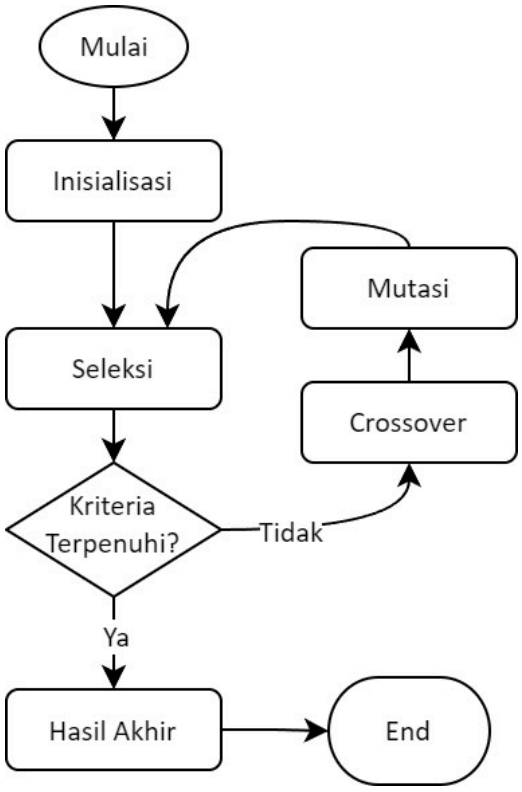
2. LANDASAN TEORI

2.1 Tinjauan Pustaka

Berikut berisikan tinjauan pustaka yang digunakan dalam pelaksanaan penelitian ini

2.1.1 Evolutionary Algorithm

Algoritma *evolutionary* atau EA merupakan metode pencarian heuristik yang terinspirasi dari teori evolusi alam Darwin berbasis populasi dimana setiap individu akan melalui proses seleksi dan evolusi hingga menyisakan individu unggul yang dapat menjadi solusi optimal untuk masalah yang dijawab (Galvan et al., 2003; Li, J., & Li, L., 2019; Masek et al., 2023; Slowik, A. & Kwasnicka, H., 2020; Weber & Notargiacomo, 2020).



Gambar 2.1 Urutan Proses Algoritma Evolusi

Seperti yang dapat dilihat pada Gambar 2.1, untuk menghasilkan individu yang unggul terdapat 3 proses utama yang selalu diulang-ulang yaitu

a. Proses seleksi

Pada proses ini sejumlah individu terlemah dalam generasinya akan dieliminasi sedangkan yang unggul akan dipertahankan dan menjadi anggota generasi berikutnya (Slowik, A. & Kwasnicka, H., 2020).

b. Proses *crossover*

Pada proses ini, akan mengembangkan sejumlah individu baru yang mewarisi sifat 2 individu yang dipilih dari generasi sebelumnya untuk menjadi anggota generasi berikutnya (Berthling-Hansen et al., 2018; Dong et al., 2023; Slowik, A. & Kwasnicka, H., 2020). Proses ini bertujuan untuk mempertahankan gen unggul dari generasi sebelumnya dan membuat individu dapat belajar dari pengalaman sebelumnya .

c. Proses mutasi

Pada proses ini, individu baru yang dihasilkan dari proses *crossover* akan mengalami mutasi dengan mengganti sebagian kecil dari sifat-nya (de Pontes & Gomes, 2020). Proses ini bertujuan untuk meningkatkan variasi individu dan mencegah terjadinya stagnasi.

Sebagai perwakilan individu, kromosom terdiri atas kombinasi sejumlah gen sebagai pembawa sifat (Fronek et al., 2020; Lim et al., 2010; Masek et al., 2023). Untuk menyimpan informasi dari *behavior tree*, jenis gen berbasis aturan merupakan jenis gen yang sesuai. Hal ini dikarenakan jenis gen ini dapat menyimpan lebih dari 1 informasi dan dapat digunakan berulang pada kromosom yang sama.

Selain itu, terdapat *fitness skor* yang merupakan suatu nilai yang diberikan kepada setiap individu berdasarkan perhitungan kriteria yang terpenuhi (de Pontes & Gomes, 2020; Dockhorn et al., 2023; Schrum et al., 2023; Thaker et al., 2015). Penggunaan *fitness function* yang tepat merupakan hal yang krusial karena *fitness function* berperan penting dalam menghasilkan individu yang unggul dan memiliki karakteristik yang diharapkan.

Menurut Schrum et al., (2023) dan Slowik, A. & Kwasnicka, H., (2020) kelebihan dari penggunaan metode EA terletak pada kemampuannya yang dapat menemukan solusi untuk permasalahan yang kompleks sekalipun karena menggunakan pencarian yang berbasis populasi. Sedangkan kelemahan terletak pada segi waktu komputasi yang relatif lama akibat ukuran populasi yang digunakan serta penggunaan *fitness function* yang tidak sesuai juga dapat membuat solusi tidak berhasil ditemukan meskipun pada masalah yang simpel (Dockhorn et al., 2023; Masek et al., 2023; Schrum et al., 2023). Meskipun demikian, hingga saat ini metode EA sudah banyak diaplikasikan untuk melakukan optimasi maupun mengembangkan *agent behavior* pada berbagai macam permainan *video* seperti Checkers, Chess, Pac-Man, Super

Mario Bros, permainan FPS, balapan mobil (Dockhorn et al., 2023; Schrum et al., 2023), DEFCON (Lim et al., 2010) dan permainan *Capture the Flag* (Fronek et al., 2020).

2.1.2 Behavior Tree

Behavior tree pertama kali diperkenalkan dalam industri permainan *video* sebagai model untuk mengontrol tingkah laku NPC yang masih banyak digunakan hingga sekarang (Agis et al., 2020; Colledanchise & Natale, 2021; Ghzouli et al., 2023). *Behavior tree* merupakan model algoritma berstruktur seperti akar pohon yang terarah, simpul dalam akar mewakili komposisi yang berfungsi untuk mengatur alur dari perilaku sedangkan simpul daun atau ujung dari akar mewakili aksi atau perintah yang akan dijalankan sehingga *agent* dapat merespon dan berinteraksi dengan lingkungan permainan (Colledanchise & Natale, 2021; Colledanchise & Natale, 2022; Guerrero-Romero et al., 2023). Simpul atau *node* dalam model *behavior tree* terbagi atas 4 jenis utama sebagai berikut:

a. *Root node* atau *node* eksekusi

Node ini merupakan pusat serta *node* teratas dalam sebuah *behavior tree* yang hanya memiliki 1 *node* sebagai *child*-nya (Agis et al., 2020; Ghzouli et al., 2023). *Root node* memiliki fungsi utama sebagai pengontrol jalannya *behavior tree* dengan memberi sinyal *tick* kepada *node* dibawahnya dengan interval yang telah ditentukan. Setiap menerima *tick, node* yang menerima akan langsung dieksekusi.

b. *Composite node*

Node ini merupakan salah satu simpul dalam yang berfungsi untuk mengatur alur atau urutan eksekusi dalam sebuah *behavior tree* (Agis et al., 2020). Tidak seperti *root node*, *node* ini dapat memiliki 2 atau lebih *node* sebagai *child*-nya. Terdapat 3 jenis *composite node* yang dibedakan berdasarkan urutan eksekusi *child*-nya.

i. *Selector*

Bekerja dengan mengeksekusi satu persatu *child-node* yang dimiliki dari kiri ke kanan dan akan langsung selesai ketika ada *node* yang berhasil dieksekusi atau tidak ada *child-node* tersisa (Agis et al., 2020).

ii. *Sequence*

Bekerja dengan mengeksekusi satu persatu *child-node* yang dimiliki dari kiri ke kanan dan hanya akan melanjutkan eksekusi ke *child-node* berikutnya ketika *child-node* sebelumnya selesai tereksekusi hingga tidak ada

child-node tersisa (Agis et al., 2020; Colledanchise & Natale, 2021; Colledanchise & Natale, 2022).

iii. *Parallel*

Berbeda dengan 2 jenis *node* diatas, *node* ini bekerja dengan mengeksekusi semua *child-node* secara bersamaan (Agis et al., 2020; Colledanchise & Natale, 2021; Colledanchise & Natale, 2022; Ghzouli et al., 2023).

c. *Decorator node*

Sebagai salah satu simpul dalam, *node* ini juga dapat mengontrol jalannya alur eksekusi namun dengan mengubah hasil dari eksekusi *node* itu sendiri. Berdasarkan Agis et al. (2020) terdapat 3 jenis *decorator node* yang dibagi berdasarkan fungsinya.

i. *Inverter*

Berfungsi untuk membalikan hasil eksekusi yaitu berhasil menjadi gagal dan gagal menjadi berhasil.

ii. *Repeater*

Berfungsi untuk memberikan perintah pada *node* untuk dieksekusi sekali lagi sejumlah batas yang diinginkan.

iii. *Condition*

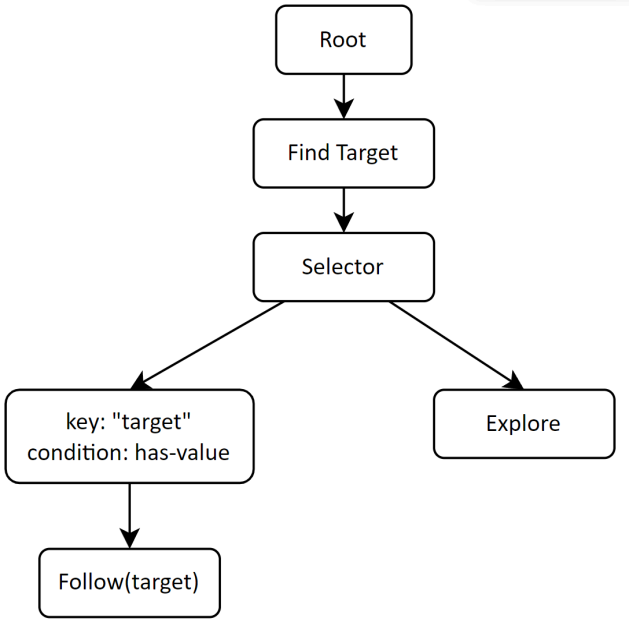
Berfungsi sebagai gerbang yang menentukan apakah *node* dapat dieksekusi atau tidak berdasarkan kondisi tertentu. Ketika kondisi tidak terpenuhi maka, *node* akan langsung dinyatakan gagal tereksekusi.

d. *Leaf node*

Node ini adalah *node* akhir dari *behavior tree* yang juga merupakan aksi atau perintah yang akan dieksekusi oleh NPC sebagai bentuk respon atau interaksi dengan lingkungan permainan.

Proses perancangan sebuah *behavior tree* secara manual diawali dengan menentukan jenis *leaf node* yang akan dimiliki *agent* seperti mengejar, patroli, menyerang dan sebagainya (Agis et al., 2020; Ghzouli et al., 2023). Setelah itu, menentukan informasi yang dapat digunakan *agent* sebagai *input* kondisi dalam menentukan aksi yang akan dilakukan. Setelah perancangan node, penyusunan *behavior tree* akan dimulai dengan memasangkan setiap *node* kondisi membentuk akar secara berurutan berdasarkan urutan pengecekan yang telah ditentukan. Kemudian, pada akhir akar akan dipasangkan node aksi sebagai aksi yang akan

dijalankan oleh *agent.* Contoh *behavior tree* dapat dilihat pada Gambar 2.2, dimana *agent* akan mengikuti target jika target ditemukan atau melakukan eksplorasi jika target tidak ditemukan.



Gambar 2.2 Contoh Struktur *Behavior Tree*

Keuntungan penggunaan model *behavior tree* terletak pada strukturnya yang bersifat fleksibel, modular, node yang dapat digunakan secara berulang kali dan simple sehingga mudah untuk dipahami oleh pengembang (Colledanchise et al., 2019; Masek et al., 2023; Robertson & Watson, 2015). Selain itu, relasi *parent-child* sederhana yang dimiliki *node* pada *behavior tree* juga menjadikan model ini mudah untuk dimodifikasi pada proses *crossover* dan mutasi milik algoritma *evolutionary* (Colledanchise et al., 2019). Penerjemahan kromosom dari proses evolusi menjadi bentuk *behavior tree* dan sebaliknya akan dilakukan menggunakan proses berikut:

2.1.2.1 Menerjemahkan Kromosom ke Behavior Tree

Dalam penelitian yang dilakukan oleh Fronek et al., (2020) untuk melakukan evolusi *behavior tree*, *behavior tree* diubah menjadi bentuk kromosom dimana setiap *node* yang terdapat pada *behavior tree* diwakili oleh gen-gen pada kromosom. Disini gen terbagi menjadi 2 jenis berdasarkan *node* yang diwakilinya yaitu gen kondisi dan gen aksi. Gen kondisi akan berperan untuk menyimpan informasi berupa jenis kondisi, operator, dan batasan yang kemudian akan dikonversi kedalam bentuk *node* kondisi di *behavior tree*. Sedangkan gen aksi

akan berperan untuk menyimpan informasi berupa jenis aksi yang kemudian akan dikonversi ke dalam bentuk *leaf node* pada *behavior tree*. Dengan begitu, kromosom dapat dengan mudah diterjemahkan kedalam bentuk *behavior tree* ketika ingin digunakan *agent* dan sebaliknya jika ingin di evolusi.

2.1.3 Procedural Content Generation (PCG)

Teknik *Procedural Content Generation* atau PCG telah digunakan dalam industri permainan *video* sejak tahun 1980-an melalui permainan berjudul "Rogue" karya Toy dan Wichman pada tahun 1980 dan "Elite" karya Acornsoft pada tahun 1984 (de Pontes & Gomes, 2020; Gravina et al., 2019; Liapis et al., 2013). PCG merupakan teknik perancangan konten menggunakan perhitungan suatu algoritma untuk dapat menghasilkan konten yang memenuhi kriteria secara otomatis (de Pontes & Gomes, 2020; Lara-Cabrera et al., 2014; Melotti & de Moraes, 2019; Soares De Lima et al., 2019). Tujuan pengaplikasian teknik PCG tidak lain dari menghasilkan beragam konten yang memenuhi kriteria yang diinginkan dan dapat dimainkan, namun berbeda satu dengan yang lainnya (Gravina et al., 2019).

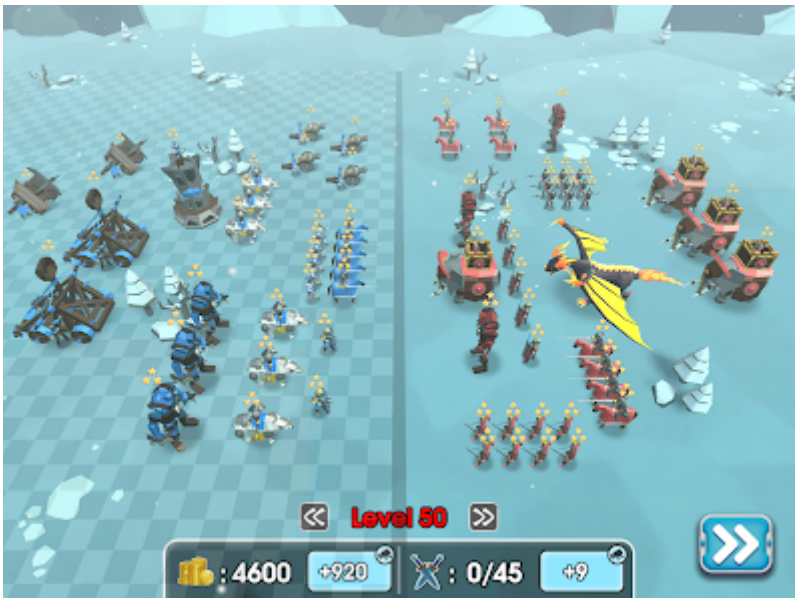
Hingga saat ini, teknik PCG telah banyak diterapkan untuk mengembangkan berbagai konten seperti level, maps, items, senjata, quest, karakter, rules (de Pontes & Gomes, 2020; Liapis et al., 2013; Liu et al., 2021; Summerville et al., 2018), misi (Lara-Cabrera et al., 2014), rewards, resource distribution, posisi, kombinasi dialog, hingga pola behavior (Melotti & de Moraes, 2019). Selain itu, penggunaan teknik PCG juga memberikan berbagai keuntungan bagi pengembang permainan *video* juga dapat memperoleh berbagai keuntungan seperti meningkatkan produktivitas hingga mengurangi biaya produksi (Hendrikx et al., 2013; Togelius et al., 2011). Penerapan teknik PCG juga dapat menjadi poin unggul dalam sebuah permainan, karena dapat meningkatkan pengalaman pemain melalui variasi konten yang beragam.

2.1.4 Real Time Strategy

Real Time Strategy atau RTS merupakan bagian dari dari kategori permainan strategi yang sering kali disajikan dalam bentuk permainan perang militer (Huat & Teo, 2012; Liu et al., 2016; Ponsen et al., 2006). Karakteristik dari kategori ini terletak pada lingkungannya yang berlangsung secara *real time* dimana setiap karakter dapat melakukan banyak aksi tanpa perlu menunggu tanggapan atau tindakan karakter lain yang berkebalikan dari kategori *turn-based* (Elfeky et al., 2021). Oleh karena itu, keputusan pemain yang tepat dan cepat dalam menghadapi situasi lingkungan yang berubah-ubah menjadi hal yang krusial untuk

keberlangsungan permainan ini (Barambones et al., 2023; Bavelier & Green, 2019; Buro, 2003; IndiaGDC, 2022; Kang et al., 2022; Liu et al., 2016). Disisi lain, karakteristik permainan RTS yang seperti simulasi dunia nyata membutuhkan variasi NPC (*Non Player Character*) yang fleksibel dan dapat berpikir. Pada permainan ini, NPC diharuskan untuk mampu memprediksi dan membuat keputusan layaknya manusia sebagai penyeimbang permainan dan pendukung imajinasi pemain (Barambones et al., 2023; Bavelier & Green, 2019; Buro, 2003; IndiaGDC, 2022; Kang et al., 2022; Liu et al., 2016; Weber & Notargiacomo, 2020).

Melihat hal itu, mengembangkan variasi AI yang mampu berpikir dan membuat strategi jangka pendek dan panjang bagaikan manusia dapat menjadi tantangan tersendiri bagi para pengembang AI NPC untuk permainan RTS (Buro, 2003; Kang et al., 2022; Riedl & Zook, 2013; Robertson & Watson, 2014). Salah satu contoh dari permainan kategori ini dapat dilihat pada Gambar 2.3.



Gambar 2.3 Permainan Epic Battle Simulator 2

Sumber: Rappid Studios (2017). Epic Battle Simulator 2. Google Play. https://play.google.com/store/apps/details?id=com.rappidstudios.simulatorbattlephysics&hl=en_US&pli=1

2.1.5 Non Player Character (NPC)

Non Player Character atau NPC merupakan sebutan untuk karakter dalam suatu permainan yang tidak dapat dikendalikan oleh pemain secara langsung (Agis et al., 2020). Pergerakan karakter ini sendiri, diatur menggunakan algoritma maupun susunan perintah yang

telah ditentukan oleh pengembang berdasarkan peran karakter tersebut dalam permainan. NPC sendiri memiliki peran yang beragam dalam permainan, seperti sebagai karakter yang melawan pemain, membantu pemain, maupun yang bersikap netral.

Dalam merancang AI NPC yang statis, *behavior tree* menjadi model yang sering digunakan oleh pengembang dikarenakan memiliki pembacaan yang mudah dan memungkinkan pengembang untuk merancang *behavior* yang kompleks melalui dengan konsep keputusan berdasarkan suatu kondisi (Agis et al., 2020). Disisi lain, *behavior* NPC yang bervariasi dan tidak mudah diprediksi pemain dapat menjadi ketertarikan sendiri bagi para pemain untuk memainkan permainan tersebut lebih mendalam.

2.1.6 Pengujian Skala Likert

Skala Likert merupakan skala penilaian yang dikembangkan pertama kali oleh Rensis Likert pada tahun 1932 (Alhassan et al., 2022). Hingga hari ini, skala likert masih sering digunakan untuk mengukur pendapat, sikap, atau persepsi individu terhadap suatu topik. Dalam pengaplikasiannya melibatkan pengambilan kuesioner yang terdiri dari serangkaian pernyataan yang mewakili pendapat yang ingin diukur (Joshi et al., 2015). Setelahnya, setiap individu akan diminta untuk memberikan tanggapan berupa tingkat persetujuan yang dibagi menjadi 5 skor yang dapat dilihat pada Tabel 2.1 (Alhassan et al., 2022; Joshi et al., 2015).

Tabel 2.1

Tabel Tingkat Persetujuan

Tingkat Persetujuan	Skor
Sangat Setuju (ST)	5
Setuju (S)	4
Netral (N)	3
Tidak Setuju (TS)	2
Sangat Tidak Setuju (STS)	1

Data yang telah diperoleh dari metode ini kemudian dapat diolah dan dianalisis untuk memperoleh informasi yang diinginkan (Joshi et al., 2015). Metode pengolahan yang dapat dilakukan juga cukup beragam dari nilai rata-rata dan standar deviasi, hingga metode statistik lainnya seperti korelasi, variasi, dan analisis regresi.

2.2 Tinjauan Studi

Berikut merupakan tinjauan studi yang berkaitan dengan pengaplikasian algoritma evolusi dalam menghasilkan *behavior agent*.

2.2.1 Procedural Creation of Behavior Trees for NPCs

Procedural creation of behavior tree merupakan perancangan behavior tree yang dilakukan secara otomatis dengan algoritma yang telah ditentukan sebelumnya (Fronek et al., 2020). Penelitian ini berfokus pada pengembangan *behavior tree agent* permainan *Capture the Flag* secara prosedural. Penelitian ini menggunakan teknik PCG yang dikombinasikan dengan metode EA untuk mengembangkan *behavior tree agent* secara otomatis. Penelitian ini menggunakan permainan *Capture the Flag* berkategori *turn-based* yang juga mengangkat sebagian aspek kategori RTS dengan harapan *behavior agent* dapat menentukan aksi untuk strategi jangka panjang maupun jangka pendek. Hasil dari penelitian ini menunjukkan bahwa pendekatan yang diterapkan mampu untuk menghasilkan beberapa *behavior agent* yang cerdas dan dapat belajar dari pengalaman tempur sebelumnya.

Kelebihan penelitian ini terletak pada keberhasilan penerapan teknik PCG dan metode EA untuk menghasilkan beberapa variasi *behavior* NPC cerdas secara otomatis yang diambil dari kromosom dengan skor tertinggi sepanjang *training* berjalan (Fronek et al., 2020). Kekurangan pada penelitian ini terletak pada variasi behavior yang dihasilkan akan langsung digunakan sebagai behavior agent yang akan sama untuk setiap pertandingannya. Disisi lain, skripsi ini akan menerapkan teknik PCG dan metode EA untuk mengembangkan *behavior* NPC yang bervariasi pada permainan *War Simulator* yang berkategori RTS. Kemudian menambahkan 3 jenis *behavior* NPC dengan fokus berbeda berupa *swordsman, shooter,* dan *staff-fighter*. Serta menggunakan generasi akhir iterasi sebagai hasil akhir.

2.2.2 Evolving Behavior Trees for the Commercial Game DEFCON

Penelitian ini berfokus pada pengembangan AI komandan untuk mengungguli AI asli dari permainan DEFCON yang juga merupakan permainan berkategori RTS (Lim et al., 2010). Penelitian ini juga menggunakan pendekatan metode evolusi untuk menghasilkan *behavior tree* yang cerdas dan dapat belajar dari pengalaman tempur sebelumnya. Hasil dari penelitian ini kemudian menunjukkan bahwa *behavior* komandan yang dikembangkan mampu mengungguli AI asli dari permainan DEFCON lebih dari 50% saat pengujian dilakukan.

Kelebihan dari penelitian ini terletak pada keberhasilannya dalam menghasilkan *behavior* komandan yang kompetitif menggunakan algoritma evolusi (Lim et al., 2010). Sedangkan kekurangan dari penelitian ini adalah *behavior* yang dikembangkan dikhususkan untuk mengungguli AI asli pada permainan DEFCON tanpa adanya variasi. Sedangkan yang dikembangkan dalam skripsi ini adalah *behavior agent* permainan *War Simulator* yang bervariasi dan berjalan sendiri tanpa pemimpin untuk melawan variasi *behavior* lainnya pada tim musuh.

2.2.3 Evolutionary Algorithm for Parameter Optimization of Context Steering Agents

Penelitian ini berfokus pada penerapan metode evolusi untuk mengoptimasi parameter *agent* sehingga *agent* mampu beradaptasi di lingkungan yang bervariatif (Dockhorn et al., 2023). Penelitian ini menggunakan lingkungan dengan struktur yang bervariasi seperti lokasi dan jumlah rintangan hingga rintangan yang dapat bergerak. Hasil dari penelitian ini kemudian menunjukkan bahwa penerapan metode evolusi berhasil mengoptimasi konfigurasi parameter *agent* sehingga mampu menelusuri lingkungan sembari menghindari rintangan yang bervariasi.

Kelebihan dari penelitian ini terletak pada keberhasilannya yang menunjukkan bahwa metode EA efisien diterapkan untuk menyelesaikan permasalahan optimasi parameter untuk AI NPC pada lingkungan yang bervariatif dimana menghasilkan 1 kombinasi terbaik (Dockhorn et al., 2023). Kekurangan dari penelitian ini adalah *behavior* yang dikembangkan belum dapat mencakup skenario yang lebih kompleks sehingga masih diperlukan penelitian yang lebih lanjut. Melihat hal itu, pada skripsi ini algoritma evolusi akan dikombinasikan dengan teknik PCG untuk menghasilkan *behavior agent* yang bervariasi untuk kondisi simulasi perang.