Implementation of a Linear Congruential Generator for Simulating the Character Event Warp System in Honkai: Star Rail

Muhammad Atallah Ramadhan - 13525015
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: atallahram@gmail.com , 13525015@std.stei.itb.ac.id

***Abstract*—Honkai Star Rail is a game that features a gacha system, in which players spend in-game currency to obtain random in-game items. One of its *gacha* systems, Character Event Warp, is a limited time *gacha* banner featuring a promotional character. This paper explores whether the implementation of Linear Congruential Generator can be used to simulate the *gacha* system of Character Event Warp.**

***Keywords*—Honkai Star Rail, Gacha system, Linear Congruential Generator, Character Event Warp**

I. Introduction

In recent years, video games have become one of the most popular forms of entertainment worldwide, even surpassing the popularity of film and music industries [1]. Through its various types of gameplay and systems, video games have attracted millions of players. To enhance player engagement, many video games use randomness and probability-based systems. One such system is the gacha mechanics, originated from Japanese capsule toy, gachapon, where the consumer buys a gacha pull from the seller, every gacha pull has its own probabilities, whether it will win the grand prize or not . This system was later used in video games and commonly described where players spend in-game currency to obtain random in-game items [2]. The gacha mechanics in video games have no significant difference from toy capsule systems, as both involve obtaining random rewards. However, gacha systems in video games generally provide more consistent probabilities and often have a guarantee mechanism known as a pity system, which limits the number of attempts required to obtain a certain item.

HoYoverse is one of the most widely recognized gacha games publishers, and one of its games, Honkai: Star Rail, is currently among the most popular games in the gaming community. Since its launch on April 26 2023, Honkai: Star Rail has become popular due to its unique turn-based gameplay, outstanding graphics, and various mechanics and systems including the gacha system, which allows players to obtain various Light Cones, items that function as weapons in the game, or characters through gacha pulls or known as warp in this game. Honkai: Star Rail has multiple types of gacha systems, and one of the systems is Character Event Warp, where players can perform warps to obtain limited time event characters. Since the outcomes of Warps are determined through randomization and probability-based mechanisms, this system can be simulated using a pseudorandom number generator (PRNG).

The implementation of the Linear Congruential Generator (LCG) is one of the ways that we could use to simulate the Character Event Warp system as LCG is a pseudorandom number generator that capable of producing random numbers by using modular arithmetic and recursive equations.

This study aims to evaluate whether the implementation of Linear Congruential Generator can be used to simulate the gacha system of Character Event Warp.

II. Fundamental Theory

A. Recurrence Relation

A recurrence relation is an equation that defines each term of a sequence as a function of one or more preceding terms. Recurrence relations are widely used in mathematics and computer science to describe iterative processes [3].

$$a_n = f(a_{n-1})$$

Where a_n represent the current term and a_{n-1} represent the previous term.

In a first-order recurrence relation, each term depends only on the immediately preceding term. The Linear Congruential Generator follows this pattern, as each generated value is computed from the previous value in the sequence [4].

B. Arithmetic Modular

Modular arithmetic is a system of arithmetic in which numbers wrap around after reaching a certain value called the modulus. The modulo operation returns the remainder of a division [5].

$$a \bmod m = r$$

where a is the dividend, m is the modulus, and r is the remainder. This operation can also be expressed as

$$a = mq + r$$

where q is an integer quotient and $0 \leq r < m$

C. Linear Congruential Generator

The Linear Congruential Generator (LCG) is one of the simplest and most widely used pseudorandom number generation algorithms. It generates a sequence of pseudorandom numbers using a recurrence relation combined with modular arithmetic [6]

$$x_{n+1} = (a\,x_n + c) \bmod m$$

where x_n is the current value in the sequence, x_{n+1} is the next generated value, a is the multiplier, c is the increment, m is the modulus, and x_0 is the initial seed.

The algorithm begins with an initial seed value x_0. For each iteration, a new value is generated using the recurrence relation above. The modulo operation ensures that all generated values remain within the range $0 \leq x_n < m$. As a result, the generated sequence appears random while remaining deterministic, meaning that the same set of parameters and seed will always produce the same sequence.

One of the most well-known implementations of the LCG is the ANSI C pseudorandom number generator, which uses the parameters $m = 2^{32}$, $a = 1103515245$, and $c = 12345$. Using an initial seed value of $x_0 = 12345$. the generator produces a deterministic sequence of pseudorandom numbers according to

$$x_{n+1} = (1103515245x_n + 12345) \bmod 2^{32}$$

These parameters have been widely adopted due to their simplicity and computational efficiency, making them a common example of LCG implementation [6].

D. Honkai: Star Rail Character Event Warp System

Based on [7], the Character Event Warp system uses a probability-based reward mechanism combined with a guarantee system, commonly referred to as a pity system. The possible outcomes of each Warp can be categorized as follows:

1. 3-star Light Cone

A 3-star Light Cone is the most common outcome in the Character Event Warp system, with a base drop rate of 94.3% [7]. Players will obtain a 3-star Light Cone whenever the conditions for obtaining a 4-star or 5-star item are not satisfied. Unlike higher-rarity items, 3-star Light Cones are not associated with any guarantee or pity mechanism.

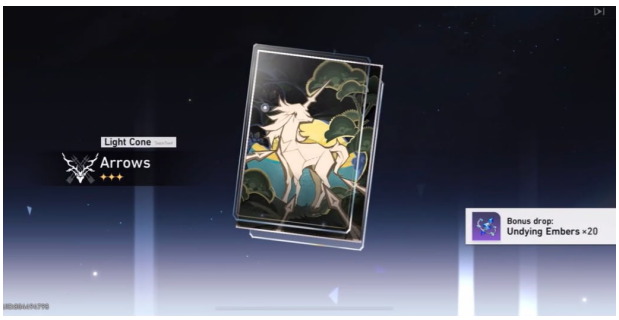


Fig. 2.1. 3-star items example. Source: Own.

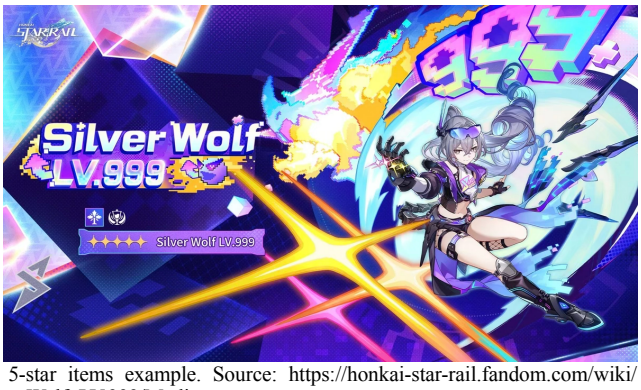
2. 4-star Character or Light Cone

A 4-star Character or Light Cone has a base drop rate of 5.1% [7]. To ensure that players periodically receive higher-rarity items, the Character Event Warp system implements a 4-star pity mechanism. If a player does not obtain any 4-star or higher-rarity item within nine consecutive Warps, the tenth Warp is guaranteed to yield at least a 4-star item. Once a 4-star item is obtained, the corresponding pity counter is reset. Furthermore, every 4-star item has a 50% probability of being one of the featured 4-star characters. If the obtained 4-star item is not a featured character, the next 4-star item is guaranteed to be one of the featured characters.

2.2. 4-star items example. Source: https://game8.co/games/Honkai-Star-Rail/archives/440625

3. 5-star Character

A 5-star Character has a base drop rate of 0.6% [7]. If a player does not obtain a 5-star character within 89 consecutive Warps, the 90th Warp is guaranteed to yield one. Each 5-star character has a 50% probability of being the featured promotional character; otherwise, the next 5-star character obtained is guaranteed to be featured. The pity counter and guarantee status are preserved across consecutive Character Event Warp banners.



2.3. 5-star items example. Source: https://honkai-star-rail.fandom.com/wiki/Silver_Wolf_LV.999/Media

III. IMPLEMENTATION

A. Python Implementation

Python was selected as the programming language for this implementation due to its clarity, flexibility, and effectiveness in developing mathematical and probabilistic models. The program aims to simulate the Character Event Warp system in Honkai: Star Rail by using a Linear Congruential Generator

(LCG) to generate pseudo-random values that determine the results of each warp.

The program defines several constants for the LCG algorithm, namely m=232, a=1103515245, and c=12345. These values are used to generate deterministic pseudo-random numbers according to the formula:

$$x_{n+1} = (1103515245x_n + 12345) \bmod 2^{32}$$

A seed value is initialized at the beginning of the program and acts as the starting point of the random number sequence. Every generated number depends on the previous value, ensuring reproducible simulation results when the same seed is used.

The LCG( ) function is responsible for generating pseudo-random numbers. It updates the current seed using the LCG formula and returns a value between 0 and 9999. This range simplifies probability calculations throughout the warp simulation.

```
# Linear Congruential Generator
# Menghasilkan bilangan 0 – 9999

def LCG():
    global seed

    seed = (A * seed + C) % M

    return seed % 10000
```

Fig. 3.1. LCG Function.

The GenerateRarity( ) function determines the rarity of a pulled character. Based on the generated random number, the function assigns a rarity of 5-star, 4-star, or 3-star according to predefined probability thresholds. Specifically, a value below 60 corresponds to a 5-star item, a value between 60 and 569 corresponds to a 4-star item, and all remaining values correspond to a 3-star item.

```
# Menentukan rarity normal

def GenerateRarity():

    r = LCG()

    if r < 60:
        return 5

    elif r < 570:
        return 4

    else:
        return 3
```

Fig. 3.2. GenerateRarity Function.

The Featured4( ) function determines whether a 4-star reward belongs to the featured pool. If the guarantee mechanism is active, the function automatically returns a featured 4-star reward. Otherwise, a random number is generated and a 50% probability is applied. If the result is not featured, the guarantee flag is activated for the next 4-star pull.

```
# Menentukan Featured 4 Star

def Featured4():

    global guarantee4

    if guarantee4:
        guarantee4 = False
        return True

    r = LCG()

    if r < 5000:
        return True

    guarantee4 = True
    return False
```

Fig. 3.3. Featured4 Function.

Similarly, the Featured5( ) function determines whether a 5-star reward is the featured event character or a standard character. The function implements the same guarantee mechanism as the actual game, ensuring that after losing a 50/50 chance, the next obtained 5-star character is guaranteed to be featured.

```
# Menentukan Featured 5 Star

def Featured5():

    global guarantee5

    if guarantee5:
        guarantee5 = False
        return True

    r = LCG()

    if r < 5000:
        return True

    guarantee5 = True
    return False
```

Fig. 3.4. Featured5 Function.

The Warp( ) function performs a single warp operation. This function first checks whether hard pity conditions have been reached. If the 5-star pity counter reaches 89, the next pull is guaranteed to be a 5-star reward. If the 4-star pity counter reaches 9, the next pull is guaranteed to be at least a 4-star reward, with a small possibility of obtaining a 5-star reward instead.

```
def Warp():

    global pity4
    global pity5

    # Hard Pity 5 Star
    if pity5 == 89:

        rarity = 5

    # Hard Pity 4 Star
    elif pity4 == 9:

        r = LCG()

        if r < 9940:
            rarity = 4
        else:
            rarity = 5

    # Pull biasa
    else:

        rarity = GenerateRarity()

    # Update Status

    if rarity == 3:

        pity4 += 1
        pity5 += 1

        print("3★")
```

Fig. 3.5. Warp Procedure part 1.

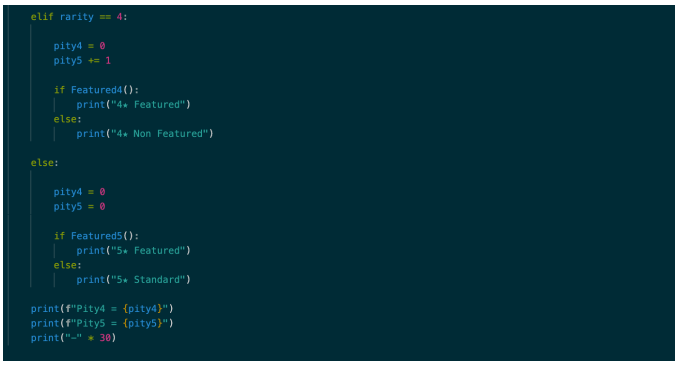


Fig. 3.5. Warp Procedure part 2.

After determining the rarity, the function updates both pity counters accordingly. Obtaining a 3-star reward increases both pity counters, while obtaining a 4-star reward resets the 4-star pity counter and increases the 5-star pity counter. Obtaining a 5-star reward resets both pity counters. The function then determines whether the obtained reward is featured or non-featured and displays the result.

Finally, the main program requests the number of warps from the user and repeatedly calls the Warp( ) function for each pull. This allows the simulation of multiple consecutive warps while accurately maintaining pity counters and guarantee states throughout the process.

One limitation of this implementation is that it only simulates the hard pity and guarantee mechanisms. Features such as soft pity, varying drop rates across different banners, character-specific pools, and weapon banners are not included. Therefore, the simulation serves as a simplified representation of the actual Honkai: Star Rail Character Event Warp system.

B. Determining the Probability Intervals

To map the generated pseudo-random numbers into reward rarities, the probability distribution is transformed into intervals over the range [0, 9999]. Since the Linear Congruential Generator (LCG) produces values that are reduced using the modulo operation ($X \bmod 10000$), there are exactly 10,000 possible outcomes. As a result, the probability of obtaining each rarity can be represented as a corresponding number of values within this range.

The Character Event Warp system uses three rarity categories: 5-star, 4-star, and 3-star rewards. Based on the official probability distribution used in this simulation, a 5-star reward has a probability of 0.6%, while a 4-star reward has a probability of 5.1%. The remaining probability is assigned to 3-star rewards.

To determine the size of each interval, the probability of each rarity is multiplied by 10,000. For 5-star rewards,

$$0.006 \times 10000 = 60$$

which means that 60 values are allocated to the 5-star interval. Therefore, any generated value between 0 and 59 inclusive will result in a 5-star reward.

Similarly, the probability of obtaining a 4-star reward is 5.1%, giving

$$0.051 \times 10000 = 510$$

possible values. Since the first 60 values have already been assigned to the 5-star interval, the next 510 values, ranging from 60 to 569, are assigned to the 4-star interval.

The remaining values are allocated to 3-star rewards. Because there are 10,000 possible outcomes in total, the number of values assigned to the 3-star interval is

$$10000 - 60 - 510 = 9430$$

Thus, all generated values from 570 to 9,999 correspond to a 3-star reward.

The resulting rarity intervals can be summarized as follows:

$$(0 \leq r < 60) \text{ —> 5-star}$$

$$(60 \leq r < 570) \text{ —> 4-star}$$

$$(570 \leq r < 10000) \text{ —> 3-star}$$

where $(r = X \bmod 10000)$.

This interval-based approach provides a simple and computationally efficient method for determining reward rarities. Instead of performing floating-point probability calculations for every warp, the program only needs to compare the generated value against predefined interval boundaries.

Furthermore, this method guarantees that the simulated rarity distribution closely matches the intended probabilities, provided that the pseudo-random numbers generated by the LCG are uniformly distributed across the interval ([0, 9999]).

IV. RESULT

To verify the implemented rarity determination mechanism, several sample values generated by the Linear Congruential Generator (LCG) were evaluated. The generated state value (X) was converted into a rarity indicator using the following equation:

$$r = X \bmod 10000$$

Where (X) is the result of LCG function.

The resulting value (r) was then compared against the predefined rarity intervals:

$(0 \leq r < 60)$ —> 5-star

$(60 \leq r < 570)$ —> 4-star

$(570 \leq r < 10000)$ —> 3-star

Table I presents several simulation examples and their corresponding warp results.

No.	Descriptioin		
	Generated Value (X)	***X mod 10000***	***Rarity***
1	123400000	0	5-star
2	999999005	5	5-star
3	100000123	123	4-star

No.	Descriptioin		
	Generated Value (X)	*X mod 10000*	*Rarity*
4	888888569	569	4-star
5	1234570	570	3-star
6	999999999	9999	3-star

Table I. Simulation Result.

From the simulation results, it can be observed that the rarity outcome depends solely on the value of ($X \mod 10000$), rather than the magnitude of the generated state itself. For example, the values 123400000 and 999999005 are significantly different in magnitude; however, both produce residues within the interval allocated for 5-star rewards and therefore generate identical rarity outcomes.

This behavior confirms that the implemented interval-based mapping correctly translates pseudo-random numbers into reward rarities according to the specified probability distribution. Furthermore, the implementation successfully reproduces the intended probabilities of 0.6% for 5-star rewards, 5.1% for 4-star rewards, and 94.3% for 3-star rewards.

The simulation demonstrates that the LCG-generated values can be effectively utilized to model the Character Event Warp system, while the modulo operation provides a simple and efficient method for assigning rarity outcomes.

V. CONCLUSION

This study successfully implemented a simulation of the Honkai: Star Rail Character Event Warp system using a Linear Congruential Generator (LCG) as the pseudo-random number generator. The LCG algorithm was utilized to generate random values that were subsequently mapped to reward rarities based on the predefined probability distribution of 3-star, 4-star, and 5-star items.

The simulation also incorporated key mechanics of the actual warp system, including pity counters, hard pity guarantees, and featured character guarantee rules. These mechanisms ensure that the generated results follow the intended probability structure while providing a more realistic representation of the in-game gacha system.

However, this simulation has several limitations. It does not implement advanced mechanics such as soft pity, banner-specific rate adjustments, or weapon banner systems. Future work may extend the model by incorporating these features and comparing the statistical properties of different pseudo-random number generators to evaluate their impact on simulation accuracy.

REFERENCES

[1] Raka Yuda Priyangga, Imam Salehudin. 2025. “Engagement and Consumption Behavior in Gacha Games”. [Online]. Available: https://journal.binus.ac.id/index.php/jggag/article/view/13115

[2] Haoyu Wen. 2025. “From Japan to China: The Evolution of the Gacha Model and Cross-Cultural Communication Research”. [Online]. Available: https://www.researchgate.net/publication/394866841_From_Japan_to_China_The_Evolution_of_the_Gacha_Model_and_Cross-Cultural_Communication_Research

[3] Ivan Gonoskov. 2014. “Closed-form solution of a general three-term recurrence relation”. [Online]. Available: https://link.springer.com/article/10.1186/1687-1847-2014-196

[4] Rinaldi Munir. 2026. “14-Barisan, rekursi-dan-relasi-rekurens-(Bagian2)-2026”. [Online]. Avalaible: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/14-Barisan,%20rekursi-dan-relasi-rekurens-(Bagian2)-2026.pdf

[5] Robert Milson. 2008. ”Introduction to Public Key Cryptography and Modular Arithmetic.”. [Online]. Available: https://www.researchgate.net/publication/1845076_Introduction_to_the_RSA_algorithm_and_modular_arithmetic

[6] Karl Entacher. 2001. “Introduction to Public Key Cryptography and Modular Arithmetic.”. [Onlinel]. Available: https://www.researchgate.net/publication/2683298_A_Collection_of_Selected_Pseudorandom_Number_Generators_With_Linear_Structures

[7] Honkai: Star Rail Wiki: Warp. Available: https://honkai-star-rail.fandom.com/wiki/Warp

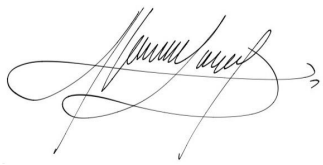
ACKNOWLEDGMENT

The author is grateful to God for the strength and opportunity to complete this paper. Appreciation is extended to Dr. Ir. Rinaldi, M.T. for providing valuable insights and instruction throughout the IF1220 Discrete Mathematics course. The author also thanks family and friends for their encouragement during the preparation of this work. Lastly, recognition is given to the developers of Honkai: Star Rail, whose game mechanics inspired the topic explored in this paper.

STATEMENT

I hereby declare that this paper is my own writing, not an adaptation, or translation of someone else's paper, and not plagiarized.

Sumedang, 19 Juni 2026



Muhammad Atallah 13525015