Aplikasi *Uniform Cost Search* untuk *Minimaxing* Artefak Karakter pada Gim "Genshin Impact"

Josep Marcello/13519164[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]jspmarcello@live.com

***Abstrak*—Dewasa ini, salah satu hiburan yang populer adalah gim. Media hiburan yang sudah ada sejak tahun 1980-an ini sangat berkembang sampai memiliki banyak judul. Salah satu judul terbaru yang menggarap kesuksesan adalah "Genshin Impact." Pada permainan "Genshin Impact," pemain dapat meningkatkan atributnya menggunakan barang yang disebut artefak. Pada makalah ini akan dibahas cara memilih dari beberapa artefak untuk membuat karakter dengan kombinasi atribut optimal menggunakan algoritma UCS. Pada makalah ini akan dibahas juga alternatif untuk meningkatkan algoritma yang digunakan.**

***Kata kunci—Uniform Cost Search*, *Minimaxing*, gim, Genshin Impact**

I. Pendahuluan

Salah satu sumber hiburan di zaman modern ini adalah gim atau permainan video. Gim sebenarnya seperti keturunan langsung dari permainan tradisional, seperti permainan papan. Hanya saja, dibuat menjadi media hiburan interaktif. Karena sifatnya yang menyenangkan, tidak mengejutkan kalau gim menjadi sangat populer dan sangat besar di zaman ini.

Seperti industri hiburan lainnya, gim memiliki banyak genre. Salah satu genre yang populer adalah RPG atau *role-playing game*. Genre ini merupakan keturunan langsung dari genre permainan papan dengan nama yang sama.

Pada genre RPG, pemain gim menjadi salah satu karakter pada dunia gim. Disebut *role-playing* karena pemain dapat mengambil peran pada permainan yang dimainkannya. Setiap karakter seperti seseorang pada dunia nyata yang memiliki keterampilan dan perang masing-masing pada dunia gim.

Layaknya seseorang pada dunia nyata, karakter pemain pada permainan RPG juga memiliki keterampilan yang dapat dikembangkan. Misalnya, di dunia nyata dengan belajar memasak dapat ditingkatkan keterampilan memasak. Bedanya dengan dunia nyata adalah cara bagaimana keterampilan ditingkatkan. Keterampilan ini bisa ditingkatkan dengan latihan juga, dengan menggunakan poin keterampilan, dengan menggunakan barang, dan lain-lain.

Pada makalah ini akan dibahas bagaimana cara memilih dari himpunan barang sehingga dapat dibuat karakter yang memiliki keterampilan (atribut) optimal dengan optimasi menuju keterampilan pilihan pemain. Pemilihan barang dilakukan dengan strategi algoritma UCS.

II. Landasan Teori

A. Minimaxing

Minimaxing adalah salah aturan keputusan yang sering digunakan dalam teori permainan, *artificial intelligence*, dan berbagai bidang matematika lainnya. *Minimaxing* adalah strategi yang digunakan dalam menentukan pilihan agar memaksimalkan pendapatan dan meminimalkan kerugian (optimal). Contoh persoalan optimasi dengan *minimaxing* adalah mencari luas bidang atau ruang maksimum dengan beberapa parameter yang diberikan.

Dalam matematika, *minimax* sering digunakan untuk dua hal, yaitu *linear programming* (LP) dan teori keputusan (*decision theory*). LP adalah metode atau strategi yang dapat digunakan untuk optimasi suatu luaran, misalnya memaksimalkan keuntungan atau menimalkan kerugian. Teori keputusan menyangkut masalah optimasi yang melibatkan orang lain. Misalnya memaksimalkan keuntungan pemain dan memaksimalkan kerugian pada pemain lawan.

B. Uniform Cost Search

Uniform Cost Search (UCS) adalah salah satu algoritma pencarian pada graf untuk menentukan "jarak" terpendek dari sudut awal ke sebuah sudut tujuan. Pencarian dengan UCS termasuk ke dalam algoritma pencarian *uninformed search* dan merupakan variasi dari algoritma Dijkstra. Perbedaan antara UCS dengan algoritma Dijkstra:

- algoritma Djikstra akan mencari jarak terpendek/termurah dari sudut awal ke semua sudut di graf, sedangkan UCS hanya mencari lintasan terpendek ke salah satu sudut tujuan;
- UCS lebih murah konsumsi ruang memorinya dibandingkan Djikstra dan juga lebih cepat;
- UCS lebih umum divisualisasikan sebagai pohon, sedangkan Djikstra digunakan pada graf "biasa";
- algoritma Dijkstra hanya bisa digunakan pada graf yang simpul-simpulnya sudah diketahui, sedangkan UCS bisa digunakan pada graf yang simpulnya terus berkembang/bertambah seiring berjalannya algoritma.

Cara kerja algoritma UCS adalah dengan mengekspansi sudut-sudut tetangga dengan *cost* termurah. Algoritma UCS memanfaatkan *priority queue* untuk menyimpan sudut dan

$g(n)$. $g(n)$ adalah *cost* dari akar untuk mencapai sudut ke-n. *priority queue* dibuat terurut naik berdasarkan nilai $g(n)$.

Proses algoritma UCS adalah

1) akar dimasukkan ke dalam *priority queue* (pq),
2) jika pq kosong, berhenti,
3) kunjungi (*dequeue* dan "ambil") sudut pertama pada pq, sebut sudut e,
4) jika e adalah tujuan, berhenti,
5) jika e sudah pernah dikunjungi, ulangi langkah ke-2,
6) ekspansi e: hitung nilai $g(n)$ sudut-sudut tetangga e lalu masukkan ke pq,
7) tandai e sudah pernah dikunjungi,
8) ulangi langkah ke-2.

Berikut merupakan *pseudocode* (kurang lebih bahasa C++) untuk algoritma UCS:

Listing 1
PSEUDOCODE UCS DALAM C++ [6]
(DENGAN PERUBAHAN SECUKUPNYA)

```
/**
 * Fungsi UCS.
 * Sudut direpresentasikan dengan sebuah karakter
 * @param graph graf, representasi adjacency list
 * @param cost harga setiap sisi pada graf
 * @param goal sudut-sudut tujuan
 * @param start sudut awal
 */
vector<int>
uniform_cost_search(map<char, vector<char>>& graph,
                    map<pair<char, char>, int>& cost,
                      char start,
                      char goal)
{
  /* minimum cost upto
   * goal state from starting
   * state */
  int answer;

  // elemen pair:
  // - first: cost
  // - second: sudut
  priority_queue<pair<int, char>> queue;

  // set the answer vector to max value
  for (int i = 0; i < goal.size(); i++)
    answer.push_back(INT_MAX);

  // insert the starting node
  queue.push(make_pair(0, start));

  // map to store visited node
  map<char, bool> visited;
  for (const pair<char, vector<char>>& kv: graph)
    visited[kv.first] = false;

  // while the queue is not empty
  while (queue.size() > 0) {
    /* get the top element of the
     * priority queue */
    pair<int, char> p = queue.top();

    // pop the element
    queue.pop();

    // get the original value
    p.first *= -1;

    /* check if the element is part of
     * the goal list */
    if (p.second == goal) {
        answer = p.first;

      // if all goals are reached
      return answer;
    }

    /* check for the non visited nodes
     * which are adjacent to present node */
    if (!visited[p.second])
      for (int i = 0;
           i < graph[p.second].size();
           i++) {
        /* value is multiplied by -1 so that
         * least priority is at the top */
        queue.push(make_pair((p.first +
          cost[make_pair(p.second,
    graph[p.second][i])]) * -1,
           graph[p.second][i]));
      }

    // mark as visited
    visited[p.second] = true;
  }

  return answer;
}
```

C. *"Genshin Impact"*

"Genshin Impact" (GI) adalah permainan buatan studio gim dari Shanghai, RRT bernama miHoYo. Gim GI dirilis pada 28 September 2020 untuk PC Windows, gawai Android dan iOS, dan konsol Sony PlayStation. Gim ini sukses besar dan berhasil menarik banyak pemain dari seluruh kalangan serta seluruh belahan dunia. Sayangnya, gim ini harus dimainkan secara daring[1], tapi dapat dimainkan secara bersama-sama sampai dengan 4 orang.

Pada GI, pemain berperan sebagai seorang petualang (*traveler*) di tempat bernama Teyvat. Selain petualang, pemain juga dapat menggunakan karakter lain. Cara mendapatkan karakter adalah dengan sistem *gacha* (undian).

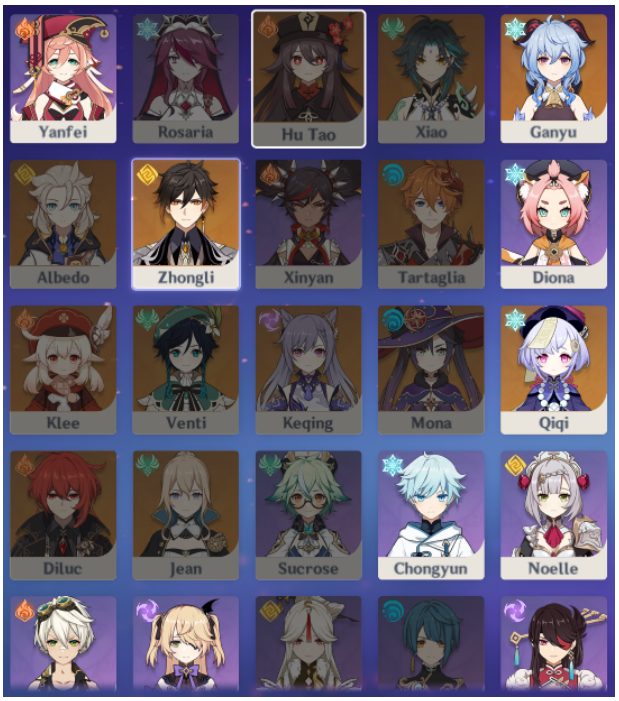
Sistem undian pada "Genshin Impact" tidak hanya digunakan pada sistem mendapatkan karakter. Sistem mendapatkan barang-barang di gim ini juga menggunakan. Barang-barang yang dimaksud di sini mencakup:

- artefak,
- senjata, dan
- bahan *level-up*.

1) Elemen: Setiap karakter dan musuh pada gim "Genshin Impact" memiliki elemen utamanya masing-masing. Elemen-elemen ini terdiri dari: *cryo*, *pyro*, *hydro*, *dendro*, *electro*, *anemo*, dan *geo*. Efek elemen biasanya dapat diaktifkan dengan cara mengaktifkan *elemental skill* atau *elemental burst* lalu dikenakan ke entitas target. Efek-efek dari elemen dapat digabungkan sehingga dapat menghasilkan sebuah reaksi elemen. Reaksi-reaksi elemen terdiri atas:

- *frozen* (*cryo* + *hydro*),
- *vaporize* (*hydro* + *pyro*),
- *electro-charged* (*hydro* + *electro*),
- *overloaded* (*pyro* + *electro*),

[1] seperti kuliah

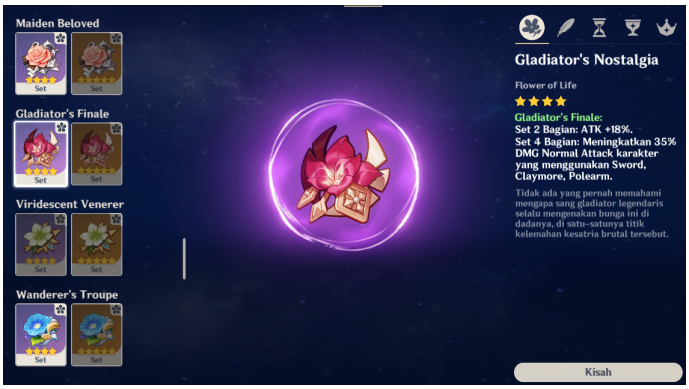


Gambar 1. Beberapa karakter yang dapat dimainkan pada "Genshin Impact"

- *superconduct* (*cryo* + *electro*),
- *burning* (*pyro* + *dendro*),
- *swirl* (*anemo* + elemen lainnya selain *geo*),
- *melt* (*cryo* + *pyro*), dan
- *crystallize* (*geo* + elemen lainnya selain *dendro* dan *anemo*),

2) Atribut: Selain elemen, setiap karakter pada gim GI memiliki atribut masing-masing. Atribut itu terdiri dari:

1) *maximum* HP,
2) atk (*attack*),
3) def (*defense*),
4) *maximum stamina*,
5) *crit rate*,
6) *crit damage*,
7) *healing bonus*,
8) *incoming healing bonus*,
9) *energy recharge*,
10) *cooldown reduction*,
11) *shield strength*,
12) *pyro damage bonus*,
13) *pyro damage resistance*,
14) *hydro damage bonus*,
15) *hydro damage resistance*,
16) *dendro damage bonus*,
17) *dendro damage resistance*,
18) *electro damage bonus*,
19) *electro damage resistance*,
20) *anemo damage bonus*,
21) *anemo damage resistance*,
22) *geo damage bonus*,



Gambar 2. Beberapa artefak pada gim "Genshin Impact"

23) *geo damage resistance*,
24) *cryo damage bonus*,
25) *cryo damage resistance*,
26) *physical damage bonus*, dan
27) *physical damage resistance*.

Hampir ke-27 atribut karakter dapat dipengaruhi oleh reaksi elemen, kombinasi elemen pada tim (disebut resonansi elemen), dan artefak.

3) Artefak: Artefak adalah barang yang dapat dikenakan karakter untuk meningkatkan atribut karakternya. Sayangnya, atribut apa yang akan dipengaruhi dan besar pertambahannya juga bagian dari sistem undian, jadi untuk mendapatkan artefak yang sempurna butuh banyak waktu dan usaha. Setiap karakter memiliki 5 kategori artefak yang dapat dikenakan dengan maksimum artefak setiap kategori adalah satu. Artefak juga dibagi berdasarkan set dan jika 2 atau 4 bagian dari sebuah set dikenakan, maka karakter akan mendapatkan bonus tambahan peningkatan atribut. Contoh penambahan peningkatan atribut adalah penambahan atk sebesar 18% ketika menggunakan 2 atau 3 artefak dari set *Gladiator's Finale*; penambahan atk sebesar 18% dan meningkatkan *damage* serangan untuk karakter yang menggunakan pedang, *claymore*, atau *polearm* ketika menggunakan 4 atau 5 artefak dari set dari set *Gladiator's Finale*.

Seperti yang sudah dijelaskan, artefak pada "Genshin Impact" terdiri dari lima kategori. Lima kategori artefak yang ada pada genshin adalah

- *Flower of Life* (ikon bunga),
- *Plume of Death* (ikon bulu burung),
- *Sands of Eon* (ikon jam pasir),
- *Goblet of Eonothem* (ikon cawan), dan
- *Circlet of Logos* (ikon mahkota).

Penambahan atribut akibat dari artefak biasanya dilakukan dengan nilai tetap (*flat*), tapi untuk atribut *maximum hp*, atk, dan def dapat ditambahkan dengan rasio/persentase. Oleh karena itu, rumus yang digunakan untuk menghitung nilai baru atribut akibat pertambahan oleh artefak berbeda untuk ketiga atribut itu. Pada umumnya, rumus menghitung nilai

baru atribut adalah

$$\text{atribut}_{baru} = atribut_{karakter} + atribut_{artefak}. \quad (1)$$

Keterangan:

$atribut_{baru}$: nilai atribut setelah dihitung,
$atribut_{karakter}$: nilai atribut yang dimiliki karakter,
$atribut_{artefak}$: nilai atribut dari artefak.

Sedangkan rumus menghitung nilai baru atribut atk:

$$\begin{aligned} atk_{baru} =& [(atk_{karakter} + atk_{senjata}) \times \\ & (1 + atk_{artefak\_persen})] + \\ & atk_{artefak\_tetap}. \end{aligned} \quad (2)$$

Keterangan:

atk_{baru}: nilai atk setelah dihitung,
$atk_{karakter}$: nilai atk yang dimiliki karakter,
$atk_{senjata}$: nilai atk dari senjata,
$atk_{artefak\_persen}$: nilai atk dari artefak (dalam persentase).
$atk_{artefak\_tetap}$: nilai atk dari artefak (bukan persentase).

Lalu *maximum* HP dan def memiliki rumus:

$$\begin{aligned} atribut_{baru} =& [atribut_{karakter} \times \\ & (1 + atribut_{artefak\_persen})] + \\ & atribut_{artefak\_tetap}. \end{aligned} \quad (3)$$

Keterangan:

$atribut_{baru}$: nilai atribut setelah dihitung,
$atribut_{karakter}$: nilai atribut yang dimiliki karakter itu,
$atribut_{artefak\_persen}$: nilai atribut dari artefak (dalam persentase).
$atribut_{artefak\_tetap}$: nilai atribut dari artefak (bukan persentase).

III. Analisis dan Pembahasan

A. Pemetaan Masalah ke UCS

Masalah ini dapat dibuat pohon dengan 6 tingkat kedalaman. Pada kedalaman ke-0 (akar), artinya karakter tidak menggunakan artefak apa-apa. Lalu, kedalaman pertama memiliki simpul-simpul yang merupakan artefak dari kategori *Flower of Life*. Kedalaman kedua adalah artefak dari kategori *Plume of Death*, kedalaman ketiga adalah kategori *Sands of Eon*, keempat adalah kategori *Goblet of Eonothem*, dan kedalaman kelima adalah artefak dari kategori *Circlet of Logos*.

Jika dianggap pemain mengenakan artifak dari kategori *Flower of Life* sampai kategori *Circlet of Logos* (kedalaman pertama sampai kedalaman kelima) secara terurut; masalah ini dapat dianggap seperti masalah UCS dengan sudut awalnya adalah akar (tidak mengenakan artefak) dan sudut tujuan adalah sudut-sudut pada kedalaman kelima. Lalu karena diinginkan maksimasi (karakter yang digunakan pemain memiliki atribut semaksimal mungkin), UCS harus diubah sedikit. Algoritma UCS yang biasanya mencari/ekspansi terurut berdasarkan *cost* termurah, harus diubah menjadi ekspansi terurut berdasarkan *cost* **termahal**.

Penentuan *cost* didasarkan pada "skor karakter" (SK). SK didapatkan dengan cara pertama-tama menentukan prioritas pemain ingin mengoptimasi untuk atribut apa. Pemain dapat memilih 5 atribut yang terurut dari yang paling ingin dioptimasi sampai tidak terlalu ingin dioptimasi. Atribut yang paling ingin dioptimasi (prioritas 1) akan dikalikan dengan konstanta 6, kedua ingin dioptimasi (prioritas 2) dikalikan dengan konstanta 5, dan seterusnya sampai (prioritas 5) dikalikan dengan 2. Atribut yang bukan prioritas akan dikalikan dengan konstanta 1. SK dapat dihitung dengan rumus:

$$SK = \sum_{i=1}^{27} (atribut_i \times konstanta\_prioritas_i) \quad (4)$$

Keterangan:

$atribut_i$: atribut indeks ke-i (lihat: bagian II-C2),
$konstanta\_prioritas_i$: Konstanta prioritas untuk atribut ke-i.

Misalnya karakter pemain dengan atribut (jika atribut tidak dituliskan artinya memiliki nilai 0):

- *maximum* HP = 12.727 (prioritas 5),
- atk (*attack*) = 1.149 (prioritas 1),
- def (*defense*) = 686,
- *maximum stamina* = 224,
- *crit rate* = 29.3 (prioritas 2),
- *crit damage* = 83 (prioritas 4),
- *energy recharge* = 157.8,
- *cryo damage bonus* = 23.1 (prioritas 3),

akan memiliki:

$$\begin{aligned} SK = & \quad 12727 \times 2 + 1149 \times 6 + 676 \times 1 + 224 \times 1 + \\ & 29.3 \times 5 + 83 \times 3 + 157.8 \times 1 + 23.1 \times 4 \\ = & \quad 33.893,7 \end{aligned}$$

Dengan memanfaatkan "skor karakter," dapat dibuat rumus untuk menghitung *cost* setiap simpul. *Cost* artefak ke-i adalah

$$\begin{aligned} g(i) =& SK \text{ jika dikenakan semua} \\ & \text{artefak dari akar sampai} \\ & \textit{parent} \text{ dari artefak ke-}i \\ & \text{dan juga artefak ke-}i. \end{aligned} \quad (5)$$

Untuk menghitung nilai atribut karakter baru setelah mengenakan satu atau lebih artefak dapat menggunaakn rumus (1), (3), dan (2) yang sudah disederhanakan menjadi:

$$\begin{aligned} atk_{baru} =& [atk_{karakter} \times (1 + atk_{artefak\_persen})] \\ & + atk_{artefak\_tetap}. \end{aligned} \quad (6)$$

Penyederhanaannya adalah $atk_{karakter}$ digabungkan dengan $atk_{senjata}$. Penyederhanaan dilakukan agar rumus untuk atk sama dengan rumus untuk *maximum* HP dan def. Selain itu, penyederhanaan juga akan mempermudah ketika pengguna membuat masukkan karena atribut atk yang ditunjukkan pada layar atribut karakter di gim adalah hasil penjumlahan atribut atk karakter dengan atribut atk senjatanya.

Perhitungan bonus tambahan peningkatan atribut dari artefak hanya akan digunakan untuk bonus yang didapatkan juga mengenakan 2 bagian dari sebuah set. Hal ini dilakukan karena bonus yang didapatkan dari mengenakan 4 bagian dari sebuah set harus memenuhi syarat yang membuat algoritma semakin kompleks karena pengecekan harus dilakukan lebih banyak.

Jika karakter mengenakan lebih dari satu artefak, dijumlahkan dahulu semua atribut tambahan dari artefak lalu baru dimasukkan ke rumus yang dibutuhkan. Misalnya, sebuah karakter, dengan atribut atk 1.149, artefak 1 pada kedalaman ke-1, dan pemain mempertimbangkan untuk menggunakana artefak 2 pada kedalaman ke-2. Data atribut artefak:

- artefak 1:
  - atk: +18%,
  - atk: 20, dan
- artefak 2:
  - atk: +8%,
  - atk: 36.

Cara menghitung nilai atribut atk yang barunya adalah

$$\begin{aligned} atk_{baru} &= \{1.149 \times [1 + (18\% + 8\%)]\} + (20 + 36) \\ &= (1.149 \times 1.26) + 56 \\ &= 1.503,74. \end{aligned}$$

Alur kerja algoritma UCS yang digunakan adalah alur kerja algoritma UCS dengan beberapa sudut tujuan (lihat: bagian II-B).

B. Pengujian

Untuk menguji algoritma UCS yang dibuat, akan dibuat beberapa data *dummy*. Data *dummy* ini terdiri atas data dari karakter tanpa artefak (kondisi akar pohon UCS) dan data beberapa artefak.

Untuk memudahkan pengujian, akan dilakukan beberapa penyederhanaan, yaitu

- dari 27 atribut yang ada, hanya akan dipilih 6 atribut,
- hanya 3 kategori artefak yang digunakan, dan
- bonus dari set artefak dianggap tidak ada.

Atribut-atribut yang dipilih adalah:

1) *maximum* HP (HP),
2) *attack* (ATK),
3) *crit rate* (CR),
4) *crit damage* (CD),
5) *energy recharge* (ER), dan
6) *elemental mastery* (EM).

Data *dummy* atribut dasar karakter (tanpa artefak):

- HP = 11.294,
- ATK = 921,
- CR = 5,
- CD = 50
- ER = 100, dan
- EM = 0.

Lalu, misalkan prioritas atribut pemain adalah

1) ATK,
2) CD,
3) CR,
4) ER,
5) HP, dan
6) EM.

Data *dummy* artefak[2]:

- kategori *Flower of Life*,

Tabel I
DATA *DUMMY* ARTEFAK PADA KATEGORI *FLOWER OF LIFE*

Nama	HP	HP%	ATK	ATK%	CR	CD	ER	EM
$F0$	645	0	16	0	0	0	3.6	0
$F1$	645	3.7	0	3.3	0	0	0	0
$F2$	645	0	0	0	0	0	4.1	0

- kategori *Plume of Death*, dan

Tabel II
DATA *DUMMY* ARTEFAK PADA KATEGORI *PLUME OF DEATH*

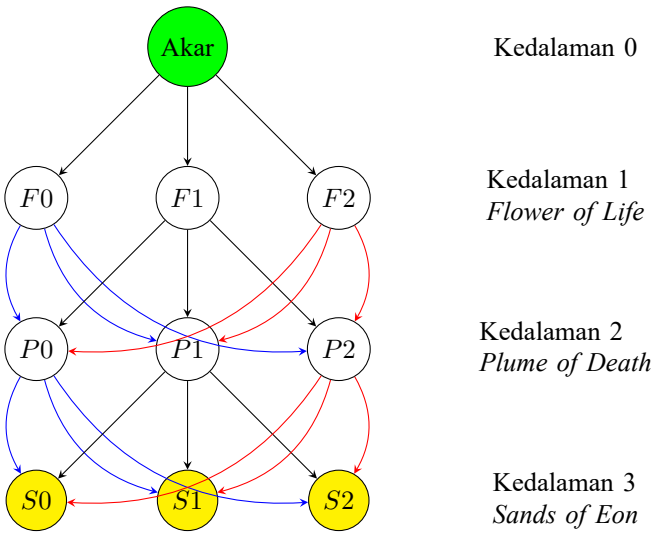
Nama	HP	HP%	ATK	ATK%	CR	CD	ER	EM
$P0$	0	0	42	0	2.2	0	0	0
$P1$	0	0	42	3.3	0	0	0	0
$P2$	0	0	42	4.2	0	0	0	0

- kategori *Sands of Eon*.

Tabel III
DATA *DUMMY* ARTEFAK PADA KATEGORI *SANDS OF EON*

Nama	HP	HP%	ATK	ATK%	CR	CD	ER	EM
$S0$	0	12	6.3	0	0	0	5.2	0
$S1$	0	6.3	16	3.3	0	0	0	0
$S2$	0	0	0	0	0	6.2	0	25

Dengan data yang ada, maka bisa dimulai proses algoritma UCS-nya. Berikut adalah ilustrasi pohon yang digunakan untuk UCS[3] dan ilustrasi setiap iterasinya



Gambar 3. Ilustrasi pohon untuk UCS

[2]ATK% dan HP% adalah penambahan dengan rasio
[3]Simpul asal berwarna hijau, simpul tujuan berwarna kuning

Tabel IV
Ilustrasi iterasi UCS

Iterasi	Simpul ekspan	Simpul hidup
1	Akar	$g(F0) = 30.070$ $g(F1) = 30.992,114$ $g(F2) = 29.986,3$
2	$F1$	$g(F0) = 30.070$ $g(F2) = 29.986,3$ $g(P0_{F1}) = 31.252,914$ $g(P1_{F1}) = 31.426,472$ $g(P2_{F1}) = 31.476,206$
3	$P2_{F1}$	$g(F0) = 30.070$ $g(F2) = 29.986,3$ $g(P0_{F1}) = 31.252,914$ $g(P1_{F1}) = 31.426,472$ $g(S0_{P2,F1}) = 34.240,166$ $g(S1_{P2,F1}) = 33.109,844$ $g(S2_{P2,F1}) = 31.509,706$
4	$S0_{P2,F1}$	$g(F0) = 30.070$ $g(F2) = 29.986,3$ $g(P0_{F1}) = 31.252,914$ $g(P1_{F1}) = 31.426,472$ $g(S1_{P2,F1}) = 33.109,844$ $g(S2_{P2,F1}) = 31.509,706$
	Sudah sampai solusi	

Berdasarkan algoritma, artefak yang harus dipilih adalah $F1$, $P2$, dan $S0$. SK akhir untuk kombinasi ini adalah 34.240,166. Kombinasi artefak yang dipilih menghasilkan atribut akhir karakter:

- HP = 13.712,158,
- ATK = 1038.375,
- CR = 5,
- CD = 50
- ER = 105.2, dan
- EM = 0.

Atribut atk akhir tidak memiliki atk yang maksimal. Akan tetapi, secara keseluruhan atribut karakter sudah paling optimal. Meskipun prioritas utama adalah atribut atk, tapi ada prioritas untuk atribut lain sehingga hasil akhir dari algoritma ini menghasilkan atribut yang secara keseluruhan sudah paling optimal (menurut algoritma ini).

IV. Kesimpulan

Algoritma ini dapat menghasilkan kombinasi artefak pada karakter pada gim "Genshin Impact." Meskipun atribut prioritas pertama belum tentu memiliki nilai paling optimal, tapi secara keseluruhan atribut karakter pasti sudah paling optimal (menurut algoritma). Algoritma ini juga dipastikan akan menemukan solusi optimal pada iterasi ke-4.

Algoritma ini pasti menemukan solusi optimal pada iterasi ke-4 karena bisa. Hal ini dapat terjadi karena, ekspansi pasti selalu ke kedalaman yang lebih dalam. Hal ini karena *cost* dari suatu simpul tidak akan menghasilkan nilai yang lebih kecil daripada *cost* simpul lainnya di kedalaman yang lebih tidak dalam atau sama. Sehingga ekspansi akan terus dilakukan ke simpul tetangga di kedalaman selanjutnya, layaknya DFS.

Algoritma ini juga memiliki sebuah kelemahan pada algoritma ini. Kelemahannya terdapat pada perhitungan SK yang dilakukan secara linear. Hal ini menyebabkan jika prioritas utama adalah HP (yang merupakan atribut dengan nilai yang pasti jauh lebih besar dari yang lainnya), perhitungan "skor karakter" akan sangat bias ke arah artefak yang menambahkan HP. Oleh karena itu, meskipun hasilnya pasti optimal, optimalnya hanya sebatas optimal "menurut algoritma ini."

Untuk menangani kelemahan algoritma ini, sebelum menghitung SK nilai atribut dapat dinormalisasi dahulu agar skalanya sama di semua atribut. Misalnya, semua atribut dibuat menjadi: $1000 \leq atribut \leq 9999$

Ucapan Terima Kasih

Penulis berterima kasih atas berkat dan rahmat dari Tuhan YME sehingga Penulis dapat menyelesaikan makalah Strategi Algoritma ini. Selain itu, terima kasih juga Penulis ucapkan kepada orang tua Penulis karena selalu memberikan dukungan kepada penulis dan sudah memungkinkan Penulis untuk melanjutkan pendidikan jenjang sekolah tinggi.

Terima kasih sebesar-besarnya juga Penulis ucapkan kepada seluruh dosen pengajar IF2211 Strategi Algoritma Tahun 2020/2021, Prof.Ir. Dwi Hendratmo Widyantoro, M.Sc.,Ph.D., Dr. Ir. Rinaldi, M.T., dan Ir. Rila Mandala, M.Eng.,Ph.D., dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc. Tanpa bantuan dan karya dosen-dosen pengajar, Penulis tidak akan memahami materi Strategi Algoritma.

Terakhir, penulis berterima kasih kepada teman-teman dekat Penulis karena sudah memberikan Penulis dukungan moral untuk menyelesaikan makalah ini dan sudah menjadi teman diskusi Penulis selama penyusunan makalah ini.

Daftar Pustaka

[1] R. Munir, "Penentuan Rute (*Route/Path Planning*): Bagian 1: BFS, DFS, UCS, Greedy Best First Search," Institut Teknologi Bandung, 2021.

[2] R. Munir, "Penentuan Rute (*Route/Path Planning*): Bagian 2: Algoritma A*," Institut Teknologi Bandung, 2021.

[3] "What does 'minmax' mean?" rpg.stackexchange.com. https://rpg.stackexchange.com/questions/64800/what-does-minmax-mean (diakses 9 Mei 2021)

[4] "Attributes." genshin-impact.fandom.org. https://genshin-impact.fandom.com/wiki/Attributes

[5] "Artifacts." genshin-impact.fandom.org. https://genshin-impact.fandom.com/wiki/Artifacts

[6] andrew1234, "Uniform-Cost Search (Dijkstra for large Graphs)." geeksforgeeks.org. https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/

[7] J. Levine, "Uniform Cost Search." youtube.com. https://www.youtube.com/watch?v=dRMvK76xQJI

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.

Tangerang Selatan, 11 Mei 2021

Josep Marcello/13519164