The Mathematics of Perfect *n*-Shuffles

Shawn Peters

Faculty Advisor: Dr. Carl Schoen

Department of Mathematics, University of Wisconsin- Eau Claire

Introduction

- A perfect two-shuffle (Faro Shuffle) is achieved by splitting a deck of cards into two sub-decks of equal magnitude. The cards are then interwoven so any two consecutive cards from either sub-deck are separated by exactly one card from the other sub-deck.[1]
- In 1983, Dr. Persi Diaconis proved that properly executed perfect two-shuffles can place the top card in any position *p* in the deck by representing *p* in binary and performing an in-shuffle for each 1 and an out-shuffle for each 0 in the binary representation.[2]
- There are eight people in the world who can perform a perfect two-shuffle of a regular deck of cards, Dr. Diaconis among them.
- Perfect shuffles can be performed for a deck with any number of cards, provided the number of cards in the deck is an integer multiple of the number of subdecks that will be formed.[3]
- Perfect *n*-shuffles are executed similarly to a perfect two-shuffle, except the deck is divided into *n* sub-decks rather than two. For example, in **Figure 1**, a perfect three-shuffle can be performed by taking a card from each deck and following the permutation (123).
- Performing multiple perfect shuffles on any qualified deck will eventually result in the deck being restored to its original state.[1,2]
- Perfect *n*-shuffles can be extended to pulling more than one card at a time from each subdeck. This means that instead of interweaving (pulling) one card from each subdeck, *u* cards can instead be pulled.

Objectives

- Through use of a specialized computer program that simulates perfect *n*-shuffles and subsequently through use of modular arithmetic, to derive an equation to predict the position of the p_0-th card after *k* shuffles (**Equation 1**).
- Use the data and abstract algebraic techniques to derive an equation which produces the number of shuffles necessary to return the deck to its original un-shuffled state (**Equation 2**).
- Extend the above findings to derive equations to determine the position of any card p_0 in the deck when pulling two cards (**Equation 3**).
- Analyze the movement of the cards when the deck does not divide evenly into two subdecks when pulling *u* cards.
  - For example, what happens when a deck of size *c* =10 is split into two subdecks and a two-card pull perfect shuffle is performed? This type of shuffle will leave one card at the bottom of each pile, which breaks the pattern of pulling two cards.
  - Fully develop and investigate two ways to approach this problem:
    - *Case 1*: splitting the deck into a larger deck and a smaller deck, so that the left deck (from the top of the pile) is larger than the right deck by *u* cards (**Equation 4**).
    - *Case 2*: splitting the deck evenly in half, and pulling *u* cards (**Figures 2, 3a, 3b**).
- Combine all results into one fully generalized equation for decks of any size (**Equation 5**).

Definition of Variables

deck size $c = nd$ where $n,d \in \mathbf{N}$, pulling *u* cards, original card position p_0; card position *p*; number of shuffles *k*

Results

Figure 1

- **Equation 1**

$$n^k p_0 - (n^k - 1) \bmod (c-1) \equiv p$$

- **Equation 2**

$$n^k \equiv 1 (\bmod\ c-1)$$

- To determine where a card in position *d* will be after *k* shuffles, choose the card in position p_0, and **Equation 1** will give its new position *p* after *k* shuffles. Notice that the cards are cycling modulo $c-1$; this happens because the first and last card in the deck stay fixed (out-shuffle). This means that when the cards move through the end of the deck, no card can ever take position $p = c$, and thus the cards will cycle mod($c-1$). **Equation 2** is a simplified version of **Equation 1,** solved for when $p_0 = p$; therefore, solving **Equation 2** for *k* will be solving for the number of shuffles required to return the deck to its original state.
- Pulling two cards will exhibit the same movement as pulling one, but the cards will cycle modulo $c-2$. This behavior is generalized in **Equation 3**.
- **Equation 3**

$$n^k p_0 - (n^k - 1) \bmod (c-2) \equiv p$$

- When pulling *u* cards from two subdecks of odd magnitude, there are two possible approaches to model the behavior of the cards' movement:
- In *Case 1*, simply split the deck such that there are *u* more cards in the left subdeck (from the top of the pile) than in the right subdeck. When the deck is split into unequal subdecks, the last *u* cards are pulled into the shuffling process and the cards can actually take the last *u* positions in the deck, and thus the cards no longer cycle modulo $c-u$, instead simply cycling modulo *c* as seen in **Equation 4**.
- **Equation 4**

$$2^k p_0 - (2^k - 1) \bmod (c) \equiv p$$

- *Case 2* is more challenging and must be handled differently than *Case 1*. Consider a deck of size $c = 2(2d-1)$. This deck will split into two subdecks of magnitude $(2d-1)$, which gives each subdeck an odd magnitude. This will cause a problem at the end of the deck when pulling two cards because there will only be one card left in each subdeck, breaking the pattern of pulling two. If the cards are just picked up, the permutations become complicated and can no longer be expressed using modular-arithmetic equations. Instead, matrices can be used to plot the passage of the cards through the deck through the following steps:

1. Construct a *c* x 1 matrix with entries x_{p0} where each x_r represents the original position of the *r*th card in the deck (**Figure 2**).
2. Cut the matrix in half and form two new matrices, and rotate the second matrix 90 degrees. The first matrix will contain the first half of the deck and be a $c/_2$ x 1 matrix, and the second matrix will contain the remaining cards in a 1 x $c/_2$ matrix.
3. Determine the number of cards *u* that will be pulled during the shuffle, and between each set of *u* elements in the matrix, insert *u* 1's (**Figure 3a**).
4. Multiply the matrices and drop any remaining 1's (**Figure 3b**).

Figure 2

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{c-1} \\ x_c \end{bmatrix}$$

Figure 3a / **Figure 3b**

$$\begin{bmatrix} x_1 \\ x_2 \\ 1 \\ 1 \\ x_3 \\ x_4 \\ \vdots \\ 1 \\ 1 \\ x_{\frac{c}{2}} \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & x_{\frac{c}{2}+1} & x_{\frac{c}{2}+2} & \cdots & 1 & 1 & x_c \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{c-2} \\ x_{c-1} \\ 1 \\ x_c \end{bmatrix} \Longrightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{c-2} \\ x_{c-1} \\ x_c \end{bmatrix}$$

- Putting all of the different situations together, a generalized equation for decks and subdecks of any size may be established (**Equation 5**).
- **Equation 5**

$$n^k p_0 - (n^k - 1) \bmod (c-u) \equiv p$$

Applications

- Perfect shuffles have many applications both related and unrelated to playing cards.
- This research is beneficial for developing counter measures for cheating in gambling. Alternatively, with a solid knowledge of the mathematics of these perfect shuffles and a properly constructed deck, it could easily be used to gain an upper hand as the dealer in games such as poker.
- The permutations of the cards can be used to create patterns for nearly anything. By assigning a color or a picture to each card in the deck, performing repeated perfect shuffles can produce various designs.
- Messages could be coded by assigning a letter to each card and sending it through a certain number of permutations, and it could be sent to the intended recipient with a number indicating the remaining shuffles required to return the deck to its original state, thus decoding the message.
- Assignment of different tones to each card could potentially produce interesting musical patterns when subjected to perfect shuffles.

Discussion

- Perfect two-shuffles can be performed in two different ways depending on which sub-deck is chosen to be on top for the shuffle.[1,2] In this manner, perfect *n*-shuffles can be executed in *n*! different ways, all of which would result in a different number of shuffles to return the deck to its original state.
  - I.e. (123…*c*), (*c*…312), (1…*c*…32), etc.
- The program I used did not give the option to change the permutation of the sub-decks for the shuffle, and this is something that could be investigated further.
- The method presented in *Case 2* would be very inefficient to undertake by hand, but a simple computer program could easily perform that type of shuffle. It is also important to note that the method in *Case 2* can be used for normal perfect shuffles as well; since the cards would interweave perfectly, there would be no need to drop any 1's in the last step.

References

1. Diaconis, P.; R.L. Graham, and W.M. Kantor (1983). "The mathematics of perfect shuffles". *Advances in Applied Mathematics* **4** (2): 175–196. doi:10.1016/0196- 8858(83)90009-X. http://www-stat.stanford.edu/~cgates/PERSI/papers/83_05_shuffles.pdf.
2. Kolata, Gina. "Perfect Shuffles and Their Relation to Math." *JSTOR: Mathematics Magazine*. American Association for the Advancement of Science, 30 Apr. 1982. Web. 14 Jan. 2010. http://www.jstor.org/stable/1688284?seq=1.
3. Mevendoff, Steve, and Kent Morrison. "Groups of Perfect Shuffles." *Mathematics Magazine* Feb. 1987: 3-14. *JSTOR: Mathematics Magazine*. Mathematical Association of America. Web. 15 Jan. 2010. <http://www.jstor.org/stable/2690131?seq=1>.

Acknowledgements

- This research was sponsored, in part, by University of Wisconsin-Eau Claire Differential Tuition and a Faculty-Student Collaborative Research Grant.
- Peter Metzger, creator of http://shawn.getzger.com