equal to 8. Only the partitions 10, and 9+1, have a part, the 10 and 9 respectively, greater than 8. We can view each partition of 10 as a distribution of the characteristic of the cards under scrutiny. Here we are concerned with the card ranks.

Instead of thinking of selecting 10 cards from a set of predefined values, let's think about taking 10 indistinguishable items and dropping each one at random into one of 13 labeled bins. All 10 can land in one bin, one each of the ten can land in a different bin, or any other arrangement in between. If we label the bins 1 through 13 to represent the value of the card ranks we can now count the number of items in each bin. The figure below illustrates the example of the partition 3 + 3 + 2 + 1 + 1.



Assuming for the moment that each bin can hold at least 10 items rather than the 8 of each available in the pack of 104 cards, the number of different ways that 10 indistinguishable items can be placed into 13 bins, applying box 4 of the twelvefold way (Stanley, 1986, p. 33), is given by $\left(\binom{13}{10}\right)$, 13 multichoose 10, which equals $\binom{13+10-1}{10} = \binom{22}{10} = 646{,}646$. Of these arrangements how many have at least one pair of adjacent bins with at least one ball in each bin? The example above is one such arrangement as the adjacent bins 10 and 11 both contain at least one ball.

Each partition represents a possible distribution of ranks. There are two features of each partition that we employ, the number of parts in the partition and the number of distinct parts. Again, using our example of $3+3+2+1+1$, we see that there are 5 parts, the 5 terms in the sum, and 3 distinct parts, two 3's, one 2, and two 1's. Partitions are commonly denoted using exponents in the form $3^2 2^1 1^2$ which shows the 3 distinct parts whose exponents sum to the number of parts.

For each partition we want to determine how many ways that the partition can occur in our $\left(\binom{13}{10}\right)$ ways. What we really need is the composition of 10 rather than the partition because the order matters when we assign each distinct part to one of the 13 bins representing the ranks. We choose 2 bins from the 13 to hold the two groups of 3, then 1 bin from the remaining 11 bins to hold the group of 2, and then 2 bins from the remaining 10 to contain the single items. This is the multinomial coefficient $\binom{13}{2,1,2}$. Stanley states that "the multinomial coefficient can also be interpreted in terms of "permutations of a multiset"" (Stanley, 1986) which is our application here. Note that the lower term of the multinomial is simply the exponents from the partition notation. We apply the multinomial using parts and distinct parts to each of the 42 partitions which sum to $\left(\binom{13}{10}\right) = 646646$.

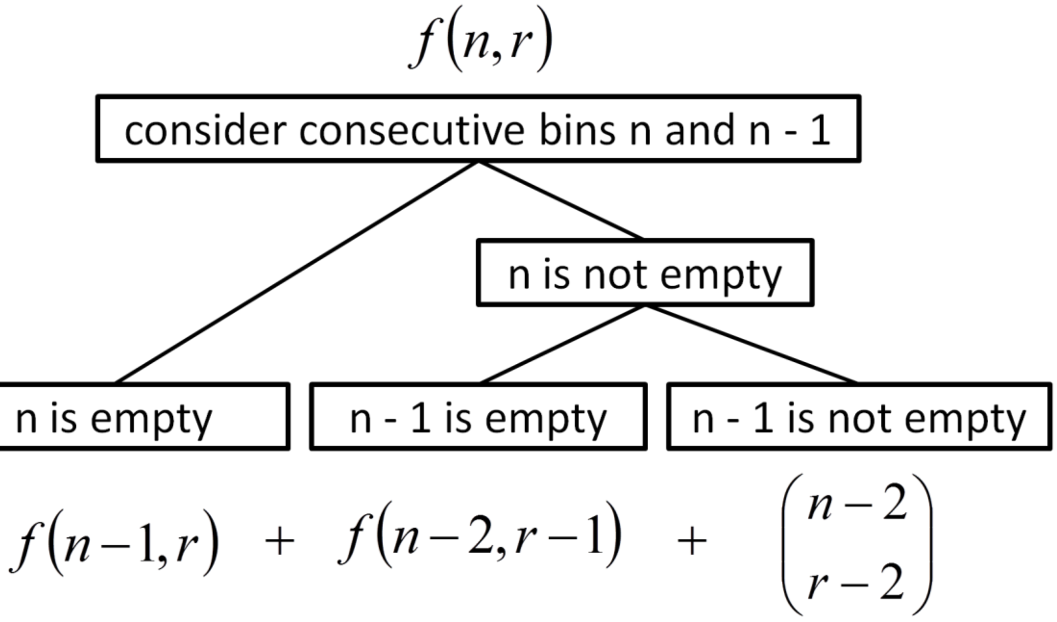
Partitions of 10										distinct				r	13	8	choose each part									104	Freq	c(n,r)	f(n,r)	Adjacent
10										1				1	13	0	1	1	1	1	1	1	1	1	1	0	0.00000	13	0	0
9	1									1	1			2	156	0	8	1	1	1	1	1	1	1	1	0	0.00000	78	12	0
8	2									1	1			2	156	1	28	1	1	1	1	1	1	1	1	4368	0.00000	78	12	672
8	1	1								1	2			3	858	1	8	8	1	1	1	1	1	1	1	54912	0.00000	286	121	23232
7	3									1	1			2	156	8	56	1	1	1	1	1	1	1	1	69888	0.00000	78	12	10752
7	2	1								1	1	1		3	1716	8	28	8	1	1	1	1	1	1	1	3075072	0.00000	286	121	1300992
7	1	1	1							1	3			4	2860	8	8	8	8	1	1	1	1	1	1	11714560	0.00000	715	505	8273920
6	4									1	1			2	156	28	70	1	1	1	1	1	1	1	1	305760	0.00000	78	12	47040
6	3	1								1	1	1		3	1716	28	56	8	1	1	1	1	1	1	1	21525504	0.00000	286	121	9106944
6	2	2								1	2			3	858	28	28	28	1	1	1	1	1	1	1	18834816	0.00000	286	121	7968576
6	2	1	1							1	1	2		4	8580	28	28	8	8	1	1	1	1	1	1	430510080	0.00002	715	505	304066560
6	1	1	1	1						1	4			5	6435	28	8	8	8	8	1	1	1	1	1	738017280	0.00003	1287	1161	665763840
5	5									2				2	78	56	56	1	1	1	1	1	1	1	1	244608	0.00000	78	12	37632
5	4	1								1	1	1		3	1716	56	70	8	1	1	1	1	1	1	1	53813760	0.00000	286	121	22767360
5	3	2								1	1	1		3	1716	56	56	28	1	1	1	1	1	1	1	150678528	0.00001	286	121	63748608
5	3	1	1							1	1	2		4	8580	56	56	8	8	1	1	1	1	1	1	1722040320	0.00007	715	505	1216266240
5	2	2	1							1	2	1		4	8580	56	28	28	8	1	1	1	1	1	1	3013570560	0.00012	715	505	2128465920
5	2	1	1	1						1	1	3		5	25740	56	28	8	8	8	1	1	1	1	1	20664483840	0.00079	1287	1161	18641387520
5	1	1	1	1	1					1	5			6	10296	56	8	8	8	8	8	1	1	1	1	18893242368	0.00072	1716	1688	18584961024
4	4	2								2	1			3	858	70	70	28	1	1	1	1	1	1	1	117717600	0.00000	286	121	49803600
4	4	1	1							2	2			4	4290	70	70	8	8	1	1	1	1	1	1	1345344000	0.00005	715	505	950208000
4	3	3								1	2			3	858	70	56	56	1	1	1	1	1	1	1	188348160	0.00001	286	121	79685760
4	3	2	1							1	1	1	1	4	17160	70	56	28	8	1	1	1	1	1	1	15067852800	0.00058	715	505	10642329600
4	3	1	1	1						1	1	3		5	25740	70	56	8	8	8	1	1	1	1	1	51661209600	0.00198	1287	1161	46603468800
4	2	2	2							1	3			4	2860	70	28	28	28	1	1	1	1	1	1	4394790400	0.00017	715	505	3104012800
4	2	2	1	1						1	2	2		5	38610	70	28	28	8	8	1	1	1	1	1	1.35611E+11	0.00520	1287	1161	1.22334E+11
4	2	1	1	1	1					1	1	4		6	51480	70	28	8	8	8	8	1	1	1	1	4.1329E+11	0.01583	1716	1688	4.06546E+11
4	1	1	1	1	1	1				1	6			7	12012	70	8	8	8	8	8	8	1	1	1	2.20421E+11	0.00844	1716	1715	2.20293E+11
3	3	3	1							3	1			4	2860	56	56	56	8	1	1	1	1	1	1	4018094080	0.00015	715	505	2837954560
3	3	2	2							2	2			4	4290	56	56	28	28	1	1	1	1	1	1	10547496960	0.00040	715	505	7449630720
3	3	2	1	1						2	1	2		5	38610	56	56	28	8	8	1	1	1	1	1	2.16977E+11	0.00831	1287	1161	1.95735E+11
3	3	1	1	1	1					2	4			6	25740	56	56	8	8	8	8	1	1	1	1	3.30632E+11	0.01267	1716	1688	3.25237E+11
3	2	2	2	1						1	3	1		5	25740	56	28	28	28	8	1	1	1	1	1	2.5314E+11	0.00970	1287	1161	2.28357E+11
3	2	2	1	1	1					1	2	3		6	102960	56	28	28	8	8	8	1	1	1	1	2.31442E+12	0.08867	1716	1688	2.27666E+12
3	2	1	1	1	1	1				1	1	5		7	72072	56	28	8	8	8	8	8	1	1	1	3.70308E+12	0.14187	1716	1715	3.70092E+12
3	1	1	1	1	1	1	1			1	7			8	10296	56	8	8	8	8	8	8	8	1	1	1.20917E+12	0.04633	1287	1287	1.20917E+12
2	2	2	2	2						5				5	1287	28	28	28	28	28	1	1	1	1	1	22149743616	0.00085	1287	1161	19981237248
2	2	2	2	1	1					4	2			6	25740	28	28	28	28	8	8	1	1	1	1	1.01256E+12	0.03879	1716	1688	9.96038E+11
2	2	2	1	1	1	1				3	4			7	60060	28	28	28	8	8	8	8	1	1	1	5.40032E+12	0.20690	1716	1715	5.39717E+12
2	2	1	1	1	1	1	1			2	6			8	36036	28	28	8	8	8	8	8	8	1	1	7.40615E+12	0.28375	1287	1287	7.40615E+12
2	1	1	1	1	1	1	1	1		1	8			9	6435	28	8	8	8	8	8	8	8	8	1	3.02292E+12	0.11582	715	715	3.02292E+12
1	1	1	1	1	1	1	1	1	1	10				10	286	8	8	8	8	8	8	8	8	8	8	3.0709E+11	0.01177	286	286	3.0709E+11
															646646											2.6101E+13	1.00000			25947965630512
																											at least 1 adjacent pair			0.994137359
																											no adjacent pairs			**0.005862641**

The spreadsheet above, divided into 4 columnar sections, is not meant to be a visual acuity test but rather a guide to following the steps in our unfolding process. So far we have covered the first two sections which list the partitions of 10 and count their permutations across the 13 choices for rank. We move now to the steps embodied in the third section.

We want to determine the number of times that any of the permutations of the distribution of ranks represented by the partitions of 10 occurs in the selection of 10 cards from the pack of 104. There are 8 cards of each rank. So for each part we compute the number of ways to choose the number of that part needed. Pictorially from our balls and bins diagram we need to choose the number of balls in a bin from the 8 possible cards with that bin's rank value. Using our same example there are $\binom{8}{3}\binom{8}{3}\binom{8}{1}\binom{8}{2}\binom{8}{2}$ ways to do so. The lower terms are simply the parts of the partition. Conveniently for our calculations, whenever the lower term is greater than 8 the result is 0. Thus, for the two partitions 10 and $9+1$ we have $\binom{8}{10}$ and $\binom{8}{9}\binom{8}{1}$ respectively which produce the result 0 effectively removing these impossible distributions from the counts. Also, conveniently, we have $\binom{0}{0}=1$. Multiplying the number of ways to get each distribution by the ways to select those number of ranks results in the number of combinations of the possible $\binom{104}{10}$ that have that particular distribution represented by the partition. Taking that count and dividing by the total combinations gives the frequency or probability with which that distribution occurs. This is shown in the column labeled "Freq" in the spreadsheet. Our example distribution has a frequency of occurrence of only 0.00831. The most common distribution of ranks is $2+2+1+1+1+1+1+1$ occurring with frequency 0.28375. Having a deal with each card a different rank occurs with frequency 0.01177 just over 1% of the time.

We have reached the final step. We now want to determine how many of the combinations for each partition have a least one adjacent pair of ranks. Our plan is to develop a ratio of combinations with adjacent pairs to the total combinations for each distribution. Here we need only think of bins occupied or not. This shifts us to box 5 of the twelvefold way (Stanley, 1986) and so the denominator of our desired ratio will be a combination of a set rather than a multiset. For each partition we can select the occupied bins equivalent to the ranks chosen as the number of parts of the partition selected from the 13 possible ranks. Our example has 5 parts and so the denominator for that partition and all the other partitions with 5 parts is $\binom{13}{5}$. The numerator of the ratio is the number of those combinations that have at least one adjacent pair.

Let's define a general counting function, $f(n,r)$, that counts the number of arrangements with at least one consecutive pair of the $\binom{n}{r}$ possibilities. We will apply the constraints that both n and r are integers, $n, r \geq 0$, and $r \leq n$. We also note that $f(n,r) \leq \binom{n}{r}$. First, we consider bin n. There are two cases, either bin n is empty or bin n is occupied. When bin n is empty the r occupied bins must occur among the remaining $n-1$ bins. In that case there will be $f(n-1,r)$ arrangements with consecutive bins occupied. Now we consider the case where bin n is occupied. There are two cases to consider here as well, namely that the $n-1$ bin is either empty or occupied. In the latter case, when occupied, we know that the arrangement has at least one adjacent occupied pair, namely bins n and $n-1$. We need to count all of the ways that the $r-2$ occupied bins can be selected from the remaining $n-2$ bins. There are $\binom{n-2}{r-2}$ ways to do so. Finally, we consider the case when n is occupied and $n-1$ is empty. Here there is no consecutive pair and so we count the adjacent occupied bins using $f(n-2,r-1)$ since we have examined 2 bins and found only one occupied. The complete formula to compute the number of choices with consecutive occupied bins is $f(n,r) = f(n-1,r) + f(n-2,r-1) + \binom{n-2}{r-2}$.


$f(n,r)$
consider consecutive bins n and n - 1
n is not empty
n is empty
n - 1 is empty
n - 1 is not empty
$f(n-1,r) \quad + \quad f(n-2,r-1) \quad + \quad \binom{n-2}{r-2}$

The diagram above summarizes the construction of the formula. A table of values computed from the formula is presented below combined with Pascal's triangle so that the ratio of counts with adjacent pairs to the number of possible combinations is apparent. Also, the recurrence relations used to construct the spreadsheet can be seen. Once again for our sample partition we have $\frac{f(n,r)}{\binom{n}{r}} = \frac{f(13,5)}{\binom{13}{5}} = \frac{1161}{1287}$. Going back to the previous spreadsheet the fourth section shows the two values used in the ratio for each partition and multiplies the total combinations by the ratio to get the count of combinations for each partition. The last column on the right sums the combinations with adjacent pairs and divides that by $\binom{104}{10}$, the total number of combinations, to arrive at our answer. A deal of 10 cards from the pack of 104 has a probability of approximately 0.00586 of having no movement possible. Conversely, at least one move will be available with probability 0.99414. We can examine initial arrangements generated during our game simulation runs described in section 5 to see if this expectation is met.

f(n,r)		r occupied bins											
		0	1	2	3	4	5	6	7	8	9	10	11
n numbered bins	0	0											
		1											
	1	0	0										
		1	1										
	2	0	0	1									
		1	2	1									
	3	0	0	2	1								
		1	3	3	1								
	4	0	0	3	4	1							
		1	4	6	4	1							
	5	0	0	4	9	5	1						
		1	5	10	10	5	1						
	6	0	0	5	16	15	6	1					
		1	6	15	20	15	6	1					
	7	0	0	6	25	34	21	7	1				
		1	7	21	35	35	21	7	1				
	8	0	0	7	36	65	56	28	8	1			
		1	8	28	56	70	56	28	8	1			
	9	0	0	8	49	111	125	84	36	9	1		
		1	9	36	84	126	126	84	36	9	1		
	10	0	0	9	64	175	246	210	120	45	10	1	
		1	10	45	120	210	252	210	120	45	10	1	
	11	0	0	10	81	260	441	461	330	165	55	11	1
		1	11	55	165	330	462	462	330	165	55	11	1
	12	0	0	11	100	369	736	917	792	495	220	66	12
		1	12	66	220	495	792	924	792	495	220	66	12
	13	0	0	12	121	505	1161	1688	1715	1287	715	286	78
		1	13	78	286	715	1287	1716	1716	1287	715	286	78
	14	0	0	13	144	671	1750	2919	3424	3003	2002	1001	364
		1	14	91	364	1001	2002	3003	3432	3003	2002	1001	364
	at least one occupied pair of adjacent bins												
	n choose r (Pascal's triangle)												

We see the construction of the counting function $f(n,r)$ in the table above. For example, the value for $f(13{,}5) = f(12{,}5) + \; f(11{,}4) + \; \binom{11}{3} = 736 + 260 + 165 = 1161$.

4. The Universe of Spider Solitaire Game Variants

Now that we have a base understanding of the constituent elements of the standard game we can further refine those elements to define game variants. Examining a smaller more accessible version of a Spider Solitaire game will provide some insight into a strategy for finding a solution to our problem and make our computer analysis that follows simpler. One such question is what constitutes a reasonable smaller version of the game? Which elements of the game influence the number of winnable verses unwinnable games? We again examine the features of the game to see which characteristics of the game we can generalize while still maintaining the playing flavor of Spider Solitaire.

4.1 The Essential Elements of a Spider Game Variant

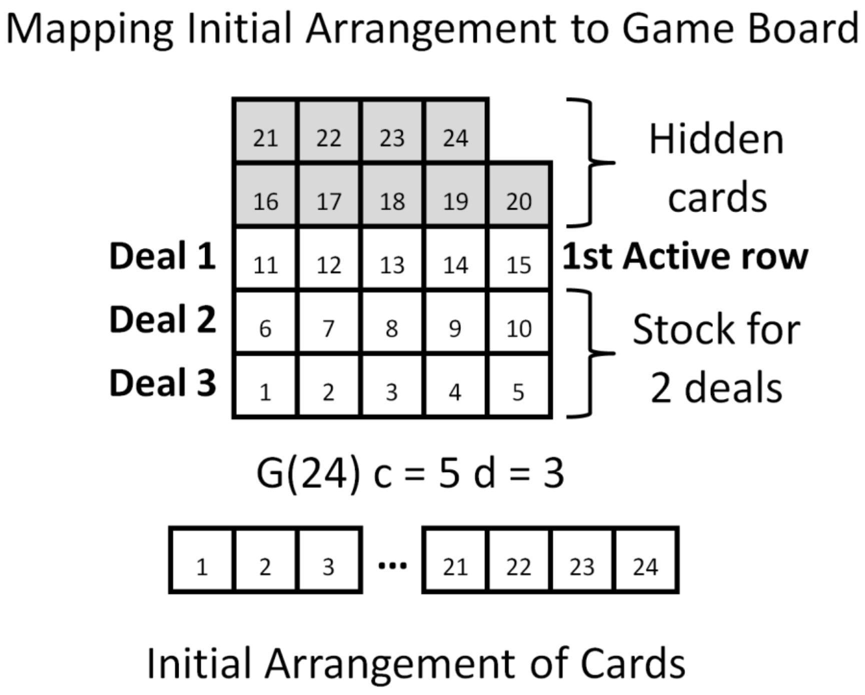
A game consists of a pack of cards in an initial configuration and rules of movement for those cards including the definition of a win. The essence of a Spider Solitaire game is contained in the rules as defined in the standard version though variations of the rules are possible. For example, one rule states that a deal of the cards from the stock of pending cards cannot occur while there is an empty column on the board. We can envision a Spider game variant in which this rule does not apply. Also, when a full block of cards is formed the cards are implicitly and immediately removed from the board. Another variant of Spider might require explicit removal of a full block via a "remove" game move. For our purposes, unless otherwise stated, we will use the rules as previously defined for the standard game. We will call this the "standard" set of rules. For the remainder of this analysis the rules of movement expressed by the 4 movement types previously defined will remain unchanged. After exploring some finite Spider Solitaire game variants, we will modify the deal rule to permit an unlimited number of deals introducing new cards as necessary. We will call these the infinite game variants. But we are getting ahead of ourselves.

4.2 Ranks, Suits, Multiples, Columns, and Deals

The ranks, suits, and multiples of cards used in the game are fundamental to the analysis that follows and so we describe them here in greater detail in terms of the standard game. The standard game consists of a deck of 52 playing cards. Each card has a rank, 1 of the 13 face values ace through king, and belongs to one of 4 suits, clubs, diamonds, hearts, or spades. Thus, an individual card has 3 attributes, a rank, a suit, and membership in a deck. We generalize these attributes to ranks, suits, and multiples. Ranks are taken from the sets of integers of the form $[n]$. Suits will be taken from any finite set of elements, for example, $\{C, D, H, S\}$ or $\{A, B, C\}$. The symbols used as elements of the set are arbitrary. The number of elements in the set is what matters. An integer will represent the multiple of decks used. This will allow the analysis to consider Spider Solitaire game variants with differing numbers of cards by varying ranks, suits, and decks. Thus, we have three integers that describe the pack of cards used in a Spider Solitaire game, the number of ranks r, the number of suits s, and the multiple of decks, m. The total number of cards n used in the game is the product of the three, $n = rsm$.

The initial board configuration of the cards consists of columns of cards divided into three sections. There is an area of hidden cards, an initial row of active cards, and a number of cards reserved for card deals. The cards are laid out in a nearly rectangular arrangement. We wish to explore variants of the standard game. The standard board is designed to accommodate 104 cards. What are the characteristics of the board that are desirable to maintain while the number of cards varies and which should remain constant? The initial board partitions the cards into two sets, the hidden cards and the cards that are dealt. The cards dealt and the hidden cards are arranged in multiples of the column size with the excess applied to the hidden cards forming the only partial row. Thus, the board configuration can be described by three numbers, the total number of cards n, the number of columns c, and the number of deals d. The following simple equations hold. Given integers $c, d, n > 0$ the number of cards reserved for dealing is cd, the number of hidden cards is $n - cd, cd \leq n$. The number of cards in the hidden partial row is $n \; mod \; c$.

Thus, for the "standard" spider game there are 104 cards divided into 60 cards dealt and 44 cards initially hidden. The first deal of ten cards is face up and active at the start of the game. A variant of this arrangement might be to add another column so that we have 6 deals of 11 cards and 38 hidden cards. Now we are in a position to ask questions such as, does increasing the number of columns also increase the number of winning games? We can also explore games with smaller numbers of cards such as a game of 12 cards with 3 columns and 2 deals giving 6 hidden cards. Note that with the equations above there can be arrangements that result in no hidden cards which is acceptable. There must always be at least one deal of cards, the initial active row. An example initial game instance with $n = 24, c = 5, d = 3$ follows.



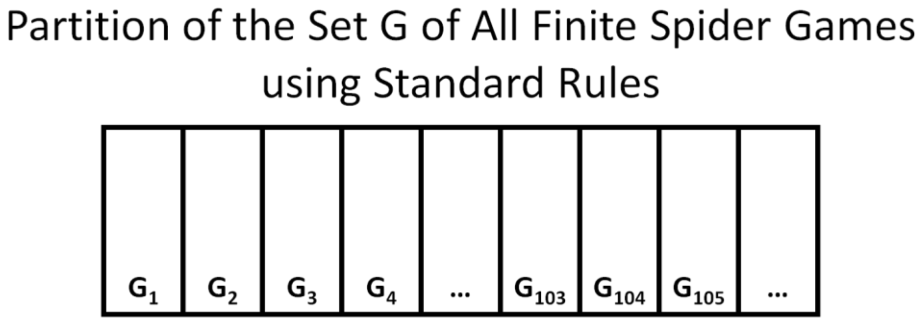
Putting this all together we now have 5 parameters that describe a game of Spider Solitaire, the ranks, suits, multiples, columns, and deals in that order. We can apply these parameters to our standard game. For the standard game 2 decks of standard playing cards are used and arranged in a board of 10 columns with 6 deals. The parameters for this game are $(13,4,2,10,6)$. As a side note the previous configuration using 4 suits typically is referred to as the "hard" Spider Solitaire game variant in most popular implementations. I am most familiar with the Microsoft version which gives options for a "hard", "medium", and "easy" game using 4, 2, and 1 suits respectively but still maintaining a total of 104 cards. Thus we have the hard game, $G_{hard}(13,4,2,10,6)$, the medium game, $G_{medium}(13,2,4,10,6)$, and the easy game, $G_{easy}(13,1,8,10,6)$. Our references to the "standard" game in this paper always refer to the "hard" variant unless otherwise mentioned.

We will refer to the 5 integer parameters as r, s, m, c, d and note that the parameters must meet the two constraints, that $r, s, m, c, d > 0$ and $rsm \geq cd$. We can define several additional game characteristics in terms of these 5 parameters. As stated previously n is the total number of cards in the game: $n = rsm$. We define b as the number of full blocks that must be assembled and removed from a game in order to win: $b = sm$. We define j as the number of join operations needed to win a game: $j = sm(r - 1) = n - b$. Finally, we define the integer h as the number of cards hidden at the start of the game: $h = rsm - cd$. The hard, medium, and easy variants described previously each have $n = 104$, $b = 8$, $j = 96$, and $h = 44$.

4.3 An Infinite Set of Finite Spider Game Variants

How many variants of Spider Solitaire can we construct using a fixed number of cards? How many variations are there of games that use 104 cards? The analysis that follows will mostly make use of smaller variants, games with lower n values. But now that we have built the foundation we will take a brief tour of the infinite possibilities for Spider game variants.

Let G be the set of all Spider Solitaire games variants meeting the standard set of rules and described by our 5 parameters. How many such games are there? Clearly the number is countably infinite for we can simply increment any of r, s, m by 1. How many Spider Games variants are there when a fixed number of cards is used, for example, when $n = 104$? A partition of the set G is made using n , so that G_n is a subset of G where n is the number of cards in the game variants as defined by the parameters r, s, m, c, d. The hard, medium, and easy standard game variants are three of the elements of G_{104}. Can we count the number of elements in G_{104}?


Partition of the Set G of All Finite Spider Games using Standard Rules
G_1
G_2
G_3
G_4
...
G_{103}
G_{104}
G_{105}
...

Let's define the counting function $g(n) = |G_n|$ to compute the number of game variants in the set G_n. The value of n constrains the values of the 5 parameters used to specify a game variant as $r\ s$, and m must be factors of n and the product of c and d must be less than or equal to n. To compute $g(n)$ we need to count all of the possible values for the 3 parameters r, s, and m and multiply that by the count of all possible values for the pair c and d. Let's examine c and d first. Let $d(n)$ be the count of the number of ordered pairs of integers $c, d > 0$ where $cd \le n$. Trying smaller values we find $d(1) = 1$, $d(2) = 3$, namely $\{(1,1), (1,2), (2,1)\}$ and $d(3) = 5$, $\{(1,1), (1,2), (1,3), (2,1), (3,1)\}$. Using Polya's sage advice (Polya, 2004) we try a few more terms to see if a pattern emerges.

$d(4) = 8\ \{(1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (3,1), (4,1)\}$

$d(5) = 10\ \{(1,1), (1,2), (1,3), (1,4), (1,5), (2,1), (2,2), (3,1), (4,1), (5,1)\}$

$d(6) = 14\ \{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,1), (2,2), (2,3), (3,1), \ldots\}$

After a few false conjectures and computing a few more values the discovery hits that for each of the n starting elements k in the pair there will be $floor\left(\frac{n}{k}\right)$ values for the second element using integer division (floor). This yields $d(n) = \sum_{k=1}^{n} floor\left(\frac{n}{k}\right)$. The sequence $(1,3,5,8,10,14,16\ \ldots)$ emerges. A quick visit to the OEIS website (OEIS.org, 2012, p. A006218) identifies the sequence and function as such and makes a reference to the

"Dirichlet divisor problem". This could be an interesting trail to follow. But we do not want to get even further astray and so we bypass that direction.

Moving to the count of the values for r, s, m we find that we need to compute the ordered triplet of factors of n. We use a similar technique as above. We defined f(n) to be the numbered of unordered triplets whose products are equal to n. We have f(1) = 1, (1 way to arrange (1,1,1)), f(2) = 3 (3 ways to arrange (2,1,1)), f(3) = 3, (3 ways to arrange (3,1,1)). Continuing we have f(4) = 6 (4,1,1) 3 ways + (2,2,1) 3 ways, f(5) = 3 (5,1,1) 3 ways, f(6) = 9 (6,1,1) 3 ways + (3,2,1) 6 ways, f(7) = 3 (7,1,1) 3 ways f(8) = 10 (8,1,1) 3 ways + (4,2,1) 6 ways + (2,2,2) 1 way.

Continuing the above for a few more values leads to the simple observation that when n is prime the answer is 3. The values otherwise involve the factorization of n. Turning to the OEIS again (OEIS.org, 2012, p. A007425) we find that the sequence (1,3,3,6,3,9,3,10 ...) represents the number of ordered factorizations of n given k terms where k = 3.

Thus, the number of Spider game variants is defined as $g(n) = f(n)d(n)$. Since the computation of both $f(n)$ and $d(n)$ involve factorization there is no known closed form. The first 12 values for $g(n)$ are (1, 9, 15, 48, 30, 126, 48, 200, 138, 243, 87, 630), a sequence, as expected, not found on the OEIS. The value of $g(104)$ is 15,060. The hard, medium, and easy variants are 3 of the 15,060.

4.4 The Unit Game

Now back to the business at hand. What is the smallest possible variant of Spider Solitaire that we can have? Is this game variant playable? Does the smallest variant adhere to the rules of the standard game?

The smallest game variant that we can define is the unit game $G_{unit}(1,1,1,1,1)$ belonging to and the sole member of the set G_1. The unit game characteristics are n = 1, b = 1, j = 0, h = 0. In the unit game there is 1 card placed faced up in the initial deal that occurs at the start of the game using the $1! = 1$ possible arrangement. Since that card forms a full block, it is immediately removed. The game is won in 0 moves. Thus, 100% of all instances of all variations of G_1 are winnable. The game adheres to the standard rules and meets all of the constraints required. Is there a use for this unit game variant? The unit game may serve as the basis step for some clever proof by induction. But any such use by this author remains elusive.

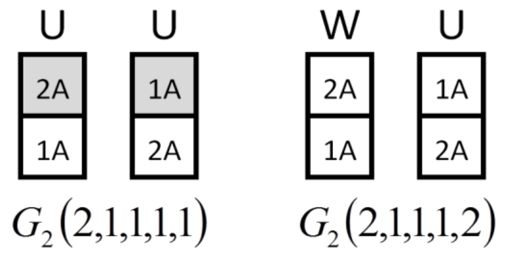
4.5 G_2 Games

We switch our attention now to the 9 elements of the set G_2 There are 3 ways to set the rsm parameters, $\{(211),(121),(112)\}$ and 3 ways to set the cd parameters, $\{(11),(12),(21)\}$. There are $n! = 2! = 2$ initial arrangements of cards to apply to each of the 9 game variants. Note that when counting game instances our analysis will normalize to permutations even when the number of unique games is smaller due to symmetries. We do this by numbering each card in a pack of n from 0 to $n-1$. This unique integer identifier i is then used to compute the rank, suit and multiple of the card using the values of the r, s, m parameters and the formulas $rank(i) = 1 + i \, mod \, r$, $suit(i) = 1 + floor\left(\frac{i}{r}\right) mod \, s$, and $multiple(i) = 1 + floor\left(\frac{i}{rs}\right)$.

Applying each of the 2 possible arrangements of cards to each of the 9 game variants will yield an outcome of either 0, 1, or 2 winning games. First, consider all of the games in which $r = 1$. Similar to $G(1)$ it would seem that all of these games must be winnable as any face up card is a full block and is immediately removed. But there is a snag. Consider an instance of the game variant $G_2(1,1,2,1,2)$. There are 2 deals, 1 initial deal with 1deal remaining. At the initial deal the first card, the full block 1, is removed leaving and an empty column that cannot be filled. Thus, the subsequent remaining deal is not permitted. This is analogous to a stalemate. For reasons to follow the "no deal with empty column" rule is amended with the addition of the clause "except when the entire board is cleared". This allows the remaining deal to occur and the game counted as winnable. Note that this amendment has no effect on the standard game. The initial board starts with 44 hidden and 10 active cards which is not a multiple of 13. Thus, the board of the standard game can never be cleared completely prior to the occurrence of the last deal. Our amended deal rule allows us to conclude that all games in the universal sets of games with $r = 1$ have a winning probability of 1.0 with the number of moves required to win 1 less than the number of deals, that is, $moves\left(G_{sm}(1, s, m, c, d)\right) = d - 1$.

The remaining games have 2 cards with ranks 1 and 2 in 3 configurations. When there are 2 columns then the board is simply the initial arrangement of cards face up. As previously noted the order of the columns does not matter, and so when there is only 1 card in each column, every initial arrangement of cards represents the same 1 unique game. Every instance of a game with only 1 row is always winnable. In general every game variant $G_c(r, s, m, c, 1)$ where $rsm = c$ has a winning probability of 1.0. The number of moves needed is equal to the number of join operations j required where $j = sm(r-1)$. In the case of $G_v(2,1,1,2,1)$ 1 move is needed, namely join the 1 to the 2 completing the full block and winning the game. For higher values of n an interesting exercise is to determine the minimum column value in relation to n that still ensures that a game variant is winnable with probability 1.0 for every initial arrangement.

We are left with the 2 variants with 2 ranks and either 1 deal or 2 deals. The variant with 1 deal has 1 hidden card. As we can see in the diagram below the 2 possible arrangements of initial cards are both unwinnable. There are no moves available in the 1 and only deal and so there is no possible way to uncover the hidden card. Earlier we saw this type of unwinnable games in the diagram depicting the standard game with no movement possible in each of its 6 deals. With 2 deals, however, there is no hidden card. And so in the case where the 2nd deal forms a full block with the 1st deal, the game is winnable.



The only 3 unwinnable G_2 games are labeled U.

cd rsm	1,1	1,2	2,1	
1,1,2	W	W	W	6
	W	W	W	
1,2,1	W	W	W	6
	W	W	W	
2,1,1	U	W	W	3
	U	U	W	
	4	5	6	15

Summary of G_2 winnable W and unwinnable U games.

4.6 G_3 and Beyond

We have now established a framework for analyzing small games variants for which we can count the number of winnable games. The thought here is that by analyzing the smaller games we might somehow be able to extrapolate to the boundaries of the number of winnable games for larger games such as our standard game. We are now positioned to ask the questions. How does each one of the 5 parameters influence the number of winnable games? Which parameters increase the number of winnable games? Which decrease the number?

Already from our simple look at the G_2 games we can make some conjectures. Increasing the number of ranks decreases the number of winnable games. Increasing the number of columns increases the number of winnable games so much so that at a certain value for a given game variant every game becomes winnable. Increasing the number of deals also increases the number of winnable games. This deal conjecture may be provable.

4.7 The Deal Theorem

Stated another way we want to prove that for any finite game of the form $G_{rsm}(r, s, m, c, d)$ that is winnable for fixed values of r, s, m, c, d the variant $G_{rsm}(r, s, m, c, d + 1)$ is also winnable provided that $c(d+1) \leq rsm$. Consider the winnable game with d deals. From the initial position there is a sequence of moves that lead to the winning position. Now consider the same game with d + 1 deals in which one more deal than the previous game is held in reserve. For this game the first move made from the initial position is a deal. Now the position is identical to the initial position of the game with d deals except that one row of hidden cards is face up instead. But from this position the same sequence of moves used to win with d deals can be followed ignoring any serendipitous joins that may have occurred. Thus, there is a path to the winning position that is one move longer, namely the deal from the initial position of the d+1 game. More deals, hence fewer hidden cards, makes the game easier.

But can we ignore all serendipitous joins? With the implicit block removal rule should a serendipitous join complete a full block, that block would be immediately removed. So the board would not quite be the same after the deal. Perhaps the same sequence of winning moves can no longer be followed. This would not be a problem if we were using an explicit removal rule. But we are not. For now we must qualify our theorem about the number of deals with the constraint that $d < r$. When the number of deals is kept less than the number of ranks, no consecutive sequence of deals can form a full block and our deal theorem holds true.

As the size of our game variants grow analysis by hand becomes increasingly difficult and soon impossible. What is needed is a computer program that can handle the chore. That is where our journey takes us next.

5. Program Analysis

Once the value of n, the number of cards in a game, exceeds about 12 examining all initial arrangements for a particular game variant becomes problematic. Larger values soon become intractable. One approach to addressing this issue is to generate random initial arrangements and run those arrangements through a computer program that simulates the play of the game to determine if the game is winnable. If the program finds a sequence of moves leading to the winning position, the game is winnable. If not, then the game may or may not be winnable as the program may have failed to find an existing winning sequence of moves due to either a limitation of the program or a programming error.

5.1 Rationale for Writing a Program

There is a webpage that contains the results of a computer analysis of 30,000 randomly generated standard games (Alex at Tranzoa Company, 2005). The program used to simulate the play of the games is freely downloadable. Of the games analyzed the program was unable to find a solution for only 11 games. Two games are presented that appear to be unwinnable. After briefly looking at the downloaded program I decided to undertake a similar computer analysis but using my own program for a few reasons. First, I wanted to be able to analyze the variants of games discussed above and collect additional statistics in addition to the number of moves in a winning solution. One such statistic is the number of each of the types of move previously defined that were used in the winning sequence of moves. I also wanted a program that would try to find the best solution to a winnable game, that is, the solution using the fewest moves. Finally, Gregory Chaitin in his book about his quest for Omega, a topic unrelated to this, argues that programming a problem helps give a deeper understanding of the problem itself (Chaitin, 2005, p. 44).

5.2 Difficulties Simulating the Play of a Standard Spider Solitaire Game

Having a program play standard Spider Solitaire games has a number of obstacles to overcome. Compared to the number of possible games, the number of games examined will be extremely small. When random arrangements of cards are generated it is not known if the small sampling produced is representative of all possible arrangements. Chapter 3 of Knuth's The Art of Computer Programming (Knuth D. E., 1981) goes into great detail both about the methods for generating random sequences and how to test for the effectiveness of the method employed. Again, this is a topic outside the scope of this paper. But keep in mind that the data collected and presented here generated from computer analysis and simulation is based on the workings of a particular implementation of a random number generator. The program uses the Random class that is part of the Java 7 Standard Edition from Oracle Corporation (docs.oracle.com, 2012). We have taken some measures to give us confidence that our random samples are representative We have at our disposal the distribution frequencies for ranks, suits, and deals with at least one adjacent pair that we calculated in an earlier section. We can compare the statistics collected from the generated random initial arrangements to these expectations. If the results are close, we gain some confidence that our random initial arrangements are representative of the full sample (Freund & Perles, 2007). The statistics for rank and suit distributions and that for deals with adjacent pairs have all closely aligned with expectations.

When the number of possible moves at each position in a game exceeds a certain threshold analyzing a game to find a winning sequence of moves becomes a heuristic search of a large tree of game positions. This is true for the standard game. The search cannot be exhaustive.

Consider, for example, that each position is the game has 2 possible moves that can be made. In actual play the average number of moves available at each position is higher than 2. Let's assume that 96 moves, the number of join operations, are required to win a particular game. That would be $2^{95} \cong 3.9614 \cdot 10^{28}$ positions to be examined which is intractable. And so the search space must be reduced using some heuristic method. But a heuristic search brings uncertainty. If the search finds a winning sequence of moves then the game is clearly winnable. The contrary is not true. A game where no winning move sequence is found might not be unwinnable. Perhaps the path to victory traversed a position that was erroneously pruned from the search tree.

Another problem arises when analyzing simulated runs of the standard game. Based on the data from our analysis to date and that on the web (Alex at Tranzoa Company, 2005) more than 99.9% of games analyzed were found to be winnable. So there is in actuality a lack of unwinnable standard games to analyze for their patterns to gain insight into what constitutes an unwinnable arrangement. The analysis of smaller games with a higher proportion of unwinnable games will help to shed light in this area.

5.3 Breadth First Search with Heuristic Pruning

We still want to simulate the play of standard games. At a minimum we can compare the outcomes of the standard games to those of smaller variants. Thus, we need to describe the search methodology used by the program to find a winning solution to a particular instance of a standard game.

We can think of the search for a winning solution to a Spider game variant as a traversal of a directed graph with board positions as vertices and moves as edges. The implementation chosen starts with the initial board position then generates a set of new positions by performing every possible move, including a deal if available, from the starting position. This creates a new set of positions a distance of 1 move from the start. If no position is the winning position, all positions for each position at the second level are then generated creating a set of positions at a distance 2 moves from the initial position. These are examined. This amounts to a breadth first search of the graph so that if the winning position is found, then we know that the game is winnable and that the route to the winning position is the shortest route, that is, uses the fewest moves possible to win the game. But there is a wrinkle. The number of positions to examine at each subsequent level grows rapidly and quickly exceeds the capacity of the computer in terms of both space and time. Exhaustively examining every position is intractable. Thus, some form of pruning of the search space, the graph of positions, is required.

5.4 The Fitness Function

A fitness function is a relative measure of a position that estimates the potential for a winning path to be found from that position to the winning position. The value of the function can be used to compare individual positions to one another. When a pruning of the search tree needs to be done, the positions with the poorest values for their fitness functions are removed while those positions with the best values are retained. The retained positions are then used to generate all the possible new positions at the next level where the pruning process is repeated as necessary.

After a few experiments with different types of functions, some overly complex, a rather simple and straight forward function was found that works effectively. The function simply sums the number of blocks that still need removed from the game, the number of join operations required to put together all of the remaining blocks, and the total number of cards that are still hidden in the game. So for a given position p_i the fitness function is $fitness(p_i) = b + j + h$ where b is blocks remaining, j is join operations remaining, and h is hidden cards remaining. Low values are favorable while high values are unfavorable. The winning position with all blocks formed and the board empty has a fitness function value of 0. When two positions produce identical fitness function values and one must be pruned, a simple tie breaking algorithm is invoked. The position with the fewest hidden cards remaining is kept. If the fitness values are still tied then the position with the fewest joins remaining is kept. Finally, if still tied, the position with the fewest blocks out of sequence is kept. In hindsight, a better final tie breaker may have been the number of occluded blocks showing for reasons presented later.

5.5 Impacts of Position Pruning

The effects of pruning positions from the search space of positions are not well understood by the author. Increasing the number of positions kept at each layer appears to increase the number of winning sequences of moves found and in some cases may find a shorter winning sequence than one found at a lower value. But the number of positions kept must be limited to prevent the computing resources, both in terms of space and time, from being exhausted. For the standard game the search tree is limited initially to a maximum of 64,000 positions kept at each level. We will call this the "width" of the search. When more than 64,000 positions are generated, the positions are sorted by fitness function value and the lowest

64,000 are retained. This value was found to give a reasonable, though still rather time consuming, game simulation that finds a winnable solution to a standard game, when there is a winnable solution, in 97.9% of the cases. In the cases where the program fails to find a solution the program tries again this time increasing the width to 128,000 positions. A solution was found for an additional 1.2% of the cases. Then, counter-intuitively, the program tries a smaller width of 32,000 positions. Here the program found 6 solutions of the 3199 found to date that neither the search with 64,000 or 128,000 positions were able to find. If a solution is still not found, the program tries one final time with a width of 256,000 positions. The sequence of widths w automatically tried by the program are the 4 values $\langle w, 2w, \frac{w}{2}, 4w \rangle$. If no solution is found, the program declares that there is no solution for that particular game instance. Of 3200 simulated games, 12 fell into this category. Of those 12, solutions were found for all but 1 by running the program with even higher width values. The maximum width possible on our current computing equipment is about 1,200,000. A single simulated game run at this width can take several hours of elapsed time.

One of the inherent problems with the pruning method employed is that a single favorable position that has a high number of available moves can in the short term quickly come to dominate the favorable positions retained. If that single position does not lead to the winning position, many other positions less favorable in the short term but leading eventually to the winning position may have been pruned away. In rarer circumstances as evidenced by the few wins found at a lower width, such a short term favorable position may be prevented from being introduced and so is not afforded the opportunity to dominate the pool of favorable positions. The literature on the subject of "Genetic Algorithms" is a good source for information on the topic of fitness functions and is beyond the scope of our current analysis.

5.6 The Number of Moves Available from a Position

A reasonable question to ask at this point, alluded to earlier, is how many moves are possible from a given position? This will give us some idea as to how rapidly the number of positions can grow from a single position.

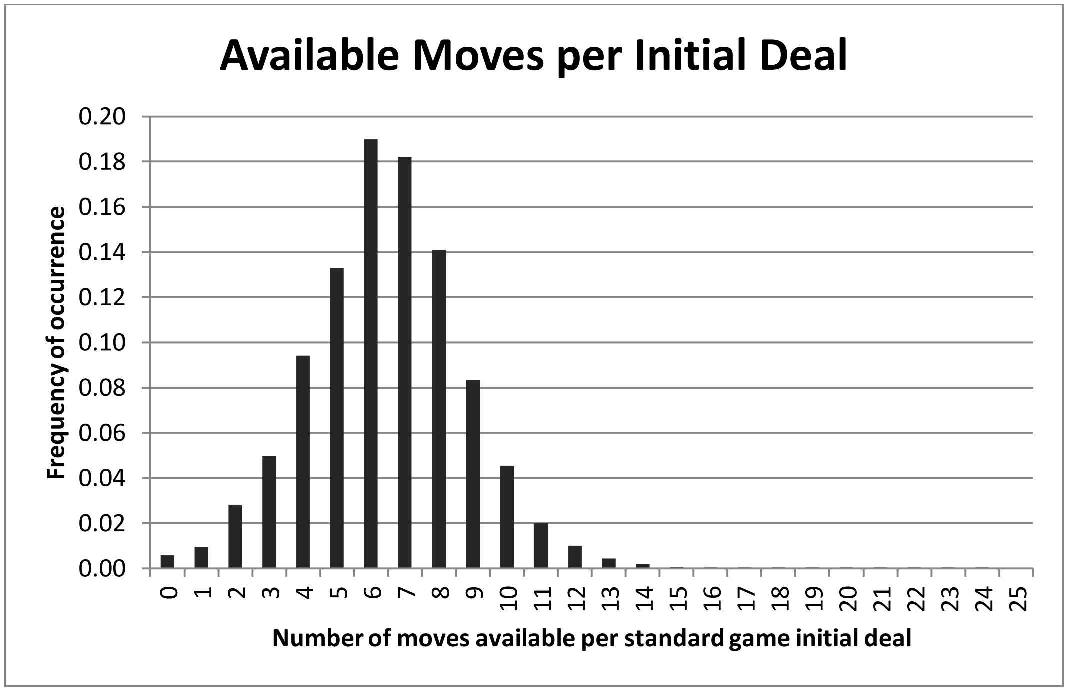
We define a function $f(c)$ that computes the maximum number of moves possible for a given column size c exclusive of any deal move possible. Also, $f(c)$ does not include any moves possible due to block splitting. One naive attempt might be to have cards of increasing rank make up the columns and as long as $r \geq c$ there are $c - 1$ moves available. But we can do better. After some playing around we discover that if we make half of the columns one rank, and the other half one rank higher, then each of the higher ranking cards can accommodate each of the lower ranking cards. When there is an even number of columns this yields $f(c) = \left(\frac{c}{2}\right)^2 = \frac{c^2}{4}$. When there is an odd number of columns the halving gives a difference of 1 so we have $f(c) = \left(\frac{c+1}{2}\right)\left(\frac{c-1}{2}\right) = \frac{c^2-1}{4}$. Both cases are condensed to one when we use integer division with its implied floor function. Thus, both cases are handled by the first function shown. We assume that $r > 2$ and compute the values for c starting at 1 to obtain the sequence $\langle 0,1,2,4,6,9,12,16,20,25,\ldots \rangle$. Another quick trip to the OEIS finds sequence A002620, the sequence for "quarter-squares" and one explanation of the "maximum product of two integers whose sum is n" (OEIS.org, 2012). For the standard game we have

$f(10) = 25$. So a given position might have as many as 25 possible positions to explore at the next level plus 1 more if there is a deal available.

But what if we allow blocks to form with each deal? How does this affect the number of possible moves? Consider the case of 4 columns where we have $f(4) = 4$, for example, if the columns hold the ranks $\langle 5,5,4,4 \rangle$. Now consider the columns holding the values $\langle 5,4,3,2 \rangle$, the row above $\langle r, r, 4,3 \rangle$, and the row above that $\langle r, r, r, 4 \rangle$. Thus the 3rd and 4th columns form blocks of 2 cards and 3 cards respectively. Now we can move the blocks from 3 columns onto the 5, 2 columns onto the 4, and 1 column onto the 3. So there are 6 possible moves which is greater than the 4 possible when we did not consider block formation. The result is simply the sum of the first $c - 1$ integers.

Finally we consider the case where there is an empty column available. Every non-empty column can move a card to the empty column. If the non-empty columns have multi-card blocks, each block can be split and moved. The worst case involves blocks that are $r - 1$ in length. But even allowing for the minimum number of moves into the empty column of $c - 1$ to that we must also add the number of moves possible among the non-empty columns which is $f(c - 1)$.

As each case above uses a formula dependent on the number of columns it becomes apparent that increasing the number of columns also increases the number of moves possible with each position. Thus, we can see that the complexity of the search tree grows as a function of the number of columns.



The preceding chart reflects data collected from generating 5 million initial arrangements from the standard game and examining the 10 cards that appear at the location of the initial deal. The number of moves available for each deal was collected with the results shown. Of note is that the expectation for the probability of 0 moves calculated previously as 0.00586 occurred in the sample collected at a frequency of 0.00585. Though hard to see in the chart every number of possible moves except for the maximum of 25 appears in the sample. There were 2 deals each with 23 and 24 moves, an extremely low frequency. The average number of moves across the entire sample is 6.45222.

5.7 Generating and Simulating Random Games

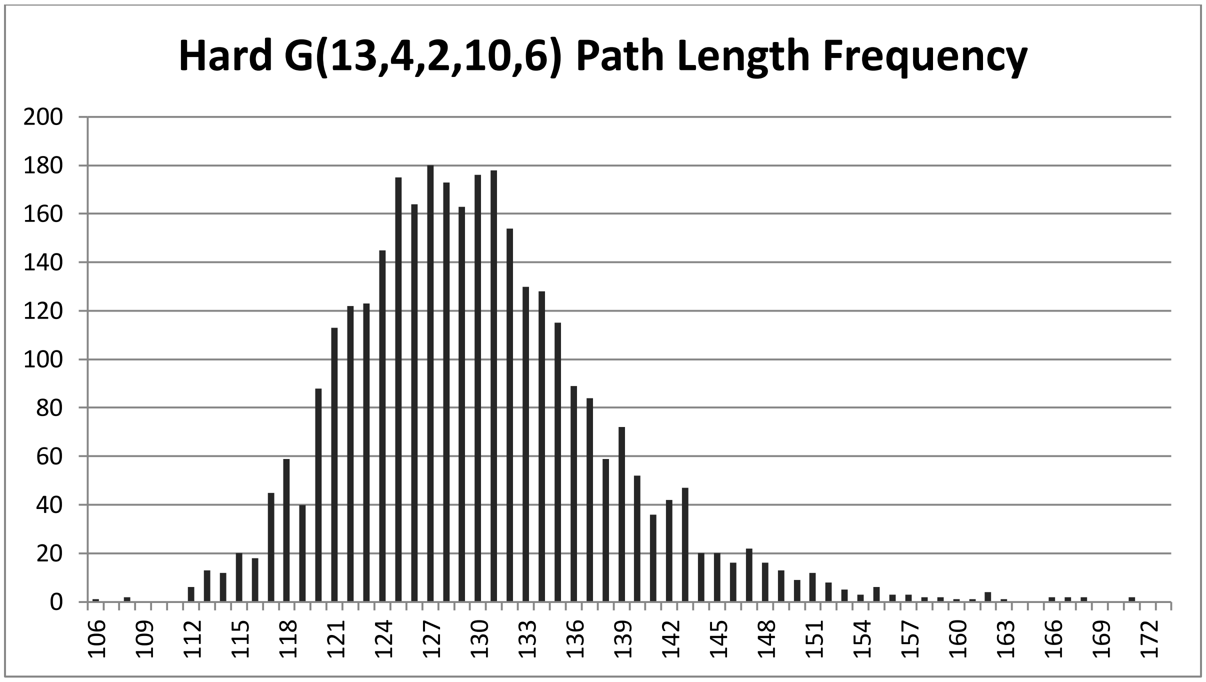
Now we put our program to work on its primary mission of generating and playing various spider game variants. The general approach is to specify the 5 parameters of the game variant and then allow the program to generate random initial arrangements using a sequence of random number "seed" values. Using a specific seed value and the same shuffling algorithm produces a pseudo random initial arrangement that can be reproduced by providing the same initial seed value. The program can be set to generate a number of initial game arrangements, simulate the play of the game using the breadth first heuristic search described above, and count the number of winnable games and those for which a winning sequence of moves could not be found. Additional statistics for each game can be collected along the way.

5.8 Simulated Game Data Presented

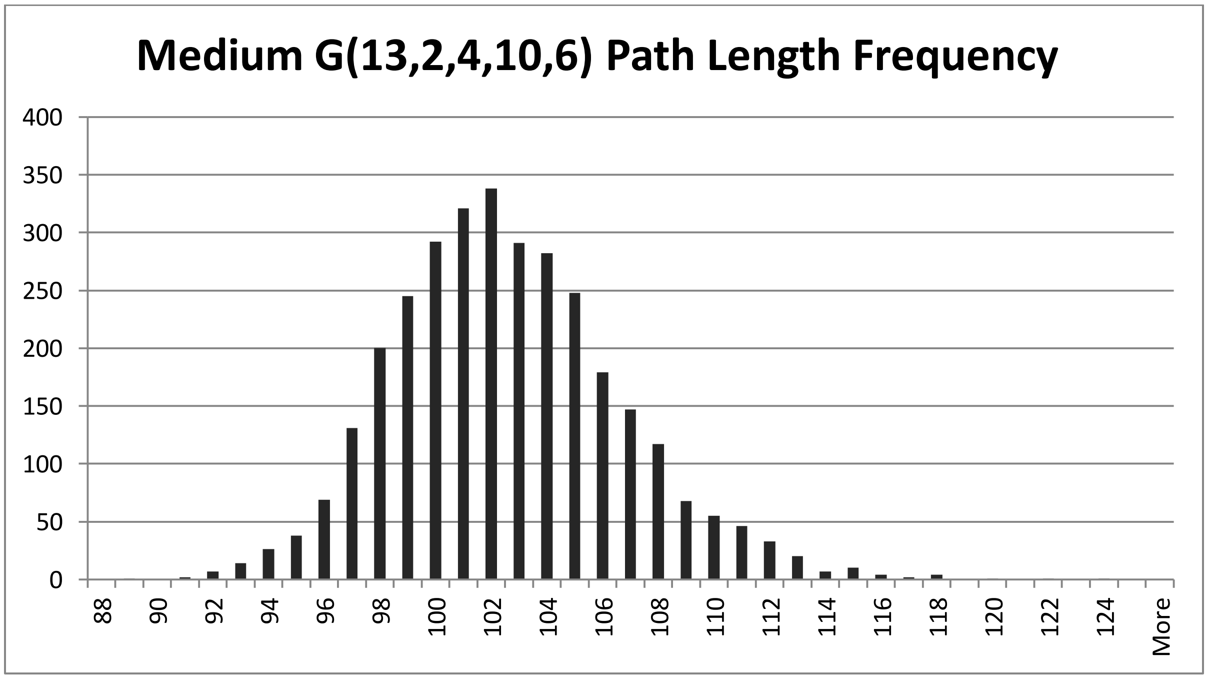
In the previous sections we looked at small games which led to some insights and conjectures. Now we can run computerized analysis on game variants ranging in size from those small games to the standard games. The data collected from these various simulation runs is shown in the charts that follow. The charts show the frequency of the path lengths in all of the winning games found. The path length is the number of moves taken from the initial position to the winning position with all cards removed. The path length for a particular game instance is not guaranteed to be the shortest where pruning of the search tree has occurred, but even in that case, the path length is biased towards the shortest route. The charts also include the game variant selected, the count of winnable versus the total number of games simulated, the percentage of winning games of the number of games analyzed, and the minimum, maximum, and average number of moves taken in winning the game.

Our simulation runs start with the charts for the three standard spider game variants, the hard, medium, and easy games. Following those is the data collected from a "half" game and some insights gathered from that data. Finally, the data for one still smaller variant is presented.

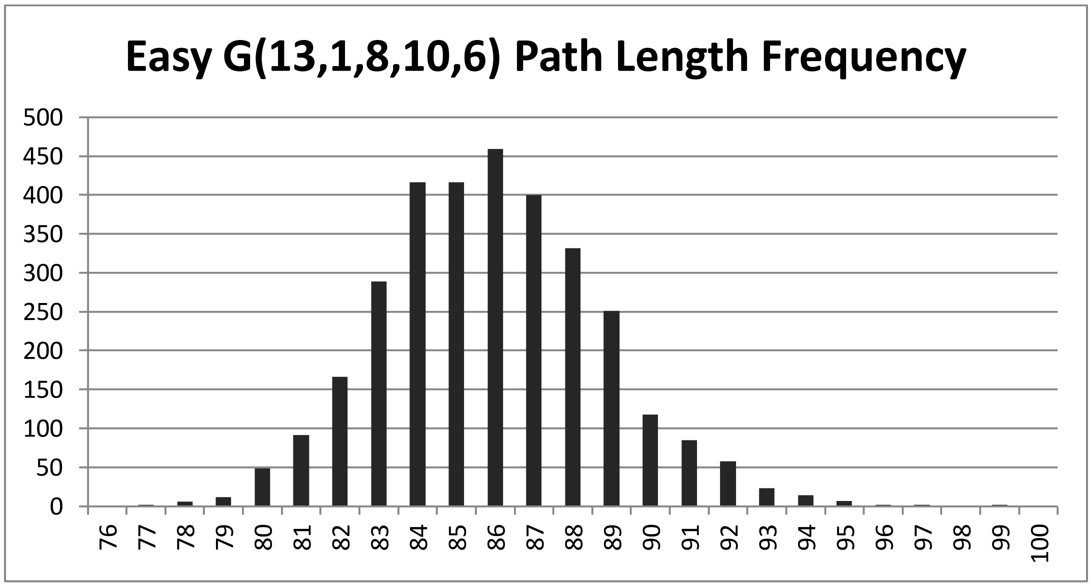
5.9 Results of Simulations Runs for the Standard Game Variants



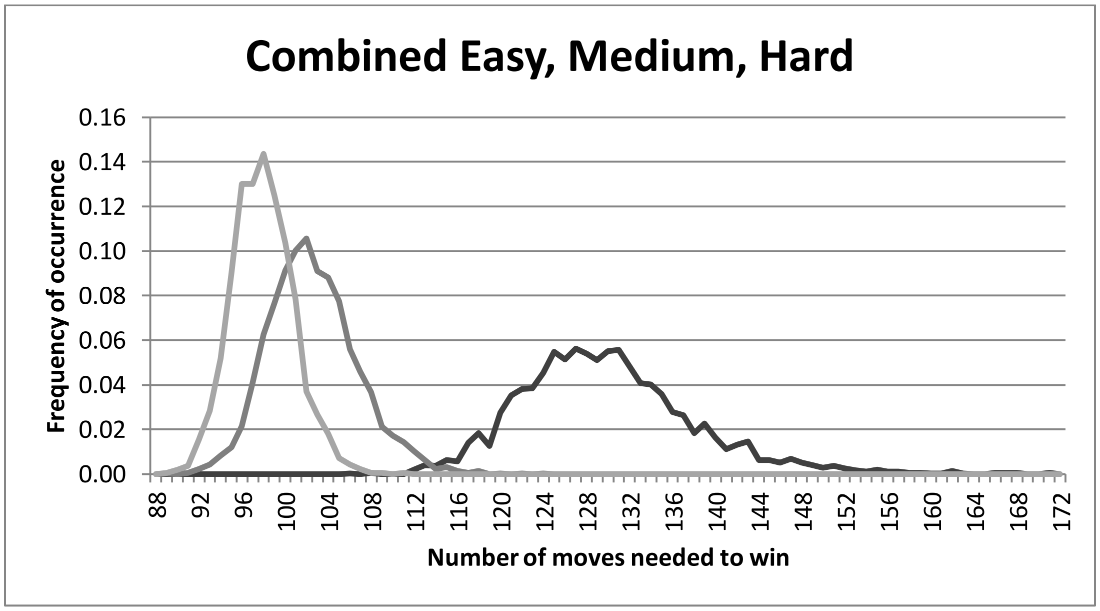
Count=3199 of 3200 (99.969%) min=106 max=171 average=129.896



Count=3200 of 3200 (100%) min=89 max=124 average=102.576



Count=3200 of 3200 (100%) min=77 max=99 average=85.934



Combined move frequencies of all winning standard games

The charts from the hard, medium, and easy variants of the standard games analyzed immediately surface the fact that one of the difficulties for the human participant is that the games require an increasing number of moves to win when going from easy to hard. The final chart shows the three variants combined so that this becomes readily apparent.

The chart of the hard game is more ragged when compared to the medium and easy charts. The hard game may be more sensitive to the width of the position search so that the true shortest winning path is not always found, perhaps being pruned away, at some earlier point in the search. Running the hard games at higher search width values may smooth out the graph to be more in line with the medium and easy charts. But our current simulation program is not efficient enough to do so in a timely manner. We intend to conduct this experiment with higher width values in the future. Our conjecture is that the frequency distribution will come to more closely resemble that shown in the charts of the medium and easy game variants though still proportionately wider as the combined chart shows.

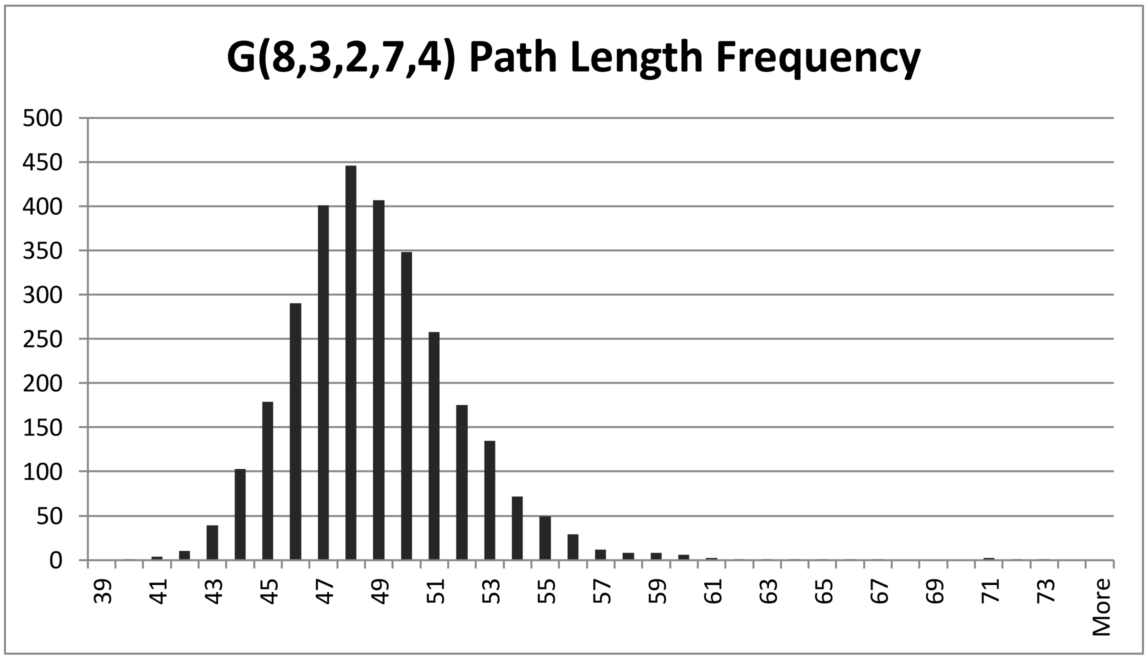
The charts of the medium and easy games reveal how the simulation program exploits the use of deal moves to produce serendipitous joins. The data shows that 88 medium games are won in less moves than the 96 explicit join operations required to win the game. The easy game averages 10 less moves than the number of explicit join operations required. The chart below shows the average number for each move type across all of the analyzed winning standard games.

Average Move Types Per Game Won			
Move Type	**Standard Game Variant**		
	Hard	**Medium**	**Easy**
Join Block	90.023	85.401	78.656
Move Block	34.158	11.712	2.042
Join Split	0.688	0.456	0.234
Move Split	0.027	0.007	0.002
Deal Joins	6.004	10.606	17.346

Upon reflection, this is easy to see for the easy game. Every movement operation unless onto an empty column is a join operation. Few move block operations are needed. Of the 3,200 easy games simulated and won the average number of explicit join operations per game was 78.656 with the average number of implicit joins occurring with deal operations 17.346.

5.10 The Half Game

The "half" game, $G_{half}(8,3,2,7,4)$, is a member of G_{48}. The parameters for the half game were selected to align as closely as practical to the characteristics of the standard game. In the standard game the number of columns falls between the number of blocks and the number of ranks, $b < c < r$. The standard game is $8 < 10 < 13$ while the half game is $6 < 7 < 8$. The ratio of hidden cards to total cards, $\frac{h}{n}$, is $\frac{44}{104} \cong 0.423$ in the standard game and $\frac{20}{48} \cong 0.416$ in the half game. Both games have an m value of 2. The half game also has a hard, medium, and easy variant, $G_{hard}(8,3,2,7,4)$, $G_{medium}(8,2,3,7,4)$, and $G_{easy}(8,1,6,7,4)$.



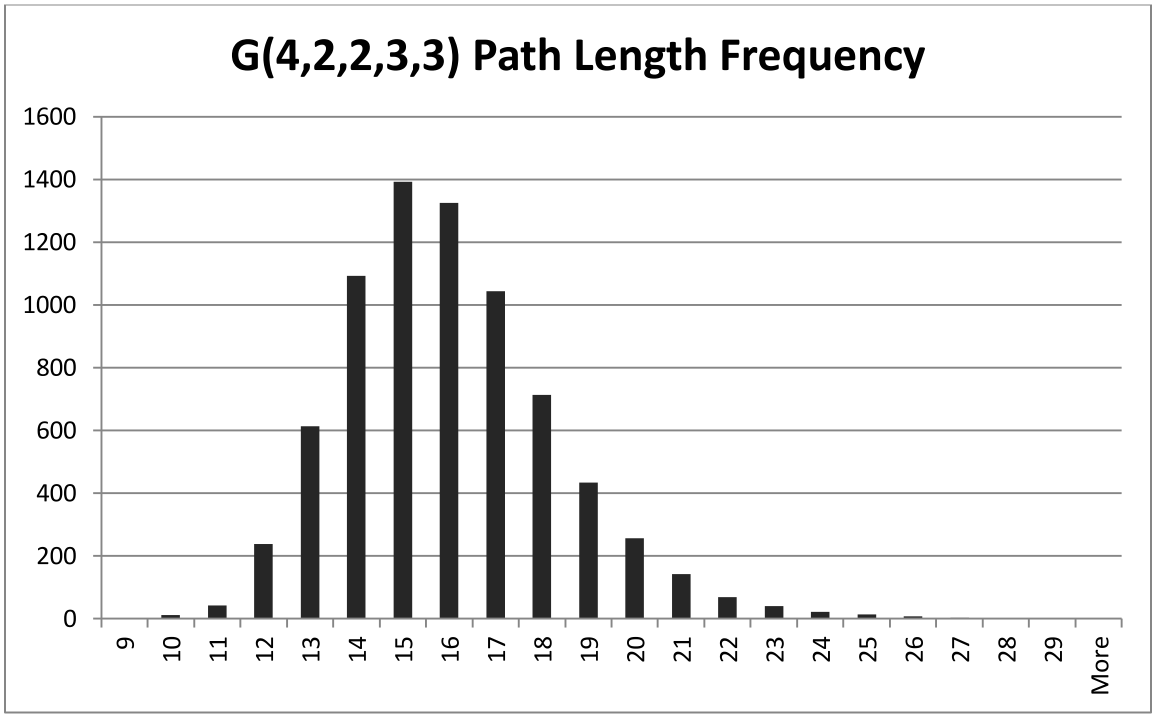
Count=2992 of 3000 (99.733%) min=40 max=73 average=48.871

The data generated from the simulation of random half games has surfaced one interesting finding. The program found 2992 of 3000 random half games winnable. 6 of the remaining 8 games were demonstrated to be unwinnable. In those cases, the number of positions examined never exceeded the maximum capacity of the computer resource available and so no pruning of the search tree needed to be done. Thus, an exhaustive search was completed and no winning sequence of moves was found. The remaining 2 games generated sufficient positions to require pruning and so fell into the unknown state between winnable and unwinnable. The 6 unwinnable initial arrangements of cards were run through the program again but this time with a modified half game that reduced the number of suits to 1, $G_{easy}(8,1,6,7,4)$. A winning solution was found for 5 of the 6 games with 1 game again demonstrated unwinnable. So far we have only seen unwinnable games based solely on card ranks. Since reducing the number of suits to 1 would not impact a game that is unwinnable based on ranks alone, those 5 games must demonstrate patterns that are unwinnable due to both card ranks and card suits. We can conclude that the 1 game unwinnable in both cases is so because of ranks alone. Thus, the set U of unwinnable games can be divided into two disjoint subsets. The set U_r contains games that are unwinnable based on card rank alone. The set U_s contains games that are unwinnable based on some combination of both rank and suit. For any finite game variant with $s = 1$ the set U_s is empty. It will be interesting to extend the number of simulation runs to see if the ratio of 5 to 1 of unwinnable by suit and rank to unwinnable by rank holds, that is, $5|U_r| \cong |U_s|$. Coincidentally, the Tranzoa website states that a winning solution for 11 of 30,000 simulated games could not be found. Two of those games are listed explicitly with commentary describing games unwinnable by rank. Both of those games were run though our simulator but with the number of suits set to 1. The program could not find a solution to either game supporting the statement that the two games are unwinnable by rank. The other 9 unwinnable games did not have a clear explanation and were not explicitly shown on the website. Perhaps a ratio similar to the half game holds for the standard game. More samples of unwinnable games are needed to further this analysis.

We make one final comment on the two unwinnable Tranzoa games. In both cases the program simulation shows that one particular card rank never appears among the active cards during the simulation run. In the game labeled 1748 the card rank 6 is never surfaced. In game 14934 it is the card rank 3. A reasonable conjecture to make is that any game that is unwinnable by rank will have at least one card rank that never appears among the active cards. In other words, all occurrences appear within the hidden cards or are part of a deal in which no movement is possible and then immediately occluded by the subsequent deal. If this conjecture is true, it follows that any game in which it is possible to remove 1 full block, even though a winning solution cannot be found, cannot be a member of the set U_r.

5.11 Minimum and Maximum Path Lengths of Winning Games

The data from the final game variant, $G_{16}(4,2,2,3,3)$ is interesting for a few reasons. First because the number of cards, and more importantly columns, is relatively small, the program is able to conduct an exhaustive search without the need for pruning throughout. Though the winning percentage is relatively low, the winning path lengths shown are accurate. Of interest is the long tail to the right of the median. The graph is coincidentally similar to the unrelated graph of available moves shown in section 5.6. But for now, returning to the discussion at hand, a natural question to ask is what is the longest path length of the shortest winning path in this game variant?

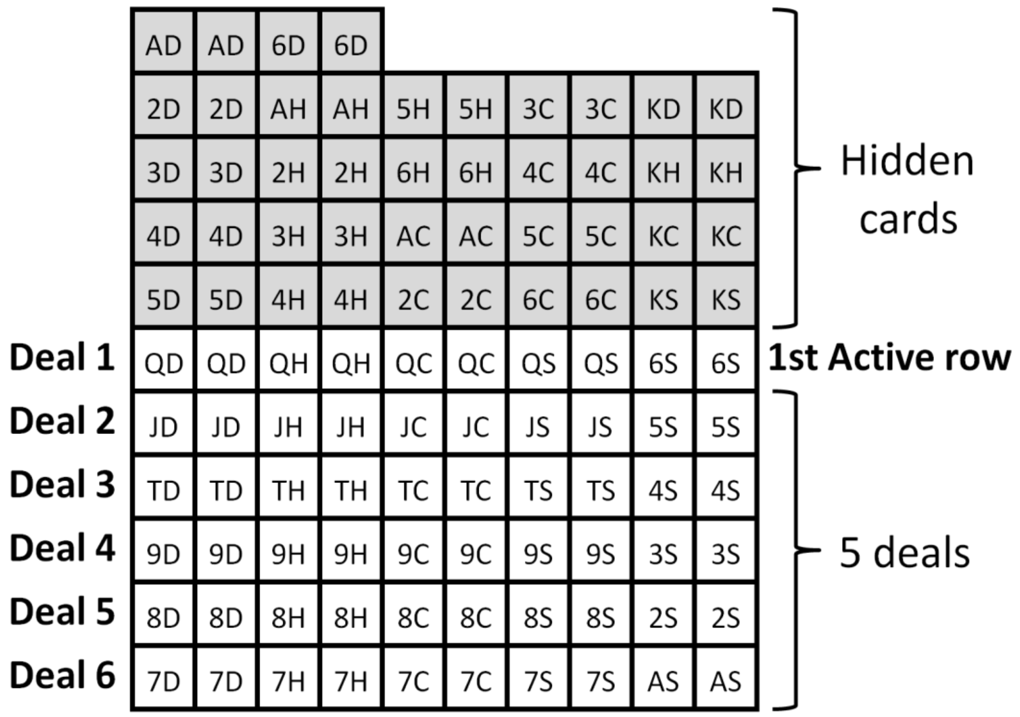


Count=7468 of 20000 %=0.3734 min=9 max=29 Average=16.0004

We can see that the answer is at least 29. But is there another initial arrangement that requires a minimum number of moves greater than 29 to win? Is there a function that can compute this value? The first chart shown in section 5.9 depicts the standard spider game, $G_{104}(13,4,2,10,6)$. That shows a similar shaped graph to this. But there many positions were pruned from the search tree in determining the winning path. And so the number of moves may overstate the actual value of the number of moves needed to win a particular game. In this case having a function to compute the maximum number of moves required to win any game of a game variant would help. The program would know to stop searching when the path length exceeded that maximum value.

The question of the minimum number of moves required to win can be computed at least to a theoretical value. To win a game all of the full blocks of cards need to be assembled. Each assembly is a join operation, either an explicit join movement type or a serendipitous join occurring during a deal subsequent to the initial deal. Theoretically, if there are c columns, then there can be c joins that occur with a single deal. Thus, we can compute the minimum by adding the number of joins required, $j = (r-1)sm$, and the number of subsequent deals that must be made, $d-1$, then subtract the potential serendipitous joins occurring during the deals, $j + (d-1) - c(d-1)$. For the standard game the minimum moves needed is $96 + 5 - 10 \cdot 5 = 51$. Is there an initial arrangement that is winnable in this theoretical minimum? The initial board configuration show below is one such arrangement. When fed into the program simulator this arrangement produced a winning path using 51 moves, 46 joins and 5 deals.

Winnable Minimum Path Length Game



	AD	AD	6D	6D							Hidden cards
	2D	2D	AH	AH	5H	5H	3C	3C	KD	KD	
	3D	3D	2H	2H	6H	6H	4C	4C	KH	KH	
	4D	4D	3H	3H	AC	AC	5C	5C	KC	KC	
	5D	5D	4H	4H	2C	2C	6C	6C	KS	KS	
Deal 1	QD	QD	QH	QH	QC	QC	QS	QS	6S	6S	**1st Active row**
Deal 2	JD	JD	JH	JH	JC	JC	JS	JS	5S	5S	5 deals
Deal 3	TD	TD	TH	TH	TC	TC	TS	TS	4S	4S	
Deal 4	9D	9D	9H	9H	9C	9C	9S	9S	3S	3S	
Deal 5	8D	8D	8H	8H	8C	8C	8S	8S	2S	2S	
Deal 6	7D	7D	7H	7H	7C	7C	7S	7S	AS	AS	

$$\min\left(G_{104}(13,4,2,10,6)\right) = 51$$

Ironically, no movement is possible in each of the 6 individual rows of deals. Only after the last deal is movement possible with the blocks formed along the way. There are 16 possible moves at that point. Each of the 8 7's can receive either of the 2 blocks of 6 through ace. But only joins should be considered. After completing 46 consecutive joins, the board will be cleared and the game won.

5.12 Analysis of the Effect of Each of the 5 Game Parameters

We now turn our simulator to work in an attempt to see if we can discover the impact of each of the 5 parameters on the ratio of winnable games to total games. We use our small game, $G_{16}(4,2,2,3,3)$, as a starting point and vary only one of the parameters at a time to see how the winning ratio changes.

We start by changing only the rank value. We use values of 3 through 6 which includes our small game. As the value of r increases the winning ratio decreases with values of 0.867, 0.372, 0.041, 0.001. It seems that the number of ranks has a dramatic effect on the winning ratio. But increasing the r value also changes the ratio, $\frac{h}{n}$, of hidden cards to total cards. Is it the increase is this ratio that is really the cause of the decrease in the win ratio? The waters are muddy indeed. As we will see all 5 parameters change the hidden ratio. What we must try to do is offset the change caused by increasing the rank by either decreasing the suit or multiple parameter, or increasing the column or deal parameter. We already have a theorem that states that increasing the value of d increases the winning ratio. So we need to try to compare the increased r values to results where other parameters have changed to bring the hidden ratio into a similar proportion. So samples from many game variants will be needed. But for now, we will conjecture that increasing the value of r decreases the winning ratio.

				c	3	3	3	3	3	4	4	4	5	5	5
	r	s	m	d	1	2	3	4	5	3	4	5	3	4	5
Rank	3	2	2		0.245	0.639	0.867	0.924		0.996					
	4	2	2		0.026	0.143	0.372	0.628	0.794	0.969	0.990		0.999		
	5	2	2		0.000	0.023	0.041	0.140	0.311	0.836	0.960	0.987	0.998	0.999	
	6	2	2		0.000	0.000	0.001	0.006	0.025	0.454	0.756	0.917	0.964	0.997	
Suit	4	2	2		0.026	0.143	0.372	0.628	0.794	0.969	0.990		0.999		
	4	3	2		0.000	0.000	0.004	0.017	0.064	0.648	0.869	0.960	0.985	0.999	
	4	4	2		0.000	0.000	0.000	0.000	0.000	0.121	0.362	0.634	0.874	0.973	0.994
	4	5	2		0.000	0.000	0.000	0.000	0.000	0.002	0.013	0.076	0.515	0.818	0.934
Multiple	4	2	2		0.026	0.143	0.372	0.628	0.794	0.969	0.990		0.999		
	4	2	3		0.001	0.009	0.046	0.148	0.307	0.829	0.957	0.985	0.992	0.999	
	4	2	4		0.000	0.000	0.002	0.013	0.041	0.619	0.830	0.939	0.970	0.993	0.999
	4	2	5		0.000	0.000	0.000	0.001	0.004	0.402	0.661	0.833	0.929	0.981	0.995

Winning Ratios for Various Game Variants

Increasing the suit parameter also increases the hidden ratio. In addition, increasing the suit parameter increases the number of full blocks that need to be removed from the game. Increasing the suit parameter decreases the winning ratio and the data from the smaller variants suggest that it does so to a greater degree than increasing the rank parameter. This is likely the underlying reason that the hard, medium, and easy variants change the suit value.

Increasing the multiple parameter also decreases the winning ratio but to a marginally lesser degree than increasing the rank parameter. More blocks will need to be removed but at the same time there are increasing multiples of each card that can be used interchangeably to form blocks somewhat dissipating the impact.

Increasing the number of columns increases the winning ratio. In the smaller game variants being analyzed the increase in the winning ratio is substantial. As previously discussed increasing the column value will eventually lead to games that are trivially winnable in all cases. When there are sufficient number of ranks and suits increasing the number of columns also increases the complexity of the game as the number of moves available at each position is a function if the number of columns.

In summary increasing the value of rank, suit, or multiple decreases the winning ratio of a game variant. Increasing the value of column or deal increases the winning ratio of a game variant. The column and suit parameters appear to have the biggest impact on the winning ratio of game variants. Further analysis is needed to isolate the effects of the each parameter.

5.13 Future Experiments Using the Program Simulator

We have only scratched the surface of the experiments that can be conducted with our program simulator and the variants of games defined by our 5 parameters. An important first step going forward will be to optimize our program to allow more efficient simulations with more positions to be included in searches. For our standard game we want to run many more randomly generated games to establish a base of unwinnable games to analyze. We also would like to run simulations using more variants to try to isolate each parameter to determine the impact each has on the winning percentage of games. We started this analysis with a few small games but the parameters, with perhaps the exception of the deal parameter, are difficult to isolate particularly because of the influence of the changing ratio of hidden cards. Perhaps by running more variants we might be able to plot a trajectory that leads to upper and lower bounds to the number of winnable games in larger variants, particularly the standard game. But for now we conclude our analysis simulating finite spider game variants. The infinite awaits.

6. Spider Game Variants Using an Infinite Number of Cards

So far our discussion and analysis has focused on finite game variants played with a fixed number of cards. Our goal has been to find a winning sequence of moves, as short as possible, from an initial arrangement of the cards. In this section we narrow our focus to individual board positions of game variants. We will talk about sets of positions, operations on positions, generating positions, and traversing positions. Instead of winnable games we speak of winning positions.

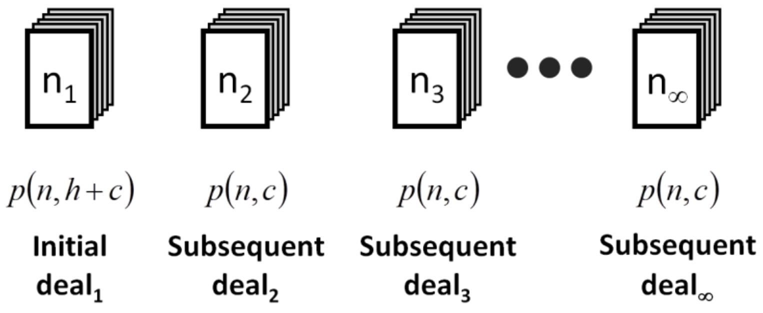
The sequence of moves in a game instance can be viewed as a path traversing a directed graph of board positions. The vertices of the graph represent each unique board position visited. The edges of the graph represent the moves taken to traverse from one position to another. We can now rephrase our original question. Is there a route in the graph from each initial game position that can be followed that leads to the winning vertex, the position with no cards left in the game? If there is such a route we call the initial position a "winning" position. But we will also be adding a new wrinkle to this analysis of positions. We place no limit on the number of times that the deal move type can occur. In other words, there is an infinite number of cards in play. We now ask the question, is every position that may occur in an infinite game a winning position? Our exploration of infinite Spider game variants begins.

6.1 Definition of an Infinite Spider Game Variant

We need a precise definition of infinite game variants. We use the notation G_∞ to distinguish the infinite variants from the finite variants and note that $G_\infty \notin G_n$ as the total number of cards that may occur in the infinite game is unlimited. Infinite game variants are defined by the same 5 parameters as finite games. Every infinite game variant $G_\infty(r,s,m,c,d)$ has a finite game variant counterpart $G_{rsm}(r,s,m,c,d)$. The starting values of r, s, and m in the infinite game, just as in the finite game, establishes the value of n, the number of cards in the pack of cards used in the game. We make the number of cards unlimited in the infinite game variant by allowing the number of cards in a game position to exceed the fixed value of n. The cards appearing through the course of the infinite game must be copies of the members of the initial pack of n cards. Conceptually, the game parameter m increases as needed during the play of the game while holding the values chosen for r and s fixed. To maintain the board structure of the infinite game the column parameter c is also fixed while the number of deals, d, is unbounded.

Deal operations embody the difference between infinite and finite games. Each deal operation in the infinite game is performed using its own identical clone of the pack of cards of size n established by the initial game parameters. We further refine deal operations into 2 types, the initial deal and the subsequent deals made throughout the play of the game. The initial deal produces the starting board position which contains hidden cards if there are any and the first row of active cards. The initial position contains an arrangement of cards, $p(n,h+c)$, selected from the pack of n cards with $h = rsm - cd$ hidden cards and c active cards. The initial deal operation produces the same starting position in both the infinite variant and its finite counterpart. In the infinite game the value d from the initial parameters is used only to establish the number of hidden cards in the starting position and is otherwise disregarded.

The subsequent deals of the infinite game differ substantially from the subsequent deals of the finite game. Each subsequent deal in the infinite game selects an arrangement of c cards from its own complete clone of the original pack of n cards, $p(n, c)$. Such a subsequent deal can be made at anytime and any number of times. Once the selection of c cards is made the remaining cards in the pack are discarded. The next subsequent deal will be taken from its own identical clone of the initial pack of n cards. We can think of the infinite game as having an infinite number of identical packs of cards established by the game parameters each assigned to a particular deal operation as depicted in the diagram below. Note that $n_1 = n_2 = \cdots = n_\infty$.



As an example, the infinite game variant $G_\infty(4,2,2,3,3)$ defines a game that has an initial pack of 16 cards arranged on a board of 3 columns. The initial position is identical to the initial position of the finite game $G_{16}(4,2,2,3,3)$. Both initial positions contain 7 hidden cards. The difference in the infinite game is that the stock of 6 cards representing the 2 subsequent deals in the finite game are not retained and take no further part in the play of the game. Instead, for each and every subsequent deal in the infinite game, 3 cards, the value of the column parameter c, are selected from a clone of the original pack of 16 cards. Since the games fixes the ranks of the cards to 4, the suits to 2, and defines an initial multiple of 2, a subsequent deal can result in 3 cards of rank 3 being selected, but only 2 such cards can be of the same suit. In this infinite game variant a subsequent deal of 3 cards of the same rank and suit is not possible.

Finally, we need to modify the definition of winning since it is impossible to exhaust an infinite supply of cards. In an infinite game variant the game is won when the board is cleared. This can result in a fewer number of full blocks being removed than would be required in the finite game counterpart. Conversely, many more full blocks may require removal due to deals introducing cards without regard for the cards already appearing on the board. There is always a minimum number of blocks that must be removed that can be determined by examining the cards, both face up and hidden, present on the board in the current position.

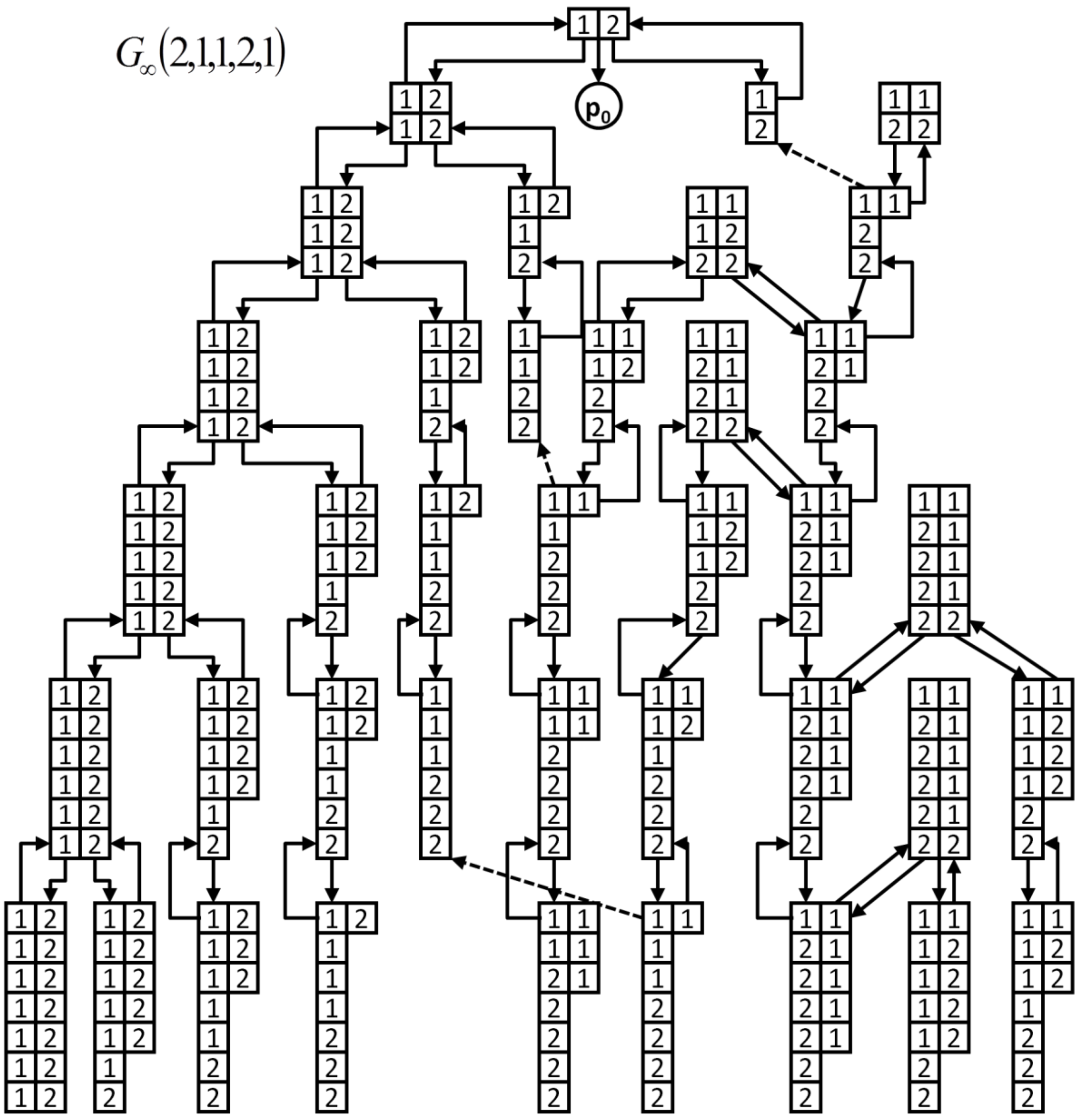
6.2 The Position Graph of an Infinite Spider Game Variant

We now undertake the construction of a position graph for a small infinite Spider game variant. We use the simple game $G_\infty(2,1,1,2,1)$ for our example. This game is played with a pack of 2 cards, a 1 and a 2 of the same suit. There are 2 possible arrangements for each subsequent deal, either a $\langle 1,2 \rangle$ or a $\langle 2,1 \rangle$. These are also the 2 initial arrangements of cards that create the only initial positions of the game. For this game $h = 0$ so that we have $p(n, h + c) = p(n, c) = p(2,2) = 2$. In general any finite or infinite game in which the number of columns is equal to the number of cards, $c = rsm$, is trivially winnable. Every initial arrangement is a winning position. Also, there is no need to consider all $n!$ initial positions to start the game. Due to column symmetry we are free to rearrange columns as their order has no bearing on game movement. Since our initial position has only 1 row all initial positions are equivalent. We will depict the permutation that is lexicographically first in our graph.

The construction of our directed graph of positions applicable to both infinite and finite game variants proceeds as follows. First we create a position vertex for each initial arrangement. We can label these initial positions as $p_1, p_2, \dots, p_{n!}$. In our simple example game we have 2 initial positions, $p_1, p_{2,}\ p_1 = p_2$. We also define the winning position, p_0, representing the empty board. For each position in the graph we consider all available moves that can be made using the rules of the standard game. Recall that a deal is a type of move. When allowed, there are no empty columns preventing a deal, there is always a deal available in an infinite game. Each move applied to a position transforms the position. Graphically, a move is represented by a directed edge in the graph that traverses from one source position to another target position. The target position may already be present in the graph but if it is not, the new target position is added. Now all moves available at the new position are applied. In this manner all positions reachable through a sequence of moves from the initial positions appear in the graph. If the position p_0 appears, then the initial position and all intermediate positions along the route to p_0 are winning positions. If p_0 does not appear in the construction of the graph then all positions constructed are unwinnable. There may be a mix of winnable and unwinnable positions constructed.

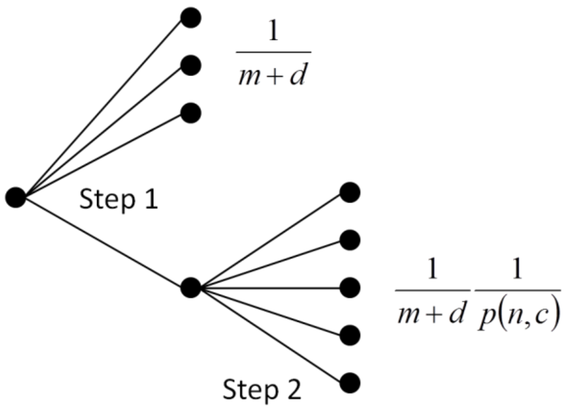
6.3 A Random Walk of an Infinite Position Graph

Returning to the construction of our graph of the simple infinite game $G_\infty(2,1,1,2,1)$ we start with one position vertex, the initial deal $\langle 1,2 \rangle$. Rather than systematically proceeding with the construction of the graph from this one initial position, we instead introduce a random walker placed on this initial vertex. Our random walker chooses, uniformly randomly of course, one of the moves available at the current position. At the first position there are two moves available, either move the 1 onto the 2 or deal. Our random walker flips a fair coin to decide which operation to select. If the move is chosen, a full block is formed, the two cards are removed, and the game is won. The diagram below depicts this move as the edge leading to the circled p_0. Thus, the initial position is a winning position since there is a route to p_0.



The left side of the graph depicts the first few positions based upon an initial G_2 game.

But our random walker may have selected to deal in which case there is an equal probability of dealing a $\langle 1,2\rangle$ or a $\langle 2,1\rangle$. The two resulting positions from these deals are shown as the two positions in the second row of the graph which are connected by the directed edges from the initial position. Upon arriving at the new position our random walker repeats the process. Thus, it is possible for our walker to continue down the left side of the graph indefinitely by a series of selections that result in a deal of $\langle 1,2\rangle$. What is the probability of this occurrence?



Considering random walks of the graphs of positions from all game variants the probability of traversing from one position to another is determined by a 2 step process. First, in step 1 the number of legal moves is enumerated. There are two types of moves to consider, the movement operations m available at the position that move a block from one column to another and a deal operation d if allowed. The probability of selecting any of these available option is $\frac{1}{m+d}$. Note that in a finite game if both m and d equal 0 the position is unwinnable. In the infinite game either a deal is available or a move is available, namely moving a card into the empty column preventing the deal, or both are available, that is, $m + d \geq 1$ in all cases. If the deal operation is selected in step 1 then step 2 selects with equal probability one of the permutations from a clone of the original n cards selected c at a time in the infinite game, $\frac{1}{p(n,c)}$. The finite game deal would select c cards from the cards remaining in the deal stock. The probability of a particular arrangement of cards being dealt at a position in an infinite game is the product of the probabilities of step 1 and step 2, $\frac{1}{(m+d)p(n,c)}$. The diagram shown above depicts the probabilities of the random walker selecting a particular edge emanating from the current position.

Returning to the graph produced by a random walk of the simple infinite game variant $G_\infty(2,1,1,2,1)$ with our random walker starting at the initial position $[1\ 2]$, we see that there is probability $\frac{1}{2}$ that the position p_0 is reached. There is probability $\frac{1}{4}$ that either one of the two positions shown on the next level connected by directed edges are reached instead. The directed edges lead to all of the positions that our walker can traverse and we can calculate the probabilities of reaching each of these positions. With a little work we can see that every position reachable by our random walker from the initial position has a minimum probability of $\frac{1}{2}$ of moving back towards the winning position p_0. The positions in the graph containing one empty column are interesting as there is only 1 movement choice available that must be taken, that is, has probability 1, leading back towards the winning position. Analogous to a random walk on a number line with probability $\frac{1}{2}$ of moving 1 in either direction and starting at 1 our random walker will reach p_0 with probability 1 irrespective of how far down the left side of the position graph the wandering goes (Mosteller, 1965). Since our random walker reaches the winning position with probability 1 in this game variant we conclude that every position reachable by our random walker, no matter how far afield, is a winning position.

As the infinite game variants become larger and the values of n and c increase calculating the probabilities of traversing to the next position becomes increasingly complex. For example, a position taken from an infinite variant of the standard game, $G_\infty(13,2,4,10,6)$, as previously calculated, has an average number of moves greater than 6. The probability of a particular deal occurring would then have a probability of $\frac{1}{7} \cdot \frac{1}{p(104,10)}$. Also, the initial position of the standard game is at least a minimum of 51 moves from the winning position. And so it seems unlikely that a random walk starting from an initial position of the infinite standard game would reach the winning position. Nonetheless, the probability is greater than 0. The calculation of the actual probability for this infinite standard game variant is beyond the scope of this paper.

We offer one final observation on our simple graph of positions. As shown in the graph every position depicted is a winning position because there is a route that can be followed to the winning position p_0. But as we can also see, there are winning positions, those depicted on the right side for the graph, that can be introduced that are not reachable by our random walker starting at the initial position. Also, our walker cannot introduce positions with hidden cards and so no position shown contains any hidden cards. We will discuss these in a later section. We first take a detour to explore an alternative method to a random walk for generating game positions.

6.4 Mapping Positions to the Line, Plane, and 3-space

We now want to explore an alternative to the graph representation of positions. Each position in a graph of positions can be mapped to a unique positive integer coordinate in 3-space. There is a one to one correspondence. We map the ranks of the cards in a position to the x coordinate, the suits of the cards to the y coordinate, and the hidden characteristic of the cards to the z coordinate.

One way to accomplish this is to fill out a given board position into its enclosing rectangle and insert a 0 as a place holder where there is no card present in a column. Now reading the ranks of the cards by row from left to right and bottom to top we produce an integer in a radix that is one more than the number of ranks present in the game variant. We use this value for the x coordinate of the position. Thus, the initial position in our graph, $[1,2]$, is the base 3 integer 12_3 or 5 in base 10. The winning position, the empty board, is equal to 0. There will be gaps in the integers that map to valid positions but each position will have a unique x coordinate integer value.

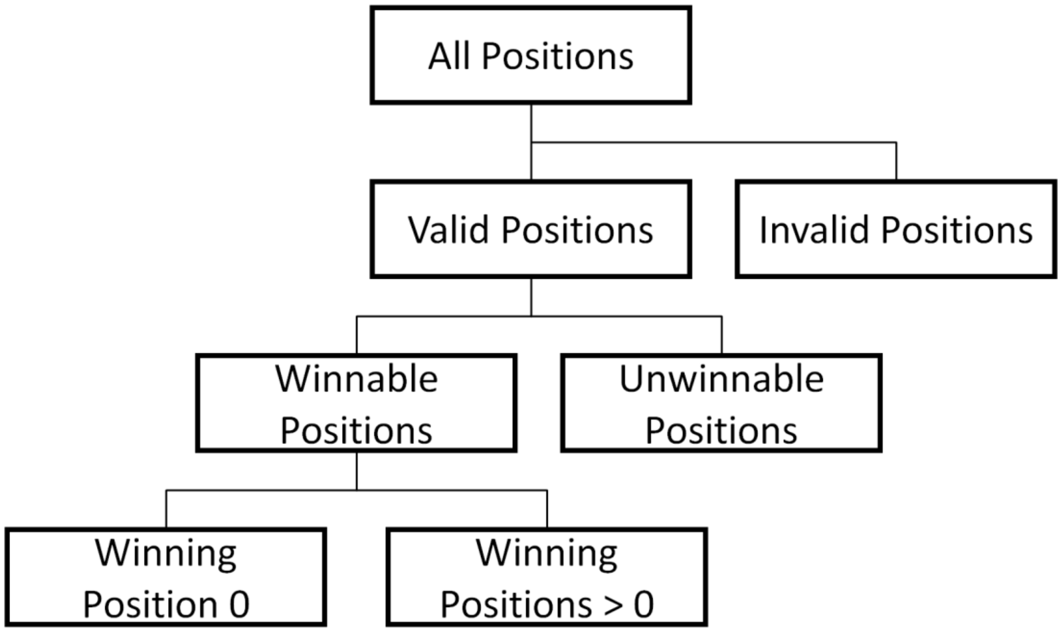
We take a similar approach to determine the y coordinate. This time we consider the number of suits, the value of s, to determine the radix. This time, however, we do not need to account for the missing cards in the columns as this is already reflected in the x axis value. For the y axis we used the zero based index of the suit, $\{0, 1, \dots, s-1\}$, and assume the first suit, the value 0, for an empty card in a column. This has the advantage of making all y values 0 in any game variant with $s = 1$ effectively eliminating the y axis from consideration.

Finally, we map whether each card in a position is face up or hidden with a value of 0 or 1 respectively with empty cards deemed face up. Once again, this eliminates the z axis from consideration when there is a game variant with no hidden cards.

When we consider the graph of our simple game variant $G_\infty(2,1,1,2,1)$ we see that there is only 1 suit and no hidden cards as $rsm = cd$. The positions in the graph map to the line, the non-negative integers of the x axis. The coordinates for our initial position, $[1,2]$, from our graph of positions are $\langle 12_3, 0,0\rangle$ or $\langle 5,0,0\rangle$ in base 10. The position in the second row with one empty column, $\begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}$, is $\langle 2010_3, 0,0\rangle = \langle 57,0,0\rangle$. Looking at the 2 suit game variant, $G_\infty(3,2,1,2,3)$, and the position with 3 cards, $\begin{bmatrix} 3A & 1B \\ 2A & 0 \end{bmatrix}$, the coordinates are $\langle 2031_4, 0001_2, 0\rangle = \langle 133,1,0\rangle$. Since this variant has no hidden cards the positions effectively map to the plane. Taking the similar variant, $G_\infty(3,2,2,2,3)$, which now has hidden cards, and taking the previous position with the card $3A$ hidden instead of face up, we have coordinates $\langle 2031_4, 0001_2, 0010_2\rangle = \langle 133,1,2\rangle$. As we see there is a one-to-one function, an injection, that maps a position from the set of all possible positions of a particular game variant to unique, non-negative, integer coordinates in 3-space. The origin, $\langle 0,0,0\rangle$, is p_0, the winning position.

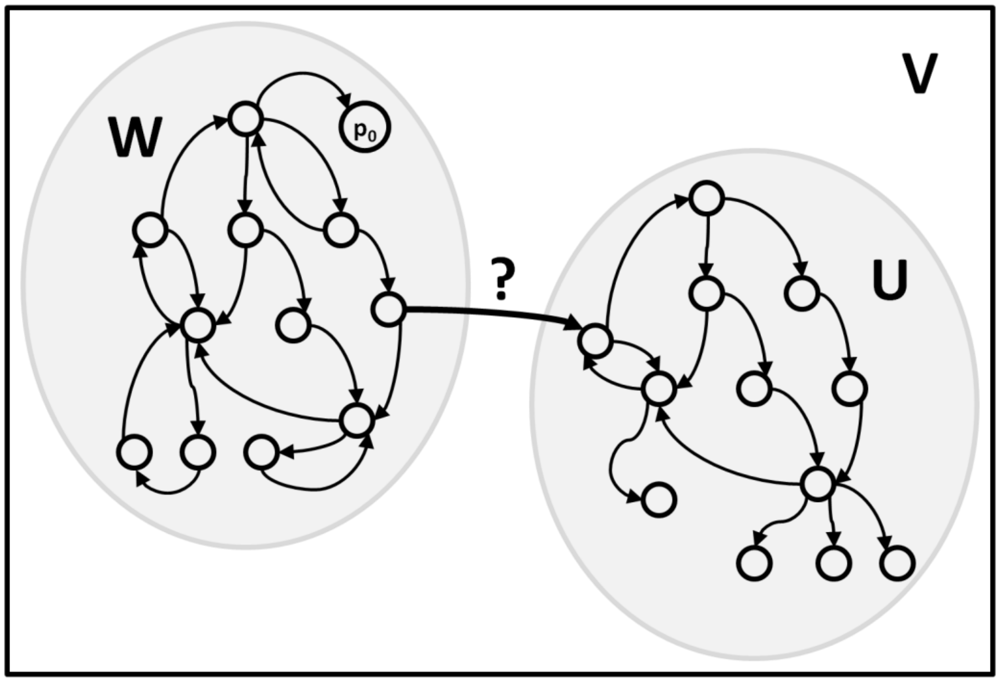
A bijection is possible if we can find a way to map each integer coordinate triple of non-negative 3-space onto a game position. We can do this if we include the subset of invalid game positions in the set of all positions for a given game. A partition of the set of all positions in 3-space would be composed of 4 subsets, the set of one element, $\{\langle 0,0,0\rangle\}$, the winning position, the set of all winnable positions, those that have a route in the graph leading to the winning position, the set of unwinnable positions, those where no route to the winning position exists, and the set of invalid positions, those that cannot occur under the rules of the current game. An example of an invalid position in $G_\infty(2,1,1,2,1)$ would be any position not containing an equal number of 1's and 2's as the pair is either removed or introduced as a unit and the initial position begins with the pair. There are many more invalid positions.

We can now use 3-space to generate new game positions by selecting a random coordinate. If the position determined by the coordinate is valid, we can use that position as the starting point for our random walker. Other transformations of particular points may be possible. Perhaps the distance from a position in 3-space to the origin provides some useful measurement. These are areas of possible future exploration. There are drawbacks, however, of this 3-space representation of positions. One is the sparseness of the number of valid positions. Another is the magnitude of the coordinate values in larger game variants such as the standard game. Both of these may make attempts at analysis using this representation untenable. The diagram below depicts the generalized set and subset hierarchy of game positions.



6.5 Sets of Winnable and Unwinnable Positions

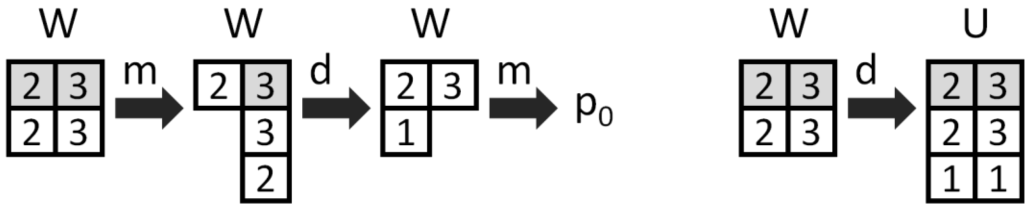
Based on the examination of the position graph of our simple infinite game the following question comes to mind. Starting at a winning position does the random walk in an infinite game generate only winning positions with probability 1? Is it possible to transition to a position that has no way to return back to a previously generated winning position so that every subsequent position traversed is a losing position? We can partition the set V containing all of the valid positions for a particular game instance into two sets, a set W containing all of the winnable positions, that is, those positions where there is a route to the single winning position, and a set U that contains all of the unwinnable positions. We have defined p_0 as the single element that is the winning position. By definition $p_0 \in W$ and $p_0 \notin U$. For every $p_i \in W, i > 0$ there is a route that leads to p_0 . All other valid positions $p_j \in V$ and $p_j \notin W$ are elements of the set of unwinnable positions, $p_j \in U$. The followings set and subset relations hold: $\emptyset \subset W \subseteq V, \emptyset \subseteq U \subset V$.



Before addressing the question for infinite games, let's consider the question for finite games. Does a path from a position in W lead to a position in U for finite games? Consider the initial board position for an unwinnable finite game. The initial position must be a member of set U along with every position reachable from that initial position. For if there were a position that led to a member of W, then all intermediate positions including the starting position would also be members of the set W and not U. We know that unwinnable finite games exist, though not for all finite variants. But if an unwinnable game exists for a finite variant the set U contains the initial position of that game and all positions leading from that initial position. Once again we have restated the original question posed by this thesis. Given an initial arrangement is the initial game position a member of set W or set U?

Consider now an initial position of a finite game that is a member of the set W. Is there a path from a position in W that leads to a position in U when U is not empty? We know that such paths exist in specific cases. The general proof, however, seems elusive. In the specific case a player may make a sequence of moves or deals that cannot be reversed and thus lead to an unwinnable position. For example, consider a standard game starting from an initial winning position where the player blindly deals all of the remaining cards as the first 5 moves. It seems unlikely, but not impossible, that the game can be won from that position. The perfect minimum game presented before presents a counter example. But for an existence proof we need to just demonstrate one example.

Finding a movement that traverses from a winning position to an unwinnable position does not appear possible with the finite games with 2 ranks and 1 suit. There are unwinnable positions in such game variants. But every such game either starts with its initial position in set W or set U. Any game of the form $G_n(2,1,m,2,d)$ with $m > 1$ is unwinnable if the last deal is the pair $\langle 2,2 \rangle$ implying that every position leading to the last position is unwinnable as well. As a side note, in an earlier section we conjectured that a game in which at least 1 block was removed could not be a game unwinnable solely because of rank. But here we have a counterexample. There could have been any number of full blocks removed prior to the last deal and yet the game is unwinnable due to rank alone. We must modify our conjecture to account for the situation where the number of blocks in a game is greater than or equal to the number of columns. In that case, a last deal of all highest ranks must always create an unwinnable game. Returning to the task at hand to find an example of an initial winning position that traverses to an unwinnable position we must look to games of rank 3. Consider the finite game $G_6(3,1,2,2,2)$ with the initial arrangement of cards $\langle 1,1,2,3,2,3 \rangle$. The initial position, shown twice below, has two rows of [2 3] with the top row hidden and with the one deal, $\langle 1,1 \rangle$, remaining.



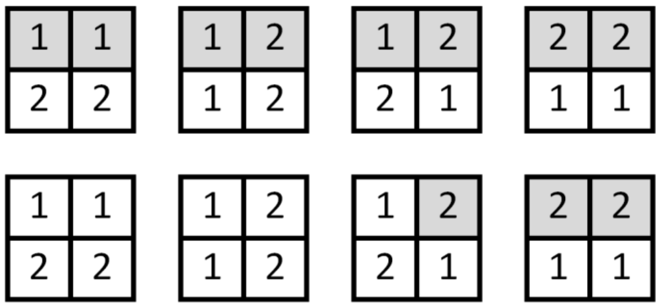
This initial position is a winning position, an element of the set W. Move the 2 onto the 3 uncovering the hidden 2 and then deal the $\langle 1,1 \rangle$. A full block will be formed in the second column and removed. One more move and the game is won. But what if instead of moving the 2 onto the 3 the player deals the $\langle 1,1 \rangle$ first? The resulting position is unwinnable. There is no movement between columns available and no deals remaining. The player has started with a winning position and traversed to an unwinnable position. For finite games a winning position can lead to an unwinnable position.

We now turn our attention to infinite games. First, we will review some of the positions of the simple $G_\infty(2,1,1,2,1)$ previously diagrammed. Though the positions may extend infinitely because there is an infinite number of deals available every position produced by a random walk is a winnable position and our random walker arrives back at p_0 with probability 1. For example, any position where one column ends in 1, and the other in 2 can be extended indefinitely by dealing a matching $[1,2]$. The probability of doing so is $\frac{1}{4}$. That deal can be "undone" by moving the 1 onto the 2 thus returning to the previous position. As we have noted previously the probability of our walker selecting this "undoing" move is $\frac{1}{2}$. The existence of an "undoing" move will be true of infinite games of this type for any number of ranks. Any position in which every column ends in a different rank and the number of columns is equal to the number of ranks can be extended indefinitely in the same manner by matching ranks in a new deal. The deal can be "undone" by making the necessary number of moves, $r-1$, to form the full block and return to the original position. As r increases the probability that our random walker finds this returning sequence of moves decreases but the probability remains greater than 0. Irrespective of our walker wandering back, we still consider the position winnable, a member of the set W, since a route leading back to the winning position exists.

So getting back to our 2 game, we now want to address the two cases where the columns terminate in either two 1's or two 2's. The graph shows no position with columns ending in two 1's. The reason is that a deal always introduces one 1 and one 2 and formation of a full block removes the same. Every position generated will contain the same number of 1's and 2's. So no matter how many pairs of 1's are stacked up, eventually the 1's must meet preceding 2's. But that means a full block would have been formed and the pair removed. Thus, there can be no positions generated from positions in our game where both columns end in 1. Unless, of course, we introduce hidden cards to the positions.

6.6 Hiddenness

What impact does a hidden card have on the play of a game? Let's consider the finite game $G_4(2,1,2,2,1)$. The initial position will have a board of 2 rows, one hidden and 1 active. There are no subsequent deals remaining. Applying all 4! initial arrangements then removing the symmetries we have the 4 following unique arrangements each depicted twice.



The first row shows the hidden row shaded as we have been doing in all of the diagrams to this point. The second row shows the same 4 unique arrangements but this time only shading those cards where the hidden characteristic, which we will dub "hiddenness", has an impact. Hiddenness only matters when it inhibits block formation. When a card is occluded by another it cannot be moved until the occluding card is removed irrespective of whether the occluded card is hidden or not. The one exception is a card that joins a block with the otherwise occluded card. In this case the hiddenness characteristic matters and must be present in order to prevent block formation. Recall that a block of cards of size less than a full block are allowed to move as a unit and so inhibiting the joining of blocks also prevents this block movement. In the second row of the diagram above the hidden 2's prevent the 1's of the row below from forming full blocks which would be implicitly removed. The positions containing the hidden 2's in the second row of initial positions above can only appear intact because of the hidden characteristic of the 2's.

6.7 Graphs of Infinite Games with Differing m Values

One problem lurking underneath the method that we have used for generating valid positions in our graph of positions of an infinite game is that certain positions can only exist when hidden cards are present. Looking back at the graph of positions generated from the initial position of $G_\infty(2,1,1,2,1)$ we see that only one of the 4 initial unique positions from the game $G_\infty(2,1,2,2,1)$ emanates from the initial position. These two game variants differ only in the value of the multiple used, which in an infinite game, determines two aspects of the game, the positions representing the initial board arrangement and the set of cards from which to select cards for subsequent deal operations. In the $m = 2$ game we have 4 unique initial positions, 2 of which have hidden cards with impact as shown in the previous section on hiddenness. There are four choices for subsequent deals, namely$\langle 1,1\rangle\langle 1,2\rangle\langle 2,1\rangle\langle 2,2\rangle$. The right side of the second layer of the position graph shows one of the initial positions$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$. Note that there is no path starting at the original initial position, $[1\ 2]$, that leads to this initial position from the variant $G_\infty(2,1,2,2,1)$. But there is a path that goes in the opposite direction. The 3 dashed edges show the points where the two sections of the position graph join. All 3 join into positions from the original game variant that have 1 empty column. The remaining 2 unique initial deals contain hidden cards that prevent full blocks from being formed and removed. When these are added to the original graph both will have paths to connect into a winning position from the original graph. For the initial position $\begin{bmatrix}1 & 2\\2 & 1\end{bmatrix}$ simply move the 1 onto the 2 which results in position [1 2]. For the initial position $\begin{bmatrix}2 & 2\\1 & 1\end{bmatrix}$ deal a [1,2] followed by a [2,1] which allows the 1 to move onto a 2 uncovering the hidden 2. Another two moves again connects back to the position [1 2]. Thus, all 4 unique initial positions are winning positions.

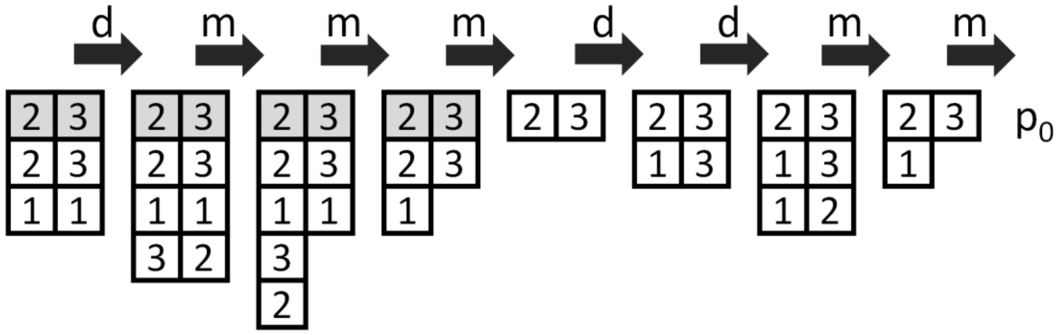
Generalizing from this one case we conjecture that the graph of all positions that can be generated from the initial positions of a game $G_\infty(r, s, m+1, c, d)$ also contains all of the positions that can be generated from the game $G_\infty(r, s, m, c, d)$. If the conjecture can be proven true then we can conclude through induction that if every position generated from a game variant $G_\infty(r, s, 1, c, d)$ is a winning position, then all possible initial positions from a $G_\infty(r, s, m, c, d)$ game are also winning positions. The implication is that all positions generated from the initial positions of an infinite game are winning positions.

	cd / rsm	**2,1**	**2,2**
G$_4$	**2,1,2**	16	20
G$_\propto$	**2,1,2**	24	24

Number of winning initial arrangements of the 4! possible.

The chart above shows a comparison of the number of winning initial arrangements between 2 infinite game variants and their finite counterparts normalized to all 4! possible initial arrangements taking each card to be distinguishable irrespective of value. Since each card in the game appears twice there are only 6 unique arrangements. Of the 6 arrangements remaining 2 pairs differ only in the order of the columns which results in the 4 initial positions previously shown. Where there are unwinnable positions in the finite variants, namely the 4 ways each to arrange the assumed distinguishable 1's and 2's in the positions $\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ and $\begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$ with top row hidden in the 1 deal game, and the 4 arrangements of $\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ in the 2 deal game, there are none in the infinite variants. Our final conjecture of this section that will be the topic for future analysis is that for all infinite game variants with $r, c > 1$ all valid positions are winning positions. The set U is empty.

Going back to the unwinnable final position of our 3 rank game discussed in a prior section and shown below as the leftmost position, the same position appearing in its infinite game counterpart, $G_\infty(3,1,2,2,2)$, is a winning position. The winning position p_0 is reached by dealing $\langle 3,2 \rangle$, making all possible block movements, then dealing in succession $\langle 1,3 \rangle$ and $\langle 1,2 \rangle$. Two block movements then clear the board.



6.8 Movement Operator

Up until now we have been creating the positions of infinite games by constructing them from initial positions of finite games or by applying a movement operation or deal to an existing position. We are viewing these unique game positions as vertices in a directed graph.

We can place a random walker on any position in the graph and let the walker travel about generating new positions as needed until the wanderer arrives at the winning position. Though the random walk may continue indefinitely down some path leading away from the winning position we now want to allow the wandering to continue in the case where the walker arrives at the winning position as well. To this end we extend and abstract the movement operation between positions.

Each position is an element of the set V of all valid game positions for a particular game. We defined a partition of V into a set W of winnable positions, and a set U of unwinnable positions. The edges of the graph of positions represent the application of a valid game movement operation from one position leading to another position. As described previously each position has a small, finite set of allowable moves that can be applied, the 4 moves types and a deal when available. Let's define an operator, +, that applies a valid movement operation to a position and produces a new position. We will call this the movement operator. For the two positions $p_i, p_j \in V$ and an allowable move from the set of M moves available to the position p_i, $m_n \in M_{p_i}$, we have $p_i + m_n = p_j$. Now we can rephrase our question as to whether all positions in a game are winning positions. Is the operator + closed under the set W? Is the operator + closed under the set U? By definition the latter must be true. No winning position can appear in the set U and so any application of the movement operator must result in another unwinnable position. For finite game variants we have demonstrated that under set W the operation is not closed. But the story is different for infinite games. We have conjectured that for infinite game variants the movement operation is closed under set W. That is, movement from a winning position always leads to another winning position.

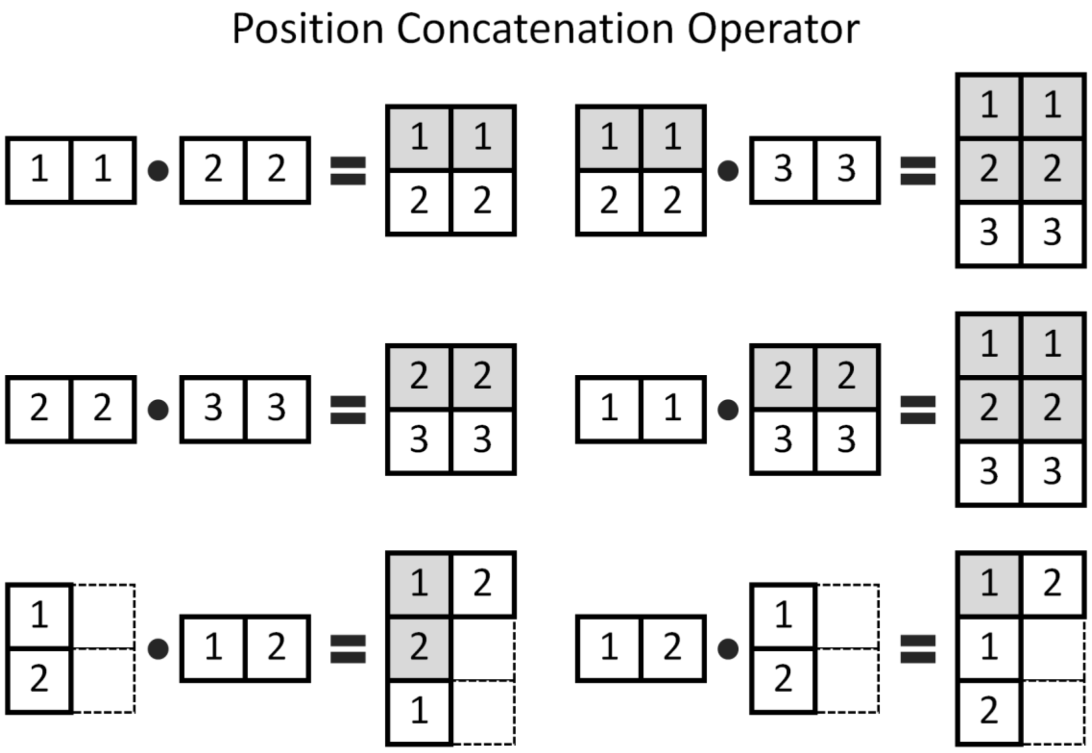
Thus, we have restated our previous conjecture that in the infinite game variants, the set U is empty. The operator + is closed under the set W.

In the finite game it is possible that a position has no allowable moves available. The stock of deals is exhausted and there are no other moves available between columns. In the infinite game, however, there is always a deal available. We use this fact to define movement from the winning position so that $p_0 + m_n = p_j$where p_j represents a valid initial deal in the particular game variant. This would allow our random walker to continue the journey after arriving at the winning position. Another possibility is to define any movement from the winning position to just return the winning position so that $p_0 + m_n = p_0$. But the former is preferred. We can now place our random walker on the winning position, generate a random initial position based on the game parameters, and let our traveler wander about and perhaps eventually arrive back at the winning position. The wandering now extends indefinitely in all cases perhaps generating an infinite number of new winning positions.

6.9 A Position Concatenation Operator for Generating New Positions

Before ending this section we propose one more method of constructing new winning positions from the set of winning positions in an infinite game variant. Perhaps our weary wanderer has wandered down some infinite trail never to return again. We need an alternative to create new positions. Also, only the initial positions from the finite game counterparts of the infinite game can create positions with hidden cards. We need a method to create positions with hidden cards that are not initial positions.

For this purpose we define a binary operator that takes two positions and produces a third, that is, $p_i \cdot p_j = p_k$. The operator works in this manner. The position p_j is concatenated below p_i. Every card in the corresponding column of p_i that is above a non-empty column in p_j is made hidden resulting in a new position p_k. Making the cards of the first operand hidden produces additional hidden cards and inhibits any block formation, and more importantly, any block removal, that might occur with the concatenation of non-empty columns. A few examples are illustrated below.

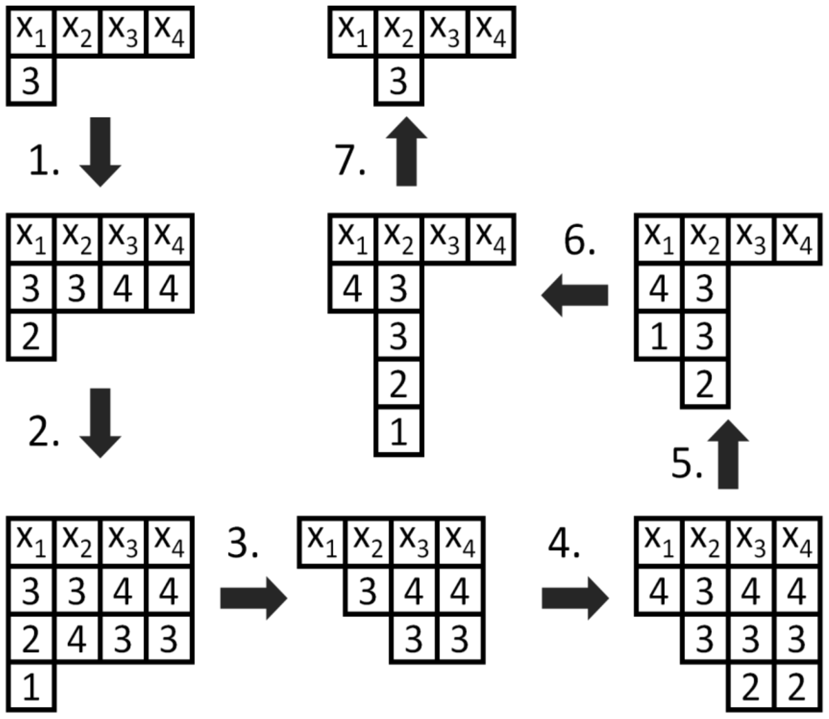


The operation is associative as $p_i \cdot (p_j \cdot p_k) = (p_i \cdot p_j) \cdot p_k$ but not commutative. The winning position, p_0, serves as the identity element so that $p_i \cdot p_0 = p_0 \cdot p_i = p_i$. The operation is closed under the set V of valid positions and for the infinite game closed under the set W of winning positions, though this is still only a strong conjecture awaiting a formal proof. But for the want of an inverse the position operator would form a group (Burn, 1985).

A formal proof for closure under the set W might proceed as follows. First, we consider removing the cards of the second operand from the combined position. Then, we consider removing the cards that remain from the first operand. Since the second operand, the lower portion of the newly constructed position, is unaltered by the operation from its original winning position then there is a sequence of moves that transforms that position to p_0. There are two cases to consider. If the original sequence of moves did not use a movement operation into an empty column, then following the same sequence of moves is possible resulting in the removal of the cards of the second operand leaving the cards of the first operand. If the sequence of moves exploited an empty column, however, then the same sequence of moves cannot be followed. An alternative sequence must be employed. Let's assume for now that such an alternative exists using the additional deals available in the infinite game. Of course we will need to prove this as well. After the second operand has been cleared the bottom cards of the first operand position will be exposed and active but the remaining cards that may have been previously face up are now hidden. If there are no newly hidden cards that inhibit block formation then the same sequence of moves that transformed the original first operand into p_0 can be used. Thus, in this case the new position is also a winning position. But in the case where a newly hidden card has an impact, the same sequence of moves would not apply. Once again we have special cases that thwart a simple proof.

If there is a proof for the following lemma, and we conjecture that there is, then the obstacles in the way of our previous proof may be eliminated. The lemma is that in the infinite game variants there is always a sequence of deals and moves that in effect moves one card from one source column to a different target column while leaving the remaining columns unaltered. The lemma states "in effect" because what actually occurs is that the source card is removed from the source column through the formation of a full block while the equivalent card is added to the target column via a deal. Perhaps a specific example will help to clarify.

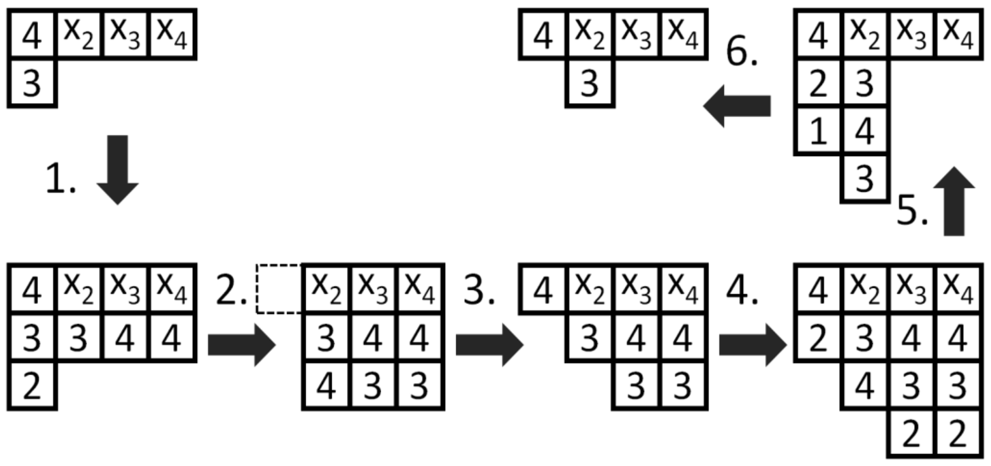
Move One Card to a New Column



The example above is using an infinite game variant with 4 ranks and 4 columns. The objective is to move the 3 that is below the card x_1 to the location below card x_2. The remaining 2 columns must remain unaffected. A sequence of 7 steps accomplishes this. Step one deals ⟨2,3,4,4⟩. This step places a new 3 in its target position. The 2 columns not participating in the move receive the highest ranking card so that no unintended block formation can result with the cards x_3 and x_4. This also serves as the starting point for full blocks that will be formed and subsequently removed. All cards introduced by the deals have the same suit as the card that is moving. Step 2 deals ⟨1,4,3,3⟩. Step 3 moves the block formed in column 1 to column 2 creating a full block that is immediately removed. Step 4 deals ⟨4,3,2,2⟩. Step 5 deals ⟨1,2,1,1⟩. This completes full blocks in the 2 non participating columns clearing the cards introduced by the previous deals. Step 6 moves the 1 onto the 2 and then step 7 completes the full block by moving the block from column 2 to column 1. The net result is that the 3 has moved from column 1 to column 2.

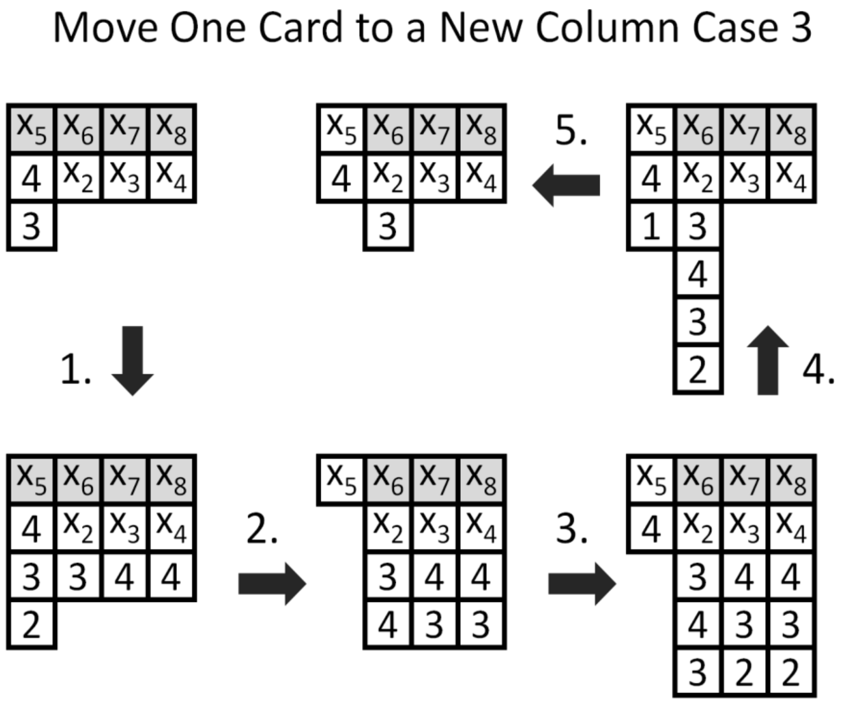
But there are special cases to consider in the lemma as well. In the example above, if the card identified as x_1 is not hidden, has rank 4, and is the same suit as the 3 being moved, then step 2 of the move sequence would create a full block removing both the 4 and 3. We therefore need a special case to handle this situation as depicted below.

Move One Card to a New Column Case 2



In case 2 above step 2 removes both the 4 and the 3 from the first column which is now empty. No deal can be made from this position. Instead, step 3 moves the 4 from the second column to the first replacing the 4 removed in step 2. Now deals of ⟨2,4,2,2⟩ and ⟨1,3,1,1⟩

and one final move to complete the full block in column 2 yields the desired result. But there still remains one final special case to cover.



Similar to case 2 where a partial block is formed with an unhidden 4 of the same suit, case 3 has an additional row of hidden cards so that column 1 does not become empty as in case 2. Instead the card x_5 is exposed and becomes unhidden in step 2 when the full block formed in column 1 is removed. Two more deals and a final move return us to the final position which differs from the original position only at the card x_5 which is now face up. But since that card is occluded by a 4, the highest ranking card in this game variant, the hiddenness of x_5 has no significance. Thus, the original position and the final position are equivalent. Any sequence of moves that can be applied to the original position can also be applied to the final position and will result in equivalent positions at the end of the move sequence.

We are confident that a formal proof for the lemma, most likely employing induction, can be developed though, as the example above demonstrates, there are many special cases that must be covered. The parity of the number of ranks and columns must be considered and care must be taken in situations where unintended full block formation may occur as we saw above. But if such a proof for the lemma is realized, then our proof that all positions produced from winning positions by our position concatenation operator can be proven as well. Move sequences that would have required moves into empty columns can be replaced by the sequence of moves described by the lemma. This allows the second operands to be removed from the concatenation result. Then to revert the hidden cards that were altered in the first operand position to their original state, we can use the lemma to move cards back and forth between columns to return the necessary cards to their original face up orientation.

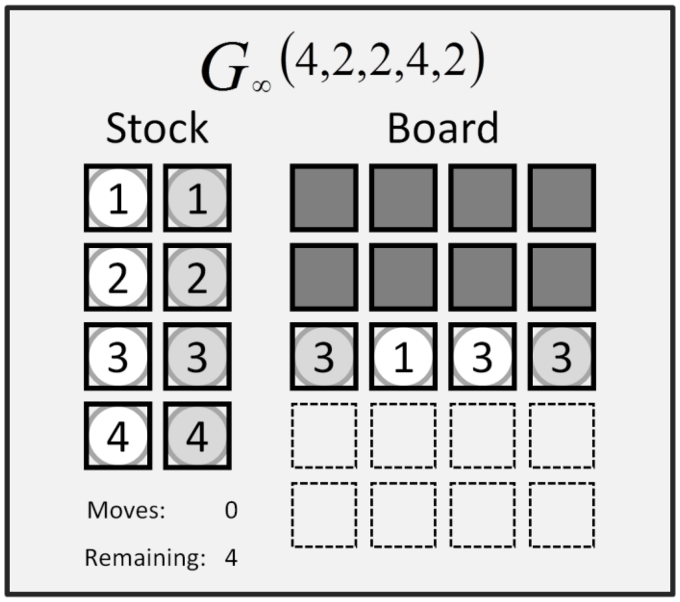
So now that we are confident that we can take any two winning positions and produce a new provably winning position we just need to add some positions demonstrated to be winnable to our set W of winning positions for a particular infinite game variant. To ensure an adequate supply of construction material the finite game counterpart for the infinite game should specify a value for the multiple that is greater than the number of columns, $m \geq c$. By doing so each deal in the infinite game can consist of any of the cards in the game including a deal of all of the same card. For our simple 2 rank game previously graphed, $G_{\infty}(2,1,2,2,2)$ does the trick. Setting the number of deals to a value that makes $cd = rsm$ or more simply because we have set $m = c, d = rs$, results in a single row of cards that can be any of the $(rs)^c$ possibilities for a deal. In this case there are 4, $\langle 1,1\rangle\langle 1,2\rangle\langle 2,1\rangle\langle 2,2\rangle$. Our 3 rank variant, $G_{\infty}(3,1,2,2,3)$, has 9 possible deals. An infinite version of the standard game,

$G_{\infty}(13,4,10,10,52)$, would have 52^{10} different deals available. Once these initial positions are demonstrated winnable, no simple feat for larger variants, then any positions that result from the application of the position concatenation operator are also winning positions.

7. Suggestion for a new type of game.

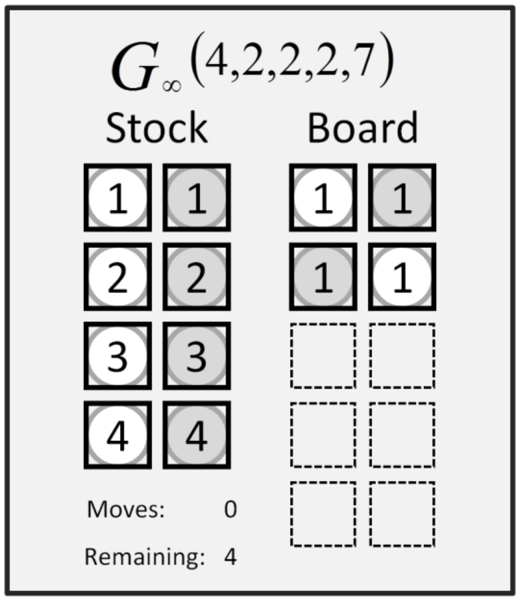
Now that we have explored the concept of an infinite Spider Solitaire game variant can we use this idea to formulate a game variant that is playable by a human participant? Assuming that our conjecture that every position is an infinite game with $c > 1$ is a winnable position then we should be able to pose any initial arrangement of cards to a player and have the player find a winning solution dealing new cards as often as necessary. But the human player likely does not have an infinite amount of time waiting for a fortuitous deal to devote to the game challenge as did our random walker. So rather than having a random arrangement of new cards introduced by a deal, what if we put the player in control and allowed the player to select the cards for the deal in the specific arrangement desired? To accommodate varying numbers of suits and ranks the new game would use numbered colored tiles rather than the standard set of playing cards. The tile ranks would be numbered 1 through r, and there would be a unique color or pattern for each s value.

The play of the game would unfold as follows. First, as in a finite game, an initial arrangement of cards, which we are now calling tiles, would be mapped to the board configuration. The player would see the active tiles at the bottom of the rows of hidden tiles. Instead of a reserve stock of deals, however, there would be a grid of the available tiles used in the game. This would be the full complement of tiles, one for each rank and suit. The player could then either make a move in accordance with the standard rules using the active tiles, or deal by selecting tiles from the grid and placing one at the bottom of each column. If desired the player could add the same tile to every column. An initial game screen might look like this.



The player would proceed until the board is cleared and receive a score equal to the number of moves taken to do so.

To give the reader a feel for how the game might play we pose the following puzzle based on the game $G_\infty(4,2,2,2,7)$. The game contains 2 each of 2 blocks of 4 arranged in 2 columns. The game's initial arrangement starts with 4 tiles selected from the initial pack of 16 arranged into 2 rows, 1 hidden row, and 1 active row representing deal 1. The remaining 12 tiles from the 6 subsequent deals are discarded. Normally a player would not be able to see the suit or rank of the hidden tiles. Perhaps revealing either the rank or color of the hidden tiles could be game options or hints made available to the player. Here we will use filled and unfilled circles to identify the 2 colors used for the tiles.



8. Are We There Yet?

So how many games of Spider Solitaire are winnable? We have taken a scenic route more closely resembling a random walk than an expressway to a definitive answer. The answer to the question still appears quite elusive. But we have set up a framework that allows for continued exploration. We summarize what we have learned along the way here.

8.1 Summary of Analysis

Most games of the standard Spider Solitaire game are winnable. Our best estimate from our simulations is in line with the one other study found and appears to be a ratio greater than 0.999 of all games. Our standard variant is also known as the "hard" game. Our data analysis shows that increasing the number of suits used in a game makes the game harder in the sense that there are more unwinnable games. It also makes the game harder in the sense that a human player will have more difficulty in finding a winning solution even when the game is winnable. Thus, based on our evidence from our simulations we are confident in our conjecture that the ratio of winnable "medium" games is greater than the ratio of "hard" winnable games, and that the ratio of "easy" winnable games is greater still. But unwinnable games are rare. We also have some degree of confidence that our random samples of game instances are representative of the vast population of unique games. We calculated expectations for rank distributions and the number of deals with no two adjacent ranks. Our random sample of games met these expectations.

We defined 5 parameters, ranks, suits, multiple, columns, and deals, that we used to specify spider game variants. This allowed the simulation and data analysis of the standard games as well as smaller games. We looked at the effects of varying the individual parameters to see if we could formulate theorems or projections of the parameter's effect on the ratio of winnable games. But the parameters are not independent. Each parameter changes the ratio of hidden cards to total cards which is an important factor as to whether a game is winnable or not. Nonetheless, we made conjectures about the parameters and started collecting data and evidence to affirm or reject those conjectures. But there are no proofs so far except for our constrained deal theorem. Our one theorem states that increasing the number of deals increases, or more strictly, does not decrease the number of winning games as long as the constraint that $d < r$ applies. In the case of the standard games this constraint is met. Thus, increasing the number of deals in a standard game from 6 to 7 will mean that more games are winnable. Conversely, reducing the number of deals will make more games unwinnable. Again, most games are already winnable. But this deal theorem gave us the notion that all games are winnable in the infinite game variants. In the infinite game there is an infinite number of deals and so the probability of winning should approach 1 based on this theorem.

Our generalized game parameters allowed us to define and explore games with an infinite number of cards and deals. This provided fertile ground for exploration. We tried to lay a foundation for a proof that all positions of infinite game variants are winnable positions. Work continues on this formal proof. Based on the conjecture that all infinite game positions are winnable we proposed a new infinite game variant playable by a human participant.

8.2 Parting Advice for the Standard Game Player

We have uncovered no magic formula for finding the sequence of moves that wins a game of standard Spider Solitaire. The good news is that there is likely a winning sequence of moves available. The bad news is that sequence is difficult to find and at anytime, unbeknownst to the player, a move might transition from a winnable position to an unwinnable position. The player can adopt the strategy embodied in the simulation program's fitness function. Prefer moves that form blocks and moves that uncover hidden cards. Also, try to empty a column to create additional movement options for forming larger blocks. Finally, keep in mind that most implementations of Spider Solitaire implement an "undo" function. Feel free to use it.

9. References

AARP.org. (2012). *Spider Solitaire*. Retrieved 02 12, 2012, from www.aarp.org: http://games.aarp.org/games/spider-solitaire.aspx

Alex at Tranzoa Company. (2005). *PLSpider*. Retrieved 12 29, 2011, from http://www.tranzoa.net/~alex/plspider.htm

Andrews, G. E., & Eriksson, K. (2004). *Integer Partitions.* Cambridge, UK: Cambridge University Press, p. 1-35.

Briggs, W. (2005). *Ants, Bikes, & Clocks.* Philadelphia, PA: Siam, chapter 2.

Burn, R. P. (1985). *Groups A Path To Geometry.* Cambridge, U.K.: Cambridge University Press, chapters 1,2,6.

Chaitin, G. (2005). *Meta Math! The Quest for Omega.* New York, New York: Pantheon Books.

Diaconis, P. (1999). *The Mathematics of Solitaire*. Retrieved from www.researchchannel.org: http://www.researchchannel.org/index.asp

docs.oracle.com. (2012). *Java 7 API Class Random*. (Oracle.com) Retrieved 02 26, 2012, from Java 7 API: http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

Freund, J. E., & Perles, B. M. (2007). *Modern Elementary Statistics.* Upper Saddle River, NJ: Pearson Prentice Hall, chapter 10.

Graham, R. L., Knuth, D. E., & Patashnik, O. (2011). *Concrete Mathematics* (Second ed.). Westford, MA: Addison-Wesley, chapters 1,2 5.

Knuth, D. E. (1981). *The Art of Computer Programming* (2nd ed., Vol. 2). Reading, Massachusetts: Addison-Wesley, chapter 3.

Knuth, D. E. (2005). *The Art Of Computer Programming Generating All Combinations and Partitions* (Vol. 4). Upper-Saddle River, NJ: Addison-Wesley.

Knuth, D. E. (2005). *The Art of Computer Programming Generating All Tuples and Permutations* (Vol. 4). Upper Saddle River, NJ: Addison-Wesley.

Microsoft Corporation. (n.d.). Spider Solitaire Version 6.0. Redmond, WA.

Mosteller, F. (1965). *Fifty Challenging Problems in Probability with Solutions.* New York: Dover Publications, Inc. Problem 35.

OEIS.org. (2012). *The On-Line Encyclopedia of Integer Sequences A002620*. Retrieved 02 19, 2012, from The On-Line Encyclopedia of Integer Sequences: https://oeis.org/A002620

OEIS.org. (2012). *The On-Line Encyclopedia of Integer Sequences A006218*. Retrieved 01 16, 2012, from The On-Line Encyclopedia of Integer Sequences: https://oeis.org/A006218

OEIS.org. (2012). *The On-Line Encyclopedia of Integer Sequences A007425*. Retrieved 01 16, 2012, from The On-Line Encyclopedia of Integer Sequences: https://oeis.org/A007425

Polya, G. (2004). *How to Solve It A New Aspect of Mathematical Method* (Expanded Princeton Science Library Edition ed.). Princeton, NJ: Princeton University Press.

Polya, G. (1990). *Mathematics and Plausible Reasoning (Twelth printing and first Princeton Paperback edition).* Princeton, New Jersey: Princeton University Press

Stanley, R. P. (1986). *Enumerative Combinatorics* (Vol. Volume 1). Belmont, CA: Wadsworth & Brooks/Cole Advanced Books & Software, chapter 1.

Winkler, P. (2010, February). Breaking Chocolate Bars. *Communications of the ACM , 53* (2), p. 120.