Adversarial Search untuk Mencari Sekuens Pergerakan Optimal dalam Permainan Adjacency Strategy Termodifikasi

Enrique Alifio Ditya[1], Ariel Jovananda[2], M. Zulfiansyah Bayu Pratama[3], Jauza Lathifah Annassalafi[4]

[1]Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, Bandung, 40116, email: 13521142@std.stei.itb.ac.id
[2]Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, Bandung, 40116, email: 13521086@std.stei.itb.ac.id
[3]Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, Bandung, 40116, email: 13521028@std.stei.itb.ac.id
[4]Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, Bandung, 40116, email: 13521030@std.stei.itb.ac.id

INTISARI — Makalah ini membahas penggunaan *adversarial search* untuk mencari sekuens pergerakan optimal dalam permainan *adjacency strategy* termodifikasi. *Adjacency strategy* adalah permainan papan dua pemain yang dimainkan di papan 8x8. Pemain bergantian menempatkan simbol X atau O pada kotak kosong, dan simbol yang berdekatan dengan simbol pemain akan diubah menjadi simbol pemain tersebut. Pada makalah ini, diimplementasikan tiga algoritma *adversarial search* untuk permainan *adjacency strategy*: minimax, *local search*, dan algoritma genetika. Makalah ini juga mengandung analisis serangkaian pertandingan antara bot yang menggunakan algoritma *adversarial search* dengan manusia. Hasil pertandingan menunjukkan bahwa bot minimax dan alpha beta pruning dapat mengalahkan manusia dalam sebagian besar pertandingan. Meskipun itu, dianalisis pula probabilitas kemenangan dari pemain bergantung kepada yang melakukan pergerakan pertama.

KATA KUNCI — *Adjacency Strategy Game, Adversarial Search, MiniMax, Alpha-beta Pruning, Local Search*, *Genetic Algorithm*.

I. PENDAHULUAN

Dalam era perkembangan teknologi informasi yang begitu pesat, permainan komputer menjadi salah satu hiburan yang sangat populer di kalangan masyarakat. Salah satu aspek yang menarik dari permainan komputer adalah kemampuan untuk merancang algoritma cerdas yang dapat berkompetisi dengan pemain manusia. Seiring dengan itu, penelitian dalam bidang kecerdasan buatan untuk mengembangkan strategi permainan semakin menarik perhatian. Salah satu permainan yang menjadi objek penelitian ini adalah *Adjacency Strategy Game*.

Adjacency Strategy Game adalah permainan papan interaktif yang dimainkan antara dua pemain dengan menggunakan X dan O pada papan 8x8 yang ditampilkan di layar komputer. Tujuan dari permainan ini adalah untuk memperoleh lebih banyak X daripada O atau sebaliknya setelah sejumlah putaran tertentu. Permainan ini dimulai dengan penempatan 4 X di sudut kiri bawah dan 4 O di sudut kanan atas papan. Setiap pemain bergantian menempatkan X atau O pada kotak kosong, dan ketika seorang pemain adalah X, semua O yang berada di kotak-kotak yang bersebelahan akan digantikan dengan X, dan sebaliknya.

Makalah ini bertujuan untuk menjelaskan metode yang digunakan dalam mengembangkan agen kecerdasan buatan (bot) untuk bermain *Adjacency Strategy Game*. Akan dijelaskan *objective function* yang digunakan, proses pencarian dengan metode minimax dengan alpha-beta pruning, algoritma *local search*, serta algoritma genetika yang diimplementasikan.

Akan dievaluasi pula kinerja bot yang dikembangkan melalui serangkaian pertandingan, termasuk pertandingan antara bot dan pemain manusia, serta pertandingan antara berbagai jenis bot. Setelah itu, dicatat jumlah kemenangan dan kekalahan dalam setiap jenis pertandingan untuk mengukur persentase kemenangan masing-masing bot.

Selain itu, makalah ini akan memberikan penjelasan tentang percobaan yang dilakukan, termasuk jumlah ronde dan alasan di balik pengaturan percobaan. Di akhir, dicantumkan kontribusi masing-masing anggota dalam kelompok penelitian ini.

II. IMPLEMENTASI

FUNGSI OBJEKTIF

Fungsi objektif merupakan salah satu komponen penting dalam pengembangan agen kecerdasan buatan untuk permainan *Adjacency Strategy Game*. Fungsi ini memiliki peran vital dalam membantu agen untuk mengevaluasi keadaan papan permainan pada suatu titik tertentu, sehingga agen dapat membuat keputusan yang lebih cerdas dalam permainan.

Fungsi objektif yang digunakan dalam permainan ini adalah perhitungan sederhana yang didasarkan pada selisih antara jumlah bidak milik agen dengan jumlah bidak milik lawan pada papan permainan. Fungsi ini dirumuskan dengan menghitung jumlah bidak yang dimiliki oleh pemain yang sedang menjalankan giliran dan jumlah bidak yang dimiliki oleh pemain lawan. Selanjutnya, nilai objektif dihitung sebagai selisih antara jumlah bidak pemain dengan jumlah bidak lawan.

Tujuan dari fungsi objektif ini adalah untuk memberikan agen pedoman dalam mengambil keputusan selama permainan. Dengan mengoptimalkan nilai fungsi objektif, agen dapat mencapai tujuan optimalnya, yang dalam konteks permainan ini adalah untuk memiliki lebih banyak bidak daripada lawan setelah sejumlah putaran tertentu. Dengan kata lain, agen berusaha untuk memaksimalkan selisih antara jumlah bidaknya dengan jumlah bidak lawan, sehingga mencapai posisi yang lebih menguntungkan dalam permainan.

Khusus pada algoritma MiniMax, fungsi objektif ini juga dapat diminimalkan agar menjadi *favourable* untuk pemain yang berperan sebagai *minimizer*. Selisih yang semakin negatif menandakan *minimizer* memiliki keunggulan yang semakin besar pada permainan.

MINIMAX

1. Ide Dasar

Algoritma minimax adalah sebuah pendekatan untuk pengambilan keputusan dalam permainan adversarial antara dua pemain. Tujuan utamanya adalah menemukan langkah terbaik yang mungkin dilakukan oleh seorang pemain dengan mempertimbangkan langkah lawan serta berusaha memaksimalkan fungsi objektif. Secara dasarnya, algoritma ini bekerja dengan menjelajahi pohon permainan dan mengevaluasi posisi permainan untuk menentukan langkah yang optimal.

Ide utama dari algoritma minimax datang dari asumsi bahwa lawan bermain secara optimal, yakni selalu memilih langkah yang memaksimalkan peluang untuk menang. Prinsip ini diterapkan secara rekursif di sepanjang pohon permainan, bergantian antara *maximizer* dan *minimizer*.

Permainan direpresentasikan sebagai sebuah pohon, dengan setiap simpul merepresentasikan keadaan permainan yang mungkin terjadi. Simpul akar adalah kondisi permainan saat ini, dan anak-anak dari simpul akar mewakili seluruh kemungkinan gerakan yang dapat dilakukan oleh pemain saat ini. Pohon ini akan bertumbuh saat algoritma mengeksplorasi berbagai gerakan dan gerakan balasan. Pada simpul daun pohon, fungsi evaluasi digunakan untuk menilai state permainan. Fungsi ini memberikan perkiraan seberapa baik posisi tersebut bagi pemain.

2. Alpha-Beta Pruning

Alpha-beta pruning merupakan teknik pengoptimalan dalam algoritma minimax yang membantu mengurangi jumlah node yang dievaluasi dalam pohon permainan. Teknik ini bekerja dengan menyimpan dua nilai: alpha dan beta, yang mewakili nilai terbaik yang ditemukan sejauh ini untuk pemain *maximizer* dan *minimizer*. Ketika sebuah cabang dalam pohon terbukti lebih buruk daripada cabang yang telah dieksplorasi sebelumnya, maka cabang tersebut akan dipangkas sehingga mengurangi ruang pencarian.

Secara arti, alpha mewakili batas bawah maksimum pada skor yang mungkin dicapai oleh pemain *maximizer*. Pada sisi lain, beta mewakili batas atas minimum pada skor yang mungkin untuk pemain *minimizer*. Dengan membandingkan nilai alpha dan beta selama pencarian, algoritma dapat memangkas cabang yang tidak akan mempengaruhi keputusan akhir, sehingga pencarian menjadi lebih efisien.

3. MiniMax pada *Adjacency Strategy Game*

Langkah pertama yang dilakukan untuk mengimplementasikan algoritma minimax pada permainan adjacency strategy adalah mendefinisikan fungsi-fungsi penegak peraturan permainan. Awalnya, papan permainan diinisialisasi sesuai peraturan, yakni empat marka X pada pojok kiri bawah dan empat marka O pada pojok kanan atas papan. Setelahnya, definisikan fungsi untuk menghasilkan semua kemungkinan langkah legal untuk pemain saat ini. Dalam permainan ini, langkah legal terdiri dari seluruh kotak kosong yang ada pada papan. Terakhir, definisikan fungsi objektif. Fungsi ini akan dipanggil untuk setiap simpul daun pada *game tree*.

Langkah kedua adalah mengimplementasikan algoritma minimax pada permainan. Berikut merupakan cara kerja algoritma minimax untuk game ini:

a. Definisikan sebuah fungsi rekursif

```
minimax(board, depth, alpha, beta, isMaximizingPlayer)
```

dengan:

- `board` merepresentasikan keadaan permainan saat ini
- `depth` merepresentasikan kedalaman saat ini dalam pohon pencarian
- `isMaximizingPlayer` merepresentasikan sebuah flag boolean yang mengindikasikan apakah pemain saat ini sedang mencoba untuk memaksimalkan skor
- `alpha` dan `beta` merepresentasikan nilai terbaik yang ditemukan sejauh ini untuk pemain maximizer dan minimizer.

b. Pada setiap pemanggilan rekursif, periksa apakah permainan telah selesai (papan telah penuh atau mencapai batas putaran yang ditetapkan) atau apabila kedalaman maksimum telah tercapai. Jika demikian, kembalikan skor evaluasi papan.

c. Jika pemain saat ini adalah pemain yang memaksimalkan (isMaximizingPlayer bernilai true), lakukan iterasi melalui semua langkah yang valid, terapkan setiap langkah ke papan, dan panggil secara rekursif minimax untuk pemain lawan (isMaximizingPlayer bernilai false). Perbarui nilai alpha dengan nilai maksimum dari alpha saat ini dan skor anak. Jika beta ≤ alpha, lakukan break dari loop (alpha-beta pruning).

d. Jika pemain saat ini adalah pemain yang meminimalkan (isMaximizingPlayer bernilai false), lakukan hal yang sama seperti pada langkah c, tetapi perbarui nilai beta dengan nilai minimum dari beta saat ini dan skor anak. Seperti sebelumnya, terapkan alpha-beta pruning dengan break dari loop lebih awal jika beta ≤ alpha.

e. Terakhir, kembalikan skor terbaik yang ditemukan. Jika maximizingPlayer bernilai True, kembalikan skor maksimum; jika maximizingPlayer bernilai False, kembalikan skor minimum.

LOCAL SEARCH

Pengimplementasian *Local Search* pada permainan *Adjacency Strategy Game* menggunakan *Local Search* versi *Steepest Hill Climbing*, agar *bot* selalu memilih opsi gerakan

yang menghasilkan jumlah simbolnya yang terbanyak ketika *bot* meletakkan simbolnya pada suatu bidak pada papan permainan.

Penerapan strategi *Steepest Hill Climbing* dimulai dengan melakukan iterasi sebanyak kotak kosong pada papan permainan. Pada setiap iterasi diperiksa apakah kotak tersebut menghasilkan fungsi objektif terbesar, jika kotak tersebut menghasilkan fungsi objektif terbesar maka simpan koordinat kotak tersebut pada suatu variabel. Ulangi proses pemeriksaan kotak kosong sampai semua *successor* telah diperiksa.

GENETIC ALGORITHM

Pengimplementasian *Genetic Algorithm* dilakukan dalam beberapa langkah. Pada langkah pertama harus didefinisikan terlebih dahulu jumlah generasi maksimum, kedalaman *game search tree, board evaluation function*, dan tingkat *mutation*. Selanjutnya, inisialisasi *reservation tree* dengan nilai *root node* $-\infty$.

Langkah ketiga merupakan menghasilkan N jumlah populasi kromosom acak dan mengevaluasi setiap nilai daun menggunakan *board evaluation function* yang telah didefinisikan, langkah selanjutnya adalah untuk meletakkan N populasi tersebut kepada *reservation tree* yang telah dibuat dan gunakan operasi *minimax* untuk memperbaharui nilai-nilai *node* yang ada pada *reservation tree*. Operasi minimax digunakan untuk mencari para *parents* yang memiliki nilai *fitness function* tertinggi (*H-K+1*).

Langkah selanjutnya adalah untuk melakukan *crossover* setelah *consecutive sequences*, dan lakukan *mutation* sesuai tingkatnya. Langkah keenam adalah untuk menghitung lagi nilai setiap *node* daun pada kromosom keturunan. Langkah selanjutnya adalah untuk meletakkan lagi kromosom *parents* dan kromosom keturunan-keturunannya pada *reservation tree* dan ulangi operasi *minmax* agar para *node* di *reservation tree* akan diperbaharui.

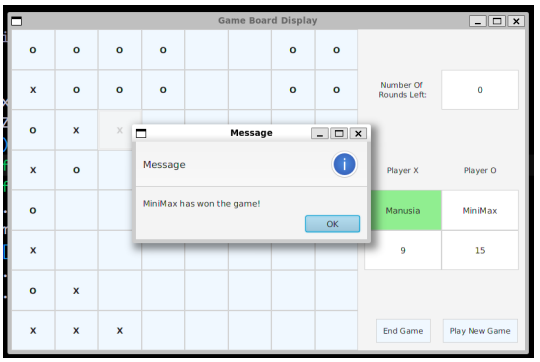
Langkah selanjutnya adalah untuk menghitung ulang *fitness function* setiap individu dan pilih lagi N jumlah kandidat *parents* yang memiliki *fitness function* terbaik. Ulangi langkah-langkah yang telah disebutkan sampai mencapai jumlah generasi maksimum.

III. ANALISIS

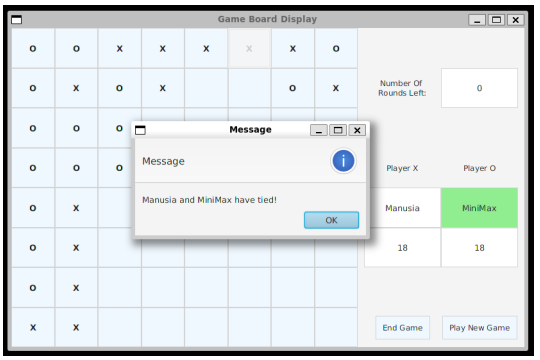
Analisis dilakukan dengan melakukan pertandingan antar dua pemain, yaitu pertandingan bot dengan manusia atau pertandingan sesama bot dengan jenis bot yang berbeda. Pertandingan dilakukan dengan jumlah ronde tertentu atau sampai *board* penuh. Setiap jenis pertandingan, akan dicatat jumlah kemenangan dan kekalahan sehingga didapatkan persentase kemenangan yang dimiliki oleh setiap jenis bot.

BOT MINIMAX VS MANUSIA

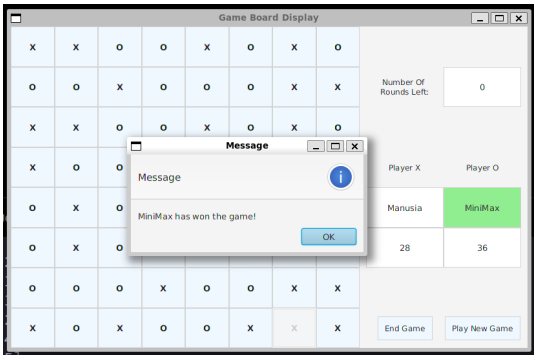
Pertandingan dilakukan antara bot minimax dengan manusia sebanyak 3 kali. Pada pertandingan ini, pemain X adalah manusia dan pemain O adalah bot minimax. Pertandingan pertama dilakukan dalam 8 ronde dengan hasil pemain O (bot minimax) berhasil mengalahkan pemain X (manusia). Pertandingan kedua dilakukan dalam 14 ronde dengan hasil seri antara kedua pemain. Pertandingan ketiga dilakukan hingga board penuh dengan hasil pemain O (bot minimax) berhasil mengalahkan pemain X (manusia). Berdasarkan hasil dari tiga pertandingan tersebut, pemain O (bot minimax) berhasil memenangkan 2 pertandingan melawan pemain X (manusia), sementara satu pertandingan berakhir dengan hasil seri, sehingga untuk bot minimax memiliki persentase kemenangan sekitar 66.67%.



Gambar 4.1.1: Pertandingan Pertama Bot Minimax Vs Manusia Sebanyak 8 Ronde.



Gambar 4.1.2: Pertandingan Kedua Bot Minimax Vs Manusia Sebanyak 14 Ronde.

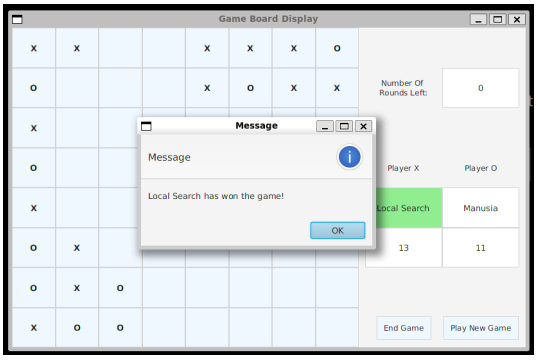


Gambar 4.1.3: Pertandingan Ketiga Bot Minimax Vs Manusia Hingga *Board* Penuh.

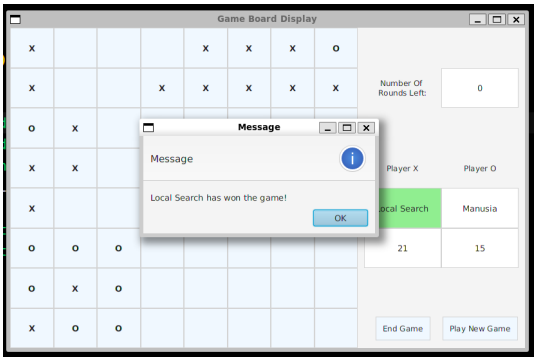
BOT LOCAL SEARCH VS MANUSIA

Pertandingan dilakukan antara bot local search dengan manusia sebanyak 3 kali. Pada pertandingan ini, pemain X adalah bot local search dan pemain O adalah manusia. Pertandingan pertama dilakukan dalam 8 ronde dengan hasil pemain X (bot local search) berhasil mengalahkan pemain O

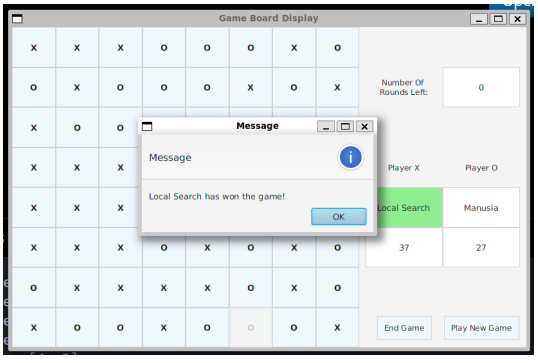
(manusia). Pertandingan kedua dilakukan dalam 14 ronde dengan hasil pemain X (bot local search) berhasil mengalahkan pemain O (manusia). Pertandingan ketiga dilakukan hingga board penuh dengan hasil pemain X (bot local search) berhasil mengalahkan pemain O (manusia). Berdasarkan hasil pertandingan tersebut, pemain X (bot local search) berhasil memenangkan setiap pertandingan melawan pemain O (manusia), maka persentase kemenangan untuk bot local search adalah 100%.



Gambar 4.2.1: Pertandingan Pertama Bot *Local Search* Vs Manusia Sebanyak 8 Ronde.



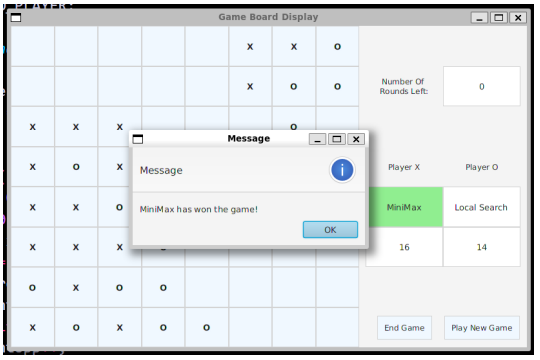
Gambar 4.2.2: Pertandingan Pertama Bot *Local Search* Vs Manusia Sebanyak 14 Ronde.



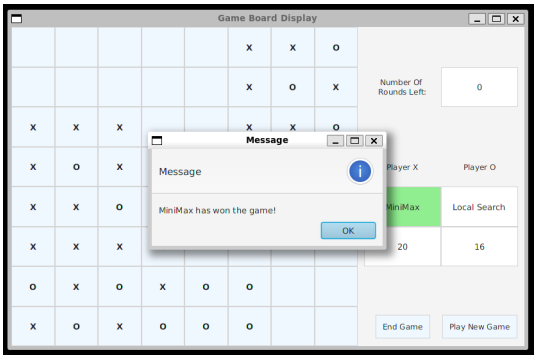
Gambar 4.2.3: Pertandingan Pertama Bot *Local Search* Vs Manusia Hingga *Board* Penuh.

BOT MINIMAX VS BOT LOCAL SEARCH

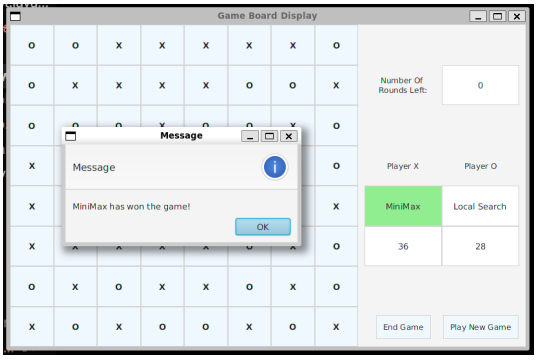
Pertandingan dilakukan antara bot minimax dengan bot local search sebanyak 3 kali. Pada pertandingan ini, pemain X adalah bot minimax dan pemain O adalah bot local search. Pertandingan pertama dilakukan dalam 11 ronde dengan hasil pemain X (bot minimax) berhasil mengalahkan pemain O (bot local search). Pertandingan kedua dilakukan dalam 14 ronde dengan hasil pemain X (bot minimax) berhasil mengalahkan pemain O (bot local search). Pertandingan ketiga dilakukan hingga board penuh dengan hasil pemain X (bot minimax) berhasil mengalahkan pemain O (bot local search). Berdasarkan hasil pertandingan tersebut, pemain X (bot minimax) berhasil memenangkan setiap pertandingan melawan pemain O (bot local search), maka persentase kemenangan untuk bot minimax adalah 100%, sementara bot local search tidak berhasil memenangkan satu pun dari pertandingan tersebut, sehingga persentase kemenangan untuk bot local search adalah 0%.



Gambar 4.3.1: Pertandingan Pertama Bot Minimax Vs Bot Local Search Sebanyak 11 Ronde.



Gambar 4.3.2: Pertandingan Kedua Bot Minimax Vs Bot Local Search Sebanyak 14 Ronde.

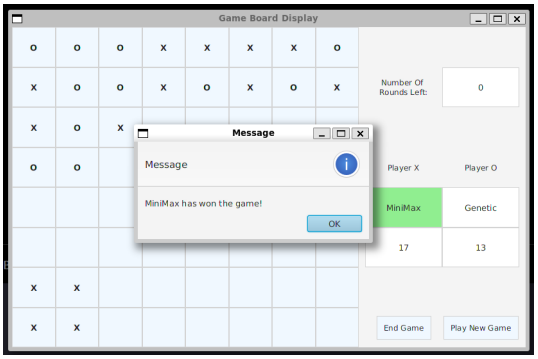


Gambar 4.3.3: Pertandingan Ketiga Bot Minimax Vs Bot Local Search Hingga Board Penuh.

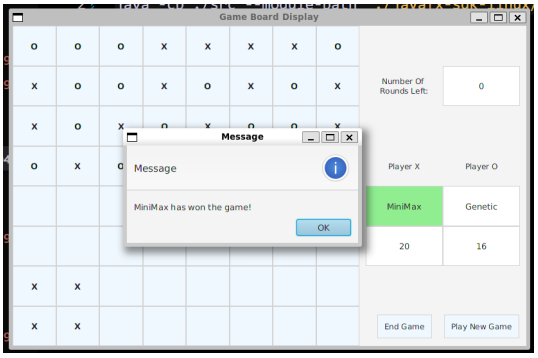
BOT MINIMAX VS BOT GENETIC ALGORITHM

Pertandingan dilakukan antara bot minimax dengan bot genetic algorithm sebanyak 3 kali. Pada pertandingan ini,

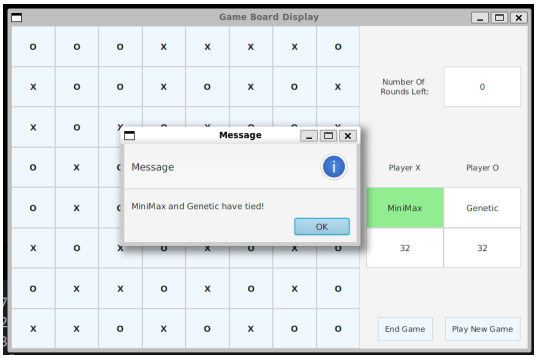
pemain X adalah bot minimax dan pemain O adalah bot genetic algorithm. Pertandingan pertama dilakukan dalam 11 ronde dengan hasil pemain X (bot minimax) berhasil mengalahkan pemain O (bot genetic algorithm). Pertandingan kedua dilakukan dalam 14 ronde dengan hasil pemain X (bot minimax) berhasil mengalahkan pemain O (bot genetic algorithm). Pertandingan ketiga dilakukan hingga board penuh dengan hasil seri antara kedua pemain. Berdasarkan hasil dari tiga pertandingan tersebut, pemain X (bot minimax) berhasil memenangkan 2 pertandingan melawan pemain O (bot genetic algorithm), sementara satu pertandingan berakhir dengan hasil seri, sehingga untuk bot minimax memiliki persentase kemenangan sekitar 66.67% dan 0% kemenangan untuk bot genetic algorithm.



Gambar 4.4.1: Pertandingan Pertama Bot Minimax Vs Bot Genetic Algorithm Sebanyak 11 Ronde.



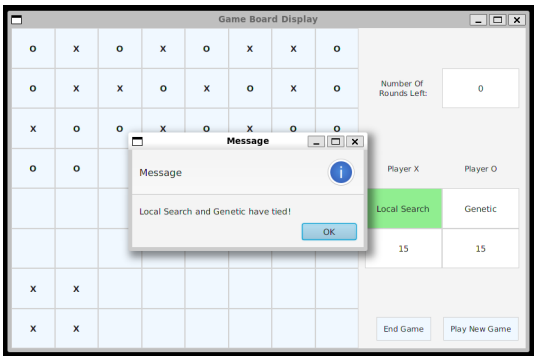
Gambar 4.4.2: Pertandingan Pertama Bot Minimax Vs Bot Genetic Algorithm Sebanyak 14 Ronde.



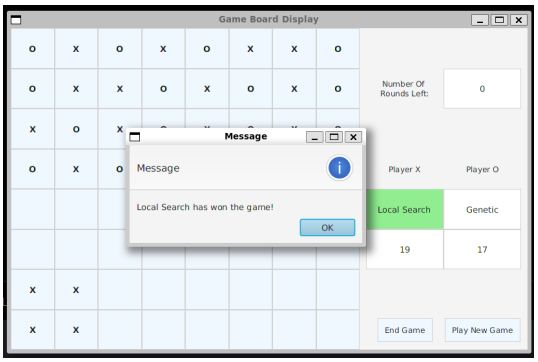
Gambar 4.4.3: Pertandingan Pertama Bot Minimax Vs Bot Genetic Algorithm Hingga *Board* Penuh.

BOT LOCAL SEARCH VS BOT GENETIC ALGORITHM

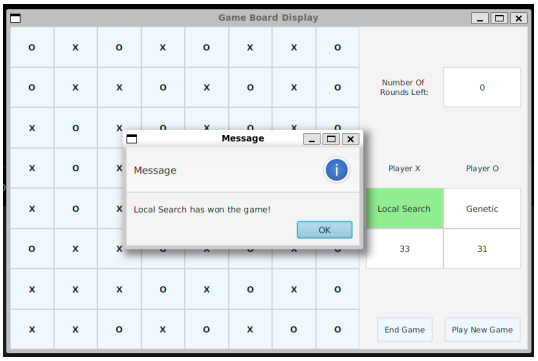
Pertandingan dilakukan antara bot local search dengan bot genetic algorithm sebanyak 3 kali. Pada pertandingan ini, pemain X adalah bot local search dan pemain O adalah bot genetic algorithm. Pertandingan pertama dilakukan dalam 11 ronde dengan hasil seri antara kedua pemain. Pertandingan kedua dilakukan dalam 14 ronde dengan hasil pemain X (bot local search) berhasil mengalahkan pemain O (bot genetic algorithm). Pertandingan ketiga dilakukan hingga board penuh dengan hasil pemain X (bot local search) berhasil mengalahkan pemain O (bot genetic algorithm). Berdasarkan hasil dari tiga pertandingan tersebut, pemain X (bot *local search*) berhasil memenangkan 2 pertandingan melawan pemain O (bot genetic algorithm), sementara satu pertandingan berakhir dengan hasil seri, sehingga untuk bot minimax memiliki persentase kemenangan sekitar 66.67% dan 0% kemenangan untuk bot genetic algorithm.



Gambar 4.5.1: Pertandingan Pertama Bot *Local Search* Vs Bot Genetic Algorithm Sebanyak 11 Ronde.



Gambar 4.5.2: Pertandingan Pertama Bot *Local Search* Vs Bot Genetic Algorithm Sebanyak 14 Ronde.



Gambar 4.5.3: Pertandingan Pertama Bot *Local Search* Vs Bot Genetic Algorithm Hingga *Board* Penuh.

TABEL KONTRIBUSI

PEKERJAAN	TIPE	KONTRIBUTOR
Environment	Kode	Enrique Alifio Ditya
MiniMax	Kode	Enrique Alifio Ditya
Local Search	Kode	Enrique Alifio Ditya
Genetic Algorithm	Kode	M. Zulfiansyah Bayu Pratama
Pendahuluan	Laporan	Enrique Alifio Ditya
Fungsi Objektif	Laporan	Enrique Alifio Ditya
MiniMax	Laporan	Enrique Alifio Ditya
Local Search	Laporan	Ariel Jovananda
Genetic Algorithm	Laporan	Ariel Jovananda
Analisis	Laporan	Jauza Lathifah Annassalafi

Tabel 4.1: Tabel Kontribusi

REFERENSI

[1] Hong, T.-P., Huang, K.-Y., & Lin, W.-Y. (2003). A genetic minimax game-playing strategy. Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics (Vol. 2, pp. 1992-1997). IEEE.

LAMPIRAN

Repositori kode sumber:
https://github.com/AlifioDitya/Adjacency-Game-AI