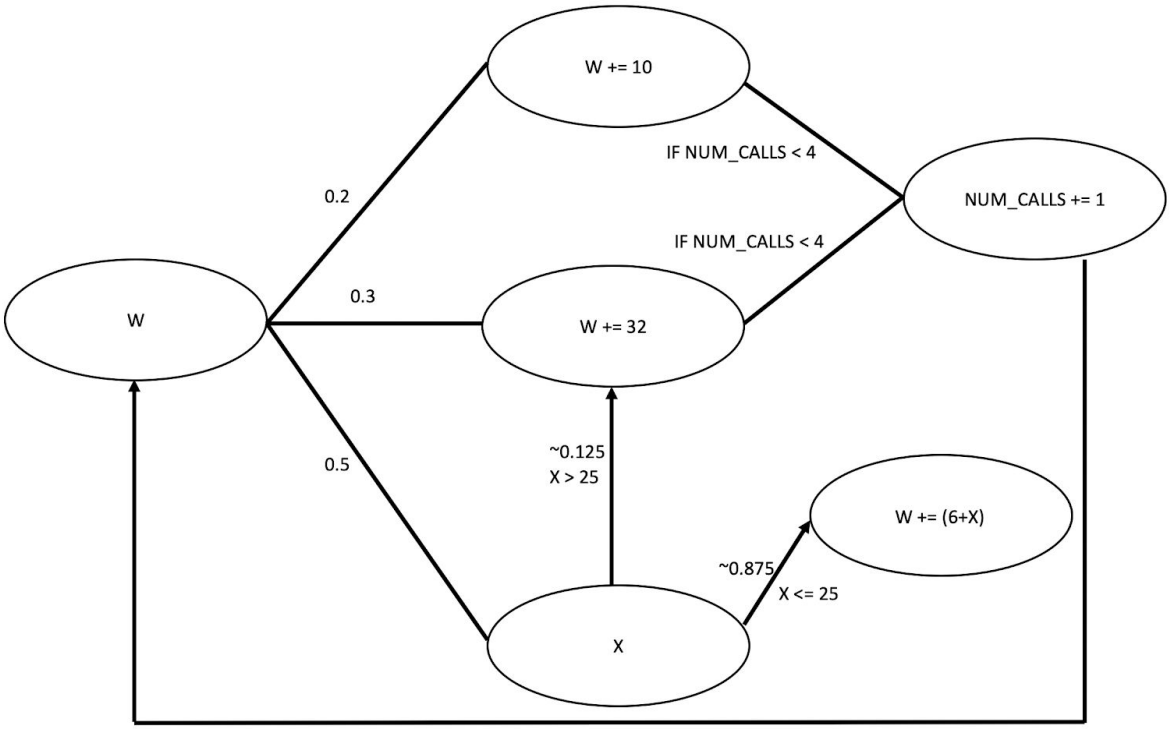
Saiteja Bevara (sb2xf) and Rithik Yelisetty (ry9bf)

APMA 3100 Project 2

Section 1 - Formulate a Model

This problem represents a customer service representative for a high speed internet company calling customers to determine their satisfaction level with their service. The representative takes t_{on} seconds to turn on her phone and dial a customer phone number. If the customer's line is busy, it will take t_{busy} seconds to detect this behavior. It also takes t_{wait} seconds to ring a total of n_{rings} times before concluding that no one will answer the phone. In either of these scenarios, it will take the representative t_{end} seconds to end the phone call. The probability that the customer's line is busy is p_{busy} and the probability that the customer is unable to answer the phone is p_{absent}. The remaining probability is p_{answer} which represents the customer answering the call within X seconds. X is a continuous random variable with an exponential distribution with a mean of 12 seconds. The CDF, F(X), of this function can be represented by the following equation: $F(X) = 1 - e^{-\frac{1}{12}X}$. The inverse of this function will be $F^{-1}(u) = -12 * ln(1-u)$. In this simulation, t_{on} = 6 seconds, t_{busy} = 3 seconds, t_{wait} = 25 seconds, n_{rings} = 5 rings, t_{end} = 1 second, p_{busy} = 0.2, p_{absent} = 0.3, p_{answer} = 0.5. Below is our tree diagram, which represents the calling process. The variable W, initialized to 0, in the tree diagram represents the total time spent by the representative calling one customer. The NUM_CALLS variable should be initialized to 0. The calling process will end when the customer answers the phone or the representative has called four times unsuccessfully.



Section 2 - Ad-hoc Experiment

We performed an ad-hoc experiment to determine the values of t_{on}, t_{busy}, t_{wait}, t_{end}, and X. The values are recorded in the table shown along with the averages and the values that are specified in the problem statement.

Trial	t_{on} (sec)	t_{busy} (sec)	t_{wait} (sec)	t_{end} (sec)	X (sec)
1	4.53	5.21	30.28	0.49	8.31
2	3.91	3.24	33.74	0.40	12.85
3	4.35	3.92	32.14	0.44	7.25
4	3.87	7.13	29.99	0.42	16.78
Average	4.165	4.875	31.5375	0.4375	11.2975
Experimental Values	6	3	25	1	12

Section 3 - Design a Monte-Carlo Algorithm

In order to create a Monte-Carlo Simulation Algorithm, we developed code to generate n realizations of our random variable, W. First, we decided to convert our tree diagram into computer code. To do this, we created a Python program that had two primary methods: a random number generator and then a single iteration simulator. The random number generator followed the procedure outlined in the problem statement. The random number generator used a constant x_0 (starting value), a (multiplier), c (increment) and K (modulus). The starting values for these variables were the following: $x_0 = 1000$, $a = 24693$, $c = 3517$ and $K = 2^{15}$. The following recursive formula was then used in the random number generator: $x_i = (a * x_{i-1} + c) \% K$. Using this formula, we were able to generate the ith random numbers as $u_i = x_i / K$, a decimal representation. For example, $u_{51} = 0.062835693359375$, $u_{52} = 0.7091064453125$, and $u_{53} = 0.072784423828125$. These random numbers served as the basis for our tree diagram code, as we used them to determine which case we entered. If the random number was less than 0.2, we determined the customer was busy. When the customer was busy, we added 10 to w (the total wait time). We then re-distributed the random number by dividing by 0.2 to ensure that the next random number would be randomized. If the random number was between 0.2 and 0.5, the customer is unavailable to answer the call. In this case, we added 32 seconds to w and re-randomized the random variable by subtracting by 0.2 and dividing by 0.3. If the random number was greater than 0.5, the customer was able to answer the call within X seconds. We determined the value of X by plugging it into the formula found for $F^{-1}(u)$. We then added 6+x to the value of w if $X < 25$. If $X > 25$, we instead counted this as an unsuccessful call and added 32 to w. We repeated this process until the num_calls variable reached 4 or the customer answered

the call. This entire process represented one realization of the variable W. Then, for the specified number of iterations (n), we repeated this process for n independent realizations, where each time i from u_i was the ith realization being simulated.

Section 4 and 5 - Simulate and Estimate

The process was simulated for n = 1000 iterations. From the sample of W, some statistics were determined:

Mean: 54.20047469315636

Median: 49.79085608456724

First Quartile: 23.35234022104266

Third Quartile: 84.0

W <= 15: 0.026

W <= 20: 0.162

W <= 30: 0.381

W > 40: 0.548

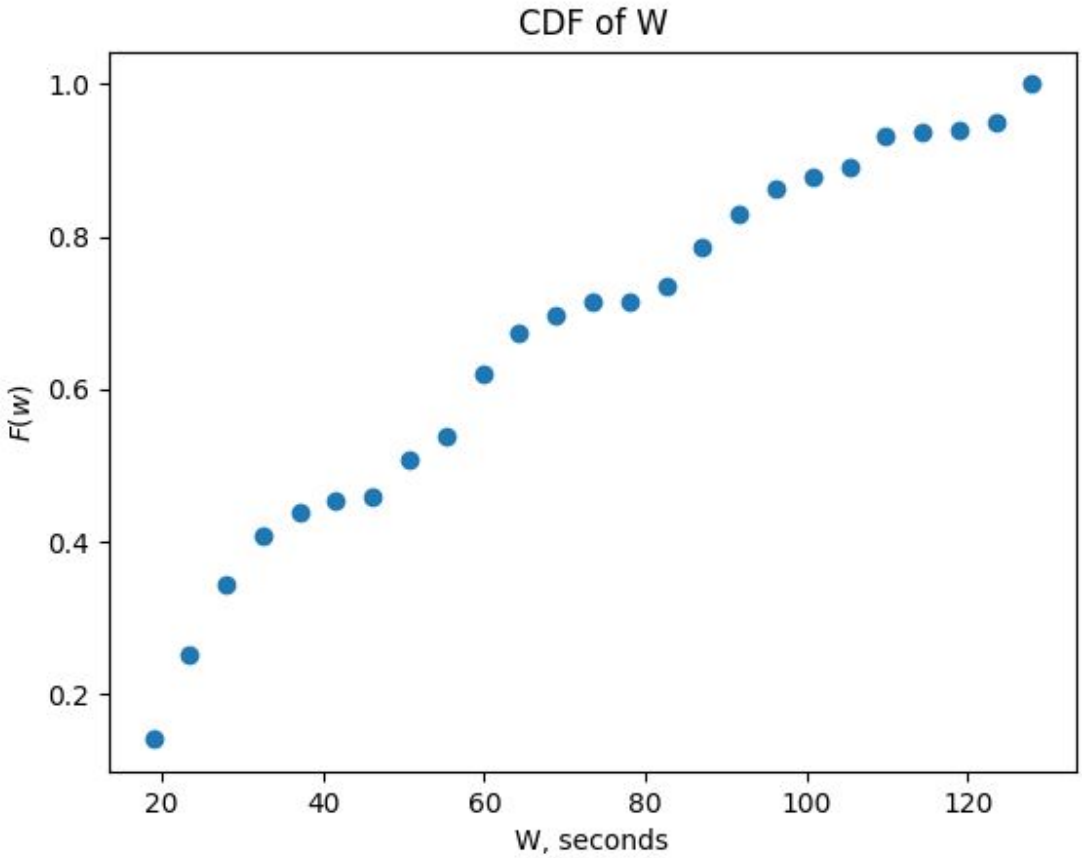
W > W5 (60): 0.375

W > W6 (110): 0.067

W > W7 (125): 0.046

Section 6 - Analyze

The median of about 49.79 seconds is significantly lower than the mean of about 54.2 seconds. This means that there is more data to the right side of the median, so the data is right skewed. The sample space of W ranges from 6 seconds on the lower bound to an upper bound of 128. This is due to the fact that at most, there will be 4 unsuccessful calls each lasting 32 seconds.



W could be an exponential random variable, as the shape of the curve does emulate that of a random variable with CDF $F(X) = 1 - e^{-\lambda x}$.

Section 7 - Comments

The most challenging part of this assignment was coming up with the correct formulation for this model, specifically coming up with the correct tree diagram. The least challenging part was coding the random number generator. The random number generator was easy to understand and was pretty simple to implement, both in an iterative and recursive fashion. The most time consuming step was designing the algorithm for running the simulation as this involved converting our tree diagram into actual code. The least time consuming step was actually running

the simulation on our computer and calculating the mean, median, first quartile and third quartile data. There may be some error in our calculations due to inconsistencies in the random number generation, however, we believe that the error has been mostly reduced by using 1000 trials.

Code

In order to complete this project, we wrote a Python program. This is available at https://github.com/rithik/Monte-Carlo.