

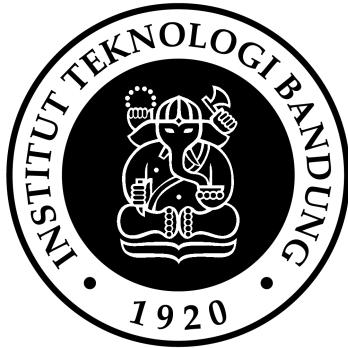
IF3170 Inteligensi Buatan

MINIMAX ALGORITHM AND ALPHA BETA PRUNING

IN ADJACENCY STRATEGY GAME

Laporan Tugas Besar I

Disusun untuk memenuhi tugas mata kuliah Intelligensi Buatan
pada Semester I (satu) Tahun Akademik 2023/2024.



Oleh

Fakhri Muhammad Mahendra	13521045
Chiquita Ahsanunnisa	13521129
Vanessa Rebecca Wiyono	13521151
Alisha Listya Wardhani	13521171

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

BAB I	
PENDAHULUAN	4
BAB II	
RANCANGAN DAN IMPLEMENTASI PROGRAM	5
2.1. Struktur Program	5
2.2. Kelas Action	5
2.3. Kelas State	6
2.4. Kelas Bot dan Turunannya	6
2.4.1. Kelas SABot	7
2.4.2. Kelas MMABot	8
2.4.3. Kelas GABot	8
2.4.4. Kelas GA2Bot	9
2.5. Kelas BotFactory	9
2.6. Perubahan dari Program Sebelumnya	9
2.6.1. Input Frame	10
2.6.2. Output Frame	10
BAB III	
ALGORITMA	11
3.1. Fungsi Objektif	11
3.2. Proses Pencarian dengan Algoritma Minimax Alpha Beta Pruning	12
3.2.1. Pemodelan Min-Max pada Tree	12
3.2.2. Mekanisme Pruning	13
3.2.3. Restriksi Kedalaman	13
3.2.4 Tahapan Proses	13
3.3. Proses Pencarian dengan Algoritma Local Search	16
3.4. Proses Pencarian dengan Genetic Algorithm (Alternatif 1)	19
3.4. Proses Pencarian dengan Genetic Algorithm (Alternatif 2)	22
3.4.1. Pembobotan Fungsi Objektif	22
3.4.2. Kromosom	22
3.4.3. Fungsi Fitness	22
3.4.4. Proses Seleksi	22
3.4.5. Proses Kawin Silang	23
3.4.6. Proses Mutasi	23
3.4.7. Hasil Akhir	23
BAB IV	
PENGUJIAN	24
4.1. Pertandingan antara Bot Minimax dan Manusia	24
4.2. Pertandingan antara Bot Local Search dan Manusia	26
4.3. Pertandingan antara Bot Minimax dan Bot Local Search	28
4.4. Pertandingan antara Bot Minimax dan Bot Genetic Algorithm Ver 1	30
4.5. Pertandingan antara Bot Local Search dan Bot Genetic Algorithm Ver 1	32
4.6. Pertandingan antara Bot Local Search dan Bot Genetic Algorithm Ver 2	33
4.7. Pertandingan antara Bot Minimax dan Bot Genetic Algorithm Ver 2	35

4.8. Ringkasan	37
BAB V	
SIMPULAN DAN SARAN	38
5.1. Simpulan	38
5.2. Saran	38
REFERENSI	39
LAMPIRAN	40
Repository Github	40

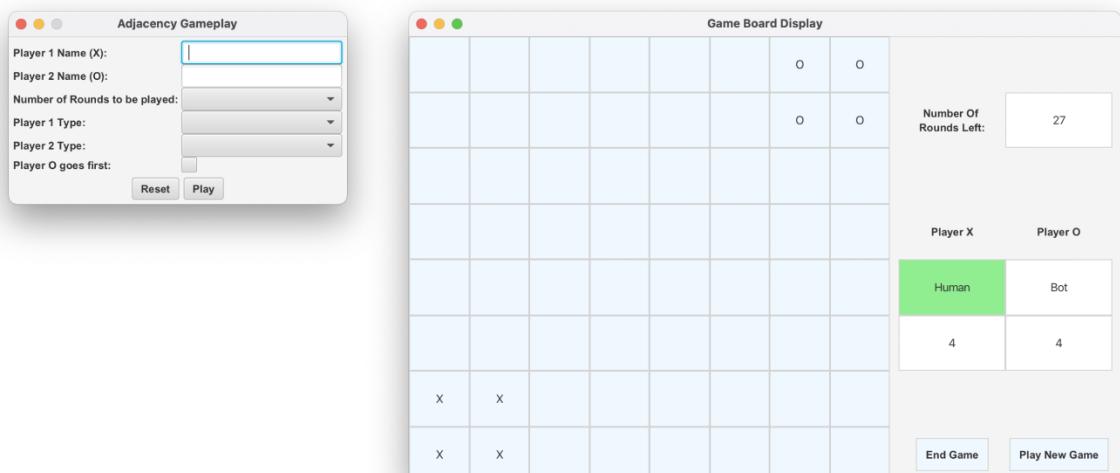
BAB I

PENDAHULUAN

Permainan Adjacency Strategy Game merupakan permainan pada papan berdimensi 8x8, dengan dua jenis pemain O dan X. Pemain menempatkan marka pada papan permainan. Tujuan yang ingin dicapai pada permainan ini adalah memaksimalkan jumlah bidak pemain dalam papan. Permainan diprogram dalam bahasa Java dan memakai framework JavaFX untuk menjalankan antarmuka. Struktur program ini dibuat berdasarkan source code pada pranala berikut.

<https://github.com/GAIB20/adversarial-adjacency-strategy-game>

Penulis mengubah tampilan awal program yang berfungsi untuk mengatur jalannya permainan. Pada konfigurasi awal menentukan nama dan tipe dari masing-masing pemain, jumlah ronde, dan pemain yang berjalan terlebih dahulu. Tipe dapat berupa Human, Simulated Annealing Bot, Minimax Bot, dan Genetic Algorithm Bot.



Gambar 1.1. Antarmuka Permainan

Dalam Tugas Besar 1 Inteligensi Buatan ini, dibuat algoritma untuk menyelesaikan permainan tersebut. Algoritma yang digunakan adalah *algoritma local search*, *adversarial search*, dan *genetic function*. Penulis memanfaatkan algoritma *simulated annealing* yang memanfaatkan random walk. Adversarial search diimplementasikan dengan algoritma Minimax dengan pemanfaatan alpha-beta pruning.

BAB II

RANCANGAN DAN IMPLEMENTASI PROGRAM

2.1. Struktur Program

Berikut merupakan struktur direktori dari program.

```
IF3170-Tubes-1-AI
```

```
    └── javafx-sdk
    └── out
    └── screenshots

    └── src
        ├── Action
        ├── Bot
        ├── BotFactory
        ├── GABot
        ├── InputFrame.fxml
        ├── InputFrameController
        ├── Main
        ├── MMABBot
        ├── OutputFrame.fxml
        ├── OutputFrameController
        ├── SABot
        └── State

    └── style.css
    └── .gitignore
    └── README.md
```

2.2. Kelas Action

Kelas Action merepresentasikan aksi yang dapat dilakukan oleh bot. Aksi yang dilakukan oleh bot yaitu memilih koordinat pada papan. Koordinat (i,j) merepresentasikan sel pada baris ke-i kolom ke-j, dengan $0 \leq i, j \leq 7$. Oleh karena itu, kelas Action memiliki atribut i dan j yang merepresentasikan koordinat tersebut.

Berikut merupakan atribut dan metode yang ada pada kelas Action.

Tabel 2.2.1. Kelas Action

Nama kelas: Action	
Atribut	
int i	Merepresenasiakan nomor baris [0..7]
int j	Merepresenasiakan nomor kolom [0..7]
Metode	
Action(int i, int j)	Konstruktor objek dari kelas Action

2.3. Kelas State

Sesuai namanya, kelas State merepresentasikan *state* pada papan. Berikut merupakan atribut dan metode yang ada pada kelas Action.

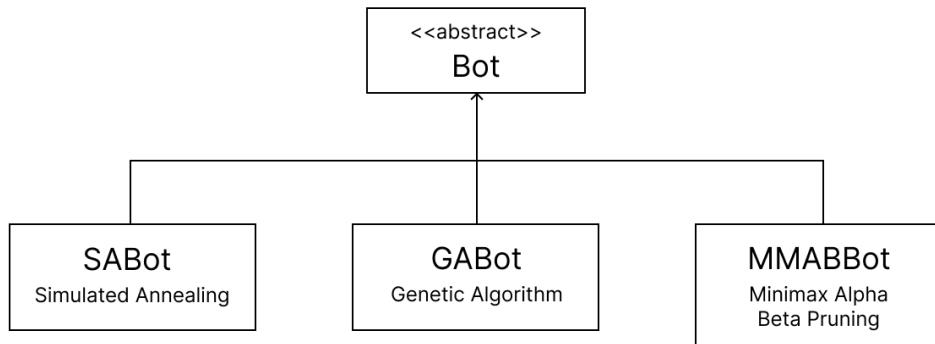
Tabel 2.3.1. Kelas State

Nama kelas: State	
Atribut	
int[][] mat	Kondisi papan saat ini dalam bentuk matriks <i>integer</i> . Elemen bisa bernilai -1, 0, atau 1. Elemen mat[i][j] bernilai 1 jika papan[i][j] bermakna sama dengan marka bot yang menginstansiasi <i>state</i> , bernilai 0 jika papan[i][j] kosong, dan bernilai -1 jika papan[i][j] bermakna sama dengan marka lawan dari bot yang menginstansiasi <i>state</i> .
boolean isOpponentTurn	Jika bernilai <i>true</i> berarti giliran tersebut dimiliki oleh lawan, jika tidak berarti sebaliknya. Lawan yang dimaksud adalah lawan dari bot yang menginstansiasi <i>state</i> tersebut.
Metode	
State(Button[][] buttons)	Konstruktor objek dari kelas State. Konstruktor ini merupakan konstruktor yang dipanggil oleh bot untuk menciptakan representasi <i>state</i> dari kondisi papan sekarang.
State(State prev, Action act)	Konstruktor objek dari kelas State. Konstruktor ini digunakan untuk men-generate <i>state</i> baru yang dihasilkan dari pemberlakuan Action act terhadap State prev. Konstruktor ini digunakan untuk membangkitkan suksesor dari State prev.
int calcValue()	Mengembalikan nilai fungsi objektif dari sebuah <i>state</i> . Nilai objektif ini juga relatif terhadap bot yang menginstansiasi <i>state</i> . Rule of thumb-nya adalah makin menguntungkan sebuah <i>state</i> bagi bot penginstansiasi, makin besar juga nilai objektifnya.
boolean isBoardFull()	Mengecek apakah seluruh sel pada papan telah diberi marka.
List<Action> getDefaultActions	Mengembalikan daftar aksi valid yang dapat dilakukan pada kondisi papan tersebut. Aksi yang valid adalah memilih pada sel yang masih kosong.

Sebagai catatan, *state* bersifat relatif terhadap bot yang menginstansiasi *state* tersebut.

2.4. Kelas Bot dan Turunannya

Kelas Bot adalah kelas abstrak yang merepresentasikan seluruh bot yang akan digunakan. Kelas ini akan diturunkan menjadi kelas-kelas bot yang menggunakan algoritma *simulated annealing* (kelas SABot), bot yang menggunakan *genetic algorithm* (kelas GABot), dan bot yang menggunakan algoritma *minimax alpha-beta pruning* (kelas MMABot). Berikut merupakan diagram kelas yang menunjukkan *inheritance* pada kelas Bot.



Gambar 2.4.1. Diagram Kelas Bot dan Turunannya

Adapun desain kelas-kelas tersebut mengaplikasikan *inheritance* agar modifikasi program yang telah ada tidak sulit karena dapat memanfaatkan *polymorphism*.

Berikut adalah metode yang ada pada kelas Bot.

Tabel 2.4.1. Kelas Abstrak Bot

Nama kelas: Bot	
Metode	
abstract Action move(State s)	Mengembalikan aksi paling optimal (lokal atau global) pada State s sesuai dengan algoritma yang diinginkan. Metode abstrak ini akan diimplementasikan oleh kelas anak sesuai dengan algoritmanya.

2.4.1. Kelas SABot

Kelas SABot merupakan kelas turunan dari Bot dan merepresentasikan bot yang menggunakan algoritma Simulated Annealing.

Tabel 2.4.1.1. Kelas SABot

Nama kelas: State	
Atribut	
int initTemp	Merepresentasikan temperatur awal algoritma. Temperatur dimulai dari nilai yang tinggi sehingga lebih memperbolehkan perubahan state. Salah satu heuristik yang digunakan dalam tugas ini adalah 100.
int stopTemp	Merepresentasikan temperatur akhir algoritma. Jika temperatur sama dengan temperatur akhir, maka pemrosesan algoritma akan berhenti.
Metode	
int schedule(int currTemp, long startTime)	Metode ini digunakan sebagai cooling schedule temperatur pada algoritma. Metode ini menerima temperatur awal dan waktu. Nilai yang dihasilkan akan berkurang secara eksponensial.
Action getRandomAction(State s)	Metode ini mengembalikan aksi yang dipilih secara acak dari daftar aksi

	yang mungkin untuk dilakukan pada permainan.
Action move (State s)	Metode ini merupakan algoritma simulated annealing dan mengembalikan aksi yang harus dilakukan bot pada permainan.

2.4.2. Kelas MMABot

Kelas MMABot merupakan kelas turunan dari Bot yang merepresentasikan bot yang menggunakan algoritma MiniMax Alpha Beta Pruning (MMAB).

Tabel 2.4.1.1. Kelas SABot

Nama kelas: State	
Atribut	
int maxTreeDepth	Merepresentasikan jumlah kedalaman pohon yang ditraversal pada algoritma MiniMax.
Metode	
Action move (State s)	Metode ini merupakan algoritma MiniMax dengan alpha beta pruning dan mengembalikan aksi yang harus dilakukan bot pada permainan.
int minValue (State s)	Metode ini mengembalikan nilai minimum yang mungkin diambil dari State anak dari s.
int maxValue (State s)	Metode ini mengembalikan nilai maksimum yang mungkin diambil dari state anak dari s.

2.4.3. Kelas GABot

Kelas GABot merupakan kelas turunan dari Bot yang merepresentasikan bot yang menggunakan algoritma Genetic.

Tabel 2.4.1.1. Kelas SABot

Nama kelas: State	
Atribut	
int NUM_PARENTS	Merepresentasikan jumlah kromosom yang dipilih menjadi populasi awal (<i>parents</i>) dalam algoritma.
Metode	
Action getRandomAction (State s)	Metode ini mengembalikan aksi yang dipilih secara acak dari daftar aksi yang mungkin untuk dilakukan pada permainan.
double fitnessValue (Action a, State s)	Metode ini mengembalikan nilai fitness dari setiap kromosom yang menjadi <i>parents</i> dalam algoritma.
Action crossoverParents (Action x, Action y, State s)	Metode ini mengembalikan kromosom hasil <i>crossover</i> dari 2 parents Actionx dan Action y.
Action move (State s)	Metode ini merupakan perhitungan algoritma Genetic, yang mengembalikan sebuah aksi yang dapat dilakukan pada papan permainan.

2.4.4. Kelas GA2Bot

Kelas GA2Bot merupakan kelas turunan dari Bot yang merepresentasikan bot yang menggunakan algoritma Genetic.

Tabel 2.4.4.1. Kelas GA2Bot

Nama kelas: State	
Atribut	
int iteration	Merepresentasikan iterasi yang dilakukan
int[][] weightTable	Merepresentasikan weight table yang digunakan pada algoritma. Weight table disini berisi nilai pembobotan yang akan dipakai untuk menghitung fungsi objektif yang berbeda. Weight Table ini juga bisa dianalogikan sebagai kromosom.
Metode	
GA2Bot(int iteration)	Konstruktor dari kelas GA2Bot dengan parameter banyaknya iterasi
void initiateWeightTable()	Mengkonstruksikan weight table yang akan digunakan pada algoritma.
void chooseWeight()	Metode yang merupakan seleksi untuk dijadikan sebuah populasi pemilihan
int chooseWeightFromFitness(int[] fitness)	Metode untuk menyeleksi berdasarkan nilai fitness yang dimiliki. Nilai fitness pada metode ini diubah menjadi persentase probabilitas diambilnya kromosom untuk dijadikan parents.
void crossOver()	Metode yang melakukan crossover pada kedua parents.
void mutate()	Metode yang melakukan mutasi pada kromosom.
void run()	Metode yang melakukan algoritma genetic.

2.5. Kelas BotFactory

Kelas BotFactory merupakan kelas *factory* yang dapat membuat tipe yang berbeda dari Bot berdasarkan mode yang disediakan. Jika mode dimulai dari “S”, metode akan mengembalikan objek SABot. Jika mode dimulai dari “G”, metode akan mengembalikan objek GABot. Jika karakter lain diterima, metode akan mengembalikan objek *default* yaitu MMABBot.

Tabel 2.5.1. Kelas BotFactory

Nama kelas: BotFactory	
Metode	
Bot createBot(String mode)	Metode untuk mengembalikan bot dengan algoritma tertentu, berdasarkan string mode.

2.6. Perubahan dari Program Sebelumnya

Untuk keperluan pemilihan algoritma bot dan pengetesan permainan antara bot dengan bot lain, perlu dilakukan perubahan pada mekanisme permainan. Namun perlu ditekankan bahwa perubahan ini sama sekali tidak mengubah aturan dan mekanisme *error* permainan.

2.6.1. Input Frame

Pada Input Frame, ditambahkan menu pemilihan pemain pertama (pemain bermarka X), pemilihan pemain kedua (pemain bermarka O). Pemain pertama (pemain bermarka X) dapat berupa manusia, *simulated annealing* bot, *genetic algorithm* bot, atau *minimax alpha-beta pruning* bot. Pemain kedua (pemain bermarka O) dapat berupa *simulated annealing* bot, *genetic algorithm* bot, atau *minimax alpha-beta pruning* bot. Pemain kedua tidak dapat berupa manusia karena program awal selalu menjadikan manusia sebagai pemain pertama. Selain itu, penulis juga mengubah label “Bot goes first:” menjadi “Player 2 goes first:” karena lebih masuk akal pada konteks permainan bot lawan bot.

2.6.2. Output Frame

Pada awalnya, Output Frame hanya mendukung permainan manusia lawan bot. Oleh karena itu, dilakukan perubahan pada Output Frame yaitu menambahkan dan menyesuaikan permainan bot lawan bot. Secara umum, perubahan yang dilakukan yaitu dengan menyimpan penanda apakah permainan yang dilakukan adalah manusia lawan bot atau bot lawan bot. Lalu, logika aksi pemilihan permainan disesuaikan dengan penanda tersebut.

BAB III

ALGORITMA

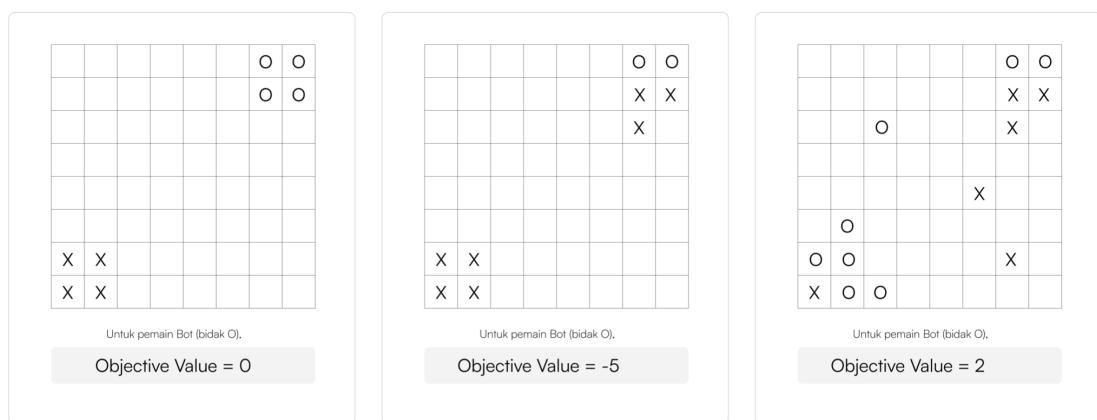
3.1. Fungsi Objektif

Pada permasalahan ini, *objective function* merupakan fungsi yang digunakan untuk mengukur seberapa baik penempatan bidak (dalam hal ini O) pada papan permainan. Tujuan dari permainan adalah memiliki jumlah bidak terbanyak pada papan permainan, maka objective function dapat dirumuskan sebagai berikut.

Objective function = jumlah bidak sendiri pada papan permainan dikurang dengan jumlah bidak lawan pada papan permainan.

Fungsi yang digunakan adalah “Maximize Objective Function”, yang memaksimalisasi jumlah bidak sendiri dikurang bidak lawan pada papan permainan. Alasan dipilihnya fungsi selisih antara bidak sendiri dengan bidak lawan karena jika masing-masing variabel tersebut berdiri sendiri sebagai fungsi objektif, maka akan menimbulkan masalah.

Berikut merupakan contoh hasil fungsi objektif pada permainan.



Gambar 3.1.1. Ilustrasi Fungsi Objektif

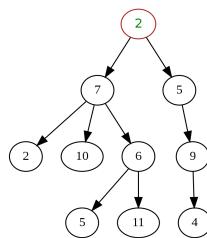
Namun pada genetic algorithm (versi 2) terdapat perubahan pada genetic function. Genetic function yang digunakan dapat ditentukan secara unik dari senarai lima bilangan bulat a_0, a_1, \dots, a_4 . Fungsi objektif ini bisa direpresentasikan sebagai fungsi matematika berikut

Anggap x_n menyatakan banyak petak yang miliki sendiri yang punya adjacent square sebanyak n . Sementara y_n adalah banyak petak milik lawan yang punya adjacent square sebanyak n . Maka

$$\text{objective}(a_0, a_1, a_2, a_3, a_4, \text{board}) = \sum_{i=0}^4 a_i * x_i - \sum_{i=0}^4 a_i * y_i .$$

3.2. Proses Pencarian dengan Algoritma Minimax Alpha Beta Pruning

Proses pencarian dilakukan dengan memodelkan pencarian sebagai suatu struktur data tree. Node menyatakan state dari permainan pada suatu waktu.. Edge memodelkan langkah yang harus dilakukan player untuk sampai ke state tersebut.

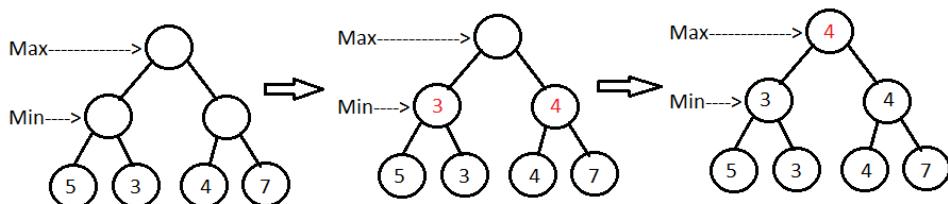


Gambar 3.2.1. Struktur Tree

Proses pencarian juga memanfaatkan algoritma branch bound dengan memanfaatkan depth first search dan memangkas jalur yang bisa dipangkas. Mekanisme pangkas dilakukan dengan melihat apakah suatu jalur masih mungkin menjadi pilihan terbaik untuk pemain pada suatu waktu.

3.2.1. Pemodelan Min-Max pada Tree

Tujuan yang diinginkan adalah memaksimalkan nilai fungsi objektif. Jika kita menghitung fungsi objektif dari sudut pandang pemain pertama (pemain yang jalan pertama), maka hal yang dilakukan orang pertama ketika gilirannya adalah memaksimalkan nilai fungsi objektif. Sebaliknya, hal yang dilakukan orang kedua ketika gilirannya adalah meminimalkan nilai fungsi objektif.



Gambar 3.2.2. Pemodelan Struktur Min-Max pada Tree

Dinamika tersebut menyebabkan pada tiap kedalaman tree, prioritas pemilihan node akan ber. Dari node di kedalaman nol harus memilih node kedalaman satu yang memiliki nilai maksimal, lalu node di kedalaman satu harus memilih node dengan nilai objective function minimal, dan seterusnya secara berganti-gantian. Node yang berusaha memaksimalkan pilihan kita sebut max-node, dan kebalikannya kita sebut dengan min-node.

3.2.2. Mekanisme Pruning

Pruning dilakukan dengan menentukan apakah suatu sub-bagian dari tree bisa berpengaruh pada hasil pencarian. Untuk melakukan hal tersebut, perlu dicatat suatu nilai yang disebut alpha dan beta pada tiap node yang sudah dikunjungi

Alpha : nilai terbaik (nilai tertinggi) dari pilihan yang sudah ditemukan pada titik manapun di jalur untuk max-node

Beta : nilai terbaik (nilai terendah) dari pilihan yang sudah ditemukan pada titik manapun di jalur untuk min-node

Pencarian alfa-beta akan memperbarui nilai alpha beta dari tiap node. Pruning akan dilakukan ketika nilai dari suatu node lebih buruk daripada nilai alpha atau beta untuk max-node atau min node pada saat ini, secara berturut-turut.

3.2.3. Restriksi Kedalaman

Apabila evaluasi fungsi objektif dilakukan hanya ketika semua state telah terisi, maka akan terdapat $2^{\wedge} (64-8)$ kemungkinan. Angka ini terlalu besar untuk dikomputasikan secara efisien. Maka dari itu perlu pembatasan kedalaman evaluasi fungsi objektif.

3.2.4 Tahapan Proses

Dengan penjabaran yang sudah dijelaskan sebelumnya, akan dipaparkan tahapan proses pencarian dengan menggunakan algoritma min-max alpha beta pruning.

Input : Keadaan papan permainan saat ini.

Output : Langkah terbaik atau mendekati langkah terbaik.

Step 1 : Inisialisasi node pertama dengan (**alpha**, **beta**) = (-inf, inf)

Step 2 : Jika sudah mencapai kedalaman maksimal atau state sudah terisi penuh, maka dapat dilakukan evaluasi dengan fungsi objektif

Step 3 : Apabila kedalaman belum maksimal dan state blm terisi penuh maka tentukan apakah state berada di max-node ataupun min-node

Step 4.1 : Apabila sedang berada di max-node, maka ekspansi node dibawahnya secara dfs, anggap sudah didapat nilai dari node dibawahnya yaitu v, bandingkan nilai tersebut dengan alpha, re-assign nilai alpha dengan aturan

```
alpha ← MAX(alpha, v)
apabila v ≥ beta maka sisa cabangnya bisa langsung di prune
```

dengan node anak sisa, pilihlah node yang memiliki nilai terbesar, dan catat path yang ditempuh

Step 4.2 : Apabila sedang berada di minx-node, maka ekspansi node dibawahnya secara dfs, anggap sudah didapat nilai dari node dibawahnya yaitu v, bandingkan nilai tersebut dengan beta, re-assign nilai alpha dengan aturan

```
beta ← MIN(beta, v)
apabila v ≤ alpha maka sisa cabangnya bisa langsung di prune
```

dengan node anak sisa, pilihlah node yang memiliki nilai terkecil, dan catat path yang ditempuh

Step 5 : Lakukan step 3, 4.1 dan, 4.2 hingga didapat nilai node teratas (parent) dengan path yang ditempuh.

Berikut merupakan cuplikan implementasi kode pada tugas ini.

```
public int minValue(State s, int alpha, int beta, int currTreeDepth) {
    if (currTreeDepth == maxTreeDepth || s.isBoardFull()) {
        return s.calcValue();
    }
    int currValue = Integer.MAX_VALUE;
    for (Action action : s.getDefaultActions()) {
        State newState = new State(s, action);
        int newValue = maxValue(newState, alpha, beta, currTreeDepth + 1);
        if (newValue < currValue) {
            currValue = newValue;
        }
        if (currValue <= alpha) {
            return currValue;
        }
        beta = Math.min(beta, currValue);
    }
    return currValue;
}
```

```
public int maxValue(State s, int alpha, int beta, int currTreeDepth) {
    if (s.isBoardFull() || currTreeDepth == maxTreeDepth) {
        return s.calcValue();
    }
    int currValue = Integer.MIN_VALUE;
    for (Action action : s.getDefaultActions()) {
        State newState = new State(s, action);
        int newValue = minValue(newState, alpha, beta, currTreeDepth + 1);
        if (newValue > currValue) {
            currValue = newValue;
        }
    }
    return currValue;
}
```

```
        }
        if (currValue >= beta) {
            return currValue;
        }
        alpha = Math.max(alpha, currValue);
    }
    return currValue;
}

public Action move(State s) {
    int currValue = Integer.MIN_VALUE;
    Action optimalAction = new Action(0,0);

    for (Action action : s.getDefaultActions()) {
        State newState = new State(s, action);
        int newValue = minValue(newState, Integer.MIN_VALUE,
Integer.MAX_VALUE, 1);
        if (newValue > currValue) {
            currValue = newValue;
            optimalAction = action;
        }
    }
    return optimalAction;
}
```

3.3. Proses Pencarian dengan Algoritma Local Search

Simulated Annealing merupakan proses pencarian yang meniru proses annealing dalam metalurgi. Pada algoritma ini, pilihan yang lebih buruk diperbolehkan sehingga dapat membantu menghindari solusi dari memuncak di local optimum. Hal ini diperbolehkan asal fungsi di bawah sebuah fungsi probabilitas tertentu. Berikut merupakan langkah penyelesaian dari algoritma local search.

Input : Keadaan papan permainan saat ini.

Output : Langkah terbaik atau mendekati langkah terbaik.

Step 1 : Mendefinisikan fungsi energi pada algoritma. Fungsi energi yang digunakan merupakan *objective function* yang telah didefinisikan pada bab sebelumnya.

Step 2 : Mendefinisikan temperatur yang digunakan pada algoritma.

Temperature Schedule yang akan digunakan merepresentasikan penurunan temperatur yang akan digunakan selama proses simulated annealing. Scheduling yang dipilih pada penyelesaian ini adalah linear annealing schedule.

$$T = T_0 - k$$

Temperatur awal (T_0) diinisialisasi dengan nilai 2.8. Temperatur awal ini dipilih karena akan menghasilkan probabilitas 50% ketika dihadapkan dengan selisih fungsi energi yang paling sering muncul, yaitu -2. *Cooling schedule* berkurang secara linear dengan variabel k yang diinisialisasi dengan nilai 0.2.

Step 3.1 : Memilih koordinat (x, y) pada papan permainan sebagai koordinat baru dan mencatat hal tersebut sebagai keadaan (*state*) sekarang (S).

Step 3.2 : Memilih koordinat (x, y) pada papan permainan sebagai koordinat baru dan mencatat hal tersebut sebagai keadaan (*state*) yang akan diuji (S_i) terhadap keadaan sekarang (S).

Step 4 : Memutuskan pemilihan penempatan bidak di koordinat tersebut adalah langkah yang tepat. Keputusan ini berdasarkan fungsi probabilitas:

$$P(S, S_i, T) = \begin{cases} 1 & \text{jika } F(S_i) > F(S) \\ e^{\frac{F(S_i) - F(S)}{T}} & \text{jika } F(S_i) \leq F(S) \end{cases}$$

dengan S adalah keadaan papan permainan saat ini,
 (S_i) adalah keadaan papan yang ingin diujikan pada iterasi berikutnya, dan
 T adalah temperatur saat ini yang mengacu pada fungsi yang telah didefinisikan di step 2.

$F(S)$ merupakan fungsi objektif yang ingin dicari poin optimalnya. Keadaan papan dikatakan lebih baik daripada keadaan sebelumnya jika diperoleh $F(S_i) > F(S)$. Jika demikian, bidak akan diletakkan pada koordinat (x, y) . Sebaliknya, jika keadaan papan lebih buruk, maka peletakan bidak bergantung pada nilai $F(S_i) - F(S)$ dan variabel T . Berikut merupakan implementasi algoritma Local Search pada program.

```
public Action move(State s) {
    // Timer
    long startTime = System.currentTimeMillis();
    long maxDuration = 4990; // dalam ms

    // Ambil salah satu random state
    Action current = getRandomAction(s);
    State currentState = new State(s, current);
    int currStateValue = currentState.calcValue();
    int currTemp = initTemp;

    while (System.currentTimeMillis() - startTime < maxDuration) {
        currTemp = schedule(currTemp);
        if (currTemp <= stopTemp) {
            break;
        }

        // Mengambil random state lainnya
        Action next = getRandomAction(s);
        State nextState = new State(s, next);

        // Kalkulasi successor
        int delta = nextState.calcValue() - currStateValue;

        double probability = new Random().nextDouble();
        double threshold = Math.exp((double) delta / currTemp);
        if (delta > 0 || (delta < 0 && probability < threshold)) {
            return next;
        }
    }
    return current;
}
```

Awalnya, algoritma ini akan membangkitkan sebuah state acak (yang kemudian disebut sebagai currentState) yang memiliki objective function tertentu. State tersebut akan dibandingkan dengan state yang dibangkitkan secara acak lainnya (sebut sebagai nextState), dengan ketentuan sebagai berikut.

1. Jika nilai objektif nextState lebih baik daripada currentState, maka nextState akan terpilih.

2. Jika nilai objektif nextState sama dengan currentState, algoritma akan membangkitkan nextState acak lainnya untuk dibandingkan.
3. Jika nilai objektif nextState lebih buruk daripada currentState, maka pemilihan nextState akan bergantung pada suatu probabilitas saat itu.

Probabilitas terpilihnya nextState pada algoritma dirumuskan sebagai edelta-currTemp, dimana delta merupakan selisih antara nilai objektif nextState dan currentState. Variabel currTemp merepresentasikan temperatur saat ini pada algoritma, yang didefinisikan sebagai berikut.

```
private double schedule(int currTemp) {  
    return (currTemp - 0.2);  
}
```

3.4. Proses Pencarian dengan Genetic Algorithm (Alternatif 1)

Genetic Algorithm merupakan suatu teknik yang dapat digunakan dalam mencari solusi optimal dari suatu permasalahan dengan banyak solusi. Sesuai dengan namanya “genetic” maka algoritma ini didasarkan pada proses genetik yang terjadi pada makhluk hidup, dimana terdapat populasi yang setiap individunya merepresentasikan sebuah solusi yang mungkin. Individu tersebut kemudian dikembangkan dengan sebuah nilai *fitness* berdasarkan fungsi *fitness* guna mendapatkan solusi terbaik. Suatu individu dengan nilai *fitness* yang tinggi akan memiliki peluang yang lebih tinggi juga untuk terpilih dan melakukan perkawinan silang dengan individu lain. Berikut merupakan langkah penyelesaian dari genetic algorithm pada penyelesaian permainan dalam tugas besar ini.

Input : Keadaan papan permainan saat ini.

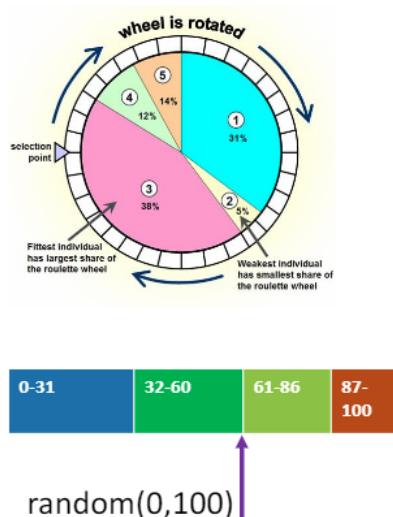
Output : Langkah terbaik atau mendekati langkah terbaik.

Step 1 : Inisiasi parent dengan cara membangkitkan solusi secara random. Jumlah parent yang diinisiasi dapat diatur sesuai kehendak pengguna.

Step 2 : Hitung nilai *fitness* pada setiap parent yang telah diinisiasi

Step 3 : Tetapkan range persentase setiap parent berdasarkan nilai *fitness* yang dimilikinya melalui pembagian nilai *fitness* suatu parent dengan jumlah total nilai *fitness* parent secara keseluruhan.

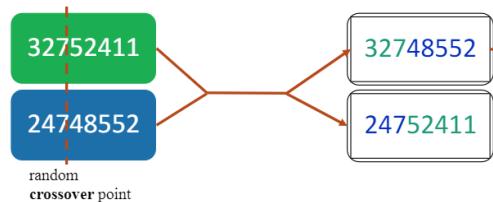
Step 4 : Generate bilangan random yang berkisar antara 0 hingga 1, lalu ambil parent yang menempati range nilai tersebut. Pada persoalan ini, kami mengambil sebanyak 2 bilangan random



Ilustrasi Random Selection of Parent States

Step 5 : Setelah ditemukan 2 solusi berdasarkan bilangan random, periksa apakah aman untuk melakukan *crossover*. Dalam hal ini, yang dimaksud aman adalah solusi yang valid dan tidak merupakan solusi yang telah digunakan. Jika aman, lakukan *crossover* antara kedua suksesor tersebut

Step 6 : Nilai *fitness* dari kedua suksesor akhir hasil *crossover* dihitung dan yang akan diambil menjadi solusi akhir adalah suksesor dengan nilai *fitness* tertinggi. (Mengingat bahwa mutasi merupakan aksi yang bersifat opsional, kami memilih untuk tidak melakukan mutasi lagi guna mencegah tingginya tingkat kerandoman)



Ilustrasi *crossover*

Berikut merupakan cuplikan implementasi kode pada tugas ini.

```

private double fitnessValue (Action a, State s){
    State nextState = new State(s, a);
    return (double) (nextState.calcValue());
}

```

```

private Action crossoverParents (Action x, Action y, State s){
    // crossover sekaligus hitung fitness value
    List<Action> actions = s.getDefaultActions();
    Action c1 = new Action (x.i, y.j);
    Action c2 = new Action (y.i, x.j);

    if (fitnessValue(c1, s) > fitnessValue(c2, s) && actions.contains(c1)) {
        return (c1);
    }
    else if (actions.contains(c2)) {
        return (c2);
    }
    else {
        //If both crossover are illegal move
        if (fitnessValue(x, s) > fitnessValue(y, s)) {
            return (x);
        } else { return (y); }
    }
}

```

```
private Action generateAction (State s) {  
    List<Action> parents = new ArrayList<>();  
    List<Double> persentase = new ArrayList<Double>();  
    List<Double> fitnessValue = new ArrayList<Double>();  
    double totalFitnessValue = 0.0;  
    double patokan = 0.0;  
    Action parent1 = getRandomAction(s);  
    Action parent2 = getRandomAction(s);  
  
    for (int i = 0; i < NUM_PARENTS; i++) {  
        Action newAction = getRandomAction(s);  
        totalFitnessValue += fitnessValue(newAction, s);  
        fitnessValue.add(fitnessValue(newAction, s));  
        parents.add(newAction);  
    }  
    for (int i = 0; i < NUM_PARENTS; i++) {  
        double persen = fitnessValue.get(i) / totalFitnessValue;  
        double urutanPersen = patokan + persen;  
        persentase.add(urutanPersen);  
        patokan += fitnessValue.get(i) / totalFitnessValue;  
    }  
    //generate random number from 0-1  
    double randomNumber1 = Math.random();  
  
    //iterate through persentase, kalau <= ambil action dengan indeks i  
    for (int i = 0; i < NUM_PARENTS; i++) {  
        if (randomNumber1 <= persentase.get(i)) {  
            parent1 = parents.get(i);  
        }  
    }  
  
    //generate random number from 0-1  
    double randomNumber2 = Math.random();  
  
    //iterate through persentase, kalau <= ambil action dengan indeks i  
    for (int i = 0; i < NUM_PARENTS; i++) {  
        if (randomNumber2 <= persentase.get(i)) {  
            parent2 = parents.get(i);  
        }  
    }  
    return crossoverParents(parent1, parent2, s));  
}
```

3.4. Proses Pencarian dengan Genetic Algorithm (Alternatif 2)

Sama halnya dengan penjelasan tentang genetic algorithm pada bagian sebelumnya, algoritma ini menggunakan pola *survival of the fittest*.

3.4.1. Pembobotan Fungsi Objektif

Pada alternatif ini, fungsi objektif akan dimodifikasi. Fungsi objektif bisa didefinisikan dari lima bilangan bulat a_0, a_1, a_2, a_3, a_4 . Anggap x_n menyatakan banyak petak yang miliki sendiri yang punya adjacent square sebanyak n . Sementara y_n adalah banyak petak milik lawan yang punya adjacent square sebanyak n . Maka

$$\text{objective}(a_0, a_1, a_2, a_3, a_4, \text{board}) = \sum_{i=0}^4 a_i * x_i - \sum_{i=0}^4 a_i * y_i$$

3.4.2. Kromosom

Pemodelan kromosom akan menggunakan senarai lima bilangan bulat a_0, a_1, a_2, a_3, a_4 yang menyatakan pembobotan dari fungsi objektif.

3.4.3. Fungsi Fitness

Untuk menghitung fitness function dari kromosom, diadakan sebuah simulasi permainan 28 ronde antara MinMaxBot1 dan MinMaxBot2. Hal yang membedakan antar MinMaxBot1 dan MinMaxBot2 adalah, MinMaxBot1 menggunakan objektif function sesuai dengan masukan kromosom. Sementara MinMaxBot2 menggunakan fungsi objektif yang standar (atau bisa dimodelkan dengan memiliki nilai $(a_0, a_1, a_2, a_3, a_4) = (1, 1, 1, 1, 1)$).

Hasil dari fungsi fitness adalah banyak petak yang dimiliki oleh MinMaxBot1 pada akhir permainan.

3.4.4. Proses Seleksi

Untuk program ini akan digunakan sampel populasi sebanyak 4 kromosom. Langkah pertama adalah menghitung fungsi objektif dari tiap kromosom

Andai hasil fungsi objektif kromosom k_0, k_1, \dots, k_3 adalah $f_0, f_1, f_2, \dots, f_3$ maka dicari nilai f_{\min}

$$f_{\min} = \min(f_0, f_1, f_2, f_3) - 1$$

Lalu dihitung nilai g_n untuk $n = 0, 1, \dots, 3$ dengan

$$g_n = f_n - f_{\min}$$

Maka didapat p_n , yaitu probabilitas kromosom n terpilih untuk tahap selanjutnya yaitu

$$p_n = g_n / (g_0 + g_1 + g_2 + g_3) \text{ untuk } n = 0, 1, \dots, 3.$$

Akan dilakukan empat kali pemilihan dengan pada tiap pemilihan kemungkinan kromosom n terpilih adalah p_n

3.4.5. Proses Kawin Silang

Dari hasil seleksi tersebut, akan dibuat pasangan dan dikawin silangkan. Anggap hasil seleksi kromosom adalah s_0, s_1, s_2, s_3 . Maka s_0 akan dipasangkan dengan s_1 , dan s_2 akan dipasangkan dengan s_3 . Untuk pasangan s_0 dan s_1 akan dipilih angka acak R dari himpunan bilangan $\{0, 1, 2, 3, 4\}$. Maka R angka pertama dari kromosom s_0 akan ditukar dengan R angka pertama dari s_1 bot. Hal yang sama terjadi untuk pasangan s_2 dan s_3 .

3.4.6. Proses Mutasi

Proses mutasi terakhir akan suatu fungsi yang dipanggil berulang-ulang. Untuk setiap pemanggilan fungsi mutasi akan mengubah nilai pada suatu kromosom acak dengan menambah nilai nya untuk suatu bilangan bulat acak -5 sampai dengan 5 (inklusif). Untuk program ini, mutate dipanggil sebanyak 4 kali. Hasil dari mutasi ini akan dilakukan dimasukkan ke proses seleksi kembali.

3.4.7. Hasil Akhir

Proses seleksi-perkawinan silang-mutasi akan di iterasi sebanyak 100 kali. Lalu akan dipilih acak dari 4 populasi dan pembobotannya akan dimasukkan kedalam simulbot (sama halnya seperti kasus MinMaxBot1 yang tadi). Simulbot inilah yang akan dipakai sebagai genetik algorithm bot yang melawan manusia atau bot lainnya.

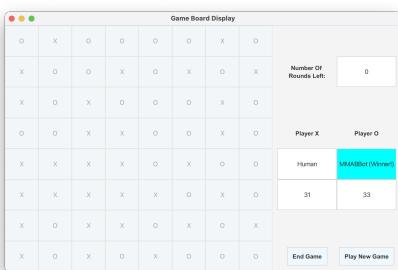
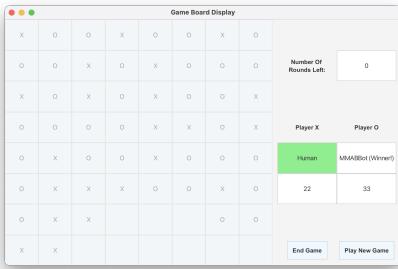
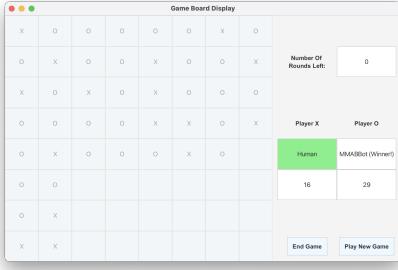
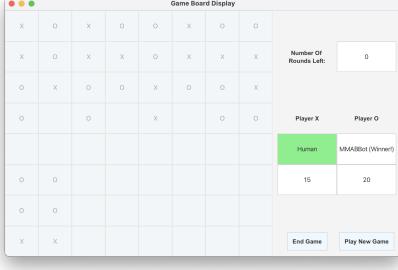
BAB IV

PENGUJIAN

4.1. Pertandingan antara Bot Minimax dan Manusia

Berikut merupakan pengujian dengan Bot Minimax sebagai pemain yang bergerak pertama.

Tabel 4.1.1. Pengujian antara Bot Minimax dan Manusia

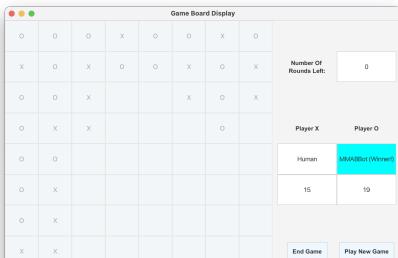
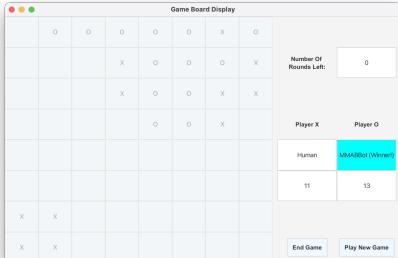
Jumlah Ronde	Tangkapan Layar	Analisis																																																																
28	 <p>Game Board Display</p> <table border="1"> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Human MMABot (Winner)</p> <p>31 33</p> <p>End Game Play New Game</p>	O	X	O	O	O	O	X	O	X	O	O	X	O	X	O	X	X	O	X	O	O	O	X	O	O	O	X	X	O	O	X	O	X	X	X	X	O	X	O	O	X	X	X	X	X	O	X	O	X	O	X	X	O	X	O	X	X	O	X	O	X	O	O	O	Pemenang: MMABot Skor MMABot: 33 Skor Human: 31
O	X	O	O	O	O	X	O																																																											
X	O	O	X	O	X	O	X																																																											
X	O	X	O	O	O	X	O																																																											
O	O	X	X	O	O	X	O																																																											
X	X	X	X	O	X	O	O																																																											
X	X	X	X	X	O	X	O																																																											
X	O	X	X	O	X	O	X																																																											
X	O	X	O	X	O	O	O																																																											
23	 <p>Game Board Display</p> <table border="1"> <tr><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>X</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>X</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td>O</td><td>O</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Human MMABot (Winner)</p> <p>22 33</p> <p>End Game Play New Game</p>	X	O	O	X	O	O	X	O	O	O	X	O	X	O	O	O	X	O	X	O	X	O	O	X	O	O	O	O	X	X	O	X	O	X	O	O	X	O	O	O	O	X	X	X	O	O	X	O	O	X	X	X	O	O	X	O	X	X					O	O	Pemenang: MMABot Skor MMABot: 33 Skor Human: 22
X	O	O	X	O	O	X	O																																																											
O	O	X	O	X	O	O	O																																																											
X	O	X	O	X	O	O	X																																																											
O	O	O	O	X	X	O	X																																																											
O	X	O	O	X	O	O	O																																																											
O	X	X	X	O	O	X	O																																																											
O	X	X	X	O	O	X	O																																																											
X	X					O	O																																																											
18	 <p>Game Board Display</p> <table border="1"> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td></td></tr> <tr><td>O</td><td>O</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>O</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Human MMABot (Winner)</p> <p>16 29</p> <p>End Game Play New Game</p>	X	O	O	O	O	O	X	O	O	X	O	O	X	O	O	X	X	O	X	O	X	O	O	O	O	O	O	O	X	X	O	X	O	X	O	O	O	X	O		O	O							O	X							X	X							Pemenang: MMABot Skor MMABot: 26 Skor Human: 19
X	O	O	O	O	O	X	O																																																											
O	X	O	O	X	O	O	X																																																											
X	O	X	O	X	O	O	O																																																											
O	O	O	O	X	X	O	X																																																											
O	X	O	O	O	X	O																																																												
O	O																																																																	
O	X																																																																	
X	X																																																																	
13	 <p>Game Board Display</p> <table border="1"> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Human MMABot (Winner)</p> <p>15 20</p> <p>End Game Play New Game</p>	X	O	X	O	O	X	O	O	X	O	X	X	O	X	X	X	O	X	O	O	X	O	O	X	O	O	O	X	X	O	O	X	O	O	O	X	O	O	O	X	O	O	O	X	O	O	O	X	X	X							Pemenang: MMABot Skor MMABot: 20 Skor Human: 15								
X	O	X	O	O	X	O	O																																																											
X	O	X	X	O	X	X	X																																																											
O	X	O	O	X	O	O	X																																																											
O	O	O	X	X	O	O	X																																																											
O	O	O	X	O	O	O	X																																																											
O	O	O	X	O	O	O	X																																																											
X	X																																																																	

8	<p>Pemenang: Manusia Skor MMABot: 12 Skor Human: 13</p>
---	---

Berikut merupakan pengujian dengan manusia sebagai pemain yang bergerak pertama.

Tabel 4.1.2. Pengujian antara Bot Minimax dan Manusia

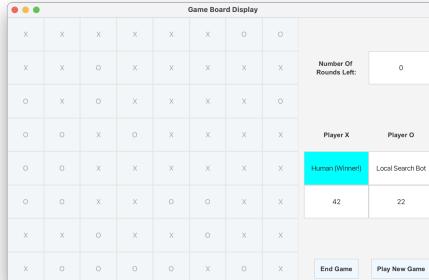
Jumlah Ronde	Tangkapan Layar	Analisis
28	<p>Pemenang: MMABot Skor MMABot: 35 Skor Human: 29</p>	
23	<p>Pemenang: MMABot Skor MMABot: 29 Skor Human: 25</p>	
18	<p>Pemenang: MMABot Skor MMABot: 23 Skor Human: 21</p>	

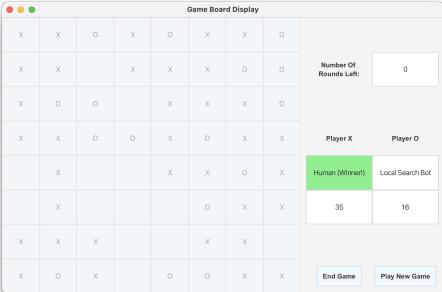
13		Pemenang: MMABot Skor MMABot: 19 Skor Human: 15
8		Pemenang: MMABot Skor MMABot: 13 Skor Human: 11

4.2. Pertandingan antara Bot Local Search dan Manusia

Berikut merupakan pengujian dengan Bot Local Search sebagai pemain yang bergerak pertama.

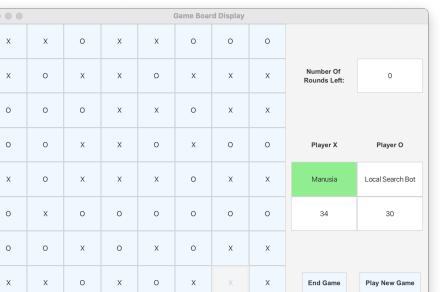
Tabel 4.2.1. Pengujian antara Bot Local Search dan Manusia

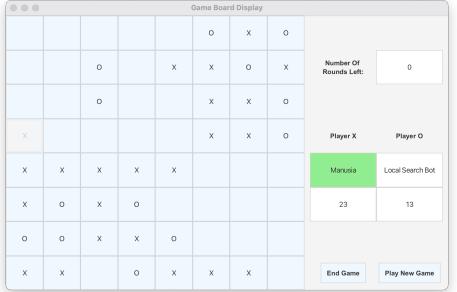
Jumlah Ronde	Tangkapan Layar	Analisis
28		Pemenang: Manusia Skor Bot Local Search: 22 Skor manusia: 42 <p>Pada permainan ini, bot beberapa kali bertemu dengan keadaan yang tidak ideal, namun tidak dipilih karena probabilitas yang tidak memungkinkan. Berikut merupakan contohnya. Jika $p < t$, maka state berikutnya akan terpilih.</p> <pre>delta -4 p = 0.7054618258717972 t = 0.2147111723416972 delta -4 p = 0.579832518117241 t = 0.18887560283756177 delta -4 p = 0.5832309284116226 t = 0.16232061118184807</pre>

21		<p>Pemenang: Manusia Skor Bot Local Search: 16 Skor manusia: 35</p> <p>Pada permainan ini, bot berkali-kali bertemu dengan keadaan yang tidak ideal, namun tidak dipilih karena probabilitas yang tidak memungkinkan.</p> <pre>delta -2 p = 0.9133548527899477 t = 0.43459820850707814 delta -2 p = 0.7770663808936373 t = 0.40289032152913284 delta -2 p = 0.5675702948606107 t = 0.36787944117144217 delta -2 p = 0.7172232194329602 t = 0.3291929878079054 delta -2 p = 0.14255643243382576 t = 0.28650479686019 hitung</pre>
14		<p>Pemenang: Manusia Skor Bot Local Search: 16 Skor manusia: 21</p>

Berikut merupakan pengujian dengan manusia sebagai pemain yang bergerak pertama.

Tabel 4.2.2. Pengujian antara Bot Local Search dan Manusia

Jumlah Ronde	Tangkapan Layar	Analisis
28		<p>Pemenang: Manusia Skor Bot Local Search: 30 Skor manusia: 34</p> <p>Pada permainan ini, bot beberapa kali memilih keadaan yang tidak ideal. Tetapi, bot juga berhasil menemukan keadaan yang paling menguntungkan baginya dimana ia mendapat delta = 4. Delta merupakan selisih nilai objektif antara state yang diuji dengan state sebelumnya.</p>

21		Pemenang: Manusia Skor Bot Local Search: 18 Skor manusia: 32 <pre>delta -2 p = 0.20818790846538715 t = 0.4633693692311752</pre>
14		Pemenang: Manusia Skor Bot Local Search: 13 Skor manusia: 23 <p>Pada permainan ini, bot beberapa kali memilih keadaan yang tidak ideal baginya. Contoh tertera dibawah. Variabel t merupakan nilai temperatur sekarang, dan p merupakan angka yang dibangkitkan secara acak. Jika $p < t$, maka aksi berikutnya diterima.</p> <pre>delta -2 p = 0.283007593465988 t = 0.4633693692311752</pre>

4.3. Pertandingan antara Bot Minimax dan Bot Local Search

Berikut merupakan pengujian dengan Bot Minimax sebagai pemain yang bergerak pertama.

Tabel 4.3.1. Pengujian antara Bot Minimax dan Bot Local Search

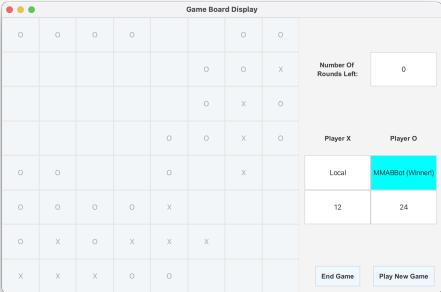
Jumlah Ronde	Tangkapan Layar	Analisis
28		Pemenang: - Skor Bot Local Search: 32 Skor Bot MMABot: 32

21	<p>Game Board Display</p> <table border="1"> <tr><td>O</td><td></td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr> <tr><td></td><td></td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td></td><td></td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td></td></tr> <tr><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td></td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Local MMABot (Winner)</p> <p>12 39</p> <p>End Game Play New Game</p>	O		O	O	X	O	O			O	X	O	O	X			O	O	O	O	X	O	O	O	O	O	O		O	X	O	O	O	O	O	X	X	O	O	X	O	O	O	O	O	X	O	X		X	O	O	O	O	O	X	Pemenang: MMABot Skor Bot Local Search: 39 Skor Bot MMAB: 12
O		O	O	X	O	O																																																				
		O	X	O	O	X																																																				
		O	O	O	O	X																																																				
O	O	O	O	O	O																																																					
O	X	O	O	O	O	O																																																				
X	X	O	O	X	O	O																																																				
O	O	O	X	O	X																																																					
X	O	O	O	O	O	X																																																				
14	<p>Game Board Display</p> <table border="1"> <tr><td>O</td><td>O</td><td></td><td>O</td><td></td><td>O</td><td>O</td></tr> <tr><td></td><td></td><td>O</td><td></td><td>O</td><td>X</td><td></td></tr> <tr><td></td><td></td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td></tr> <tr><td>O</td><td>O</td><td></td><td></td><td>O</td><td></td><td></td></tr> <tr><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td></td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td></td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Local MMABot (Winner)</p> <p>10 27</p> <p>End Game Play New Game</p>	O	O		O		O	O			O		O	X				O	O	O	X	X	O	O			O			O	X	O	X	O	O		X	O	O	O	O	O	X	O	O	O	O	O	O		X	O	O	O	X	O	X	Pemenang: MMABot Skor Bot Local Search: 27 Skor Bot MMAB: 10
O	O		O		O	O																																																				
		O		O	X																																																					
		O	O	O	X	X																																																				
O	O			O																																																						
O	X	O	X	O	O																																																					
X	O	O	O	O	O	X																																																				
O	O	O	O	O	O																																																					
X	O	O	O	X	O	X																																																				

Berikut merupakan pengujian dengan Bot Local Search sebagai pemain yang bergerak pertama.

Tabel 4.3.2. Pengujian antara Bot Minimax dan Bot Local Search

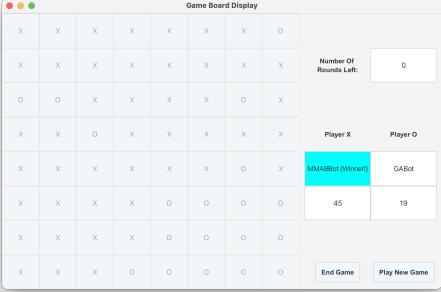
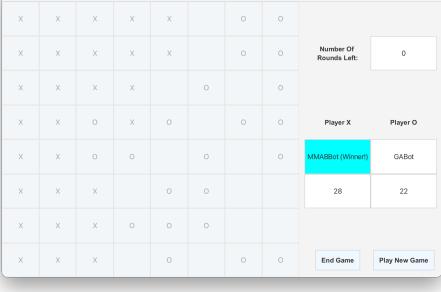
Jumlah Ronde	Tangkapan Layar	Analisis																																																								
28	<p>Game Board Display</p> <table border="1"> <tr><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Local MMABot (Winner)</p> <p>23 41</p> <p>End Game Play New Game</p>	X	X	O	O	O	O	O	X	O	O	O	O	O	X	O	X	X	O	O	O	X	X	O	O	O	X	O	X	X	O	O	O	O	X	O	X	O	O	O	O	X	X	X	O	O	O	X	O	O	X	X	X	O	O	O	O	Pemenang: MMABot Skor Bot Local Search: 23 Skor Bot MMAB: 41
X	X	O	O	O	O	O																																																				
X	O	O	O	O	O	X																																																				
O	X	X	O	O	O	X																																																				
X	O	O	O	X	O	X																																																				
X	O	O	O	O	X	O																																																				
X	O	O	O	O	X	X																																																				
X	O	O	O	X	O	O																																																				
X	X	X	O	O	O	O																																																				
21	<p>Game Board Display</p> <table border="1"> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>O</td><td></td><td>O</td><td>X</td><td>X</td></tr> <tr><td></td><td></td><td>X</td><td>X</td><td>O</td><td>O</td><td></td></tr> <tr><td>O</td><td>O</td><td></td><td></td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>O</td><td></td><td></td><td>O</td><td>X</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> </table> <p>Number Of Rounds Left: 0</p> <p>Player X Player O</p> <p>Local MMABot (Winner)</p> <p>15 35</p> <p>End Game Play New Game</p>	X	O	O	O	O	O	O	O	O	O		O	X	X			X	X	O	O		O	O			O	O	O	O	O			O	X	O	O	X	X	O	O	O	X	O	X	O	X	O	X	X	X	O	O	O	O	O	O	Pemenang: MMABot Skor Bot Local Search: 15 Skor Bot MMAB: 35
X	O	O	O	O	O	O																																																				
O	O	O		O	X	X																																																				
		X	X	O	O																																																					
O	O			O	O	O																																																				
O	O			O	X	O																																																				
O	X	X	O	O	O	X																																																				
O	X	O	X	O	X	X																																																				
X	O	O	O	O	O	O																																																				

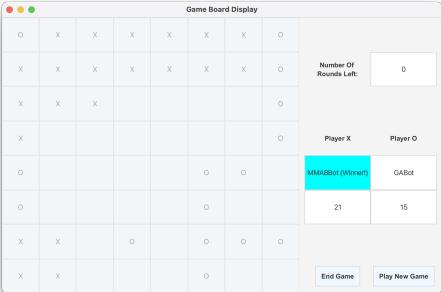
14		Pemenang: MMABBot Skor Bot Local Search: 12 Skor Bot MMAB: 24
----	---	--

4.4. Pertandingan antara Bot Minimax dan Bot Genetic Algorithm Ver 1

Berikut merupakan pengujian dengan Bot Minimax sebagai pemain yang bergerak pertama.

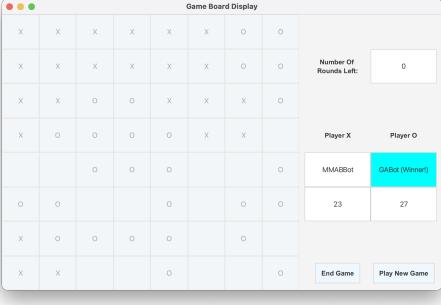
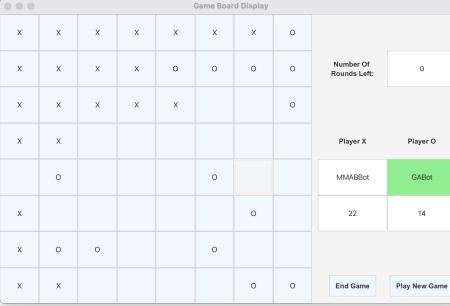
Tabel 4.4.1. Pengujian antara Bot Minimax dan Bot Genetic Algorithm Ver 1

Jumlah Ronde	Tangkapan Layar	Analisis
28		Pemenang: MMABBot Skor Bot GABot: 19 Skor Bot MMAB: 45
21		Pemenang: MMABBot Skor Bot GABot: 22 Skor Bot MMAB: 28

14		<p>Pemenang: MMABBot Skor Bot GABot: 15 Skor Bot MMAB: 21</p>
----	---	--

Berikut merupakan pengujian dengan Bot Genetic Algorithm Ver 1 sebagai pemain yang bergerak pertama.

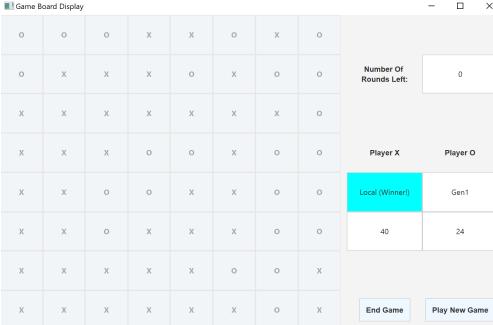
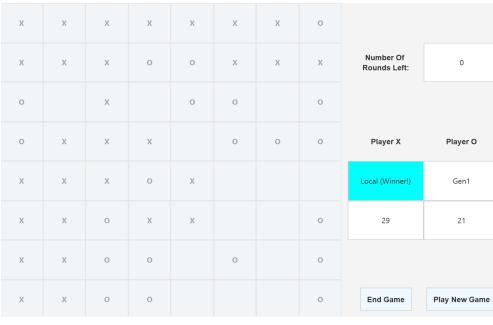
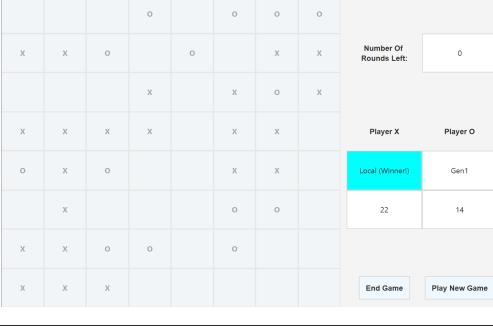
Tabel 4.4.2. Pengujian antara Bot Minimax dan Bot Genetic Algorithm Ver 1

Jumlah Ronde	Tangkapan Layar	Analisis
28		<p>Pemenang: MMABBot Skor MMABBot: 47 Skor GABot: 17</p>
21		<p>Pemenang: GABot Skor MMABBot: 27 Skor GABot: 23</p>
14		<p>Pemenang: MMABBot Skor MMABBot: 22 Skor GABot: 14</p>

4.5. Pertandingan antara Bot Local Search dan Bot Genetic Algorithm Ver 1

Berikut merupakan pengujian dengan Bot Local Search sebagai pemain yang bergerak pertama.

Tabel 4.5.1. Pengujian antara Bot Local Search dan Bot Genetic Algorithm Ver 1

Jumlah Ronde	Tangkapan Layar	Analisis
28	 <p>Pemenang: Local Skor Local: 40 Skor GA1Bot: 24</p>	
21	 <p>Pemenang: Local Skor Local: 29 Skor GA1Bot: 21</p>	
14	 <p>Pemenang: Local Skor Local: 22 Skor GA1Bot: 14</p>	

Berikut merupakan pengujian dengan Bot Genetic Algorithm Ver 1 sebagai pemain yang bergerak pertama.

Tabel 4.5.2. Pengujian antara Bot Local Search dan Bot Genetic Algorithm Ver 1

Jumlah Ronde	Tangkapan Layar	Analisis
--------------	-----------------	----------

28	<tr><td>O</td><td>O</td><td>O</td><td></td><td></td><td>O</td><td>O</td><td>O</td></tr> <tr><td></td><td>X</td><td></td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td></td><td>X</td><td>X</td><td>O</td><td></td></tr> <tr><td>O</td><td>O</td><td></td><td>O</td><td>X</td><td>X</td><td>O</td><td></td></tr> <tr><td>X</td><td>O</td><td>X</td><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td></td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td></td><td>O</td><td>O</td><td>X</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>O</td><td></td><td>O</td><td>X</td><td>O</td><td>X</td></tr>	O	O	O			O	O	O		X		O	O	O	O	O	X	O	X		X	X	O		O	O		O	X	X	O		X	O	X		O	O	O	O	X	O	X	O	O	X		X	X	O	O		O	O	X	X	X	O	O		O	X	O	X
O	O	O			O	O	O																																																										
	X		O	O	O	O	O																																																										
X	O	X		X	X	O																																																											
O	O		O	X	X	O																																																											
X	O	X		O	O	O	O																																																										
X	O	X	O	O	X		X																																																										
X	O	O		O	O	X	X																																																										
X	O	O		O	X	O	X																																																										

 Number Of Rounds Left: 0
 Player X: Gen1
 Player O: Local (Winner)
 Score: 18 vs 32
 Buttons: End Game, Play New Game
 Pemenang: Local Skor Local: 32 Skor GA1Bot: 18 || 21 | | | | | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | | O | O | O | | | O | O | O | | | X | | O | O | O | O | O | | X | O | X | | X | X | O | | | O | O | | O | X | X | O | | | X | O | X | | O | O | O | O | | X | O | X | O | O | X | | X | | X | O | O | | O | O | X | X | | X | O | O | | O | X | O | X | Number Of Rounds Left: 0 Player X: Gen1 Player O: Local (Winner) Score: 12 vs 24 Buttons: End Game, Play New Game | Pemenang: Local Skor Local: 24 Skor GA1Bot: 12 |
| 14 | | | | | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | | O | O | | | | O | O | O | | | O | | | O | X | X | O | | X | | | O | | O | | | | X | | | O | | O | | | | X | X | | | | | | | | O | O | O | | O | O | | | | O | O | | O | O | X | X | | | X | O | O | O | X | X | X | | Number Of Rounds Left: 0 Player X: Gen1 Player O: Local (Winner) Score: 12 vs 24 Buttons: End Game, Play New Game | Pemenang: Local Skor Local: 24 Skor GA1Bot: 12 |

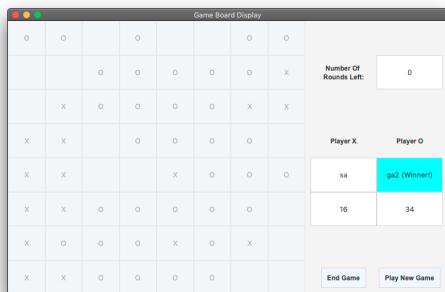
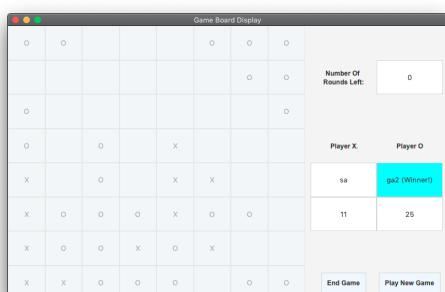
4.6. Pertandingan antara Bot Local Search dan Bot Genetic Algorithm Ver 2

Berikut merupakan pengujian dengan Bot Local Search sebagai pemain yang bergerak pertama.

Tabel 4.6.1. Pengujian antara Bot Local Search dan Bot Genetic Algorithm Ver 2

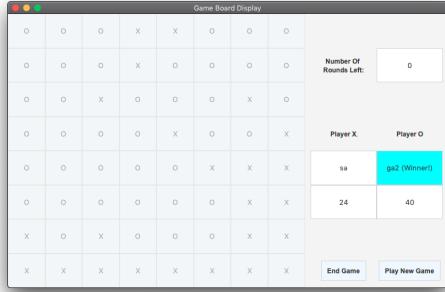
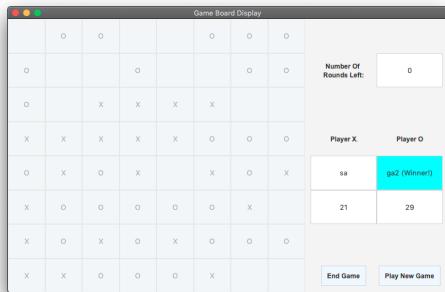
Jumlah Ronde	Tangkapan Layar	Analisis																																																															
28	<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr> <tr><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr> <tr><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td></tr> <tr><td>X</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>	O	O	O	O	O	O	O	O	O	X	X	O	O	O	O	O	X	O	X	O	O	O	O	O	O	O	X	O	O	O	O	O	O	X	O	X	X	O	X	O	X	O	O	O	X	O	O	X	X	O	X	O	X	X	O	X	X	X	O	O	O	X	X	X
O	O	O	O	O	O	O	O																																																										
O	X	X	O	O	O	O	O																																																										
X	O	X	O	O	O	O	O																																																										
O	O	X	O	O	O	O	O																																																										
O	X	O	X	X	O	X	O																																																										
X	O	O	O	X	O	O	X																																																										
X	O	X	O	X	X	O	X																																																										
X	X	O	O	O	X	X	X																																																										

 Number Of Rounds Left: 0
 Player X: ga2 (Winner)
 Player O: sa
 Score: 42 vs 22
 Buttons: End Game, Play New Game
 Pemenang: GA2Bot Skor SABot: 22 Skor GA2Bot: 42 |

21		Pemenang: GA2Bot Skor SABot: 16 Skor GA2Bot: 34
14		Pemenang: GA2Bot Skor SABot: 11 Skor GA2Bot: 25

Berikut merupakan pengujian dengan Bot Genetic Algorithm Ver 2 sebagai pemain yang bergerak pertama.

Tabel 4.6.2. Pengujian antara Bot Local Search dan Bot Genetic Algorithm Ver 2

Jumlah Ronde	Tangkapan Layar	Analisis
28		Pemenang: GA2Bot Skor SABot: 24 Skor GA2Bot: 40
21		Pemenang: GA2Bot Skor SABot: 21 Skor GA2Bot: 29

14		Pemenang: GA2Bot Skor SABot: 13 Skor GA2Bot: 23
----	--	---

4.7. Pertandingan antara Bot Minimax dan Bot Genetic Algorithm Ver 2

Berikut merupakan pengujian dengan Bot Minimax sebagai pemain yang bergerak pertama.

Tabel 4.7.1. Pengujian antara Bot Minimax dan Bot Genetic Algorithm Ver 2

Jumlah Ronde	Tangkapan Layar	Analisis
28		Pemenang: GA2Bot Skor GA2Bot: 38 Skor MMABBot: 26
21		Pemenang: GA2Bot Skor GA2Bot: 32 Skor MMABBot: 18

14	<table border="1" style="margin-top: 10px; width: 100%; text-align: center;"> <tr> <td colspan="9">Game Board Display</td> </tr> <tr> <td>O</td><td></td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td></td><td></td> </tr> <tr> <td>O</td><td></td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td></td> </tr> <tr> <td></td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td></td> </tr> <tr> <td></td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td> </tr> <tr> <td>O</td><td></td><td>O</td><td></td><td>X</td><td></td><td></td><td></td><td></td> </tr> <tr> <td>O</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td>O</td><td>O</td><td></td><td></td><td></td><td></td><td></td> </tr> </table>	Game Board Display									O		O	O	X	O	O			O		O	O	O	O	O	X			O	O	X	O	O	O	X			X	O	X	X	X	X	X		O		O		X					O									X	X								X	X	O	O						<p>Pemenang: GA2Bot Skor GA2Bot: 22 Skor MMABBot: 14</p>
Game Board Display																																																																																			
O		O	O	X	O	O																																																																													
O		O	O	O	O	O	X																																																																												
	O	O	X	O	O	O	X																																																																												
	X	O	X	X	X	X	X																																																																												
O		O		X																																																																															
O																																																																																			
X	X																																																																																		
X	X	O	O																																																																																

Berikut merupakan pengujian dengan Bot Genetic Algorithm Ver 2 sebagai pemain yang bergerak pertama.

Tabel 4.7.2. Pengujian antara Bot Minimax dan Bot Genetic Algorithm Ver 2

Jumlah Ronde	Tangkapan Layar	Analisis																																																																																	
28	<table border="1" style="margin-top: 10px; width: 100%; text-align: center;"> <tr> <td colspan="9">Game Board Display</td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td> </tr> </table>	Game Board Display									X	O	X	O	X	X	O	O		X	O	X	O	O	X	O	X		X	O	O	O	X	O	X	X		X	O	O	O	X	X	O	X		O	X	O	O	X	O	O	X		X	O	X	O	X	O	O	X		X	X	O	X	O	X	O	X		X	X	X	O	X	X	X	X		<p>Pemenang: MMABBot Skor GABot: 29 Skor MMABBot: 35</p>
Game Board Display																																																																																			
X	O	X	O	X	X	O	O																																																																												
X	O	X	O	O	X	O	X																																																																												
X	O	O	O	X	O	X	X																																																																												
X	O	O	O	X	X	O	X																																																																												
O	X	O	O	X	O	O	X																																																																												
X	O	X	O	X	O	O	X																																																																												
X	X	O	X	O	X	O	X																																																																												
X	X	X	O	X	X	X	X																																																																												
21	<table border="1" style="margin-top: 10px; width: 100%; text-align: center;"> <tr> <td colspan="9">Game Board Display</td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>O</td><td></td><td>X</td><td>X</td><td></td><td>O</td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td>O</td><td>O</td><td></td><td></td><td></td><td></td><td></td> </tr> </table>	Game Board Display									X	O	X	O	X	X	O	O		X	O	X	O	O	X	O	X		X	O	O	O	X	O	X	X		X	O	O	O	X	X	O	X		O	O	O	O	X	X	O	X		O		X	X		O				X	X								X	X	O	O						<p>Pemenang: GA2Bot Skor GABot: 25 Skor MMABBot: 25</p>
Game Board Display																																																																																			
X	O	X	O	X	X	O	O																																																																												
X	O	X	O	O	X	O	X																																																																												
X	O	O	O	X	O	X	X																																																																												
X	O	O	O	X	X	O	X																																																																												
O	O	O	O	X	X	O	X																																																																												
O		X	X		O																																																																														
X	X																																																																																		
X	X	O	O																																																																																
14	<table border="1" style="margin-top: 10px; width: 100%; text-align: center;"> <tr> <td colspan="9">Game Board Display</td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td>O</td><td>O</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>X</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>X</td><td>X</td><td></td> </tr> <tr> <td>X</td><td>X</td><td>O</td><td></td><td></td><td>O</td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>O</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>X</td><td>X</td><td>O</td><td>O</td><td></td><td></td><td></td><td></td><td></td> </tr> </table>	Game Board Display									X	O	X	O	X	X	O	O		X	O	X	O	O	X	O	X		X	O	O	O	X	O	X	X		X	X	O			O				X									O									X	X								X	X	O	O						<p>Pemenang: MMABBot Skor GABot: 17 Skor MMABBot: 19</p>
Game Board Display																																																																																			
X	O	X	O	X	X	O	O																																																																												
X	O	X	O	O	X	O	X																																																																												
X	O	O	O	X	O	X	X																																																																												
X	X	O			O																																																																														
X																																																																																			
O																																																																																			
X	X																																																																																		
X	X	O	O																																																																																

4.8. Ringkasan

Berikut adalah ringkasan dari seluruh pertandingan yang dilakukan

Tabel 4.8.1. Ringkasan Pertandingan

Bot	Jumlah Kemenangan	Jumlah Pertandingan	Win Rate
Simulated Annealing (Local Search, SABot)	5	24	20,83%
Minimax Alpha Beta Pruning (MMABot)	21	28	75%
Genetic Algorithm Ver 1 (GABot)	2	12	16,67%
Genetic Algorithm Ver 2 (GA2Bot)	10	12	83,33%

BAB V

SIMPULAN DAN SARAN

5.1. Simpulan

Pada tugas besar I Intelligensi Buatan ini telah diimplementasikan algoritma local search berupa simulated annealing, algoritma minimax, dan algoritma genetik ver 2 untuk menyelesaikan Adjacency Strategy Game. Berdasarkan tabel pengujian, diperoleh bahwa algoritma yang performanya paling baik adalah minimax alpha-beta pruning dan genetik versi 2.

Algoritma minimax alpha-beta pruning secara umum memang merupakan algoritma yang paling cocok dengan permainan *turn-based*. Algoritma genetik yang digunakan sebenarnya berbasis pada algoritma minimax alpha-beta pruning juga. Namun yang membedakan adalah fungsi objektif yang dipakai. Pada genetic algorithm bot. Fungsi objektif yang digunakan sudah lebih *specialized* dari berkali-kali iterasi. Pada fungsi objektif yang digunakan algorithm bot juga lebih kompleks dan memperhatikan lebih banyak variabel.

Namun, berdasarkan percobaan yang dilakukan, algoritma minimax alpha-beta pruning juga lebih unggul dibanding algoritma genetic ver 2 saat jumlah ronde yang dibatasi berada di pertengahan (tidak sampai ke ronde 28). Hal ini dikarenakan simulasi yang digunakan algoritma genetik ver 2 adalah permainan dengan 28 ronde. Hal ini menyebabkan fungsi objektif yang dipakai algoritma genetik ver 2 lebih *specialized* untuk permainan yang sampai ke ronde 28.

5.2. Saran

Tugas Besar I Semester I Tahun 2023/2024 menjadi salah satu tugas yang memberikan pelajaran baru bagi penulis. Berdasarkan pengalaman penulis mengerjakan tugas ini, berikut merupakan saran untuk pembaca yang ingin melakukan atau mengerjakan hal yang serupa.

1. Perancangan algoritma dan struktur program perlu diperhatikan dalam mengerjakan tugas ini. Hal ini disebabkan oleh struktur program awal yang restriktif dan hanya menyediakan antarmuka untuk pengguna manusia dan bot. Penulis harus melakukan banyak refaktor kode yang memungkinkan *gameplay* bot vs bot.
2. Keefektifan dalam kerja sama tim merupakan hal yang penting dalam mengerjakan tugas ini. Selain pembagian tugas yang merata, penulis terbantu oleh beberapa kali kerja kelompok dan pemakaian *real-time collaboration app*, seperti Google Docs. Selain itu, penggunaan version control (Github) disarankan untuk memudahkan mengelola pekerjaan secara asinkron.

REFERENSI

- Aryanmishra2252. (2023). Difference Between Hill Climbing and Simulated Annealing Algorithm. <https://www.geeksforgeeks.org/difference-between-hill-climbing-and-simulated-annealing-algorithm/>. Diakses pada 15 September 2023.
- Khodra, Masayu Leylia. (2023). Beyond Classical Search: Classical vs Local Search. Teknik Informatika Insitut Teknologi Bandung.
- Khodra, Masayu Leylia. (2023). Beyond Classical Search: Hill Climbing Search. Teknik Informatika Insitut Teknologi Bandung.
- Khodra, Masayu Leylia. (2023). Beyond Classical Search: Simulated Annealing. Teknik Informatika Insitut Teknologi Bandung.
- Khodra, Masayu Leylia. (2023). Beyond Classical Search: Genetic Algorithm. Teknik Informatika Insitut Teknologi Bandung.
- Khodra, Masayu Leylia. (2023). Adversarial Search. Teknik Informatika Insitut Teknologi Bandung.
- Tzung-Pei, Hong, Ke-Yuan, Huang, Wen-Yang, Lin. (2001). *Adversarial Search by Evolutionary Computation*. Evolutionary Computation 9(3):371-385.

LAMPIRAN

Repository Github

<https://github.com/alishalistyaa/IF3170-Tubes-1-AI.git>

Lembar Kontribusi

NIM	Nama	Kontribusi
13521045	Fakhri Muhammad Mahendra	GA2Bot, MMABBot
13521129	Chiquita Ahsanunnisa	Strukturisasi Program, Abstrak Bot
13521151	Vanessa Rebecca Wiyono	GA1Bot
13521171	Alisha Listya Wardhani	Simulated Annealing Bot