Perancangan Algoritma Greedy pada AI Permainan Turn Based Strategy

Benedikus Holyson Tjuatja - 13510101
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510101@std.stei.itb.ac.id

***Abstrak*— Seiring berjalannya dengan era automatisasi, video games merupakan permaian yang sangat populer pada zaman ini. Turn based strategy merupakan suatu tema permainan penyusunan strategi pertempuran yang digemari oleh para pemain video games. Dalam bermain video games, tidak lengkap rasanya tanda adanya suatu musuh yang dikendalikan oleh AI. Tingkat kesulitan suatu permainan juga tak lepas dari bagaimana cara AI menyusun strategi dan melakukan pergerakan. Dalam makalah ini akan dibahas bagaimana struktur algoritma greedy dapat menentukan strategi yang akan digunakan oleh AI.**

***Kata Kunci*— Turn Based Strategy, Greedy Algorithm, Artificial Inteligent (AI), Strategy making, greedy algorithm in game.**

I. PENDAHULUAN

Turn Based Strategy (TBS) adalah suatu permainan yang bergenre strategi dimana para pemainnya bermain secara bergiliran atau bergantian. Banyak permainan yang bergenrekan turn based strategy ini, mulai dari games klasik hingga video games. Game klasik seperti catur, ludo, monopoli, dsb, bisa dikatakan bergenre turn based strategy, karena seorang pemain harus menunggu pemain lainnya bergerak hingga giliran dia kembali. Untuk permainan game video seperti Civilization, Heroes of Might and Magic, Brigandine, Final Fantasy Tactics, dsb juga merupakan permainan yang bertipekan turn based strategy.

Pada dasarnya permainan berjeniskan turn based strategy mengambil tema mengenai peperangan, baik secara historikal maupun fantasi. Tujuan dari jenis inipun biasanya sama yaitu menguasai seluruh daerah yang ada, atau mengalahkan semua lawan hingga pemain merupakan pemegang kekuasaan yang mutlak. Video games yang menggunakan jenis permainan turn based strategy ini tercatat pertama kali digunakan oleh permainan Empire yang didevelop oleh Walter Bright pada tahun 1977. Meskipun video games tentang genre ini telah ada sejak tahun 1977, namun kepopuleran permainan dengan genre ini mulai muncul pada tahun 1990an dimana era dari permainan video games berada pada tahap popularitasnya.

Gambar 1.1 Catur dapat dikategorikan permainan TBS

Permainan yang berjenis turn based stratey ini biasanya memberikan para pemainnya sejumlah unit atau pasukan yang dapat digunakan untuk mengalahkan musuh. Unit-unit ini biasanya memiliki suatu ciri khas yang berbeda beda, dan pemain diharapkan dapat menyusun suatu strategi dengan unit-unit tersebut agar dapat mengalahkan pemain lainnya. Pemain biasanya dibatasi oleh suatu aturan yang menandakan selesainya giliran pemain tersebut, dan pemain harus menunggu pemain lainnya untuk melakukan gilirannya hingga gilirannya selesai. Aturan-aturan tersebut dapat berupa berbagai macam, missalkan pada permainan catur, pemain hanya dapat menggerakkan 1 buah gerakan unit, lalu giliran pemain tersebut selesai. Namun pada permainan video games saat ini selesainya giliran suatu pemain biasanya ditandai dengan telah selesainya bergerak semua unit yang dimiliki, namun banyak juga permainan yang memiliki aturannya sendiri. Ketika gilirannya, sang pemain bebas menggerakkan unit yang dimilikinya dan disinilah strategi dari pemain diterapkan. Untuk batasan selesainya suatu permaian biasanya dilihat dari pemain yang berhasil mengalahkan semua unit lawannya, atau jika permainan berdasarkan waktu, maka dapat dilihat dari perbandingan jumlah unit tersisa.

Gambar 1.2 Contoh permainan TBS pada Video games

Pada permainan video games yang bergenre turn based strategy, tingkat kesulitan dari permainan biasanya ditentukan oleh strategi yang digunakan oleh AI (Artificial Intelegent) yang menjadi musuh pemain. Pada tingkat kesulitan mudah, AI dari permainan ini biasanya menggunakan strategi yang sederhana dan mudah ditebak pergerakkannya, semakin tinggi tingkat kesulitan permainan semakin rumit pula strategi yang digunaka oleh AI tersebut.

Penyusunan strategi yang akan digunakan oleh AI pada games ini menjadi tantangan bagi para games designer untuk membuat suatu struktur algoritma yang dapat membuat AI tersebut lebih terlihat hidup layaknya bermain dengan pemain lain. Makalah ini akan mencoba untuk membahas bagaimana cara membangun struktur algoritma strategi yang akan digunakan oleh AI berdasarkan pendekatan algoritma greedy.

II. DASAR TEORI DAN METODE

Algoritma greedy merupakan suatu bentuk metode heuturistik dalam penyelesaian masalah untuk mencari suatu nilai optimum. Metode greedy ini populer dalam mencari nilai optimum lokal suatu masalah baik itu mencari nilai minimum atau maksimum. Biasanya harapan dari penggunaan metode greedy ini adalah dengan menemukan nilai optimum lokal dari suatu tahap, kita akan menemukan nilai optimum global dari masalah tersebut, namun tak jarang metode ini gagal dalam menemukan nilai optimum global dari permasalahan yang ada. Prinsip dari algoritma greedy sendiri adalah “Take what you can get now”, sehingga nilai optimum yang akan dipiih oleh algoritma greedy merupakan nilai optimum yang terjadi pada saat itu tanpa mempertimbangkan langkah selanjutnya.

Dalam memilih optimasi pada algoritma greedy dibutuhkan beberapa elemen-element yaitu:

1. Himpunan Kandidat (C)
   Himpunan yang berisikan kemungkinan elemen/kandidat yang dapat menjadi solusi dari permasalahan yang ada.
2. Himpunan Solusi (S)
   Berisikan himpunan dari kandidat-kandidat yang terpilih sebagai solusi dari persoalan. Himpunan ini merupakan himpunan bagian dari himpunan kandidat
3. Fungsi Seleksi / Predikat Seleksi
   Fungsi yang digunakan untuk memilih kandidat-kandidat yang paling mungkin untuk mendapatkan solusi optimum dari masalah yang ada.
4. Fungsi Kelayakan (Feasible)
   Fungsi ini digunakan untuk memeriksa apakah kandidat yang telah dipiih dapat melakukan solusi secara layak dan tidak melanggar aturan-aturan yang ada.
5. Fungsi Obyektif
   Fungsi ini digunakan untuk memilih nilai minimum atau nilai maksimum dari suat himpunan solusi.

Algoritma greedy ini membentuk suatu solusi berdasarkan langkah perlangkah. Seperti yang telah dibahas sebelumnya, setiap solusi optimum yang diambil dari suatu tahap dilakukan tanpa memperhatikan konsekuensi kedepannya, oleh karena itu solusi yang dipilih belum tentu merupakan suatu solusi optimum dari masalah tersebut secara umum. Algoritma greedy bekerja dengan cara melakukan pencarian himpunan solusi dari himpunan kandidat yang ada dengan menggunakan fungsi seleksi dan kemudian hasil dari seleksi tersebut akan diverifikasi dengan menggunakan fungsi kelayakan. Hasil optimum yang akan dipilih dilihat berdasarkan solusi yang memberikan jawab yang paling optimal dengan menerapkan fungsi obyektif. Dari hal diatas, pemilihan solusi yang dihasilkan oleh algoritma greedy dapat disimpulkan disebabkan oleh 2 faktor berikut :

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada
2. Pemilihan fungsi seleksi untuk mencapai solusi optimum biasanya didasarkan pada fungsi obyektif (fungsi seleksi dan obyektif bisa saja identik). Jika fungsi seleksi tidak identik, maka kita harus memilih fungsi yang tepat dalam menghasilkan nilai optimum.

Dalam makalah ini algoritma greedy digunakan untuk menentukan strategi yang akan digunakan oleh AI pada permaian turn based strategi. Struktur data dalam penentuan himpunan kandidat, himpunan solusi,fungsi seleksi, fungsi kelayakan dan fungsi onjektif dapat mencerminkan tingkat kesusahan atau kepintaran AI dalam menyusun strategi. Strategi yang digunakan diharapkan dapat memberikan solusi optimum yang dipilih AI dalam memenangi suatu permainan, meskipun

terdapat faktor-faktor lain yang dapat mempengaruhi jalannya permainan.

III. PENERAPAN ALGORITMA

Seperti yang telah dijelaskan sebelumnya, banyak sekali permainan yang bergenre turn based strategy ini bertemakan hal-hal seputar peperangan, oleh karena itu pada makalah ini akan dibahas strategi-strategi peperangan yang seperti apa saja yang akan digunakan oleh AI permainan peperangan dalam mengalahkan lawan-lawannya. Untuk algoritma struktur yang digunakan akan dicoba dijelaskan bagaimana cara pendekatan dengan menggunakan algoritma greedy dalam membangun tingkat kesulitan atau kepintaran AI dalam menyusun strategi peperangan. Dalam suatu giliran permainan peperangan, seorang pemain biasanya dapat melakukan kira-kira 2 kali pergerakan untuk setiap unitnya, gerakan itu terbagi atas gerakan (posisi dari unit) dan aksi (unit tersebut menyerang, bertahan, mengeluarkan sihir, atau menunggu). Pendekatan algoritma greedy yang akan digunakan dapat kita bagi menjadi beberapa bagian :

- Greedy by Movement
- Greedy by Damage
- Greedy by Health

A. Greedy by Movement

Permainan atau peperangan antar pemain biasanya dilakukan dalam suatu peta atau wilayah yang dibatasi. Wilayah peperangan tersebut kemudian dibagi-bagi menjadi blok-blok yang tiap bloknya hanya dapat ditempati oleh satu unit saja. Bentuk bloknya sendiri bermacam-macam bisa berupa kotak/persegi, segilima, segienam, ataupun segidelapan sama sisi, namun yang umumnya digunakan adalah blok yang berbentuk persegi. Setiap unit biasanya hanya dapat menyerang unit yang berada pada tetangga pertama dari blok yang ditempati unit tersebut, namun tidak tertutup kemungkinan jika unit tersebut memiliki suatu kemampuan khusus yang dapat membuat unit tersebut menyerang unit yang berada lebih jauh dari jarak 2 tetangga maupun lebih. Ketika jarak serang unit itu hanya 1 tetangga yang berada disebelahnya, maka dapat dikatakan bahwa suatu unit hanya dapat diserang oleh unit musuh sejumlah n sisi dari balok tersebut. Pada awal mula permainan posisi unit-unit antara 2 pemain biasanya diletakkan lebih jauh diantara pemain lainnya, dan untuk mencapai posisi menyerang maka suatu unit harus dapat digerakkan hingga mencapai posisi yang dapat membuat dia menyerang unit musuh. Dalam permaian turn based strategy, setiap unit memiliki jumlah gerakan yang berbeda-beda. Jumlah gerakan ini akan mempengaruhi alur strategi yang akan digunakan. Semakin besar atau jauh jumlah pergerakan yang dapat dilakukan oleh suatu unit, maka semakin banyak kemungkinan musuh yang dapat diserang oleh unit tersebut. Jika hanya sedikit atau kecil pergerakan yang dapat dilakukan oleh suatu unit, maka sedikit pula jumlah musuh yang dapat diserang oleh unit tersebut. Karena setiap pemain mempunyai jumlah unit yang lebih dari satu, dengan demikian jika suatu unit menempuh jarak terjauh yang dimilikinya maka unit lain yang memilki jarak tempuh lebih rendah dibandingkan unit tersebut lebih banyak memiliki pilihan penempatan suatu posisi. Jika suatu unit tidak mengambil jarak terjauh yang dapat dilakukan oleh unit tersebut maka akan ada kemungkinan unit lainnya tidak dapat bergerak diakibatkan posisi yang hanya dapat ditempati oleh dirinya telah ditempati oleh unit lain. Dari hal ini kita dapat membuat suatu asumsi bahwa dengan menggunakan greedy by movement maka strategi yang akan digunakan oleh AI dapat memberikan hasil yang optimum.

Algoritma greedy dengan melakukan optimasi pergerakan dari suatu unit, kita dapat menentukan elemen-elemen algoritmanya sebagai berikut:

- Himpunan kandidat

  Semua posisi atau blok yang dapat ditempati oleh suatu unit.
- Himpunan solusi

  Posisi terjauh yang dapat ditempati oleh unit
- Fungsi seleksi

  Menempati posisi terjauh yang dapat dicapai
- Fungsi layak

  Apakah posisi unit tersebut tidak menghalangi posisi unit lainnya dan dapat menyerang unit musuh
- Fungsi obyektif

  Semua unit dapat mencapai posisi terjauhnya

Dari elemen-elemen diatas kita dapat menyusun skema algoritma greedy yang digunakan.

Pertama kita akan menyusun urutan pemilihan giliran dari suatu unit, atau bisa dikatan urutan elemen dari himpunan kandidatnya yang akan diproses terlebih dahulu. Untuk urutannya, kita akan membuat agar unit yang memiliki jumlah pergerakan yang paling kecil digerakkan terlebih dahulu, jika ada beberapa unit yang memiliki jumlah pergerakan yang sama maka kita akan menggunakan sistem antrian (first in first out).

Kemudian kita akan proses unit yang telah kita urutkan sebelumnya. Pertama akan dilihat semua posisi terjauh yang dapat ditempati oleh unit tersebut, kemudian akan dilihat apakah posisi tersebut mengurangi jarak dengan musuh terdekat (dari posisi yang akan ditempati) dibandingkan dengan posisi sekarang. Jika benar, maka unit akan bergerak ke arah sana, jika tidak makan proses akan diulang dengan pemilihan jarak adalah jarak sebelumnya (jarak pertama adalah jarak terjauh yang dapat dilakukan oleh suatu unit) dikurangi 1. Proses ini dilakukan secara terus menerus hingga semua unit telah selesai digerakkan.

Pseudocode untuk algoritma greedy tersebut sebagai berikut :

```
if (S!=NULL){
    // S adalah himpunan unit yang akan digerakkan
    A = S.First(); //Posisi pertama elemen
    j = A.Jarak; // Jarak kemampuan gerak suatu unit
    while(!gerak && S!=0){
        H = getAllPosition(A,j)
        // Isi himpunan H dengan semua posisi yang
        // dapat dikunjungi A sejauh j
        while(!gerak && H!=NULL){
            if (A bisa gerak ke posisi H.First){
                // Posisi A merupakan posisi yang valid
                Move(A,H.First);
                // Gerakkan A ke posisi tersebut
                gerak = true;
            }else{
                Remove(H.First);
                // Posisi tersebut dibuang dari H
            }
        }
        j--;
        // Jarak dikurangi 1
    }
    if (isEnemyNearby()){
        Attack();
        // Jika ada musuh disekitar serang
    }else{
        Wait();
        // Jika tidak ada bertahan atau menunggu
    }
    Remove(S.First);
    // Cek unit selanjutnya
}
```

B. Greedy by Damage

Tujuan dari suatu game perperangan pastilah mengalahkan semua unit musuh yang ada, hal ini tidak terkecuali pada alur permainan turn based strategy yang bertemakan peperangan. Suatu unit musuh dikatakan kalah atau mati jika jumlah Health Point atau kesehatan dari musuh tersebut telah mencapai lebih kecil sama dengan 0. Salah satu cara yang dapat menyebabkan kesehatan dari suatu unit lebih kecil sama dengan 0 adalah ketika unit tersebut diserang oleh unit lainnya. Jumlah kesehatan suatu unit yang diserang biasanya akan berkurang sesuai dengan jumlah serangan/damage yang diberikan oleh unit penyerang. Namun ada juga beberapa game yang membuat aturan ketika suatu unit diserang, maka unit tersebut akan langsung mati. Besar serangan oleh suatu unit akan berbeda dengan unit lainnya, begitu juga dengan jumlah kesehatan yang dimiliki oleh suatu unit. Beberapa permainan juga menerapkan sebuah penanda atau status yang dapat membuat serangan suatu unit terhadap unit lainnya menjadi lebih besar daripada serangan biasanya. Status ini biasanya direpresentasikan sebagai unit penyerang merupakan musuh alami dari unit korban. Contoh dari sistem ini adalah suatu unit dengan ras tikus merupakan musuh alami dari suatu unit dengan ras kucing, oleh karena itu ketika unit dengan ras kucing menyerang unit dengan ras tikus, maka unit ras tikus akan menerima damage yang lebih besar dari pada besar serangan ras kucing. Dengan sistem ini bisa disimpulkan bahwa menyerang musuh alami dari suatu unit akan memberikan keuntungan yang lebih besar kepada pihak penyerang.

Dalam menyerang suatu unit musuh, semakin besar damage yang diberikan maka akan semakin besar pula kerugian yang diterima oleh pemain lawan. Biasanya suatu unit dikatakan mati jika kesehatan unit tersebut mencapai maksimal 0, dengan demikian ketika serangan yang dapat membuat kesehatan suatu unit berkurang hingga mencapai nilai dibawah 0 (missalkan -10) akan memberikan kerugian yang sama kepada pemain lawan dengan serangan yang hanya dapat membuat kesehatan suatu unit tepat 0. Misalkan suatu serangan dengan kekuatan 10 menyerang unit dengan kesehatan 3 akan menyebabkan kerugian yang sama kepada pemain lawan jika serangan dengan kekuatan 3 menyerang unit yang sama. Penyerangan pertama yang dapat membuat kerugian kepada pihak musuh sebesar 10 point direduksi sebesar 7 point menjadi 3 point. Hal ini tentu akan membuat penyerang tersebut menjadi kurang efektif. Ketika bermain dengan unit yang banyak, kita dapat mengatur agar unit penyerang yang memiliki serangan yang lebih kecil yang menyerang unit yang sekarat, dan unit dengan serangan yang lebih besar dapat menyerang unit lainnya, dengan demikian kerugian tersebut dapat diminimalisir.

Dari hal diatas kita dapat menyusun suatu algoritma greedy yang berdasarkan dengan jumlah damage atau kerugian yang dapat disebabkan unit pemain kepada pemain lawan. Elemen-elemen yang membangun algoritma tersebut dapat kita susun sebagai berikut :

- Himpunan kandidat

  Semua kemungkinan kerugian yang dapat diterima oleh pihak musuh.
- Himpunan solusi

  Kerugian terbesar yang dapat disebabkan kepada pihak musuh.
- Fungsi seleksi

  Kerugian pihak musuh sebesar jumlah serangan maksimal yang dapat disebabkan oleh unit penyerang.
- Fungsi layak

  Apakah serangan unit tersebut merupakan serangan yang memberikan kerugian terbesar kepada pihak musuh.
- Fungsi obyektif

  Serangan suatu unit diterima penuh oleh unit yang diserang.

Susunan urutan unit yang akan diproses diurutkan berdasarkan damage terbesar suatu unit dan sistem FIFO. Jika diurutkan berdasarkan unit dengan damage terkecil maka akan ada muncul kemungkinan permasalahan pengurangan kerugian akibat kesehatan unit musuh yang rendah.

Proses yang akan dialami oleh unit tersebut adalah memeriksa unit mana yang dapat diserang dan akan memberikan damage terbesar kepada pihak lawan.

Proses ini akan dilakukan secara terus menerus hingga semua unit telah selesai melakukan penyerangan.

Pseudocode untuk algoritma greedy by damage ini dapat dilihat sebagai berikut :

```
if (S!=NULL){
    // S adalah himpunan unit yang akan digerakkan
    A = S.First(); //Posisi pertama elemen
    E = getAllEnemy();
    // E merupakan suatu himpunan yang berisikan semua
    // unit musuh yang terurut berdasarkan kesehatan
    // terbesar
    U = E.First(); // Posisi pertama elemen musuh
    while (!Attack && U!=NULL){
        if (isCanAttack(A,U)){
            A.Attack(U); // A memnyerang U
            if (isAlive(U)){
                // Jika U masih hidup
                reArrange(E);
                // Himpunan E disusun ulang
            }else{
                Remove(U);
                // Hapus U dari himpunan
            }
            Attack = true;
            // unit A telah menyerang
        }else{
            U = E.Next(U);
            // Periksa unit berikutnya
        }
    }
    if (!Attack){
        // Jika A tidak dapat menyerang unit apapun
        A.Wait();
    }
    Remove(S.First);
    // Cek unit selanjutnya
}
```

C. Greedy by Health

Seperti yang telah dijelaskan sebelumnya, suatu unit dikatakan mati apabilah kesehatan dari unit tersebut bernilai 0 atau dibawahnya. Ketika suatu unit mati, ini tidak hanya memberikan kerugian pada pihak lawan, tapi juga memberikan keuntungan lain pada pihak penyerang. Ketika suatu unit hidup, maka unit tersebut dapat menyerang unit lawan, melindungi unit lain yang hampir mati ataupun melakukan aksi-aksi lain yang dapat mendukung suatu strategi permainan. Ketika suatu unit mati, maka strategi yang telah dibuat sebelumnya akan harus dirubah lagi karena akan dibutuhkan unit lain untuk mengisi kekosongan strategi unit yang telah mati. Pihak penyerangpun dapat menikmati keuntungan lebih karena pada gilirang selanjutnya, unit musuh yang dapat menyerang unit pemain lainnya telah berkurang. Karena itu dapat dikatakan bahwa kematian dari suatu unit akan memberikan kerugian lebih dari sekedar kerugian damage yang diterima. Walaupun ada beberapa game yang menerapkan suatu aturan unik untuk unit tertentu dan akan memberikan suatu keuntungan ketika ada unit yang mati, contohnya saja adalah dibuat status amarah, status ini akan meningkatkan damage dari unit lainnya jika ada suatu unit teman ada yang mati. Untuk khasus-khasus tertentu hal ini justu akan merugikan penyerang, karena unit lainnya akan lebih susah dikalahkan.

Dari hal tersebut kita dapat menyusun suatu algoritma greedy yang berdarkan jumlah kesehatan musuh yang akan diserang. Elemen-elemen yang membangun algoritma tersebut dapat dituliskan sebagai berikut :

- Himpunan kandidat

  Semua musuh yang dapat diserang oleh suatu unit.
- Himpunan solusi

  Serangan yang dapat membuat suatu unit mati atau sekarat.
- Fungsi seleksi

  Besar serangan unit penyerang lebih besar daripada jumlah kesehatan unit bertahan.
- Fungsi layak

  Apakah serangan unit tersebut merupakan serangan yang dapat membuat unit musuh mati.
- Fungsi obyektif

  Semakin banyak jumlah unit musuh yang mati dalam satu giliran.

Pembuatan pengecekkan proses unit yang akan menyerang akan diurutkan berdasarkan unit yang memilki kekuatan serang paling rendah. Hal ini disebabkan karena ketika suatu unit musuh dapat dibunuh dengan unit yang memiliki serangan lebih kecil, maka unit lainnya dapat menyerang musuh berikutnya.

Proses yang akan dilakukan adalah pengecekkan apakah suatu unit dapat menyerang unit musuh. Jika dapat maka himpunan unit musuh yang telah disusun dari kesehatan tertinggi akan diperiksa dengan unit penyerang, apakah penyerang dapat membunuh unit tersebut. Jika serangan unit tersebut lebih besar dari pada kesahatan unit musuh, maka unit musuh akan diserang. Jika hingga akhir unit tidak ada yang memiliki kesehatan yang lebih kecil dibandingkan dengan serangan penyerang, maka unit akan menyerang unit terakhir yang memiliki darah terkecil. Fungsi ini akan dilakukan secara terus-menerus hingga semua unit telah selesai menyerang.

Pseudocode untuk algoritma greedy ini :

```
if (S!=NULL){
    // S adalah himpunan unit yang akan digerakkan
    A = S.First(); //Posisi pertama elemen
    E = getNearbyEnemy(A);
    // E merupakan suatu himpunan yang berisikan semua
    // unit musuh yang dapat diserang oleh A dan disusun
    // terurut berdasarkan kesehatan terbesar
    U = E.First(); // Posisi pertama elemen musuh
    while (!Attack){
        if (A.Damage > U.Health && E.Next(U)==NULL){
            // Damage A dapat membunuh U atau U merupakan
            // unit musuh yang memiliki darah terkecil
            // yang dapat diserang oleh A
            A.Attack(U); // A memnyerang U
            Remove(U);
            // Hapus U dari list unit musuh
            Attack = true;
            // unit A telah menyerang
        }else{
            U = E.Next(U);
            // Periksa unit berikutnya
        }
    }
```

```
    if (!Attack){
        // Jika A tidak dapat menyerang unit apapun
        A.Wait();
    }
    Remove(S.First);
    // Cek unit selanjutnya
}
```

IV. KESIMPULAN

Pembuatan suatu algoritma strategi yang dirancang oleh sebuah AI dalam permainan turn based strategi dapat menggunakan berbagai macam metode algoritma. Salah satu algoritma untuk menciptakan strategi tersebut adalah algoritma greedy. Dengan algoritma greedy inipun masih banyak perbandingan strategi yang dapat dicipatakan seperti greedy pada saat pembelian unit, greedy pada saat pengaturan posisi penyerangan, greedy saat menyerang, greedy saat bertahan, dll. Namun pada makalah ini strategi algoritma yang dibahas adalah greedy berdsarkan pergerakan, kekuatan serangan, dan kesehatan.

Masing-masing algoritma memiliki keuntungan dan kekurangannya sendiri, kesamaan dari penggunaan algoritma ini adalah algoritma ini hanya menggunakan solusi optimum yang dapat dia temui pertama kali tanpa mempedulikan tahap selanjutnya. Penggunaan satu buah tipe algoritma tidak akan menghasilkan AI yang maksimal. Untuk menciptakan suatu AI yang dapat membuat strategi peperangan yang bagus, maka dapat dilakukan penggabungan lagoritma yang telah dibuat, yaitu AI tersebut akan menyusun strategi berdasarkan beberapa buah algoritma, seperti penggunaan algoritma greedy by movement lalu dikombinasikan dengan greedy by damage. Penggabungan algoritma ini diharapkan dapat menghasilkan suatu solusi yang efektif tidak hanyak berlaku untuk kasus lokal saja namun juga global.

REFERENCES

[1] G. Gutin, A. Yeo and A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Applied Mathematics 117 (2002), 81–86.

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

[3] http://www.catonmat.net/blog/mit-introduction-to-algorithms-part-eleven/

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, 19 Desember 2012

ttd

Benedikus Holyson Tjuatja
13510101