Pemilihan Monster yang Akan Digunakan dalam Permainan Yu-Gi-Oh! Capsule Monster Coliseum

Analisis menggunakan algoritma Greedy untuk memilih monster yang terbaik

Bervianto Leo P - 13514047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514047@std.stei.itb.ac.id

***Abstract*—Permainan Yu-Gi-Oh Capsule Monster merupakan board game yang unik. Pada permainan ini dibutuhkan taktik atau strategi yang baik. Dalam sesi pertandingan akan mempengaruhi hasil kemenangan, namun tidak kalah penting sebelum pertandingan yaitu memilih monster yang cocok dan terbaik untuk melawan musuhnya. Dengan algoritma greedy akan dipilih monster terbaik untuk melawan musuh. Monster akan dipilih menggunakan parameter tertentu sehingga mencari parameter dan batasan yang cocok untuk menghasilkan pemilihan monster yang terbaik.**

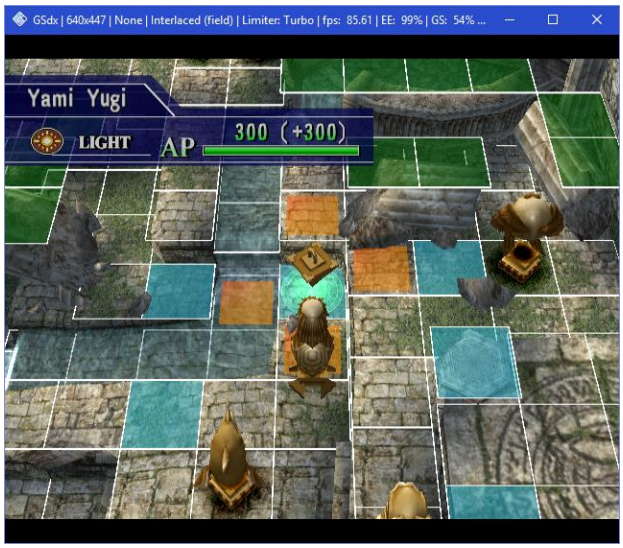
Keywords—algorithm; greedy; strategy; tactics; yu-gi-oh;

I. INTRODUCTION

Dalam permainan Yu-Gi-Oh! Capsule Monster Coliseum berbeda dengan permaianan Yu-Gi-Oh lainnya. Permainan Yu-Gi-Oh pada dasarnya menggunakan beberapa kartu dengan jenis kartu, *monster* (dapat menyerang dan bertahan), *magic/spell* dan juga *trap* "jebakan". Dari berbagai kombinasi kartu tersebut akan menentukan kemenangan dalam permainan Yu-Gi-Oh.

Gambar 1 Sedang dalam permainan Yu-Gi-Oh
Sumber : https://i.ytimg.com/vi/cdIA-GDkd9Y/hqdefault.jpg
diakses pada tanggal 9 Mei 2015

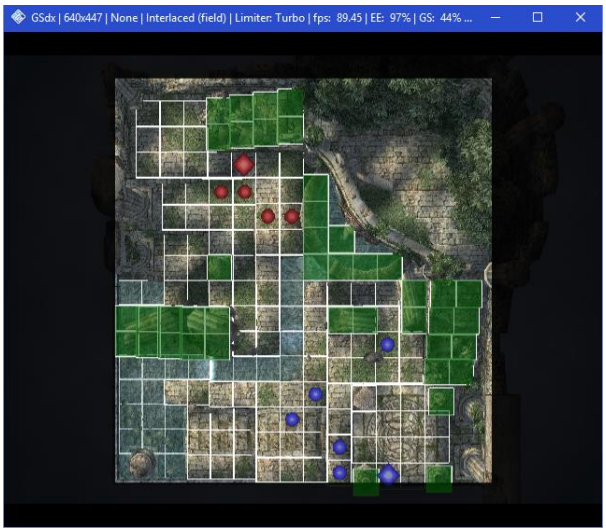
Namun, ada perbedaan dengan permainan Yu-Gi-Oh! Capsule Monster Coliseum yaitu interaksi permainannya. Pada Yu-Gi-Oh! Capsule Monster Coliseum permainan seperti board game berjenis catur. Setiap monster akan memiliki bentuk jalan masing-masing, ada yang miring, lurus, atau 'L' seperti kuda dalam catur dan jumlah gerakan maksimum yang berbeda-beda.



Gambar 2 Lingkup penyerangan dan pergerakan

Sumber : dokumen pribadi

Begitu juga ruang lingkup penyerangan yang berbeda-beda. Berikut ini contoh ruang permainan dan sedang dalam pertandingan.



Gambar 3 Ruang Permainan

Sumber : dokumen pribadi

II. ALGORITMA GREEDY

A. *Prinsip Algoritma Greedy*

Dalam kehidupan sehari-hari, banyak terdapat persoalan yang menuntut pencarian solusi optimum. Persoalan tersebut dinamakan persoalan optimasi (*optimization problems*). Persoalan optimasi adalah persoalan yang tidak hanya mencari sekedar solusi, tetapi mencari solusi terbaik (*best*). Solusi terbaik adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. [1]

Algoritma *greedy* mungkin merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Algoritma ini sederhana dan lempang (*straightforward*). Secara harafiah *greedy* artinya rakus atau tamak, yaitu sifat berkonotasi negatif. Prinsip *greedy* adalah : "*take what you can get now!*". Ambil apa yang dapat anda peroleh sekarang! Prinsip ini juga diadopsi dalam pemecahan masalah optimasi, tetapi tentu dalam konteks positif. [1]

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*). [1]

Dari pendekatan yang disebutkan sebelumnya, dapat mendefinisikan algoritma *greedy* sebagai berikut:

Algorima greedy adalah 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,
2. berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. [1]

B. *Aplikasi Algoritma Greedy*

Dalam kehidupan sehari-hari, hal-hal berikut ini seringkali menggunakan prinsip *greedy*, misalnya:

1. Memilih beberapa jenis investasi (penanaman modal)
2. Mencari jalur tersingkat dari Jakarta ke Bandung
3. Memilih jurusan di Perguruan Tinggi
4. Bermain kartu remi. [1]

C. *Skema Umum Algoritma Greedy*

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

1. Himpunan kandidat, C.

   Himpunan ini berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi S.

   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan bagian dari himpunan kandidat.

3. Fungsi seleksi – dinyatakan dengan predikat SELEKSI – yaitu fungsi 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 (*feasible*) – dinyatakan dengan predikat LAYAK – 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 kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi objektif, yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi.

III. ATURAN DAN CARA BERMAIN YU-GI-OH CAPSULE MONSTER COLISEUM

Pada permainan Yu-Gi-Oh Capsule Monster Coliseum ini memiliki aturan dan cara bermain yang unik. Berikut ini aturan dan cara bermain secara umum yang perlu diketahui dan akan digunakan sebagai parameter yang diamati.

A. Sebelum Pertandingan

Sebelum pertandingan, pemain akan memilih 8 kartu tertutup berisi angka. Pemain dengan angka terbesar akan memilih untuk menyerang pertama atau kedua. Berikut ilustrasi pemilihannya.

Gambar 4 Pemilihan Kartu Angka

Sumber : dokumen pribadi

Jika pemain (*player* 1 atau pemain utama) menang maka dapat memilih untuk menyerang terlebih dahulu atau tidak. Keuntungan dari menyerang terlebih dahulu yaitu akan memasang monsternya terakhir, sedangkan musuhnya akan memasang terlebih dahulu sehingga pemain utama dapat dengan baik memutuskan memasang monster untuk melawan musuhnya. Pemain akan mengetahui elemen-elemen dan monster yang digunakan oleh musuh sehingga dapat memasang monster *counter* untuk mengatasinya.

Gambar 5 Pemilihan attack order

Sumber : dokumen pribadi

Setelah pemilihan *attack order* pemain akan memilih monster yang akan digunakan. Teknik pemilihan akan dijelaskan pada bab atau bagian selanjutnya.

B. Saat Pertandingan

Saat pertandingan akan ada aturan jumlah maksimum gerakan termasuk *spawn* atau menghidupkan monster. Sebelum monster bergerak perlu dilakukan spawn yang akan meningkatkan jumlah maksimum gerakan. Jumlah maksimum gerakan diberikan istilah AP pada permainan ini. Namun untuk memenangkan permainan ini akan sangat bergantung gerakan yang dilakukan dan pemilihan monster. Pemecahan masalah dalam pertandingan tidak akan dibahas dalam tulisan ini dikarenakan masalah yang ada akan cukup kompleks dan sulit diprediksi. Namun pada dasarnya ada 2 teknik yang dapat digunakan yaitu menyerang dan bertahan dengan kedua teknik tersebut dapat diperluas lagi. Penulis biasa menggunakan teknik semi-menyerang dengan memperhatikan pertahanan dengan baik dan akan menyerang saat musuh terperangkap.

Gambar 6 Status AP

Sumber : dokumen pribadi

C. Aturan Dasar

Pada permainan ini memiliki aturan dasar mengenai elemen. Setiap elemen memiliki kelemahan (*disadvantage*) dan kekuatan (*advantage*). Pemilihan monster yang tepat dengan memperhatikan elemen setiap monsternya akan mempengaruhi 40 persen kemenangan. Sedangkan 60 persen kemenangan lainnya akan dipengaruhi taktik saat pertandingan.

Gambar 7 Graf kelemahan dan kekuatan setiap elemen

Sumber : dokumen pribadi

Pada gambar tersebut, setiap yang ditunjuk berarti "lemah dengan". Sebagai contoh *dark* akan kuat melawan *earth* tapi akan lemah melawan *light*.

IV. Analisis Pemilihan Monster Berdasarkan Berbagai Parameter

Monster akan dipilih dengan berbagai kriteria yang akan dibuat oleh penulis dan akan membandingkan dengan kemungkinan kemangan yang akan didapat. Selain itu dalam penentuan monster juga akan bergantung dengan giliran. Namun pada saat ini akan dibahas mengenai pemilihan jika mendapat giliran menyerang pertama atau dengan kata lain mengetahui monster yang digunakan oleh musuh. Sebagai contoh kasus, ditetapkan musuh dengan elemen *light* sejumlah 3, *fire* sejumlah 1, dan *earth* sejumlah 2. Kasus ini diambil dengan melawan satu orang tertentu yang sudah disiapkan oleh permainan. Serta kasus monster yang tersedia merupakan monster yang sudah didapatkan oleh penulis. Berikut ini beberapa kriteria yang digunakan.

A. *Hanya Menggunakan MP*

Pada pemilihan hanya menggunakan MP, akan dipilih monster dengan MP terbesar hingga MP sudah tidak bisa ditambah lagi dan jumlah maksimum monster yang diperbolehkan.

Gambar 8 Pemilihan Monster

Sumber : dokumen pribadi

Setiap monster akan di-*sort*-ing menggunakan parameter MP. Batasan yang akan digunakan yaitu MP yang masih tersedia dan jumlah monster yang diperbolehkan. Algoritma yang digunakan sebagai berikut. MP semakin besar dianggap sebagai monster yang lebih baik.

```
function ChooseMonster (input C : himpunan_monster, N : integer, MP : integer) → himpunan_monster

{ C merupakan himpunan monster yang tersedia, N merupakan jumlah maksimum monster yang diperbolehkan, MP merupakan jumlah MP yang diperbolehkan, fungsi ini akan mengembalikan himpunan monster dengan MP terbesar}

Deklarasi

   S : himpunan_monster

   x : monster

Algoritma

   S ← {}

   while (∑(MP monster di dalam S) ≤ MP) and (C ≠ {}) and (∑(monster di dalam S) ≤ N)) do

      x ← monster dengan MP terbesar dalam C

      C ← C – {x}

   if (∑(MP monster di dalam S) + MP monster x ≤ MP)) then

      S ← S ∪ {x}

   endif

   endwhile

   if (∑(MP monster di dalam S) ≤ MP) or (∑(monster di dalam S) ≤ N)) then

      return S

   endif
```

Dalam algoritma tersebut akan mendapatkan beberapa monster dengan ranking MP terbesar. Namun jika dianalisis lebih dalam. Indikator hanya menggunakan MP ini sedikit kurang efektif untuk melawan musuh. Dikarenakan dimungkinkan elemen yang terpilih menjadi lemah melawan musuh. Selain itu juga jumlah monster yang terpilih dimungkinkan hanya sedikit dibatasi oleh jumlah MP dan MP setiap monster. Sebagai contoh kasus, memiliki batas MP 850 dan maksimum monster 8 seperti gambar 8 sehingga mendapatkan monster dengan detail sebagai berikut.

1. MP 235 - Dark
2. MP 228 – Dark
3. MP 179 – Dark
4. MP 167 – Dark
5. MP 40 – Light

Jika dilihat MP yang didapatkan cukup baik untuk 4 teratas sedangkan selanjutnya sudah tidak mungkinkan dan mendapat yang terlalu jauh rentang MP-nya. Lalu saat melihat elemennya, kemungkinan kalah cukup besar dikarenakan musuh memiliki elemen *light* yang banyak dan tidak memiliki *counter* setelah pemilihan. Monster yang terpilih yaitu 4 pertama akan mudah dikalahkan dengan *light* walaupun MP-nya besar. Dapat disimpulkan parameter hanya MP tidak efektif atau tidak optimum.

B. Menggunakan MP dan Elemen Musuh

Pada bagian ini akan sedikit lebih rumit. Namun tidak terlalu kompleks. Hal pertama yang harus dilakukan yaitu mengkategorikan yang harus diambil dan melakukan algoritma greedy terhadapnya. Sebagai contoh kasus yang telah diberikan. Maka akan dikategorikan yang cocok melawan *light*, *earth*, dan *fire*. Untuk melawan *light* akan perlu *electric*, untuk melawan *earth* perlu *dark* namun perlu berhati-hati dengan *light* musuh lalu untuk *fire* perlu *water*. (lihat graf elemen untuk lebih jelas) Langkah pertama yang perlu diambil yaitu mengambil counter masing-masing minimal jumlahnya sama dengan jumlah milik musuh. Namun perlu dilakukan prioritas pengambilan, yaitu dari yang terbanyak hingga terdikit. Algoritma yang akan dilakukan sebagai berikut.

```
   function     ModifiedChooseMonster(input
C:himpunan_monster,     E:himpunan_monster,
MP:integer) → himpunan_monster
Deklarasi
      S : himpunan_monster
      D : himpunan_monster
      i : integer
      MP1 : integer
Algoritma
      MP1 ← MP;
      for (i=0; i<∑(kategori dalam E); i++)
do
      for (j=0; j<∑(monster yang digunakan
dalam kategori E(i); j++) do
            D ← monster C dalam kategori E
            S     ←     ChooseMonster(C,D,
∑(monster  yang  digunakan  dalam  kategori
E(i),MP1)
      endfor
endfor
return S
```

Dalam algoritma ini memiliki keunggulan untuk melawan musuh dengan baik. Agar monster elemen *dark* untuk tetap bertahan perlu dilindungi oleh monster lain. Algoritma ini akan kelihatan hanya keberuntungan. Namun algoritma ini cukup optimal untuk melawan musuh. Algoritma ini perlu dilengkapi jika masih dimungkinkan pertambahan monster dapat ditambahkan dengan monster yang paling banyak elemennya atau monster yang tidak terpengaruh sama sekali dengan monster musuh. Berikut ini hasil sementara dari algoritma tersebut.

1. MP 150 – Lightning
2. MP 107 – Lightning
3. MP 107 – Lightning
4. MP 235 – Dark
5. MP 102 – Dark
6. MP 139 – Water

Dalam hal ini jumlah MP yang digunakan sudah cukup baik yaitu tersisa 10 MP. Jumlah monster yang digunakan sudah cukup atau imbang. Namun sudah dipastikan perlu melindungi elemen *dark*. Elemen *lightning* juga perlu dilindungi dari elemen *earth*. Dalam hal ini dapat terlihat jika musuh memiliki pasangan elemen yang baik. Elemen yang digunakan musuh yaitu memiliki elemen yang dapat melawan kelemahan musuh. Dapat diringkas bahwa dua parameter ini sudah cukup baik dan cukup optimal. Jika dengan cepat menghabiskan musuh yang elemen terbanyak, akan mempermudah pergerakan monster elemen lain.

C. Menggunakan Parameter Lebih Banyak dan Kompleks

Dapat ditambahkan parameter lain untuk mendapatkan hasil semakin lebih baik. Misalkan dengan paremeter dominasi tempat akan cukup membantu kekuatan monster yang akan digunakan. Hal lain yang dapat ditambahkan yaitu seperti menggunakan level monster dan AP maksimum yang dapat digunakan. AP dapat diefektifkan agar tidak terjadi pemborosan. AP digunakan untuk pergerakan dan *summon*. Saat AP sudah mencapai maksimum namun masih ada yang bisa di-*spawn* hanya akan melakukan pemborosan AP. Agar lebih efektif dapat dihitung AP yang akan digunakan dan dihasilkan. Namun perlu diperhatikan akan semakin sulit menentukan monster yang harus digunakan dan belum tentu efektif untuk melawan musuh.

V. SIMPULAN

Pemilihan monster dapat menggunakan algoritma *greedy* parameter yang disuplai untuk cukup optimal menggunakan parameter MP dan elemen musuh. Dibutuhkan parameter yang simpel dan tidak begitu banyak untuk menentukan monster lebih mudah dan optimum. Agar lebih optimal dapat menggunakan parameter lebih banyak namun akan semakin sulit memilih monster yang akan digunakan. Semakin sedikit parameter yang digunakan akan semakin mudah dipilih namun semakin kurang baik untuk melawan musuh. Namun semakin banyak parameter belum menentukan akan sangat optimal. Perlu strategi khusus saat pertandingan untuk melawan. Sebelum pertandingan atau pemilihan monster akan memberikan kemenangan sebesar 40 persen yang cukup besar mempengaruhi hasil kemenangan. Selain itu jika mendapat giliran menyerang kedua, dapat menggunakan taktik yaitu memiliki monster yang dapat melawan kelemahan monster tersebut.

ACKNOWLEDGMENT

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa untuk segala karunia-Nya sehingga penulis dapat menyelesaikan makalah ini. Terima kasih kepada dosen pengampu IF2211 Strategi Algoritma, Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi untuk pengajaran yang telah diberikan dalam kuliah Strategi Algoritma secara khusus mengenai Algoritma *Greedy* yang menjadi dasar dalam penulisan makalah ini.

REFERENCES

[1] Munir, Rinaldi. "Diktat Kuliah IF2211 Strategi Algoritma". Bandung: Teknik Informatika ITB, 2007.

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, 9 Desember 2015

Bervianto Leo P - 13514047