Penerapan Algoritma Optimasi Knapsack pada Sistem Combo Video Game Hyperdimension Neptunia Re;Birth1

Mohammad Husein Amrullah
Program Studi Teknik Informatika
Institut Teknologi Bandung
Indonesia
13516046@std.stei.itb.ac.id, mhuseinamr@gmail.com

***Abstract*—Combo adalah salah satu mekanisme dalam video game di mana suatu karakter melakukan serangkaian gerakan secara beruntun dengan urutan tertentu untuk mencapai suatu efek tertentu. Sistem combo sangat sering digunakan dalam video game bergenre RPG dan Fighting. Pada suatu video game, sistem combo dapat sudah terdefinisi dari awal, atau dapat diubah-ubah sesuai keinginan pemain. Untuk kasus combo yang dapat diubah-ubah, persoalan mendapatkan combo terbaik dengan batasan tertentu dapat dianggap sebagai sebuah persoalan knapscak. Dalam makalah ini, penulis akan membahas penggunaan algoritma Greedy untuk optimasi combo pada video game Hyperdimension Neptunia Re;Birth1.**

Keywords—combo, greedy, knapsack, RPG

I. PENDAHULUAN

Hyperdimension Neptunia adalah sebuah seri video game bergenre RPG yang dikembangkan oleh beberapa perusahaan video game Jepang yaitu Idea Factory, Compile Heart, dan Felistella. Seri ini diawali dengan sebuah video game dengan nama sama yang dirilis di Jepang untuk PlayStation 3. Pada perkembangan selanjutnya, video game pertama dari seri ini di-remake dengan judul Hyperdimension Neptunia Re;Birth1 (untuk selanjutnya, akan disebut dengan singkatan HDN1). Hingga pada saat makalah ini dibuat, telah dirilis setidaknya 15 game dalam seri ini. Setiap game memilki cerita dan mekanisme yang berbeda-beda. Makalah ini akan secara khusus membahas HDN1.

Setting dari HDN1 adalah Gamindustri, sebuah dunia fantasi yang terbagi menjadi 4 negara yang masing-masing dikuasai oleh seorang CPU (Console Patron Unit): Planeptune, Lastation, Lowee, dan Leanbox. Masing-masing negara merepresentasikan sebuah console video game: Lastation merepresentasikan Sony PlayStation, Lowee merepresentasikan Nintendo Wii, Leanbox merepresentasikan Microsoft Xbox, dan Planeptune merepresentasikan Sega Neptune.

Gambar 1. Tampilan peta Gamindustri

Pada awal permainan, pemain memainkan karakter Neptune, CPU dari Planeptune. Seiring berjalannya permainan, pemain dapat memilih beberapa karakter lain. Salah satu fokus utama HDN1 adalah eksplorasi dungeon. Pada berbagai dungeon yang tersebar di Gamindustri, pemain dapat mengumpulkan berbagai item, men-trigger event cerita, atau bertarung melawan berbagai monster (dan, kadang-kadang, karakter lain) yang ada pada dungeon tersebut.

Gambar 2. Karakter Neptune di dalam salah satu dungeon

HDN1 memiliki mekanisme pertarungan yang cukup kompleks, menggunakan sistem turn-based. Setelah pemain memutuskan untuk bertarung dengan suatu monster, pemain dapat memainkan karakter yang dimilikinya secara bergantian.

Pemain dapat mengubah posisi masing-masing karakter, melakukan serangan, menggunakan item, kabur, dst. Terdapat banyak jenis serangan yang dapat dipilih, tapi secara garis besar dapat dibagi menjadi serangan biasa dan serangan spesial.

Gambar 3. Animasi karakter Neptune (versi HDD) saat melakukan serangan

Serangan biasa pada HDN1 memiliki sistem combo yang dapat diubah-ubah sesuai keinginan pemain. Pada pembuatan combo, pemain diberikan sejumlah CP (Cost Points) untuk setiap karakter, dan pilihan serangan yang dapat digunakan. Pemain dapat memilih hingga 9 serangan (3 slot serangan untuk setiap tipe). Setiap serangan memiliki beberapa karakteristik, salah satunya adalah jumlah damage yang dihasilkan. Pada makalah ini, penulis merancang dan melakukan percobaan untuk membuat combo yang dapat memaksimalkan jumlah damage dengan mengaplikasikan persoalan knapsack pada sistem combo HDN1.

II. LANDASAN TEORI

A. *Persoalan Knapsack*

Persoalan knapsack adalah sebuah masalah optimasi yang dapat didefinisikan sebagai berikut:

Diberikan n buah objek dan sebuah knapsack (karung, tas, dsb.) dengan kapasitas bobot K. Setiap objek memiliki properti bobot (weight) wi, dan keuntungan (profit) pi. Bagaimana memilih objek-objek yang dimasukkan ke dalam knapsack sehingga tidak melebihi kapasitas knapsack namun memaksimumkan total keuntungan yang diperoleh?

Persoalan knapsack sering muncul pada masalah alokasi sumber daya, dan banyak dipelajari pada berbagai bidang seperti kombinatorial, ilmu komputer, kriptografi, dan matematika terapan. Beberapa contoh penerapan persoalan knapsack pada dunia nyata antara lain pemilihan investasi saham dan pembuatan kunci untuk algoritma kriptografi Merkle-Hellman.

Solusi dari persoalan knapsack dapat dinyatakan sebagai vektor n-tupel:

$x = \{x_1, x_2, \ldots, x_n\}$

di mana nilai x_i menunjukkan berapa banyak objek ke-i yang dimasukkan ke dalam knapsack. Sebagai contoh, X = {1, 0, 2, 0} adalah sebuah solusi yang memasukkan objek ke-1 sebanyak 1 buah dan objek ke-3 sebanyak 2 buah ke dalam knapsack, sedangkan objek ke-2 dan ke-4 tidak dimasukkan.

Berdasarkan pilihan nilai yang mungkin untuk x_i, terdapat tiga variasi utama dari persoalan knapsack:

1. Persoalan Knapsack 0/1

   Pada variasi ini, nilai x_i yang mungkin hanya 0 (menunjukkan objek tidak diambil) atau 1 (menunjukkan objek diambil)

2. Persoalan Knapsack Bounded

   Pada variasi ini, nilai x_i dapat merupakan nilai apa saja dari 0 sampai batas tertentu (terdapat beberapa objek identik x_i yang dapat dimasukkan ke dalam knapsack)

3. Persoalan Knapsack Unbounded

   Pada variasi ini, nilai x_i dapat merupakan nilai apa saja dari 0 sampai tak hingga (terdapat tak hingga banyaknya objek identik x_i yang dapat dimasukkan ke knapsack)

Secara matematis, persoalan knapsack dapat dirumuskan sebagai berikut:

maksimasi $\sum_{i=1}^{n} p_i x_i$

dengan batasan $\sum_{i=1}^{n} w_i x_i \leq K$

B. *Algoritma Greedy*

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Algoritma ini sederhana dan lempang (straightforward). Prinsip greedy adalah: “take what you can get now!” Ambil apa yang dapat diperoleh sekarang! Contoh penerapan prinsip greedy dalam kehidupan sehari-hari antara lain:

1. Memilih beberapa jenis investasi (penanaman modal)
2. Mencari jalur tersingkat dari Jakarta ke Klaten
3. Memilih jurusan di perguruan tinggi

Algoritma greedy membentuk solusi langkah per langkah (step by step). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.

Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang “tampaknya” memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (local optimum) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (global optimum).

Algoritma greedy secara umum dapat didefinisikan sebagai algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip “take what you can get now!”)

2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global

Pada setiap langkah di dalam algoritma greedy kita baru memperoleh optimum lokal. Bila algoritma berakhir, kita berharap optimum lokal menjadi optimum global. Algoritma greedy mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

Persoalan optimasi dalam konteks algoritma greedy disusun oleh elemen-elemen sebagai berikut:

1. Himpunan kandidat, C

Himpunan ini berisi elemen-elemen pembentuk solusi.

2. Himpunan solusi, S

Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. Fungsi seleksi, SELEKSI

Fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan, LAYAK

Fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar batasan yang ada.

5. Fungsi objektif

Fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Pseudo-code algoritma greedy secara umum adalah sebagai berikut:

```
Function greedy(input C: himpunan_kandidat) → himpunan_kandidat
{
Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}

Deklarasi
    x : kandidat
    S : himpunan_kandidat

Algoritma
    S ← {} { inisialisasi S dengan kosong }
    while (not SOLUSI(S)) and (C ≠ {}) do
        x ← SELEKSI(C) { pilih sebuah kandidat dari C }
        C ← C - {X} { elemen himpunan kandidat berkurang satu }
        if LAYAK(S union {x}) then
            S ← S union {x}
        endif
    endwhile
    { SOLUSI(S) or C = {} }

    if SOLUSI(S) then
        return S
    else
        write('tidak ada solusi')
    endif
```

Algoritma greedy tidak selalu berhasil memberikan solusi yang benar-benar optimum. Hal ini terutama disebabkan karena algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode exhaustive search). Selain itu, algoritma greedy sangat bergantung pada pemilihan fungsi SELEKSI yang tepat untuk menghasilkan pilihan yang optimum.

Meskipun demikian, jika jawaban terbaik mutlak tidak diperlukan, maka algoritma greedy sering berguna untuk menghasilkan solusi hampiran (approximation) optimum, daripada menggunakan algoritma yang lebih kompleks untuk menghasilkan solusi yang eksak.

III. Metode Percobaan

A. *Definisi Persoalan*

Sebelum mendefinisikan masalah pembuatan combo pada HDN1 sebagai persoalan knapsack, akan dijelaskan terlebih dahulu secara lebih rinci mengenai sistem combo pada HDN1.

Gambar 4. Tampilan menu pembuatan combo

Seperti yang terlihat pada gambar, pada dasarnya, suatu combo dalam HDN1 terdiri dari 4 serangan. Serangan pertama adalah default attack yang bergantung pada senjata yang dipakai oleh karakter yang bersangkutan. Serangan ke-2, ke-3, dan ke-4 merupakan bagian dari sistem combo yang dapat diubah-ubah. Pemain dapat memilih serangan manapun yang dia inginkan selama total CP (Cost Points) yang digunakan tidak melebihi kapasitas CP pada karakternya.

Pada tampilan gambar, setiap baris pada combo melambangkan tipe serangan. Masing-masing tipe serangan tersebut adalah Rush, Power, dan Break. Setiap tipe serangan memiliki ciri khas, tapi pada makalah ini dianggap sama untuk menyederhanakan persoalan. Kemudian, setiap kolom pada combo melambangkan pilihan serangan yang dapat diambil pada setiap urutan (ke-2, ke-3, atau ke-4).

Perlu digarisbawahi bahwa pada sistem combo ini, setiap serangan dapat dipilih lebih dari satu kali, sehingga dimungkinkan bagi pemain untuk memilih satu serangan secara beruntun untuk serangan ke-2 hingga ke-4. Selain itu, tidak setiap slot combo harus terisi. Hal ini berarti bahwa jika pemain memilih untuk mengosongkan semua slot combo-nya, maka dia tidak dapat melakukan combo sama sekali. Namun, jika setidaknya pada setiap kolom terisi satu serangan, maka pemain dapat melakukan combo seperti biasa (meskipun pilihannya menjadi terbatas).

Setiap serangan pada combo memiliki beberapa atribut. Atribut tersebut antara lain tipe combo, afinitas elemen, kategori (physical/magic), dsb. Pada percobaan ini, kita hanya akan memperhatikan 3 atribut yang berkaitan langsung dengan damage yang dihasilkan, yaitu atribut Hit Count, Power, dan Guard Damage.

Damage dasar yang dihasilkan suatu serangan dapat dihitung dengan rumus berikut:

Damage = Hit Count x (Power + Guard Damage)

Dari informasi di atas, dapat dibuat sebuah definisi persoalan sebagai berikut:

Diberikan nilai maksimum CP dan daftar serangan yang dapat dipilih, serangan mana sajakah yang harus dipilih untuk memaksimalkan nilai damage?

Tentu saja, pada praktiknya, nilai damage saja tidak cukup untuk memperoleh combo yang terbaik. Pemain juga perlu memperhatikan tipe serangan, jangkauan, dsb. untuk dapat menghadapi berbagai jenis lawan. Meskipun demikian, metode pemilihan combo yang didefinisikan pada makalah ini sudah cukup baik dalam membantu pemain untuk memenangkan permainan pada tingkat kesulitan Normal.

B. Implementasi Algoritma Optimasi Knapsack

Sejalan dengan definisi persoalan, pada persoalan ini dipilih algoritma greedy. Pemilihan algoritma greedy didasarkan pada pertimbangan berikut:

1. Untuk mempermudah perancangan algoritma
2. Dari pengamatan, ditemukan bahwa pada sebagian besar kasus, nilai kapasitas CP jauh lebih besar dari kebutuhan CP serangan. Oleh karena itu, algoritma greedy hampir selalu dapat menemukan combo dengan setidaknya 3 serangan. Hal ini meminimalisir kebutuhan untuk melakukan runut-balik.

Pseudo-code algoritma yang digunakan pada percobaan ini adalah sebagai berikut:

```
function pick_combo(input attacks : himpunan_attack,
max_cp : integer) → himpunan_attack

{
Mengembalikan himpunan attack yang terpilih
dimasukkan ke dalam combo dengan kapasitas max_cp

attack merupakan sebuah struktur data berisi
informasi nama attack, tipe attack, nilai cp yang
diperlukan, dan nilai damage

}


Deklarasi
    combo : himpunan_attack
    next_attack : attack
    current_cp, i, n : integer
    terpilih_serangan : boolean


Algoritma
    combo ← {}


    urutkan attacks menurun berdasarkan nilai damage


    next_attack ← elemen pertama attacks
    current_cp ← 0
    for i ← 1 to n do
        terpilih_serangan ← false
        while (not terpilih_serangan) do
            if (masih ada slot combo) and
(current_cp <= max_cp) then
                tambahkan next_attack pada combo
                tambahkan nilai cp next_attack pada
current_cp
                terpilih_serangan ← true
            else if (attacks belum habis) then
                next_attack ← elemen attacks
berikutnya
            else
                return combo


    return combo
```

C. Prosedur Percobaan

Langkah-langkah percobaan secara prosedural adalah sebagai berikut:

1. Pilih sebuah karakter yang akan dibuat combonya
2. Catat semua pilihan serangan yang dimiliki karakter tersebut
3. Catat nilai kapasitas CP yang dimiliki karakter tersebut
4. Aplikasikan algoritma optimasi knapsack pada data no (2) dan (3)
5. Catat hasil eksekusi algoritma

Untuk percobaan pada makalah ini, dipilih 3 karakter berikut:

1. Neptune, pada level 2 (6 pilihan serangan, 155 kapasitas CP)
2. Compa, pada level 22 (6 pilihan serangan, 211 kapasitas CP)
3. IF, pada level 45 (9 pilihan serangan, 275 kapasitas CP)

IV. Hasil dan Pembahasan

A. *Hasil Combo*

1. Percobaan Pertama

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Rush	Rush	25	140
2	Rapid Rush	Rush	37	219
3	Power Edge	Power	20	170
4	Power Slash	Power	30	236
5	Break Hit	Break	16	251
6	Break Crush	Break	24	429

Tabel 1. Kandidat serangan untuk percobaan pertama

Gambar 5. Hasil combo untuk percobaan pertama

Pada percobaan pertama, diperoleh combo sebagai berikut:

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Power Slash	Power	30	236
2	Power Slash	Power	30	236
3	Power Edge	Power	20	170
4	Break Crush	Break	24	429
5	Break Crush	Break	24	429
6	Break Crush	Break	24	429

Tabel 2. Solusi serangan untuk percobaan pertama

Combo ini mempunyai batas atas damage sebesar
429 + 429 + 429 = 1287

2. Percobaan Kedua

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Bonk!	Rush	25	132
2	Stab-y, Stab-y!	Rush	37	204
3	Injection Time!	Rush	50	213
4	Fire, Syringe!	Power	20	170
5	Slap	Break	16	381
6	Poison	Break	16	251

Tabel 3. Kandidat serangan untuk percobaan kedua

Gambar 6. Hasil combo untuk percobaan kedua

Pada percobaan kedua, diperoleh combo sebagai berikut:

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Injection Time!	Rush	50	213
2	Injection Time!	Rush	50	213
3	Injection Time!	Rush	50	213
4	Slap	Break	16	381
5	Slap	Break	16	381
6	Slap	Break	16	381

Tabel 4. Solusi serangan untuk percobaan kedua

Combo ini mempunyai batas atas damage sebesar
381 + 381 + 381 = 1143

3. Percobaan Ketiga

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Rush	Rush	25	132
2	Rapid Hit	Rush	37	204
3	Demon Slice	Rush	62	308
4	Power Hit	Power	20	170
5	Powered Hit	Power	30	240
6	Break Hit	Break	16	251
7	Venom Edge	Break	24	426
8	Paralyze Edge	Break	24	424
9	Spectral Edge	Break	32	584

Tabel 5. Kandidat serangan untuk percobaan ketiga

Gambar 7. Hasil combo untuk percobaan ketiga

Pada percobaan ketiga, diperoleh combo sebagai berikut:

No.	Nama Serangan	Tipe Serangan	CP	Damage
1	Demon Slice	Rush	62	308
2	Demon Slice	Rush	62	308
3	Powered Hit	Power	30	240
4	Power Hit	Power	20	170
5	Spectral Edge	Break	32	584
6	Spectral Edge	Break	32	584
7	Spectral Edge	Break	32	584

Tabel 6. Solusi serangan untuk percobaan ketiga

Combo ini mempunyai batas atas damage sebesar
584 + 584 + 584 = 1752

B. Analisis hasil

Dari data yang diperoleh, dapat ditarik beberapa kesimpulan terkait algoritma optimasi knapsack yang dipakai. Pertama, algoritma ini cenderung memilih satu serangan yang sama (dengan damage terbesar) secara berulang-ulang. Hal ini bukanlah sesuatu yang tidak wajar. Bahkan, pada praktiknya, teknik ini memang dapat menghasilkan damage terbesar. Hal ini akan tampak jelas jika kita menuliskan daftar damage maksimal yang bisa diperoleh suatu serangan, kemudian membandingkannya dengan hasil algoritma. Dengan mengasumsikan semua serangan dapat dipilih maksimal tiga kali, rumus perhitungan damage maksimal adalah

Max Damage = Damage x 3

No.	Nama Serangan	Damage	Max Damage
1	Rush	140	420
2	Rapid Rush	219	657
3	Power Edge	170	510
4	Power Slash	236	708
5	Break Hit	251	753
6	Break Crush	429	1287

Tabel 7. Tabel damage maksimal untuk percobaan pertama

Nilai maksimum dari combo untuk percobaan pertama adalah 1287, sama dengan nilai maksimum kolom Max Damage pada tabel di atas.

No.	Nama Serangan	Damage	Max Damage
1	Bonk!	132	369
2	Stab-y, Stab-y!	204	612
3	Injection Time!	213	639
4	Fire, Syringe!	170	510
5	Slap	381	1143
6	Poison	251	753

Tabel 8. Tabel damage maksimal untuk percobaan kedua

Nilai maksimum dari combo untuk percobaan kedua adalah 1143, sama dengan nilai maksimum kolom Max Damage pada tabel di atas.

No.	Nama Serangan	Damage	Max Damage
1	Rush	132	396
2	Rapid Hit	204	612
3	Demon Slice	308	924
4	Power Hit	170	510
5	Powered Hit	240	720
6	Break Hit	251	753
7	Venom Edge	426	1278
8	Paralyze Edge	424	1272
9	Spectral Edge	584	1752

Tabel 9. Tabel damage maksimal untuk percobaan ketiga

Nilai maksimum dari combo untuk percobaan ketiga adalah 1752, sama dengan nilai maksimum kolom Max Damage pada tabel di atas.

Kedua, algoritma ini cenderung memilih serangan bertipe Rush dan Break. Meskipun pada praktiknya, serangan yang dapat menghasilkan damage efektif paling besar adalah Power (karena nilai atribut Power yang tinggi) dan Break (karena nilai atribut Guard Damage yang tinggi). Serangan bertipe Rush cenderung menghasilkan damage efektif yang kecil karena nilai atribut Power dan Guard Damage yang kecil. Namun, pada algoritma ini, Rush dianggap memiliki potensi damage yang besar karena nilai atribut Hit Count yang jauh di atas Power dan Break.

V. PENUTUP

Dari percobaan yang telah dilakukan, dapat disimpulkan bahwa algoritma yang dirancang oleh penulis telah berhasil memberikan solusi knapsack yang cukup bagus untuk menyelesaikan masalah pembuatan combo pada video game RPG. Algoritma ini dapat dikembangkan lebih lanjut, misalnya untuk pembuatan sistem saran combo pada video game selanjutnya.

Percobaan yang telah dilakukan juga dapat dikembangkan lagi dengan mencoba algoritma lain untuk menyelesaikan permasalahan, misalnya dengan menggunakan pendekatan penelusuran graf (DFS, BFS, Branch & Bound, dsb.) atau program dinamis.

UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan syukur dan terima kasih kepada Tuhan Yang Maha Esa atas berkat-Nya sehingga

makalah ini dapat terselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir atas materi yang diberikan baik melalui diktat maupun website. Tak lupa juga penulis mengucapkan terima kasih kepada rekan-rekan terutama Saudara Akmal Narendra S. yang telah membantu penulis dalam memperoleh video game yang menjadi sumber inspirasi dari makalah ini.

REFERENSI

[1] R. Munir, Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Institut Teknologi Bandung, 2018.

[2] Idea Factory International, situs resmi Hyperdimension Neptunia Re;Birth1. http://ideafintl.com/rebirth1/. Diakses pada 13 Mei 2018.

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, 14 Mei 2018

Mohammad Husein Amrullah, 13516046