Penerapan Program Dinamis dalam Proses Pathfinding pada Permainan Civilization V

Rubiano Adityas / 13510041
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510041@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas tentang program dinamis serta penerapannya dalam sebuah permainan *desktop* yaitu Civilization V. Program dinamis sendiri seringkali digunakan untuk menyelesaikan persoalan *pathfinding*, dengan menemukan alur penelusuran dari titik awal menuju titik akhir yang memiliki total bobot minimum. Civilization merupakan salah satu seri permainan *turn-based strategy* yang ternama, sementara Civilization V merupakan versi terbarunya. Pada permainan Civilization V pemain memimpin sebuah peradaban untuk melewati transisi-transisi zaman, sembari mengelola sumber daya dan unit yang dimiliki. Unit bergerak di dunia yang dimodelkan dengan petak-petak heksagonal, dimana bobot penjelajahan petak berbeda-beda, bergantung pada kondisi petak dan unit. Penerapan program dinamis dapat menentukan jalur paling optimal dari suatu petak ke petak lain bagi suatu unit.**

Kata Kunci* — program dinamis, Civilization V, *turn-based strategy*, *video game

I. PENDAHULUAN

Dengan semakin berkembangnya teknologi informasi yang tersedia bagi publik dan pertumbuhan jumlah pengguna teknologi informasi, pangsa pasar *video game* juga semakin bertambah. Banyak sekali macam inovasi genre permainan saat ini, antara lain adalah puzzle, *action*, *arcade*, strategi, dan juga subgenre dari tiap-tiap genre yang ada. Perkembangan industri ponsel membuat banyak sekali orang yang saat ini memiliki *smartphone*, karena harganya yang terjangkau. Karena banyaknya pengguna *smartphone*, banyak *startup* perusahaan pengembang permainan yang bersifat *mobile*, dimainkan pada *smartphone*. Pengembang-pengembang permainan yang sebelumnya berfokus di platform *desktop* dan *console* banyak yang beralih atau membuat divisi baru untuk pengembangan permainan *mobile*. Hal ini dikarenakan pangsa pasar yang lebih luas, maka keuntungan yang diperoleh juga lebih banyak.

Salah satu indikasi beralihnya selera masyarakat ke permainan *mobile* adalah selain banyaknya startup perusahaan pengembang *mobile*, banyak perusahaan pengembang permainan *desktop* dan *console* yang mengalami kebangkrutan, sehingga harus gulung tikar atau *merger* dengan perusahaan pengembang lain. Walau pangsa pasar untuk permainan *mobile* saat ini lebih menggiurkan, tidak semua pengembang permainan mengalihkan fokusnya pada permainan *mobile*. Dari semua perusahaan yang masih setia pada platform *desktop*, salah satunya adalah Firaxis. Firaxis merupakan pengembang permainan *desktop* yang cukup tua usianya, dan menghasilkan produk-produk dan seri permainan yang bertahan cukup lama, salah satunya adalah seri permainan Civilization.

Civilization merupakan sebuah permainan *desktop*, dimana pemain berperan sebagai pemimpin peradaban yang mengelola sumber daya dan pergerakan unit. Unit bergerak di dunia yang terbagi menjadi petak-petak, dimana bentuk petaknya berbeda tergantung seri keberapa permainan Civilization tersebut. Pada seri terbaru, Civilization V, petak dunianya berbentuk heksagonal, dengan bobot penelusuran yang berbeda-beda tergantung kondisi petaknya. Dengan menggunakan program dinamis, kita bisa menganalisa jalur mana yang paling efisien bagi sebuah unit untuk ditelusuri.

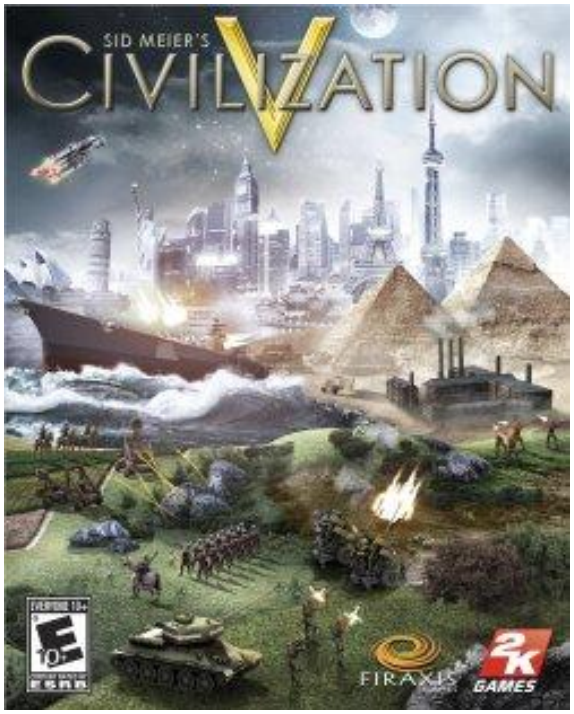
II. PROGRAM DINAMIS DAN DESKRIPSI PERMAINAN

A. Program Dinamis

Program dinamis adalah sebuah metode pemecahan persoalan dengan cara mengurai solusi persoalan menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) Dengan pembagian tersebut, diharapkan solusi akhir persoalan yang didapat akan berupa sekumpulan solusi tiap langkah/tahapan yang saling terkait.

Tahap awal dari analisis program dinamis adalah mencari sejumlah berhingga pilihan yang mungkin. Pilihan-pilihan tersebut kemudian diolah menjadi solusi tahap, dimana solusi setiap tahap dibangun dari hasil solusi tahap sebelumnya. Solusi yang dicari adalah solusi yang optimal, dan proses optimisasi harus dilakukan di tiap tahap pencarian solusi.

B. Civilization V



Gambar 1. Cover art Civilization V

Civilization V adalah sebuah permainan *desktop* bergenre strategi untuk platform Windows dan Mac OS X uang dikembangkan oleh Firaxis Games, dan dipublikasikan oleh 2K Games & Aspyr. Dalam Civilization V, pemain berperan sebagai seorang pemimpin peradaban, dimana sang pemimpin dan peradaban yang dipimpinnya didasarkan pada peradaban asli dunia. Tiap peradaban memiliki keunggulan dan kekurangan yang unik, yang menentukan cara memainkan peradaban tersebut dengan efektif. Sebagai pemimpin, sang pemain harus bisa membawa peradabannya melewati transisi zaman, bersaing dan berdiplomasi dengan peradaban pemain lain, serta memenuhi salah satu dari berberapa kondisi spesifik yang diperlukan untuk memenangkan sebuah sesi permainan.

Civilization V menerapkan sistem *turn-based*, yakni semua pemain (baik itu manusia maupun komputer) menjalankan gilirannya secara bergantian. Pemain pertama melaksanakan gilirannya, diikuti pemain kedua hingga pemain terakhir, setelah itu kembali giliran pemain pertama lagi dan seterusnya. Dalam setiap gilirannya, banyak sumber daya peradaban yang bisa dikelola oleh pemain, antara lain adalah uang, makanan, produksi, budaya, sains, dan unit.

Sebuah sesi permainan Civilization dilakukan pada sebuah pemodelan dunia yang dalam kondisi normal digenerasi secara acak, artinya dunia yang digunakan akan berbeda untuk tiap sesi. Namun pemain bisa memilih untuk menggunakan sebuah model dunia nyata (benua tertentu, atau seluruh dunia) apabila menginginkannya. Dunia tersebut dibagi menjadi petak-petak heksagonal yang sama ukurannya, dimana jumlahnya akan menyesuaikan dengan ukuran dunia yang dibuat.

Gambar 2. Dunia permainan dilihat dengan mode Normal View

Gambar 3. Dunia permainan dilihat dengan mode Strategic View

Tiap-tiap petak bisa merepresentasikan kondisi geologis yang berbeda, misalnya hutan, rawa, sungai, padang pasir, laut, padang rumput, dan sebagainya. Kondisi dari suatu petak menentukan atribut petak yang memberi dampak bagi unit, salah satunya adalah bobot penelusuran petak. Setiap giliran, sebuah unit memiliki hak untuk bergerak dari satu petak ke petak lain. Seberapa jauh unit bisa bergerak dalam satu giliran, bergantung dari bobot jalur, dan sebuah atribut unit yang bernama *movement point*. *Movement point* bernilai integer positif, akan dikurangi dengan bobot petak tempat unit berpindah, dan akan kembali ke nilai semua ketika giliran berikutnya. Tiap unit memiliki movement poin yang independen, dan bisa berbeda dengan unit lainnya.

Selain atribut *movement point*, faktor lain yang mempengaruhi pergerakan adalah jenis petak dan jenis unitnya. Jenis unit pasukan laut bisa bergerak bebas di petak laut, sementara sipil dan pasukan darat bisa bergerak bebas di petak darat. Pada umumnya

unit pasukan laut memiliki *movement point* yang lebih tinggi dibanding pasukan darat dan sipil.

Tabel 1. Nama dan Bobot Petak

Nama Petak	Jenis Petak	Bobot Petak
Hill	Darat	2
Marsh	Darat	2
Jungle	Darat	2
Forest	Darat	2
Grassland	Darat	1
Plains	Darat	1
Desert	Darat	1
Tundra	Darat	1
Ocean	Laut	1
Coast	Laut	1

Aturan pergerakan unit dari satu petak ke petak lain adalah sebagai berikut:

a) Dua buah unit bertipe sama (sipil/pasukan darat/pasukan laut) tidak bisa menempati satu petak yang sama.

b) Unit sipil dan pasukan darat bisa bergerak ke petak laut apabila sudah memiliki teknologi yang diperlukan, namun unit pasukan laut tidak bisa bergerak ke petak darat.

c) Ketika sebuah unit bergerak menuju petak yang bersebelahan, *movement point* unit tersebut dikurangi dengan bobot petak tujuannya.

d) Ketika sebuah unit ingin bergerak ke petak yang bersebelahan yang bobotnya melebihi *movement point* yang tersisa (tidak kosong), unit tersebut diperbolehkan untuk pindah, dan *movement point* untuk giliran tersebut dikosongkan.

e) Ketika sebuah unit ingin bergerak menuju petak yang bersebelahan yang dibatasi oleh sungai (garis warna biru pada Gambar 3), *movement point* unit untuk giliran tersebut dikosongkan. Kondisi petak tujuan diabaikan.

f) Satu-satunya saat dimana unit pasukan laut bisa menempati petak darat adalah apabila sebuah kota dibangun di daratan sebelah laut, maka unit pasukan laut mampu menempati petak darat/kota tersebut.

Kegunaan unit pasukan secara umum adalah untuk merebut dan menjaga kota, yang merupakan aset utama peradaban. Kota merupakan tempat penghasil sumber daya (uang, makanan, produksi, budaya, dan sains) dan tempat memproduksi unit. Kota dibangun oleh sebuah unit sipil bernama Settler, sementara unit sipil Worker berguna untuk meningkatkan nilai guna dari petak-petak sekitar kota untuk dieksploitasi. Tiap-tiap sumber daya memiliki kegunaan yang berbeda.

Gambar 4. HUD yang menampilkan sumber daya, dan teknologi yang sedang diteliti

a) Uang : Biaya perawatan unit dan bangunan tiap giliran, membeli bangungan, aset untuk berdiplomasi.

b) Makanan : Meningkatkan laju pertumbuhan penduduk, yang menentukan penghasilan kota.

c) Produksi : Merepresentasikan kecepatan produksi unit dan bangunan.

d) Budaya : Perluasan wilayah peradaban, dan pengambilan kebijakan publik.

e) Sains : Meneliti teknologi, untuk memberi akses produksi unit dan bangunan yang lebih modern.

Semua sumber daya yang dimiliki harus dimanfaatkan oleh pemain untuk mencapai berberapa kondisi menang. Pemain bisa mengakumulasikan banyak budaya hingga berhasil mengimplementasikan kebijakan publik dengan jumlah tertentu, mendapatkan cultural victory, atau akumulasi sains untuk scientific victory. Kondisi menang yang lain adalah diplomatic victory, dimana pemain berhasil memenangkan pemungutan suara untuk menjadi pemimpin PBB, dan domination victory, dimana pemain berhasil menghancurkan semua peradaban lain. Peradaban apa yang kita pilih, cara kita bermain, dan seberapa banyak sumber daya yang kita miliki, menjadi faktor pertimbangan dalam skenario kemenangan mana yang ideal untuk dituju pemain.

III. DESKRIPSI MASALAH

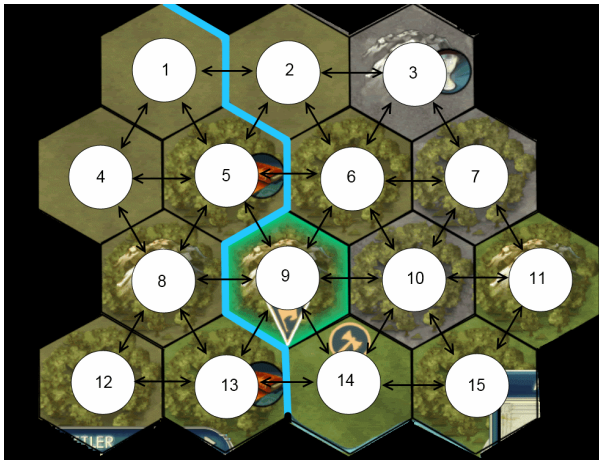
Dalam proses penelusuran petak, diketahui bahwa bobot penelusuran petak tidaklah konstan. Berdasarkan aturan pergerakan, sebuah unit yang memiliki *movement point* tersisa (tidak kosong) lebih sedikit dari bobot petak tujuan, masih bisa untuk bergerak ke petak tersebut. Dalam kasus tersebut,

bisa dianggap bahwa bobot dari petak menyesuaikan (menurun) terhadap *movement point* sisa si unit. Sama halnya dengan dua petak yang dipisahkan oleh sungai, bobot untuk menyeberanginya bia dianggap sama dengan *movement point* sisa yang dimiliki unit sebelum menyeberang.

Adanya dinamika bobot seperti ini, membuat perlu adanya metode penampungan bobot yang bervariasi pada program dinamis agar bisa menangani kedua macam kasus tersebut. Bobot yang digunakan akan disesuaikan dengan *movement point* yang sedang dimiliki unit. Harapannya, dengan pendekatan program dinamis, kita bisa menentukan jalur teroptimisasi bagi unit yang bergerak dari satu petak ke petak lainnya.

IV. IMPLEMENTASI PROGRAM DINAMIS

Kita tinjau kembali Gambar 3 pada bab sebelumnya, dan dari potongan peta lokasi permainan tersebut, dibuat pemodelan grafnya.



Gambar 4. Pemodelan graf terhadap Gambar 3

Untuk detilnya, simpul {1, 2, 4} adalah *plains*, simpul {3, 8, 9, 11} adalah *hill*, simpul {5, 6, 7, 10, 12, 13, 15} adalah *forest*, dan simpul 14 adalah *grassland*. Untuk node yang memiliki 2 kondisi (*hill* dan *forest*) diambil bobot kondisi yang paling tinggi, yaitu dua, dan dianggap sebagai *hill*. Sisi yang menyebrangi sungai adalah sisi yang menghubungkan simpul 1 dan 2, 2 dan 5, 5 dan 6, 5 dan 9, 8 dan 9, 9 dan 13, serta 13 dan 14.

Untuk memodelkan semua kemungkinan bobot diperlukan sebuah tabel yang menampung matriks keterhubungan, dimana tiap indeks tabel merepresentasikan bobot graf ketika sisa *movement point* unit bernilai sesuai dengan indeks. Pemodelan seperti ini dilakukan karena sisa *movement point* sebuah unit menentukan bobon petak yang akan dikunjungi, seperti pada penjelasan bab deskripsi masalah.

Dibuat pseudo code sebuah fungsi bernama createWeightArray, yang berfungsi untuk menghasilkan array sekumpulan matrik keterhubungan. createWeightArray akan menerima masukan berupa total *movement point* dari unit yang akan berjelajah, dan graf awal kondisi maksimum. Kondisi maksimum yang dimaksud adalah asumsi unit bisa menelusuri sisi berbobot tak hingga, sehingga penyeberangan sungai bisa terlebih dahulu diinisialisasi dengan nilai tak hingga. Nilai penyeberangan sungai yang sesungguhnya baru akan dihitung oleh fungsi. Sebagai ganti nilai tak hingga, sisi yang tak valid akan diberi nilai -1, dan akan diganti menjadi tak hingga oleh fungsi.

-1 ∞ -1 1 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
∞ -1 2 -1 ∞ 2 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 2 2 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1 -1 2 -1 -1 2 -1 -1 -1 -1 -1 -1 -1
1 ∞ -1 1 -1 ∞ -1 2 ∞ -1 -1 -1 -1 -1 -1
-1 1 2 -1 ∞ -1 2 -1 2 2 -1 -1 -1 -1 -1
-1 -1 2 -1 -1 2 -1 -1 -1 2 2 -1 -1 -1 -1
-1 -1 -1 1 2 -1 -1 -1 ∞ -1 -1 2 2 -1 -1
-1 -1 -1 -1 ∞ 2 -1 ∞ -1 2 -1 -1 ∞ 1 -1
-1 -1 -1 -1 -1 2 2 -1 2 -1 2 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 2 -1 -1 2 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 2 -1 -1 -1 -1 2 -1 -1
-1 -1 -1 -1 -1 -1 -1 2 ∞ -1 -1 2 -1 ∞ -1
-1 -1 -1 -1 -1 -1 -1 -1 2 2 -1 -1 ∞ -1 2
-1 -1 -1 -1 -1 -1 -1 -1 -1 2 2 -1 -1 1 -1

Gambar 5. Matriks representasi graf awal

Berikut adalah pseudo code fungsi createWeightArray()

```
function createWeightArray
(input mp : integer, G : Graf) :
array3D of integer

// mp adalah total movement point
unit

// G adalah Graf awal hasil
penghitungan

bobot : array3D

bobot <-
  integer[mp][nrow Graf][ncol Graf]

for (int i = mp; i <= 1; i--)

   bobot[mp] <- Graf

   foreach element in output[mp]

      if element > i

         elemen di indeks tsb <- i

      else if element = -1
```

```
            elemen di indeks tsb <-
               tak-hingga
        endif
    endfor
endfor
```

Misal sebuah unit ingin menjelajahi peta dari simpul 3 menuju simpul 12, dan unit tersebut memiliki total *movement point* 3. Maka dengan fungsi createWeightArray(), akan dihasilkan array berisikan 3 buah matriks keterhubungan, masing-masing merepresentasikan bobot penelusuran petak ketika *movement point* si unit tersisa 1, 2, dan 3.

∞	3	∞	1	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	2	∞	3	2	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	1	∞	∞	∞	2	2	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	2	∞	∞	2	∞	∞	∞	∞	∞	∞	∞
1	3	∞	1	∞	3	∞	2	3	∞	∞	∞	∞	∞	∞
∞	1	2	∞	3	∞	2	∞	2	2	∞	∞	∞	∞	∞
∞	∞	2	∞	∞	2	∞	∞	∞	2	2	∞	∞	∞	∞
∞	∞	∞	1	2	∞	∞	∞	3	∞	∞	2	2	∞	∞
∞	∞	∞	∞	3	2	∞	3	∞	2	∞	∞	3	1	∞
∞	∞	∞	∞	∞	2	2	∞	2	∞	2	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	2	∞	∞	2	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	2	∞	∞
∞	∞	∞	∞	∞	∞	∞	2	3	∞	∞	2	∞	3	∞
∞	∞	∞	∞	∞	∞	∞	∞	2	2	∞	∞	3	∞	2
∞	∞	∞	∞	∞	∞	∞	∞	∞	2	2	∞	∞	1	∞

Gambar 6. Matriks keterhubungan dengan sisa *movement point* 3

∞	2	∞	1	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	2	∞	2	2	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	1	∞	∞	∞	2	2	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	2	∞	∞	2	∞	∞	∞	∞	∞	∞	∞
1	2	∞	1	∞	2	∞	2	2	∞	∞	∞	∞	∞	∞
∞	1	2	∞	2	∞	2	∞	2	2	∞	∞	∞	∞	∞
∞	∞	2	∞	∞	2	∞	∞	∞	2	2	∞	∞	∞	∞
∞	∞	∞	1	2	∞	∞	∞	2	∞	∞	2	2	∞	∞
∞	∞	∞	∞	2	2	∞	2	∞	2	∞	∞	2	1	∞
∞	∞	∞	∞	∞	2	2	∞	2	∞	2	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	2	∞	∞	2	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	2	∞	∞
∞	∞	∞	∞	∞	∞	∞	2	2	∞	∞	2	∞	2	∞
∞	∞	∞	∞	∞	∞	∞	∞	2	2	∞	∞	2	∞	2
∞	∞	∞	∞	∞	∞	∞	∞	∞	2	2	∞	∞	1	∞

Gambar 7. Matriks keterhubungan dengan sisa *movement point* 2

∞	1	∞	1	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	1	∞	1	1	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	1	∞	∞	∞	1	1	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	1	∞	∞	1	∞	∞	∞	∞	∞	∞	∞
1	1	∞	1	∞	1	∞	1	1	∞	∞	∞	∞	∞	∞
∞	1	1	∞	1	∞	1	∞	1	1	∞	∞	∞	∞	∞
∞	∞	1	∞	∞	1	∞	∞	∞	1	1	∞	∞	∞	∞
∞	∞	∞	1	1	∞	∞	∞	1	∞	∞	1	1	∞	∞
∞	∞	∞	∞	1	1	∞	1	∞	1	∞	∞	1	1	∞
∞	∞	∞	∞	∞	1	1	∞	1	∞	1	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	1	∞	∞	1	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	1	∞	∞
∞	∞	∞	∞	∞	∞	∞	1	1	∞	∞	1	∞	1	∞
∞	∞	∞	∞	∞	∞	∞	∞	1	1	∞	∞	1	∞	1
∞	∞	∞	∞	∞	∞	∞	∞	∞	1	1	∞	∞	1	∞

Gambar 8. Matriks keterhubungan dengan sisa *movement point* 1

Sesudah didapat array matriks keterhubungan, analisis solusi dengan program dinamis.

```
currentmp : integer
currentmp <- mp
programdinamis {
// basis
        f(i, ∅) = bobot[currentmp]_{i,1}
// rekurens
f(i, S) = min{bobot[currentmp]_{i,j} + f(j, S − {j})}
if currentmp = 1
   currentmp <- mp
else
   currentmp--
}
```

Penghitungan bisa dilakukan secara manual, dengan catatan, bobot yang diambil dari matriks harus sesuai dengan *movement point* unit yang tersisa. *Movement point* harus selalu direduksi untuk tiap rekurens, hingga mencapai nilai 1, dimana nilainya kemudian akan dikembalikan ke nilai total sebelumnya. Dengan metode ini, kita bisa mensimulasikan bobot sisi graf yang dinamis.

Setelah dikalkulasi, jalur dari simpul 3 ke simpul 12 yang paling optimal (bobot minimum) untuk unit dengan *movement point* 3 adalah 3, 2, 5, 8, 12. Bobot totalnya adalah 6. Penghematan bobot terjadi ketika perpindahan dari simpul 2 ke simpul 5 (melewati sungai menghabiskan semua *movement point* yang tersisa, dan yang tersisa 2), dan dari simpul 8 ke

simpul 12 (bobot petak tujuan 2, sementara *movement point* sisa tinggal 1). Jumlah giliran pemain yang dihabiskan untuk memindahkan unit juga bisa dihitung dengan melihat berapa kali *movement point* dikembalikan ke nilai awal, jumlah giliran akan selalu lebih satu dari jumlah pengembalian nilai. Pengembalian nilai *movement point* menjadi 3 terjadi satu kali ketika *movement point* unit habis di simpul 5, maka dari itu pemain memerlukan 2 giliran untuk memindahkan unit tersebut daru simpul 3 ke simpul 12.

V. KESIMPULAN

Program dinamis dapat digunakan untuk menentukan jalur optimal sebuah unit dalam permainan Civilization V, bila diberikan graf kondisi awal dan total *movement point* dari unit yang bersangkutan. Karena *movement point* berkaitan erat dengan giliran pemain, algoritma bisa juga digunakan untuk optimisasi jumlah giliran, selain optimisasi bobot.

REFERENSI

[1] R. Munir,*Strategi Algoritma*, Teknik Informatika, Bandung, 2009.

[2] http://www.civilization5.com/
Tanggal akses: 21 Desember 2012, pukul 08:15 WIB

[3] http://forum.rpg.net/
Tanggal akses: 21 Desember 2012, pukul 09.00 WIB

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, 21 Desember 2012

Rubiano Adityas
13510041