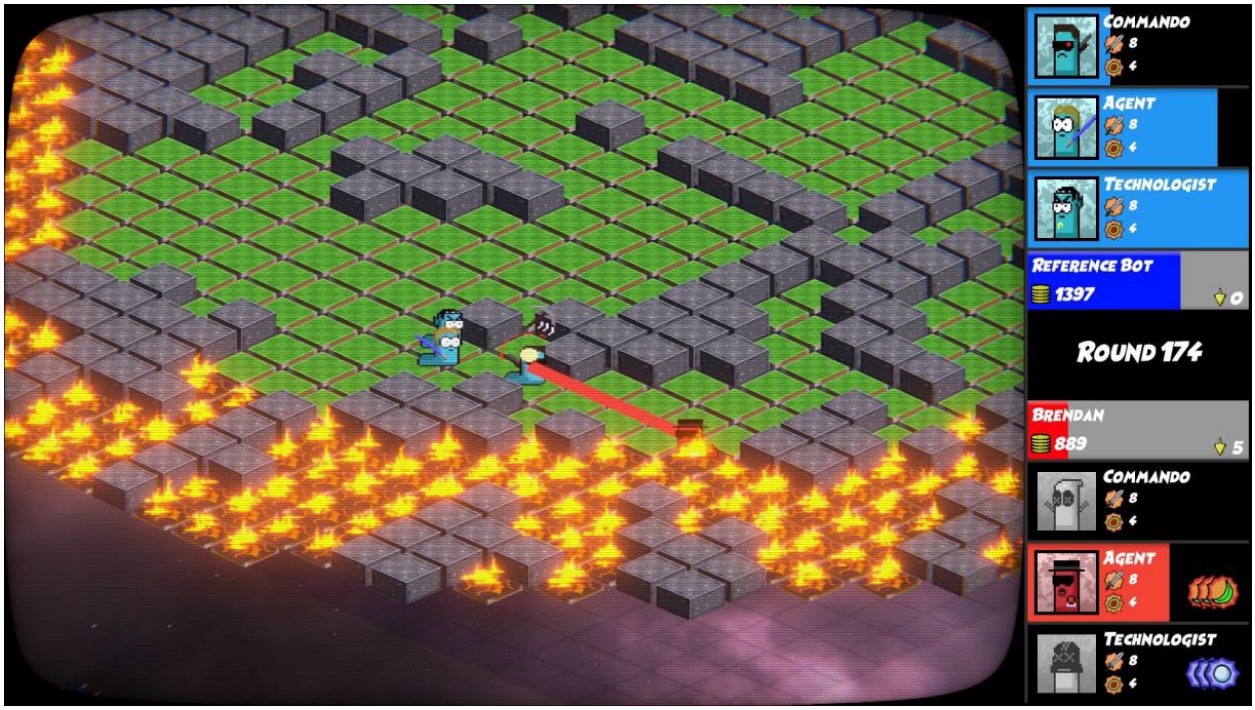
Laporan Tugas Besar Strategi Algoritma 1

Daffa Ananda Pratama Resyaly - 13519107
I Gede Govindabhakta - 13519139
Stefanus Jeremy Aslan - 13519175

Institut Teknologi Bandung
Teknik Informatika
2021

BAB I
DESKRIPSI TUGAS

Worms adalah sebuah turned-based game yang memerlukan strategi untuk memenangkannya. Setiap pemain akan memiliki 3 worms dengan perannya masing-masing. Pemain dinyatakan menang jika ia berhasil bertahan hingga akhir permainan dengan cara mengeliminasi pasukan worms lawan menggunakan strategi tertentu.



Gambar 1. Contoh tampilan permainan Worms

Pada tugas besar pertama Strategi Algoritma ini, mahasiswa diminta untuk menggunakan sebuah game engine untuk mengimplementasikan permainan Worms. Game engine dapat diperoleh pada laman https://github.com/EntelectChallenge/2019-Worms.

Tugas mahasiswa adalah mengimplementasikan seorang “pemain” Worms, dengan menggunakan strategi *greedy* untuk memenangkan permainan. Untuk mengimplementasikan seorang “pemain” tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada starter bot di dalam starter pack pada laman https://github.com/EntelectChallenge/2019-Worms/releases/tag/2019.3.2.

Spesifikasi permainan yang digunakan pada tugas besar ini disesuaikan dengan spesifikasi yang disediakan oleh game engine Worms pada tautan di atas. Beberapa aturan umum adalah sebagai berikut.

1. Peta permainan berukuran 33x33 cells. Terdapat 4 tipe cell, yaitu air, dirt, deep space, dan lava yang masing-masing memiliki karakteristik berbeda. Cell dapat memuat powerups yang bisa diambil oleh worms yang berada pada cell tersebut.

2. Di awal permainan, setiap pemain akan memiliki 3 pasukan worms dengan peran dan nilai health points yang berbeda, yaitu::

a. Commando

b. Agent

c. Technologist

3. Pada setiap round, masing-masing pemain dapat memberikan satu buah command untuk pasukan worm mereka yang masih aktif (belum tereliminasi). Berikut jenis-jenis command yang ada pada permainan:

a. Move

b. Dig

c. Shot

d. Do Nothing

e. Banana Bomb

f. Snowball

g. Select

4. Command dari kedua pemain akan dieksekusi secara bersamaan (bukan sekuensial) dan akan divalidasi terlebih dahulu. Command juga akan dieksekusi sesuai urutan prioritas tertentu.

5. Beberapa command, seperti shot dan banana bomb dapat memberikan damage pada worms target yang terkena serangan, sehingga mengurangi health pointsnya. Jika health points suatu worm sudah habis, maka worm tersebut dinyatakan tereliminasi dari permainan.

6. Permainan akan berakhir ketika salah satu pemain berhasil mengeliminasi seluruh pasukan worms lawan atau permainan sudah mencapai jumlah round maksimum (400 rounds).

Adapun peraturan yang lebih lengkap dari permainan Worms, dapat dilihat pada laman https://github.com/EntelectChallenge/2019-Worms/blob/develop/game-engine/game-rules.md.

BAB II
LANDASAN TEORI

2.1. Algoritma Greedy

Dalam menghadapi permasalahan, dibutuhkan penciptaan solusi lengkap dengan tahapan-tahapan logis dan terstruktur yang disebut dengan algoritma. Dalam pembuatan algoritma, dibutuhkan strategi sebagai arahan. Salah satu strategi algoritma yang umum yaitu strategi *greedy*, yaitu strategi memilih opsi terbaik dalam setiap tahapan tanpa memikirkan tahapan selanjutnya. Karena konsepnya yang sederhana, algoritma berstrategi greedy atau algoritma greedy cenderung mudah diciptakan untuk menghadapi berbagai permasalahan. Akan tetapi, algoritma greedy belum tentu dapat memberikan solusi dengan performa yang baik, bahkan terkadang memberikan solusi yang salah atau tidak memberikan solusi sama sekali.

2.2. Pemanfaatan *Game Engine*

Pemrogram dapat memanfaatkan *game engine* dalam beberapa hal pada program ini. Pemrogram dapat menambahkan implementasi strategi *greedy* dalam bentuk kode menggunakan bahasa java dengan memodifikasi file “Bot.java” yang berada di dalam folder “starter-bot/java/src”. Selain itu, pemrogram juga dapat melihat kelas-kelas yang telah dibuat sebelumnya dan berada pada folder “starter-bot/java/src” sebagai referensi untuk memodifikasi bot. Selain melihat kelas-kelas tersebut, pemrogram juga dapat melihat konfigurasi dari permainan yang dapat digunakan sebagai referensi untuk memodifikasi kode juga pada file “game-config.json” yang berada pada folder “starter-bot”.

Pemrogram juga dapat mengubah nama, email, *nickname*, lokasi dijalankannya bot, nama *executable* file untuk bot, dan bahasa pemrograman yang digunakan pada bot tersebut dengan memodifikasi file “bot.json” yang terdapat pada folder “starter-bot/java”. Pemrogram dapat juga mengatur banyaknya pemain dari permainan ini dan bot yang digunakan oleh tiap pemain dengan memodifikasi file “game-runner-config.json” yang juga terdapat pada folder “starter-bot”. Untuk memulai turnamen dengan pemain lainnya secara *online*, pemrogram dapat mengubah *state* “is-tournament-mode” yang berada pada file “game-runner-config.json” menjadi *true*. Setelah itu, pemain dapat mengatur jalur koneksi yang digunakan untuk berhubungan dengan pemain lainnya, bot yang ingin digunakan, serta tempat disimpannya sejarah pertempuran dengan memodifikasi *instance* dari kelas turnamen yang juga berada pada file “game-runner-config.json”.

Dalam hal menjalankan *game engine*, pemrogram dapat melakukan inisialisasi terlebih dahulu terhadap program, yaitu dengan membuka file “pom.xml” yang terdapat pada folder “src”, lalu klik kanan dan klik “Add as Maven Project”. Kemudian, pilih “java-sample-bot” yang terdapat pada tab “Maven”, pilih “Lifecycle”, dan kemudian pilih “install”. Dengan ini, *game engine* telah berhasil diinisialisasi dan pemrogram dapat langsung menjalankan *game*. Ada dua cara untuk menjalankan *game* ini. Cara pertama adalah dengan menjalankan file “run.bat” secara manual, yaitu dengan meng-klik dua kali pada file “run.bat”. Windows Command Prompt

kemudian akan secara otomatis terbuka untuk menjalankan *game* tersebut. Cara kedua untuk menjalankan *game* ini adalah dengan menjalankannya melalui Windows Command Prompt. Untuk melakukannya, pemrogram harus membuka Command Prompt, baik melalui *system search* maupun *run*. Kemudian, ubah direktori ke folder tempat program berada. Setelah itu, jalankan perintah “run.bat” dan *game* akan langsung berjalan pada Command Prompt tersebut.

BAB III
PEMANFAATAN STRATEGI GREEDY

3.1 Mapping Persoalan Algoritma

1. Himpunan Kandidat

Himpunan perintah (*command)* yang dapat dijalankan oleh bot, yaitu *Move, Dig, Shoot,* dan *Do Nothing*.

2. Himpunan Solusi

Urutan *command* yang dijalankan oleh bot.

3. Fungsi Seleksi

Command Move atau *Dig* menuju power-up, yaitu *Healing Kit*, kemudian akan menuju musuh yang tertinggi dalam hirarki *worms*, yaitu yang memiliki darah lebih rendah. *Command Shoot* akan menuju lawan dalam axis, ordinat, atau diagonal yang sesuai.

4. Fungsi Kelayakan

Memeriksa apakah *command* yang dijalankan telah berhasil mencapai *Cell* yang ingin dituju.

5. Fungsi Objektif

Bot memilih power-up yang terdekat atau musuh yang darahnya paling rendah dalam melakukan *command Move* dan *Dig*, serta bot melakukan *command Shot* terhadap musuh yang berada pada axis, ordinat, atau diagonal yang sesuai.

3.2 Alternatif Solusi Strategi Greedy

1. Hunt and Run

Strategi *Hunt and Run* merupakan alternatif solusi yang memfokuskan pemburuan worm musuh yang lemah sebelum worm musuh dapat mencetak score. Basis pengukuran tingkat kekuatan musuh berdasarkan role (Agent > Technologist > Commando) dan total HP yang dimiliki musuh. Apabila worm yang lebih kuat dekat dengan bot yang lebih lemah, bot akan memerintah worm teman untuk memburu worm musuh yang lebih lemah tersebut. Sebaliknya, worm teman akan digerakkan menjauh apabila worm musuh yang lebih kuat mendekat. Dalam proses lari dari musuh, memprioritaskan move daripada dig. Apabila terdapat medkit yang dapat diakses tanpa risiko, worm akan digerakkan ke arah medkit untuk memulihkan diri.

2. Straight to the Middle

Strategi *Straight to the Middle* merupakan alternatif solusi yang memfokuskan pergerakkan setiap worm teman secara lurus menuju tengah map. Ini ditujukan untuk menjauhi lava yang akan tampil pada turn-turn mendatang. Setelah di tengah map, worm teman akan melakukan move dengan pola melingkar untuk mencetak score tambah dan menunggu musuh. Ketika musuh sudah datang, worm dengan darah tertinggi, Commando, akan diutus untuk menyerang musuh yang muncul (menjauhi tengah map). Apabila HP Commando lebih kecil dari 50,

Commando akan move menjauhi musuh dengan berlari ke arah worm teman. Selanjutnya, worm Technical akan diutus untuk menyerang lawan yang sudah dilukai Commando, lalu kemudian Agent setelah Technical sekarat.

3. Best Current Possible Move

Strategi *Best Current Possible Move* mengumpamakan permainan sebagai permasalahan serupa *travelling salesman* yang harus menempuh setiap node pada suatu graf. Dalam konteks permainan worms, algoritma bertraversal dari node yang menunjukkan ronde sekarang ke node yang menunjukkan ronde berikutnya, dengan edge yang menghubungkan kedua node tersebut adalah semua langkah yang dapat diambil. Bobot dari edge tersebut adalah nilai ekspektasi poin dari mengambil langkah tersebut sehingga algoritma berharap untuk mendapatkan bobot atau poin sebanyak mungkin. Pada setiap ronde, algoritma selalu mengambil edge ke node berikutnya dengan bobot yang tertinggi. Apabila suatu aksi tidak dapat dilakukan, bobot tersebut menjadi negatif.

4. Prioritize Hunt and Spread

Strategi *Prioritize Hunt and Spread* merupakan alternatif solusi lainnya dimana Worm pemain akan memprioritaskan serangan terhadap musuh yang profitnya tertinggi dan Worm pemain yang satu dengan lainnya menyebar atau, dalam kata lain, jaraknya saling berjauhan. Faktor dari profit dalam hal ini adalah HP, *damage* serangan, dan terutama serangan spesial yang dimiliki Worm musuh. Karena faktor yang paling berpengaruh adalah serangan spesial maka Worm musuh yang memiliki serangan spesial tertentu akan lebih diprioritaskan untuk diserang oleh para *allied* Worm. Dalam strategi ini, Worm pemain dibuat menyebar agar Worm musuh kebingungan untuk menyerang Worm pemain dan juga agar para Worm pemain tidak menerima *damage* dan efek secara bersamaan dari *Banana Bomb* dan *Snowball* sehingga hal ini akan mengurangi efektivitas penggunaan serangan spesial dari Worm musuh.

3.3 Analisis Algoritma yang Dipilih: *Hunt and Run*

Kelebihan dari algoritma *Hunt and Run*

- Efisien
- Secara aktif memburu lawan
- Mudah diimplementasikan
- Efektif terhadap strategi yang agresif

Kekurangan dari algoritma *Hunt and Run*

- Tidak optimal jika jumlah lawan lebih dari jumlah worm yang masih aktif
- Lemah terhadap strategi yang defensif

Algoritma *Hunt and Run*	
Efisiensi	Penentuan target untuk strategi *Hunt and*

	Run memiliki kompeksitas konstan O(1), semua aksi yang melibatkan pengujian pada cell-cell dalam suatu area berdimensi dua seperti pengecekan cell disekitar *currentWorm* atau pencarian musuh dalam range *bananaBomb* memiliki kompleksitas $O(n^2)$.
Efektivitas	Efektivitas dari strategi *Hunt and Run* berbanding lurus dengan perbedaan dari *health* dari *worms* lawan. Semakin tinggi perbedaan *health* semakin efektif. Strategi *Hunt and Run* harus mendapatkan keuntungan di awal permainan agar dapat efektif. Jika strategi gagal mendapatkan posisi yang lebih baik dibanding lawan pada masa awal permainan, strategi menjadi tidak efektif.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Strategi Greedy

Pseudocode dari algoritma Hit and Run yang diimplementasikan

```
// Memilih strategi
if (currentRound <= 10) then
       currentStrategy  <- runToCenter
else
       currentStrategy <- huntEnemies


// Mengecek jika terdapat command snowBall yang valid dan optimal
if (AllyTech.health > 0 AND snowballCount > 0) then
       SBTarget = IdealThrowTarget(Technologist); // Mencari target ideal untuk
snowball

       // Jika melempar snowball ke SBTarget adalah command yang menguntungkan
       if (SnowballScore(SBTarget) > 0) then
              if (currentWorm = Technologist)
                     -> SnowballCommand(SBTarget)
              else if (remainingWormSelections > 0)
              // Jika bukan Technologist tapi masih bisa melakukan select
                     -> SelectCommand(Technologist, Snowball, SBTarget)


// Mengecek jika terdapat command bananaBomb yang valid dan optimal
if (AllyAgent.health > 0 AND bananaCount > 0) then
       SBTarget = IdealThrowTarget(Agent); // Mencari target ideal untuk banana bomb

       // Jika melempar snowball ke SBTarget adalah command yang menguntungkan
       if (BananaBombScore(SBTarget) > 0) then
              if (currentWorm = Agent)
                     -> BananaBombCommand(SBTarget)
              else if (remainingWormSelections > 0)
              // Jika bukan Technologist tapi masih bisa melakukan select
                     -> SelectCommand(Agent, BananaBomb, SBTarget)

// Mengecek jika terdapat command shoot yang valid
enemyWorm = getFirstWormInRange()
if (enemyWorm <> null) then
       direction = resolveDirection(currentWorm.position, enemyWorm.position)
       -> new ShootCommand(direction)

// Menyimpan semua block yang berada disekitar currentWorm
surroundingBlocks <- getSurroundingCells(currentWorm.position)

// Jika target berdasarkan strategi bukan -999, bergerak ke posisi target
if (getTargetPos(currentStrat) <> -999) then
       block <- ClosestCelltoTarget(surroundingBlocks, getTargetPos(currentStrat))
       if (block.type = AIR) then
              -> new MoveCommand(block.x, block.y)
       else if (block.type = DIRT) then
              -> new DigCommand(block.x, block.y)
else
       // Jika tidak, bergerak ke block yang random
       block <- random(surroundingBlocks)
```

```
      if (block.type = AIR) then
            -> new MoveCommand(block.x, block.y)
      else if (block.type = DIRT) then
            -> new DigCommand(block.x, block.y)

// Jika semua aksi lain tidak menemukan prasyaratnya, doNothing
-> new DoNothingCommand()
```

4.2 Struktur Data yang digunakan

Struktur Data	Attribut
Position	int x, axis int y, ordinat
Cell	int x, axis int y, ordinat enum type, tipe dari cell berupa AIR, DIRT, LAVA, atau DEEPSPACE PowerUp powerUp, data dari powerup jika terdapat pada cell
GameState	int currentRound, nomor ronde yang sedang berlangsung int mapSize, ukuran map int currentWorm, id dari worm yang sedang digunakan MyPlayer myPlayer, data pemain Opponent opponents[], data dari lawan Cell map[][], matrix dari peta
MyPlayer	int id, id dari pemain int score, skor dari pemain int health, health dari pemain MyWorm worms[], worm milik pemain int remainingWormsSelection, jumlah token untuk selectCommand
MyWorm	Weapon weapon, senjata milik worm Snowballs snowballs, snowball milik worm BananaBombs bananaBombs, bananaBomb miliki worm
Worm	int id, id dari pemain int health, health dari worm Position position, posisi dari worm int diggingRange, range dari command dig int movementRange, range dari command move int roundsUntilUnfrozen, lama hingga worm bebas dari freeze
Opponent	int id, id lawan int score, skor lawan int Worm[] worms, worms milik lawan

PowerUp	enum type, tipe dari powerUp (umumnya healthpack) int value, nilai dari powerUp tersebut (banyaknya health)
SnowBalls	int freezeDuration, lamanya efek freeze int range, range dari command int count, jumlah snowBall tersedia int freezeRadius, radius dari efek
BananaBombs	int damage, damage dari bomb int range, range dari command int count, jumlah bananaBomb tersedia int damageRadius, radius dari bomb

4.3 Analisis Algoritma Greedy

Strategi algoritma greedy yang dipakai, Hunt and Run, merupakan strategi yang memburu musuh dengan hp yang lebih kecil dari currentWorm ketika dijalankan. Dalam praktiknya melawan Reference Bot, bot yang telah disediakan, bot berhasil melawan Reference Bot hingga titik akhir berupa stand off. Akan tetapi, bot yang dibuat akan mengalami kekalahan apabila tidak berhasil melakukan damage yang cukup untuk membawa keunggulan dalam sisa pertempuran. Salah satu penentu kondisi yang penting adalah *health pack, power ups* berada di area tengah permainan. Apabila Reference Bot dapat bergerak ke titik tengah terlebih dahulu, maka kemungkinan besar Bot yang diciptakan akan mengalami kekalahan. Strategi ini cocok untuk musuh yang terpisah-pisah, dan kurang ideal melawan musuh yang mengelompok dengan cepat sebelum damage benar-benar dapat diberikan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Terdapat beberapa alternatif solusi algoritma *greedy* yang dapat diimplementasikan pada modifikasi strategi untuk bot dalam game “Worms”, tetapi hanya satu solusi yang dipilih, yaitu solusi yang paling mungkin dapat direalisasikan dan mengeluarkan hasil paling optimal. Dalam hal ini, alternatif solusi yang dipilih adalah solusi *Hunt and Run* sebagaimana yang telah dijelaskan sebelumnya. Penggunaan algoritma *greedy* dalam game “Worms” ini dapat dibilang lebih mangkus jika dibandingkan dengan menggunakan algoritma *brute-force*. Walaupun begitu, dalam beberapa kasus algoritma ini menjadi tidak optimal, seperti pada .

5.2 Saran

Dalam pengembangan algoritma *Hunt and Run* dapat ditambahkan perilaku untuk menghindari garis tembak musuh agar dapat menangani lawan yang lebih agresif. Kelemahan *Hunt and Run* jika tidak mendapatkan kondisi yang unggul pada ronde awal perlu diperbaiki. Dari segi efisiensi, kompleksitas pengecekan area dalam bentuk bidang dapat dikurangi terutama saat pengujian cell yang dekat dengan pemain.

DAFTAR PUSTAKA

https://github.com/EntelectChallenge/2019-Worms
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf
https://www.geeksforgeeks.org/activity-selection-problem-greedy-algo-1/