Penerapan Algoritma Greedy pada *Spring Challenge* 2022 Codingame

Ahmad Romy Zahran - 13520009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520009@std.stei.itb.ac.id

***Abstract*—Multiplayer bot programming* adalah persoalan optimasi yang bertujuan mengoptimalkan kriteria tertentu agar bisa menang. Namun, persoalan ini memiliki banyak kemungkinan dan bergantung pada beberapa pihak sehingga sulit atau mungkin mustahil untuk mendapatkan solusinya dalam waktu cepat. Oleh karena itu, makalah ini menunjukkan penerapan algoritma greedy yang menyediakan pendekatan praktis dan cepat dan dapat membuat keputusan dengan dasar yang jelas serta sederhana dalam waktu singkat.**

Keywords—greedy; bot; multi-agent; codingame

I. Pendahuluan

Persoalan optimasi seringkali membutuhkan waktu yang lama untuk menyelesaikannya. Ditambah bila persoalan tersebut melibatkan perilaku beberapa pihak pengendali yang tidak mengikuti aturan tertentu. Contohnya adalah persoalan *multiplayer bot programming*. Namun, terdapat beberapa pendekatan yang bisa digunakan untuk mendekati solusinya. Salah satunya adalah algoritma greedy. Algoritma greedy menyediakan pendekatan praktis dan cepat. Dengan memilih keputusan berdasarkan pengoptimalan beberapa kriteria tertentu, kita dapat membuat keputusan dengan dasar yang jelas dan sederhana dalam waktu singkat. Makalah ini menunjukkan contoh penerapan algoritma greedy pada gim bot *multiplayer* yang diambil dari platform Codingame yang cukup rutin mengadakan kontes terbuka untuk persoalan *multiplayer bot programming*.

II. Landasan Teori

A. Algoritma Greedy

Algoritma greedy merupakan jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya. Nilai maksimum sementara ini dikenal dengan istilah *local optimum*. Solusi yang diharapkan dari memilih *local optimum* adalah solusi yang paling baik secara keseluruhan atau dikenal juga sebagai *global optimum*. Algoritma greedy biasanya digunakan dalam permasalahan optimasi. Persoalan optimasi adalah persoalan-persoalan yang mencari *global optimum*. Terdapat 2 jenis permasalahan optimasi yaitu maksimasi (mencari solusi terbesar) dan minimasi (mencari solusi terkecil).

Dalam algoritma greedy, terdapat beberapa elemen yang digunakan untuk membantu proses penyelesaian masalah. Elemen-elemen tersebut diantaranya:

1. Himpunan kandidat: himpunan yang berisi elemen pembentuk solusi.
2. Himpunan solusi: himpunan yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi: fungsi untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
4. Fungsi kelayakan: fungsi yang memeriksa apakah suatu kandidat bersama dengan himpunan solusi memenuhi batasan tertentu.
5. Fungsi solusi: fungsi yang memberi tahu bila himpunan solusi yang terbentuk sudah merupakan solusi.
6. Fungsi objektif: fungsi yang ingin dioptimalkan.

III. Deskripsi Masalah

Persoalan diambil dari *Spring Challenge* 2022 yang diadakan Codingame pada tanggal 21 April 2022 - 2 Mei 2022. Codingame adalah platform pelatihan untuk programmer berbasiskan *challenge*. Codingame sering mengadakan kontes terbuka dengan persoalan biasanya seputar *multiplayer bot programming* dengan banyak pilihan bahasa pemrograman yang boleh digunakan. Persoalan pada *Spring Challenge* 2022 sendiri berjudul *Spider Attack* berupa persoalan *multiplayer bot programming* yang berkaitan dengan *multi-agent* dan *resource management*. Tersedia beberapa liga dengan setiap liga memiliki boss yang harus dikalahkan bila ingin lanjut ke liga selanjutnya. Liga yang ada diantaranya *wood league 2, wood league 1, bronze league, silver league, gold league,* dan *legend league*. Liga yang berbeda mungkin memiliki spesifikasi persoalan yang berbeda.

A. Deskripsi singkat

Tujuan dari gim adalah melindungi markas dari serangan monster dan bertahan hidup lebih lama dari musuh. Kedua player mengontrol tim terdiri atas 3 *hero*. Tim mulai di ujung yang berseberangan (kiri atas dan kanan bawah) di dekat markasnya. Seiring berjalannya gim, monster akan muncul dari

sisi peta. Bila monster mencapai suatu markas, monster akan memberi *damage*. Markas hancur bila menerima terlalu banyak *damage* dan tim kalah. Untungnya *hero* dapat membunuh monster.

Untuk *wood league 1* dan di atasnya, jika gim mencapai 220 ronde pemenang dibandingkan dari *health* markas terbanyak lalu dari *wild mana.*

B. Peta

Gim dijalankan dalam peta berbentuk persegi panjang. Pixel pada peta diindeks dari kiri ke kanan dengan x = 0 hingga x = 17630 dan diindeks dari atas ke bawah dengan y = 0 hingga y = 9000. Markas player berada pada koordinat (0,0) dan (17630,9000).

Untuk *wood league 1* dan di atasnya, terdapat kabut pada peta. Sehingga entitas yang terlihat oleh bot hanya radius 6000 unit dari markas sendiri dan radius 2200 unit dari setiap *hero*.

C. Hero

Pada setiap giliran, bot perlu mengirim command kepada setiap *hero.* Command yang dapat dikirim adalah:

- WAIT, *hero* diam di posisinya.
- MOVE, diikuti dengan koordinat peta yang membuat *hero* berpindah ke arah titik tersebut dengan perpindahan maksimal 800 unit.

Setelah semua command dijalankan, setiap *hero* akan memberi *damage* 2 pada setiap monster yang berada dalam jangkauan 800 unit.

Untuk *wood league 1* dan di atasnya terdapat command SPELL.

- SPELL WIND <x> <y>. Ditambah sejak *wood league* 1. Efek: semua entitas selain *hero* sendiri dalam radius 1280 unit akan terhempas 2200 unit searah dengan arah (x,y) dari *spellcaster*. Arah perpindahan setelah terkena WIND akan dirandom.
- SPELL CONTROL <id> <x> <y>. Ditambah sejak *Bronze league*. Efek: entitas akan bergerak ke arah (x,y) mulai ronde berikutnya. Untuk *hero* hanya bertahan 1 ronde.
- SPELL SHIELD <id>. Ditambah sejak *Bronze league*. Efek: melindungi entitas dari spell mulai ronde berikutnya selama 12 ronde.

Untuk SPELL CONTROL dan SHIELD, entitas harus berjarak maksimal 2200 unit. Dan untuk semua SPELL, dihabiskan 10 mana. Mana didapat 1 poin setiap *hero* memberi *damage* pada monster. Mana yang didapatkan di luar radius markas disebut dengan *wild mana*.

D. Monster

Setiap monster muncul dengan *health* awal 10. Bila pada akhir sebuah ronde, *health* monster berkurang menjadi 0 atau negatif, monster akan hilang dari gim. Monster muncul secara random dengan arah gerak awal random. Monster bergerak dengan arah garis lurus dengan kecepatan 400 unit per ronde.

Monster dengan jarak maksimal 5000 unit dari markas akan menarget markas tersebut dan berjalan ke arah markas. Bila monster berjarak maksimal 300 unit dari suatu markas pada akhir ronde, monster akan memberi 1 *damage* dan hilang dari gim. Monster dalam radius markas tidak dapat keluar dari peta.

Untuk Bronze league dan di atasnya, *health* awal monster bervariasi dan lebih besar di akhir gim.

E. Input-output Bot

Bot menerima informasi *health* player. Bot menerima informasi id setiap entitas yang terjangkau dan koordinatnya. Untuk setiap monster, bot menerima informasi *health* saat ini, arah kecepatan monster (vx dan vy), boolean nearBase (apakah monster menarget markas), integer threatFor (0 bila monster tidak akan mencapai markas, 1 bila monster akan mencapai markas sendiri pada suatu saat, 2 bila monster akan mencapai markas lawan pada suatu saat).

Pada *wood league 1* terdapat tambahan sebagai berikut. Bot menerima informasi mana player. Untuk setiap entitas, bot menerima informasi shieldLife (banyak ronde hingga efek shield habis) dan isControlled

Output setiap ronde adalah 3 command untuk masing-masing *hero*.

IV. PEMBAHASAN

Pemetaan persoalan pada elemen greedy:

1. Himpunan kandidat: kumpulan kemungkinan tripel command untuk setiap *hero*.
2. Himpunan solusi: tripel command yang sudah dipilih.
3. Fungsi solusi: *health* salah satu player sudah habis atau sudah 220 ronde.
4. Fungsi seleksi: kriteria untuk memilih kandidat. Untuk *challenge* pada *wood league* 2 dan 1 digunakan fungsi seleksi berupa greedy by *dist* dan *threat.*
5. Fungsi kelayakan: kandidat yang dipilih harus layak. Untuk *challenge* pada *wood league 1* dan *bronze league*, setiap command spell layak bila mana terpenuhi.
6. Fungsi objektif: meminimalkan *damage* yang diterima, memaksimalkan *damage* pada lawan, dan memaksimalkan banyak mana yang didapat di luar radius markas.

A. Strategi Greedy untuk Spesifikasi Wood League 2

Algoritma yang digunakan adalah sebagai berikut.

1. Seleksi monster dengan *threat_for* = 1. Untuk menjaga markas, hero perlu membasmi monster-monster yang mengancam saja.
2. Urutkan hasilnya berdasarkan jarak ke markas sendiri. Monster yang lebih dekat akan lebih cepat sampai

sehingga perlu dibasmi segera bila tidak mau menerima *damage*.

3. Selanjutnya *assign hero* pada tiga monster paling mengancam.
    a) Caranya dengan memilih *hero* terdekat yang belum ter-*assign* pada setiap monster.
    b) Bila ada *hero* yang belum ter-*assign* karena jumlah monster yang mengancam lebih sedikit. Pilih monster paling mengancam untuk *hero* tersebut.

Berikut implementasi greedy untuk mencari monster yang paling mengancam.

```
// greedy by dist dan threat untuk setiap monster
vector<Monster> threats;
for(int i=0;i<sz(monsters);i++) {
    Monster m = monsters[i];
    if(m.threat_for==1) {
        threats.push_back(m);
    }
}
auto cmp = [](Monster a, Monster b) {
    return sqDist(a.x,a.y,base_x,base_y) < sqDist(b.x,b.y,base_x,base_y);
};
sort(threats.begin(), threats.end(), cmp);
vector<string> commands(3, "WAIT");
```

Gambar 1. Cuplikan greedy by dist dan threat untuk setiap monster

Sumber: dokumentasi pribadi

Berikut implementasi greedy untuk mencari *hero* terdekat dan *assign* hero pada monster.

```
// assign threats ke hero terdekat
for(int i=0;i<min(sz(threats),sz(myHeroes));i++) {
    Monster m = threats[i];
    int minDist = 1000000000;
    int minIdx = -1;
    for(int j=0;j<sz(myHeroes);j++) {
        if(commands[j]!="WAIT") continue;
        Hero h = myHeroes[j];
        int dist = sqDist(h.x,h.y,m.x,m.y);
        if(dist<minDist) {
            minDist = dist;
            minIdx = j;
        }
    }
    if(minIdx!=-1) {
        commands[minIdx] = "MOVE " + to_string(m.x) + " " + to_string(m.y);
    }
}
// bila ada hero yang masih WAIT, assign hero ke threats tertinggi
for(int i=0;i<sz(commands);i++) {
    if(commands[i]=="WAIT" && sz(threats)>0) {
        commands[i] = "MOVE " + to_string(threats[0].x) + " " + to_string(th
    }
}
```

Gambar 2. Cuplikan algoritma greedy untuk assign monster pada hero

Sumber: dokumentasi pribadi

Berikut hasil pengujian melawan Boss 1 (bot *wood league 1*).

Gambar 3. Ketiga Hero mengenali ancaman dari jarak jauh saat diuji di arena Wood League 2

Sumber: Screenshot dari [1]

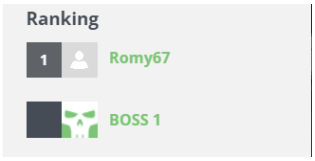
Saat diuji, program mengeluarkan output yang diinginkan untuk setiap *hero* dan dapat mengenali ancaman pada ronde itu.

Gambar 4. Bukti ketiga ancaman dikejar oleh Hero terdekat saat diuji di arena Wood League 2

Sumber: Screenshot dari [1]

Pemilihan hero terdekat untuk ancaman juga berjalan baik.

Berikutnya program diuji di arena *wood league* 2, didapat hasil lolos ke *wood league* 1.



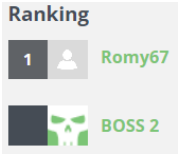
Gambar 5. Rank saat test di arena wood league 2

Sumber: Screenshot dari [1]

Berikutnya program diuji di arena *wood league* 1, didapat hasil lolos ke *bronze league*. Walaupun deteksi monster menjadi lebih sempit, program dapat menjaga markas dan menang dari Boss 2.

Gambar 6. Pengujian di arena wood league 1

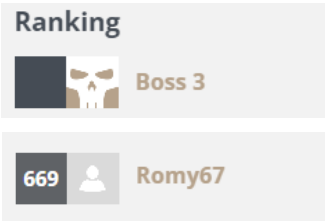
Sumber: screenshot dari [1]



Gambar 7. Rank saat test di arena wood league 1

Sumber: Screenshot dari [1]

Berikutnya program diuji di arena *bronze league*, didapat hasil kalah melawan Boss 3 dengan rank 618/1920 di *bronze league*.



Gambar 8. Rank saat test di arena Bronze league

Sumber: Screenshot dari [1]

B. Strategi Greedy untuk Spesifikasi Bronze League

Pertama dipilih posisi ideal untuk memaksimalkan jarak pandang tetapi tetap dekat dengan markas sebagai berikut.

Gambar 9. Posisi ideal yang dipilih untuk menambah jarak pandang

Sumber: screenshot dari [1]

Ketiga *hero* akan menuju posisi ideal bila tidak ada ancaman. Posisi ideal yang dipilih adalah dengan membagi rata seperempat keliling lingkaran. Berikut perhitungannya.

```
60      // compute ideal position
61      int angle[3] = {15, 45, 75};
62      for(int i=0;i<3;i++) {
63          int x = (int) round(abs(base_x - (int)(sin(angle[i]*M_PI/180.0)*6000)));
64          int y = (int) round(abs(base_y - (int)(cos(angle[i]*M_PI/180.0)*6000)));
65          idealPos[i] = Pos(x, y);
66      }
```

Gambar 10. Perhitungan posisi ideal

Sumber: dokumentasi pribadi

```
131     if(sz(threats)==0) {
132         // go to idealPos
133         for(int j=0;j<3;j++) {
134             int minDist = 1000000000;
135             int minIdx = -1;
136             for(int i=0;i<sz(myHeroes);i++) {
137                 if(commands[i]!="WAIT") continue;
138                 Hero h = myHeroes[i];
139                 int dist = sqDist(h.x,h.y,idealPos[j].x,idealPos[j].y);
140                 if(dist<minDist) {
141                     minDist = dist;
142                     minIdx = i;
143                 }
144             }
145             if(minIdx!=-1) {
146                 commands[minIdx] = "MOVE " + to_string(idealPos[j].x) + " " + to_string(idealPos[j].y);
147             }
148         }
```

Gambar 11. Kondisional untuk menuju posisi ideal

Sumber: dokumentasi pribadi

Strategi berikutnya adalah gunakan mana ketika sangat dibutuhkan atau ketika efeknya maksimal atau ketika mana terlalu banyak. Rinciannya sebagai berikut.

1. Pertama simpan daftar Monster dalam jangkauan spell setiap *hero*.
2. Cek banyak ancaman dalam radius 1280 unit setiap *hero* (jangkauan WIND). Gunakan WIND bila banyak ancaman ≥ 3. Hal ini memberikan efek WIND yang cukup maksimal.
3. Gunakan spell CONTROL ketika sangat dibutuhkan. Yaitu adanya ancaman dengan *health* > 2* banyak ronde hingga dia menyerang markas.
4. Bila mana terlalu banyak sehingga masih cukup walaupun dipakai terus menerus, gunakan spell CONTROL.
5. Untuk *hero* yang tidak memenuhi kondisi spell, digunakan command MOVE seperti sebelumnya.

Berikut implementasi dalam programnya.

```
        // USE WIND if necessary
        // if very dangerous, USE CONTROL
        // if still wait, assign hero to threats
        map<int,int> idControlled;
        vector<Monster> eachHeroThreats[3];
        for(int i=0;i<3;i++) {
            for(int j=0;j<sz(threats);j++) {
                Monster m = threats[j];
                if(sqDist(m.x,m.y,myHeroes[i].x,myHeroes[i].y)<=2200*2200) {
                    eachHeroThreats[i].push_back(m);
                }
            }
        }
        for(int i=0;i<3;i++) {
            int ctDeket=0;
            for(int j=0;j<sz(eachHeroThreats[i]);j++) {
                Monster m = eachHeroThreats[i][j];
                if(sqDist(m.x,m.y,myHeroes[i].x,myHeroes[i].y)<=1280*1280) {
                    ctDeket++;
                }
            }
            if(ctDeket>=3 && mana>=10) {
                commands[i] = "SPELL WIND " + to_string(myHeroes[i].x) + " " + to_string
                (myHeroes[i].y+myHeroes[i].y-base_y);
                mana -=10;
            }
        }
        for(int i=0;i<3;i++) {
            if(commands[i]=="WAIT") {
                for(int j=0;j<sz(eachHeroThreats[i]);j++) {
                    Monster m = eachHeroThreats[i][j];
                    if(m.is_controlled==0 && idControlled.find(m.id)==idControlled.end()) {
                        if(2*(((int) round(sqrt(sqDist(base_x,base_y,m.x,m.y)))-300)/400)<m.
                        health && mana>=10) {
                            commands[i] = "SPELL CONTROL " + to_string(m.id) + " " + to_string
                            (17630-base_x) + " " + to_string(9000-base_y);
                            mana -=10;
                            idControlled[m.id] = 1;
                            break;
                        }
                    }
                }
            }
        }
```

Gambar 12. Implementasi Strategi Spell langkah 1-3

Sumber: dokumentasi pribadi

```
        // bila mana kebanyakan, use control
        if(mana>=30*(220-curRonde) && mana >=30) {
            for(int i=0;i<3;i++) {
                if(commands[i]=="WAIT") {
                    for(int j=0;j<sz(eachHeroThreats[i]);j++) {
                        Monster m = eachHeroThreats[i][j];
                        if(m.is_controlled==0 && idControlled.find(m.id)==idControlled.end())
                        {
                            commands[i] = "SPELL CONTROL " + to_string(m.id) + " " + to_string
                            (17630-base_x) + " " + to_string(9000-base_y);
                            mana -=10;
                            idControlled[m.id] = 1;
                            break;
                        }
                    }
                }
            }
        }

        // bila ada hero yang masih WAIT, assign hero ke threats
        for(int i=0;i<sz(commands);i++) {
            if(commands[i]=="WAIT" && target[i]!=-1) {
                commands[i] = "MOVE " + to_string(threats[target[i]].x) + " " + to_string
                (threats[target[i]].y);
            } else if(commands[i]=="WAIT" && sz(threats)>0) {
                commands[i] = "MOVE " + to_string(threats[0].x) + " " + to_string(threats[0].
                y);
            }
        }
    }
```

Gambar 13. Implementasi Strategi Spell langkah 4-5

Sumber: dokumentasi pribadi

Perhatikan bahwa kelayakan selalu dicek, seperti mana ≥ 10 setiap ingin melakukan spell, monster sedang tidak dikontrol untuk spell CONTROL, dan tidak memberi command untuk mengontrol monster yang sama pada satu ronde.

Berikut pengujian saat melawan Boss 3.

Gambar 14. Spell Control untuk monster dengan health yang banyak

Sumber: screenshot dari [1]

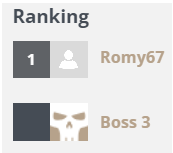
Spell Control berjalan dengan sesuai. Contohnya pada gambar terdapat 2 monster dengan *health* 23/23 dan 25/25 yang dapat ke markas dalam 11 atau 12 langkah namun membutuhkan 12 serangan dan 13 serangan. Oleh karena itu digunakan spell control untuk mengubah arah monster.

Gambar 15. Spell Wind ketika ada 3 ancaman dalam radius 1280 unit

Sumber: screenshot dari [1]

Spell Wind juga berjalan dengan sesuai, yaitu ketika ada ≥ 3 ancaman di sekitar suatu *hero*. Pada gambar X, terdapat 3 ancaman di sekitar *hero* 1.

Untuk hasil akhir melawan boss 3 sendiri didapat hasil yang bervariasi, sebagian menang sebagian kalah. Namun lebih banyak menang. Adapun untuk pengujian di arena *Bronze league* didapat hasil …



Gambar 16. Rank saat test kedua di arena Bronze league

Sumber: screenshot dari [1]

Berikutnya program diuji di arena *Silver league*, didapat hasil kalah melawan Boss 4 dengan rank 1153/2186.

V. KESIMPULAN

Persoalan *Spider Attack* berhasil dipetakan dalam elemen algoritma Greedy yang kemudian didesain dan diimplementasikan dalam bahsasa C++. Ketika pengujian, program berjalan sesuai yang diharapkan. Program diadukan dengan bot lain dan berhasil masuk ke *Silver league* dengan rank 1153/2186.

VI. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat dan berkat-Nya makalah ini dapat selesai. Penulis juga ingin mengucapkan terima kasih kepada:

1. Orang tua penulis yang selalu memberi dukungan,
2. Bapak dan Ibu dosen pengampu mata kuliah Strategi Algoritma IF2211 terutama Pak Rinaldi Munir yang sudah membimbing Kelas K3 selama satu semester ini,
3. Teman-teman yang saling memberi dukungan,
4. Codingame, platform yang menyediakan kontes dan tempat penulis mengambil screenshot visual gim.
5. Pihak-pihak lain yang tidak sempat disebutkan.

REFERENCES

[1] https://www.codingame.com/multiplayer/bot-programming/spring-challenge-2022. [diiakses 20 Mei 2022]

[2] M. Hammel, "SPIDERS EVERYWHERE ! - Spring Challenge 2022". https://www.youtube.com/watch?v=MyHjWftmMfQ. [diakses 20 Mei 2022]

[3] R. Munir. "Algoritma Greedy 2021 Bag 1". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. [diakses 20 Mei 2022]

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, 24 Mei 2022

Ahmad Romy Zahran (13520009)