ISEN 609: Probability for Engineering Decisions

Assignment 1

Name: Dwarkanath Prabhu

Table of Contents

[DTMC Modeling 2](#_Toc497214031)

[Steady-state DTMC 4](#_Toc497214032)

[Calculations 5](#_Toc497214033)

[Components in Bins 5](#_Toc497214034)

[Throughput 6](#_Toc497214035)

[Experiments 7](#_Toc497214036)

[Changing Bin 1 capacity 7](#_Toc497214037)

[Changing Bin 2 capacity 9](#_Toc497214038)

[Changing Machine 1 Success Rate 11](#_Toc497214039)

[Changing Machine 2 Success Rate 13](#_Toc497214040)

[Observations and Inferences 15](#_Toc497214041)

# DTMC Modeling

Let Xn be the number of components in the Bin1 at the beginning of the n-th hour. If a component is present in both bins, the two are assembled immediately, so at the beginning of an hour, there can be components in only one of the 2 bins. So, let Xn grow negatively when there are components in Bin2 instead of Bin 1.

So the state space then becomes, {-B2, -B2+1,…,0, 1, 2,…,B1}

Relevant Python code snippet:

def getStateSpace(B1,B2):

return(list(range(-B2,B1+1)))

S = getStateSpace(B1,B2)

print("The state space is " + str(S))

**For B1 = 3, B2 = 4, α1= 0.7 andα2 = 0.6, the output is as follows:**

The state space is [-4, -3, -2, -1, 0, 1, 2, 3]

*[Side Note: The reason to express the state space in this way is to make it easier to write code. The edge cases i.e. when the bins are full are at the edge of the state space while all middle states have similar transitions. Also, since the state space is 1-dimensional instead of 2-dimensional, the functions can be of O(n) instead of O(n2)]*

So, the transition diagram can be obtained as follows:

Considering edge cases when the bins are full and their corresponding machines are shut.

If Xn = -B2,

Xn+1 = - B2 wp 1-α1 i.e. a defective component is produced by machine 1 and,

Xn+1 = - B2 +1 wp α1 i.e. a non-defective component is produced by machine 1.

All other states are beyond reach in one step.

Similarly,

If Xn = B1,

Xn+1 = B1 - 1 wp α2 i.e. a non-defective component is produced by machine 2 and,

Xn+1 = B1 wp 1-α2 i.e. a defective component is produced by machine 2.

All other states are beyond reach in one step.

For all other values of Xn in S,

Xn = Xn - 1 wp 1-α2 i.e. a defective component is produced by machine 2 and,

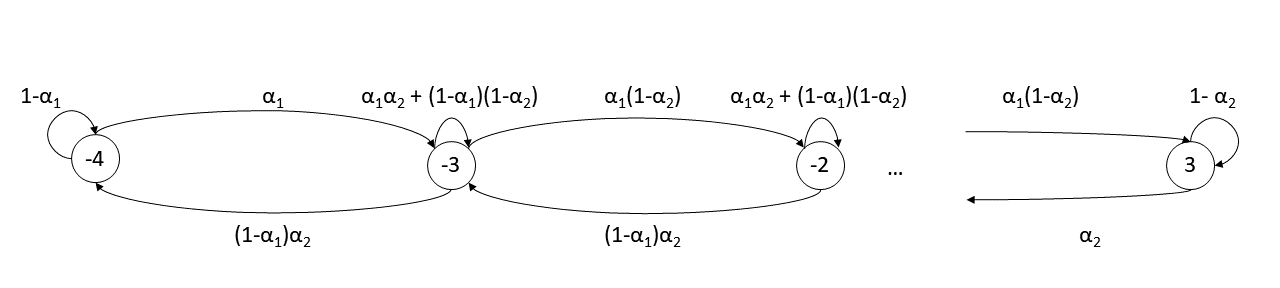
Xn+1 = Xn wp α1α2 + (1-α1)(1-α2) i.e. either both machines produce a defective component or both machines produce

Xn+1 = Xn +1 wp α2 i.e. a non-defective component is produced by machine 2.

**For B1 = 3 and B2 = 4, the transition matrix looks like this:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **-4** | **-3** | **-2** | **-1** | **0** | **1** | **2** | **3** |
| **-4** | 1-α1 | α1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **-3** | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) | 0 | 0 | 0 | 0 | 0 |
| **-2** | 0 | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) | 0 | 0 | 0 | 0 |
| **-1** | 0 | 0 | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) | 0 | 0 | 0 |
| **0** | 0 | 0 | 0 | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | (1-α1)α2 | α1α2 + (1-α1)(1-α2) | α1(1-α2) |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | α2 | 1- α2 |

And the transition diagram like this:



**For B1 = 3, B2 = 4, α1= 0.7 andα2 = 0.6, the transition matrix can be obtained using:**

import numpy as np

alpha1 = 0.7

alpha2 = 0.6

B1 = 3

B2 = 4

def getPMatrix(B1, B2, alpha1, alpha2):

P = [[0]\*(B1+B2+1)]

P[0][0] =1-alpha1

P[0][1] = alpha1

for i in range(B1+B2-1):

P.append([0]\*(B1+B2+1))

P[i+1][i] = (1-alpha1)\*alpha2

P[i+1][i+1] = alpha1\*alpha2 + (1-alpha1)\*(1-alpha2)

P[i+1][i+2] = alpha1\*(1-alpha2)

P.append([0]\*(B1+B2+1))

P[B1+B2][B1+B2-1] = alpha2

P[B1+B2][B1+B2] =1-alpha2

return(np.matrix(P))

P = getPMatrix(B1,B2,alpha1, alpha2)

print(P)

**Output:**

[[ 0.3 0.7 0. 0. 0. 0. 0. 0. ]

[ 0.18 0.54 0.28 0. 0. 0. 0. 0. ]

[ 0. 0.18 0.54 0.28 0. 0. 0. 0. ]

[ 0. 0. 0.18 0.54 0.28 0. 0. 0. ]

[ 0. 0. 0. 0.18 0.54 0.28 0. 0. ]

[ 0. 0. 0. 0. 0.18 0.54 0.28 0. ]

[ 0. 0. 0. 0. 0. 0.18 0.54 0.28]

[ 0. 0. 0. 0. 0. 0. 0.6 0.4 ]]

# Steady-state DTMC

Since, the DTMC is irreducible and aperiodic, we can solve the following equation to find the steady state probabilities.

This can be solved by substituting the last column in I – P with 1 and inverting that matrix. The code for this when is as below

def getPi(P):

I = np.identity(P.shape[0])

Q = I-P

Q[:,P.shape[0]-1] = 1

a = [0]\*(P.shape[0]-1)

a.append(1)

pi = np.matrix([a])\*Q.getI()

return(pi.round(4))

pi = getPi(P)

print(pi)

**For B1 = 3, B2 = 4, α1= 0.7 andα2 = 0.6, the output is as follows:**

[[0.0091 0.0354 0.0551 0.0858 0.1334 0.2076 0.3229 0.1507]]

These are the steady state probabilities of all the states rounded off to the 4th decimal.

# Calculations

## Components in Bins

Let Yi be the number of components in Bin i for i = 1,2. The average number of components at the beginning of an hour is then given by E[Yi].

Here,

where

where

The code for this can be written as follows:

def getBin1Components(B1, B2, alpha1, alpha2):

P = getPMatrix(B1, B2, alpha1, alpha2)

pi = getPi(P)

result = 0

for i in range(1,B1+1):

result += i\*pi.item(B2 + i)

return(round(result,4))

def getBin2Components(B1, B2, alpha1, alpha2):

P = getPMatrix(B1, B2, alpha1, alpha2)

pi = getPi(P)

result = 0

for i in range(B2):

result += (B2-i)\*pi.item(i)

return(round(result,4))

print(getBin1Components(B1, B2, alpha1, alpha2))

print(getBin2Components(B1, B2, alpha1, alpha2))

**For B1 = 3, B2 = 4, α1= 0.7 andα2 = 0.6, the output is as follows:**

1.3055

0.3386

## Throughput

Let Z be the number of assembled products in an hour. The average number of products assembled is then given by E[Z].

where

where

where

where

The code for this can be written as follows:

def getItems(B1,B2, alpha1, alpha2):

P = getPMatrix(B1, B2, alpha1, alpha2)

pi = getPi(P)

items = 0

for i in range(B2):

items += pi.item(i)\*alpha1

items += pi.item(B2)\*alpha1\*alpha2

for i in range(B1):

items += pi.item(B2+i+1)\*alpha2

return(items)

print(getItems(B1,B2, alpha1, alpha2))

**For B1 = 3, B2 = 4, α1= 0.7 andα2 = 0.6, the output is as follows:**

0.594528

# Experiments

## Changing Bin 1 capacity

import matplotlib.pyplot as plt

Bin1Comp = []

Bin2Comp = []

ShippedProds = []

B1range = list(range(1,11))

for i in B1range:

Bin1Comp.append(getBin1Components(i, B2, alpha1, alpha2))

Bin2Comp.append(getBin2Components(i, B2, alpha1, alpha2))

ShippedProds.append(getItems(i, B2, alpha1, alpha2))

def drawPlots(x, xname):

plt.plot(x, Bin1Comp, 'r^', label = 'Bin 1')

plt.plot(x, Bin2Comp, 'bx', label = 'Bin 2')

plt.legend(loc="upper left")

plt.ylim(0,10)

plt.ylabel("Average Components")

plt.xlim(0,max(x)+x[1]-x[0])

plt.xlabel(xname)

plt.show()

plt.plot(x, ShippedProds, 'go')

plt.ylim(0,1)

plt.xlim(0,max(x)+x[1]-x[0])

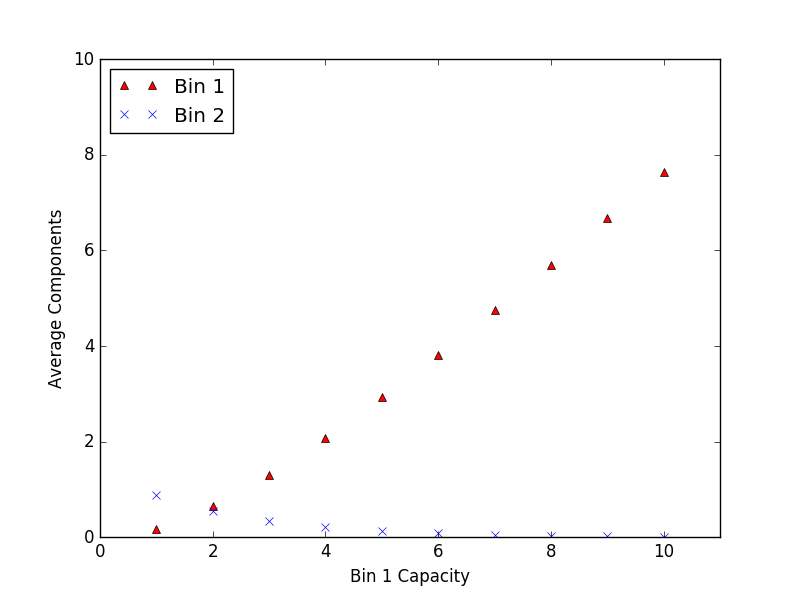
plt.ylabel("Average Shipped Products")

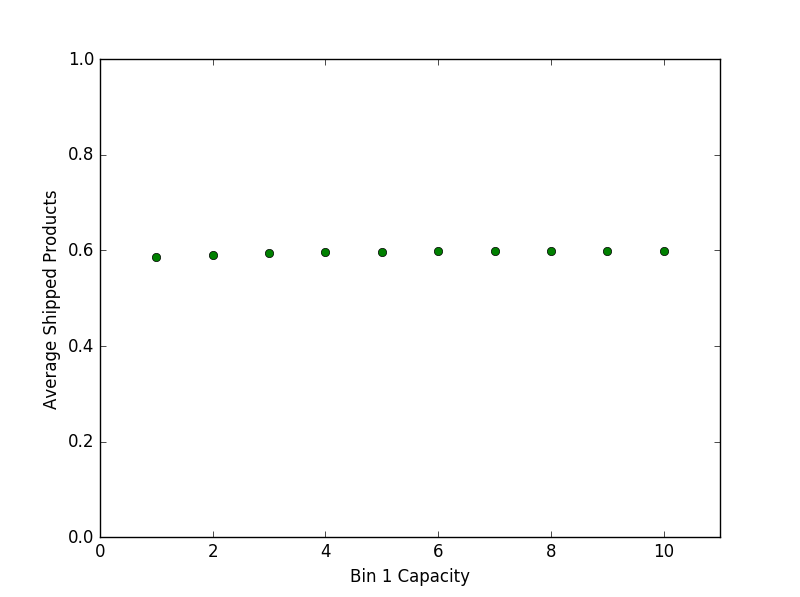
plt.xlabel(xname)

plt.show()

drawPlots(B1range, "Bin 1 Capacity")

**Output:**





## Changing Bin 2 capacity

Bin1Comp = []

Bin2Comp = []

ShippedProds = []

B2range = list(range(1,11))

for i in B2range:

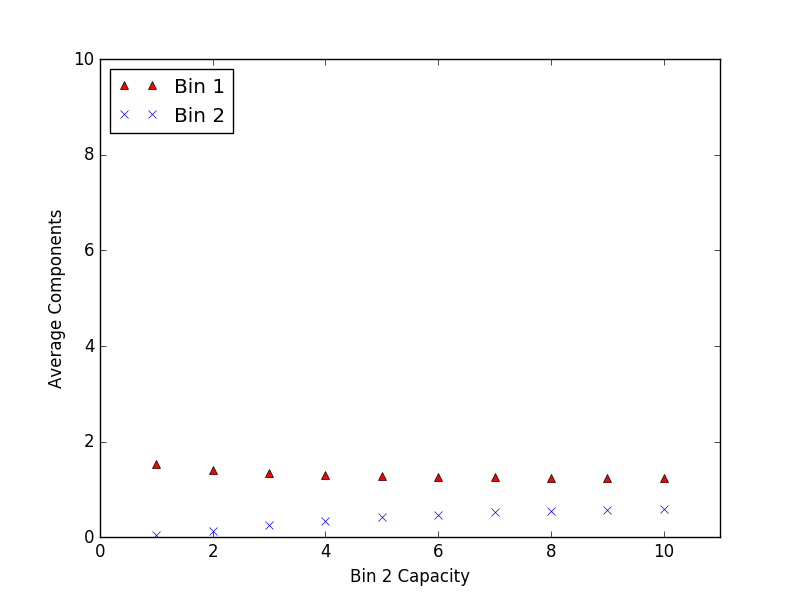
Bin1Comp.append(getBin1Components(B1, i, alpha1, alpha2))

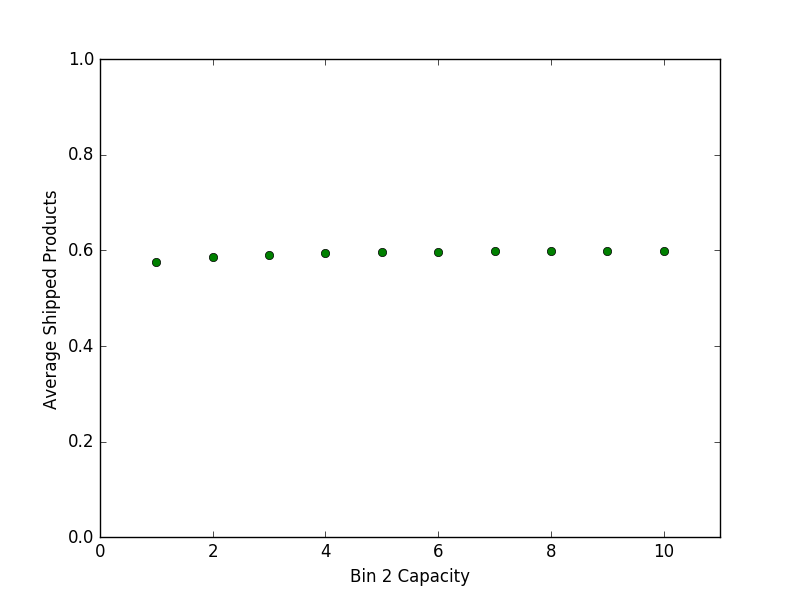
Bin2Comp.append(getBin2Components(B1, i, alpha1, alpha2))

ShippedProds.append(getItems(B1, i, alpha1, alpha2))

drawPlots(B2range, "Bin 2 Capacity")

**Output:**





## Changing Machine 1 Success Rate

Bin1Comp = []

Bin2Comp = []

ShippedProds = []

alpha1range = np.arange(0.1, 1.0, 0.1)

for i in alpha1range:

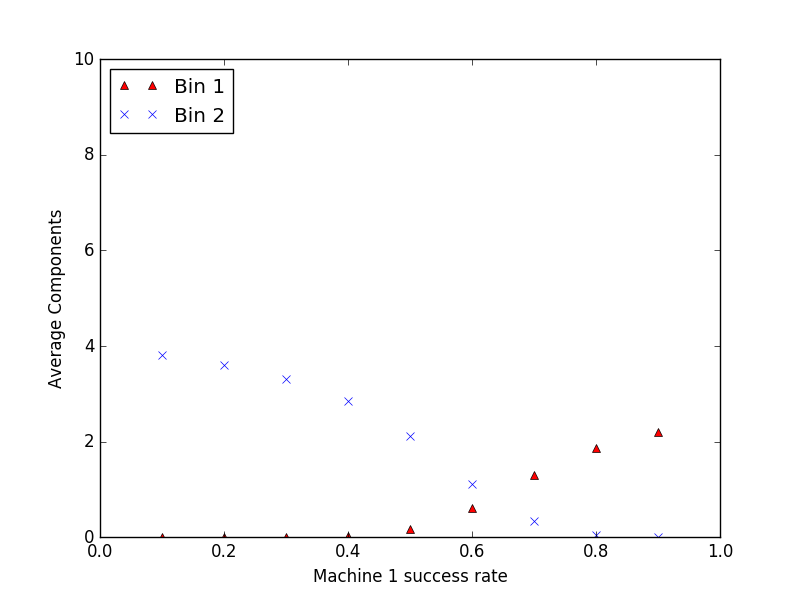
Bin1Comp.append(getBin1Components(B1, B2, i, alpha2))

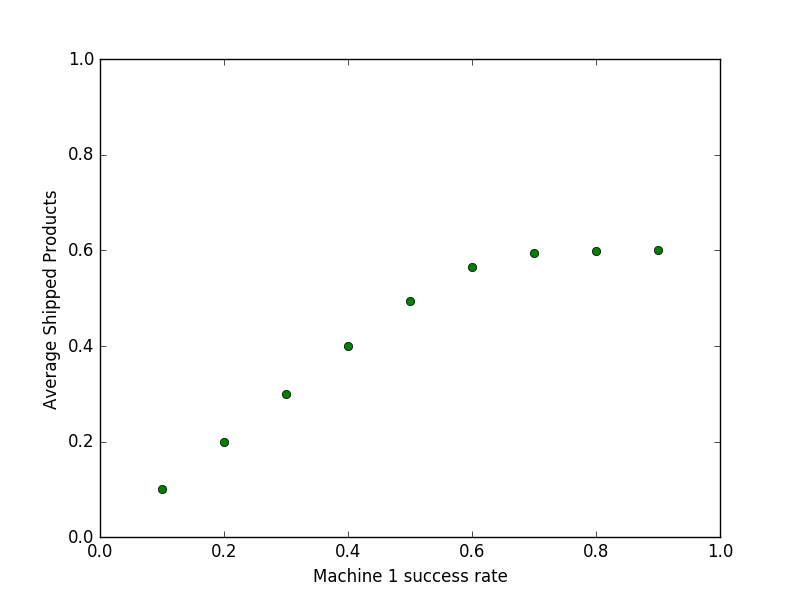
Bin2Comp.append(getBin2Components(B1, B2, i, alpha2))

ShippedProds.append(getItems(B1, B2, i, alpha2))

drawPlots(alpha1range, "Machine 1 success rate")

**Output:**





## Changing Machine 2 Success Rate

Bin1Comp = []

Bin2Comp = []

ShippedProds = []

alpha2range = np.arange(0.1, 1.0, 0.1)

for i in alpha2range:

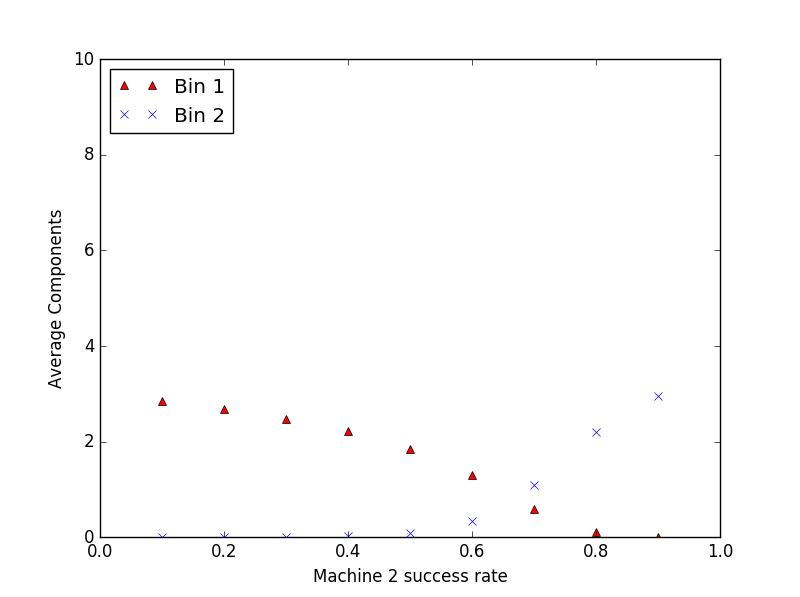
Bin1Comp.append(getBin1Components(B1, B2, alpha1, i))

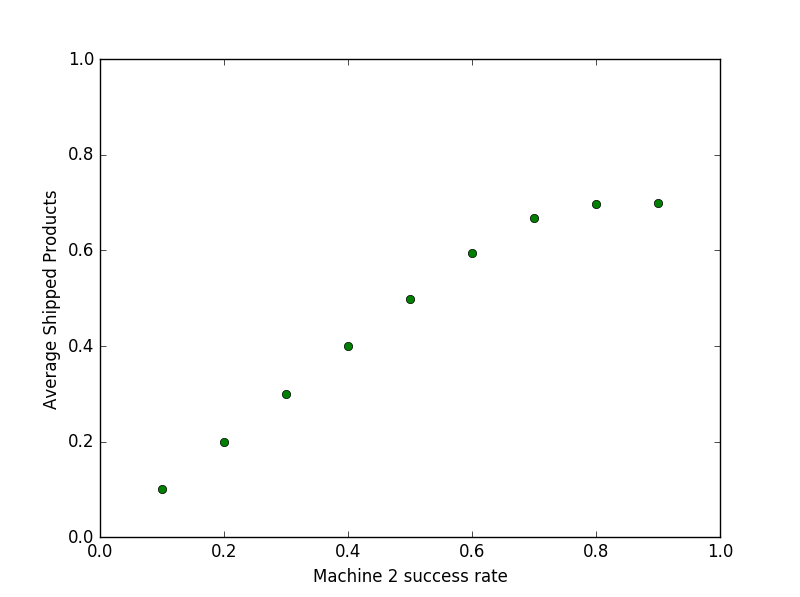
Bin2Comp.append(getBin2Components(B1, B2, alpha1, i))

ShippedProds.append(getItems(B1, B2, alpha1, i))

drawPlots(alpha2range, "Machine 2 success rate")

**Output:**





# Observations and Inferences

As the bin 1 capacity increase, the average components in it also increase while those in bin 2 steadily decline. Changes to the bin 2 capacity show little effect on the number of average components. This is because the probability of machine 1 producing a non-defective component is greater than the probability for machine 2. So, over the long run, the probability of Bin 2 getting filled up diminishes while there is no such limiting restriction on Bin 1. Bin capacities have virtually no effect on the average products shipped per hour. This is explained by changing α1 andα2.

The lesser of the two probabilities of machines producing a non-defective product is the limiting probability of assembling a product in an hour. In other words, the ‘slower’ of the machines is the ‘speed-determining’ machine. This can be seen by changing the values of α1 andα2. Since only 1 product can be assembled in an hour, the average number of products shipped per hour is approximately the same as the lesser of the two values of α1 andα2.

Also, as the probability of a machine producing a non-defective component increases, so does the long-run average number of components in the corresponding bin while the long-run average number of components in the other bin decreases.