PENERAPAN ALGORITMA GREEDY PADA TACTICAL ROLE-PLAYING GAME

Franciscus Borgias Dian Paskalis - 13507048

Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganeca 10, Bandung
e-mail: dianpaskalis@yahoo.com

ABSTRAK

***Tactical Role-Playing Game* (TRPG) adalah permainan yang dimainkan pada konsol tertentu (komputer, *Playstation*, dan sebagainya) dan memiliki unsur strategi berbasis giliran (*turn-based*). Beberapa contoh dari TRPG adalah Disgaea, Final Fantasy Tactics, Jeanne D'Arc, dan sebagainya. TRPG merupakan bagian dari *genre* RPG (Role-Playing Game). Pada hampir semua RPG selalu terdapat beberapa macam karakter seperti karakter utama, teman, netral, dan lawan. Biasanya karakter utama, teman, dan lawan memiliki jurus-jurus tertentu yang dapat digunakan. Pada layar permainan jenis TRPG biasanya terdapat beberapa karakter pada saat yang sama dan pemain dapat menggunakan taktik tertentu seperti menggunakan serangan biasa, jurus, benda-benda, maupun memanfaatkan medan permainan untuk dapat mengalahkan lawannya. Unsur taktik makin diperjelas dengan adanya *turn-based* pada TRPG, artinya pemain dapat menggerakkan karakternya sesuka hati saat giliran pemain tersebut selama masih memenuhi ketentuan permainan. Elemen lain yang biasanya ada adalah benda-benda yang dapat digunakan dan medan permainan yang mungkim dapat mepengaruhi jalur pergerakan karakter.**

Algoritma *greedy* adalah algoritma yang selalu mengambil solusi lokal terbaik sambil mencoba mencari penyelesaian akhir. Algoritma *greedy* dapat menemukan solusi akhir terbaik untuk beberapa masalah optimasi, tetapi dapat juga menemukan solusi yang kurang optimal untuk beberapa masalah lainnya. Algoritma *greedy* ini dapat dipakai pada saat suatu karakter akan menentukan lawan mana yang akan dilawannya pada saat tertentu. Penentuan solusi algoritma *greedy* pada jenis permainan ini biasanya bergantung pada jarak dan parameter karakter.

Kata kunci: algoritma, *greedy*, *tactical*, *role*, *playing*, *game*.

1. PENDAHULUAN

Tactical Role-Playing Game (TRPG) adalah permainan yang dimainkan pada konsol tertentu dan memiliki unsur strategi berbasis giliran (*turn-based*). Pada layar permainan jenis TRPG biasanya terdapat beberapa karakter pada saat yang sama dan pemain dapat menggunakan taktik seperti menggunakan serangan biasa, jurus, benda-benda, maupun memanfaatkan medan permainan untuk mengalahkan lawannya.

Gambar 1. Salah satu contoh TRPG, seri *Disgaea*.

Urutan tahap pada satu giliran pemain biasanya adalah penempatan karakter-karakter, pemilihan aksi karakter, dan dilanjutkan eksekusi aksi. Pada kebanyakan TPRG, setelah eksekusi aksi dilakukan pemain hanya dapat menggerakkan karakternya ke tempat tertentu selama masih memenuhi ketentuan permainan. Lawan pemain (dalam hal ini biasanya adalah A.I) tentu saja juga bergerak dengan mematuhi aturan-aturan permainan ini menggunakan algoritma yang ada. Pada beberapa TPRG, pemain dapat memilih mode pergerakan karakter secara otomatis dengan menggunakan algoritma yang sudah ditentukan dari permainan, pada keadaan seperti ini pemain sudah menyerahkan pergerakan karakternya pada komputer sehingga pemain hanya tinggal melihat perkembangan permainan saja.

Gambar 2. Peletakkan karakter sebelum melakukan aksi.

Pada kebanyakan permainan TRPG, algoritma pergerakan otomatis suatu karakter hanya dengan mencari karakter lawan yang posisinya terdekat untuk dilawan. Jika hanya menggunakan hal ini, tak jarang langkah yang diambil pada suatu giliran tidak efektif.

Misalnya karakter kita memiliki kekuatan serang 5 dan bisa berjalan sejauh 7 langkah, kemudian ada karakter musuh A dengan jarak 3 dari kita memiliki HP (*Hit Point*/ tenaga) sebanyak 1 dan B yang HP-nya 100 berjarak 2 dari kita. Dengan menggunakan algoritma mencari lawan terdekat, otomatis algoritma akan memilih B sebagai lawan yang akan dihadapi karakter kita, padahal untuk mengalahkan B, karakter kita perlu 20 giliran (100/5=20). Dalam 20 giliran tersebut, bukan tidak mungkin jika seandainya A ikut mengeroyok kita bersama B. Seandainya lawan yang dipilih adalah A, karakter kita dapat langsung mencapai dan mengalahkannya pada giliran tersebut sehingga langkah yang diambil dapat lebih efektif dalam menyelesaikan permainan.

Gambar 3. Mengalahkan lawan dalam 1 giliran akan lebih efektif dalam menyelesaikan permainan.

Gambar 4. Pemilihan langkah yang kurang tepat dapat menyebabkan karakter dikeroyok lawan dan mempersulit permainan.

Hal di atas itulah yang menurut penulis sebenarnya masih dapat ditingkatkan lagi agar permainan berjalan lebih efektif. Peningkatan algoritma untuk intelegensia buatan tersebut dapat dicapai salah satunya menggunakan algoritma *greedy*.

2. ALGORITMA GREEDY

Algoritma *greedy* adalah algoritma yang selalu mengambil solusi lokal terbaik sambil mencoba mencari penyelesaian akhir.

2.1 Gambaran Umum Algoritma *Greedy*

Ada banyak macam penerapan algoritma *greedy*, tetapi umumnya algoritma ini selalu:

- Mengambil apa yang dapat diambil pada saat itu juga tanpa melihat konsekuensi ke depannya
- Memilih sebuah solusi optimal lokal (bagian) dengan harapan proses kelanjutannya akan mencapai solusi optimal global (keseluruhan)

Persoalan optimasi algoritma *greedy* biasanya disusun oleh elemen-elemen berikut :

1. Himpunan kandidat, C.
   Himpunan ini berisi elemen-elemen pembentuk solusi. Pada tiap langkah, satu buah kandidat diambil dari himpunannya.
2. Himpunan solusi, S.
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi adalah himpunan bagian dari himpunan kandidat.
3. Fungsi seleksi – dinyatakan dengan predikat SELEKSI – yaitu fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan untuk mencapai solusi optimal. Kandidat yang sudah pernah terpilih pada suatu langkah tidak akan

pernah dipertimbangkan lagi pada langkah selanjutnya. Biasanya pada tiap kandidat, *x*, diberikan suatu nilai numerik dan fungsi seleksi memilih *x* yang memiliki nilai terbesar atau memilih *x* yang memiliki nilai terkecil.

4. Fungsi kelayakan – dinyatakan dengan predikat LAYAK – yaitu fungsi yang memeriksa apakah kandidat yang akan dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif, yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain, persoalan optimasi yang diselesaikan dengan algoritma *greedy* melibatkan pencarian himpunan bagian S dari himpunan solusi S dengan menggunakan fungsi seleksi dan kandidat tersebut harus memenuhi kriteria pada fungsi kelayakan. Kandidat-kandidat pada himpunan solusi S kemudian akan dioptimasi oleh fungsi obyektif agar dapat menghasilkan solusi (lokal) yang diinginkan.

2.2 Penerapan Algoritma *Greedy*

Dalam penerapannya, dapat digunakan paling tidak tiga algoritma *greedy*. Pertama, algoritma *greedy* digunakan saat pemilihan jalur terpendek untuk tiap lawan. Kedua, algoritma digunakan untuk memilih lawan yang pasti dapat dikalahkan dalam giliran tersebut. Ketiga, algoritma *greedy* digunakan untuk memilih lawan yang paling dekat dengan karakter yang akan digerakkan.

Secara garis besar, alur kerja algoritma adalah seperti ini:

1. Mencari jalur terpendek antara karakter dengan tiap lawan menggunakan optimasi *greedy*.
2. Dari tiap-tiap lawan dipilih 1 yang dapat dikalahkan oleh karakter tertentu pada giliran tersebut dan posisinya paling dekat. Jika algoritma ini tidak dapat dipenuhi, maka akan berlanjut ke algoritma 3.
3. Memilih lawan yang posisinya paling dekat dengan karakter tersebut.

Elemen-elemen algoritma *greedy* yang digunakan untuk algoritma 1 adalah sebagai berikut :

1. Himpunan kandidat : area permainan yang dapat dilewati karakter.
2. Himpunan solusi : semua jalur yang menghubungkan karakter dengan 1 lawan tertentu.
3. Fungsi seleksi : menggunakan algoritma pencarian jejak, bisa menggunakan A*, *backtracing*, dan sebagainya.
4. Fungsi kelayakan : memeriksa apakah seluruh daerah kemungkinan pergerakan karakter sudah diperiksa.
5. Fungsi obyektif : memilih 1 jalur yang paling pendek.

Elemen-elemen yang digunakan untuk algoritma 2 adalah :

1. Himpunan kandidat : antrian jalur terpendek untuk tiap lawan.
2. Himpunan solusi : semua lawan yang dapat dikalahkan pada giliran tersebut.
3. Fungsi seleksi : memilih lawan yang dapat dikalahkan pada giliran tersebut. Pemilihan dilakukan dengan menggunakan perhitungan status karakter dan lawan
4. Fungsi kelayakan : memeriksa apakah lawan dapat dikalahkan pada giliran tersebut dan meyakinkan semua rute telah diperiksa.
5. Fungsi obyektif : memilih 1 lawan terdekat yang dapat dikalahkan pada giliran tersebut.

Sedangkan elemen-elemen algoritma *greedy* yang digunakan untuk algoritma 3 adalah sebagai berikut :

1. Himpunan kandidat : antrian jalur terpendek untuk tiap lawan.
2. Himpunan solusi : tidak diperlukan.
3. Fungsi seleksi : tidak diperlukan.
4. Fungsi kelayakan : tidak diperlukan.
5. Fungsi obyektif : memilih 1 lawan yang posisinya paling dekat dengan karakter tersebut.

Berikut adalah gambaran umum *pseudo code* yang digunakan :

```
procedure enemyMinRoutes (input M :
map, A : chara, enemies : list of
chara, output R : list of route)
{program menerima masukan berupa
matriks map, karakter yang akan
digerakkan, dan list musuh}

Deklarasi
enemy := chara;
lrute := list of route;
rute := route;

Algoritma
enemy := first(enemies);
while (isMember(enemy,enemies)) do
{fungsi kelayakan}
  lrute := getRoutes(enemy,A,M);
  {fungsi seleksi}
```

```
  rute := shortest(lrute);
  {fungsi obyektif}
  insert(rute,R);
  enemy := next(enemies);
-------------------------------------

procedure killEnemy (input R : list of
route, A : chara, output E : route)
{program menerima masukan berupa list
jalur terpendek musuh dan karakter yang
akan digerakkan lalu mengeluarkan rute
musuh terdekat yang dapat dibunuh pada
giliran tersebut}

Deklarasi
enemy := chara;
lrute := list of route;
trute := list of route;
rute := route;
krute := route;

Algoritma
lrute = R;
rute := rfirst(lrute);
while (isMember(rute,lrute)) do {fungsi
kelayakan}
  krute := kill(A,getChara(rute));
  {fungsi seleksi}
  insert(krute,trute);
  rute := rnext(lrute);
E := shortest(trute));
{fungsi obyektif}
-------------------------------------

procedure sEnemy (input R : list of
route, output E : route)
{program menerima masukan berupa list
jalur terpendek ke musuh dan
mengeluarkan rute ke musuh terdekat}

Deklarasi
lrute := list of route;

Algoritma
lrute = R;
E := shortest(lrute));
{fungsi obyektif}
```

3. KESIMPULAN

Berdasar uraian di atas, dapat disimpulkan bahwa :

- Algoritma *greedy* berguna untuk mencari solusi optimal lokal dengan harapan dapat membantu tercapainya solusi optimal global.
- Penggunaan algoritma *greedy* dalam *Tactical Role-Playing Game* dapat meningkatkan efektifitas langkah yang diambil. Dengan efektifitas yang meningkat tentu akan meningkatkan tantangan dan kemudahan bagi pemain permainan tersebut. Penggunaan algoritma yang dijabarkan diatas tentu berakibat pada waktu proses (waktu tunggu bagi pemain) dan penggunaan memori. Tetapi meskipun demikian, penulis beranggapan bahwa hal tersebut tidak terlalu mempengaruhi *user experience* karena biasanya dalam TRPG tidak terdapat lebih dari 30 karakter sekaligus dalam peta permainan sehingga proses tidak akan terlalu berat.
- Penggabungan beberapa algoritma *greedy* di atas menunjukkan bahwa dapat digunakannya berbagai macam algoritma dalam merancang intelegensia buatan yang baik. Dari algoritma yang sudah ditulis di atas juga sebenarnya masih dapat dilakukan pengembangan-pengembangan sesuai kebutuhan dan keinginan pembuat permainan..

REFERENSI

[1] Munir, Rinaldi, "Diktat kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika ITB, 2009.

[2] Definition of RPG, http://dictionary.babylon.com/RPG, diakses pada tanggal 3 Januari 2010.

[3] Greedy Algorithm, http://www.itl.nist.gov/div897/sqg/dads/HTML/greedyalgo.html, diakses pada tanggal 3 Januari 2010.