Aplikasi Game Strategi Menggunakan Metode Algoritma Genetika untuk Pembuatan Pasukan

Johan Pranata[1], Kristo Radion Purba[2], Liliana[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: pranata_johan@yahoo.com[1], kristo@petra.ac.id[2], lilian@petra.ac.id[3]

ABSTRAK

Salah satu cara yang dapat digunakan untuk melakukan pendekatan edukasi yang tidak membuat bosan adalah dengan menggunakan *game*. Salah satu *genre game* yang dapat digunakan untuk memberikan aspek hiburan dan edukasi pada pemainnya adalah *game turn-based strategy*. Sebagai *game turn-based strategy*, Battle of Batavia memerlukan *AI* lawan yang dapat mengambil keputusan untuk dapat membuat dan menggerakkan pasukan dengan baik agar membuat *game* lebih menarik untuk dimainkan.

Untuk membuat *AI* yang dapat berpikir dengan baik, dibuatlah suatu sistem *AI* yang dapat berpikir dengan baik di dalam *game*. Sistem yang dibuat menggunakan metode algoritma genetika dalam mengambil keputusan dalam membuat dan menggerakan pasukan. Aplikasi yang dibuat untuk skripsi ini adalah *game* Battle of Batavia itu sendiri.

Hasil pengujian menunjukkan bahwa *AI* algoritma genetika dapat dikembangkan agar *AI* dapat berpikir lebih baik lagi dengan menentukan kriteria *fitness* apa saja yang berpengaruh pada penentuan pembuatan dan pergerakkan pasukan. Hasil juga menunjukkan bahwa metode *AI* algoritma genetika mampu mengambil keputusan yang lebih baik daripada metode *random* pada beberapa keadaan *game.*

Kata Kunci: Kecerdasan buatan, Algoritma Genetika, *Turn-based strategy game*.

ABSTRACT

One way that can be used to provide education in an interesting way is using game. A kind of game's genre that can be used to provide entertainment and educational to the player is a turn-based strategy game. As a turn-based strategy game, Battle of Batavia requires AI as opponent which can take good decision for making and controlling units to make the game become more attractive.

To create AI that can think well, an AI system that can think properly in game is created. The system uses genetic algorithm method to take decision in making and controlling units. In this thesis, there are 2 component created for this thesis, the first is the genetic algorithm system and the second is Battle of Batavia game.

The results show that the genetic algorithm system can be developed so the AI can think much better by determining fitness criteria that affect on taking decision in making and controlling units. The results also show that the genetic algorithm system can make a better decision than random method on some in-game conditions.

Keywords: *Artificial Intelligence , Genetic Algorithm, Turn-based strategy game.*

1. PENDAHULUAN

Perkembangan teknologi saat ini melaju dengan pesat, diikuti dengan kebutuhan manusia, tak terkecuali dalam bidang hiburan. Salah satu teknologi yang berkembang dalam bidang hiburan adalah *game* / permainan. Definisi dari *game* adalah sebuah aktivitas bermain yang dilakukan dalam konteks realita yang semu, yang mana patisipan mencoba untuk meraih sebuah tujuan, dengan berlaku mengikuti aturan-aturan [1]. Melalui sifat kompetitif yang dihasilkan oleh *game*, *game* mampu menumbuhkan daya kreasi tentang strategi yang diperlukan untuk mengalahkan tim / pemain lainnya.

Game memiliki banyak jenis / *genre* yang saat ini berkembang, diantaranya adalah *game* strategi. *Game* strategi diminati karena mampu memberi tantangan dalam memecahkan suatu masalah dengan mengatur siasat / strategi untuk mampu mengalahkan pemain lain ataupun kecerdasan buatan dari *game* itu. *Game turn-based strategy* berkembang cukup pesat saat ini, karena *game* tipe mampu memberikan tantangan bagaimana seorang pemain mampu mengalahkan pemain lain menggunakan siasatnya masing-masing. *Game* seperti ini biasa diminati oleh orang yang suka dengan tantangan (kompetitif) atau menggemari strategi perang.

Setiap *game* strategi memiliki kecerdasan buatan / *Artificial Intellegence* sebagai lawan bermain dari pemain. Sebagai lawan bermain dari pemain tentunya kecerdasan buatan (AI) yang cerdas akan memberikan tantangan pada pemain. Namun pada beberapa *game* strategi, AI hanya memperhatikan *cost* pembuatan pasukan tanpa melihat pasukan musuh sehingga AI menjadi kurang maksimal dan kurang memberikan tantangan. Dengan demikian perlu dilakukan peningkatan pada AI *game* strategi. Sebagai peningkatan dari AI *game* strategi maka diterapkan algoritma genetik sebagai pemilihan pasukan yang dipilih oleh AI.

Algoritma genetika meniru proses dari evolusi pada alam, untuk menemukan solusi yang sesuai dari suatu masalah. Berdasarkan teori evolusi, hanya kromosom yang memiliki tingkat kecocokan (*fitness*) yang baik akan bertahan dan menghasilkan keturunan dan mewariskan kelebihannya kepada keturunannya dengan beberapa faktor genetik [4]. Fitur dari aplikasi game strategi penyerbuan di Batavia adalah menggunakan metode algoritma genetika pada pemilihan kombinasi pasukan yang dari AI aplikasi *game*. Sehingga dihasilkan AI yang mampu melakukan pemilihan kombinasi pasukan meliputi pasukan apa saja yang akan dibuat oleh AI yang sesuai dengan kondisi pasukan musuh.

2. LANDASAN TEORI

2.1 Penyerbuan di Batavia

Pada tahun 1621 Mataram mulai menjalin hubungan dengan VOC. Akan tetapi, VOC ternyata menolak membantu saat

Mataram menyerang Surabaya. Akibatnya, hubungan diplomatik kedua pihak putus.

Penyerangan pertama Mataram terjadi pada tanggal 25 Agustus dengan menyerang benteng Bommel. Di bawah pimpinan Letnan Jacob van der Plaetten berhasil menghalangi Mataram, setelah pertempuran yang dahsyat.

Kegagalan serangan pertama diantisipasi dengan cara mendirikan lumbung-lumbung beras tersembunyi di Karawang dan Cirebon. Namun pihak VOC yang menggunakan mata-mata berhasil menemukan dan memusnahkan semuanya.

Walaupun kembali mengalami kekalahan, serangan kedua Sultan Agung ini berhasil membendung dan mengotori Sungai Ciliwung, yang mengakibatkan timbulnya wabah penyakit kolera melanda Batavia. Gubernur jenderal VOC yaitu Jan Pieterszoon Coen meninggal menjadi korban wabah tersebut [2].

2.2 *Strategy Game*

Strategy game [6] adalah salah satu *genre* dari *game* yang menggunakan pemikiran dan perencanaan untuk mencapai kemenangan. Seorang pemain harus merencanakan terlebih dahulu sebelum melakukan tindakan melawan musuh, dan mengurangi pasukan musuh biasanya sebagai sebuah tujuan. Kemenangan diraih melalui perencanaan yang matang, dan pengambilan kesempatan yang tepat. Pada kebanyakan *strategy game*, pemain melihat dalam pandangan dari atas dan dapat melihat seluruh bagian map permainan dan secara tidak langsung mengontrol karakter dalam permainan yang berada di bawah komandonya.

2.3 *Genetic Algorithm*

Menurut Michael Negnevitsky (2011) algoritma genetika [5] adalah sebuah algoritma pencarian stokastik berdasarkan evolusi biologis. Prinsip dasar dari Algoritma Genetika adalah seleksi, persilangan, dan mutasi individu dengan tujuan menghasilkan generasi yang lebih baik dari sebelumnya. Berikut adalah langkah-langkah dari algoritma genetika:

1. Dibuat individual awal dengan cara merandom gen dari kromosom individu.
2. Dihitung nilai *fitness* dari setiap individu menggunakan fungsi *fitness*.
3. Dilakukan seleksi dengan metode mesin *roulette* sebagai berikut:
   - Dihitung total *fitness* dari semua individu.
   - Dihitung probabilitas setiap individu menggunakan Persamaan 1 sebagai berikut:

$$P_{individu} = \frac{Fitness_{individu}}{Fitness_{total}} \quad (1)$$

   Pada Persamaan 1, P individu mewakili peluang terpilihnya individu tersebut, dimana dipengaruhi oleh *fitness* individu yang merupakan *fitness* dari individu tersebut dan *fitness* total yang merupakan total dari seluruh *fitness* individu yang ada pada populasi.
4. Dihitung *range* terpilihnya individu berdasarkan probabilitas dimulai dari angka 1 hingga 100. Berikut contoh dari *range* terpilihnya individu dapat dilihat pada Tabel 1.

Tabel 1. Tabel probabilitas individu

Individu	Probabilitas	Range
1	0.1	1 – 10
2	0.25	11 – 35
3	0.4	36 – 75
4	0.15	76 – 90
5	0.1	91 – 100

5. Dilakukan pengacakan sebanyak N kali dalam *range* 1 – 100, angka yang didapat dari pengacakan menentukan individu yang terpilih berdasarkan *range* dari individu seperti pada Tabel 1.
   Contoh :
   N = 5
   Random = 37 → individu yang terpilih ke - 1 : 3
   Random = 90 → individu yang terpilih ke - 2 : 5
   Random = 10 → individu yang terpilih ke - 3 : 1
   Random = 74 → individu yang terpilih ke - 4 : 3
   Random = 23 → individu yang terpilih ke - 5 : 2
6. Didapatkan individu yang terpilih untuk masuk ke langkah berikutnya.
7. Dilakukan persilangan / *cross-over* antara 2 individu. Prinsip dari persilangan ini adalah melakukan operasi gen-gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *cross-over* dilakukan pada setiap individu dengan probabilitas *cross-over* yang ditentukan.
8. Dilakukan mutasi terhadap individu pada generasi yang baru. Proses mutasi terhadap individu memiliki probabilitas dan hasil mutasi didapat dari hasil random angka dengan range tertentu.
9. Dilakukan langkah 2 hingga 8 menggunakan generasi baru hasil mutasi. Langkah – langkah ini dilakukan terus menerus hingga mencapai hasil yang diinginkan.

2.4 *A* Pathfinding*

A pathfinding* adalah salah satu metode pencarian *heuristic* yang memberikan prioritas pada *node-node* yang seharusnya lebih baik daripada yang lain. Inti utama dari algoritma *A** adalah adanya *node* lokasi untuk dilakukan perhitungan *pathfinding*. *Node-node* itu dapat dibuat menggunakan *graph*, dengan *graph* sedikit untuk kecepatan, dan *graph* banyak untuk keakuratan. Untuk *game* yang menggunakan sistem *grid* seperti Civilization IV, metode *A** dapat diadopsi agar setiap kotak pada *grid* dianggap sebagai sebuah *node* sendiri [3].

3. DESAIN SISTEM

3.1 Perancangan Sistem Game

Seperti pada Gambar 2, *game* dimulai dengan setiap pemain memiliki 1 bangunan utama sebagai tempat membuat *unit*. Bangunan utama juga perlu dilindungi karena apabila bangunan utama milik salah satu pemain hancur, level akan berakhir dan pemain dinyatakan kalah.

Pembuatan *unit* dilakukan saat giliran dari pemain. *Unit* yang dibuat akan muncul disekitar bangunan utama. Pembuatan *unit* tidak membutuhkan waktu / instan.

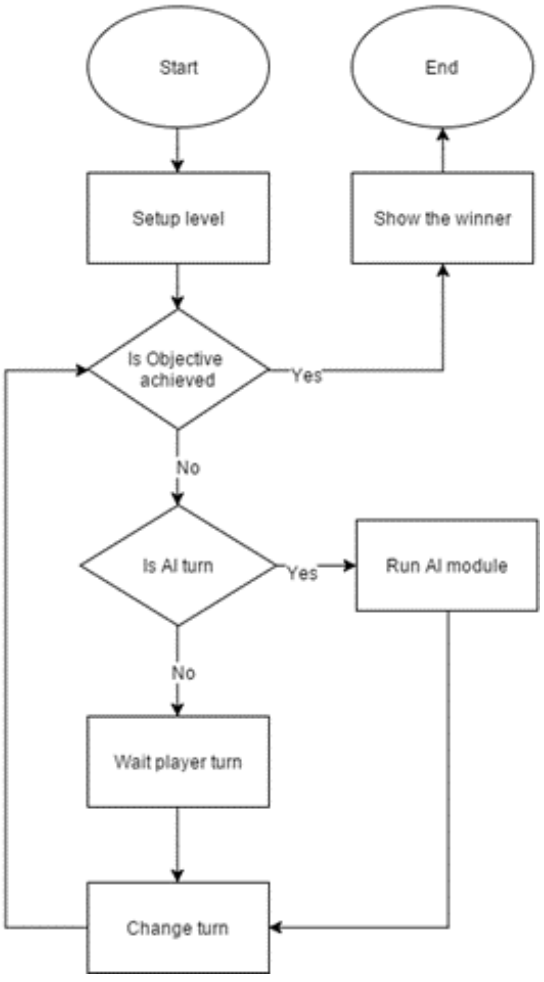
Kedua pemain mendapatkan jumlah uang yang sama pada awal. Pada setiap gilirannya pemain mendapatkan uang tambahan. Uang digunakan untuk membuat *unit* pada bangunan utama. Apabila setiap *unit* berhasil membunuh *unit* milik musuh, pemilik *unit* mendapatkan uang.

Sebuah *level* dimulai dengan dilakukan *loading map* dan memanggil modul awal untuk melakukan setting awal untuk kedua pemain dan *level*. Modul awal ini mencakup *setting*

sistem *grid* sesuai dengan map, penempatan bangunan utama dan beberapa variabel seperti giliran, dan lain-lain.

Setelah *loading map* dan *setting map*, *game* dimulai dengan giliran pemain pertama. Jika pemain melakukan seleksi pada sebuah *unit* maka *user interface* (UI) akan menampilkan statistik dari *unit* yang diseleksi. Pemain dapat melakukan 2 macam action pada *unit* yang diseleksi, *move* dan *attack*. Pergerakan *unit* akan diatur menggunakan *A* Pathfinding*. Pemain hanya dapat melakukan 2 macam *action* tersebut jika *unit* tersebut milik pemain yang menyeleksi. Setiap *unit* hanya dapat melakukan *action move* dan *attack* sekali setiap giliran. Setelah *unit* melakukan penyerangan akan dilakukan pengecekan pada bangunan utama kedua pemain. Apabila bangunan utama salah satu pemain hancur maka pemain tersebut kalah dan pemain lain menjadi pemenang.

Pemain dapat mengakhiri gilirannya dengan menekan tombol "endturn" pada UI. Setelah seorang pemain mengakhiri gilirannya, maka giliran akan diganti dengan pemain lawan. Apabila pemain tersebut adalah AI, maka modul AI dijalankan.



Gambar 2. *Flowchart* sistem *game* Battle of Batavia

3.2 Implementasi Algoritma Genetika

Pada AI pada game Battle of Batavia secara umum terdapat 2 bagian, antara lain, bagian pembuatan *unit*, dan bagian pergerakan *unit*. Sehingga terdapat 2 metode algoritma genetika dalam sebuah AI, yaitu implementasi pada perencanaan *deploy* pasukan, dan perencanaan pergerakan pasukan.

3.2.1 Implementasi pada Deploy Pasukan

Pada perencanaan *deploy* pasukan, metode algoritma genetika digunakan untuk menentukan pasukan yang akan dibuat pada gilirannya. Hasil pasukan yang diinginkan adalah set pasukan yang mampu mengatasi pasukan musuh. Untuk mencapai hasil pasukan yang diinginkan terdapat beberapa kriteria yang harus terpenuhi, yaitu:

- Harga total untuk membangun pasukan.
- Potensi tingkat kerusakan (*damage*) yang dapat dihasilkan oleh pasukan terhadap pasukan musuh.
- Total health pasukan yang akan dibuat.
- Persentase uang yang disimpan pada pembuatan pasukan.

Selain kriteria-kriteria diatas sebagai pertimbangan untuk urutan pembuatan pasukan digunakan variabel berupa prioritas, dimana prioritas mendapatkan nilai dari potensi tingkat kerusakan yang dihasilkan oleh pasukan yang akan dibuat tersebut.

Dengan demikian dihasilkan desain individu untuk algoritma genetika *deploy* pasukan dapat dilihat pada Gambar 3.

Jum. I	Prio. I	Jum. K	Prio. K	Jum. M	Prio. M	Jum. A	Prio. A

Gambar 3. Desain individu untuk pada *deploy* pasukan

Keterangan:
Jum. I: Jumlah infantri.
Prio. I: Prioritas infantri
Jum. K: Jumlah kavaleri
Prio. K: Prioritas kavaleri
Jum. A: Jumlah artileri
Prio. A: Prioritas artileri

Setiap tipe pasukan memiliki jumlah dan prioritas pada setiap kromosomnya. Jumlah mewakili jumlah tipe pasukan yang akan dibuat. Prioritas berfungsi untuk memilih tipe pasukan yang dibuat terlebih dahulu.

Sebuah individu memuat kelima jumlah tipe pasukan, total kerusakan (*damage*), dan harga total (*total cost*). Kedua desain individu tersebut memiliki kemiripan struktur kromosomnya karena kedua pihak memiliki tipe *unit* yang identik sehingga susunan kromosomnya mirip. Langkah-langkah dari algoritma genetika untuk *deploy* pasukan:

1. Dilakukan *generate* seluruh individu generasi pertama dengan merandom angka sebagai isi dari kromosom. Nilai yang di-generate hanya pada kromosom jumlah infantry/prajurit milisi hingga jumlah kapal perang. Dihitung pula *total cost* dari individu yang di-*generate*, dengan menjumlahkan *training cost* setiap *unit* dan dikalikan dengan jumlah unit tersebut.
2. Dilakukan perhitungan total *damage* pada setiap individu yang di-*generate*. *Attack* setiap *unit* dipertimbangkan mengenai relevansi serangan dengan pertahanan milik musuh. Contoh kasus:
   Pemain memiliki 2 Prajurit Milisi dengan tipe armor *Infantry*, dan hasil individu menghasilkan 1 *Infantry* dengan tipe serangan *Melee* dan 1 *Musketeer* dengan tipe serangan *Bullet*. Total *damage* dihasilkan berdasarkan kerusakan terbesar yang dihasilkan oleh unit yang dibuat terhadap satu *unit* musuh. Dalam kasus ini total *damage* yang dihasilkan sebagai berikut:
   Total Damage = Attack Infantry + 1.5 * Attack Musketeer.

Attack dari Musketeer bertambah sebanyak 0.5 kali dari *attack* biasa dikarenakan *attack* dengan tipe Bullet memiliki kelebihan terhadap tipe pertahanan Infantry.

3. Dilakukan perhitungan total health pasukan dengan menjumlahkan *max health point* (HP) setiap unit dan dikalikan dengan jumlah unit.
4. Dilakukan perhitungan persentase uang yang disimpan dengan menghitung sisa dari uang apabila individu tersebut dijalankan dibagi dengan uang yang dimiliki. Hasil dari persentase dibandingkan dengan persentase yang diinginkan, untuk mencegah pembuatan unit secara berlebihan ataupun kurangnya pembuatan unit. Contoh kasus:
   Individu Deploy : 1 Infantry 1 Musketeer
   Uang yang dimiliki : 80
   Total Cost : 10 + 25 = 35
   Persentase uang yang disimpan: 1 - 35/80 = 0.5625
   Persentase yang diinginkan: 0.2
   PercentageSavedMoney : Abs(0.5625 – 0.2) = 0.3625
5. Dilakukan perhitungan *fitness* dari setiap individu *deploy* pasukan dengan menggunakan formula *fitness* dapat dilihat pada Persamaan 2.

$$Fitness = \frac{5 \times TotalDamage + TotalHP}{1 + 2.5 \times TotalCost \times PercentageSavedMoney} \quad (2)$$

   - *TotalDamage* memiliki konstanta sebesar 5 dikarenakan apabila semakin besar total nilai *damage* yang dihasilkan dari pasukan, semakin besar kemungkinan pasukan musuh yang mampu dieliminasi.
   - *TotalHP* mengindikasikan ketahanan dari seluruh pasukan pada suatu individu. Semakin tinggi total HP dari individu, semakin kuat individu tersebut untuk dapat bertahan dalam serangan.
   - *TotalCost* mengindikasikan seberapa mahal pembuatan seluruh pasukan pada individu *deploy*. *TotalCost* dijadikan sebagai faktor yang mampu mereduksi nilai dari *fitness*, dikarenakan semakin tinggi *TotalCost* dari pasukan maka semakin tinggi pula penggunaan uang dari AI. Dimana tujuan dari AI pada *deploy* pasukan adalah menemukan kombinasi dari tipe pasukan dengan penggunaan jumlah uang yang terendah yang mampu menghasilkan tingkat kerusakan yang tinggi.
   - *PercentageSavedMoney* ditambahkan pada formula *fitness* untuk membatasi pembelian yang dilakukan oleh AI sehingga uang yang disimpan dapat digunakan pada giliran berikutnya. Apabila jumlah pasukan pemain lebih banyak daripada jumlah pasukan AI maka nilai dari *PercentageSavedMoney* diubah menjadi 0. Hal ini mencegah AI melakukan penyimpanan uang pada saat penguasaan dari *map* berkurang.

   Apabila hasil dari *fitness* untuk individu *deploy* pasukan semakin tinggi, semakin bagus kualitas individu tersebut.
6. Dilakukan *elitism* pada generasi berikutnya dengan menambahkan sejumlah individu yang memiliki *fitness* tertinggi.
7. Dilakukan seleksi pada kromosom-kromosom menggunakan metode *roulette wheel*. Metode *roulette wheel* menggunakan *fitness* sebagai probabilitas sebuah individu untuk masuk pada generasi berikutnya. Individu yang merupakan bagian dari *elitism* langsung lolos dari seleksi. Berikut formula dari seleksi menggunakan metode *roulette wheel*:

$$Probabilitas_{Individu} = \frac{Fitness_{Individu}}{Fitness_{Total}} \quad (3)$$

   Pada Persamaan 3, probabilitas individu mewakili peluang terpilihnya individu tersebut, dimana dipengaruhi oleh *fitness* individu yang merupakan *fitness* dari individu tersebut dan *fitness* total yang merupakan total dari seluruh fitness individu yang ada pada populasi.
8. Dari hasil probabilitas individu, dapat dihasilkan probabilitas terpilihnya setiap individu untuk menjadi individu pada generasi berikutnya.
9. Dilakukan persilangan / cross-over antara 2 individu secara acak sesuai dengan probabilitas individu. Setelah penukaran kromosom nilai total *damage*, total HP, total cost dan persentase uang yang disimpan perlu dilakukan perhitungan ulang karena mungkin berubah pada saat penukaran.
10. Dilakukan mutasi pada beberapa individu dengan probabilitas tertentu. Mutasi dilakukan pada kromosom jumlah unit yang akan dibuat, dengan merandom ulang nilai jumlah unit dari kromosom.
11. Langkah 2 hingga 10 dilakukan berulang ulang-ulang dengan input individu generasi sebelumnya. hingga mencapai generasi yang diinginkan.
    Nilai dari generasi yang diinginkan pada *deploy* pasukan sebesar 50 generasi.
12. Dari hasil generasi yang terakhir diambil individu yang terbaik dan digunakan untuk langkah AI melakukan *deploy* pasukan.

4. PENGUJIAN SISTEM

4.1 Pengujian Sistem Algoritma Genetika

AI Algoritma Genetik dari *game* Battle of Batavia akan dibandingkan dengan metode AI random yang telah dibuat dalam *game* Battle of Batavia dan hasil pengujian oleh *tester* dari *game* Battle of Batavia. Perbandingan yang dibuat meliputi beberapa aspek, antara lain:

- Perbandingan dalam segi keuangan, meliputi uang yang digunakan, dan uang yang dihasilkan.
- Perbandingan dalam segi tingkat kerusakan yang dihasilkan dari masing-masing metode, meliputi tingkat kerusakan yang diterima dan tingkat kerusakan yang dihasilkan.
- Perbandingan dalam segi *unit* / pasukan, meliputi *unit* yang dibuat dan *unit* yang mati.

Perbandingan dalam segi giliran yang diperlukan untuk mampu memenangkan permainan atau mengalahkan AI.

4.1.1 Pengujian Sistem AI terhadap Pasukan dengan Tingkat Kerusakan Rendah hingga Sedang

Subbab ini digunakan untuk menguji sistem AI algoritma genetika pada pembuatan pasukan dan pergerakan pasukan dalam kondisi melawan pasukan dengan tingkat kerusakan sedang hingga rendah. Dalam hal ini, yang dimaksud dengan pasukan dengan tingkat kerusakan sedang dan rendah adalah tipe kavaleri dan prajurit milisi. Pengujian dilakukan secara langsung pada *game Battle of Batavia*.

Pengujian dilakukan dengan menggerakkan pembuatan pasukan pada pihak kesultanan Mataram diatur dengan *rule*, jumlah uang awal yang dimiliki kedua pihak sebesar 75, pada setiap gilirannya pemain dan AI akan menerima uang sebesar 20. Pembuatan pasukan ditentukan hanya pasukan dengan tipe kavaleri dan prajurit milisi. Pihak kesultanan Mataram menggunakan seluruh uang yang dimiliki pada pembuatan pasukan dengan tipe kavaleri pada setiap gilirannya.

Sedangkan untuk pergerakan pasukan pada pihak kesultanan Mataram diatur dengan *rule*, setiap pasukan yang dimiliki kesultanan Mataram akan bergerak ke arah pasukan AI yang terdekat dan melakukan serangan pada pasukan tersebut.

Tabel 2. Tabel hasil individu terpilih pada pengujian melawan pasukan dengan tingkat kerusakan rendah hingga sedang

Turn	Hasil Individu pada *Deploy* Pasukan
2	Inf: 1 Cav: 1 Mus: 1 Art: 1 Total Damage 57 Total HP 130 Total Cost 90 Percent Save: -0.2 Fitness: 700
6	Inf: 1 Cav: 0 Mus: 0 Art: 1 Total Damage 20 Total HP 65 Total Cost 40 Percent Save: 0.09090909 Fitness: 138.8095
10	Inf: 1 Cav: 0 Mus: 0 Art: 1 Total Damage 25 Total HP 65 Total Cost 40 Percent Save: 0.1111111 Fitness: 149.2105
14	Inf: 0 Cav: 0 Mus: 1 Art: 0 Total Damage 22 Total HP 30 Total Cost 25 Percent Save: -0.1363636 Fitness: 15.85014

Tabel 3. Pasukan yang dibuat pada pengujian pada pasukan dengan tingkat kerusakan rendah hingga sedang

Turn	AI Genetic Algorithm Unit Created	AI Random Unit Created
2	Musketeer, Cavalry, Infantry	Artilery
4	Musketeer, Infantry	Artilery, Infantry, Infantry
6	Artilery, Infantry	Artileri
8	Musketeer, Infantry	Infantry, Infantry, Infantry, Infantry
10	Artilery, Infantry	Infantry, Infantry
12	Musketeer, Infantry	Infantry, Infantry
14	-	Infantry, Infantry

Tabel 2 menunjukkan hasil individu yang terpilih untuk setiap giliran. Hasil ini didapatkan dengan menjalankan metode AI algoritma genetika pada setiap giliran dengan formula *fitness* untuk *deploy* pasukan dan *movement* pasukan yang sudah dibuat pada bab sebelumnya.

Seperti yang dapat diperhatikan dari Tabel 3, pasukan yang dibuat pada pengujian pada pasukan dengan tingkat kerusakan rendah hingga sedang, AI algoritma genetika melakukan *deploy* pasukan sebagian besar dengan tipe Infantry dimana tipe pasukan ini memiliki tipe serangan yang normal terhadap pasukan dengan tipe *armor Infantry*. Tipe pasukan ini memiliki *training cost* yang terendah. Selain tipe pasukan Infantry, hasil *deploy* pasukan menghasilkan tipe pasukan secara bergantian antara tipe pasukan Musketeer dan Artileri. Tipe pasukan Musketeer memiliki tipe serangan *Bullet* yang merupakan kelemahan dari tipe armor *Infantry*, sehingga pada perhitungan kerusakan yang dihasilkan nilai serangan dari Musketeer mendapatkan tambahan kerusakan pada pasukan musuh yang bertipe Cavalry dimana pasukan ini memiliki tipe *armor Infantry.* Sedangkan pada tipe pasukan Artileri memiliki tipe serangan *Explosive,* yang menghasilkan kerusakan yang lebih rendah terhadap pasukan Cavalry daripada tipe pasukan Musketeer. Namun, bangunan utama dari musuh memiliki tipe pertahanan *Fortified* yang merupakan kelebihan dari tipe serangan *Explosive.*

Tabel 4. Hasil *test case* terhadap pengujian AI pada pasukan dengan tingkat kerusakan rendah hingga sedang

Aspek	AI Random		AI Genetic Algorithm		Kenaikan*	
	Mataram	VOC	Mataram	VOC	Mataram	VOC
Unit Created	13	19	8	13	-5	-6
Unit Loss	0	17	8	3	8	-14
Money Used	265	250	200	245	-65	-5
Money Earned	810	330	319	724	-491	394
Total Damage Dealt	778	156	134	525	-644	369
Total Damage Received	156	778	525	134	369	-644
Total Turn	14		14		0	
Winner	Mataram		VOC		VOC	

*Kenaikan didapatkan dengan membandingkan hasil dari kedua AI. Perbandingan dilakukan dengan mencari selisih dari AI *Random* dan AI Algoritma Genetika.

Pada Tabel 4 dapat dilihat perbandingkan hasil dari pengujian dari kedua metode AI, yaitu *random* dan algoritma genetika pada pasukan dengan tingkat kerusakan rendah hingga sedang. Pada aspek *Unit Created* terlihat bahwa AI algoritma genetika mampu menekan jumlah pasukan yang dibuat oleh Mataram dengan jumlah pasukan yang lebih sedikit daripada pasukan yang AI *random* buat. Pada aspek *Unit Loss,* AI *random* tidak mampu mengalahkan satupun pasukan pada pihak VOC sedangkan pada AI algoritma genetika mampu mengalahkan 8 pasukan milik Mataram dengan jumlah pasukan yang mati jauh lebih sedikit daripada AI Random.

Pada aspek *Money Used,* Mataram menggunakan uang lebih sedikit pada saat melawan AI algoritma genetika daripada AI *random,* namun AI algoritma genetika menggunakan uang lebih

sedikit daripada AI *random.* Pada aspek *Money Earned* terlihat perbedaan yang mencolok, uang yang dihasilkan oleh Mataram sewaktu melawan AI algoritma genetika jauh lebih sedikit dibandingkan pada saat melawan AI *random.* Begitu pula pada uang yang dihasilkan oleh VOC jauh lebih banyak saat digunakan AI algoritma genetika daripada AI *random.*
Pada aspek *Damage,* tingkat kerusakan yang dihasilkan oleh Mataram jauh lebih kecil dan tingkat kerusakan yang dihasilkan oleh VOC jauh lebih tinggi ketika AI algoritma genetika diterapkan daripada metode *random.*

4.1.2 Pengujian Sistem AI terhadap Pasukan dengan Tingkat Kerusakan Tinggi

Subbab ini digunakan untuk menguji sistem AI algoritma genetika pada pembuatan pasukan dan pergerakan pasukan dalam kondisi melawan pasukan dengan tingkat kerusakan tinggi. Dalam hal ini, yang dimaksud dengan pasukan dengan tingkat kerusakan tinggi adalah tipe musketeer dan artileri.

Pengujian dilakukan dengan menggerakkan pembuatan pasukan pada pihak kesultanan Mataram diatur dengan *rule*, jumlah uang awal yang dimiliki kedua pihak sebesar 75, pada setiap gilirannya pemain dan AI akan menerima uang sebesar 20. Pembuatan pasukan ditentukan hanya pasukan dengan tipe kavaleri dan prajurit milisi. Pihak kesultanan Mataram menggunakan seluruh uang yang dimiliki pada pembuatan pasukan dengan tipe musketeer dan artileri pada setiap gilirannya.

Sedangkan untuk pergerakan pasukan pada pihak kesultanan Mataram diatur dengan *rule*, setiap pasukan yang dimiliki kesultanan Mataram akan bergerak ke arah pasukan AI yang terdekat dan melakukan serangan pada pasukan tersebut.

Tabel 5. Hasil *test case* terhadap pengujian AI pada pasukan dengan tingkat kerusakan tinggi

Aspek	AI Random (normalisasi)		AI Genetic Algorithm		Kenaikan	
	Mataram	VOC	Mataram	VOC	Mataram	VOC
Unit Created	22 (9.17)	23 (9.58)	7	14	-2.17	4.42
Unit Loss	14 (5.84)	24 (10)	8	2	2.16	-8
Money Used	645 (268.75)	610 (254.17)	210	300	-65	-5
Money Earned	1157 (482.08)	691 (287.92)	177	895	-365.08	607.08
Total Damage Dealt	1054 (439.17)	622 (259.17)	126	550	-313.17	290.83
Total Damage Received	622 (259.17)	1054 (439.17)	550	126	290.83	-313.17
Total Turn	36 (15)		15		0	
Winner	Mataram		VOC		VOC	

Pada Tabel 5 dibandingkan hasil dari pengujian dari kedua metode AI, yaitu *random* dan algoritma genetika pada pasukan dengan tingkat kerusakan tinggi. Pada jumlah giliran untuk AI *random* dan algoritma genetika tidak sama maka dilakukan normalisasi terhadap nilai – nilai aspek milik AI *random* untuk mempermudah pengecekan.

Pada *test case* ini, AI *random* mampu bertahan hingga giliran ke 36 namun kembali dikarenakan kurangnya pengaturan pada pembuatan pasukan dan pergerakan pasukan, AI *random* tidak dapat menilai sebuah keputusan, sehingga AI random mengalami kekalahan. Lain halnya dengan AI algoritma genetika yang mampu menghasilkan hasil yang jauh lebih baik dari AI dengan metode *random.*

5. KESIMPULAN DAN SARAN

Selama perencanaan, analisa dan desain, pembuatan program sampai dengan penulisan buku terdapat beberapa kesimpulan yang diperoleh. Kesimpulan yang telah diperoleh antara lain:

- Penggunaan algoritma genetika sebagai AI lebih cocok jika digunakan pada *game* ber-*genre strategy turn-based* dikarenakan waktu untuk melakukan proses pembuatan generasi yang memakan waktu cukup lama.
- Apabila algoritma genetika hendak diterapkan sebagai AI dari suatu *game.* Diperlukan pengujicobaan secara berkala untuk menemukan pengaturan dari *game* yang seimbang, karena hasil dari algoritma genetika sangat dipengaruhi oleh statistik dari game.
- Penggunaan algoritma genetika sebagai AI terbukti lebih baik dari pada metode *random*.

Selain kesimpulan diatas, terdapat beberapa saran yang dapat digunakan dan berguna bagi pembaca untuk proses pengembangan lebih lanjut, antara lain:

- Meningkatkan performa dari algoritma genetika pada AI.
- Menambahkan training dari jaringan saraf tiruan untuk melakukan menentukan formula *fitness* dari algoritma genetika.
- Perumusan dari formula *fitness* perlu diobservasi lebih lagi agar dapat menyelesaikan permasalah lain.

6. DAFTAR PUSTAKA

[1] Adams, Ernest. 2014. Fundamentals of Game Design. Berkeley: New Riders.

[2] Bertrand, Romain. 2011. L'Histoire à parts égales. Récits d'une rencontre Orient-Occident (XVIe-XVIIe siècles). Paris: Histoire (H.C.).

[3] Cui, X., Shi, H. 2011. Direction Oriented Pathfinding in Video Games. Australia: Victoria University.

[4] Ismail, I.A., N.A.El Ramly, M.M.El_Kafrawy, M.M.Nasef. 2007. Game Theory Using Genetic Algorithms. London: WCE.

[5] Negnevitsky, Michael. 2011. Artificial Intellegence A Guide to Intelligent Systems. Kanada: Pearson Education Canada.

[6] Rollings, Andrew, Ernest Adams. 2013. Fundamentals of Game Design. Amerika: New Riders Publishing.