OST dalam Reroll Akun Honkai : Star Rail

Penggunaan Optimal Stopping Theorem (OST) dalam Reroll akun Honkai : Star Rail Untuk Memaksimalkan Item dan Mengurangi Waktu yang Digunakan

Ishaq Irfan Farizal - 13524094
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: ishaqirfanfrzl@gmail.com , 13524094@std.stei.itb.ac.id

***Abstract*—HSR merupakan salah satu game paling populer saat ini, di mana banyak pemain yang suka untuk mengumpulkan semua karakter dan senjata di game ini. Salah satu caranya adalah dengan reroll, dimana pemain menggunakan pull gratis saat membuat akun baru untuk mendapatkan karakter dan senjata langka. Makalah ini bertujuan untuk optimisasi cara reroll dengan menggunakan Optimal Stopping Theorem, agar pemain dapat memaksimalkan hasilnya dengan menggunakan paling sedikit waktu.**

Kata Kunci—reroll; Honkai : Star Rail; pull; Optimal Stopping Theorem

I. PENDAHULUAN

Gambar 1. Ilustrasi Honkai : Star Rail
Sumber: https://www.playstation.com/en-us/games/honkai-star-rail/, diakses pada 20 Juni 2025

Gacha atau *gachapon* awalnya berasal dari mesin mainan di Jepang yang memberikan item secara acak. Istilah ini kemudian digunakan dalam game untuk sistem undian berhadiah item *in-game*. Salah satu *game* yang menggunakan fitur gacha adalah Honkai: Star Rail (HSR) adalah *game turn-based RPG* bertema *space fantasy* buatan HoYoverse, dirilis 26 April 2023. Game ini memiliki sistem *gacha* dengan *banner* karakter/senjata, peluang 0.6% untuk mendapatkan item eksklusif, serta sistem pity yang menjamin item tertentu setelah jumlah *pull* tertentu.

Setiap akun baru HSR *pull* gratis. *Pull* ini digunakan pemain untuk memaksimalkan karakter/senjata pemain pada akun yang ingin dmainkan kedepannya. Stratgei ini disebut *reroll*, yang merupakan strategi dimana pemain membuat akun baru terus menerus sampai mendapatkan item yang diinginkan. Namun, *reroll* memakan waktu pemain. Karena itu, makalah ini bertujuan untuk melihat apakah *Optimal Stopping Theory* bisa diterapkan untuk meningkatkan peluang dan efisiensi dalam melakukan *reroll*.

Demi kesederhanaan dalam simulasi, ada beberapa batasan yang ditetapkan:

1. Sistem *pity* diabaikan
2. Semua item hanya dibedakan berdasarkan *rarity*-nya, dengan target yang dicari A atau *5 Star*
3. Akun yang dapat diterima pemain hanya satu.

II. LANDASAN TEORI

A. Teori Kombinatorika [1][4]

Kombinatorika adalah cabang matematika untuk menghitung jumlah penyusunan objek-objek tanpa harus mengenumerasi, menghitung satu per satu, semua kemungkinan susunannya.

Kaidah dasar menghitung dalam kombinatorika dibagi menjadi 2,

a. Kaidah Penjumlahan

Untuk ”m cara melakukan A, n cara melakukan B, dan kejadian A dan B tidak terjadi bersamaan,” maka total cara adalah

$$Total = m + n$$

b. Kaidah Perkalian

Untuk "m cara melakukan A, n cara melakukan B, dan kejadian A dan B saling bebas," maka total cara adalah

$$Total = m \times n$$

Dalam pembuatan susunan, ada permutasi dan kombinasi:

1) Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek. Jika terdapat n objek tanpa pengulangan dan harus disusun dengan memperhatikan urutannya, menurut kaidah perkalian, permutasinya adalah

$$n(n-1)(n-2)\dots(2)(1) = n!$$

Untuk kasus permutasi k objek dari n objek, didapat jumlah permutasi:

$$P(n,k) = \frac{n!}{(n-k)!}$$

2) Kombinasi

Kombinasi adalah jumlah susunan berbeda tanpa memperhatikan urutannya. Jumlah kombinasi untuk r dari n objek adalah

$$C(n,k) = \frac{n!}{k!\,(n-k)!}$$

Contoh penerapan permutasi dan kombinasi adalah beriku:

Digambarkan ada 10 orang mahasiswa, banyak cara menbentuk susunan perwakilan yang beranggotakan 5 orang tanpa melihat urutan adalah

$$C(10,5) = \frac{10!}{5!\,(10-5)!} = 252$$

Sedangkan jika, perwakilan itu terdiri dari ketua, wakil, sekertaris, bendahara, dan anggota. Maka banyak susunan harus memperhitungkan urutan jadi susunan yang dapat dibentuk adalah:

$$P(10,5) = \frac{10!}{(10-5)!} = 30240$$

Teori permutasi dan kombinasi ini banyak digunakan dalam perhitungan peluang, sebagai cara mencari semua susunan yang dapat dibentuk

B. Teori Peluang [4]

Teori peluang sangat berguna di berbagai bidang, salah satunya ada penerapannya dalam bidang komputer adalah dalam pengambilan keputusan dalam situasi tidak pasti, seperti dalam mengelompokkan data.

Peluang suatu kejadian adalah perbandingan antara jumlah hasil kejadian yang mencukupi(syarat) dengan jumlah seluruh hasil yang mungkin, dengan asumsi bahwa semua hasil memiliki kemungkinan muncul yang sama.

Seluruh hasil yang mungkin disebut ruang peluang. Untuk setiap hasil diberi angka peluang dan jumlah semua peluang dalam satu ruang peluang yang sama harus bernilai 1.

Peluang dibagi dua, yaitu peluang diskrit dan peluang kontinu. Pada penelitian ini hanya akan digunakan peluang diskrit. Peluang diskrit dapat dirumus kan menjadi:

$$P(E) = \frac{|E|}{|S|}$$

Dengan,

S = himpunan semua hasil yang mungkin

E = subhimpunan S atau kejadian yang memenuhi syarat

Berikut salah satu contoh aplikasi teori peluang:

Untuk kejadian 1 dadu dilempar dan muncul angka 6.

$$S = \{1,2,3,4,5,6\}$$

$$E = \{6\}$$

Maka,

$P(E) = \frac{1}{6}$

C. Optimal Stopping Theorem [2]

Optimal Stopping Theorem adalah teori yang digunakan untuk dapat memperoleh hasil terbaik dari suatu proses pengambilan keputusan. Misalnya, ingin dipilih kandidat terbaik dari total n orang yang datang satu per satu secara acak (semua urutan mungkin muncul dan memiliki peluang yang sama). Setiap kali dilihat satu kandidat, harus langsung diputuskan apakah ingin memilih kandidat tersebut atau tidak, dan pilihan ini tidak bisa ulang. Dengan menggunakan teorema ini, dapat dimaksimalkan peluang untuk memilih kandidat terbaik.

Misal kandidat terbaik ada di posisi ke-**i**, dengan i dari 1 sampai n.

Dengan syarat,

- Kandidat terbaik harus muncul setelah r orang pertama, aritnya $i > r$
- Dan kandidat terbaik ini harus lebih bagus daripada semua kandidat sebelum dia (di antara posisi 1 sampai i−1)

Dapat dihitung,

- Peluang terjadinya kandidat terbaik muncul di posisi ke-i, yaitu:

$$P1 = \frac{1}{n}$$

- Peluang kandidat terbaik diantara posisi 1 sampai i - 1, muncul di antara r orang pertama, yaiu

$$P2 = \frac{r}{i-1}$$

Maka, didapat total peluang

$$P = \sum_{i=r-1}^{n} \frac{1}{n}\frac{r}{i+1}$$

$$= \frac{r}{n}\sum_{i=r+1}^{n} \frac{1}{i-1}$$

$$= \frac{r}{n}\sum_{j=r}^{n-1} \frac{1}{j}$$

Dengan pendekatan integral didapat,

$$P \approx \frac{r}{n}\int_{r}^{n} \frac{1}{x}dx = \frac{r}{n}\ln\left(\frac{n}{r}\right)$$

Misal $x = \frac{r}{n}$, maka didapat

$$P \approx -xlnx$$

Dengan menurunkan persamaan, akan didapat nilai maksimumnya

$$P' = \frac{d(-xlnx)}{dx} = -lnx - 1 = 0$$

$$\ln = -1$$

$$x = \frac{1}{e}$$

Maka didapat bahwa: dengan melewati 1/e atau sekitar 36.8% dari total kandidat, lalu diambil yang pertama lebih bagus dari sebelumnya akan didapat peluang keberhasilan mendapat kandidat terbaik maksimal. Yaitu 1/e atau sekitar 36.8%.

Algoritma ini bisa dibagi menjadi 2 fase:

1. Eksplorasi, yaitu fase dimana algoritma hanya melihat dan melewati 37% kandidat awal.
2. Eksploitasi, yaitu fase dimana algoritma mulai memilih kandidat terbaik.

D. Peluang dan Reroll dalam Honkai : Star Rail [3][5]

Pull dalam konteks gacha adalah tindakan mengumpulkan item menggunakan uang virtual. Dalam HSR pemain bisa melakukan *pull* dengan hadiah terbesarnya item dengan *rarity 5-star* . *Rarity* sendiri merupakan tingkat kelangkaan *item*, dengan yang paling besar biasanya dinamakan 5-star. *Item* dengan *rarity 5-star* ini memiliki kekuatan yang mayoritas lebih besar dengan *item* lainnya dengan *rarity* lebih rendah, yaitu *4-star* dan *3-star.*

Menurut sumber komunitas Honkai : Star Rail, untuk setiap kali pull gacha dalam game, pemain memiliki kesempatan 94.300% mendapatkan item *3-star*, 5.100% mendapatkan item *4-star*, dan 0.6% mendapatkan item *5-star*. Jadi kesempatan pemain mendapatkan item *5-star* sama dengan 1/163 setiap kali pull.

Reroll merupakan tindakan mengulang ulang sebuah akun dari awal. Tindakan ini bertujuan untuk mendapatkan hadiah gratis yang didapat pemain saat membuat akun baru dan menggunakannya untuk mendapatkan akun yang kuat. Dalam HSR sekali *reroll* dibutuhkan waktu 30 - 40 menit untuk mendapat 20 *pull*, 50 - 60 menit untuk mendapat 30 *pull*, dan 120 - 150 menit untuk mendapat 40 - 50 *pull*.

III. METODE

Dalam pembuatan peluang algoritma yang dapat menyimulasi *gacha*, penulis menggunakan bahasa python dengan *library* numpy untuk membuat kode. Untuk algoritma akan dibuat dengan tujuan menghitung waktu yang dihemat dengan menggunakan OST, mencari rata-rata jumlah *5-star* atau A di setiap akun untuk strategi normal dan strategi OST dan mencari banyak rerolls yang digunakan untuk OST.

Pertama, terdapat parameter-parameter yang dapat diubah untuk mengatur simulasi. Pada parameter ini akan digunakan banyak *reroll* maksimal 30 kali. Selain itu banyak *pull* per akun baru akan diberi 30 untuk meminimalisir waktu yang digunakan per *reroll*. Algoritma akan mengulang sebanyak 10.000 kali iterasi agar didapat hasil yang adil.

Kedua, ada algoritma normal atau kontrol yang akan melakukan semua *reroll* 100 iterasi. Algoritma ini bertujuan membeli hasil maksimum yang dapat diambil jika pengguna melakukan semua *reroll*-nya.

Ketiga, ada algoritma OST yang akan mengambil akun terbaik setelah fase eksplorasinya. Program juga akan menghitung banyak rata-rata *reroll* yang dibutuhkan untuk sampai pada hasil tersebut.

Keempat, ada algoritma OST modified. Algoritma ini akan mengambil akun terbaik dari awal sampai akhir fase eksploitasinya. Jadi program akan melakukan banyak *reroll* yang sama dengan OST biasa tapi akun hasilnya adalah yang terbaik, yang pernah di lihat. Ini mencerminkan kenyataan dimana pemain dapat mengambil akun terbaik saja, tanpa harus mengikuti aturan OST.

Berikut adalah kode yang digunakan:

```python
import numpy as np

# Parameter
n = 30               # total akun
pulls = 30           # banyak pull
p = 0.006            # 0.6% kesempatan dapat  A
explore = int(n / np.exp(1))  # fase eksplorasi
trials = 10_000      # banyak pengulangan

def print_all(title, data):
    print(f"{title}:")
    total = 300000
    for s in range(0, 6):
        cnt = np.sum(data == s)
        prob = cnt / total * 100
        print(f"- {s} A: {cnt:,} acc ({prob:.1f}%)")

def print_sel(title, data):
    print(f"{title}:")
    total = len(data)
    for s in range(0, 6):
        cnt = np.sum(data == s)
        prob = cnt / total * 100
        print(f"- {s} A: {cnt:,} acc ({prob:.1f}%)")

def normal_run():
    best = 0
    all_acc = []
    for _ in range(n):
        a = np.sum(np.random.rand(pulls) < p)
        all_acc.append(a)
        if a > best:
            best = a
    return all_acc, best

def ost():
    best_exp = 0
    kept = 0
    used = n
    for r in range(1, n + 1):
        a = np.sum(np.random.rand(pulls) < p)
        if r <= explore:
            if a > best_exp:
                best_exp = a
        else:
            if a > best_exp:
                kept = a
                used = r
                break
    if kept == 0:
        kept = a
    return kept, used
```

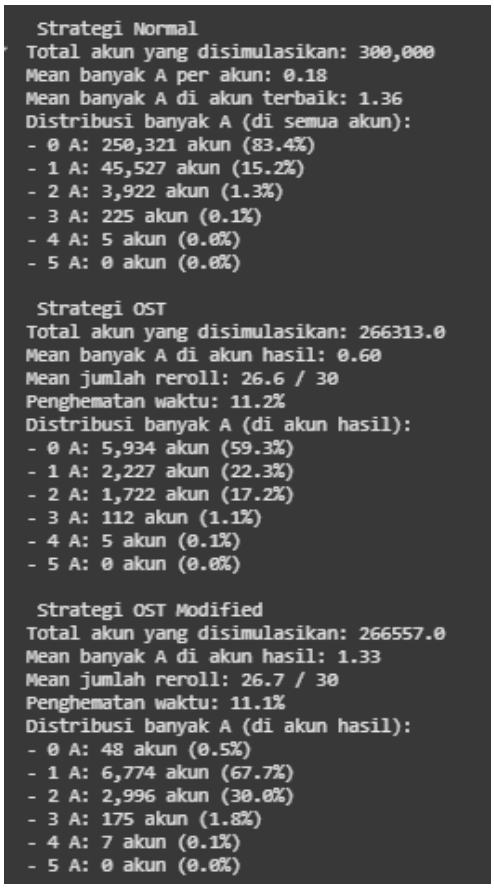
Gambar 2. Potongan kode (1)
Sumber: dokumentasi pribadi

```python
def ost_mod():
    best_all = 0
    kept = 0
    used = n
    best_exp = 0
    for r in range(1, n + 1):
        a = np.sum(np.random.rand(pulls) < p)
        if a > best_all:
            best_all = a
        if r <= explore:
            if a > best_exp:
                best_exp = a
        else:
            if a > best_exp:
                kept = a
                used = r
                break
    if kept == 0:
        kept = best_all
    return kept, used

# Run simulasi
norm_res = [normal_run() for _ in range(trials)]
ost_res = [ost() for _ in range(trials)]
ost_mod_res = [ost_mod() for _ in range(trials)]

# nalisis normal
all_acc = np.array([acc for acc, _ in norm_res])
best_acc = np.array([b for _, b in norm_res])

print("\nNormal Strategy")
print(f"Total akun: {trials*n:,}")
print(f"Mean A per akun: {np.mean(all_acc):.2f}")
print(f"Mean A terbaik: {np.mean(best_acc):.2f}")
print_all("Distribusi A (all)", all_acc)

# nalisis OST
ost_acc = np.array([a for a, _ in ost_res])
ost_used = np.array([u for _, u in ost_res])

print("\nOST Strategy")
print(f"Total akun: {np.mean(ost_used)/n *n*trials:.1f}")
print(f"Mean banyak A tersimpan: {np.mean(ost_acc):.2f}")
print(f"Mean rerolls: {np.mean(ost_used):.1f}/{n}")
print(f"Waktu dihemat: {100-np.mean(ost_used)/n*100:.1f}%")
print_sel("Distribusi A (tersimpan)", ost_acc)

# analisis OST modifed
mod_acc = np.array([a for a, _ in ost_mod_res])
mod_used = np.array([u for _, u in ost_mod_res])

print("\nModified OST")
print(f"Total akun: {np.mean(mod_used)/n *n*trials:.1f}")
print(f"Mean banyak A tersimpan: {np.mean(mod_acc):.2f}")
print(f"Mean rerolls: {np.mean(mod_used):.1f}/{n}")
print(f"Waktu dihemat: {100-np.mean(mod_used)/n*100:.1f}%")
print_sel("Distribusi A (tersimpan)", mod_acc)
```

Gambar 3. Potongan kode (2)
Sumber: dokumentasi pribadi

IV. PEMBAHASAN & HASIL

A. Hasil



Gambar 4. Hasil Program
Sumber: dokumentasi pribadi

B. Analisis Hasil

Dari hasil dapat ditemui beberapa hal. Pertama untuk banyak akun yang yang dibuat totalnya untuk strategi normal adalah banyak 300000 sesuai dengan teori. Untuk metode OST keduanya kurang lebih berada pada 266313 akun, dari sini didapat waktu yang dihemat kira-kira 11%. Perbedaan ini membuat metode OST, untuk lama waktu *reroll* per akun 40 menit, menghemat 120 - 160 menit, dengan asumsi banyak *reroll* 30 kali.

Selain itu rata-rata banyak A akun hasil, akun terbaik itu strategi normal, adalah 1.36 untuk strategi normal, 0.6 untuk strategi OST, dan 1.333 untuk strategi OST modified. Dari sini dapat dilihat bahwa tidak banyak perbedaan antara strategi normal dan OST modified, namun strategi OST biasanya memiliki rata-rata yang lebih rendah. Hal ini dapat disebabkan karena banyak akun terbaik yang terbuang saat algoritma di fase eksplorasi.

Untuk distribusi banyak A setiap pada strategi OST dan OST modified. Pada stategi OST modified, untuk distribusi banyak A yang lebih banyak, lebih besar jumlahnya. Hal ini menunjukkan bahwa OST modified memberi hasil yang lebih baik dikasus ini.

Namun perlu diperhatikan bahwa semua hasil ini berbasis *random*, sehingga tidak mutlak. Hasil akan berubah setiap kali simulasi. Oleh karena itu hasil ini tidak bisa dipandang mutlak.

V. KESIMPULAN

Dalam strategi reroll akun Honkai : Star Rail pemain masih masih dapat dimaksimalkan terutama dalam efisiensi waktu yang digunakan. Selain itu perlu juga diperhatikan item yang didapat selain melihat rarity-nya saja.

Aplikasi Optimal Stopping Theorem modified dalam akun *game* Honkai : Star Rail memberikan penghematan waktu yang cukup besar, sekitar 11%. Namun penghematan waktu ini menyebabkan hasil terbaik yang terpilih memiliki item *5 star* lebih sedikit dari strategi normal, dengan perbedaan 0.03. Jadi jika pemain memerlukan efisiensi waktu dengan tetap mendapatkan *reward* dalam batasan sumber daya yang ada, dapat digunakan OST modifed dalam *reroll*-nya. Namun jika ingin memaksimalkan *reward*, pemain dapat menggunakan strategi normal.

VI. SARAN

Saran untuk peneliti selanjutnya adalah melakukan optimasi lebih lanjut terhadap algoritma yang digunakan, khususnya dengan menambahkan batasan yang relevan. Misalnya, selain mempertimbangkan kapan waktu terbaik untuk berhenti pada algoritma OST modified peneliti dapat mengeksplorasi cara untuk mempercepat proses pemilihan dari hasil *pull* awal, agar pemain tidak perlu memainkan *game* sebanyak 40 menit untuk mendapatkan semua *pull*.

Saran untuk peneliti selanjutnya adalah optimasi algoritma, dengan menambahkan batasan lain untuk memaksimalkan waktu yang hemat. Selanjutnya peneliti selanjutnya item tidak hanya diukur berdasarkan *rarity* namun dengan memberi nilai untuk semua kemungkinan item sesuai dengan nilai fungsionalitasnya, hal ini karena *item* dengan *rarity* lebih tinggi belum tentu lebih bagus. Jadi dengan nilai ini, dapat diambil item yang bagus sesuai dengan nilainya.

LINK VIDEO YOUTUBE

https://youtu.be/rvS_2Oo0KYE

TERIMA KASIH

Pertama-tama, penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa yang telah memberikan penulis kesempatan untuk mengerjakan dan menyelesaikan makalah ini, yang berjudul “Optimal Stopping Theorem dalam Reroll Akun Honkai : Star Rail”. Selain itu penulis mengucapkan terima kasih atas Bapak Arrival Dwi Sentosa, S.Kom., M.T. sebagai dosen pengampu yang telah memberikan bimbingan dan ajaran kepada penulis. Tak lupa penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT. Sebagai

salah satu dosen IF 2024 yang telah memberikan banyak materi dan bahan referensi. Terakhir, penulis mengucapkan terima kasih kepada keluarga, teman, dan kepada seluruh pihak yang sudah membantu penulis dalam pembuatan makalah ini.

REFERENSI

[1] Rinaldi Munir, "Aljabar Boolean (Bag. 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf, diakses 18 Juni 2025 pukul 21.00.

[2] Ferguson, T. S. (2008). *Optimal Stopping and Applications* (Chapter 1: Stopping Rule Problems). UCLA Department of Mathematics.

[3] Prydwen Institute, "Honkai: Star Rail Reroll guide," https://www.prydwen.gg/star-rail/guides/reroll/. diakses 18 Juni 2025 pukul 23.00.

[4] A. V. Aho and J. D. Ullman, *Foundations of Computer Science*, C ed., Computer Science Press, 1992

[5] Di Jess Reyes, "Gacha games explained: Banners, pulls, pity systems, and more," https://store.epicgames.com/it/news/gacha-games-explained-banners-pulls-pity-systems-and-more, diakses 20 Juni 2025 pukul 16.00.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Juni 2025

Ishaq Irfan Farizal - 13524094