Behind the Mask: A Mathematical Analysis of Persona 5: The Phantom X's Gacha System

Varun Nadkarni

1 Introduction

Gacha games, a popular genre in gaming, rely on randomized mechanics to distribute in-game rewards, often resembling lottery or loot box systems. This term comes from the gachapon machines, found in Japan, where you entered money and had a chance to get a capsule with what you wanted. These games use probability-based models to determine outcomes. Understanding the mathematical foundations behind gacha mechanics is essential for analysing fairness, expected costs, and player behaviour.

I have been really interested in diving into the gacha mechanics for a game I have been playing for almost a year called Persona 5: The Phantom X.

I am especially interested in analysing the mechanics of this game as it has 2 different systems you can use to roll for the newest characters. Analysing the differences and using math to find the optimal rolling strategy can help me decide how to spend my game currency effectively as a free to play player.

2 Description of the Systems

To try and model the systems, first, I need find adequate descriptions and details to fully understand them. These details are gotten from in-game descriptions and from analysis of data from pulling.

2.1 Terms to understand

There are 4 rarities in this game: 2$\star$, 3$\star$, 4$\star$ and 5$\star$, in increasing order of their rarity. I will be focusing only on 5$\star$ characters and their pull rates.

To understand the descriptions of these systems properly, I must explain common terms used by players of gacha games for the systems:

- **Limited:** a character that can only be gotten through pulling in a limited time. In the game, there is only 1 limited 5$\star$ character per banner and is the character you are looking to get.
- **Pull:** a pull refers to one chance to attempt to get the character.
- **Pity:** this term refers to the numbers of pulls since the last 5$\star$ character pulled. The term pity is used as the games implement a pity system.
- **Base Rate:** this is the base chance to get a 5$\star$ character per pull.
- **Pity System:** a system implemented to prevent terrible loss streaks while pulling.
- **Soft Pity:** this refers to the pity number where the rate of getting the 5$\star$ starts increasing from the base rate and the rate of increase is studied later in the essay.
- **Hard Pity:** this refers to the pity number where the rate of getting the 5$\star$ reaches 100%, thus guaranteeing you a 5$\star$ character.
- **Rate-up Chance:** This is the chance of getting the limited 5$\star$ when you get a 5$\star$ character. The two most common systems are 100% rate up chance (known as guaranteed system) and 50% rate up chance (known as a 50/50 system).

2.2 Exclusive Bond (Guarantee Banner)

Base 5⋆ Rate: 0.2% (per pull).
Rate-Up Chance: 100% (every 5⋆ is the limited character).
Pity System:

- Soft Pity starts at 70 pulls.
- Hard Pity at 110 pulls.

Additionally, you receive a non-limited 5⋆ character every 165 pulls.
In-game Description
In this event contract EXCLUSIVE BOND, the probability to obtain 5⋆ is as follows: The base probability of 5⋆ Character is 0.2%; consolidated probability (incl. guarantee) is 1.2%.
When obtaining a 5⋆ Character, there is a 100% chance to obtain the current RATE UP 5⋆ Character.

2.3 Fated Bond (50/50 Banner)

Base 5⋆ Rate: 0.8% (per pull).
Rate-Up Chance: 50% (every 5⋆ is the limited character).
Pity System:

- Soft Pity starts at 65 pulls.
- Hard Pity at 80 pulls.

In-game Description
In this event Contract FATED BOND, the probability to obtain 5⋆ is as follows: The base probability of 5⋆ Character is 0.8%; consolidated probability (incl. guarantee) is 1.8%.
When obtaining a 5⋆ Character, there is a 50% chance to obtain the current RATE UP 5⋆ Character.

3 Modelling the Systems

We can approximate the gacha system of the game using piecewise geometric distribution.

3.1 Fated Bond

To start, we must define the domain of the function $P(X = k)$ where k is the number of pulls since last 5$\star$ character.

$$Domain\ of\ P(X = k) : \{x : x \in Z^+, 0 < x \leq 80\} \tag{1}$$

Following the rules defined above about base rate, soft pity and hard pity, we can define 3 different regions of the domain.

1. $0 < x \leq 64$
2. $65 \leq x \leq 79$
3. $x = 80$

1. **Region 1**
   In this region, there is a fixed base rate of 0.2% from 1 to 64 pulls.

   $$p_1 = \frac{0.8}{100} = 0.008 \tag{2}$$

   Using geometric distribution, we can find the probability mass function of the first region.

   $$P(X = k) = (1 - p_1)^{k-1} \times p_1 \tag{3}$$

   $$P(X = k) = (1 - 0.008)^{k-1} \times 0.008 \tag{4}$$

   To find the expected value of this region, which will be used later for calculating the overall expected value, we use the following formula.

   $$E[X_1] = \sum_{k=1}^{64} k \cdot (0.992)^{k-1} \cdot 0.008 \tag{5}$$

   $$E[X_1] = 11.97\ pulls \tag{6}$$

2. **Region 2**
   It is harder to find the expected value of this region as the rate is changing every trial. We will assume that this increase is linear. Hence, the formula for the probability is:

   $$p_k = 0.008 + (0.992 \cdot \frac{k - 64}{80 - 64}) \tag{7}$$

Hence, PMF for this region is:

$$P(X = k) = p_k \times \prod_{r=65}^{k-1} (1 - p_r) \tag{8}$$

Hence, we can find the expected value using the same summation value. However, we need to add 64 to the value as it takes 64 pulls to reach this region.

$$E[X_2] = 64 + (\sum_{n=65}^{79} ((n - 64) \times p_n \times \prod_{k=65}^{n-1} (1 - p_k))) \tag{9}$$

Calculating using Desmos, we get an expected value:

$$E[X_2] = 68.62 \; pulls \tag{10}$$

3. **Region 3**
   The third region is the easiest as the probability is 100%.

$$p_3 = 1 \tag{11}$$

Hence, the expected value is also just 1 and the 79 earlier trials.

$$E[X_3] = 79 + 1 = 80 \tag{12}$$

Calculating Overall Expected Value

Now to find the total expected value, we use the formula, where m is the number of trials in region 1 and j is the number of trials in region 2:

$$E[X] = E[X_1] + (1-p_1)^m \times E[X_2] + (1-p_1)^m \times (\prod_{k=m+1}^{j} (1-p_k)) \times E[X_3] \tag{13}$$

We multiply the expected values of the regions with the chance of all earlier trials failing to weight their contribution to the expected value accurately. Hence:

$$E[X] = 11.97 + (1-0.008)^{64} \times 68.62 + (1-0.008)^{64} \times (\prod_{k=65}^{79} (1-p_k)) \times 80 \tag{14}$$

Doing the calculations, we get:

$$E[X] = 53.00 \; pulls \tag{15}$$

Accounting for the 50/50 system

This system has a 50% rate-up chance. Hence, we have to use conditional probability to calculate the total expected value for getting a limited 5$\star$.

1. **Case 1: Win the 50/50**
   In this case, your expected value is just 53 pulls.

2. **Case 2: Lose the 50/50 and go to Guarantee**
   In this case, your expected value is just 53 (for getting a 5 $\star$ and losing) + 53 (for reaching the guarantee) = 106 pulls.

Hence, we can find the complete expected value by doing the following:

$$E[X|2X] = 0.5 \times 53 + 0.5 \times 106 \tag{16}$$

$$E[X|2X] = 79.5 \; pulls \tag{17}$$

3.2 Exclusive Bond

To start, we must again define the domain of the function $P(X = k)$ where k is the number of pulls since last 5$\star$ character.

$$Domain \; of \; P(X = k) : \{x : x \in Z^+, 0 < x \leq 110\} \tag{18}$$

Following the rules defined above about base rate, soft pity and hard pity, we can define 3 different regions of the domain.

1. $0 < x \leq 69$
2. $70 \leq x \leq 109$
3. $x = 110$

1. **Region 1**
   In this region, there is a fixed base rate of 0.2% from 1 to 64 pulls.

   $$p_1 = \frac{0.2}{100} = 0.002 \tag{19}$$

   Using geometric distribution, we get the PMF:

   $$P(X = k) = (1 - 0.002)^{k-1} \times 0.002 \tag{20}$$

   To find the expected value of this region:

   $$E[X_1] = \sum_{k=1}^{69} k \cdot (0.998)^{k-1} \cdot 0.002 \tag{21}$$

   $$E[X_1] = 4.41 \; pulls \tag{22}$$

2. **Region 2**
   Again, we assume that the increase is linear. Hence, the formula for the probability is:

$$p_k = 0.002 + (0.998 \cdot \frac{k-69}{110-69}) \tag{23}$$

   Hence, PMF for this region is:

$$P(X = k) = p_k \times \prod_{r=70}^{k-1} (1 - p_r) \tag{24}$$

   Hence, we can find the expected value using the same summation value.

$$E[X_2] = 69 + (\sum_{n=70}^{109} ((n-69) \times p_n \times \prod_{k=70}^{n-1} (1 - p_k))) \tag{25}$$

   Calculating using Desmos, we get an expected value:

$$E[X_2] = 76.64 \; pulls \tag{26}$$

3. **Region 3**
   The third region is the easiest as the probability is 100%.

$$p_3 = 1 \tag{27}$$

   Hence, the expected value is also just 1 and the 79 earlier trials.

$$E[X_3] = 109 + 1 = 110 \tag{28}$$

Calculating Overall Expected Value
Now to find the total expected value, we use the same formula:

$$E[X] = E[X_1] + (1-p_1)^m \times E[X_2] + (1-p_1)^m \times (\prod_{k=m+1}^{j} (1-p_k)) \times E[X_3] \tag{29}$$

$$E[X] = 4.41 + (1-0.002)^{69} \times 76.64 + (1-0.002)^{69} \times (\prod_{k=70}^{109} (1-p_k)) \times 110 \tag{30}$$

Doing the calculations, we get:

$$E[X] = 71.16 \; pulls \tag{31}$$

4 Using Code Models to Verify Numbers

To make sure my calculations were valid and held up, I made their programs in Python to see if the same expected value appears. Additionally, we can see what the mode will be, finding more useful information to decide which system is better.

4.1 Fated Bond

The program for this system is:

```
import random
import matplotlib.pyplot as plt
from collections import Counter

# Simulation parameters
pulls = 1000000000
fS = 0
lfS = 0
pity = 1
guran = False
sum_pity = 0
pity_values = []

# Pull simulation
for i in range(1, pulls + 1):
    if pity < 65:
        x = 0.008
    elif pity < 80:
        x = 0.008 + (0.992 * ((pity - 65) / (80 - 65)))
    else:
        x = 1

    if random.randint(1, 1000) <= x * 1000:
        fS += 1
        if guran is False:
            if (random.randint(1, 100) <= 50):
                lfS += 1
            else:
                guran = True
        else:
            lfS += 1
            guran = False
        sum_pity += pity
```

```
            pity_values.append(pity)
            pity = 1
        else:
            pity += 1

# Average pity
avg = sum_pity / fS
bavg = sum_pity / lfS
print(f"Pulls: {pulls}\nFive Stars: {fS}\nAVERAGE: {avg:.2f}\nTHE
    BETTER AVERAGE: {bavg:.2f}")

# Count and plot pity frequencies
pity_counter = Counter(pity_values)
sorted_pity = sorted(pity_counter.items())
x_vals, y_vals = zip(*sorted_pity)

plt.figure(figsize=(12, 6))
plt.bar(x_vals, y_vals, color='skyblue')
plt.title("Frequency of Pity Values at 5-Star Pulls")
plt.xlabel("Pity Value")
plt.ylabel("Number of Times Pity Was Hit")
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

As you can see in the program, I ran the simulation for 1 billion iterations. The following were the results.

```
Pulls: 1000000000
Five Stars: 18692717
Average per 5 star: 53.50
Average per limited 5 star: 80.25
```

Figure 1: Fated Bond Simulation Results

We can also graph the values calculated by the simulation as a histogram with pity values on the x-axis.

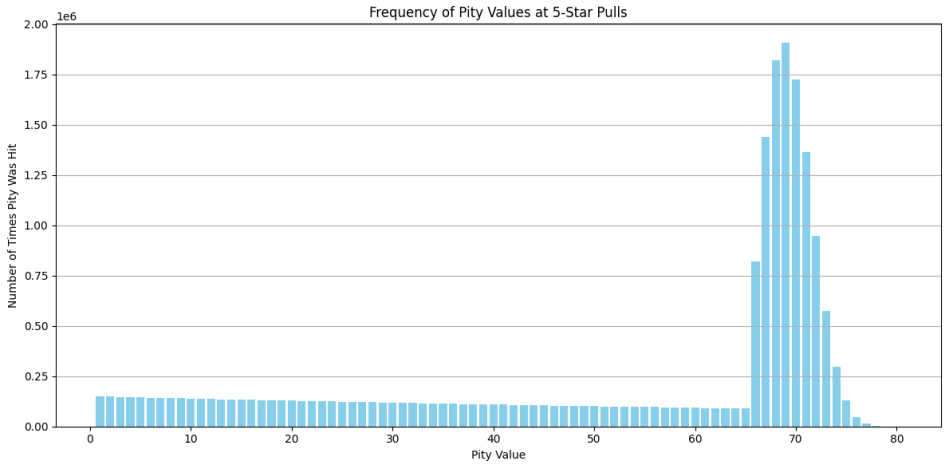

Figure 2: Frequency v/s Pity Graph for Fated Bond

Using the graph, we can see the mode of the graph is 69 pulls. Also, in the results, you can see that the average per 5⋆ found is 0.5 pulls away from the calculated value. Additionally, the average per limited 0.75 pulls away from the calculated value. We can also confirm if our model is accurate by calculating its consolidated probability and comparing with the given value.

I calculated the consolidated probability by finding the number of pulls taken to get 10 5⋆ characters, then repeat this 9 more times and find their average. Then, we can get the final value of consolidated probability by:

$$Consolidated\ Probability = \frac{10}{Avg.\ of\ 10\ trials} * 100 \tag{32}$$

I calculated this using code, which is as follows:

```
import random
import matplotlib.pyplot as plt

# Pull simulation
def fated(res):
    pulling = 0
    fS = 0
    lfS = 0
    pity = 1
    guran = False
    sum_pity = 0
```

```
    pity_values = []
    pulls = 1000000000
    for i in range(1, pulls + 1):
        pulling += 1
        if fS == res:
            break
        if pity < 65:
            x = 0.008
        elif pity < 80:
            x = 0.008 + (0.992 * ((pity - 65) / (80 - 65)))
        else:
            x = 1

        if random.randint(1, 1000) <= x * 1000:
            fS += 1
            if guran is False:
                if (random.randint(1, 100) <= 50):
                    lfS += 1
                else:
                    guran = True
            else:
                lfS += 1
                guran = False
            sum_pity += pity
            pity_values.append(pity)
            pity = 1
        else:
            pity += 1
    return pulling

def run(f_name, f, pieces, tries):
    print("==== " + f_name + " ====")
    ban = [f(pieces) for _ in range(tries)]
    avg = sum(ban) / len(ban)
    print("average: " + str(avg))
    print("cons. probability: " + str(pieces/avg * 100))
    print("max: " + str(max(ban)))
    print("")

run ("fated", fated, 10, 10)
```

The results of the code were:

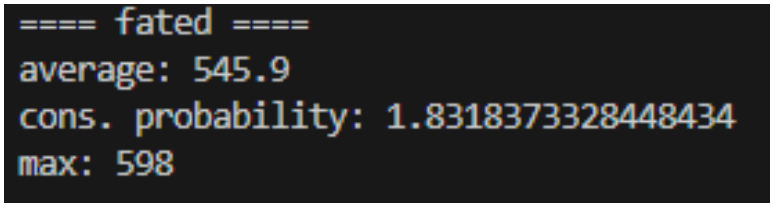

Figure 3: Fated Bond Consolidated Probability Calculation Results

The calculated consolidated probability from the code is 1.83% which is 0.03% away from the given value of 1.8%, hence, we can conclude that the model is very close to, if not the exact same as the in-game system.

4.2 Exclusive Bond

The program for this system is:

```
import random
import matplotlib.pyplot as plt
from collections import Counter

# Simulation parameters
pulls = 1000000000
fS = 0
pity = 1
sum_pity = 0
pity_values = []

# Pull simulation
for i in range(1, pulls + 1):
    if pity < 69:
        x = 0.002
    elif pity < 110:
        x = 0.002 + (0.998 * ((pity - 69) / (110 - 69)))
    else:
        x = 1

    if random.randint(1, 1000) <= x * 1000:
        fS += 1
        sum_pity += pity
        pity_values.append(pity)
        pity = 1
    else:
        pity += 1
```

```
# Average pity
avg = sum_pity / fS
print(f"Pulls: {pulls}\nFive Stars: {fS}\nAVERAGE: {avg:.2f}")

# Count and plot pity frequencies
pity_counter = Counter(pity_values)
sorted_pity = sorted(pity_counter.items())
x_vals, y_vals = zip(*sorted_pity)

plt.figure(figsize=(12, 6))
plt.bar(x_vals, y_vals, color='skyblue')
plt.title("Frequency of Pity Values at 5-Star Pulls")
plt.xlabel("Pity Value")
plt.ylabel("Number of Times Pity Was Hit")
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

Again, I ran the simulation for 1 billion iterations. The following were the results.

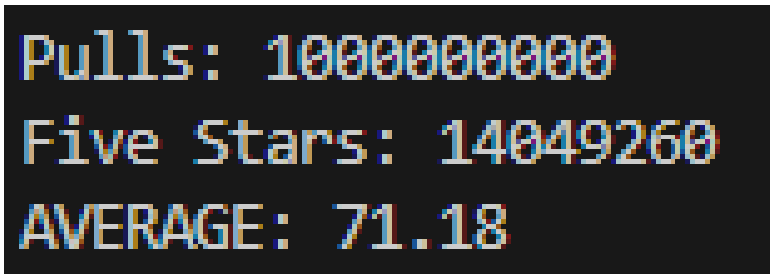

Figure 4: Exclusive Bond Simulation Results

On graphing, we get:

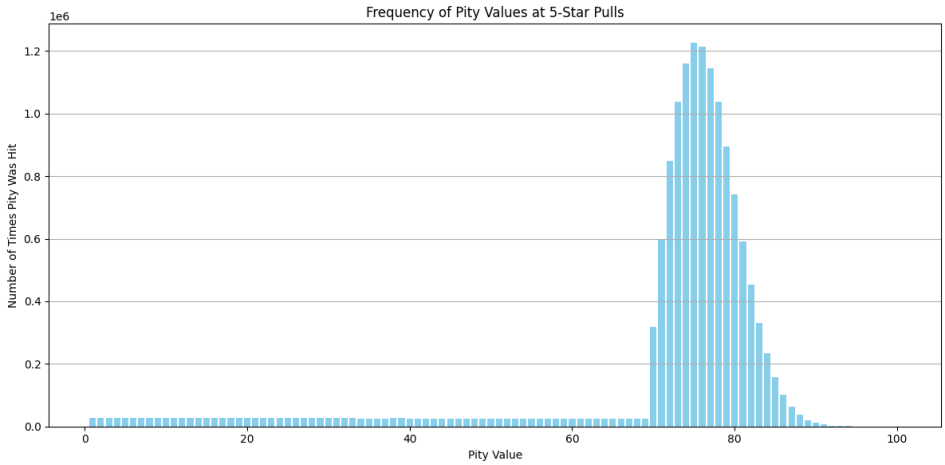

Figure 5: Frequency v/s Pity Graph for Exclusive Bond

Using the graph, we can see the mode of the graph is 75 pulls. In the results, the average per 5⋆ found is 0.02 pulls away from the calculated value. We can again confirm if our model is accurate using consolidated probability using the same method. I calculated this using code, which is as follows:

```
import random
import matplotlib.pyplot as plt


# Pull simulation
def exclusive(res):
  pulling = 0
  pulls = 1000000000
  fS = 0
  pity = 1
  sum_pity = 0
  pity_values = []

# Pull simulation
  for i in range(1, pulls + 1):
    pulling += 1
    if fS == res:
      break
    if pity < 69:
        x = 0.002
    elif pity < 110:
```

```
        x = 0.002 + (0.998 * ((pity - 69) / (110 - 89)))
    else:
        x = 1

    if random.randint(1, 1000) <= x * 1000:
        fS += 1
        sum_pity += pity
        pity_values.append(pity)
        pity = 1
    else:
        pity += 1
  return pulling

def run(f_name, f, pieces, tries):
    print("==== " + f_name + " ====")
    ban = [f(pieces) for _ in range(tries)]
    avg = sum(ban) / len(ban)
    print("average: " + str(avg))
    print("cons. probability: " + str(pieces/avg * 100))
    print("max: " + str(max(ban)))
    print("")

run ("exclusive", exclusive, 10, 10)
```

The results of the code were:

```
==== exclusive ====
average: 7115.33
cons. probability: 1.40541619292429 17
max: 7538
```

Figure 6: Exclusive Bond Consolidated Probability Calculation Results

The calculated consolidated probability from the code is 1.4% which is 0.2% away from the given value of 1.2%. Since the probabilities are so low, 0.2% is a big difference, from which I concluded that the increase of the rates from soft pity were, most likely, not linear.

5 Remodelling Exclusive Bond

Since my consolidated probability was very off, I tried other relations to see which would be more accurate to use here. Since my consolidated probability was higher than the given value, I can infer that the increase is more gradual at the start and becomes greater at the end, making the increase exponential and lowering the consolidated probability.

The standard form of an exponential function is:

$$f(x) = a \cdot e^{bx} \tag{33}$$

We need it to pass through the points (69, 0.002) and (110, 1). Firstly, we can shift the graph 69 units to the right and replace a with our base rate. This makes sure it passes through (69, 0.002).

$$f(x) = 0.002 \cdot e^{b(x-69)} \tag{34}$$

We can use the same scaling method as last time to make it such at the x term of the exponent equals to 1 at 110.

$$f(x) = 0.002 \cdot e^{b(\frac{x-69}{110-69})} \tag{35}$$

Now, to find the value of b, we can equate it to (110, 1).

$$1 = 0.002 \cdot e^{b(\frac{110-69}{110-69})} \tag{36}$$

$$1 = 0.002 \cdot e^{b} \tag{37}$$

$$e^{b} = \frac{1}{0.002} \tag{38}$$

Now, we can rewrite the above equation using natural logarithm to find the value of b.

$$b = \ln(\frac{1}{0.002}) \tag{39}$$

$$b = \ln(500) \tag{40}$$

So, now we have our function equation:

$$f(x) = 0.002 \cdot e^{\ln(500)\cdot(\frac{x-69}{110-69})} \tag{41}$$

I tweaked the following part of the code and then on running the code with this function:

```
...
# Pull simulation
  for i in range(1, pulls + 1):
    pulling += 1
    if fS == res:
      break
    if pity < 69:
        x= 0.002
    elif pity <= 110:
        a = 0.002 # starting probability
        x = pity - 69
        scale = 110 - 69 # span of ramp
        exponent = (x / scale)
        x= a * math.exp(math.log(1/a) * exponent)
    else:
        x= 1.0


    if random.randint(1, 1000) <= x * 1000:
        fS += 1
        sum_pity += pity
        pity_values.append(pity)
        pity = 1
    else:
        pity += 1
  return pulling
...
```

The output comes in a range between 1.14% to 1.16% consolidated probability. Hence, I had to add another variable to change the rate of increase. I raised the $\frac{x-69}{110-69}$ term by a variable c, which I will call the easing factor. This factor's value will change the rate at which it increases.

$$f(x) = 0.002 \cdot e^{\ln(500)\cdot(\frac{x-69}{110-69})^c} \tag{42}$$

Different values of c affect the graph differently. For example, let's take c = 0.8 and c = 1.2 and graph them using Desmos.

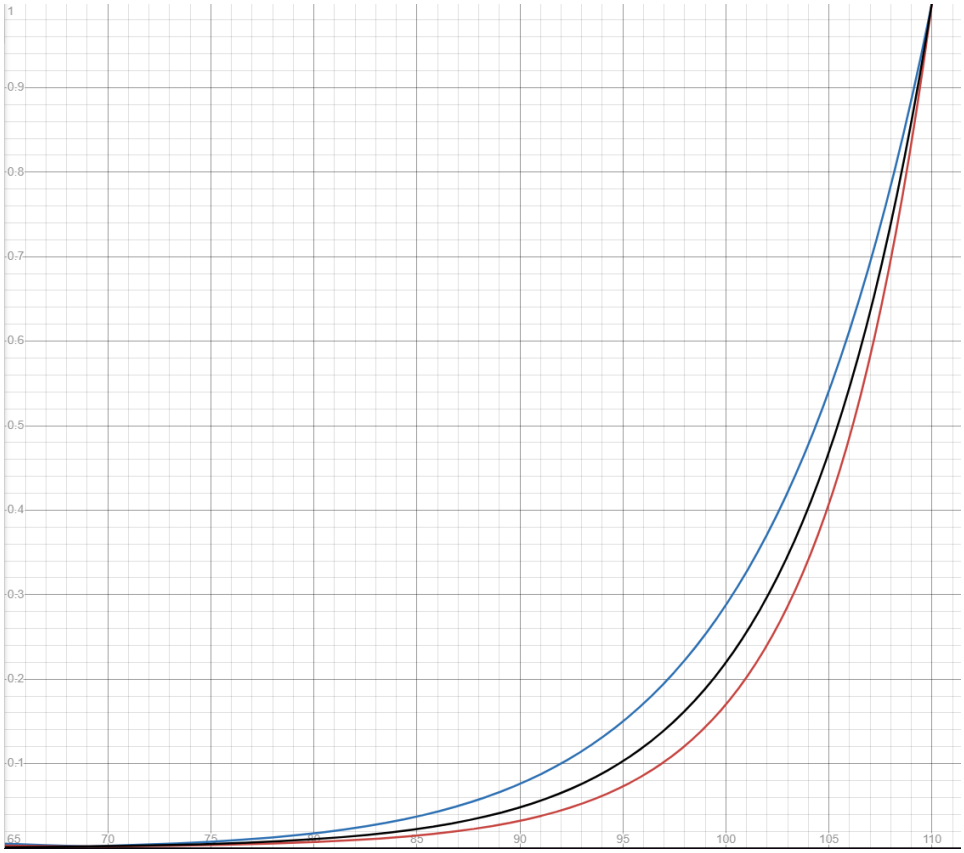

Figure 7: Different values of c

The black line is the function before adding the easing factor. The red line is when c = 1.2 and the blue line is when c = 0.8. The red line is flatter at the start and then shoots up sharply, whereas the blue line rises more gradually and is less flat. Hence, we can infer that:

- $c < 1$: Makes the curve rise more smoothly and earlier.
- $c = 1$: Normal exponential curve.
- $c > 1$: Makes the curve rise sharply towards the end and is flatter earlier.

Since at c = 1, my consolidated probability is lesser than the given 1.2%, I decided to decrease the value of c, so that the rise of probability is more gradual and the consolidated probability value increases. After tweaking the

code to include the c variable and testing multiple different values of c, I eventually landed on c = 0.5. The curve for c = 0.5 looks like this:

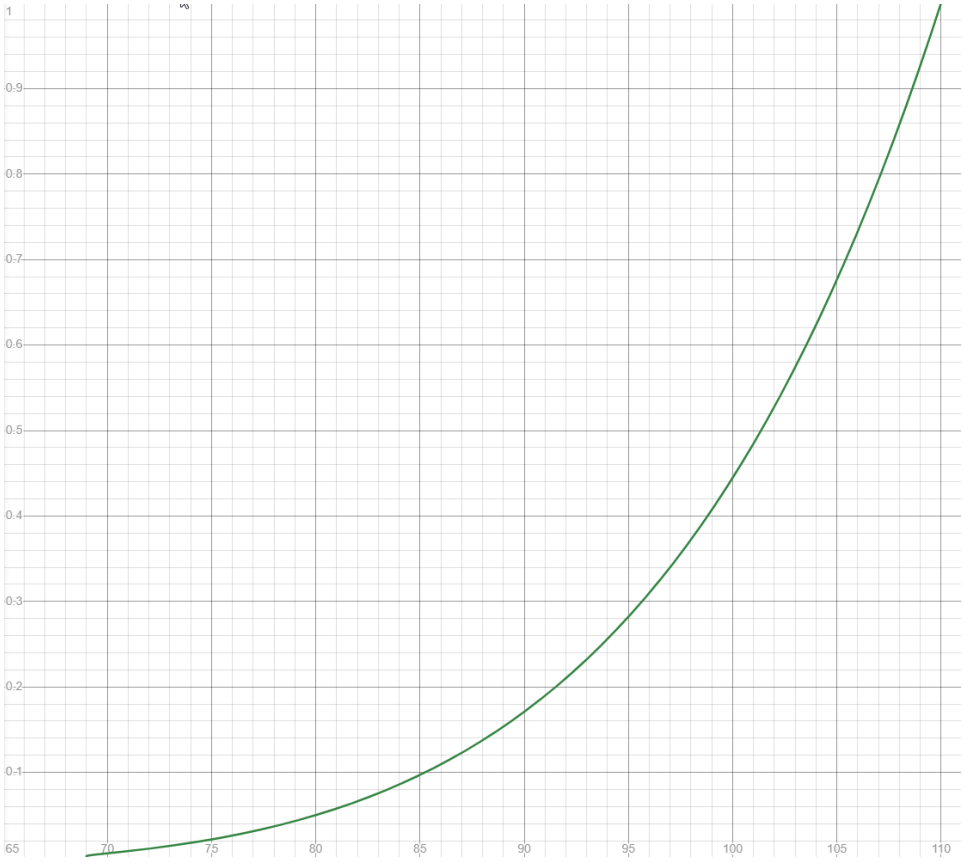

Figure 8: Curve for f(x) when c = 0.5

The modified code is as follows:

```
...
# Pull simulation
  for i in range(1, pulls + 1):
    pulling += 1
    if fS == res:
      break
    if pity < 69:
        x= 0.002
    elif pity <= 110:
        # Custom curved exponential
```

```
            a = 0.002 # starting probability
            x = pity - 69
            scale = 110 - 69 # span of ramp
            c = 0.5 # controls how fast the curve bends upward
            exponent = (x / scale) ** c
            x= a * math.exp(math.log(1/a) * exponent)
        else:
            x= 1.0


        if random.randint(1, 1000) <= x * 1000:
            fS += 1
            sum_pity += pity
            pity_values.append(pity)
            pity = 1
        else:
            pity += 1
    return pulling
...
```

On running the simulation, we get:

```
==== exclusive ====
average: 828.3
cons. probability: 1.2072920439454304
max: 873
```

Figure 9: New Exclusive Bond Model Consolidated Probability Calculation

The value of the consolidated probability ranged from 1.20% to 1.26%. Hence, I decided to land on this value for my pity function. Hence, the function for the rate of obtaining at 5⋆ from 70 to 109 pulls is:

$$f(x) = 0.002 \cdot e^{\ln(500)\cdot(\frac{x-69}{110-69})^{0.5}} \tag{43}$$

Now, we can calculate the expected value for the Exclusive Bond again. The values for region 1 and 3 do not change so I only recalculated the expected value for region 2.

$$E[X_2] = 69 + (\sum_{n=70}^{109}((n-69) \times f(n) \times \prod_{k=70}^{n-1}(1-f(k)))) \tag{44}$$

$$E[X_2] = 85.55 \; pulls \tag{45}$$

Now to find the total expected value, we use the same formula:

$$E[X] = 4.41 + (1-0.002)^{69} \times 85.55 + (1-0.002)^{69} \times (\prod_{k=70}^{109} (1-f(k)) \times 110 \tag{46}$$

Doing the calculations, we get:

$$E[X] = 78.92 \; pulls \tag{47}$$

After modifying the simulation code with the same logic for the pity calculation, I again ran the code for a billion iteration and the following were the results.

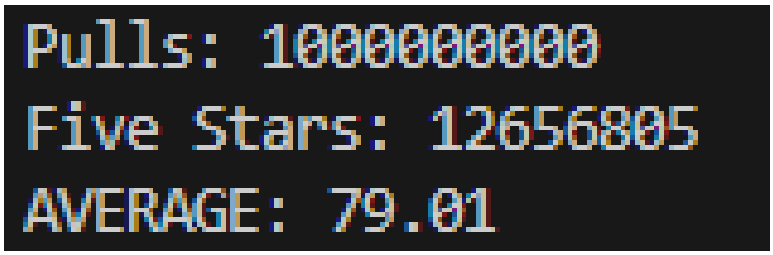

Figure 10: New Exclusive Bond Simulation Results

The value from the simulation is 0.08 pulls away from the calculated expected value and the consolidated probability is also close to the given value. Hence, I can, with some confidence, say that this is close to the system that is used in the game. Now, looking at the histogram for the new simulation:

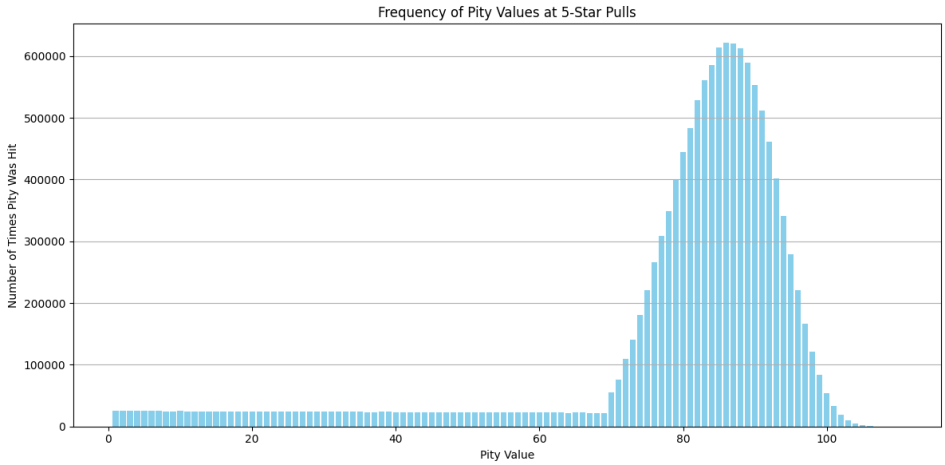

Figure 11: New Exclusive Bond Simulation Graph

The new mode of the graph is at 86 pulls. Now, we have collected the data for both the graphs and I can compare them to real world data and try to draw a conclusion to see which one is better.

6 Real World Data Comparison

Now the data we have about the two systems through our models are:

Name of System	**Mode**	**Average**	
Fated Bond	75	5★: 53.00	Limited 5★: 71.16
Exclusive Bond	86	78.92	

Table 1: Table of data of the two systems found by modelling.

Now, since October of 2024, I have been collecting data from players to try and find these data. The data can be found here. The data, at the time of writing, is:

AVG PITY FOR EXCLUSIVE	MEDIAN	MODE	Sample Size	
78.2739726	86.5	100	146	
AVG PITY FOR FATED	MEDIAN	MODE	Sample Size	50/50 Win Rate
53.36018957	68	72	211	0.5369127517
	WINS COUNT	LOSSES COUNT	GUARANTEE COUNT	LOSS + GUARANTEE PITY
	80	69	63	105.980859
				TOTAL PITY
				79.74042951

Figure 12: Data from Google Form Data Collection

Comparing with my data, the mode values are very off, but that is expected due to the very low sample size, as the game does not have a large following. However, on comparing the average values, we can see that they are very close:

Name of Value	**Calculated Value**	**Real World Value**	**Difference & Error**
Exclusive Bond Average	78.92	78.27	0.65 = 0.8%
Fated Bond Average for 5⋆	53.00	53.36	-0.36 = 0.6%
Fated Bond Average for Limited 5⋆	79.50	79.74	-0.24 = 0.3%

Table 2: Comparison Table

Hence, the models I have found for the two systems are accurate by around $\pm 0.8\%$.

7 Comparing the two systems.

Now, we can use the models to compare the two systems and try to see which one is better in what scenario.

7.1 Details for Both Systems

7.1.1 Fated Bond

- Mode: 75
- Average for a 5⋆: 53 pulls
- Average for a Limited 5⋆: 79.5 pulls
- Worst Case for a Limited 5⋆: 160 pulls

7.1.2 Exclusive Bond

- Mode: 86
- Average for a non-limited 5⋆: 165 pulls
- Average for a Limited 5⋆: 78.9 pulls
- Worst Case for a Limited 5⋆: 110 pulls

7.2 Scenario Analysis

7.2.1 Most amount of 5⋆ characters

If one wants the most amount of 5⋆ characters, regardless of if they are limited or non-limited, Fated Bond is the more useful system.

7.2.2 Most amount of Limited 5⋆ characters

If one wants the most amount of Limited 5⋆ characters it is a bit more complicated as if you get very lucky, the Fated Bond is the best option. However, it can also be worse if you lose the 50/50 chance. Looking at the averages I have calculated, Exclusive Bond has a better average but by a very tiny margin. However, looking at the worst case scenarios, Exclusive Bond has a lower worst case and hence, in my opinion is the better system in this scenario.

7.2.3 Equal Mix of Non-Limited and Limited 5⋆ characters

Modifying the Exclusive Bond code to include the non-limited 5⋆ character and running both codes for 10,000 iterations, we can see which one gives more.

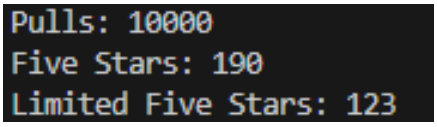

Figure 13: Fated Bond

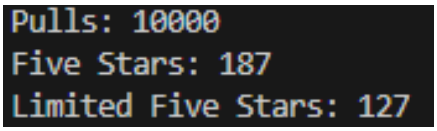

Figure 14: Exclusive Bond

As we can see, the values are negligibly different. Even at lower pull counts like 500, I get very similar or the same values. Hence, both are equal in performance here.

7.3 Time and Resource Constraint

If you are unable to get a lot of the currency required to pull in the game or if the character you are summoning for is going away very soon, then the earlier the 5⋆ character appears, the better. In this niche scenario, Fated Bond is better due to higher odds.

8 Limitations of Calculations

- I have no way to confirm that the models are 100%. With the comparisons showed above, I can say I am 99% accurate, but not 100%.
- The real world data collected is a very low sample size due to the small community of the game that I have access to.
- Additionally, we are dependent on others to submit accurate data. If they only enter lucky instances, like getting the 5⋆ character before soft pity, or if they input inaccurate data, the data can be skewed.
- There is no way to know what pseudo-random number generation the game uses, which could cause inaccuracies.

9 Conclusion

Concluding from this, I am personally going to use the Exclusive Bond system more often as it is better for getting limited 5⋆ characters. However, at the end of the day, the purpose of the game is to have fun and the game developers have made the systems pretty close to each other in performance. Hence, you can use either system depending on what type of player you are. If you are a more calculated person and does not like risk-taking, Exclusive Bond is better. However, if you enjoy the thrill of chance taking, Fated Bond might be for you. Additionally, having higher rates, a 10x pull on Fated Bond could have more than 1 5⋆ character, a phenomenon that is rare and very exciting to get. Your choice is dependent on the scenario you are faced with and your interpretation of the data I have collected here.