Penerapan Dynamic Programming untuk Optimasi Rotasi Skill Serangan dan Pertahanan pada Pertarungan Turn-Based RPG dengan Batasan Cooldown dan Resource Khusus

Mahatma Brahmana - 13524015
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mahatma.brahmana1@gmail.com , 13524015@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas penerapan Dynamic Programming untuk optimasi rotasi skill pada pertarungan turn-based RPG sederhana dengan batasan cooldown dan resource. Model pertarungan melibatkan satu party dan satu boss, dengan skill yang memiliki damage, shield, healing, cooldown, cost, dan gain resource. Hasil Dynamic Programming dibandingkan dengan Greedy Algorithm pada tiga skenario eksperimen yang dibuat. Berdasarkan hasil, Dynamic Programming menghasilkan rotasi skill yang lebih baik karena mampu untuk mempertimbangkan kondisi pada turn berikutnya, sedangkan Greedy hanya memilih skill terbaik pada turn saat ini**

Keywords—Dynamic Programming, Greedy Algorithm, turn-based RPG, rotasi skill, cooldown, resource

I. PENDAHULUAN

Turn-based RPG merupakan salah satu genre permainan yang populer di dunia saat ini. Permainan ini menggunakan sistem pertarungan berbasis giliran. Berbeda dengan game *real-time* seperti *MOBA* atau *action game* yang menuntut pemain untuk bergerak dan bereaksi secara langsung, turn-based RPG ini memberi pemain waktu untuk memilih aksi pada setiap *turn*. Walaupun secara tampilan *gameplay* biasanya terlihat sederhana karena pemain hanya memilih aksi atau skill secara bergantian, game dengan genre ini tetap menarik karena menekankan strategi, perencanaan, dan pengambilan keputusam. Pemain tidak hanya perlu memilih skil yang terlihat kuat, tetapi juga harus memperhatikan kondisi pertarungan, seperti HP, *resource, cooldown*, dan kemungkinan serangan musuh pada turn berikutnya.

Memilih pemilihan *skill* menjadi cukup menantang dan menarik karena keputusan pada satu turn dapat memengaruhi kondisi pada turn berikutnya, mirip seperti game catur. Misalnya menggunakan skill damage terlalu besar diawal dapat menghabiskan *resource*, atau memaksa menggunakan skill damage terus menerus membuat karakter team mudah mati, dan lain lainnya. Dengan demikian, rotasi skill bukanlah hanya persoalan tentang memilih skill terkuat, tetapi menyusun urutan aksi yang seimbang antara serangan, pertahanan, dan lain-lainnya.

Setiap game turn-based RPG itu berbeda beda. Ada game yang memiliki banyak karakter dalam satu party, sistem *speed, elemental advantage, buff, debuff, critical hit,* hingga pola serangan yang kompleks. Karena ruang lingkup tersebut sangatlah luas, makalah ini menggunakan model pertarungan yang lebih sederhana agar fokus pembahasan tetap berada pada optimasi rotasi skill.

Akan digunakan dua pendekatan untuk mencari optimasi rotasi skill ini. Pendekatan pertama adalah *Dynamic Programming* dan pendekatan kedua adalah *Greedy Algorithm* sebagai pembanding pendekatan pertama.

Berdasarkan permasalahan tersebut, makalah ini membahas penerapan Dynamic Programming untuk mencari rotasi skill yang lebih optimal pada pertarungan turn-based RPG sederhana. Model yang digunakan itu akan berfokus pada pemilihan skill setiap turn dalam batas turn tertentu dengan mempertimbangkan *HP party, HP boss, cooldown, resource, damage, shield*, dan *healing*. Hasil dari Dynamic Programming ini kemudian dibandingkan dengan Greedy Algorithm untuk melihat perbedaan strategi dan performanya.

II. DASAR TEORI

A. Turn-Based RPG dan Rotasi Skill

Turn-based RPG adalah game RPG yang menggunakan sistem giliran dalam pertarungannya. Dalam sistem ini, karakter tidak bisa menyerang secara bersamaan, melainkan melakukan aksi secara bergantian. Pada setiap aksi/turn, pemain dapat memilih satu aksi, seperti menyerang, bertahan, menggunakan skill, atau melakukan healing.

Dalam pertarungannya, pemilihan urutan skill atau rotasi skill menjadi hal yang penting. Rotasi skill adalah urutan penggunaan skill dari awal pertarungan sampai akhir pertarungan. Rotasi yang baik bukan hanya memperhatikan skill dengan damage terbesar setiap saat. Tetapi, pemain harus memperhatikan beberapa faktor lainnya, seperti sisa HP, cooldown, resource, heal, shield, dan lain-lainnya. Oleh karena itu, rotasi skill ini bisa dianggap sebagai masalah

optimisasi, karena pemain perlu untuk memilih urutan aksi untuk mengasilkan hasil yang terbaik.

B. Cooldown dan Resource Khusus

Cooldown adalah batasan yang membuat suatu skill tidak bisa langsung digunakan kembali setelah skill tersebut dipakai. Misalnya, sebuah skill memiliki cooldown 3 turn, maka setelah skill tersebut digunakan, skill tersebut harus menunggu 3 turn agar dapat digunakan kembali. Cooldown ini membuat pemilihan skill menjadi penting, karena pemain harus memikirkan kapan sebuah skill sebaiknya digunakan.

Resource khusus dapat dianggap sebagai energi, mana, atau poin aksi yang dibutuhkan untuk mengaktifkan skill tertentu. Skill yang kuat biasanya membutuhkan resource yang besar, sehingga pemain perlu untuk mengumpulkan resource terlebih dahulu sebelum dapat menggunakannya. Dengan adanya cooldown dan resource, keputusan pada satu turn dapat mempengaruhi pilihan skill pada turn-turn berikutnya.

C. Dynamic Programming

Dynamic Programming adalah metode penyelesaian masalah optimasi dengan cara membagikan persoalan itu menjadi beberapa tahap. Pada setiap tahap, terdapat keputusan yang harus diambil, dan keputusan itu dapat mempengaruhi kondisi pada tahap-tahap berikutnya.

Dalam Dynamic Programming, persoalan itu biasanya direpresentasikan sebagai *stage* dan *state*. Stage menunjukkan tahapan penyelesaian masalah, dan state menunjukkan kondisi yang mungkin terjadi pada suatu tahap. Dari setiap state, algoritma mencoba beberapa kemungkinan keputusan, lalu memilih keputusan dengan nilai terbaik berdasarkan fungsi objektif.

Pada penelitian ini, Dynamic Programming digunakan untuk mencari rotasi skill terbaik dalam pertarungan turn-based RPG. Setiap turn akan dianggap sebagai tahap dan kondisi pertarungan seperti HP party, HP boss, resource, dan cooldown skill dipandang sebagai state. Dengan ini, DP tidak hanya akan melihat keuntungan skil pada turn sekarang, tetapi juga akan mempertimbangkan dampaknya pada turn-turn berikutnya.

Dari sisi kompleksitasnya, Dynamic Programming membutuhkan waktu dan memori yang lebih besar dibandingkan dengan Greedy karena harus menyimpan dan mengevaluasi banyak kemungkinan state. Namun, dengan *memoization*, setiap state itu cukup dihitung satu kali. Oleh karena itu, DP layak digunakan untuk menyelesaikan masalah ini karena rotasi skill memiliki struktur keputusan bertahap, state yang jelas, dan ketergantungan antarturn.

D. Greedy Algorithm

Greedy Algorithm adalah metode penyelesaian masalah dengan mengambil keputusan terbaik pada setiap langkah. Anggapannya, dia mengambil keputusan terbaik di kondisi saat itu, tanpa mempertimbangkan kondisi selanjutnya. Algoritma ini berharap bahwa keputusan lokal tersebut bisa menghasilkan solusi optimal untuk global.

Pada penelitian ini, Greedy digunakan sebagai algoritma pembanding terhadap Dynamic Programming. Pada setiap turn, Greedy akan memilih skill valid dengan skor terbesar. Setelah skill dipilih, kondisi pertarungan diperbarui, dan proses yang sama dilakukan di turn berikutnya.

Pendekatan Greedy ini lebih sederhana karena tidak mengevaluasi kemungkinan state pada turn selanjutnya. Jika jumlah turn adalah T dan jumlah skill adalah S, maka kompleksitas waktunya dapat ditulis sebagai O(T x S). Namun, Greedy tidak memastikan solusi optimal karena keputusan terbaik dalam satu turn belum tentu menghasilkan hal yang baik di turn berikutnya.

III. IMPLEMENTASI ALGORITMA

A. Model Pertarungan

Pada implementasi ini, pertarungan dimodelkan sebagai pertarungan sederhana antara satu party dan satu bos. Satu parti dianggap sebagai satu entitas dengan satu nilai HP dan bos dianggap sebagai satu entitas dengan HP dan satu serangan dengan damage tetap. Pertarungan ini berlangsung dalam jumlah turn yang terbatas.

Pada setiap turn nya, party memilih satu skill untuk digunakan. Terdapat berbagai skill, ada yang bisa memberikan damage ke boss, ada yang bisa memberikan shield ke party, ada yang memberikan heal ke party, dan ada yang menambah dan memakai resource. Jika boss belum kalah setelah efek skill dari party dijakankan, maka bos akan menyerang balik party dengan damage tetap.

Pseudocode alur pertarungan secara umum adalah sebagai berikut.

```
for setiap turn sampai MAX_TURN:
    party memilih satu skill
    terapkan efek damage ke boss
    terapkan efek heal ke party
    perbarui resource party

    if boss kalah:
        pertarungan selesai

    hitung damage boss setelah dikurangi shield
    kurangi HP party

    if party kalah:
        pertarungan selesai

    perbarui cooldown skill
```

B. Skill

Setiap skill memiliki beberapa atribut, yaitu damage, shield, heal, cooldown, *cost*, dan *gain*. Damage digunakan untuk menyerang musuh sehingga mengurangi HP boss. Shield digunakan untuk memberikan HP sementara ke party yang berlangsung pada satu turn. Heal digunakan untuk memulihkan HP party. Cooldown menentukan jumlah turn yang harus dilewati agar skill bisa digunakan kembali. Cost menunjukkan

resource yang dibutuhkan untuk menggunakan skill. Dan gain menunjukkan resource yang diperoleh setelah skill digunakan.

Skill yang digunakan pada eksperiman ditunjukkan pada Tabel 1 berikut.

Skill	*Role*	*Dmg*	*Shield*	*Heal*	*CD*	*Cost*	*Gain*
Basic Attack	DPS	2000	0	0	0	0	2
Power Strike	DPS	5600	0	0	1	1	0
Ulrimate Burst	Burst DPS	13000	0	0	4	6	0
Guard Stance	Tank	0	2300	0	2	0	1
Healing Light	Healer	0	0	3200	3	3	0
Battle Chant	Support	0	800	0	2	0	4

[a.] Tabel 1. Atribut Skill

C. *Fungsi Objektif*

Fungsi objektif digunakan untuk menghitung kualitas dari suatu rotasi skill. Skor bukan hanya dihitung dari damage, tapi mempertimbangkan shield, healing, kill bonus, dan survive bonus. Untuk setiap skill yang digunakan, skor dihitung berdasarkan *effective damage* yang merupakan damage yang benar benar mengurangi HP boss, *effective shield* yang merupakan shield yang benar-benar mengurangi serang boss, dan *effective heal* yang merupakan heal yang benar benar memulihkan HP tanpa melebihi HP maksimum.

Perhitungan skor skill pada satu turn adalah:

score_skill = 0.5 × effective_damage + 0.6 × effective_shield + dynamic_heal_weight × effective_heal

Bobot heal dibuat dinamis agar healing bisa memiliki nilai lebih tinggi jika HP party lebih rendah. Bobot heal dinamis dihitung sebagai berikut:

dynamic_heal_weight = 0.8 + (1 - party_hp / party_max_hp)

Selain perhitungan skor dari skill, terdapat juga skor dari *kill bonus* dan *survive bonus*. Kill bonus diberikan jika boss berhasil dikalahkan dan survive bonus diberikan jika party masih bertahan hidup pada akhir simulasi atau setelah boss dikalahkan. Pada skenario Boss Kill Priority (dijelaskan lebih lanjut di bab IV), kill bonus dikalikan berdasarkan turn ketika boss dikalahkan:

kill_bonus_turn = kill_bonus × (max_turn - kill_turn + 1)

Tujuan dari rumus ini adalah untuk membunuh boss secepat mungkin, semakin cepat boss dikalahkan, semakin tinggi besar skor yang diperoleh.

D. *Dynamic Programming*

Dynamic Programming digunakan untuk mencari rotasi skill dengan skor maksimal. Pada masalah ini, keputusan pada suatu turn itu dapat mempengaruhi kondisi pada turn berikutnya, seperti cooldown skill, sisa resource, HP party, dan HP boss.

State yang digunakan pada Dynamic Programming adalah sebagai berikut:

DP(turn, resource, party_hp, boss_hp, cooldowns)

1) *turn :* turn keberapa yang sedang diproses
2) *resource :* jumlah resource yang dimiliki
3) *party_hp : sisa HP party*
4) *boss_hp : sisa HP boss*
5) *cooldowns : sisa cooldown dari setiap skill*

State ini dipilih karena seluruh komponen tersebut berpengaruh terhadap keputusan yang diambil, yaitu pemilihan skill.

Pada setiap state, algoritma mencoba semua skill yang valid. Skill dikatakan valid jika cooldown skill tersebut bernilai 0 dan resource party sama atau lebih banyak dari cost skill tersebut. Setelah skill digunakan, kondisi pertarungan akan diperbaharui, seperti HP boss dikurangi berdasarkan damage skill, HP party ditambah berdasarkan heal skill, dan resource diperbarui berdasarkan cost dan gain skill. Jika boss belum kehabisa HP setelah skill digunakan, boss akan menyerang balik party. Damage yang diterima party dari serangan boss adalah damage serangan boss dikurangi shield yang diberikan oleh skill shield pada turn tersebut.

Cooldown diperbarui setelah skill digunakan. Skill yang baru digunakan akan kembali ke nilai cooldown dasarnya, dan cooldown skill lain akan dikurangi 1 dengan nilai minimum 0. Dengan aturan ini, algoritma bisa menentukan skill apa yang terbaik digunakan memenuhi tujuan.

Relasi transisi Dynamic Programming dapat ditulis sebagai berikut:

DP(state) = max(score_skill + DP(next_state))

Rumus tersebut menunjukkan bahwa DP memilih skill dengan total skor terbesar. Total skor dihitung dari skor skill pada turn saat ini ditambah skor terbaik dari state berikutnya. Jadi, DP tidak hanya memilih skill yang terlihat bagus saat ini, tetapi juga mempertimbangkan apakah pilihan tersebut akan menghasilkan kondisi yang baik atau tidak untuk turn selanjutnya.

Terdapat beberapa base case pada algoritma DP ini. Jika HP boss sudah bernilai 0 atau kurang, maka boss dianggap kalah dan skor ditambah dengan kill bonus dan survive bonus. Jika HP party bernilai 0 atau kurang, maka party dianggap kalah dan state diberikan *loss score*. Jika turn sudah melewati batas maksimum, sedangkan party dan boss masih memiliki HP, maka skor ditambah dengan survive bonus.

Pseudocode Dynamic Programming yang digunakan adalah sebagai berikut:

```
DP(turn, resource, party_hp, boss_hp, cooldowns):
```

```
    if boss_hp <= 0:
        return kill_bonus + survive_bonus

    if party_hp <= 0:
        return loss_score

    if turn > max_turn:
        return survive_bonus

    best_score = loss_score

    for setiap skill:
        if skill masih cooldown:
            continue

        if resource tidak cukup:
            continue

        hitung score skill saat ini
        hitung next_boss_hp
        hitung next_party_hp
        hitung next_resource

        if boss kalah setelah skill digunakan:
            total_score = score_skill + kill_bonus +
survive_bonus
        else:
            party menerima serangan boss setelah dikurangi
shield

            if party kalah:
                total_score = loss_score
            else:
                hitung next_cooldowns
                total_score = score_skill + DP(next_state)

        best_score = max(best_score, total_score)

    return best_score
```

E. Greedy

Algoritma Greedy digunakan untuk pembanding dengan Dynamic Programming. Untuk setiap turn, Greedy akan memilih skill valid dengan skor langsung terbesar.

Greedy tidak mempertimbangkan kondisi beberapa turn berikutnya, sehingga skill yang dipilih Greedy terlihat menguntungkan pada turn saat ini, tetapi belum tentu menguntungkan untuk turn berikutnya. Oleh karena itu, Greedy cocok digunakan sebagai *baseline* untuk melihat perbedaan hasil antara keputusan lokal dan keputusan dengan memperhatikan jangka panjang.

F. Implementasi Program

Program diimplementasikan menggunakan bahasa Python dan dijalankan melalui *terminal*. Di dalam program, data skill disimpan menggunakan dataclass dan algoritma Dynamic Programming dan Greedy dibuat dalam fungsi terpisah.

Program menerima parameter yang dibutuhkan untuk pengecekan skenario melalui konstanta di awal *file*, seperti jumlah maksimum turn, HP party, HP boss, damage boss, bonus, dan bobot skor. Setelah dijalankan, program menampilkan hasil pertarungan untuk Dynamic Programming dan Greedy, meliputi skor, status akhir, rotasi skill, total damage, total shield, total heal, sisa HP party, sisa HP boss, dan resource akhir. Output inilah yang digunakan sebagai data untuk bab IV.

IV. HASIL EXPERIMEN DAN ANALISIS

A. Skenario dan Parameter Eksperimen

Eksperimen ini dilakukan untuk membandingkan hasil akhir rotasi skill yang diperoleh dari *Dynamic Programming* dan *Greedy Algorithm.* Kedua algoritma ini akan diuji pada model pertarungan *turn-based* RPG sederhana, yaitu satu party melawan satu boss dalam jumlah turn yang terbatas. Pada setiap turnnya, party akan memilih satu skill dan melakukan aksi sesuai skill tersebut, dan kemudian boss akan menyerang balik.

Pada eksperimen ini, sudah disiapkan tiga skenario. Skenario pertama adalah Standard Combat, yaitu pertarungan normal dengan tujuan untuk mencari rotasi terbaik dari menyerang, bertahan, healing, dan pengelolaan resource. Skenario kedua adalah Boss Kill Priority, yaitu pertarungan dengan tujuan untuk mengalahkan boss secepat mungkin. Semakin cepat boss dikalahkan, semakin tinggi bonus yang diperoleh. Skenario ketiga adalah High Pressure Survival, yaitu skenario dengan damage boss yang lebih tinggi dengan tujuan untuk mempertahankan party tetap hidup sampai batas turn.

Parameter utama pada setiap skenario ini dapat dilihat pada Tabel 2.

Skenario	***Max Turn***	***Boss HP***	***Boss Damage***	***Turn Kill Bonus***
Standard Combat	10	35000	2500	Tidak
Boss Kill Priority	8	25000	1800	Iya
High Pressure Survival	10	35000	2700	Tidak

[b.] Tabel 2. Parameter utama pada setiap skenario

Pada seluruh skenario, party memiliki HP maksimum 12000, resource awal 0, dan resource maksimum 10. Skill yang digunakan adalah skill damage, shield, healing, dan support. Setiap komponen skill ini diberikan bobot berbeda dalam perhitungan skor. Damage diberikan bobot 0.5, shield diberikan bobot 0.6, dan heal diberikan bobot 0.8. Khusus untuk heal, bobotnya dibuat dinamis berdasarkan kondisi HP party. Semakin rendah HP partynya, semakin tinggi nilai heal pada perhitungan skor. Selain itu, setiap skenario menggunakan kill bonus dan survive bonus sebesar 100000. Dengan pembobotan ini, diharapkan algoritma bisa mempertimbangkan penyerangan, pertahanan, healing, dan kondisi akhir pertarungan dengan baik.

B. Hasil Eksperimen

Hasil experimen akan ditampilkan secara terpisah untuk setiap skenario. Setiap tabel membandingkan hasil Dynamic Programming dengan Greedy berdasarkan score, status, total damage, total shield, total heal, HP party akhir, dan HP boss akhir.

Pada skenario Standard Combat, hasil experimen ditunjukkan pada Tabel 3.

Srandard Combat	*Dynamic Programming*	*Greedy*
Score	125660.0	-1000000000000000
Status	Max turn reached, turn 10	Party defeated, turn 7
Damage	24200	20800
Shield	7000	4600
Heal	6400	0
Party HP	400	0
Boss HP	10800	14200

[c.] Tabel 3. Hasil pengujian skenario Standard Combat

Pada skenario Boss Kill Priority, hasil experimen ditunjukkan pada Tabel 4.

Boss Kill Priority	*Dynamic Programming*	*Greedy*
Score	413460.0	117520.0
Status	Boss defeated, turn 6	Max turn reached, turn 8
Damage	25000	20800
Shield	1600	3600
Heal	0	3200
Party HP	4600	4400
Boss HP	0	4200

[d.] Tabel 4. Hasil pengujian skenario Boss Kill Priority

Pada skenario High Pressure Survival, hasil experimen ditunjukkan pada Tabel 5.

High Pressure Survival	*Dynamic Programming*	*Greedy*
Score	125093.33	-1000000000000000
Status	Max turn reached, turn 10	Party defeated, turn 7
Damage	18600	20800
Shield	9300	4600
Heal	6400	0
Party HP	700	0
Boss HP	16400	14200

[e.] Tabel 5. Hasil pengujian skenario High Pressure Survival

Nilai -1000000000000000 digunakan sebagai loss score. Nilai ini diberikan ketika party kalah, sehingga jalur yang menyebabkan kekalahan tidak dipilih sebagai solusi optimal.

C. Analisis Hasil

Berdasarkan hasil experimen ini, model Dynamic Programming mendapatkan hasil yang lebih baik dibandingkan model Greedy pada ketiga skenario. Pada skenario Standard Combat, Dynamic Programming berhasil mempertahankan party hingga akhir turn ke-10, sedangkan Greedy kalah pada turn ke-7. Pada skenario Boss Kill Priority, Dynamic Programming berhasil mengalahkan boss pada turn ke-6, sedangkan Greedy hanya mampu mempertahankan party hingga akhir turn ke-8. Pada skenario High Pressure Survival, Dynamic Programming berhasil mempertahankan party hingga turn ke-10, sedangkan Greedy kalah pada turn ke-7.

Perbedaan ini bisa terjadi karena Dynamic Programming mempertimbangkan langkah selanjutnya juga. Di implementasi ini, skill bukan hanya dinilai dari efek langsungnya, tetapi juga dari kondisi lanjutan yang dihasilkan, seperti sisa HP party, HP boss, resource, dan cooldown skill. Dengan alur ini, Dynamic Programming bisa memilih skill yang pada saat itu terlihat kurang menguntungkan, tetapi ternyata pemilihan skill itu penting untuk menguntungkan party di turn-turn berikutnya.

Di Greedy Algorithm, implementasinya hanya memilih skill dengan nilai terbaik pada turn sekarang. Cara ini lebih sederhana dengan model Dynamic Programming karena tidak perlu memperhatikan next turn. Namun karena hal itulah model ini tidak bisa mempertimbangkan jangka panjang karena permainannya memiliki cooldown dan resource untuk skill nya. Greedy mampu menghasilkan damage yang cukup besar, tetapi kurang memperhatikan kebutuhan bertahan hidup. Hal ini dapat dilihat pada skenario Standard Combat dan High Pressure Survival, ketika Greedy tidak menggunakan healing dengan baik sehingga party kalah sebelum mencapai batas turn.

Rotasi skill yang dihasilkan Dynamic Programming pada ketiga skenario adalah sebagai berikut.

1) Standard Combat:

Battle Chant → Power Strike → Guard Stance → Battle Chant → Healing Light → Power Strike → Guard Stance → Battle Chant → Healing Light → Ultimate Burst

2) Boss Kill Priority:

Battle Chant → Basic Attack → Power Strike → Battle Chant → Power Strike → Ultimate Burst

3) High Pressure Survival:

Guard Stance → Battle Chant → Power Strike → Guard Stance → Battle Chant → Healing Light → Ultimate Burst → Guard Stance → Battle Chant → Healing Light

Dari rotasi tersebut, dapat dilihat bahwa DP menggunakan skill yang cukup bervariasi. Skill yang dipilih tidak hanya berisikan skill damage, tetapi juga skill support, defense, dan healing.

Battle Chant cukup sering muncul pada rotasi DP. Skill ini hanya memberikan shield yang kecil, tetapi sangat berguna untuk menambah resource. Resource ini kemudian bisa digunakan untuk mengaktifkan skill dengan cost lebih tinggi, seperti Power Strike dan Ultimate Burst. Contohnya pada skenario Boss Kill Priority, DP menggunakan Battle Chant diawal untuk menambah resource, dan menggunakan Ultimate Burst di turn ke-6 untuk membunuh bos. Sementara itu, Greedy tidak mampu untuk mengalahkan boss walaupun party masih bertahan hidup sampai akhir turn.

Selain itu, Guard Stance dan Healing Light juga muncul pada skenario Standard Combat dan High Pressure Survival. Kedua skill ini berperan penting untuk menjaga party tetap hidup, karena memberikan shield dan heal yang cukup

signifikan. Pada skenario High Pressure Survival, penggunaan Guard Stance lebih dominan dibandingkan skenario lainnya. Hal ini dikarenakan skenario tersebut perlu menggunakan shield karena boss memiliki damage yang lebih tinggi agar party tidak kalah sebelum batas turn. Sementara itu, Greedy kurang memanfaatkan skill heal dan shield dengan baik sehingga pada skenario Standard Combat dan High Pressure Survival, Greedy kalah sebelum mencapai turn akhir.

Dapat kita lihat perbedaan rotasi skill DP untuk ketiga skenario yang menunjukkan bahwa DP dapat menyesuaikan strategi dengan tujuan pertarungan. Pada Boss Kill Priority, rotasi lebih diarahkan untuk mengumpulkan resource untuk menggunakan skill damage tinggi yang membutuhkanr resource. Sementara itu, pada High Pressure Survival, rotasi lebih banyak menggunakan skill defensif dan healing. Dari hal ini, dapat kita simpulkan bahwa hasil rotasi tidak bersifat tetap, tetapi berubah mengikuti kondisi dan tujuan masing-masing skenario.

D. Keterbatasan Model

Model pertarungan yang dibuat dan digunakan di eksperimen ini merupakan penyederhanaan dari game turn-based RPG yang sudah ada. Pada model ini, party itu hanya dianggap sebagai satu kesatuan/entitas dengan satu nilai HP dan satu penggunaan skill. Boss juga dibuat hanya satu entitas yang hanya bisa memberikan damage.

Selain dari hal itu, model ini juga belum mempertimbangkan beberapa mekanisme yang umum pada game RPG, seperti buff, debuff, critical hit, akurasi, speed, elemental advantage, efek status, dll.

Walaupun begitu, model ini sudah cukup untuk merepresentasikan masalah utama yang mau dianalisis, yaitu pemilihan rotasi skill dengan batasan cooldown dan resource. Hanya dengan adanya HP party, HP boss, skill, cooldown, dan resource sudah membuat setiap keputusan skill memiliki pengaruh terhadap kondisi turn berikutnya. Sehingga, model ini tetap relevan untuk menunjukkan bagaimana Dynamic Programming bisa digunakan untuk mencari rotasi skill yang optimal.

Model ini dapat dikembangkan lebih lanjut, seperti memisahkan party dan musuh menjadi beberapa karakter individual dengan pola serangan yang berbeda, menambahkan sistem turn order berdasarkan speed, memasukkan efek buff, debuff, dan-lain lainnya. Dengan pengembangan itu, model pertarungan akan menjadi lebih dekat dengan sistem turn-based RPG yang sebenarnya.

LINK GITHUB

https://github.com/llennmaha/turn-based-skill-rotation-dp

UCAPAN TERIMA KASIH

Saya mengucap syukur kepada Tuhan Yang Maha Esa, ataas segala kasih dan berkatnya yang menuntuk dan menyertai saya dan teman teman dari awal dimulainya mata kuliah Strategi Algoritma ini hingga saat selesainya tugas makalah sebagai tugas terakhir mata kuliah ini sehingga saya dapat menjalani perkuliahan mata kuliah Strategi Algoritma dengan penuh damai dan sejahtera.

Saya juga mengucap syukur kepada sosok yang telah mengajari dan membimbing kami, terutama kepada Pak Rinaldi Munir sebagai dosen K3 sehingga saya dapat menulis ini dengan menggabungkan ilmu yang saya dapat dari pembelajaran di kelas. Semoga hal yang saya buat ini bisa bermanfaat untuk khalayak umum.

REFERENCES

[1] R. Munir, "Pengantar Strategi Algoritma," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/01-Pengantar-Strategi-Algoritma-(2026).pdf . [Accessed: Jun. 16, 2026].

[2] R. Munir, "Algoritma Greedy Bagian 1," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-(2026)-Bag1.pdf . [Accessed: Jun. 16, 2026].

[3] R. Munir, "Algoritma Greedy Bagian 2," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/05-Algoritma-Greedy-(2026)-Bag2.pdf . [Accessed: Jun. 16, 2026].

[4] R. Munir, "Algoritma Greedy Bagian 3," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/06-Algoritma-Greedy-(2026)-Bag3.pdf . [Accessed: Jun. 16, 2026].

[5] R. Munir, "Program Dinamis Bagian 1," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/25-Program-Dinamis-(2026)-Bagian1.pdf . [Accessed: Jun. 16, 2026].

[6] R. Munir, "Program Dinamis Bagian 2," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/26-Program-Dinamis-(2026)-Bagian2.pdf . [Accessed: Jun. 16, 2026].

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, 18 Juni 2026

Mahatma Brahmana - 13524015