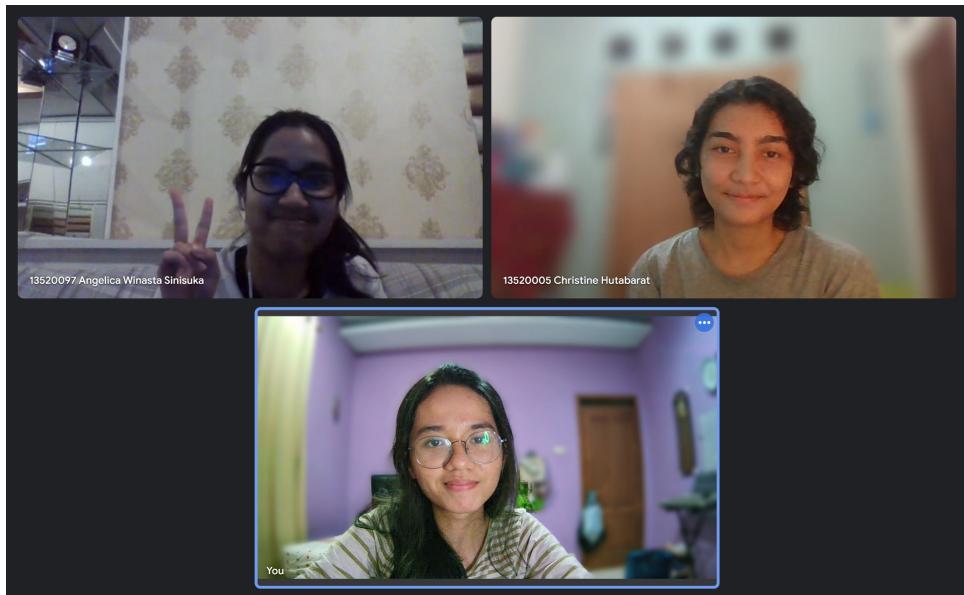


LAPORAN TUGAS BESAR I

IF2211 Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Overdrive”



Dipersiapkan oleh:
Kelompok 34 - Bandiscoot Cram

| | |
|---------------------------|----------|
| Christine Hutabarat | 13520005 |
| Monica Adelia | 13520096 |
| Angelica Winasta Sinisuka | 13520097 |

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

Daftar Isi

| | |
|--|-----------|
| Daftar Isi | 2 |
| BAB I | 3 |
| BAB II | 5 |
| Algoritma Greedy | 5 |
| Game Overdrive | 6 |
| Pengaturan Game Engine | 7 |
| Bot “Overdrive” dan Algoritma Greedy | 7 |
| BAB III | 9 |
| Persoalan overdrive dan Algoritma Greedy | 9 |
| Eksplorasi Alternatif Persoalan Greedy dan Analisisnya | 10 |
| Greedy by Speed | 10 |
| Greedy by Position | 10 |
| Greedy by Amount of Obstacles | 10 |
| Greedy by Powerup | 11 |
| Greedy by Damage | 11 |
| BAB IV | 13 |
| Repositori GitHub | 13 |
| Link Youtube | 13 |
| Implementasi Algoritma Greedy pada Program Bot dalam Game Engine | 13 |
| Penjelasan Struktur Data | 14 |
| Analisis dari Desain Solusi Algoritma Greedy | 15 |
| BAB V | 19 |
| Daftar Pustaka | 20 |

BAB I

Deskripsi Tugas

Overdrive adalah sebuah game yang mempertandingan 2 bot mobil dalam sebuah ajang balapan. Setiap pemain akan memiliki sebuah bot mobil dan masing-masing bot akan saling bertanding untuk mencapai garis finish dan memenangkan pertandingan. Agar dapat memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu untuk dapat mengalahkan lawannya.

Pada tugas besar pertama Strategi Algoritma ini, gunakanlah sebuah game engine yang mengimplementasikan permainan Overdrive. Game engine dapat diperoleh pada laman berikut:

<https://github.com/EntelectChallenge/2020-Overdrive>.

Tugas mahasiswa adalah mengimplementasikan bot mobil dalam permainan Overdrive dengan menggunakan strategi greedy untuk memenangkan permainan. Untuk mengimplementasikan bot tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada starter-bots di dalam starter-pack pada laman berikut ini:

<https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4>

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

1. Peta permainan memiliki bentuk array 2 dimensi yang memiliki 4 jalur lurus. Setiap jalur dibentuk oleh block yang saling berurutan, panjang peta terdiri atas 1500 block. Terdapat 5 tipe block, yaitu Empty, Mud, Oil Spill, Flimsy Wall, dan Finish Line yang masing-masing karakteristik dan efek berbeda. Block dapat memuat powerups yang bisa diambil oleh mobil yang melewati block tersebut.
2. Beberapa powerups yang tersedia adalah:
 - a. Oil item, dapat menumpahkan oli di bawah mobil anda berada.
 - b. Boost, dapat mempercepat kecepatan mobil anda secara drastis.
 - c. Lizard, berguna untuk menghindari lizard yang mengganggu jalan mobil anda.
 - d. Tweet, dapat menjatuhkan truk di block spesifik yang anda inginkan.
 - e. EMP, dapat menembakkan EMP ke depan jalur dari mobil anda dan membuat mobil musuh (jika sedang dalam 1 lane yang sama) akan terus berada di lane yang sama sampai akhir pertandingan. Kecepatan mobil musuh juga dikurangi 3.
3. Bot mobil akan memiliki kecepatan awal sebesar 5 dan akan maju sebanyak 5 block untuk setiap round. Game state akan memberikan jarak pandang hingga 20 block di depan dan 5 block di belakang bot sehingga setiap bot dapat mengetahui kondisi peta permainan pada jarak pandang tersebut.
4. Terdapat command yang memungkinkan bot mobil untuk mengubah jalur, mempercepat, memperlambat, serta menggunakan powerups. Pada setiap round, masing-masing pemain dapat memberikan satu buah command untuk mobil mereka. Berikut jenis-jenis command yang ada pada permainan:
 - a. NOTHING
 - b. ACCELERATE
 - c. DECELERATE
 - d. TURN_LEFT
 - e. TURN_RIGHT
 - f. USE_BOOST
 - g. USE_OIL
 - h. USE_LIZARD

- i. USE_TWEET <lane> <block>
 - j. USE_EMP
 - k. FIX
5. Command dari kedua pemain akan dieksekusi secara bersamaan (bukan sekuensial) dan akan divalidasi terlebih dahulu. Jika command tidak valid, bot mobil tidak akan melakukan apa-apa dan akan mendapatkan pengurangan skor.
 6. Bot pemain yang pertama kali mencapai garis finish akan memenangkan pertandingan. Jika kedua bot mencapai garis finish secara bersamaan, bot yang akan memenangkan pertandingan adalah yang memiliki kecepatan tercepat, dan jika kecepatannya sama, bot yang memenangkan pertandingan adalah yang memiliki skor terbesar.

Adapun peraturan yang lebih lengkap dari permainan Overdrive, dapat dilihat pada laman :

<https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md>

BAB II

Landasan Teori

A. Algoritma Greedy

Algoritma Greedy merupakan algoritma yang sering digunakan untuk memecahkan masalah optimasi, yaitu persoalan untuk mencari solusi yang paling optimal. Hanya terdapat dua macam persoalan optimasi yaitu, maksimasi dan minimasi.

Terdapat beberapa persoalan yang menggunakan algoritma greedy diantaranya adalah persoalan penukaran uang, persoalan memilih aktivitas, minimasi waktu dalam sistem, integer knapsack problem, fractional knapsack problem, penjadwalan job dengan tenggat waktu, pohon merentang minimum, lintasan terpendek, kode huffman, pecahan mesir, travelling salesperson problem, dan permainan othello.

Pada prinsipnya, algoritma greedy memecahkan persoalan secara langkah per langkah sehingga setiap langkah yang diambil merupakan pilihan terbaik tanpa melihat konsekuensi kedepannya dan berharap dengan memilih optimum lokal akan berakhir di optimum global.

Elemen-elemen algoritma greedy adalah sebagai berikut:

1. Himpunan Kandidat, C
Himpunan ini berisi kandidat yang akan dipilih tiap langkah.
2. Himpunan Solusi , S
Himpunan ini berisi kandidat yang dipilih.
3. Fungsi Solusi
Fungsi ini menentukan jika himpunan yang dipilih memberikan solusi.
4. Fungsi Seleksi
Fungsi ini memilih kandidat berdasarkan strategi greedy yang bersifat heuristik. Heuristik berarti kumpulan syarat untuk meningkatkan peluang berhasil penyelesaian masalah.
5. Fungsi Kelayakan
Fungsi ini memeriksa jika kandidat yang dipilih sudah benar jika masuk ke dalam himpunan solusi.
6. Fungsi Objektif
Fungsi ini menunjukkan jika persoalannya memaksimumkan atau meminimumkan.

Jika elemen-elemen di atas digabungkan, maka bisa dikatakan bahwa algoritma greedy menggunakan pencarian dari suatu himpunan bagian S yang didapat dari himpunan kandidat C. Beberapa kriteria harus dipenuhi oleh S, yaitu harus menyelesaikan suatu solusi dan dioptimasi dengan fungsi objektif.

Contoh persoalan yang akan dijelaskan adalah persoalan penukaran uang. Elemen greedy pada penukaran uang adalah sebagai berikut:

- Himpunan Kandidat:
 - Himpunan koin dengan pecahan nilai 1, 5 10. 25
- Fungsi Solusi:
 - Koin yang terpilih
- Fungsi Seleksi:
 - Pilihan Koin yang tertinggi dari himpunan koin yang tersisa
- Fungsi Kelayakan
 - Memeriksa jika koin yang dipilih yang kemudian dijumlahkan dengan semua koin yang berada dalam himpunan tidak melebihi jumlah yang perlu dibayar.
- Fungsi objektif:
 - Jumlah koin yang digunakan minimum

Berikut merupakan pseudocode penerapan algoritma greedy secara umum

```

function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
    x : kandidat
    S : himpunan_solusi
Algoritma:
    S ← {} {inisialisasi S dengan kosong}
    while (not SOLUSI(S)) and (C != {}) do
        x ← SELEKSI(C) {pilih sebuah kandidat dari C}
        C ← C – {x} {buang x dari C karena sudah dipilih}
        if LAYAK(S {x}) then {x memenuhi kelayakan untuk dimasukkan ke dalam
        himpunan solusi}
            S ← S u {x} {masukkan x ke dalam himpunan solusi}
        endif
    endwhile
    {SOLUSI(S) or C = {}}

    if SOLUSI(S) then {solusi sudah lengkap}
        return S
    else
        write('tidak ada solusi')
    endif

```

Tidak selalu optimum global merupakan solusi yang paling tepat dan benar. Bisa saja solusi sub-optimum atau pseudo-optimum merupakan solusi yang paling tepat karena Algoritma greedy tidak memeriksa semua kasus solusi yang bisa terjadi seperti exhaustive search. Kemudian fungsi seleksi yang digunakan berbeda-beda sehingga perlu mengidentifikasi fungsi yang paling tepat/optimal untuk sebuah persoalan seperti persoalan penukaran uang.

B. Game Overdrive

Overdrive adalah sebuah permainan yang terdiri dari dua *player* yang masing-masing memiliki dua mobil dan akan balapan untuk melawan satu sama lain. Tujuan dari permainan ini adalah untuk menjadi yang pertama yang sampai *finish line*. Peta dari balapan ini terdiri dari empat *lane* yang tersusun oleh *blocks*. Terdapat beberapa tipe *block*, yaitu *empty* (*block* kosong yang tidak memberi pengaruh pada kecepatan mobil), *mud* (memperlambat kecepatan mobil), *oil spill* (memperlambat kecepatan mobil), *flimsy wall*, dan *finish line* (akhir dari peta). *Block* memuat *powerups* yang dapat diambil oleh mobil yang melewati *block* tersebut. *Powerups* yang tersedia adalah *oil item* (menumpahkan oli di bawah mobil anda berada), *lizard* (mempercepat kecepatan mobil secara drastis), *tweet* (menjatuhkan truk di *block* spesifik yang diinginkan), dan *EMP* (menembakkan EMP ke depan jalur mobil yang akan mempengaruhi kecepatan dan *lane* musuh). Bot mobil akan memiliki kecepatan awal sebesar 5 dan akan maju sebanyak 5 *block* untuk setiap round. *Game state* memberikan jarak panjang 20 *block* di depan dan 5 *block* di belakang. Terdapat *command* yang dapat digunakan bot mobil untuk memenangkan permainan, yaitu

a. *NOTHING*

Command ini berarti mobil tidak melakukan apa-apa pada *round* tersebut. Kecepatan dan letak mobil tidak mengalami perubahan.

b. *ACCELERATE*

Command ini akan meningkatkan kecepatan mobil ke status kecepatan yang berikutnya. Mobil tetap pada jalur yang sama.

c. *DECELERATE*

Command ini akan menurunkan kecepatan mobil ke status kecepatan sebelumnya. Mobil tetap pada jalur yang sama.

d. *TURN_LEFT*

Command ini mengubah jalur mobil ke jalur berikutnya di sebelah kiri. Jika mencoba

- belok kiri di luar lintasan, mobil akan tetap pada jalur saat ini.
- e. *TURN_RIGHT*
Command ini mengubah jalur mobil ke jalur berikutnya di sebelah kanan. Jika mencoba belok kiri di luar lintasan, mobil akan tetap pada jalur saat ini.
 - f. *USE_BOOST*
Command ini digunakan untuk memakai *boost powerup* yang telah dikumpulkan mobil.
 - g. *USE_OIL*
Command ini akan membuat *block oil* tepat di bawah mobil yang mengakibatkan setiap mobil yang melewatinya akan mengalami penurunan kecepatan ke status kecepatan sebelumnya.
 - h. *USE_LIZARD*
Command ini digunakan untuk melompat untuk menghindari kadal berlari melintasi lintasan
 - i. *USE_TWEET*
Command ini digunakan untuk memunculkan truk *cyber*. Jika musuh bertabrakan dengan truk *cyber*, mereka terjebak di belakang truk *cyber* sampai akhir ronde dan kecepatannya berkurang menjadi 3.
 - j. *USE_EMP*
Command ini digunakan untuk menembak ledakan ke depan dalam bentuk kerucut ke depan dari mobil.
 - k. *FIX*
Command ini digunakan untuk melakukan perbaikan pada mobil.

C. Pengaturan Game Engine

Pada *game overdrive*, terdapat dua *player* yang akan bertarung, masing-masing memiliki *car*. Penambahan *bot* dilakukan dengan menyesuaikan *bot* yang sudah di build. Cara membangun *game engine* adalah sebagai berikut:

1. Di dalam starterpack terdapat file *game-runner-config.json*. Ubah directory pada bagian player-a atau player-b dengan directory letak tempat repository ini.
2. Ubah bagian player lainnya di *game-runner-config.json* dengan directory bot lawan. Jangan lupa untuk di *save*
3. Jalankan file *run.bat*

D. Bot “Overdrive” dan Algoritma Greedy

Dalam bot untuk overdrive ini terdapat beberapa bagian. Yang pertama adalah *game-engine*. *Game engine* bertanggung jawab untuk menegakkan aturan-aturan dalam *game* seperti yang telah dijelaskan pada bagian sebelumnya. *Game-engine* memastikan keberjalanan game agar mematuhi aturan yang sudah dibuat seperti aturan *command* dan lain-lain. *Game-engine* menegakkan aturan *game* dengan menerapkan perintah bot ke status game, jika valid. Bagian berikutnya adalah *game-runner*. *Game-runner* bertanggung jawab untuk menjalankan pertandingan antar pemain. *Game-runner* memanggil perintah sesuai yang diberikan oleh *bot* dan menyerahkannya ke *game-engine* untuk dieksekusi.

Bagian lainnya dari bot ini adalah *reference-bot*. *Reference-bot* ini hanyalah sebagai referensi logika yang akan memainkan game berdasarkan aturan yang telah ditentukan. Bagian ini dapat dimanfaatkan untuk menguji *bot* yang telah dibuat. Bagian berikutnya dari *bot* overdrive ini adalah *starter-bots*. *Starter-bot* adalah bagian tempat kita mengembangkan *bot* dengan logika sehingga *bot* dapat memenangkan permainan.

Strategi algoritma greedy dapat diimplementasikan dalam pengembangan *bot* ini. Algoritma Greedy merupakan algoritma yang sering digunakan untuk memecahkan masalah optimasi, yaitu persoalan untuk mencari solusi yang paling optimal, dimana solusi paling optimal dicari setiap *round* dengan harapan dapat menciptakan solusi optimal global. Algoritma greedy digunakan pada tiap *round* untuk menentukan pilihan paling optimal dalam memberikan *command*. Terdapat beberapa kemungkinan yang dapat dipilih ketika membuat algoritma greedy paling tepat untuk

menghasilkan *bot* yang dapat memenangkan *game*, misalnya yang fokus untuk selalu menggunakan *powerups*, yang fokus untuk menjaga kecepatan agar terus maksimal, fokus untuk memberi damage pada musuh, dan lain-lain.

BAB III

Aplikasi Strategi Greedy

A. Persoalan overdrive dan Algoritma Greedy

Strategi Greedy digunakan oleh bot dalam hal pemilihan jalan dan penggunaan power up. Bot diharuskan memilih jalan yang memiliki paling sedikit rintangan. Dalam hal ini, strategi greedy pemilihan jalan dapat diuraikan berdasarkan komponen-komponennya seperti sebagai berikut.

- Himpunan Kandidat:
 - Tetap berada di jalur, belok ke kanan, atau belok ke kiri.
- Fungsi Solusi:
 - Jalur yang dipilih merupakan jalur dengan rintangan minimum.
- Fungsi Seleksi:
 - Dari ketiga jalur, dipilih jalur dengan jumlah rintangan yang paling sedikit, dengan mengutamakan pemilihan jalur yang tengah dilalui (tidak berbelok).
 - Jika seluruh jalur memiliki bobot rintangan yang sama, gunakan power up Lizard jika ada.
- Fungsi Kelayakan:
 - Memeriksa jika command yang dikembalikan tidak melanggar batasan (tidak berbelok ke kiri saat berada di jalur paling kiri dan tidak berbelok ke kanan saat berada di jalur paling kanan).
 - Jalur yang dipilih memungkinkan mobil untuk terus dapat berjalan setelah round selesai.
- Fungsi Objektif:
 - Kecepatan bot dapat bertahan atau meningkat setelah round selesai.

Pada penggunaan power up oil, strategi greedy power up digunakan ketika mobil lawan berada pada jarak dekat di belakang mobil sendiri. Hal ini merupakan pilihan terbaik bagi bot sehingga memenuhi sifat greedy. Adapun penguraian komponen greedy dari strategi ini adalah seperti sebagai berikut.

- Himpunan Kandidat:
 - Power up oil digunakan, atau disimpan.
- Fungsi Solusi:
 - Power up oil digunakan dan berhasil dilewati oleh lawan pada round berikutnya.
- Fungsi Seleksi:
 - Oil dituangkan jika mobil berada pada jarak kurang atau sama dengan dari dua block di depan mobil lawan.
- Fungsi Kelayakan:
 - Oil berhasil dituangkan dan berhasil dilewati oleh lawan pada round ini atau round berikutnya.
- Fungsi Objektif:
 - Menuangkan oil dan menciptakan rintangan baru bagi mobil lawan.

Power up EMP digunakan jika mobil lawan berada di depan pada jalur yang sama atau sebelah kanan/kiri dengan mobil pemain. Strategi ini dipilih dengan harapan bahwa player dapat menyerang secara efisien dan berhasil memberikan damage pada lawan.

- Himpunan Kandidat:
 - EMP diluncurkan dan mobil lawan berhasil diserang atau EMP disimpan.
- Fungsi Solusi:
 - EMP berhasil diluncurkan dan mengenai lawan.
- Fungsi Seleksi:
 - Jika lawan berada di depan pemain dan pada jalur yang sama atau sebelah kanan/kiri dengan pemain, EMP diluncurkan. Jika tidak, EMP disimpan.
- Fungsi Kelayakan:

- Power up EMP diluncurkan dan mengenai lawan.
- Fungsi Objektif:
 - Lawan mendapatkan damage dan melambat dari serangan EMP.

Power up BOOST digunakan ketika damage cukup besar sehingga mempengaruhi *speed* dan juga ketika *player* musuh akan membalap kita.

- Himpunan Kandidat:
 - BOOST digunakan untuk mencepat *car*
- Fungsi Solusi:
 - BOOST berhasil digunakan dan menambah kecepatan *car*
- Fungsi Seleksi:
 - Jika *speed* sudah di bawah 6 atau ketika *player* musuh akan membalap.
- Fungsi Kelayakan:
 - *Speed* sudah cukup rendah sehingga akan mempengaruhi kecepatan *car*.
- Fungsi Objektif:
 - Kecepatan *car* meningkat.

B. Eksplorasi Alternatif Persoalan Greedy dan Analisisnya

a. Greedy by Speed

Alternatif strategi *greedy* yang pertama adalah *greedy by speed*. Dalam teknis permainan, *speed* mempengaruhi banyaknya *block* perpindahan yang bisa dilakukan *car* dalam satu *round*. Pada *game*, hal yang dapat mengurangi *speed car* adalah *obstacle* (*Mud*, *Oil Spill*, *Flimsy Wall*). Sehingga, strategi ini akan menghindari *obstacle* sebisa mungkin. Hal kedua yang mempengaruhi *speed* suatu *car* adalah berpindah jalur. Strategi ini akan membuat *car* untuk tetap berada pada satu jalur tanpa banyak berpindah kecuali karena ada *obstacle*. Hal terakhir yang dapat mempengaruhi *speed* adalah EMP dari *player* lain. Namun, karena fokusnya adalah untuk menjaga *speed*, strategi ini membuat *car* tidak mengambil *powerups* maupun menyerang lawan. Jadi, tidak ada yang dapat dilakukan jika terserang EMP karena pada dasarnya strategi ini fokus pada mempertahankan kecepatan *car*.

Kami memutuskan untuk tidak menggunakan strategi ini karena hanya fokus pada *speed* akan mengabaikan aspek *damage*. Jika *damage* semakin besar, batas maksimum dari *speed* akan semakin kecil. Selain itu, pemanfaatan *powerup* akan sangat membantu *car* menjadi lebih cepat karena bisa menghindari *block* yang dapat mengurangi *speed*, sedangkan dengan strategi ini, *powerups* yang kita miliki cenderung sedikit. Oleh karena itu, strategi ini tidak terlalu bagus untuk dapat memenangkan permainan. Strategi ini memiliki efisiensi biasa saja dengan perkiraan kompleksitas $O(n)$.

b. Greedy by Position

Alternatif strategi *greedy* lainnya adalah *greedy by position*. Strategi ini cukup simpel, yaitu membuat *car* selalu bergerak maju. Dari aturan permainan, kita ketahui bahwa yang pertama sampai *finish block* akan memenangkan pertandingan. Apapun keadaannya, yang penting *car* bergerak maju dengan harapan semakin cepat sampai *finish block*. Strategi ini akan selalu berusaha untuk bergerak maju apapun kondisinya. Baru ketika *car* tidak dapat bergerak, *powerups* akan digunakan.

Kami memutuskan untuk tidak menggunakan strategi ini karena strategi ini sangat rentan kehabisan *speed*. Karena strategi ini tidak memperdulikan *obstacle* maupun serangan lawan, *car* cenderung lebih mudah kehabisan *speed* dan mendapat banyak *damage*. Ketika *speed* sedikit, pergerakan *car* walaupun maju terus, tapi pergerakanya cenderung kecil sehingga menghambat *car* untuk segera memenangkan *race*. Efisiensi dari algoritma ini diperkirakan standard dengan kompleksitas $O(n)$.

c. Greedy by Amount of Obstacles

Alternatif strategi *greedy* lainnya adalah *greedy by lane*. Strategi ini fokus ke *lane*

dengan obstacles terkecil. Pada aturan permainan, obstacles memiliki efek ke player merugikan dan memiliki dampak masing-masing. Prinsip ini berpatokan pada jumlah obstacles setiap *lane* yang dapat diakses oleh pemain. Semakin sedikit obstacles yang dikenai semakin sedikit efek terhadap *player*. Jadi, *car* pasti akan berpindah ke *lane* dengan jumlah obstacle terkecil.

Kami memutuskan untuk tidak menggunakan strategi ini karena strategi ini tidak mempertimbangkan efek setiap obstacles pada *car* dan terlalu menggeneralisasi obstacles menjadi satu jenis. Setiap obstacles mempunyai efek untuk increase damage, atau reduce speed, ataupun keduanya yang pasti akan mempengaruhi performa *car* dan skornya. Strategi ini memiliki efisiensi dengan kasus average yaitu $O(n)$

d. Greedy by *Powerup*

Alternatif strategi greedy lainnya adalah greedy by powerup. Strategi ini berpatokan pada jumlah powerup setiap *lane*. Pada permainan ini, setiap powerup memiliki efek yang dapat merugikan mobil lawan selain itu, setiap pengambilan power up score *car* bertambah sebanyak 4. Pengambilan dan penggunaan berbanding lurus dengan score *car*. Semakin banyak yang diambil, semakin banyak powerup yang dapat digunakan. Setiap powerup digunakan, score *car* akan bertambah sebanyak 4. Namun, apapun keadaan *car*, pasti akan berbelok ke *lane* dengan jumlah powerup terbanyak.

Kami memutuskan untuk tidak menggunakan strategi ini karena strategi ini tidak menyeimbangi antara *score* dan *speed car*. Permainan ini menentukan pemenang dari mobil yang melewati *finish line* terlebih dahulu. Apabila terjadi kasus mobil lawan dan pemain tiba ke *finish line* dengan waktu yang sama, maka *score* yang akan menentukan. Oleh karena itu, strategi ini masih dianggap belum maksimal sebab memprioritas hal kedua dari objective permainan. Efisiensi dari strategi ini diperkirakan $O(n)$.

e. Greedy by *Damage*

Alternatif strategi greedy lainnya adalah greedy by *damage*. Strategi ini fokus pada *lane* dan *health* mobil. Pada permainan, setiap obstacles menyumbang *damage* yang berbeda-beda. Terdapat obstacles yang memiliki *damage* 2 atau 1. Hal itu akan berdampak pada *speed* mobil karena dengan *damage* yang semakin tinggi, kecepatan mobil semakin kecil. Pemeriksaan dilakukan dengan memeriksa jumlah obstacles pada setiap *lane* dikalikan dengan efek damagenya. Setelah dijumlahkan, setiap *lane* diperiksa *lane* yang memiliki *damage* yang paling sedikit. Selain itu, *damage* yang dialami mobil juga diperiksa, apabila *damage* mobil naik akan dilakukan fix terlebih dahulu kemudian bergerak kembali.

Urutan prioritas dari *greedy by damage* ini adalah meminimalisir *damage* yang diterima, meminimalisir dampak dari *damage* yang diperoleh *car*, dan memberikan *damage* paling besar pada musuh. Meminimalisir *damage* yang diterima dilakukan dengan melakukan FIX pada *car* setiap *damage* sudah melebihi 1 karena *damage* cukup besar pengaruhnya pada kecepatan *car*. Lalu, *damage* diminimalisir dengan menghindari *obstacle* se bisa mungkin. *Car* akan bergerak ke kiri atau kanan jika tidak ada player lain atau truck. Jika ada player lain atau truk yang menghalangi, akan digunakan *command* LIZARD. Jika total *obstacle* di *lane* sekarang lebih kecil dibandingkan di *lane* kanan atau kiri, akan diberikan *command* ACCELERATE. Namun, jika di *lane* kiri atau kanan lebih sedikit, *command* yang diberikan adalah TURN_LEFT atau TURN_RIGHT. Begitu juga dengan keberadaan *powerups*, *car* akan menerima *command* untuk memilih *lane* dengan *power ups* terbanyak. Berikutnya, untuk meminimalisir *damage* yang sudah diterima, bot akan menggunakan BOOST (jika punya) atau ACCELERATE. Prioritas yang terakhir adalah memberikan *damage* sebesar-besarnya pada musuh. Jika *player* lawan berada didepan kita baik di *lane* yang sama maupun berbeda, bot akan mengeluarkan *command* EMP. Selain itu, bot juga akan menggunakan TWEET jika memiliki *powerup* tersebut dan akan menggunakan bila ada truck di *lane* sekarang. Terakhir, bot akan mengeluarkan *command* OIL bila kita punya *powerup* tersebut dan ada musuh di belakang kita serta mereka akan membalap atau 1-3 *block* di belakang kita.

Kami memutuskan untuk menggunakan strategi ini karena kecepatan dari *car* konsisten sehingga dapat mencapai *finish line* lebih cepat. Selain itu, jika mobil terserang oleh lawan, *car* dapat memperbaiki dirinya menjadi semula kembali dengan kecepatan yang tidak terlalu rendah dan dapat kembali lagi menjadi kecepatan yang konsisten dengan harapan dapat mengalahkan kecepatan lawan. Efisiensi dari strategi ini diperkirakan $O(n)$.

BAB IV

Implementasi dan Pengujian

A. Repotori GitHub

<https://github.com/adeliaaaa/Tubes1-STIMA.git>

B. Link Youtube

bit.ly/VideoYoutubeBandiscootCram

C. Implementasi Algoritma Greedy pada Program Bot dalam Game Engine

Pseudocode sebagai berikut merupakan bagian dari implementasi program dalam game engine. Untuk komentar dimulai dengan “//” untuk memperjelas code dan bukan merupakan bagian dari pseudocode.

```
//Greedy by Damage

//Meminimalisir damage yang diterima
//Fix first if too damaged to move so speed can be consistent
if(damage car > 1) then
    FIX

// Meminimalisir damage yang diterima
// ambil jalan yang tidak ada halangan
if (ada cyberTruck atau terjebak di belakang player) then
    if (lane kiri tidak ada cyberTruck atau player) then
        TURN_LEFT
    else if (lane kanan tidak ada cyberTruck atau player) then
        TURN_RIGHT
    else
        if (lane kiri, lane sekarang, dan lane kanan ada obstacle) then
            if (punya power up lizard) then
                LIZARD
            if (obstacle lane sekarang lebih kecil daripada kanan & kiri) then
                ACCELERATE
            else if (obstacle lane kiri lebih besar daripada kanan) then
                TURN_RIGHT
            else if (obstacle lane kanan lebih besar daripada kiri) then
                TURN_LEFT
            else
                if (powerup lebih banyak di lane sekarang) then
                    ACCELERATE
                if (powerup lebih banyak di lane kiri) then
                    TURN_LEFT
                if (powerup lebih banyak di lane KANAN) then
                    TURN_RIGHT

// Meminimalisir dampak damage
if (kecepatan di bawah 6) then
    if (punya boost) then
        BOOST
    else
        ACCELERATE

// Memberikan damage pada player musuh sebesar mungkin
// Menggunakan emp jika ada opponent di depan kita dan lane kanan kiri current player
if (musuh di depan, lane kanan, atau lane kiri kita) then
    if (punya EMP) then
```

```

EMP

// Menggunakan tweet
if (punya tweet) then
    if (ada truck di lane kita dan ke block) then
        ACCELERATE
    else
        TWEET

//menggunakan oil jika ada opponent di belakang kita sejauh 3/2/1 blocks dari player
if (punya oil & musuh di belakang kita) then
    if (musuh sejauh 1-3 block atau mau membalap kita) then
        OIL

    if (punya boost) then
        BOOST

// jika tidak memenuhi semua kondisi di atas, akan ACCELERATE

```

D. Penjelasan Struktur Data

Struktur data yang digunakan dibagi menjadi 3 bagian, yaitu command, entitas, dan enum. File bot.java dan main.java disertakan dalam directory yang sama. Manfaat dari bagian-bagian tersebut adalah sebagai berikut:

1. Command digunakan untuk merender class yang telah disediakan dari game-engine. Bagian command berisi:
 - a. AccelerateCommand
Command yang digunakan untuk menambahkan *speed car* player sesuai dengan *current level* speednya
 - b. BoostCommand
Command yang digunakan untuk menambahkan speed car player sampai dengan batas maksimum car.
 - c. ChangeLaneCommand
Command yang digunakan untuk berpindah lane
 - d. AccelerateCommand
Command yang digunakan untuk menambah speed car player sesuai dengan current level speednya
 - e. DecelerateCommand
Command yang digunakan untuk mengurangi speed car player sesuai dengan current level speednya
 - f. DoNothingCommand
Command yang digunakan untuk tidak melakukan apa-apa
 - g. EMPCmd
Command yang digunakan untuk menyerang lawan dengan mengurangi speednya menjadi 3 (level speed 1)
 - h. FixCommand
Command yang digunakan untuk mengurangi damage *car* sebanyak 2
 - i. LizardCommand: command yang digunakan untuk melewati rintangan sebanyak *current speed car*
 - j. OilCommand
Command yang digunakan untuk menambah terrain oil di belakang block car
 - k. TweetCommand
Command yang digunakan untuk menambah *cyber truck*
2. Entitas digunakan sebagai object dalam gamestate Bagian Entitas berisi:
 - a. Car
Car merupakan *class* yang memiliki atribut id tipe *integer*, position bertipe *Position*,

speed tipe *integer*, state tipe *State*, damage bertipe *integer*, Powerup bertipe list *PowerUp*, boosting tibertipe pe *boolean*, dan boostcounter bertipe *integer*.

b. GameState

Merupakan class yang memiliki atribut *currentRound* bertipe *integer*, *maxRounds* bertipe *integer*, *player* bertipe *Car*, *opponent* tipe *Car*, *worldmap* bertipe *list of array of lanes*.

c. Lane

Merupakan class yang memiliki atribut *position* bertipe *Position*, *surfaceObject* bertipe *Terrain*, *occupiedByPlayerId* bertipe *integer*, *isOccupiedByCyberTruck* bertipe *boolean*

d. Position

Merupakan class yang memiliki atribut *lane* bertipe *integer* dan *block* bertipe *integer*

3. Enums digunakan untuk iterasi object pada setiap kemungkinan yang ada. Bagian enums berisi:

a. Direction

Berisi arahan mobil bergerak

b. PowerUps

Berisi jenis powerups yang tersedia

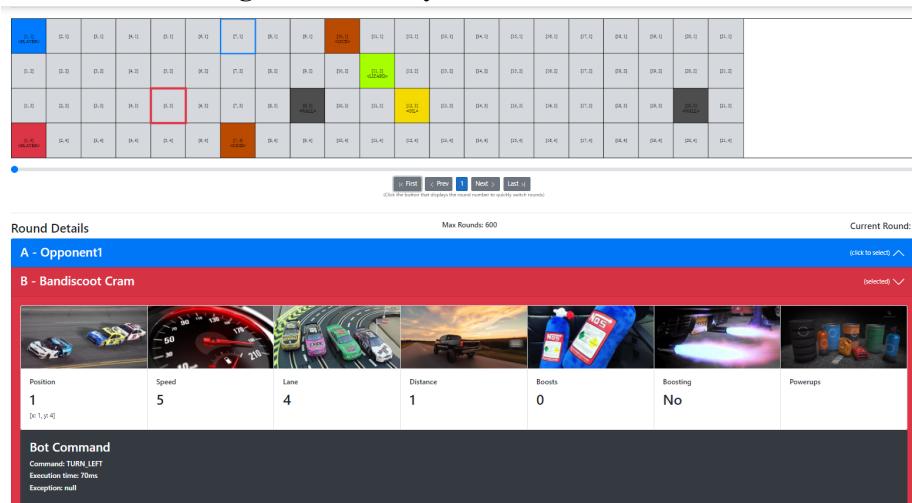
c. State

Berisi kemungkinan state yang dialami oleh car

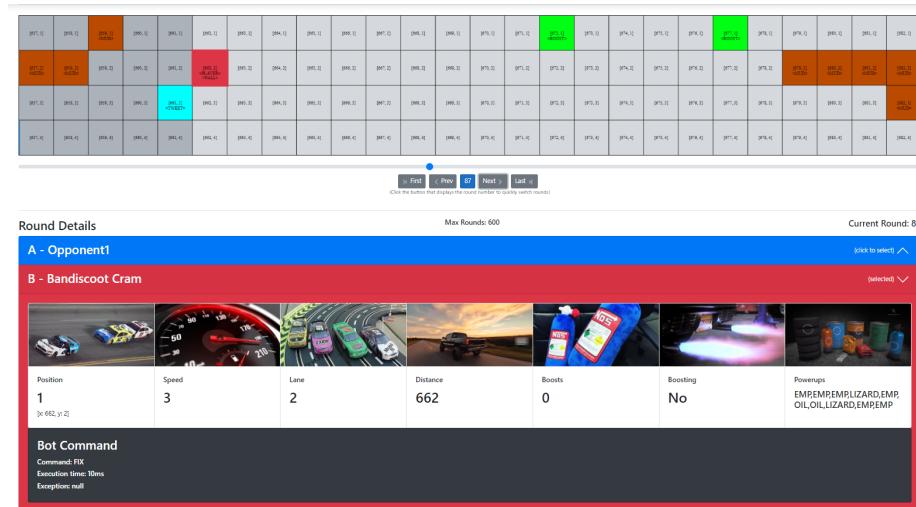
d. Terrain

Berisi object pada terrain

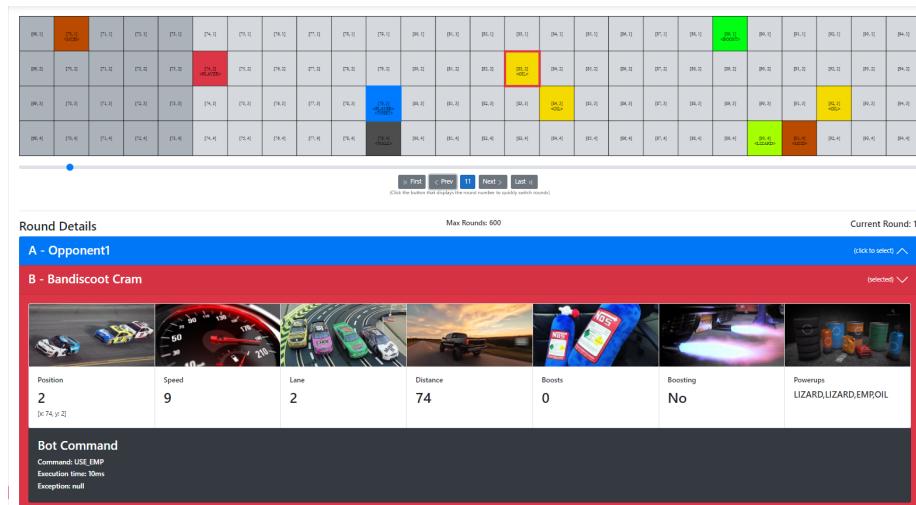
E. Analisis dari Desain Solusi Algoritma Greedy



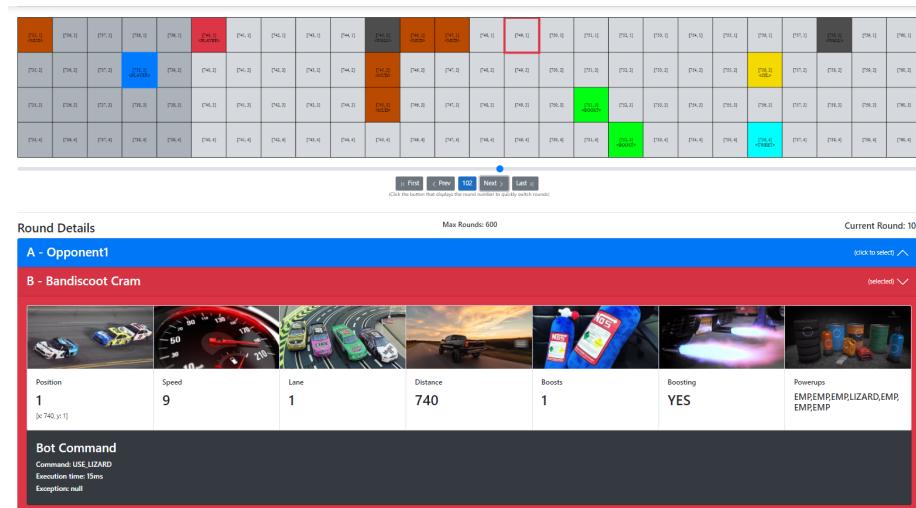
Gambar 4.1 Awal Permainan



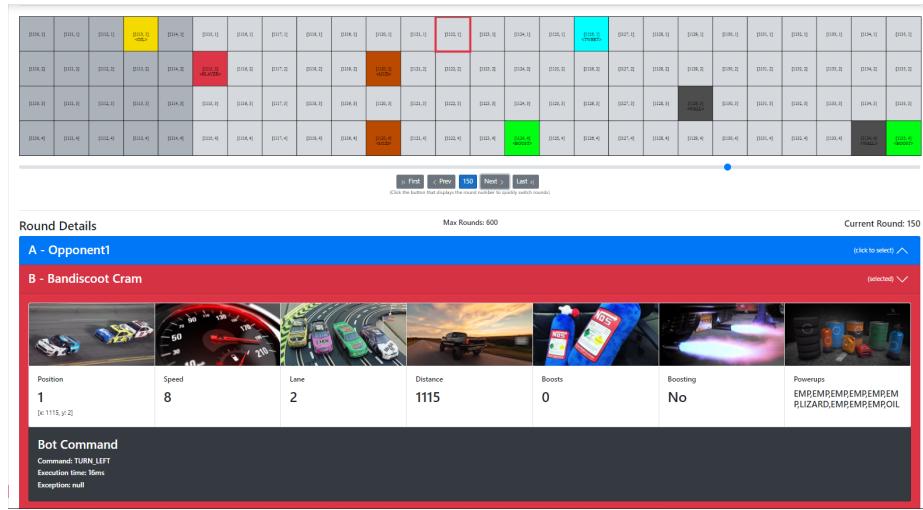
Gambar 4.2 FIX damage



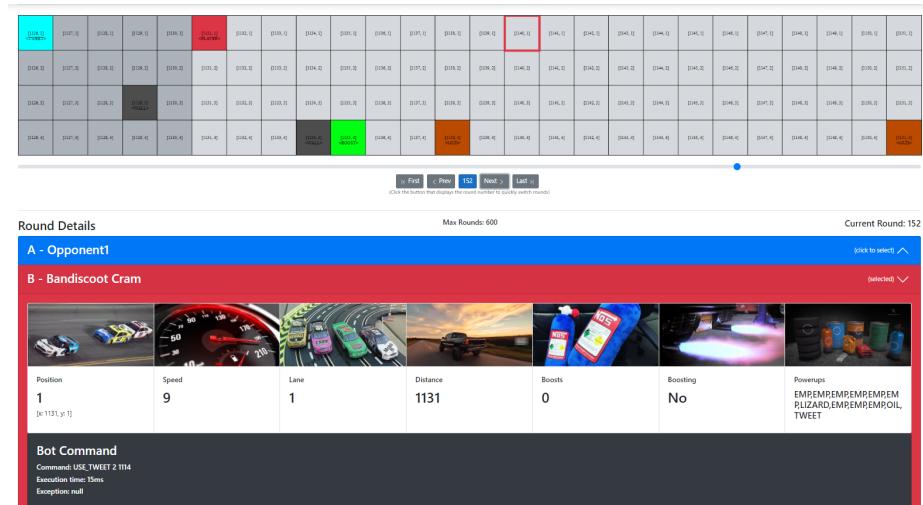
Gambar 4.3 Command EMP



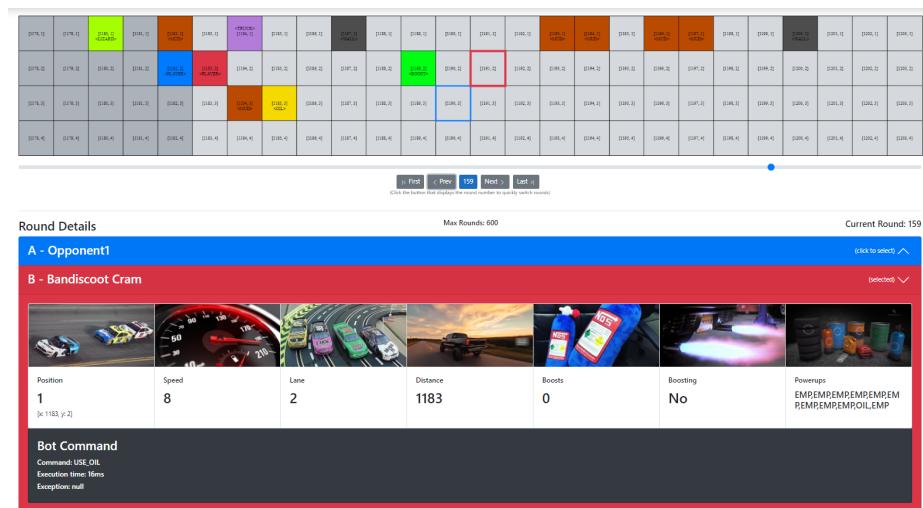
Gambar 4.4 Penggunaan LIZARD



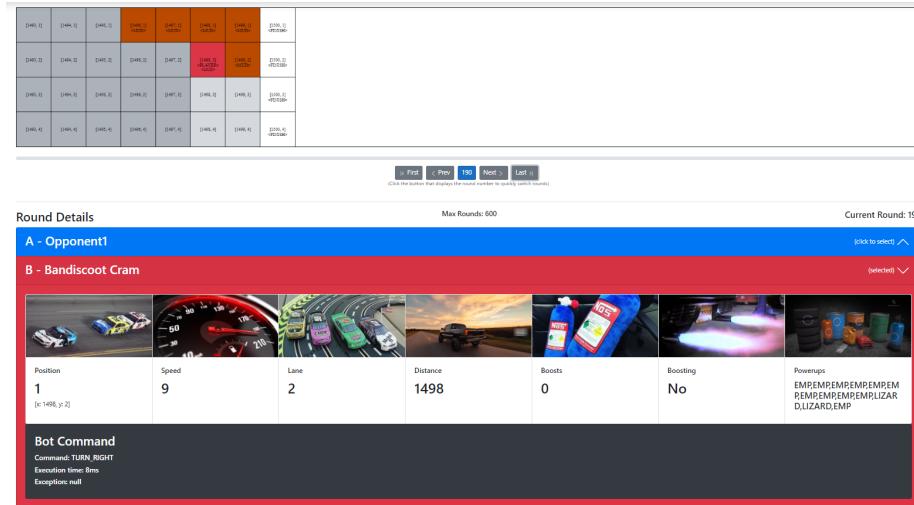
Gambar 4.5 TURN_LEFT



Gambar 4.6 Penggunaan TWEET



Gambar 4.7 Command OIL



Gambar 4.8 Akhir Permainan

Terdapat beberapa pengujian yang kami lakukan. Dari pengujian,kame menggunakan algoritma greedy by damage yang berhasil mendapatkan solusi optimal speed car. Algoritma greedy yang digunakan menurut kami solusi yang paling baik karena konsistensi speed car dan maximum speed car tetap tinggi sehingga boost yang digunakan tidak sia-sia karena memiliki counter-boost sebanyak 5. Bot ini mampu melakukan aksi greedy damage dengan menggunakan lane dengan damage yang paling sedikit. Kalau misalnya mempunyai powerup lizard dan tidak memiliki alternatif lane yang kosong maka bot menggunakannya. Bot kami dapat melakukan aksi menyerang musuh dengan EMP, TWEET, OIL, sesuai dengan fungsi seleksi kami. Bot kami juga dapat menghindari obstacles yang diset oleh lawan.

Terdapat beberapa kasus ketika strategi ini tidak optimal, yaitu ketika *car* melakukan boosting dan berada di depan lawan, *car* tidak dapat menghindari serangan dari lawan berupa EMP. Selain itu, karena fokus pada *damage*, bot menjadi fokus pada *speed* juga karena berusaha mempertahankan *damage* yang diterima seminimal mungkin. Karena itu, bot menjadi jarang menyerang bahkan disaat kritis, seperti ketika akan sampai di *finish block*. Dapat dilihat juga dari gambar 4.8 bahwa banyak EMP tersisa karena tidak digunakan.

BAB V

Kesimpulan

Algoritma Greedy berhasil diimplementasikan sebagai strategi dari suatu bot untuk dapat memenangkan permainan Overdrive. Algoritma greedy yang digunakan mencakup beberapa aspek strategi, yaitu strategi pemilihan jalan, strategi perbaikan mobil, serta strategi penyerangan lawan. Setelah melalui percobaan dan pertimbangan, kelompok kami menyimpulkan bahwa strategi *greedy* terbaik yang dapat digunakan adalah *greedy by damage*. Oleh karena itu, strategi utama yang digunakan bot kami adalah strategi *greedy by damage*, di mana bot memilih jalur dengan konfigurasi yang menghasilkan *damage* seminimal mungkin dan mengutamakan melakukan perbaikan terhadap mobil ketika didapati adanya kerusakan. Hal ini didukung dengan penggunaan *power up BOOST* yang cukup sering dilakukan bot dengan tujuan untuk mencegah adanya *damage* ketika bot menabrak *obstacle* atau diserang oleh lawan. Pengoptimalan strategi kemudian menghasilkan strategi yang tidak hanya berfokus pada *damage*, melainkan juga pada banyaknya *power up* yang dapat diambil di suatu jalur, serta pemanfaatan *power up* semaksimal mungkin dalam menjaga kestabilan kondisi mobil dan menyerang lawan.

Salah satu kelemahan yang dimiliki oleh bot kami adalah penggunaan *power up* yang berlebihan, sehingga bot sering unggul hanya dalam waktu singkat sebelum kembali tertinggal oleh lawan. Hal ini dapat diperbaiki dengan pemilihan prioritas yang lebih baik, sehingga kedudukan mobil terhadap lawan dapat dipertahankan. Selain itu, kelompok kami menemukan bahwa kombinasi dari strategi dengan perhitungan yang tepat memberikan hasil yang lebih jauh lebih baik daripada hanya menggunakan satu strategi. Untuk dapat menemukan kombinasi ini, diperlukan banyak analisis terhadap perlombaan yang telah dijalankan, mencari berbagai jenis lawan dan memperhatikan faktor-faktor yang menyebabkan kekalahan bot.

Daftar Pustaka

Munir, Rinaldi. 2022. Slide Kuliah Strategi Algoritma - Semester II Tahun 2021/2022, diakses dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm#SlideKuliah>