S&DS 238 Problem Set #8

Shaun Radgowski

November 8, 2020

Problem 1

(1a)

For f to be stationary for P, the detailed balance condition states that:

$$f(i)P(i,j) = f(j)P(j,i)$$

$$\sum_i f(i)P(i,j) = f(j)$$

For this case of the Ehrenfest chain, $B(n, \frac{1}{2})$ is the number of balls in the first urn, as each of the n balls has a probability of 0.5 to be in the first urn. The n-dimensional vector f would then be such that $f(i)$ is the probability that i balls are in the first urn. The probability transition matrix in this case $P(i,j)$ is the probability that the number of balls in urn 1 goes from i to j. Because only one ball is moved in each time period, the difference between i and j can only be +1 or -1. Thus:

$$P(i, i+1) = 1 - \frac{i}{n}$$

$$P(i, i-1) = \frac{i}{n}$$

And all other $P(i,j) = 0$. Thus:

$$f(i) = \binom{n}{i}(\tfrac{1}{2})^n = \tfrac{n!}{(n-i)!\ i!\ 2^n}$$

$$\sum_i f(i)P(i,j) = f(0)P(0,j) + f(1)P(1,j) + ... + f(n)P(n,j)$$

$$= f(j-1)P(j-1,j) + f(j+1)P(j+1,j) = f(j-1)\tfrac{n-j+1}{n} + f(j+1)\tfrac{j+1}{n}$$

$$= \tbinom{n}{j-1}(\tfrac{1}{2})^n \tfrac{n-j+1}{n} + \tbinom{n}{j+1}(\tfrac{1}{2})^n \tfrac{j+1}{n} = [\tfrac{n!\ (n-j+1)}{(n-j+1)!\ (j-1)!\ n} + \tfrac{n!\ (j+1)}{(n-j-1)!\ (j+1)!\ n}]\tfrac{1}{2^n}$$

$$= [\tfrac{n!\ j}{(n-j)!\ j!\ n} + \tfrac{n!\ (n-j)(j+1)}{(n-j)!\ j!\ (j+1)n}]\tfrac{1}{2^n} = \tfrac{n!}{(n-j)!\ j!\ 2^n}[\tfrac{j}{n} + \tfrac{n-j}{n}]$$

Where the portion in parentheses crunches down to $\frac{n}{n} = 1$. This leaves $\sum_i f(i)P(i,j) = \frac{n!}{(n-j)!\ j!\ 2^n} = f(j)$. This proves that the distribution $B(n, \frac{1}{2})$ satisfies the detailed-balance condition per the definition above.

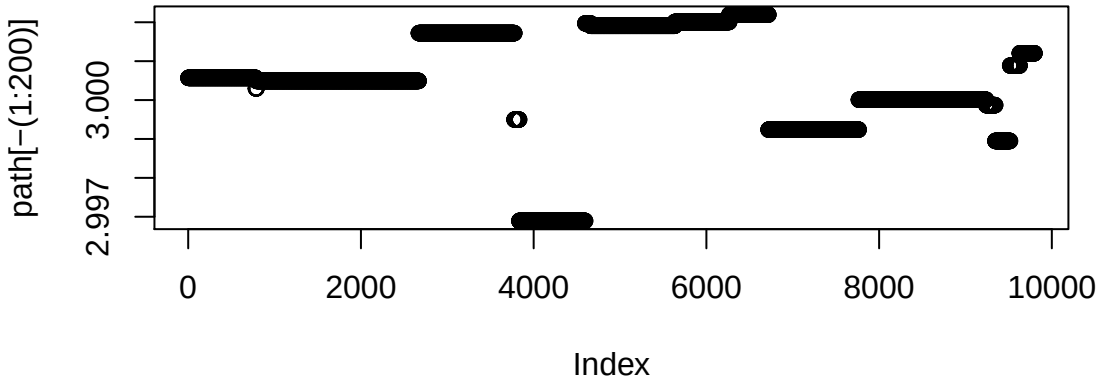
(1b)

The Binomial Distribution $B(n, \frac{1}{2})$ is stationary for the Ehrenfest chain because every possible permutation of the n balls belongs to this distribution. Each ball can considered to be in a binary state of in either urn 1 or urn 2, like a 0 or 1. Thus, any string of 0s and 1s (assignments to urn 1 or urn 2) exists in this distribution, and any transition from one $X_0 \sim B(n, \frac{1}{2})$ to another $X_1 \sim B(n, \frac{1}{2})$ will not change the distribution. This signifies that the distribution is stationary for this Markov Chain.

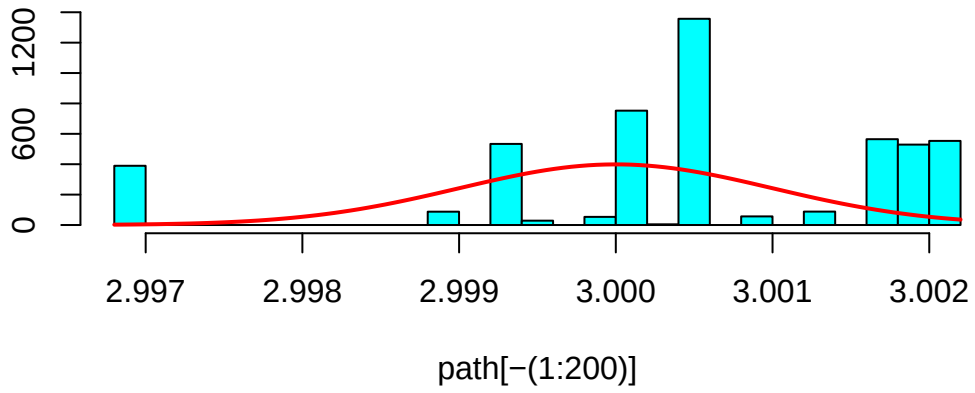
Problem 2

(2a)

```
f <- function(x) {
  return(1/(3*sqrt(2*pi)) * exp(-0.5*((x - 3 )/ 0.001)^2))
}

nit <- 10000
path <- rep(0, nit)
state <- 3  # Initial state
path[1] <- state
scale <- 1
for (i in 2:nit) {
  candidate <- runif(n = 1, min = state - scale, max = state + scale)
  ratio <- f(candidate) / f(state)
  u <- runif(n = 1, min = 0, max = 1)
  if (u < ratio) {
    state <- candidate
  }
  path[i] <- state
}

plot(path[-(1:200)])
```



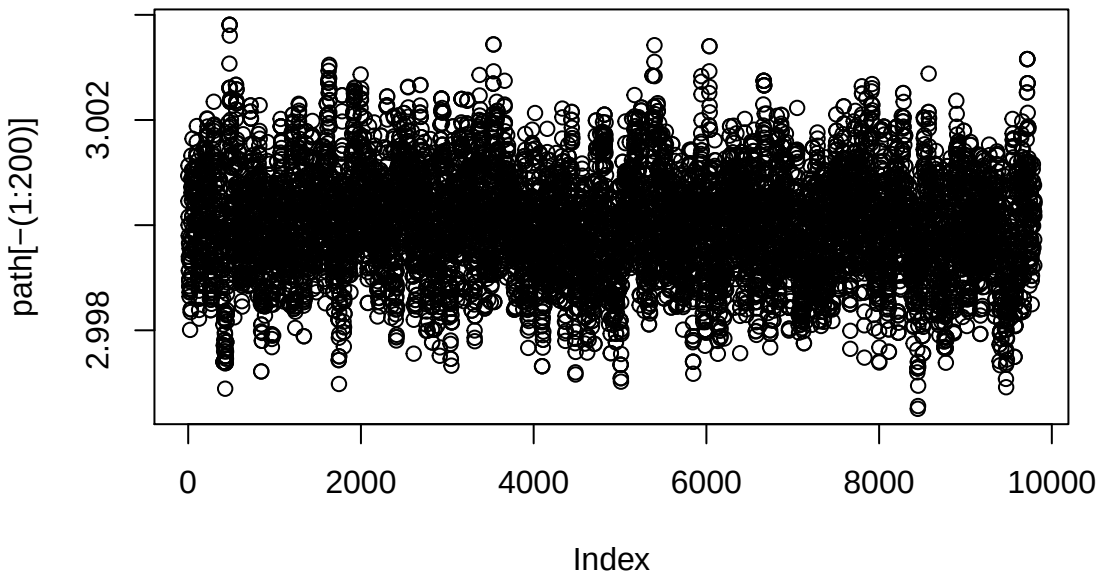
```
library(MASS)
truehist(path[-(1:200)])
con <- integrate(f, 2.9, 3.1)$value
fnormalized <- function(x){f(x)/con}
curve(fnormalized(x), add=TRUE, col="red", lwd=2)
```



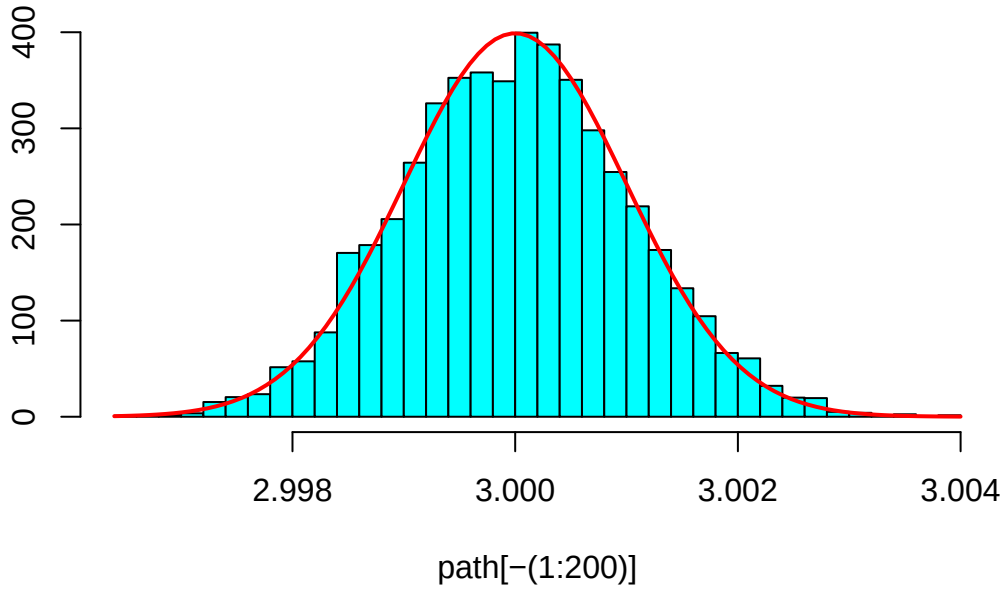
The performance of this method seems very poor, where the bars seem sporadic and very disconnected from the Normal density curve with many gaps. This happened because the scale value is too high at 1, while the standard deviation of the distribution is actually only 0.001. Thus, the step size was too large and the changes overshot the relationship actually depicted in the Normal distribution, thus rejecting many candidate points and concentrating instead on only a few points.

(2b)

```
path <- rep(0, nit)

state <- 3  # Initial state
path[1] <- state

scale <- 0.001
for (i in 2:nit) {
  candidate <- runif(n = 1, min = state - scale, max = state + scale)
  ratio <- f(candidate) / f(state)
  u <- runif(n = 1, min = 0, max = 1)
  if (u < ratio) {
    state <- candidate
  }
  path[i] <- state
}

plot(path[-(1:200)])
```



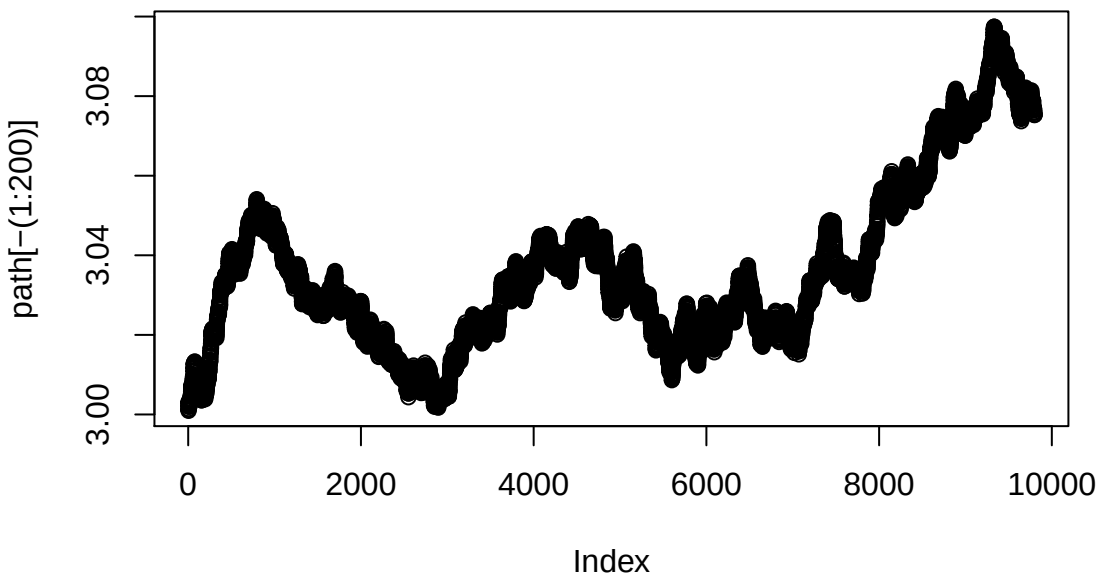

```
library(MASS)
truehist(path[-(1:200)])
curve(fnormalized(x), add=TRUE, col="red", lwd=2)
```



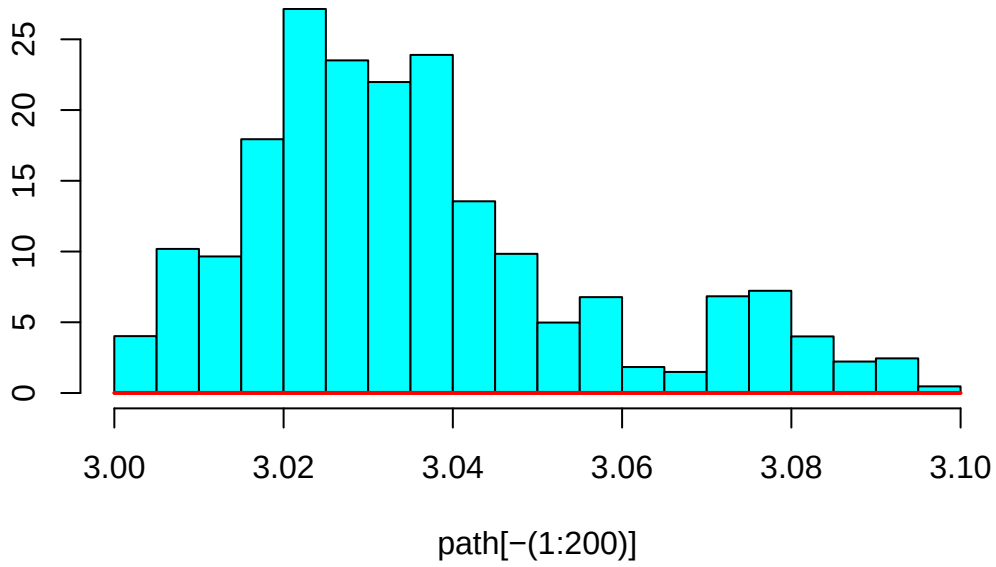
This performance is much better, where the results actually fit the normal distribution shape (as shown by the bell-shaped histogram). This is because the scale value is appropriate given the standard deviation of the distribution.

(2c)

```
f <- function(x) {
  return(1/sqrt(2*pi) * exp(-(x^2)/2))
}

path <- rep(0, nit)

state <- 3  # Initial state
path[1] <- state

scale <- 0.001
for (i in 2:nit) {
  candidate <- runif(n = 1, min = state - scale, max = state + scale)
  ratio <- f(candidate) / f(state)
  u <- runif(n = 1, min = 0, max = 1)
  if (u < ratio) {
    state <- candidate
  }
  path[i] <- state
}

plot(path[-(1:200)])
```



```r
con <- integrate(f, -5, 5)$value
fnormalized <- function(x){f(x)/con}

library(MASS)
truehist(path[-(1:200)])
curve(fnormalized(x), add=TRUE, col="red", lwd=2)
```



Again, this performance is poor because the shape of the bars does not seem to match the Normal distribution. This happened because the scale value is too low at 0.001, while the standard deviation of the distribution is actually 1. Thus, the step size was too small and the changes undershot the relationship actually depicted in the Normal distribution. Additionally, the starting state is already 3 standard deviations away from the mean of this distribution, lowering the probability of a successful performance.

Problem 3

(3a)

$$Q = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The probability vector v such that $vQ = v$ is such that v_i is the number of lines coming from node i, divided by the total number of node/line connections (i.e. 2 times the number of lines).

$$v = (\frac{3}{14} \quad \frac{3}{14} \quad \frac{3}{14} \quad \frac{2}{7} \quad \frac{1}{14})$$

(3b)

We can first use the formula $P(i,j) = Q(i,j)\min\{1, \frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}\}$ to calculate all of the non-diagonal matrix elements, where π is our desired stationary distribution $(\frac{1}{15}\ \frac{2}{15}\ \frac{3}{15}\ \frac{4}{15}\ \frac{5}{15})$. To simplify the process for myself, I will do that using R:

```
Q <- rbind(c(0, 1/3, 1/3, 1/3, 0), c(1/3, 0, 1/3, 1/3, 0), c(1/3, 1/3, 0, 1/3, 0),
           c(0.25, 0.25, 0.25, 0, 0.25), c(0, 0, 0, 1, 0))
dist_pi <- c(1/15, 2/15, 3/15, 4/15, 5/15)
P <- matrix(0, nrow=5, ncol=5)

for (i in 1:5) {
  for (j in 1:5) {
    if (i != j) {
      denominator <- dist_pi[i]*Q[i, j]
      if (denominator != 0) {
        fraction <- (dist_pi[j]*Q[j, i]) / denominator
      }
      else {
        fraction <- 1
      }

      P[i, j] <- (Q[i, j] * min(1, fraction))
    }
  }
}

fractions(P)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0  1/3  1/3  1/3    0
## [2,]  1/6    0  1/3  1/3    0
## [3,]  1/9  2/9    0  1/3    0
## [4,] 1/12  1/6  1/4    0  1/4
## [5,]    0    0    0  1/5    0
```

From there, we can calculate the diagonal elements using the formula $P(i,i) = 1 - \sum_{j \neq i} P(i,j)$. Thus, our final P is:

```
for (i in 1:5) {
  P[i, i] <- 1 - sum(P[i,])
}
fractions(P)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0  1/3  1/3  1/3    0
## [2,]  1/6  1/6  1/3  1/3    0
## [3,]  1/9  2/9  1/3  1/3    0
## [4,] 1/12  1/6  1/4  1/4  1/4
## [5,]    0    0    0  1/5  4/5
```

(3c)

To prove that $(\frac{1}{15}\ \frac{2}{15}\ \frac{3}{15}\ \frac{4}{15}\ \frac{5}{15})$ is stationary for the P above, we can simply confirm that $\pi P = \pi$.

```
dist_pi %*% P
```

```
##            [,1]      [,2] [,3]      [,4]      [,5]
## [1,] 0.06666667 0.1333333  0.2 0.2666667 0.3333333
```

```
dist_pi
```

```
## [1] 0.06666667 0.13333333 0.20000000 0.26666667 0.33333333
```

Because these two lines are equivalent, we can say that π is stationary for the P above.

(3d)

```
nsteps <- 100000
results <- rep(NA, nsteps)
results[1] <- 1 # Initial location

for (i in 2:nsteps) {
  probs <- P[results[i - 1],]
  results[i] <- sample(c(1, 2, 3, 4, 5), size=1, prob=probs)
}

table(results)
```

```
## results
##     1     2     3     4     5
##  6791 13338 19989 26704 33178
```

```
nsteps * dist_pi
```

```
## [1]  6666.667 13333.333 20000.000 26666.667 33333.333
```

The table above shows that our chain does appear to be working the way we intended, yielding results approximately matching our expectations from $(\frac{1}{15}\ \frac{2}{15}\ \frac{3}{15}\ \frac{4}{15}\ \frac{5}{15})$.

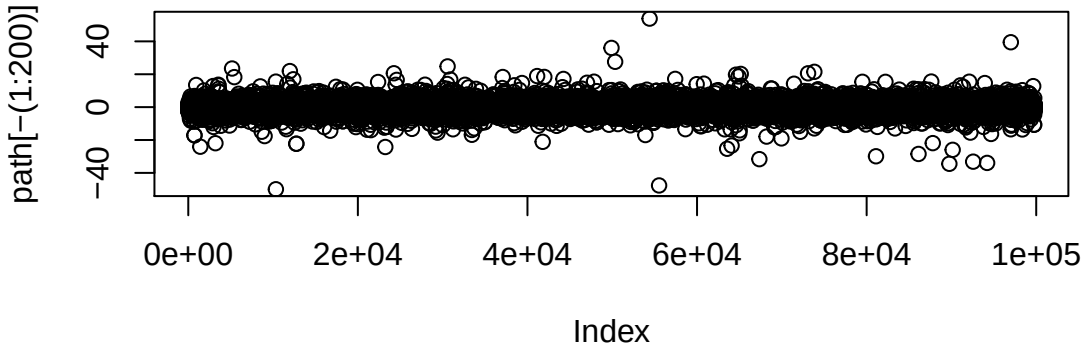
Problem 4

(4a)

The canonical example of the Metropolis-Hastings acceptance probability is $A(i,j) = \min\{1, \frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}\}$. Rewriting this in terms of the variables in question, we have $A(x,y) = \min\{1, \frac{f(y)q(y,x)}{f(x)q(x,y)}\}$. In this specific case, we are told that the probability/density of proposing y does not depend on the current state x, such that $q(x,y) = g(y)$ or $q(y,x) = g(x)$. Making that substitution into our equation above, we are left with the form $A(x,y) = \min\{1, \frac{f(y)g(x)}{f(x)g(y)}\}$.

(4b)

```
f <- function(x) {
  return((1 + (x^2)/3)^(-0.5*4))
}

nit <- 100000
path <- rep(NA, nit)
state <- 0 # Initial state
path[1] <- state
for (i in 2:nit) {
  angle <- runif(n = 1, min = -pi/2, max = pi/2)
  candidate <- tan(angle)
  ratio <- (f(candidate) * dcauchy(state)) / (f(state) * dcauchy(candidate))
  u <- runif(n = 1, min = 0, max = 1)
  if (u < ratio) {
    state <- candidate # Accept the candidate
  }
  path[i] <- state
}

plot(path[-(1:200)])
```



(4c)

```
quantile(abs(path), probs = c(0.9, 0.95, 0.99))
```

```
##      90%      95%      99%
## 2.357615 3.183734 5.820372
```

```
# Factoring in the absolute values, for comparison:
qt(p = c(0.95, 0.975, 0.995), df = 3)
```

```
## [1] 2.353363 3.182446 5.840909
```

These two lines are approximately equal, thus showing our MCMC sample is behaving as anticipated.

Problem 5

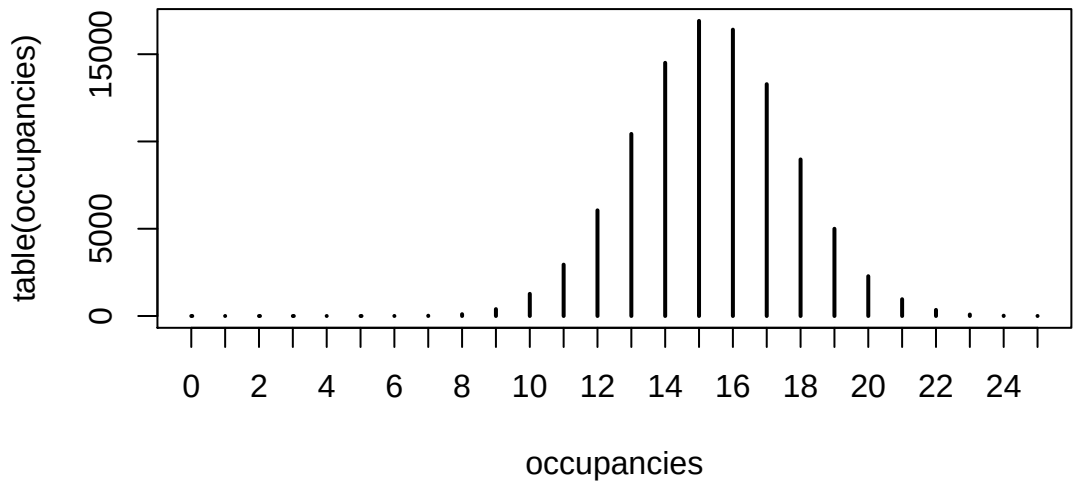
(5a)

In this case, a proposal only belongs to the desired distribution if it satisfies the hard core constraint (meaning it does not violate the constraint's adjacency rules). Thus, to generate sample lattices that fit the distribution, the acceptance probability for a candidate lattice should be 1 if it does not violate the hard core constraint and 0 if it does violate the hard core constraint, because all valid (acceptable) lattices have equal probability in this uniform distribution. This ensures that all accepted candidates belong to the desired distibution and all candidates belonging to the desired distribution are accepted. Given that the starting state lattice will be an accepted candidate (i.e. a valid lattice), flipping one site that does not violate the constraint will result in another valid lattice while flipping one site that violates the constraint will result in an invalid lattice. Putting these pieces together, if the one proposed flip (from 0 to 1 or vice versa) for the selected site will not violate the constraint, it will therefore produce another valid lattice from the desired distribution that should be accepted with probability 1 because of the uniform distribution. If the proposed flip will violate the constraint, it will not produce a lattice from the desired distribution and thus should be rejected (meaning accepted with probability 0).

(5b)

```
nit <- 100000
path <- array(0, dim=c(8, 8, nit))

meets_constraint <- function(lattice) {
  # Check for vertical pairs
  for (i in 1:8) {
    for (j in 1:7) {
      if (lattice[i, j] + lattice[i, j + 1] == 2) {
        return(FALSE)
      }
    }
  }
```

```
  # Check for horizontal pairs
  for (j in 1:8) {
    for (i in 1:7) {
      if (lattice[i, j] + lattice[i + 1, j] == 2) {
        return(FALSE)
      }
    }
  }

  return(TRUE);
}

for (k in 2:nit) {
  x <- sample(1:8, size=1)
  y <- sample(1:8, size=1)
  this_lattice <- path[,,k - 1]
  this_lattice[x, y] <- 1 - this_lattice[x, y]
  if (meets_constraint(this_lattice)) {
    path[,,k] <- this_lattice
  }
  else {
    path[,,k] <- path[,,k - 1]
  }
}

occupancies <- rep(0, nit)
for (i in 1:nit) {
  occupancies[i] <- sum(path[,,i])
}

plot(table(occupancies))
```



(5c)

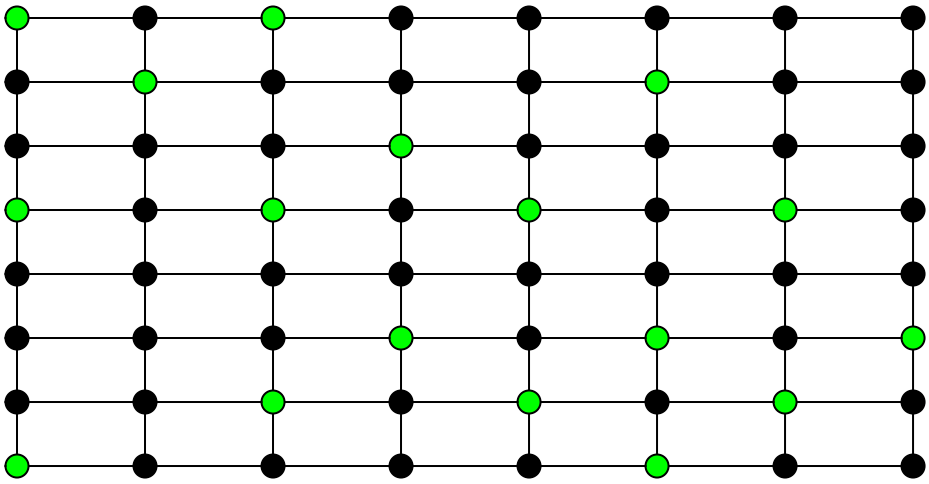
Judging by the graph above, it seems that the 10th percentile is around 12, the median is around 15, and the 90th percentile is around 18. This can be confirmed using R:

```
quantile(occupancies, c(0.1, 0.5, 0.9))
```

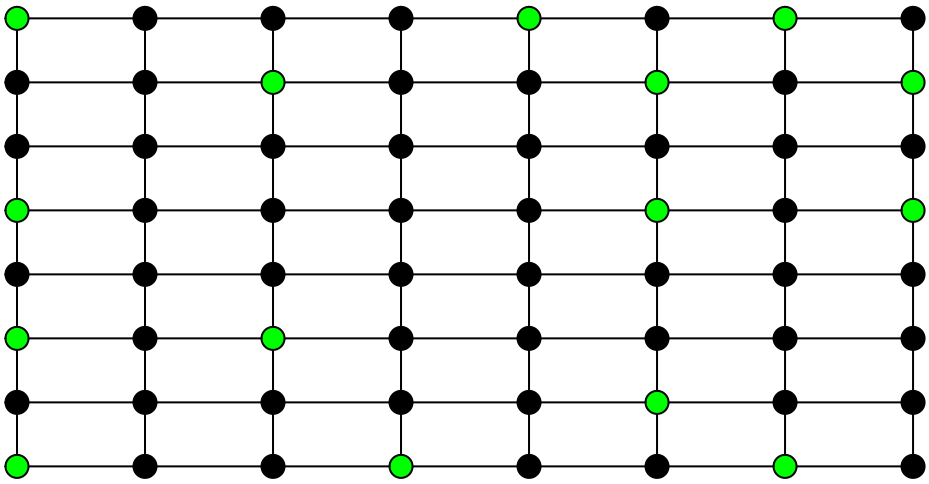
```
## 10% 50% 90%
##  12  15  18
```

(5d)

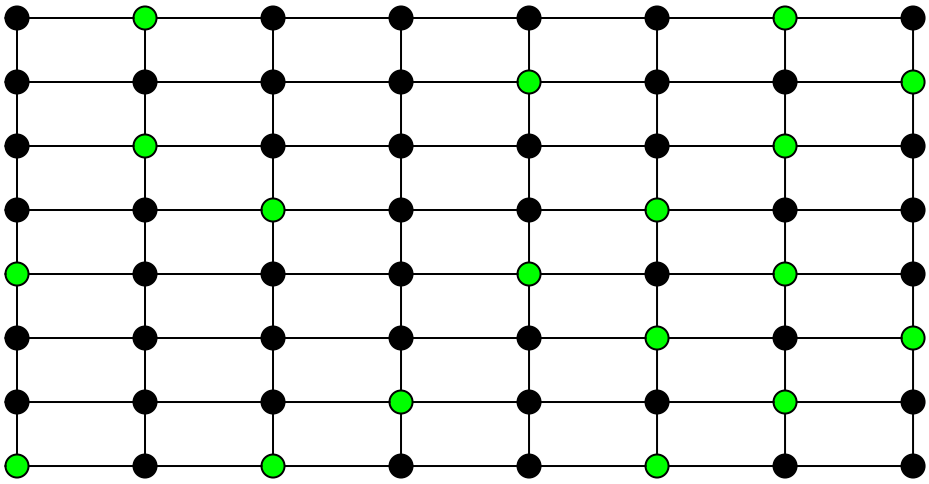
```
library(grid)

plot.hc <- function(mat){
  m <- dim(mat)[1]
  n <- dim(mat)[2]

  grid.newpage()
  rad = .02
  xx = function(j) {j/(n+1)}
  yy = function(i) {(m+1-i)/(m+1)}

  for(i in 1:m){
    grid.lines(c(xx(1),xx(n)),c(yy(i),yy(i)))
  }
  for(j in 1:n){
    grid.lines(c(xx(j),xx(j)),c(yy(1),yy(m)))
  }
  for(j in 1:n){
    for(i in 1:m){
      mycolor <- c('black','green')[1+mat[i,j]]
      grid.circle(xx(j), yy(i), rad, gp=gpar(fill=mycolor))
    }
  }
}

plot.hc(path[,,2500])
```



```
plot.hc(path[,,5000])
```



```
plot.hc(path[,,7500])
```



```
plot.hc(path[,,10000])
```