Penentuan Langkah Sederhana dalam Permainan Kartu Hearthstone dengan Algoritma Greedy

Muhammad Umar Fariz Tumbuan - 13515050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515050@std.stei.itb.ac.id

***Abstract*—Permainan *Hearthstone* adalah sebuah permainan kartu yang dikelola oleh pengembang *game* Blizzard. Dalam permainan *Hearthstone*, ada banyak kemungkinan langkah yang dapat dilakukan dalam suatu pertandingan. Akan dilakukan percobaan untuk membuat suatu algoritma penentu langkah optimal menggunakan konsep algoritma *greedy*.**

Keywords—Hearthstone,algoritma, greedy, card game;

I. PENDAHULUAN

Hearthstone adalah sebuah permainan kartu yang bersifat *free-to-*play, dikembangkan oleh Blizzard Entertainment. *Hearthstone* dirilis di seluruh dunia pada 11 Maret 2014, dan sampai saat penulisan ini masih memiliki popularitas yang tinggi di kalangan banyak pemain, baik yang kasual ataupun yang *hardcore*. Saat rilis, hearthstone hanya tersedia pada *platform* Microsoft Windows dan macOS. Kemudian pada akhir tahun 2014, dukungan untuk *platform* iOS dan Android diadakan. Hal ini menjadi salah satu faktor peningkatan popularitasnya.

Sudah menjadi hal umum dalam sebuah *game* untuk memiliki sebuah intelejensi buatan atau bot. Tetapi dalam sebuah permainan kartu seperti *Hearthstone* pengembangan sebuah bot dapat menjadi sebuah hal yang kompleks. Hal ini dikarenakan langkah yang diambil tiap bot perlu mempertimbangkan banyak hal. Beberapa diantaranya adalah bagaimana mengatur penggunaan *mana*, penentuan menyerang *minion* atau langsung ke *hero* lawan, penggunaan *spell* yang efisien, pelaksanaan strategi terkait *deck* yang digunakan, dan lain-lain. Karena itu penulis ingin melakukan langkah pertama dalam pengembangan bot *Hearthstone* dengan batasan pengelolaan mana dan penentuan langkah yang bersifat umum untuk kebanyakan *deck*. Hal yang tidak dibahas dalam makalah ini adalah penyusunan *deck* dan strategi yang bersifat lanjut.

II. DASAR TEORI

A. Algoritma Greedy

Algoritma *Greedy* merupakan salah algoritma yang umum digunakan dalam masalah optimasi. Algoritma ini akan mencari lnagkah yang terlihat sebagai langkah yang paling optimal tanpa mempertimbangkan langkah setelahnya. Pengambilan solusi optimum lokal ini diharapkan dapat membawa kita ke solusi yang mendekati optimum global. Tetapi solusi akhir yang diambil oleh algoritma *greedy* cenderung tidak msolusi yang terbaik tetapi *sub-optimum*. Hal ini dikarenakan :

1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode *exhaustive search*).
2. Terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optimal.

Karena sifatnya yang cenderung menghasilkan solusi *sub-optimum*, algoritma *greedy* dipakai saat hanya diperlukan aproksimasi dalam penyelesaian masalah. Karena algoritma *greedy* langsung menentukan sebuah langkah, maka algoritma ini cenderung lebih cepat saat dijalankan. Tingkat kerumitannya pun cenderung lebih rendah dibandingkan algoritma lain.

Dalam pengaplikasiannya, algoritma *greedy* memiliki beberapa komponen yang perlu ditentukan sebelumnya:

1. Himpunan kandidat (C)

   Himpunan kandidat adalah himpunan solusi yang elemennya berisi bagian solusi yang dapat diambil.

2. Himpunan solusi (S)

   Himpunan solusi adalah solusi yang diambil dari himpunan kandidat berdasarkan criteria yang sudah ditentukan. Himpunan solusi merupakan himpunan bagian dari Himpunan kandidat.

3. Fungsi seleksi

   Fungsi Seleksi adalah fungsi yang menentukan solusi yang bisa kita ambil dari himpunan kandidat sebagai calon optimum lokal.

4. Fungsi layak

   Fungsi layak menentukan apakah solusi yang diambil sejauh ini masih memenuhi batasan yang ditentukan.

5. Fungsi obyektif

   Fungsi obyektif adalah fungsi yang membantu memaksimumkan atau meminimumkan solusi yang diambil.

B. Aturan Permainan Hearthstone

Dalam permainan *Hearthstone*, dua orang pemain memainkan salah satu dari 9 kelas yang ada dengan sebuah *deck* berisikan 30 kartu. Tiap kelas memiliki sebuah kekuatan unik dan beberapa kartu yang hanya dimiliki kelas tersebut. Aksi pemain dibatasi dengan *mana* yang dimiliki saat gilirannya. Pada awal permainan, tiap pemain hanya memilki 1 *mana.* Pada awal sebuah ronde baru, tiap pemain akan mendapatkan tambahan satu *mana,* sampai nilai maksimumnya yaitu sepuluh. Pemain dapat memainkan kartu di tangannya selama memilki *mana* yang mencukupi. Pada ronde pertama, pemain mengambil tiga kartu dari *deck* masing-masing. Kemudian pemain dapat memilih kartu yang ingin dikembalikan ke *deck* untuk mengambil ulang. Pemain yang mendapat giliran kedua mendapatkan tambahan satu kartu dan sebuah koin yang dapat menggantikan satu *mana.* Kedua pemain memulai permainan dengan 30 *health point* dan berusaha menghabiskan *health point* lawan terlebih dahulu untuk memenangkan pertandingan. Dalam satu waktu pemain hanya dapat memilki 10 kartu di tangan. Kartu yang diambil pada saat tangan pemain penuh akan hangus. Jika kartu di *deck* pemain habis, pemain akan mengalami *fatigue*, di mana pemain akan kehilangan *health point* sebanyak 1 x jumlah ronde pemain kehabisan kartu.

Gambar 1 Papan permainan

Pada saat penulisan makalah, terdapat 2 jenis kartu, yaitu *minion, spell. Minion* adalah kartu yang dapat di-*summon* atau dimainkan ke papan permainan sebagai bawahan pemain. Tiap *minion* memiliki *health point, attack point,* dan *mana cost*. Sebuah *minion* dapat menyerang *minion* lawan atau ke lawan langsung dan mengurangi *health point* keduanya sebesar *attack point* yang dimilikinya. *Minion* akan mati ketika *health pointnya* mencapai nilai nol. Ketika sebuah *minion* menyerang *minion* lain, keduanya akan kehilangan *health point* sesuai dengan *attack point* yang dimiliki lawannya. Saat pertama dimainkan, *minion* harus menunggu satu giliran sebelum bisa menyerang. Dalam satu waktu, seorang pemain hanya dapat memiliki tujuh *minion* di atas papan permainan.

Beberapa minion bisa memilki beberapa kemampuan yang dapat digunakan. Sebuah *minion* dapat memilki lebih dari satu kemampuan. Jenis-jenis kemampuan yang ada dalam permainan pada masa penulisan makalah ini adalah sebagai b erikut.

1. Kemampuan pasif – *Minion* bisa memilki kemampuan yang selalu aktif selama dia berada di atas papan permainan atau sesuai deskripsi yang dimilki *minion*.
2. *Battlecry* – Kemampuan yang diaktifkan pada saat *minion* dimainkan ke papan. Kemampuan tidak diaktifkan jika *minion* di-*summon* ke papan tidak melalui tangan pemain.
3. *Taunt* – Sifat yang dimilki *minion*, yaitu *minion* lawan harus mengalahkan *minion* yang memilki *taunt* sebelum bisa menarget *minion* tanpa taunt atau *hero* pemain.
4. *Stealth* – Sifat yang dimilki *minion,* yaitu *minion* tidak bisa ditarget lawan selama *stealth* masih aktif. *Minion* dengan *stealth* masih bisa diserang dengan serangan *area of effect* dan tidak bisa mengaktifkan sifat *taunt*-nya.
5. *Poisonous* – Sifat yang dimilliki *minion,* yaitu *minion* yang diserang akan hancur meskipun masih memiliki *health point* yang tersisa.
6. *Charge* – Sifat yang dimiliki *minion,* yaitu *minion* bisa langsung menyerang setelah di-*summon* ke papan permainan.
7. *Freeze* – *Minion* yang terserang oleh *freeze* tidak dapat menyerang pada ronde berikutnya.
8. *Immunity* – Karakter yang memiliki sifat *immunity* tidak bisa diserang dengan cara apapun.
9. *Divine shield* – *Minion* memilki sebuah pelindung yang melindunginya dari satu buah serangan bernilai apapun.
10. *Draw cards* – Pemain dapat mengambil kartu dari *deck* jika memenuhi syarat tertentu saat *minion* dimainkan.
11. *Deathrattle* – Kemampuan ini diaktifkan pada saat *minion* mati.
12. *Inspire* – Kemampuan ini diaktifkan pada saat pemain mengaktifkan kekuatan *hero*-nya selama *minion* yang memilki *inspire* berada di atas papan permainan.
13. *Discover* – Pemain dapat memilih satu dari tiga kartu dari luar *deck*. Ketiga kartu tersebut dipilih secara acak oleh permainan.
14. *Adapt* – Pemain dapat memilih satu dari tiga pilihan *buff* yang dapat diberikan ke *minion* yang memilki sifat *adapt*. Terdapat 10 jenis *buff* yang dapat dipilih.

Gambar 2 Kartu *Minion* dan Kartu *Spell*

Jenis kartu kedua adalah kartu *spell*. Kartu *spell* adalah kartu yang memilki efek yang langsung aktif pada saat dimainkan. Sama seperti kartu *minion,* Efek tersebut dapat berupa *damage*, *buff,* atau efek yang serupa dengan kemampuan *minion* seperti *draw cards* dan *discover*. Sub-jenis kartu spell adalah kartu *trap* dan *quest*. Kartu *trap* adalah adalah spell yang hanya bisa aktif pada saat lawan melakukan suatu aksi. Kartu *trap* tidak bisa diaktif pada saat giliran yang memainkan kartu *trap* tersebut. Jenis lainnya adalah kartu *quest*. Kartu *quest* adalah kartu yang aktif setelah pemain memainkkannya dan memenuhi syarat yang dimilki kartu tersebut. Setelah pemain memenuhi syarat, pemain akan mendapatkan kartu yang memilki efek unik dan dahsyat. Kartu *quest* selalu muncul pada saat pengambilan tiga kartu di awal permainan.

III. IMPLEMENTASI

Dalam menerapkan algoritma *greedy* jenis langkah yang mungkin akan diberikan nilai. Strategi yang akan dilakukan bot adalah mengalahkan minion musuh dengan spell dan menyerang *hero* lawan menggunakan *minion*. Penulis berencana menerapkan algoritma per-aksi karena terdapat beberapa kemampuan kartu yang dapat menambah kartu baru ke tangan sehingga berpotensi mengubah optimum lokal setelah aksi dilakukan. Akibatnya dalam satu giliran, algoritma bisa dijalankan lebih dari satu kali Urutan aksi dari yang bernilai tertinggi ke yang terendah adalah sebagai berikut.

1. Menggunakan kartu *spell* atau kekuatan *hero* jika terdapat karakter lawan yanag dapat dikalahkan, *minion* atau *hero*. *Hero* akan diprioritaskan terlebih dahulu.
2. Memainkan minion dengan total *health point* dan *attack point* tertinggi. Efek yang bersifat menyerang lawan ditujukan ke *minion* dengan *attack point* tertinggi atau *hero* lawan jika tidak ada *minion* lawan di papan.
3. Menyerang *hero* lawan menggunakan *minion* di papan kecuali terdapat *minion* dengan *taunt*. *Minion taunt* dengan *health point* terbesar yang akan ditarget terlebih dahulu. *Minion* dengan *attack point* tertinggi yang akan menyerang terlebih dahulu.
4. Menggunakan *spell* atau kekuatan *hero* untuk menyerang *minion* dengan *attack point* tertinggi atau *hero* lawan dengan mana yang tersisa.
5. Mneggunakan *spell* apapun yang tersisa di tangan selama mana masih mencukupi.
6. Mengakhiri giliran jika sudah tidak ada kartu yang bisa dipakai.

Langkah selanjutnya adalah menentukan kriteria kelima komponen dari algoritma *greedy*. Berikut kriteria yang penulis buat.

1. Himpunan kandidat (C)

   Himpunan kandidat yang dimilki bot adalah kartu yang berada di tangan beserta target yang bisa dipilih (jika memungkinkan).

2. Himpunan solusi (S)

   Himpunan solusi pada kasus ini adalah aksi yang bisa dilakukan dengan batasan *mana* dan kartu yang dimilki. Himpunan solusi tiap aplikasi dibatasi sebanyak satu elemen.

3. Fungsi seleksi

   Memilih aksi dengan nilai prioritas tertinggi.

4. Fungsi layak

   Aksi yang akan dilakukan memenuhi batasan *mana* yang dimilki.

5. Fungsi obyektif

   Aksi yang dilakukan menggunakan semaksimum mungkin mana yang dimilki.

Langkah berikutnya adalah mendesain dan membuat fungsi dan prosedur yang digunakan. Berikut fungsi dan prosedur yang penulis buat dan penjelasannya.

```
function himpun_aksi_yang_mungkin(input
kartu_di_tangan: larik_kartu, input sisa_mana: int) -->
aksi_yang_mungkin
    Deklarasi
        i      :    int
        j      :    int
        C      :    aksi_yang_mungkin
    Algoritma
        i <-- 0
        while (i < jumlah_kartu_di_tangan) do
            if (harga_kartu_ke_i <= sisa_mana) then
                if (kartu_perlu_target) then
                    if (ada_target) then
                        j <-- 0
                        while (j <
jumlah_target_yang_mungkin) do
                                C <-- aksi_dengan_kartu_ke_i
                else
                    C <-- aksi_dengan_kartu_ke_i
        --> C
```

Fungsi himpun_aksi_yang_mungkin akan mengembalikan sebuah himpunan aksi yang bisa dilakukan dengan mempertimbangkan kartu yang ada di tangan dan jumlah *mana* yang dimilki pada giliran tersebut.

```
procedure Seleksi(input x : elemen_aksi_yang_mungkin,
output S : aksi_yang_diambil)
    Deklarasi

    Algoritma
        if (nilai_elemen_C > nilai_S) then
        S <-- elemen_C
```

Prosedur Seleksi akan menerima input sebuah aksi dan membandingkan nilainya dengan nilai aksi yang pada solusi saat ini. Prosedur kemudian mengubah nilai solusi yang dijadikan input.

```
procedure pilih_aksi(input C: aksi_yang_mungkin, output
      sisa_mana : int, output S: aksi_yang_diambil)
    Deklarasi
        i   : int

    Algoritma
        if (C <> {}) then
            S <-- elemen_C_pertama
            i <-- 1
            while (i < jumlah_elemen_C) do
                    Seleksi(elemen_C_ke_i, S)
                    i <-- i + 1
           sisa_mana <-- mana_yang_digunakan_S
```

Prosedur plilih_aksi menerima input himpunan aksi yang mungkin. Kemudian prosedur ini akan mengubah nilai solusi yang akan dilkaukan. Akhirnya nilai sisa *mana* saat ini akan dikurangi sebesar harga aksi yang dilakukan.

```
procedure jalankan_giliran(input/output sisa_mana: int,
input/output kartu_di_tangan : himpunan_kartu,
input/output solusi)
    Deklarasi

    Algoritma
        solusi <-- akhiri_giliran
  pilih_aksi(himpun_aksi_yang_mungkin(kartu_di_tangan,
sisa_mana), sisa_mana, solusi)

        kartu_di_tangan <-- kartu_di_tangan - solusi
```

Prosedur jalankan_giliran akan menggunakan semua fungsi dan prosedur yang telah dibuat sebelumnya. Fungsi ini akan menerima jumlah *mana* maksimal yang dimiliki pada awal ronde dan larik kartu yang ada di tangan.

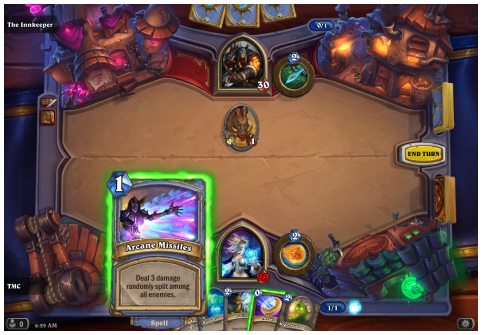
```
procedure mainkan_game()
    Deklarasi
        mana : int
        game_over : boolean
        kartu_di_tangan : himpunan_kartu
        solusi : aksi_yang_diambil

    Algoritma
        mana <-- 0
        game_over <-- false

        ambil_kartu_awal(kartu_di_tangan)
        while (not game_over) do
            if (mana < 10) then
                mana <-- mana + 1

            ambil_kartu(kartu_di_tangan)

            while
(himpun_aksi_yang_mungkin(kartu_di_tangan, mana) !=
{}) do
                solusi <-- jalankan_giliran(mana,
kartu_di_tangan)
                lakukan(solusi)

            if (hp_lawan <= 0 or hp_self <= 0) then
            game_over <-- true
```

Prosedur mainkan_game akan menyatukan fungsi dan prosedur sebelumnya dengan alur permainan. Terdapat beberapa prosedur yang tidak dibuat penulis karena tidak berkaitan langsung dengan pengambilan keputusan.

IV. Analisis

A. *Penggunaan Algoritma Dalam Pertandingan*

Berikut satu game di mana algoritma di atas digunakan. Tidak semua aksi dalam pertandingan ditampilkan, tetapi hanya beberapa untuk memberikan gambaran aksi. Kelas yang digunakan adalah kelas *Mage* yang memilki kekuatan untuk melakukan 1 *damage* ke karakter lawan dengan harga 2 *mana.*



Gambar 3 Ronde 1

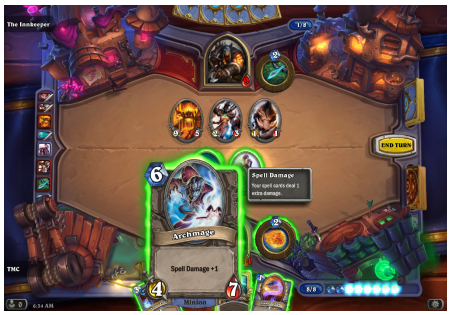
Pada pertandingan ini, pemain mendapatkan giliran kedua. Musuh mengeluarkan sebuah minion dengan *health point* bernilai 1. Terdapat dua aksi yang mungkin, yaitu menggunakan kartu *spell* yang bisa mengahncurkan minion dan menggunakan koin. Kedua aksi tersebut dilakukan dan pemain menghancurkan *minion* lawan. Dalam kasus ini, penggunaan koin bukanlah aksi yang diharapkan.

Gambar 4 Ronde 2

Pada ronde kedua, terdapat pilihan menggunakan *spell* yang melukai *hero* lawan sebanyak 3 poin dan mensummon sebuah *minion*. Aksi men-*summon* minion dilakukan. Hal serupa dilakukan untuk 2 ronde berikutnya karena musuh tidak memainkan apapun.

Gambar 5 Ronde 5

Pada ronde kelima, terdapat dua aksi prioritas tertinggi yang dapat dilakukan, yaitu menggunakan spell untuk menghancurkan karakter lawan. Tetapi dengan *pseudo code* yang digunakan, spell terakhir yang akan digunakan. Penggunaan *spell* ini tidak efektif karena *damage* dan harga yang besar digunakan untuk menghancurkan *minion* yang lemah, sedangkan terdapat *spell* lain yang lebih murah dan *damage* yang cukup.



Gambar 6 Ronde 8

Pada ronde kedelepan, terdapat dua pilihan aksi dengan prioritas tinggi, yaitu men-*summon* minion dan menggunakan kekuatan *hero* untuk menghancukan *minion* dengan satu *health point*. Karena masih terdapat sisa mana, *minion minion* pada gambar di-*summon*. Menggunakan algoritma yang telah dibuat, *minion* pemain akan mengambaikan *minion* lawan dan mengincar *hero* lawan. Aksi tersebut memenangkan pertandingan ini.

B. *Analisis Algoritma*

Algoritma yang dibuat berhasil memenangkan sebuah pertandingan walaupun dengan melakukan beberapa aksi yang tidak optimum. Tetapi hal itu menunjukkan bahwa algoritma yang dibuat merupakan satu langkah mendekati algoritma yang menghasilkan hasil optimum.

Meskipun dalam percobaan ini pertandingan berhasil dimenangkan, menurut penulis masih banyak kekurangan yang dimiliki algoritma yang telah dibuat. Pertama adalah dalam hal pemilihan *spell yang digunakan*. Dalam pertandingan di atas terdapat sebuah kasus di mana pemain menggunakan *spell* yang terlalu kuat untuk menghancurkan sebuah *minion* dengan *health point* yang kecil. Padahal terdapat pilihan lain yang lebih murah dan memilki *damage*

yang cukup. Menurut penulis, hal ini bisa diperbaiki dengan menyimpan data *attack* dan *health minion* musuh, kemudian pemnyimpan data *spell* yang menyebabkan *damage* di tangan. Tiap *minion* musuh akan dibandingkan dengan *damage spell* di tangan. *Spell* dengan nilai *damage* lebih besar dari *health* minion dan dengan selisih *damage* dan *health* terkecil yang akan digunakan. Hal ini dilakukan dengan harapan *spell* yang akhirnya digunakan adalah *spell* yang terbaik dengan target yang terbaik pula.

Kekurangan kedua adalah algoritma belum mempertimbangkan kemampuan yang dimiliki *minion* dan *spell* selain *damage*. Hal yang dipertimbangkan dalam memainkan *minion* hanyalah jumlah status dan *spell* seperti *trap* hamper tidak akan digunakan.

Kekurangan lainnya adalah ketidakmampuan algoritma untuk melakukan kombo. Algoritma yang dibuat masih terbatas hanya memainkan *minion* dan menggunakan *spell damage*. Padahal terdapat berbagai sinergi yang dimilki di antara kartu-kartu yang ada dalam *Hearthstone.*

V. KESIMPULAN

Algoritma *greedy* bisa digunakan untuk menentukan langkah dalam permainan *Hearthstone*. Tetapi langkah yang diambil tidak selalu menuju ke langkah yang optimum. Masih banyak perbaikan dan pengembangan yang bisa dilakukan untuk mendapatkan hasil yang lebih optimum. Walau begitu, algoritma *greedy* kurang cocok untuk *deck* yang bergantung pada kombo yang rumit.

Ucapan Terima Kasih

Saya bersyukur kepada Allah SWT karena dengan petunjuk dan rahmat dari Allah lah saya bisa menyelesaikan makalah ini. Rasa terima kasih juga saya tujukan kepada Dr. Nur Ulfa Maulidevi, S.T., M.Sc., Dr. Ir. Rinaldi Munir, M.T., dan Dr. Masayu Leylia Khodra, S.T., M.T. selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma yang telah membantu saya memahami banyak hal.

Referensi

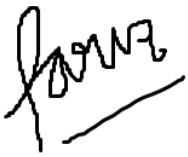
[1] Munir, Rinaldi, "Diktat Kuliah IF2211 Strategi Algoritma". Bandung: Teknik Informatika, ITB, 2017.

PERNYATAAN

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

Bandung, 19 Mei 2017

Ttd



Muhammad Umar Fariz Tumbuan / 13515050