Randomizing

George Daccache, Linghai Liu, Zhiyuan Zhou*

August 28, 2020

Advisor: Giorgio Cocomello

1 Introduction

This work explores the idea of "randomizing" in two contexts. Randomizing is the process of modifying a system step by step until each state in the state space is equally likely to appear with one modification. It means that a "random" state has such mathematical unpredictability that it is independent of its initial state.

The problem of randomizing is extremely important in both the mathematical world and the real one. Randomization is widely used in sampling works such as sampling for opinion polls and statistical sampling in quality control systems. It is also widely used in many computational algorithms, such as the Monte Carlo Method and the genetic algorithms, let alone its applications in medicine, sports, politics, and so on. Therefore, it is of great interest to dive deep into the theory behind the process of randomisation.

The remaining sections of this paper will be organized as follows: Section 2 will attempt to solve the process of randomizing in the context of a random walk on a circle, using knowledge from Information Theory and Markov Chains. Computer simulation is also used to form and test out conjectures. Section 3 will address the the problem in the context of shuffling cards, which can be seen as a more complicated random walk problem.

2 Problem 1: Random Walk on a Circle

2.1 Introduction: the problem

Take n points arranged on a circle. Suppose that a particle starts at a fixed position (one of our points), and then moves right or left each with probability 1/4, or otherwise stays at the same point. After a sufficiently long time, the position of the particle is essentially random, which means that it is approximately equally likely to be at any place. What is this time, and how is this equilibrium approached? What if one changes the basic probabilities?

*Authors contributed equally to this work

2.2 Randomizing through Entropy

One direction our team took was to translate the problem through the lens of information theory. Our key insight was that to analyze this system, instead of sampling the circle at different numbers of steps, it is equivalent to sample multiple circles once after the same number of steps.

To set up the problem, we consider a circle with n nodes. We pick a node, label it node 1, and label subsequent nodes by moving counterclockwise. A ball is set at node 1. The state space describes all the positions the ball can occupy, with 0 indicating no ball in that position and 1 indicating its presence. Thus we have: $S = \{(1,0,0,...,0),(0,1,0,...,0),...,(0,0,0,...,1)\}$. We have n possible states. We define n Bernoulli indicator random variables, and write $X_i = 1$ to denote a circle in state i (so the ball is on node i), and $X_i = 0$ otherwise.

Now, consider n_0 statistically identical circles, whose states are described by n_0 random variables $\{X_1, X_2, ..., X_{n_0}\}$. We are interested in the proportion of circles that are in a given state after k steps ($k \gg n$ so the circle has a nonzero probability of being in any state). Mathematically, this is $\hat{p_i} = \frac{1}{n_0} \times$ (number of circles in state i) $= \frac{1}{n_0} \sum_{i=1}^{n_0} \mathbb{1}_{X_i=1}$. The *empirical distribution* is then the vector $\hat{p} = (\hat{p_1}, \hat{p_2}, ..., \hat{p_n})$.

Then the probability of getting a certain empirical distribution is proportional to the number of ways we can attain it, which we showed to be the following:

$$P(\hat{p} = p) \propto C(p) = \frac{n_0!}{(n_0 p_1)!...(n_0 p_n)!}$$

We also showed that $\lim_{n_0 \to \infty} \frac{1}{n_0} \log C(p) = -\sum_{i=1}^{n} = p_i \log(pi) := \mathcal{H}(p)$, where $\mathcal{H}$ is the entropy. From this, we found out (but did not formally prove using Lagrange multipliers) that the distribution that maximizes entropy is the uniform distribution $p^* = \left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right)$.

Looking at the problem from this viewpoint allowed us to discover a new way of measuring how close the system is to a random system: relative entropy. We measure how close the empirical distribution is to uniform with this "distance" pseudo-metric:

$$D(\hat{p}||p) = \sum_{x=1}^{n} \hat{p_x} \log\left(\frac{\hat{p_x}}{p_x}\right) = \sum_{x=1}^{n} \hat{p_x} \log(n\hat{p_x}) = \log n - \mathcal{H}(\hat{p})$$

2.3 Randomizing through Markov Chains

The approach we ultimately went forward with was to view the system as a discrete Markov chain. Since the probabilities for moving (counter)clockwise do not depend on past positions, but only the current position, it is indeed a Markov chain. See (3) for the construction of the transition matrix. The Markov chain derived below is irreducible, since the system can traverse the entire circle in either direction with a nonzero probability. It is also aperiodic in the general case (see subsection 2.7), since there is a nonzero probability of the ball remaining at its node, meaning the GCD of any period for the ball returning to its state has to be 1. Thus, there exists a unique stationary distribution π. The following section will show that the system converges to this distribution.

2.4 Proof of Randomness

According to subsection 2.3, we know that given a transition matrix of the Markov Chain of this problem, the aperiodicity and irreducibility of the Markov Chain suggests that the state vector will eventually reach an unique equilibrium. But in order for the equilibrium to be considered "random", one has to ensure that the equilibrium follows a uniform distribution. This section will prove that random walk on a circle, with a non-zero probability of staying and any probability of going left and right, will eventually have a random position.

Proof. Let the probabilities of going left, right, and staying put be denoted by L, R, S. Then, we have

$$L + R + S = 1 \tag{1}$$

When the state vector $\vec{x}$ reaches equilibrium, it will not change when applied to the transition matrix T.

$$\vec{x} * T = \vec{x}, \text{ where } \vec{x} = (x_1, x_2, ..., x_n), \tag{2}$$

$$T = \begin{bmatrix} S & L & 0 & 0 & 0 & ... & 0 & R \\ R & S & L & 0 & 0 & ... & 0 & 0 \\ 0 & R & S & L & 0 & ... & 0 & 0 \\ ... & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & ... & S & L \\ L & 0 & 0 & 0 & 0 & ... & R & S \end{bmatrix} \tag{3}$$

Writing out equation (2) in equations format gives:

$$\begin{cases} Lx_n - (L+R)x_1 + Rx_2 = 0 & (4) \\ Lx_1 - (L+R)x_2 + Rx_3 = 0 & (5) \\ ... & (6) \\ Lx_{n-1} - (L+R)x_n + Rx_1 = 0 & (7) \end{cases}$$

When $R = 0$, the equations become:

$$\begin{cases} Lx_n - Lx_1 = 0 & (8) \\ Lx_1 - Lx_2 = 0 & (9) \\ ... & (10) \\ Lx_{n-1} - Lx_n = 0 & (11) \end{cases}$$

It is apparent that $x_1 = x_2 = x_3 = ... = x_n$

$$\text{since } x_1 + x_2 + x_3 + ... + x_n = 1$$

$$\text{therefore } x_1 = x_2 = x_3 = ... = x_n = \frac{1}{n}$$

This means that the final equilibrium vector x is of a uniform distribution, hence reaching a state of "randomness"

When $R \neq 0$ We can rewrite this group of equations into:

$$\begin{cases} Lx_n - (L+R)x_1 + Rx_2 = 0 & (12) \\ Lx_{n-1} - (L+R)x_n + Rx_1 = 0 & (13) \\ Lx_i - (L+R)x_{i+1} + Rx_{i+2} = 0 \text{ for } i = 1, 2, 3, ..., n-2 & (14) \end{cases}$$

First we focus on the equation

$$Lx_i - (L+R)x_{i+1} + Rx_{i+2} = 0 \text{ for } i = 1, 2, 3, ..., n-2 \tag{15}$$

A difference equation can be constructed to solve for this equation:

$$L - (L+R)\lambda + R\lambda^2 = 0 \tag{16}$$

We have:

$$\lambda_1 = 1, \lambda_2 = \frac{L}{R} \tag{17}$$

When $L \neq R$, the solution to equation(15) is

$$x_n = A\lambda_1^{i-1} + B\lambda_2^{i-1} = A + (\frac{L}{R})^{i-1} \text{ for } A, B \in \mathbb{R} \tag{18}$$

Plug in the results in (18) into equation(12) and (13), we have

$$\begin{cases} B(\frac{L^i}{R^{i-1}} - L) = 0 & (19) \\ B(L - \frac{L^i}{R^{i-1}}) = 0 & (20) \end{cases}$$

Since the above two equations have to hold for all values of i, we come to the conclusion that

$$B = 0$$

Plug this back in to equation(18), we have the final solution

$$x_i = A \text{ for } i = 1, 2, 3, ..., n \tag{21}$$

$$\therefore x_1 = x_2 = x_3 = ... = x_n$$

$$\because x_1 + x_2 + x_3 + ... + x_n = 1$$

$$\therefore x_1 = x_2 = x_3 = ... = x_n = \frac{1}{n}$$

When $L = R$, the solution to the difference equation(15) is

$$x_i = A + Bn \tag{22}$$

Similar to the $L \neq R$ case, this also gives

$$x_1 = x_2 = x_3 = ... = x_n = \frac{1}{n}$$

So, in conclusion, the unique equilibrium state of $\vec{x}$ is the uniform distribution.

□

2.5 Computer Simulation & Data Analysis

2.5.1 Simulation

We first take in the number of points on the circle n, the probabilities of going left L, right R, and stay S, and the tolerance of error ϵ we are to bear. Then we randomly pick a starting point, going around the circle, and use an n-dimensional row vector $\vec{x}$ to represent all the probabilities of the particle's presence at that specific point. Provided the equilibrium is not yet reached, we will continue to right-multiply the current vector of probabilities $\vec{x}$ by the transition matrix T (3). When the equilibrium is reached, we record the number of points on the circle n, the number of steps N the randomization takes, the three probabilities L,R,S, and ϵ . The Matlab code and data (in .csv format) can be viewed in the *matlab_sim* directory in our code base repo.

The pseudocode for problem1 is the following:

Algorithm 1 Problem1 Simulation

Require: n(number of points), t(upperbound of steps), probabilities lp, sp, rp, error e
Ensure: $n \in \mathbb{N}$ and $lp, sp, rp, e \in (0, 1)$
  Construct the transition matrix M using lp, sp, rp
  Construct the row vector $v = (1, 0, ..., 0)$
  $count = 0$ (record number of steps taken)
  while not random and $count < t$ **do**
    Update v: $v \leftarrow v \cdot M$
  end while
  record the value of *count*
  ouput .csv file

Since we do not expect the equilibrium to be where all the inputs of the row vector $\vec{x}$ to be exactly the same when represented as a float in a computer program, we come up with the following definition of equilibrium:

2.5.2 Definition of Equilibrium

For a tolerance $\epsilon \in (0, 1)$,

$$\max_{1 \le i \le n} |x_i - \frac{1}{n}| < \epsilon$$

2.5.3 Data Analysis

We first discuss the relationship between the number of points n and number of steps N. We stick with the base probabilities: $\frac{1}{4}$ for left and right, respectively, and $\frac{1}{2}$ for staying. We set the error to be $5 \cdot 10^{-5}$, and calculate the number of steps N from $n = 1$ to $n = 1001$, step size equals 10. The data is available at Appendix 5.1

After plotting the data on a graph, we have to find the function that fits the best. Considering its fast growth, we attempt to fit it with an exponential function $N = a \cdot e^{bx}$, , or a power function $N = a \cdot n^b$, where parameters $a, b \in \mathbb{R}$.

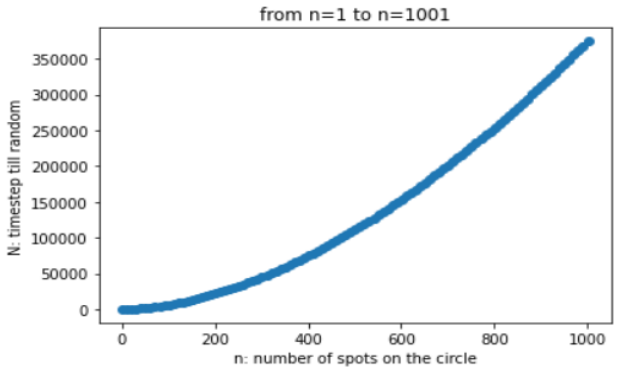
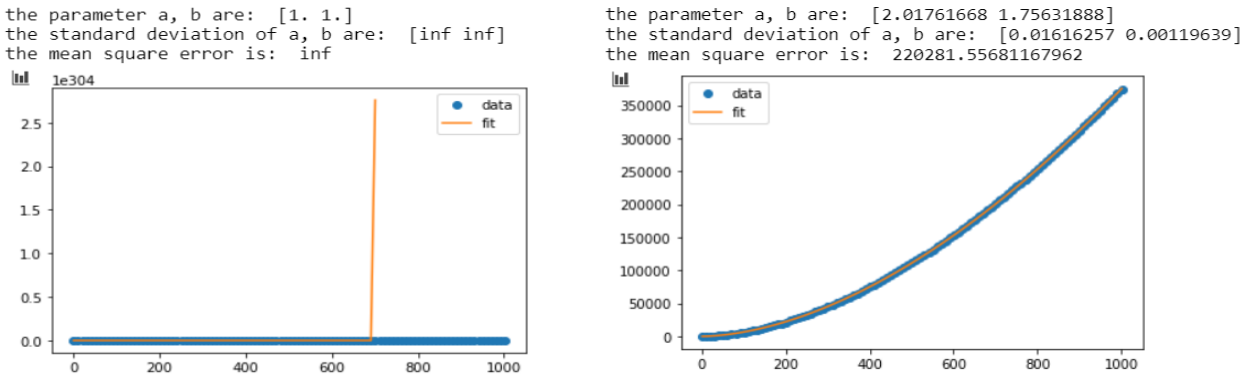


Figure 1: Source of data: BUMP-code/matlab_sim/data3.csv



(a) exponential fit

(b) power fit

Figure 2: Plotting: BUMP-code/data1_plotting.ipynb

By the plotted data in figure 2, we can conclude that N follows a power growth of n.

Then we shift our attention to the relationship between the tolerance of error ϵ and number of steps N, still sticking with the base probabilities $\frac{1}{4}, \frac{1}{4}, \frac{1}{2}$. Yet this time, we hold n as a constant (36/24/15/12, whereas $n = 36$ shows the greatest difference), and manipulate the value of ϵ from $2 \cdot 10^{-6}$ to 10^{-4}, step size equals $2 \cdot 10^{-6}$. Intuitively, we would expect larger N for smaller ϵ. We come up with two types of functions: a multiplicative inverse $N = \frac{a}{x^b}$ or a logarithmic function $N = a \cdot log(b)$, where parameters $a, b \in \mathbb{R}$. The data for the following graphs can be found at Appendix 5.2

In figure 4, both functions fit the data to some extent. However, if we pay attention to the mean square error of the fitting functions respectively, the MSE of a logarithmic fit is much smaller than that of an inverse fit. It suffices to show that N follows a logarithmic growth of ϵ.

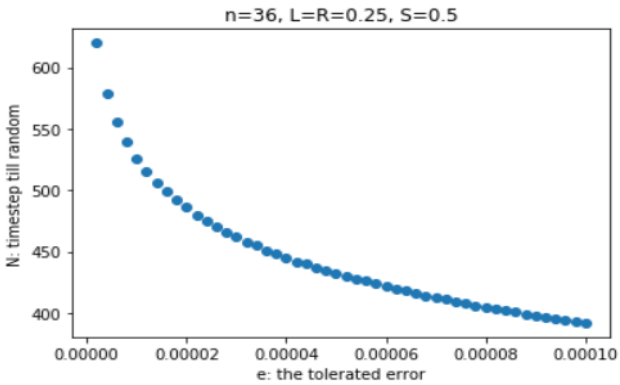
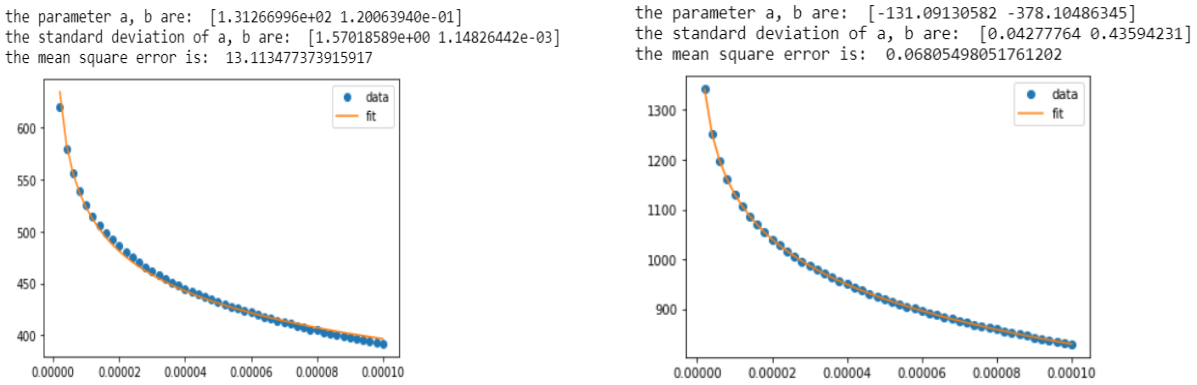


Figure 3: Source of data: BUMP-code/matlab_sim/data_e_N_6.csv



(a) inverse fit

(b) logarithmic fit

Figure 4: Plotting: BUMP-code/epsilon_plotting.ipynb

However, it is noteworthy that all the above graphs and discussions are predicated upon the assumption that $L = R$ (symmetric). In cases where $L \neq R$ (asymmetric), convergence to randomness may take a different time, compared with the symmetric cases where $L = R = \frac{1-S}{2}$. In the following figure, we fix $n = 36$, and plot the logarithmic fits of $L = R = \frac{1}{4}$ and $L = \frac{1}{3}, R = \frac{1}{6}$ respectively. The data for the symmetric case is referred to in the previous discussion; the data for the asymmetric case can be viewed at Appendix 5.3

It is clearly shown in figure 5 that asymmetric probabilities would lead to delayed convergence to randomness. Hence, we come up with the first conjecture from problem 1.

2.6 Conjectures from the Problem

According to numerical simulation, we found that the convergence time to random is longer if $L \neq R, S \neq 0$. This leads us to the following conjecture:

Conjecture 1. The location of the particle on the circle converges to random distribution the fastest when the probabilities of going left, right, and staying put are all $\frac{1}{3}$.

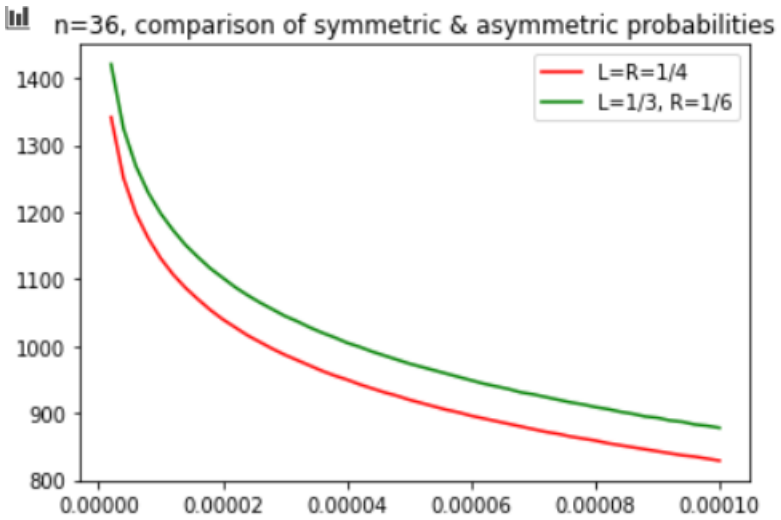


Figure 5: Source of data: BUMP-code/matlab_sim/data_e_N_6.csv, BUMP-code/matlab_sim/data_e_N_8.csv

2.7 Exception: when the probability of staying is zero

There exist cases when the probability of staying, $S = 0$, while neither left nor right probabilities L & R is zero ($L \cdot R \neq 0$).

In the first place, we index all the points on the circle from 1 to n, and record the current probability at each point to be p_i. What we do is putting all the probability, 1, on the 1st point initially. When t=1, $p_1 = 0$, p_n and p_2 is positive (I just list the points on which changes take place, for simplicitys sake, as the rest probabilities are zero). When t=2, $p_n = p_2 = 0$, while p_1, p_3, p_{n-1} are all positive. We can observe a pattern: when t is even, the probabilities on all odd-indexed points are nonzero, and vice versa.

If we model it as a Markov Chain problem, and suppose, for example, $L = R = \frac{1}{2}, n = 4$. The transition matrix P_4 is: $\begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \end{bmatrix}$. We can diagonalize it by $P_4 = SJS^{-1}$ (J being diagonal). J turns out to be $\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, and $J^i = \begin{bmatrix} (-1)^i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ We also know that ${P_4}^{\mathrm{i}} = SJ^{\mathrm{i}}S^{-1}$, so the distribution will oscillate between two states without ever converging.

3 Problem 2: Random Shuffling of Cards

3.1 Introduction: the problem

Start with a deck of n cards, arranged in a circle on the table. At every turn, we either take two adjacent cards and swap them, or else do nothing. How often does this have to be repeated until the ordering of the cards becomes approximately random? How about other shuffling methods (such as exchanging two randomly chosen cards in arbitrary position)?

3.2 Modeling the Problem & Markov Chain Representation

Since all n cards are in a circle, the number of possible configurations of them should be $(n-1)!$. Take n = 4 as an example. We have 6 distinct states: sample space $S = \{(1,2,3,4),(1,2,4,3),(1,3,2,4),(1,3,4,2),(1,4,2,3),(1,4,3,2)\}$.The key is to note that each group of states has exactly one element that starts with 1.

We now apply the following ordering of all the configurations when $n = 4$:

1. All elements of S begin with 1, so we move to the second position, so we move to the second card (starting with card 1).
2. Two elements have 2 in the second position: (1,2,3,4) and (1,2,4,3). Since for the third card $3 < 4$, we call (1,2,3,4) state 1 and (1,2,4,3) state 2.
3. Two elements have 3 in the second position: (1,3,2,4) and (1,3,4,2). We call (1,3,2,4) state 3 and (1,3,4,2) state 4 for the same reason.
4. Two elements have 4 in the second position: (1,4,2,3) and (1,4,3,2). Similarly, we call (1,4,2,3) state 5 and (1,4,3,2) state 6.

Thus we have a way to number the states for any n; this method also ensures that (1,2,...,n) is always state 1, represented by a row vector (1,0,...,0). We also constructed the transition diagram among the states.

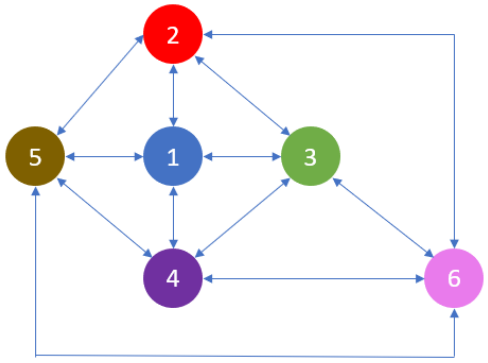


Figure 6: Transition Diagram of Configurations, n=4

If we have a probability $p \in [0,1]$ to swap two cards and the probability of doing nothing

is $(1-p)$, we can write the transition matrix:

$$T = \begin{bmatrix} 1-p & 0.25p & 0.25p & 0.25p & 0.25p & 0 \\ 0.25p & 1-p & 0.25p & 0 & 0.25p & 0.25p \\ 0.25p & 0.25p & 1-p & 0.25p & 0 & 0.25p \\ 0.25p & 0 & 0.25p & 1-p & 0.25p & 0.25p \\ 0.25p & 0.25p & 0 & 0.25p & 1-p & 0.25p \\ 0 & 0.25p & 0.25p & 0.25p & 0.25p & 1-p \end{bmatrix} \tag{23}$$

Similarly, we can write the transition matrix for $n = 5$:

1-p	p/5	p/5	0	0	0	p/5	0	0	p/5	0	0	0	0	0	0	0	0	p/5	0	0	0	0	0
p/5	1-p	0	0	p/5	0	0	p/5	0	0	0	p/5	p/5	0	0	0	0	0	0	0	0	0	0	0
p/5	0	1-p	p/5	0	0	0	0	0	0	0	0	p/5	0	0	p/5	0	0	0	p/5	0	0	0	0
0	0	p/5	1-p	0	p/5	p/5	0	0	0	0	0	0	p/5	0	0	0	p/5	0	0	0	0	0	0
0	p/5	0	0	1-p	p/5	0	0	0	0	0	0	0	p/5	0	0	0	0	p/5	0	0	p/5	0	0
0	0	0	p/5	p/5	1-p	0	p/5	0	0	0	0	0	0	0	0	0	0	0	p/5	0	0	0	p/5
p/5	0	0	p/5	0	0	1-p	p/5	p/5	0	0	0	0	0	0	0	0	0	0	0	p/5	0	0	0
0	p/5	0	0	0	p/5	p/5	1-p	0	0	p/5	0	0	0	p/5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	p/5	0	1-p	p/5	0	0	0	p/5	p/5	0	0	0	0	0	0	p/5	0	0
p/5	0	0	0	0	0	0	0	p/5	1-p	0	p/5	0	0	0	p/5	p/5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	p/5	0	0	1-p	p/5	0	0	0	p/5	0	0	0	p/5	p/5	0	0	0
0	p/5	0	0	0	0	0	0	0	p/5	p/5	1-p	0	0	0	0	0	0	0	0	0	p/5	p/5	0
0	p/5	p/5	0	0	0	0	0	0	0	0	0	1-p	p/5	p/5	0	0	0	0	0	0	0	p/5	0
0	0	0	p/5	p/5	0	0	0	p/5	0	0	0	p/5	1-p	0	0	p/5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	p/5	p/5	0	0	0	p/5	0	1-p	p/5	0	0	0	0	0	0	0	p/5
0	0	p/5	0	0	0	0	0	0	p/5	p/5	0	0	0	p/5	1-p	0	p/5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	p/5	0	0	0	p/5	0	0	1-p	p/5	p/5	0	0	0	p/5	0
0	0	0	p/5	0	0	0	0	0	0	0	0	0	0	0	p/5	p/5	1-p	0	0	p/5	0	0	p/5
p/5	0	0	0	p/5	0	0	0	0	0	0	0	0	0	0	0	p/5	0	1-p	p/5	p/5	0	0	0
0	0	p/5	0	0	p/5	0	0	0	0	p/5	0	0	0	0	0	0	0	p/5	1-p	0	0	p/5	0
0	0	0	0	0	0	p/5	0	0	0	p/5	0	0	0	0	0	0	p/5	p/5	0	1-p	p/5	0	0
0	0	0	0	p/5	0	0	0	p/5	0	0	p/5	0	0	0	0	0	0	0	p/5	1-p	0	p/5	
0	0	0	0	0	0	0	0	0	0	0	p/5	p/5	0	0	0	p/5	0	0	p/5	0	0	1-p	p/5
0	0	0	0	0	p/5	0	0	0	0	0	0	0	0	p/5	0	0	p/5	0	0	0	p/5	p/5	1-p

Figure 7: Transition Matrix when n=5

We also have a matlab file for generating the transition matrices for larger n's. See the file at: BUMP/cardTransitionMatrixScript.m

Note that these matrices are irreducible (look at the transition diagram). They are also aperiodic for the same reason the circles with nonzero probability of stay put were in problem 1. This means that there is a unique stationary distribution and we will reach it.

3.3 Computer Simulation & Data Analysis

3.3.1 Remodeling

Unlike problem 1, it is computationally expensive to model problem 2 using the transition matrix put forward in the Markov Chain Representation. A possible way to walk out this

plight when programming is to just swap the two elements in a list (or an array). We also take on the strategy of defining the configurations as numbers starting with 1 (card 1), and store them in a hashmap as keys ($(n-1)!$ entries in total, still very computational expensive, yet better than a $(n-1)!$ by $(n-1)!$ matrix multiplied at each step. The values of a key is the number of times we observe at the end of each step.

The pseudocode for this simulation is as following:

Algorithm 2 Problem2 Simulation

Require: n(number of cards), $pofs$(prob of swap, t(upperbound of steps), error e
Ensure: $n \in \mathbb{N}$ and $pofs, e \in (0,1)$
  list all the permutations of numbers 1 to n starting with 1 in list p
  use the permutations in p as keys to make hashmap *pmap*
  counter $= 0$ (record number of steps taken)
  while not random and *counter* $< t$ **do**
    choose two cards (this differs in adjacent and random cases)
    randomly generate $prob \in (0,1)$
    if $prob < pofs$ **then**
      swap cards (this differs in adjacent and random cases)
      update current card configuration
    end if
    update the value of corresponding configuration in *pmap*
    counter $++$
  end while
  record the value of *counter*
  ouput .csv file

3.3.2 Definition of Equilibrium

If we define equilibrium absolutely, i.e. each configuration should appear no more/less than x times ($x \in \mathbb{N}$) than the supposed times $\frac{t}{(n-1)!}$, where t is the number of steps we take, we would find it nearly impossible to reach equilibrium when n is really large (because we have to set x larger correspondingly).
Therefore, we come up with our relative definition:

For a tolerance $\epsilon \in (0,1)$,

$$\max_{1 \le i \le (n-1)!} |x_i - \frac{t}{(n-1)!}| < \frac{\epsilon \cdot t}{(n-1)!}$$

where t is number of steps taken prior to when the equilibrium is checked.

Note: ϵ is denoted as e in our data in Appendix 5.4 - 5.7.

3.3.3 Data Analysis

In this section, we used Monte-Carlo Method to acquire our data in Appendix 5.4 through Appendix 5.7 by sampling 1000 times for each ϵ when $n = 4$.

To begin with, we set the probability of swapping two adjacent cards (random swapping will be discussed later) at each step to be $\frac{1}{2}$. We let $n = 4$ and carry out simulation for ϵ from 0.10 to 0.01, step size equals -0.01, and the data for the base simulation is here. It is intriguing to observe that the product of the square of ϵ and the number of steps taken to be approximately a constant $c \in (17, 19)$, which implies a relationship of multiplicative inverse between the two parameters.

The data is plotted here, and the source of which is at Appendix 5.4.

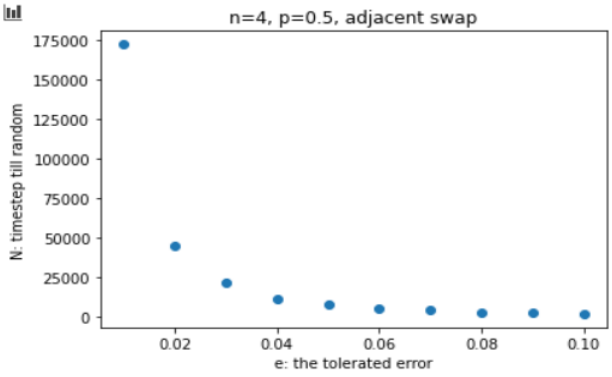
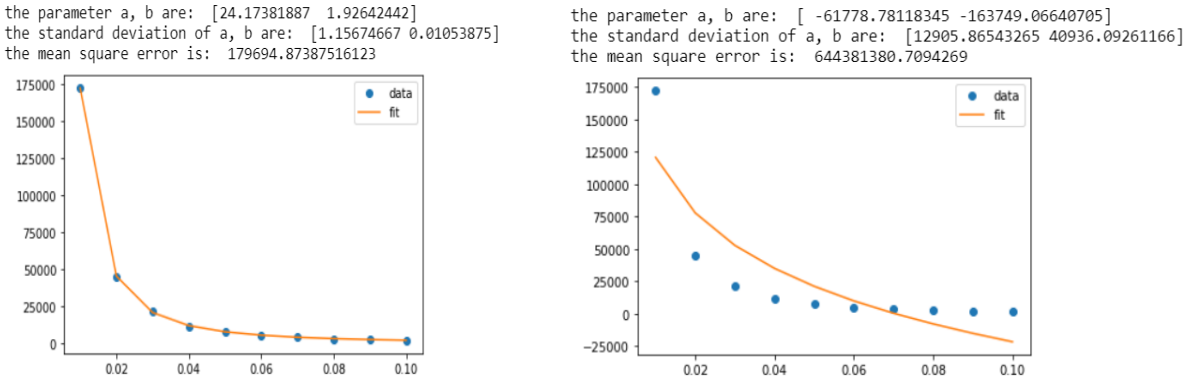


Figure 8: Source of data: BUMP-code/card_sim/epsilon_t.csv



(a) inverse fit

(b) logarithmic fit

Figure 9: Plotting: BUMP-code/card_sim/plotting.ipynb

Comparing the standard deviations and mean square errors of the coefficients of the two potential fits, we can conclude that an inverse fit is better. Our next question is how, if possible, the number of steps required for us to arrive at the equilibrium would be affected by the probability to swap or not at each step, when the method of swapping stays unchanged (only swap two adjacent cards). In addition to $p = 0.5$, we also collected data for $p = 0.3$ and $p = 0.7$, where the number of cards $n = 4$ and error ϵ goes from 0.1 to 0.01, with step size -0.1. The data for $p = 0.3, 0.7$ can be viewed at Appendix 5.6 & 5.7, and is plotted in figure 10.

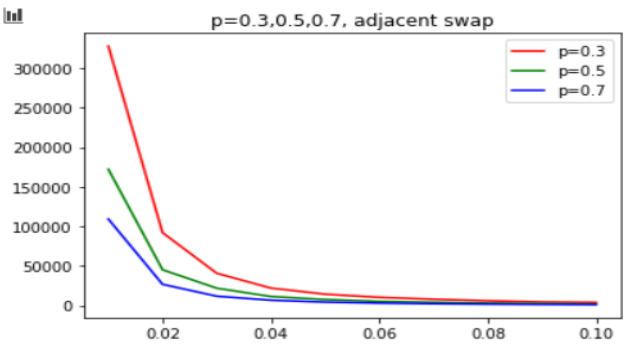


Figure 10: Source of data: BUMP-code/card_sim/epsilon_t.csv, BUMP-code/card_sim/epsilon_t2.csv, BUMP-code/card_sim/epsilon_t3.csv

Figure 10 portrays the speed of convergence to ϵ for different probabilities of swapping while holding n constant, there is evidence showing that the smaller the probability of swapping, the slower the speed for us to arrive at equilibrium.

We also need to account for the influence the method of swapping may have on the randomizing process. Hence, we collect data and control $n = 4, p = 0.5$. By plotting and comparing it with what we get when the swapping is adjacent, we observe no significant difference in the data, which implies that adjacent and random swapping would result in the same order of magnitude.

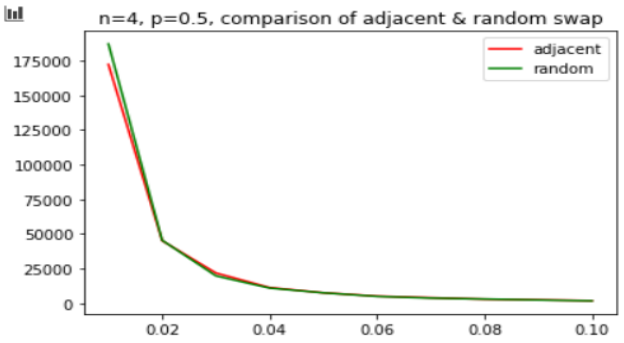


Figure 11: Source of data: BUMP-code/card_sim/epsilon_t.csv, BUMP-code/card_sim/epsilon_t1.csv

3.4 Conjectures from the problem

Considering the data we get for different values of p, we think that p determines the rate of randomization to some extent. Greater p renders more frequent shuffling. Therefore, we come up with this conjecture intuitively:

Conjecture 2. Suppose we have an array A with finite number of elements and randomize by exchanging the elements at different steps. Then the speed of randomization is independent of the choice of elements, but depends on the probability of exchanging the elements at each step. The greater the probability, the faster A will be randomized.

4 Conclusion

From the two problems discussed in previous sections, it is proved that we would finally arrive at a complete randomization after a fair amount of time (or steps), regardless the method of conducting the randomization process (shown in problem 2). The factors contributing most to speed of randomization are (i) the number of elements to randomize (ii) the selection of error we choose for each context.

1. The elements to randomize. In problem 1, we plotted the data to suggest a power growth of the number of steps required N to the number of elements n, approximated by the function $N = 2.0176 \cdot n^{1.7562}$.

2. The selection of maximum error ϵ chosen for each context. In the first problem, we calculated the likelihood of a point on each of the n points around the circle, and compared the likelihood on each point with the expected value $\frac{1}{n}$. The tolerance ϵ we can bear is measured in terms of probability. We find that N grows logarithmically with respect to ϵ. Yet in problem 2, we do not calculate the real probability of each configuration of cards since this method is too computationally expensive. Instead, we used Monte-Carlo Method (MCM) to conduct 1000 independent computer simulations for each pair of (N, ϵ). Rather than probability, we measured ϵ in proportion to the number of expected observations given the number of steps already taken. The data we collected using this method suggests multiplicative inverse relationship between N and ϵ.

The reason that the best fitting functions we get from the two problems are different may be: (a) the data we collected are insufficient (b) more generative measurement of tolerance of error for checking equilibrium should be defined. Precise probabilities are more mathematically persuasive, yet it is more practical to record observations by running numerous random samples and compare our observations with our expected results.

Despite the discussion above, it should be agreed upon that the Markov Chain Theory is particularly important in randomization problems, as it assumes a stochastic model of the world. Its formulation of the process of randomizing allows us to devolve nearly all problems into those of linear algebra, which can be solved on a computer even for large scale problems.

Future work on this could be directed in one of several aspects: Firstly, the foundational theory in both of the problems explored in this paper could be strengthened. There remained many interested properties to be discovered and conjectures to be proved. Secondly, one could take the problem of randomizing to a more applied aspect. The substitution cypher problem, for example, could be an interesting entry point to examine how the process of randomization could be used in decyphering encoded messages. Thirdly, it would be interesting to explore the generalization of these two problems.

5 Appendix

The code base of this work can be found at `https://github.com/lliu58b/BUMP-code`.

5.1 Problem 1, Data 1

n	N	lp	rp	sp	e
1	0	0.25	0.3	0.5	5.00E-05
11	100	0.25	0.3	0.5	5.00E-05
21	337	0.25	0.3	0.5	5.00E-05
31	697	0.25	0.3	0.5	5.00E-05
41	1172	0.25	0.3	0.5	5.00E-05
51	1756	0.25	0.3	0.5	5.00E-05
61	2445	0.25	0.3	0.5	5.00E-05
71	3235	0.25	0.3	0.5	5.00E-05
81	4122	0.25	0.3	0.5	5.00E-05
91	5106	0.25	0.3	0.5	5.00E-05
101	6182	0.25	0.3	0.5	5.00E-05
111	7349	0.25	0.3	0.5	5.00E-05
121	8605	0.25	0.3	0.5	5.00E-05
131	9948	0.25	0.3	0.5	5.00E-05
141	11376	0.25	0.3	0.5	5.00E-05
151	12889	0.25	0.3	0.5	5.00E-05
161	14484	0.25	0.3	0.5	5.00E-05
171	16161	0.25	0.3	0.5	5.00E-05
181	17918	0.25	0.3	0.5	5.00E-05
191	19754	0.25	0.3	0.5	5.00E-05
201	21668	0.25	0.3	0.5	5.00E-05
211	23658	0.25	0.3	0.5	5.00E-05
221	25725	0.25	0.3	0.5	5.00E-05
231	27866	0.25	0.3	0.5	5.00E-05
241	30082	0.25	0.3	0.5	5.00E-05
251	32371	0.25	0.3	0.5	5.00E-05
261	34732	0.25	0.3	0.5	5.00E-05
271	37165	0.25	0.3	0.5	5.00E-05
281	39668	0.25	0.3	0.5	5.00E-05
291	42242	0.25	0.3	0.5	5.00E-05
301	44885	0.25	0.3	0.5	5.00E-05
311	47596	0.25	0.3	0.5	5.00E-05
321	50376	0.25	0.3	0.5	5.00E-05
331	53223	0.25	0.3	0.5	5.00E-05
341	56137	0.25	0.3	0.5	5.00E-05
351	59117	0.25	0.3	0.5	5.00E-05
361	62162	0.25	0.3	0.5	5.00E-05
371	65273	0.25	0.3	0.5	5.00E-05
381	68448	0.25	0.3	0.5	5.00E-05
391	71687	0.25	0.3	0.5	5.00E-05
401	74989	0.25	0.3	0.5	5.00E-05
411	78354	0.25	0.3	0.5	5.00E-05
421	81782	0.25	0.3	0.5	5.00E-05
431	85271	0.25	0.3	0.5	5.00E-05
441	88822	0.25	0.3	0.5	5.00E-05
451	92434	0.25	0.3	0.5	5.00E-05
461	96106	0.25	0.3	0.5	5.00E-05
471	99838	0.25	0.3	0.5	5.00E-05
481	103630	0.25	0.3	0.5	5.00E-05
491	107480	0.25	0.3	0.5	5.00E-05
501	111390	0.25	0.3	0.5	5.00E-05

(a) n=1 to n=501

n	N	lp	rp	sp	e
511	115360	0.25	0.25	0.5	5.00E-05
521	119390	0.25	0.25	0.5	5.00E-05
531	123470	0.25	0.25	0.5	5.00E-05
541	127610	0.25	0.25	0.5	5.00E-05
551	131810	0.25	0.25	0.5	5.00E-05
561	136060	0.25	0.25	0.5	5.00E-05
571	140370	0.25	0.25	0.5	5.00E-05
581	144740	0.25	0.25	0.5	5.00E-05
591	149160	0.25	0.25	0.5	5.00E-05
601	153640	0.25	0.25	0.5	5.00E-05
611	158170	0.25	0.25	0.5	5.00E-05
621	162750	0.25	0.25	0.5	5.00E-05
631	167390	0.25	0.25	0.5	5.00E-05
641	172090	0.25	0.25	0.5	5.00E-05
651	176830	0.25	0.25	0.5	5.00E-05
661	181630	0.25	0.25	0.5	5.00E-05
671	186480	0.25	0.25	0.5	5.00E-05
681	191390	0.25	0.25	0.5	5.00E-05
691	196350	0.25	0.25	0.5	5.00E-05
701	201350	0.25	0.25	0.5	5.00E-05
711	206420	0.25	0.25	0.5	5.00E-05
721	211530	0.25	0.25	0.5	5.00E-05
731	216690	0.25	0.25	0.5	5.00E-05
741	221900	0.25	0.25	0.5	5.00E-05
751	227170	0.25	0.25	0.5	5.00E-05
761	232480	0.25	0.25	0.5	5.00E-05
771	237840	0.25	0.25	0.5	5.00E-05
781	243260	0.25	0.25	0.5	5.00E-05
791	248720	0.25	0.25	0.5	5.00E-05
801	254230	0.25	0.25	0.5	5.00E-05
811	259790	0.25	0.25	0.5	5.00E-05
821	265400	0.25	0.25	0.5	5.00E-05
831	271060	0.25	0.25	0.5	5.00E-05
841	276760	0.25	0.25	0.5	5.00E-05
851	282520	0.25	0.25	0.5	5.00E-05
861	288320	0.25	0.25	0.5	5.00E-05
871	294170	0.25	0.25	0.5	5.00E-05
881	300060	0.25	0.25	0.5	5.00E-05
891	306010	0.25	0.25	0.5	5.00E-05
901	312000	0.25	0.25	0.5	5.00E-05
911	318030	0.25	0.25	0.5	5.00E-05
921	324110	0.25	0.25	0.5	5.00E-05
931	330240	0.25	0.25	0.5	5.00E-05
941	336420	0.25	0.25	0.5	5.00E-05
951	342640	0.25	0.25	0.5	5.00E-05
961	348900	0.25	0.25	0.5	5.00E-05
971	355210	0.25	0.25	0.5	5.00E-05
981	361570	0.25	0.25	0.5	5.00E-05
991	367970	0.25	0.25	0.5	5.00E-05
1001	374410	0.25	0.25	0.5	5.00E-05

(b) n=521 to n=1001

Figure 12: Source: BUMP-code/matlab_sim/data3.csv

5.2 Problem 1, Data 2

n	N	lp	rp	sp	e
36	1342	0.25	0.25	0.5	2.00E-06
36	1251	0.25	0.25	0.5	4.00E-06
36	1198	0.25	0.25	0.5	6.00E-06
36	1161	0.25	0.25	0.5	8.00E-06
36	1131	0.25	0.25	0.5	1.00E-05
36	1107	0.25	0.25	0.5	1.20E-05
36	1087	0.25	0.25	0.5	1.40E-05
36	1070	0.25	0.25	0.5	1.60E-05
36	1054	0.25	0.25	0.5	1.80E-05
36	1040	0.25	0.25	0.5	2.00E-05
36	1028	0.25	0.25	0.5	2.20E-05
36	1016	0.25	0.25	0.5	2.40E-05
36	1006	0.25	0.25	0.5	2.60E-05
36	996	0.25	0.25	0.5	2.80E-05
36	987	0.25	0.25	0.5	3.00E-05
36	979	0.25	0.25	0.5	3.20E-05
36	971	0.25	0.25	0.5	3.40E-05
36	963	0.25	0.25	0.5	3.60E-05
36	956	0.25	0.25	0.5	3.80E-05
36	950	0.25	0.25	0.5	4.00E-05
36	943	0.25	0.25	0.5	4.20E-05
36	937	0.25	0.25	0.5	4.40E-05
36	931	0.25	0.25	0.5	4.60E-05
36	926	0.25	0.25	0.5	4.80E-05
36	920	0.25	0.25	0.5	5.00E-05

(a) ϵ from $2 \cdot 10^{-6}$ to $5 \cdot 10^{-5}$

n	N	lp	rp	sp	e
36	915	0.25	0.25	0.5	5.20E-05
36	910	0.25	0.25	0.5	5.40E-05
36	905	0.25	0.25	0.5	5.60E-05
36	901	0.25	0.25	0.5	5.80E-05
36	896	0.25	0.25	0.5	6.00E-05
36	892	0.25	0.25	0.5	6.20E-05
36	888	0.25	0.25	0.5	6.40E-05
36	884	0.25	0.25	0.5	6.60E-05
36	880	0.25	0.25	0.5	6.80E-05
36	876	0.25	0.25	0.5	7.00E-05
36	872	0.25	0.25	0.5	7.20E-05
36	869	0.25	0.25	0.5	7.40E-05
36	865	0.25	0.25	0.5	7.60E-05
36	862	0.25	0.25	0.5	7.80E-05
36	859	0.25	0.25	0.5	8.00E-05
36	855	0.25	0.25	0.5	8.20E-05
36	852	0.25	0.25	0.5	8.40E-05
36	849	0.25	0.25	0.5	8.60E-05
36	846	0.25	0.25	0.5	8.80E-05
36	843	0.25	0.25	0.5	9.00E-05
36	840	0.25	0.25	0.5	9.20E-05
36	837	0.25	0.25	0.5	9.40E-05
36	835	0.25	0.25	0.5	9.60E-05
36	832	0.25	0.25	0.5	9.80E-05
36	829	0.25	0.25	0.5	0.0001

(b) ϵ from $5.2 \cdot 10^{-5}$ to 10^{-4}

Figure 13: Source: BUMP-code/matlab_sim/data_e_N_6.csv

5.3 Problem 1, Data 3

n	N	lp	rp	sp	e
36	1421	0.333333	0.166667	0.5	2.00E-06
36	1325	0.333333	0.166667	0.5	4.00E-06
36	1269	0.333333	0.166667	0.5	6.00E-06
36	1229	0.333333	0.166667	0.5	8.00E-06
36	1198	0.333333	0.166667	0.5	1.00E-05
36	1173	0.333333	0.166667	0.5	1.20E-05
36	1151	0.333333	0.166667	0.5	1.40E-05
36	1133	0.333333	0.166667	0.5	1.60E-05
36	1116	0.333333	0.166667	0.5	1.80E-05
36	1102	0.333333	0.166667	0.5	2.00E-05
36	1088	0.333333	0.166667	0.5	2.20E-05
36	1076	0.333333	0.166667	0.5	2.40E-05
36	1065	0.333333	0.166667	0.5	2.60E-05
36	1055	0.333333	0.166667	0.5	2.80E-05
36	1045	0.333333	0.166667	0.5	3.00E-05
36	1037	0.333333	0.166667	0.5	3.20E-05
36	1028	0.333333	0.166667	0.5	3.40E-05
36	1020	0.333333	0.166667	0.5	3.60E-05
36	1013	0.333333	0.166667	0.5	3.80E-05
36	1005	0.333333	0.166667	0.5	4.00E-05
36	999	0.333333	0.166667	0.5	4.20E-05
36	992	0.333333	0.166667	0.5	4.40E-05
36	986	0.333333	0.166667	0.5	4.60E-05
36	980	0.333333	0.166667	0.5	4.80E-05
36	974	0.333333	0.166667	0.5	5.00E-05

(a) ϵ from $2 \cdot 10^{-6}$ to $5 \cdot 10^{-5}$

n	N	lp	rp	sp	e
36	969	0.333333	0.166667	0.5	5.20E-05
36	964	0.333333	0.166667	0.5	5.40E-05
36	959	0.333333	0.166667	0.5	5.60E-05
36	954	0.333333	0.166667	0.5	5.80E-05
36	949	0.333333	0.166667	0.5	6.00E-05
36	944	0.333333	0.166667	0.5	6.20E-05
36	940	0.333333	0.166667	0.5	6.40E-05
36	936	0.333333	0.166667	0.5	6.60E-05
36	931	0.333333	0.166667	0.5	6.80E-05
36	928	0.333333	0.166667	0.5	7.00E-05
36	924	0.333333	0.166667	0.5	7.20E-05
36	920	0.333333	0.166667	0.5	7.40E-05
36	916	0.333333	0.166667	0.5	7.60E-05
36	913	0.333333	0.166667	0.5	7.80E-05
36	909	0.333333	0.166667	0.5	8.00E-05
36	906	0.333333	0.166667	0.5	8.20E-05
36	902	0.333333	0.166667	0.5	8.40E-05
36	899	0.333333	0.166667	0.5	8.60E-05
36	895	0.333333	0.166667	0.5	8.80E-05
36	893	0.333333	0.166667	0.5	9.00E-05
36	889	0.333333	0.166667	0.5	9.20E-05
36	887	0.333333	0.166667	0.5	9.40E-05
36	883	0.333333	0.166667	0.5	9.60E-05
36	881	0.333333	0.166667	0.5	9.80E-05
36	878	0.333333	0.166667	0.5	0.0001

(b) ϵ from $5.2 \cdot 10^{-5}$ to 10^{-4}

Figure 14: Source: BUMP-code/matlab_sim/data_e_N_8.csv

5.4 Problem 2, Data 1

n	p	e	avg_steps	e^2 * avg_steps
4	0.5	0.1	1845.093	18.45093
4	0.5	0.09	2425.962	19.6502922
4	0.5	0.08	2908.035	18.611424
4	0.5	0.07	3993.81	19.569669
4	0.5	0.06	5119.66	18.430776
4	0.5	0.05	7543.809	18.8595225
4	0.5	0.04	11337.57	18.1401072
4	0.5	0.03	21829.74	19.6467669
4	0.5	0.02	45140.62	18.056248
4	0.5	0.01	172250.9	17.2250852

Figure 15: Source: BUMP-code/card_sim/epsilon_t.csv

5.5 Problem 2, Data 2

n	p	e	avg_steps	e^2 * avg_steps
4	0.5	0.1	1735.32	17.3532
4	0.5	0.09	2349.801	19.0333881
4	0.5	0.08	3027.001	19.3728064
4	0.5	0.07	3731.818	18.2859082
4	0.5	0.06	4941.687	17.7900732
4	0.5	0.05	7511.855	18.7796375
4	0.5	0.04	10964.81	17.5436944
4	0.5	0.03	19780.72	17.8026444
4	0.5	0.02	45421.19	18.1684772
4	0.5	0.01	187069.9	18.7069856

Figure 16: Source: BUMP-code/card_sim/epsilon_t1.csv

5.6 Problem 2, Data 3

n	p	e	avg_steps	e^2 * avg_steps
4	0.7	0.1	1137.873	11.37873
4	0.7	0.09	1326.409	10.7439129
4	0.7	0.08	1709.825	10.94288
4	0.7	0.07	2196.239	10.7615711
4	0.7	0.06	3002.884	10.8103824
4	0.7	0.05	4259.679	10.6491975
4	0.7	0.04	6654.925	10.64788
4	0.7	0.03	11745.88	10.5712911
4	0.7	0.02	26720.64	10.6882544
4	0.7	0.01	109306.9	10.9306864

Figure 17: Source: BUMP-code/card_sim/epsilon_t2.csv

5.7 Problem 2, Data 4

n	p	e	avg_steps	e^2 * avg_steps
4	0.3	0.1	3889.985	38.89985
4	0.3	0.09	4441.739	35.9780859
4	0.3	0.08	5929.979	37.9518656
4	0.3	0.07	7900.867	38.7142483
4	0.3	0.06	10297.67	37.0716048
4	0.3	0.05	14292.92	35.7323
4	0.3	0.04	21965.08	35.144128
4	0.3	0.03	40702.62	36.6323607
4	0.3	0.02	92032.52	36.813006
4	0.3	0.01	327828.3	32.7828333

Figure 18: Source: BUMP-code/card_sim/epsilon_t3.csv