

LAPORAN TUBES 1
“Adversarial Adjacency Strategy Game”

IF3170 Intelegensi Buatan



Disusun oleh :

Laila Bilbina Khoiru Nisa	13521016
Ilham Akbar	13521068
Yanuar Sano Nur Rasyid	13521110
Rayhan Hanif Maulana Pradana	13521112

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

Daftar Isi

Daftar Isi.....	2
1. Latar Belakang.....	3
2. Objective Function.....	4
3. Minimax dengan Alpha Beta Pruning.....	4
4. Local Search.....	5
5. Genetic Algorithm.....	6
6. Hasil Pertandingan.....	8
7. Kontribusi.....	15
8. Lampiran.....	15

1. Latar Belakang

Dalam era modern ini, Intelegensi Buatan (AI) telah menjadi salah satu bidang penelitian yang paling dinamis dan relevan di dunia teknologi. AI memberikan kemampuan komputer untuk mengambil keputusan, belajar dari pengalaman, dan secara cerdas berinteraksi dengan lingkungan sekitarnya. Salah satu aplikasi yang paling menonjol dalam AI adalah pengembangan sistem cerdas untuk bermain permainan, yang telah menjadi subjek penelitian yang sangat menarik dalam beberapa dekade terakhir.

Permainan adalah representasi yang baik dari masalah pengambilan keputusan strategis yang kompleks. Dalam konteks permainan, komputer atau pemain AI ditantang untuk membuat keputusan yang optimal untuk mencapai tujuan tertentu, sering kali dengan keterbatasan sumber daya.

Namun, selain permainan catur, masih ada berbagai permainan strategi lainnya yang menawarkan tantangan intelektual yang sama. Dalam konteks tugas besar ini, kami akan memfokuskan perhatian pada permainan strategi yang disebut "Adjacency Strategy Game." Permainan ini melibatkan dua pemain yang bersaing untuk menguasai papan permainan dengan meletakkan tanda mereka dan mengubah tanda-tanda lawan. Tujuan akhirnya adalah untuk memiliki tanda yang lebih banyak daripada lawan pada akhir permainan.

Algoritma Minimax adalah salah satu pendekatan yang paling sering digunakan dalam permainan untuk mengambil keputusan yang optimal. Ini didasarkan pada prinsip pemikiran strategis, di mana pemain mencoba memaksimalkan keuntungan mereka dan meminimalkan kerugian potensial. Pemangkasan Alpha-Beta adalah teknik yang mengoptimalkan Algoritma Minimax dengan mengurangi jumlah simpul yang harus dievaluasi, sehingga meningkatkan efisiensi permainan.

Tugas besar ini bertujuan untuk mengetahui penerapan Algoritma Minimax dan Teknik Pemangkasan Alpha-Beta dalam permainan Adjacency Strategy Game. Dengan melakukan hal ini, kami berharap untuk memahami kinerja dan efektivitas Algoritma Minimax dan Pemangkasan Alpha-Beta dalam konteks permainan ini. Hasil penelitian ini dapat membantu memahami bagaimana teknik-teknik ini dapat diterapkan dalam berbagai permainan strategi dan membantu pengembangan AI yang lebih cerdas dalam konteks permainan yang beragam.

2. Objective Function

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/*marker* 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 bidak yang dimiliki oleh pemain lawan, pada setiap kemungkinan posisi ke depan yang telah ditentukan. 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.

3. Minimax dengan Alpha Beta Pruning

Proses pencarian dengan menggunakan algoritma Minimax dengan Alpha-Beta Pruning pada permainan *Adjacency Strategy Game* dengan papan ukuran 8x8 bertujuan untuk menentukan langkah optimal peletakan bidak bagi bot. Dalam konteks ini, bot mencoba untuk memaksimalkan skor dalam permainan dengan mengantisipasi langkah lawan. Berikut adalah langkah-langkah proses pencarian menggunakan Minimax dengan Alpha-Beta Pruning pada permainan *Adjacency Strategy Game*:

1. Pengembangan Tree

Tree akan memiliki node yang dinamakan sebagai state pada implementasi. Setiap state dapat memiliki sebuah parent/predecessor dan satu atau lebih child/successor. Setiap state juga menyimpan keadaan papan apabila serangkaian langkah permainan dilakukan untuk mencapai state tersebut. Atribut terakhir yang dimiliki oleh setiap state adalah value,

yang nantinya akan di-update saat algoritma minimax berjalan. State terminal dapat menghitung secara langsung valuenya, dengan menggunakan fungsi objektif yang telah didefinisikan sebelumnya.

2. Fungsi rekursif minimax

Algoritma minimax sendiri akan dijalankan menggunakan fungsi rekursif minimax. Pemanggilan untuk fungsi rekursif ini sendiri dibatasi oleh depth yang didefinisikan. Artinya, algoritma tidak akan mengeksplor semua keadaan state yang optimal kecuali depth tersebut didefinisikan sedalam 28, yaitu jumlah ronde maksimal pada permainan. Ketika pemanggilan fungsi mencapai kedalaman/depth yang telah didefinisikan, yang artinya traversal tree telah mencapai sebuah terminal state, fungsi akan meng-update value dari state tersebut sesuai pada bagian 1. Value tersebut kemudian akan dibandingkan dengan value pada state parent-nya, dan state parent akan menerima value tersebut apabila value yang disebutkan lebih besar dari value dari parent itu sendiri ketika MAX dan sebaliknya ketika MIN. Pada akhirnya, state root akan mendapatkan value optimal dan bot akan menjalankan next move sesuai child state dari root dengan value yang disebutkan.

3. Alpha-Beta Pruning

Pruning dilakukan melalui perbandingan antara masing-masing alpha dan beta dengan state child dari state yang sedang dikunjungi. Untuk alpha, dilakukan ketika MAX dengan membandingkan alpha state tersebut dengan value dari child state menggunakan fungsi max (artinya mengembalikan nilai yang lebih besar). Sebaliknya dilakukan untuk beta. Ketika beta kurang dari sama dengan alpha, fungsi yang sedang dipanggil akan dihentikan. Penggunaan alpha-beta pruning ini cukup membantu dalam efisiensi algoritma minimax yang diimplementasikan, dimana ketika alpha-beta pruning tidak diterapkan, depth yang feasible sebesar 3, sedangkan saat alpha-beta pruning diterapkan, depth yang feasible meningkat menjadi 4 atau bahkan 5.

4. Local Search

Algoritma Local Search yang digunakan adalah algoritma yang bersifat *greedy*. Algoritma ini hanya akan memilih kandidat yang memiliki pendapatan *score* tertinggi dari permainan. Cara pemilihan yang dilakukan algoritma local search adalah sebagai berikut.

1. Pemilihan Move Berdasarkan *Score*: menghitung banyaknya simbol musuh yang *adjacent*. Semakin banyak simbol musuh maka akan semakin tinggi poin yang didapatkan.
2. Pemilihan Move yang Menghasilkan Poin Permanen: Langkah ini dilakukan setelah langkah pertama dilakukan. Jika ada sebuah kotak yang dikonversi dan posisi *adjacent* kotak tersebut penuh maka musuh tidak dapat merubah simbol tersebut dan skor itu menjadi permanen.
3. Pemilihan Move dengan Cost Terendah: Jika dari langkah sebelumnya masih ada lebih dari satu kandidat, dilakukan perhitungan cost yaitu kemungkinan poin yang didapatkan musuh dari sekitar kotak kandidat *move* yang akan dipilih.

Alasan pemilihan algoritma *greedy* adalah karena algoritma ini sederhana untuk diimplementasikan dan cepat dalam memproses perhitungan karena perhitungan hanya dilakukan pada *state* ronde saat ini saja. Selain itu, algoritma ini menghasilkan hasil yang relatif bagus dibandingkan dengan algoritma lainnya karena fokus untuk mendapatkan poin tertinggi.

5. Genetic Algorithm

Algoritma Genetika (Genetic Algorithm, GA) adalah salah satu metode optimisasi yang terinspirasi dari mekanisme evolusi dalam alam. Dalam konteks permainan seperti Adjacency Strategy Game, GA dapat digunakan untuk mengembangkan strategi permainan yang cerdas untuk agen.

Berikut adalah cara Algoritma Genetika dapat diterapkan dalam Adjacency Strategy Game:

1. Representasi Kromosom: Setiap individu dalam populasi GA direpresentasikan sebagai kromosom, yang merupakan kumpulan langkah-langkah atau keputusan permainan. Dalam Adjacency Strategy Game, ini dapat berarti urutan tanda-tanda yang akan ditempatkan oleh agen.
2. Fungsi Fitness: Fungsi fitness digunakan untuk menilai sejauh mana kinerja strategi yang direpresentasikan oleh kromosom dalam mencapai tujuan permainan. Dalam kasus ini, tujuannya adalah untuk memiliki tanda yang lebih banyak daripada lawan pada akhir permainan. Oleh karena itu, fungsi fitness harus memperhitungkan jumlah tanda yang dimiliki oleh pemain setelah sejumlah langkah atau hingga akhir permainan.
3. Seleksi: Individu-individu yang memiliki kinerja terbaik, seperti yang ditentukan oleh fungsi fitness, dipilih untuk menghasilkan keturunan. Ini mensimulasikan seleksi alam yang menguntungkan individu yang paling cocok.
4. Reproduksi: Individu yang terpilih akan digunakan untuk menghasilkan keturunan dengan cara seperti crossover (penukaran informasi genetik antara kromosom) dan mutasi (perubahan acak pada kromosom). Dalam konteks permainan, ini mengacu pada penggabungan strategi dari individu yang berbeda dan mungkin adanya variasi dalam strategi tersebut.
5. Generasi Baru: Setelah reproduksi, generasi baru dari individu terbentuk. Proses ini dilakukan berulang kali dalam beberapa generasi untuk mengembangkan strategi yang semakin baik.
6. Evaluasi: Setiap generasi dievaluasi menggunakan fungsi fitness untuk menentukan sejauh mana perbaikan strategi. Proses ini dilakukan berulang kali hingga strategi yang memadai atau optimal ditemukan.
7. Terminasi: Permainan GA dapat berakhir setelah sejumlah generasi atau ketika strategi yang memadai ditemukan, tergantung pada aturan yang ditentukan sebelumnya.

6. Hasil Pertandingan

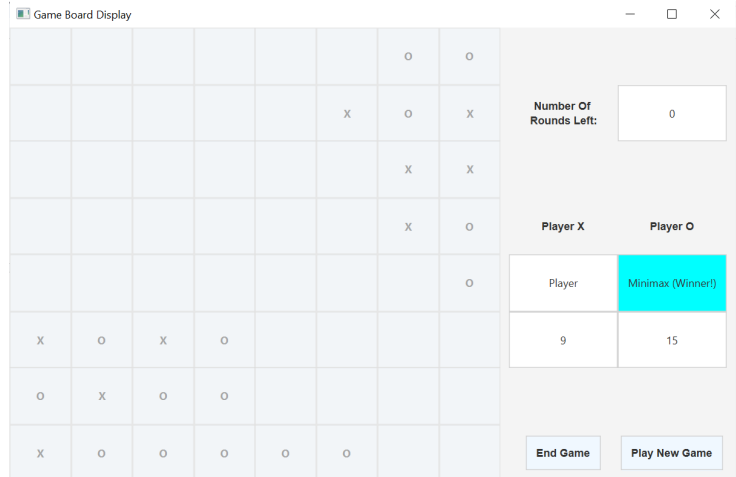
Pertandingan/Score	Penjelasan
<p>Bot minimax vs. manusia (sebanyak 5 kali)</p> <p>Score bot : 3</p> <p>Score manusia : 1</p>	 <p>Dengan 28 ronde, bot berhasil menang melawan player dengan perbedaan score 34 dan 30.</p>  <p>Dengan 24 ronde, bot seri dengan player.</p>

Game Board Display									
					X	X	O	Number Of Rounds Left:	0
						X	X		
				O		O	X		
X	X	O	X	O	O	O	X	Player X	Player O
X	O	X	O	X	O	O	X	Player	Minimax (Winner!)
X	O	O	O	O	O	O	X	23	25
O	X	X	X	X	X	X	X		
X	O	O	O	O	O	O	O	End Game	Play New Game

Dengan 20 ronde, bot berhasil menang melawan player dengan perbedaan score 25 dan 23

Game Board Display									
					X	X	O	Number Of Rounds Left:	0
					X	X	X		
							X		
								Player X	Player O
			X	O	O	O		Player (Winner!)	Minimax
X	X	X	O	O	X	X	O	19	15
O	X	X	X	X	O	O	O		
X	O	O	O	O	X	X		End Game	Play New Game

Dengan 13 ronde, player berhasil menang melawan bot dengan perbedaan score 19 dan 15.



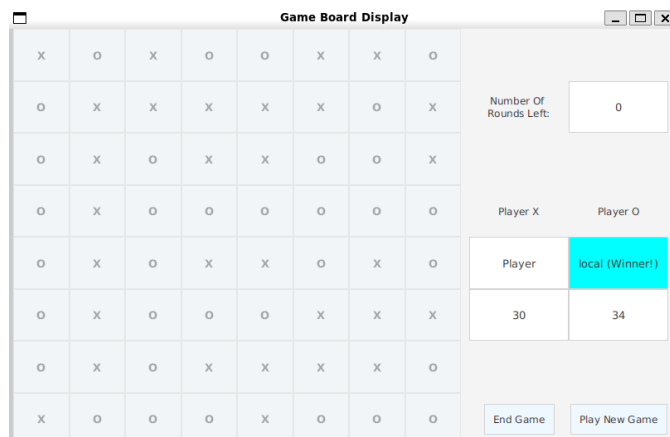
Dengan 8 ronde, bot berhasil menang melawan player dengan perbedaan score 15 dan 9.

Bot minimax memiliki kemungkinan kalah atau seri dengan player karena *depth* dari algoritma minimax yang ditunjukkan pada contoh dengan 13 ronde dan 24 ronde, karena *depth* dari algoritma minimax yang diimplementasikan terbatas pada 4. Konsekuensinya, bot tidak dapat melihat keseluruhan keadaan dari papan yang mungkin dan dapat mengambil langkah yang kurang optimum dibandingkan jika bot memiliki *depth* yang lebih dalam.

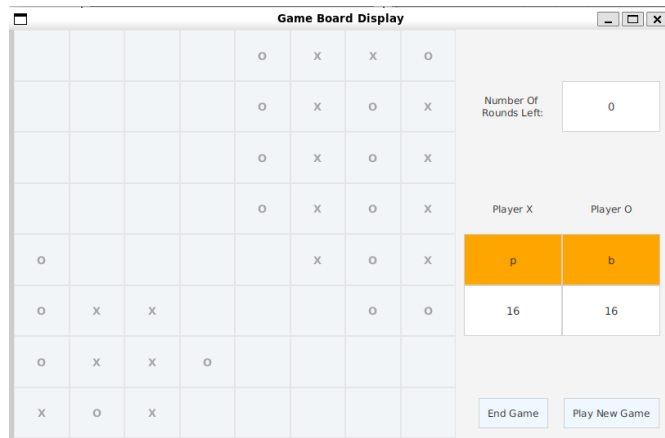
Bot local search vs. manusia
(sebanyak 3 kali)

Score bot : 1

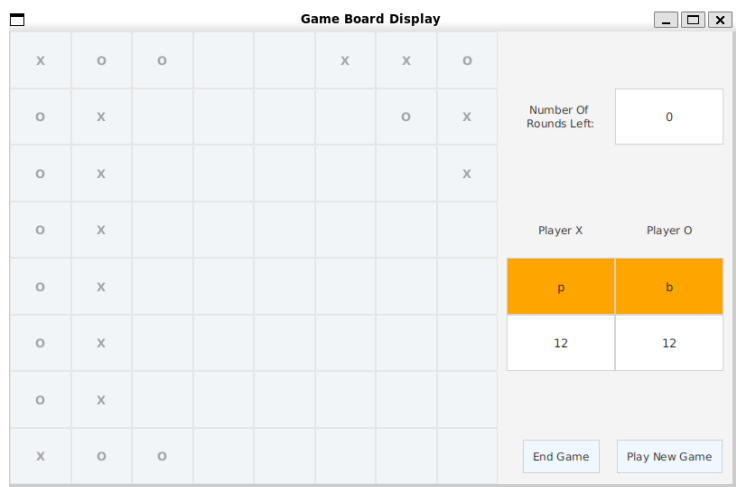
Score manusia : 0



Dengan 28 ronde bot berhasil menang melawan player dengan perbedaan score 34 dan 30.



Dengan 12 ronde bot seri dengan player.



Dengan 8 ronde bot seri dengan player.

Permainan dengan 28 ronde player kalah karena semakin banyak ronde semakin banyak kemungkinan move yang mungkin dimainkan dan memungkinkan player membuat *move* yang buruk dibandingkan dengan bot yang selalu memilih move yang mendapatkan poin terbanyak. Sementara itu, pada percobaan lain terjadi seri karena move yang diperhitungkan lebih sedikit.

Bot minimax vs bot local search
(sebanyak 3 kali)

Score bot Minimax : 3

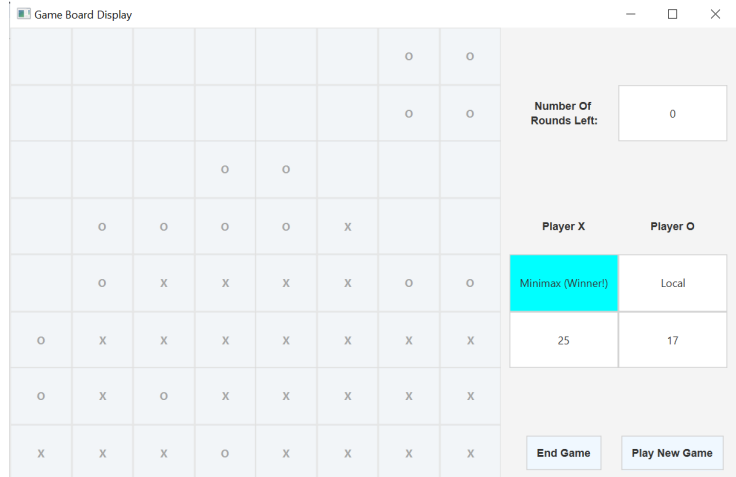
Score bot Local Search : 0

Game Board Display								Number Of Rounds Left: 0	
O	O	X	O	O	O	X	O	Player X Player O	
O	X	X	O	O	X	O	O		
O	X	X	O	X	X	X	O	Minimax (Winner!) Local	
X	X	X	O	X	X	X	X		
O	X	X	X	O	X	X	X	44	20
X	X	X	X	X	X	X	X	End Game Play New Game	
O	X	O	X	X	X	X	X		
X	X	X	O	X	X	X	X		

Dengan 28 ronde, bot Minimax berhasil mengalahkan bot Local Search dengan skor 44 dan 20.

Game Board Display								Number Of Rounds Left: 0	
						O	O	Player X Player O	
						O	O		
						O	X	Minimax (Winner!) Local	
O	O	O	O	O	X	X	X		
O	X	X	X	O	X	X	X	31	17
X	X	X	X	X	X	X	X	End Game Play New Game	
O	X	O	X	X	X	X	X		
X	X	X	O	X	X	X	X		

Dengan 19 ronde, bot Minimax berhasil mengalahkan bot Local Search dengan skor 31 dan 17.



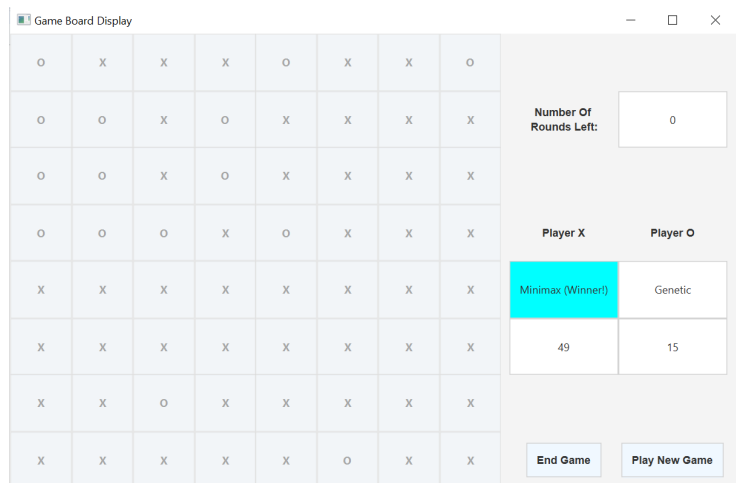
Dengan 17 ronde, bot Minimax berhasil mengalahkan bot Local Search dengan skor 25 dan 17.

Bot minimax berhasil mengalahkan bot local search dengan telak karena pada dasarnya algoritma minimax sendiri “melihat” state-state yang mungkin beberapa langkah ke depan dengan memanfaatkan tree. Sedangkan, algoritma local search hanya “melihat” state-state pada satu langkah ke depan, yaitu state apabila bot local search meletakkan bidaknya di papan.

Bot minimax vs bot genetic algorithm (sebanyak 3 kali)

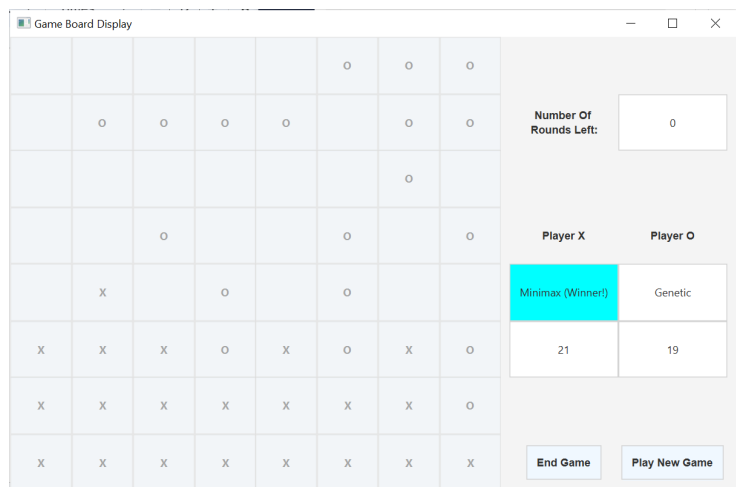
Score bot Minimax: 2

Score bot Genetic Algorithm: 0

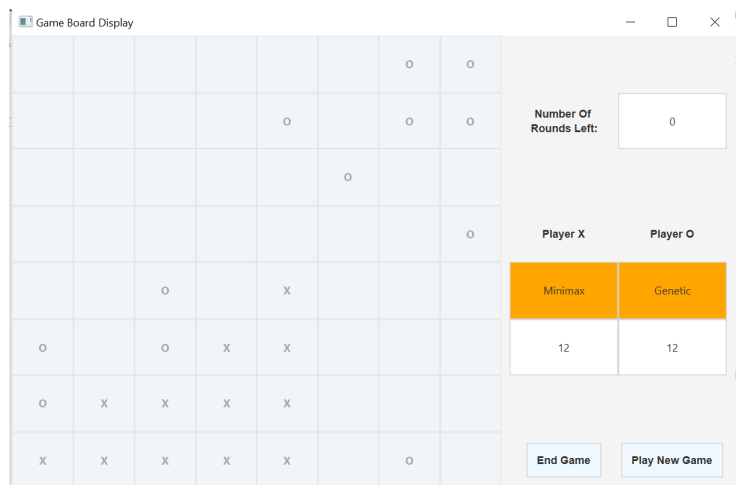


Dengan 28 ronde, bot Minimax berhasil mengalahkan

bot Genetic Algorithm dengan skor 49 dan 15.



Dengan 16 ronde, bot Minimax berhasil mengalahkan bot Genetic Algorithm dengan skor 21 dan 19.



Dengan 24 ronde, bot Minimax seri dengan bot Genetic Algorithm.

Bot Minimax dapat mengalahkan dan seri dengan bot Genetic Algorithm dikarenakan algoritma minimax sendiri “melihat” state-state yang mungkin beberapa langkah ke depan dengan memanfaatkan tree. Selain itu, implementasi algoritma Genetic belum lengkap.

Bot local search vs bot genetic
algorithm (sebanyak 3 kali)

Score bot Local Search :3

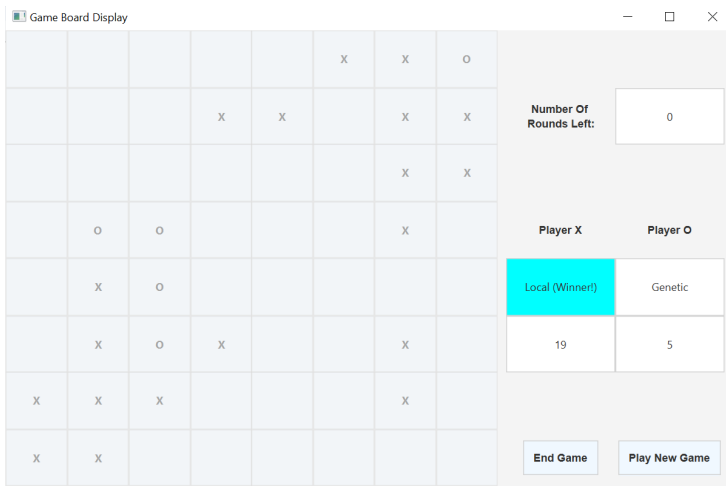
Score bot Genetic Algorithm :0

Game Board Display								Number Of Rounds Left: 0	
O	X	X	X	O	X	X	O	Player X Player O	
X	O	X	O	O	X	O	X		
X	X	X	O	O	O	X	X	Local (Winner!) Genetic	
X	O	X	X	O	O	X	X		
O	X	X	X	X	X	X	O	41	23
X	X	X	X	X	X	X	O	End Game Play New Game	
X	O	X	X	X	X	O	O		

28 Ronde bot Local Search berhasil mengalahkan bot
Genetic Algorithm dengan skor 41-23

Game Board Display								Number Of Rounds Left: 0	
O	O	X			X	X	O	Player X Player O	
		O	X		X	X	X		
X	X	X					X	Local (Winner!) Genetic	
O	X	O	X		O				
		X	X	X	X	X		27	9
				X	X	O		End Game Play New Game	
X	X			X	X	O			
X	X				X				

14 Ronde bot Local Search berhasil mengalahkan bot
Genetic Algorithm dengan skor 27-9

	 <p>8 Ronde bot Local Search berhasil mengalahkan bot Genetic Algorithm dengan skor 19-5</p> <p>Bot Local Search dapat mengalahkan dengan telak bot Genetic Algorithm dikarenakan algoritma local search hanya “melihat” state-state pada satu langkah ke depan, yaitu state apabila bot local search meletakkan bidaknya di papan. Selain itu implementasi algoritma Genetic belum lengkap.</p>
--	--

7. Kontribusi

Nama / NIM	Kontribusi
Laila Bilbina/13521016	Genetic Algorithm,Laporan
Ilham Akbar/13521068	Genetic Algorithm,Laporan
Yanuar Sano/13521110	Local Search
Rayhan Hanif/13521112	Minimax

8. Lampiran

Link Github : https://github.com/Ilhamgzzlr/Tubes-1-Adjacency_Strategy_Game