Tugas Besar I IF3170 Inteligensi Buatan

Minimax Algorithm dan Alpha Beta Pruning in Adjacency Strategy Game

Oleh:

Fazel Ginanda 13521098

Akhmad Setiawan 13521164

Irgiansyah Mondo 13521167

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

I. Penjelasan Objective Function

Objective function yang digunakan adalah penghitungan banyak marka yang dimiliki oleh salah satu pemain. Dalam implementasi tugas besar ini, kami mendefinisikan bot sebagai pemain O dan musuh yang dihadapi oleh bot tersebut sebagai pemain X. Oleh karena itu, objective function secara lebih rinci didefinisikan sebagai banyak marka yang dimiliki oleh pemain O. Penentuan objective function ini berdasar kepada aturan permainan ini. Aturan tersebut menyatakan bahwa pemenang dari permainan ini adalah pemain yang memiliki marka terbanyak pada papan.

```
public double evaluate(String[][] board) {
    double score = 0;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (board[i][j].equals(anObject: "O")){
                score += 1;
            }
        }
    }
    return score;
}
```

Implementasi objective function

II. Proses Pencarian dengan Minimax dan Alpha Beta Pruning

Algoritma minimax dengan alpha beta pruning melakukan pencarian berbasis depth-first search. Algoritma ini memiliki kompleksitas waktu eksponensial. Dengan mempertimbangkan banyaknya langkah untuk mencapai *goal state*, *branching factor,* serta batasan waktu berpikir bot yang ditentukan dalam spesifikasi tugas besar ini, kami menetapkan kedalaman pohon ruang status sebesar lima. Artinya, bot akan melakukan *move* dengan mempertimbangkan lima langkah ke depan.

Algoritma Minimax dengan Alpha dan Beta Pruning yang kami rancang menggunakan parameter-parameter sebagai berikut.

No	Parameter	Keterangan
1	State	Kondisi papan saat bot akan mulai

		melakukan pencarian
2	Depth	Kedalaman pohon pencarian yang menyatakan langkah yang diperiksa
3	Alpha	Nilai batas untuk pemain MAX
4	Beta	Nilai batas untuk pemain MIN
5	Timeout	Batas waktu berpikir bot yang ditentukan dalam spesifikasi tugas ini, yaitu 5 detik
6	Turn	String yang menyatakan giliran pemain pada suatu saat tertentu, yaitu “O” dan “X”

Langkah-langkah dalam proses pencarian menggunakan algoritma minimax dengan alpha beta pruning adalah sebagai berikut.

1. Inisialisasi kondisi awal pencarian. Parameter yang diinisialisasi terlebih dahulu adalah state dan waktu terjadinya timeout. Setelah itu, dilakukan pencarian seluruh move yang mungkin berdasarkan state saat itu. Ini diperlukan untuk menyediakan *fallback plan* berupa pemilihan langkah random apabila bot melebihi waktu *timeout* yang sudah ditentukan sebelumnya.
2. Lakukan iterasi sebanyak empat kali. Setiap iterasi, dilakukan pencarian langkah yang akan diambil pada kedalaman ke-i. Pencarian dimulai pada kedalaman ke-1.
3. Pada setiap iterasi, lakukan pemanggilan fungsi yang menjalankan algoritma minimax dengan alpha beta pruning. Fungsi ini kami beri nama abpruning. Fungsi ini mengembalikan array of Object. Elemen pertama dari array ini menyatakan value hasil evaluasi algoritma minimax dengan alpha beta pruning. Sementara itu, elemen kedua menyatakan *move* yang akan dijalankan.
4. Periksa apakah waktu saat ini sudah melebihi waktu timeout. Jika sudah melebihi waktu timeout, maka fungsi abpruning akan mengembalikan value bernilai sama dengan hasil evaluasi objective function pada saat itu serta mengembalikan move

bernilai null. Apabila waktu saat ini belum melebihi batas waktu timeout, maka lanjut ke tahap berikutnya.

5. Periksa apakah kedalaman saat ini sama dengan 0. Jika kedalaman sama dengan 0, maka panggil fungsi evaluate untuk mengevaluasi objective function. Kemudian, kembalikan value bernilai sama dengan objective function serta move bernilai null. Jika kedalaman tidak sama dengan 0, maka pencarian dilanjutkan ke tahap berikutnya.
6. Cari seluruh move yang mungkin berdasarkan state saat ini. Simpan seluruh move tersebut ke dalam array allMoves.
7. Periksa apakah saat ini merupakan giliran bot atau musuh. Jika saat ini adalah giliran bot, maka nilai dimaksimumkan. Akan tetapi, jika saat ini adalah giliran musuh, maka nilai akan diminimumkan.
8. Inisialisasi nilai value yang akan dikembalikan. Jika giliran saat ini adalah MAX, maka inisialisasi bestMaxScore dengan *negative infinity*. Sebaliknya, jika giliran saat ini adalah min, maka inisialisasi bestMinScore dengan positive infinity.
9. Lakukan pengulangan terhadap setiap move yang menjadi elemen dari array allMoves yang sudah didapatkan pada langkah 6. Pada setiap iterasi, untuk setiap move, lakukan aksi yang bersesuaian dengan move tersebut yaitu mengisi seluruh kotak yang terdampak dari move tersebut dengan marka yang sesuai. Kondisi papan setelah aksi tersebut dilakukan disimpan ke dalam variabel appliedBoardMax.
10. Lakukan pemanggilan kembali fungsi abpruning. Ini merupakan bagian rekursif. Fungsi ini dipanggil dengan beberapa perubahan parameter. Parameter state diisi dengan appliedBoardMax yaitu papan yang telah diperbarui setelah langkah 9 dilakukan. Kemudian, parameter depth diisi dengan depth-1. Ini bermakna bahwa pencarian selanjutnya akan dilakukan terhadap kedalaman sebelumnya. Parameter alpha, beta, dan timeout sama dengan sebelumnya. Kemudian, parameter turn diganti dengan pemain yang merupakan lawan dari pemain saat ini.
11. Lakukan pemeriksaan apakah value yang dikembalikan akan mengubah dari value saat ini. Jika memenuhi, maka ubah value saat ini menjadi value yang

dikembalikan tersebut. Selain itu, ubah juga move menjadi move yang saat ini menjadi bagian dari iterasi.

12. Perbarui nilai alpha atau beta jika terjadi perubahan nilai batas.
13. Jika nilai alpha sudah lebih dari atau sama dengan beta, maka hentikan pencarian karena node tersebut tidak akan dipilih dan tidak berpengaruh terhadap hasil akhir pencarian.
14. Fungsi abpruning mengembalikan value terbaik dan move terbaik dalam bentuk array of Object.

III. Penjelasan Algoritma Local Search Hill Climbing with Sideway Moves

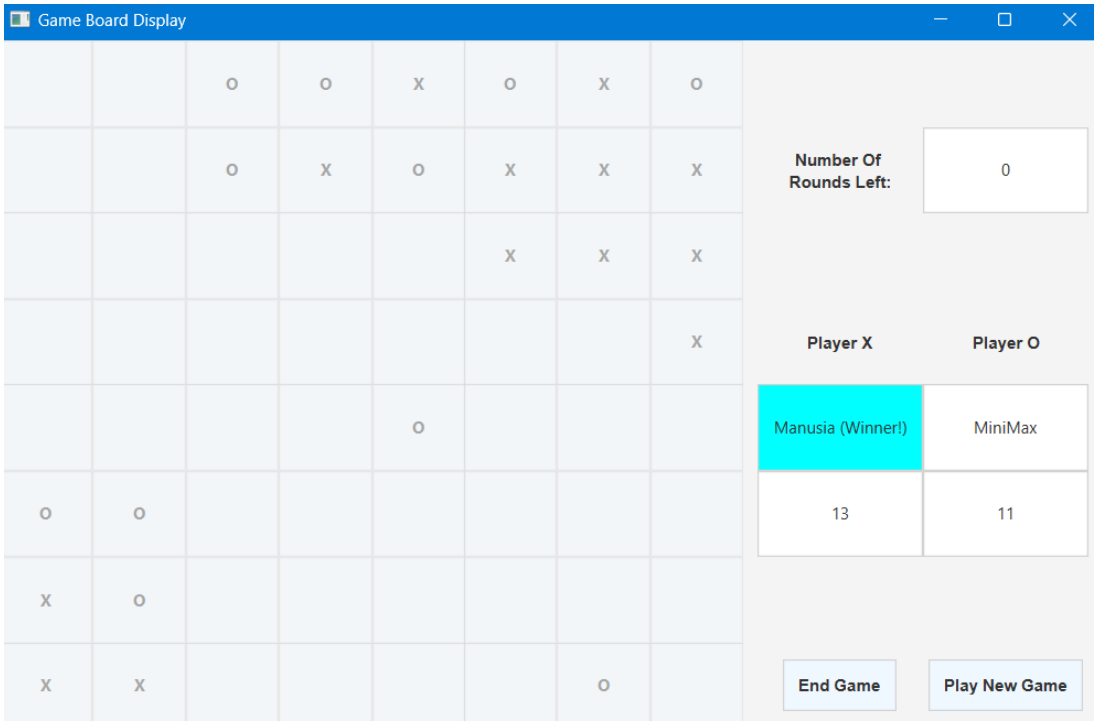
Algoritma local search yang kami pilih untuk digunakan pada pembuatan bot ini adalah hill climbing with sideway moves. Berikut adalah langkah dari algoritma hill climbing with sideway moves yang diimplementasikan.

1. Inisialisasi array result untuk menyimpan hasil pencarian.
2. Inisialisasi maxScore, variabel yang menyatakan nilai optimum terbaik yang diperoleh, dengan nilai 0.
3. Inisialisasi score, variabel yang menyatakan objective function yang diperoleh pada suatu langkah, dengan nilai 1.
4. Inisialisasi array moves yaitu array yang menyimpan seluruh move yang mungkin berdasarkan kondisi papan saat ini.
5. Lakukan pengulangan untuk setiap elemen dari array moves, yaitu untuk setiap konfigurasi yang mungkin, lakukan penghitungan objective function.
6. Apabila nilai objective function dari move yang diperiksa saat ini lebih dari maxScore, maka nilai maxScore di-*assign* dengan nilai objective function tersebut. Kemudian, hapus elemen dari array result saat ini dan tambahkan move saat ini ke array result.
7. Apabila nilai objective function dari move yang diperiksa saat ini sama dengan maxScore, maka move tersebut ditambahkan ke dalam array result.
8. Setelah pengulangan selesai, kembalikan array result.

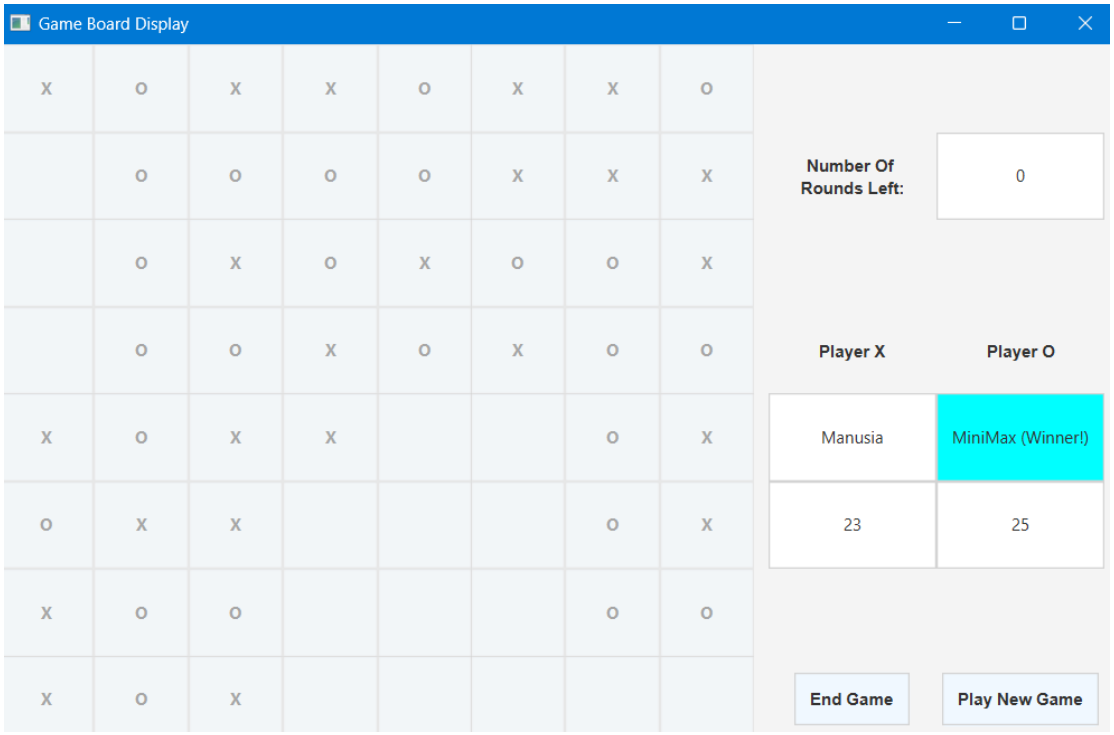
9. Move yang diambil adalah move pertama yang tersimpan pada array result. Hal ini dikarenakan move yang dilakukan telah dirandom sehingga jika ada berapapun kandidat dalam array result akan terambil satu random saja.

IV. Hasil Pertandingan

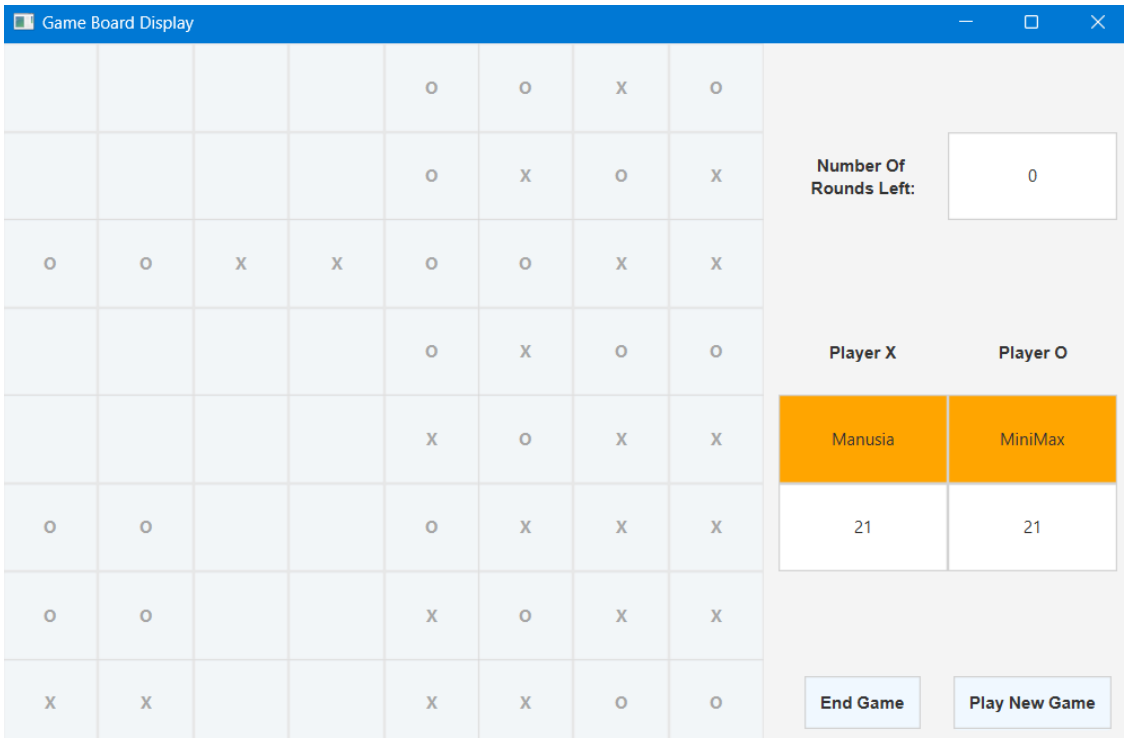
1. Manusia vs MiniMax



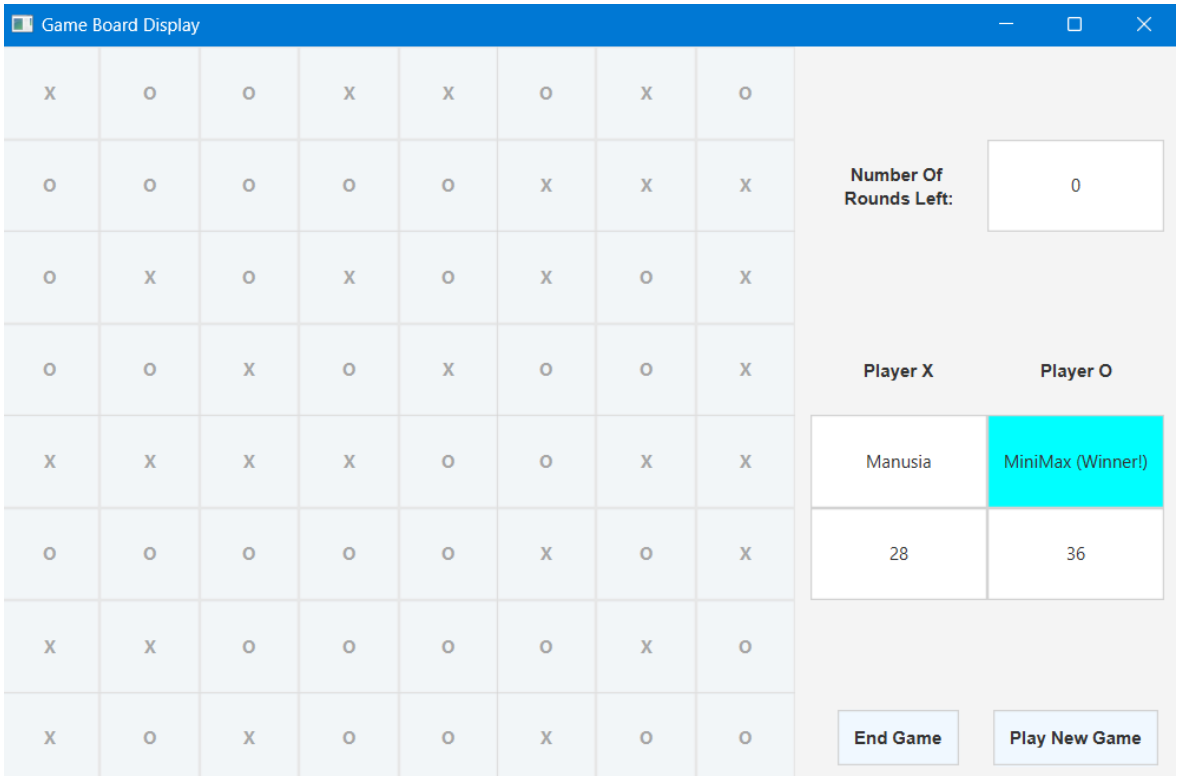
Pertandingan 1: 8 ronde



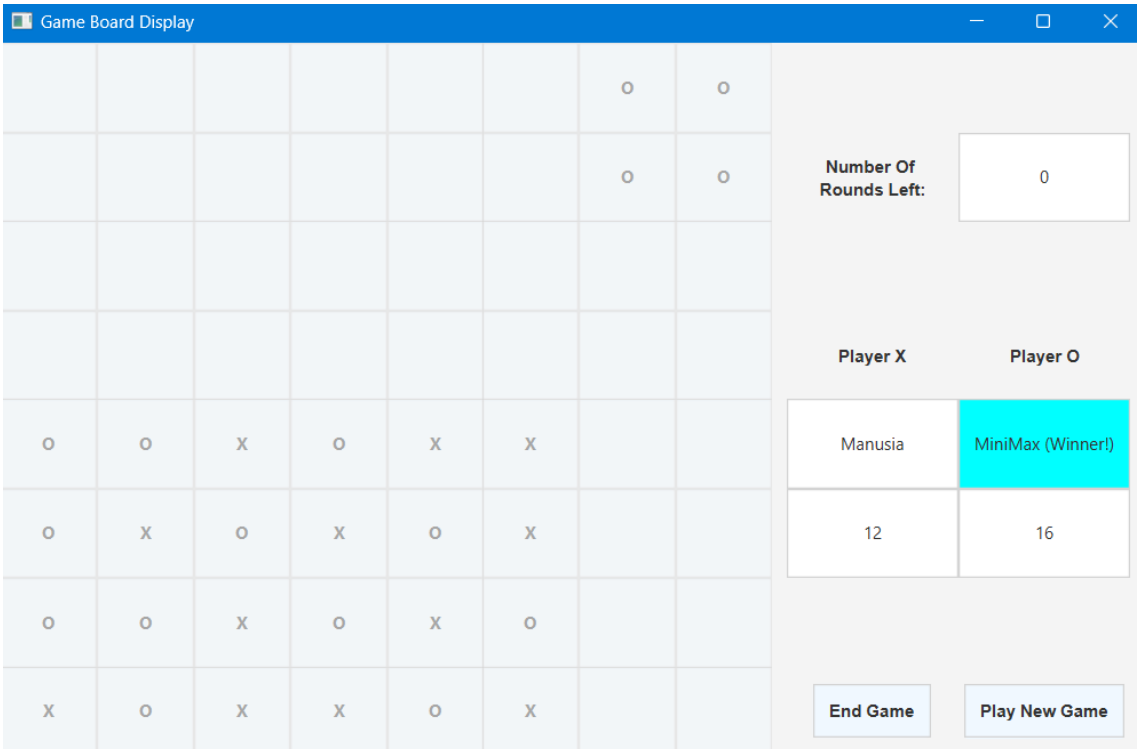
Pertandingan 2: 20 ronde



Pertandingan 3: 17 ronde



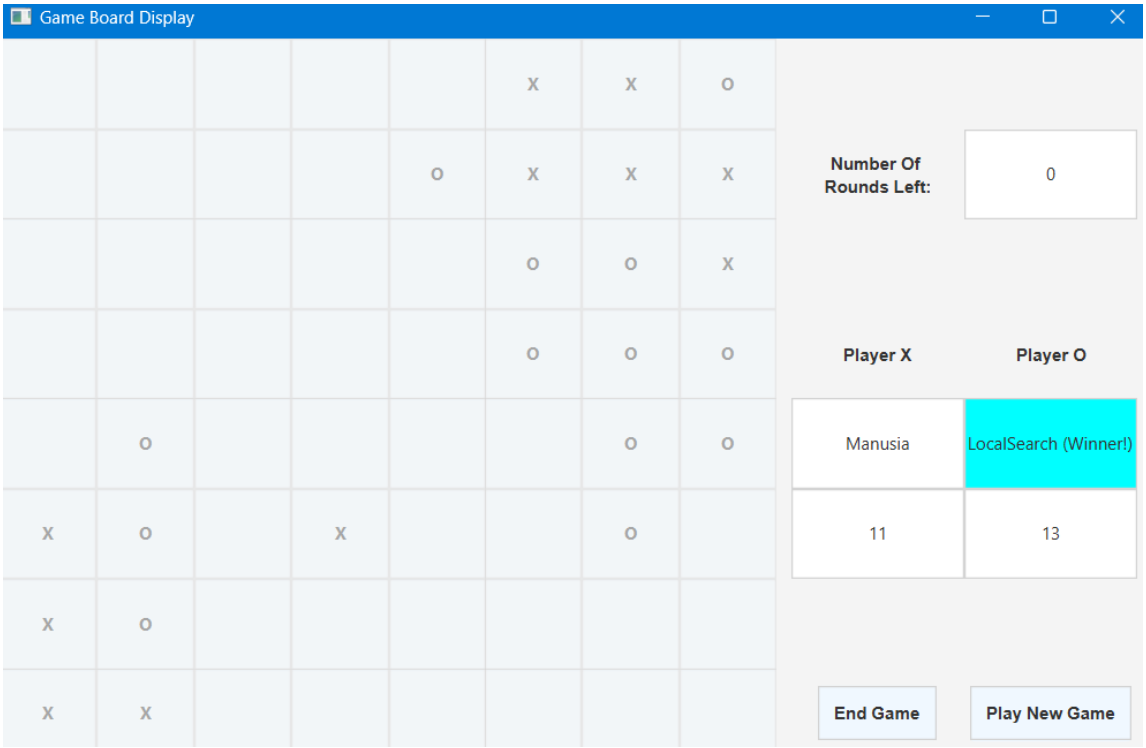
Pertandingan 4: 28 ronde



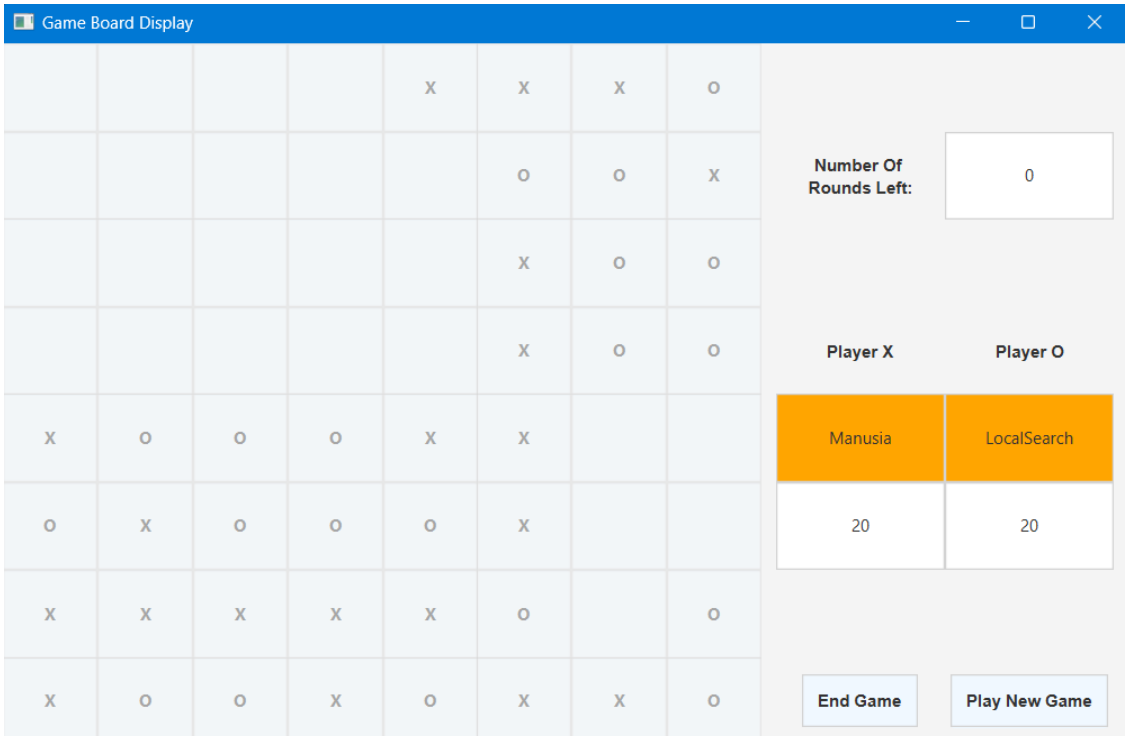
Pertandingan 5: 10 ronde

Dari hasil beberapa percobaan, bot MiniMax yang melawan manusia cenderung memiliki kemenangan yang sedikit lebih banyak. Hal ini dapat didukung jika pemain manusia melakukan kesalahan pada penempatan marka saat gilirannya. Namun jika manusia yang bermain benar-benar serius, bot MiniMax masih dapat seri atau bahkan dikalahkan. Hal ini juga dikarenakan *tree* yang kedalamannya dibatasi.

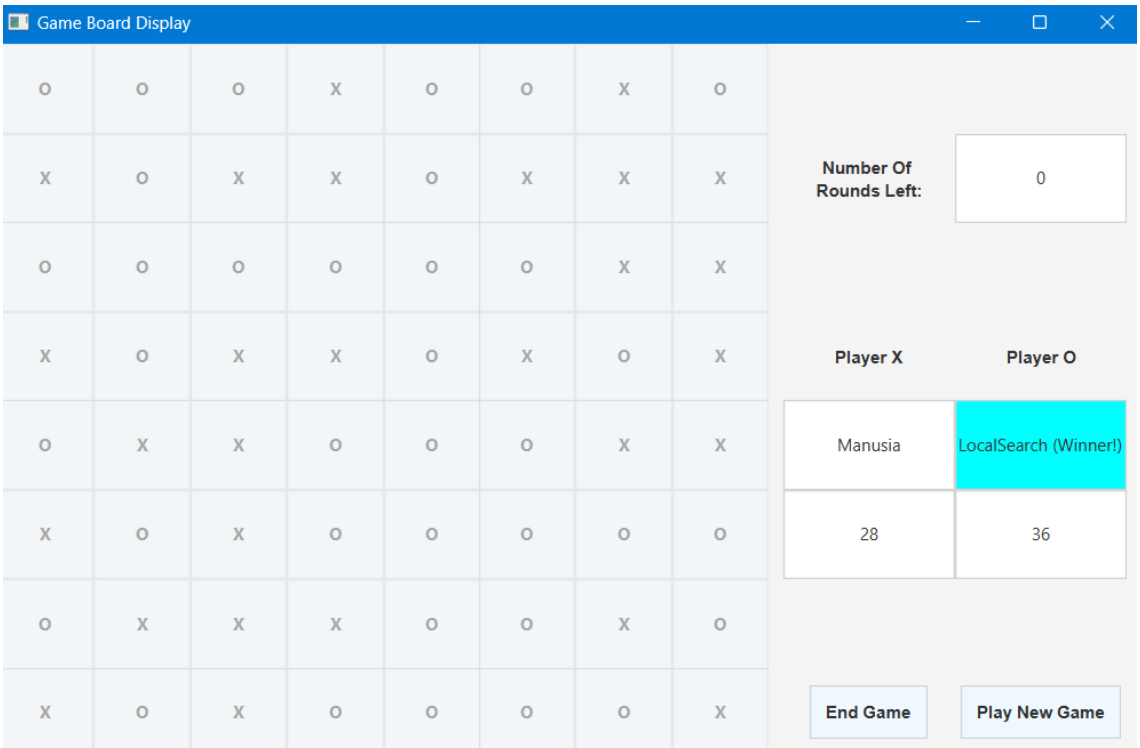
2. Local Search vs Manusia



Pertandingan 1: 8 ronde



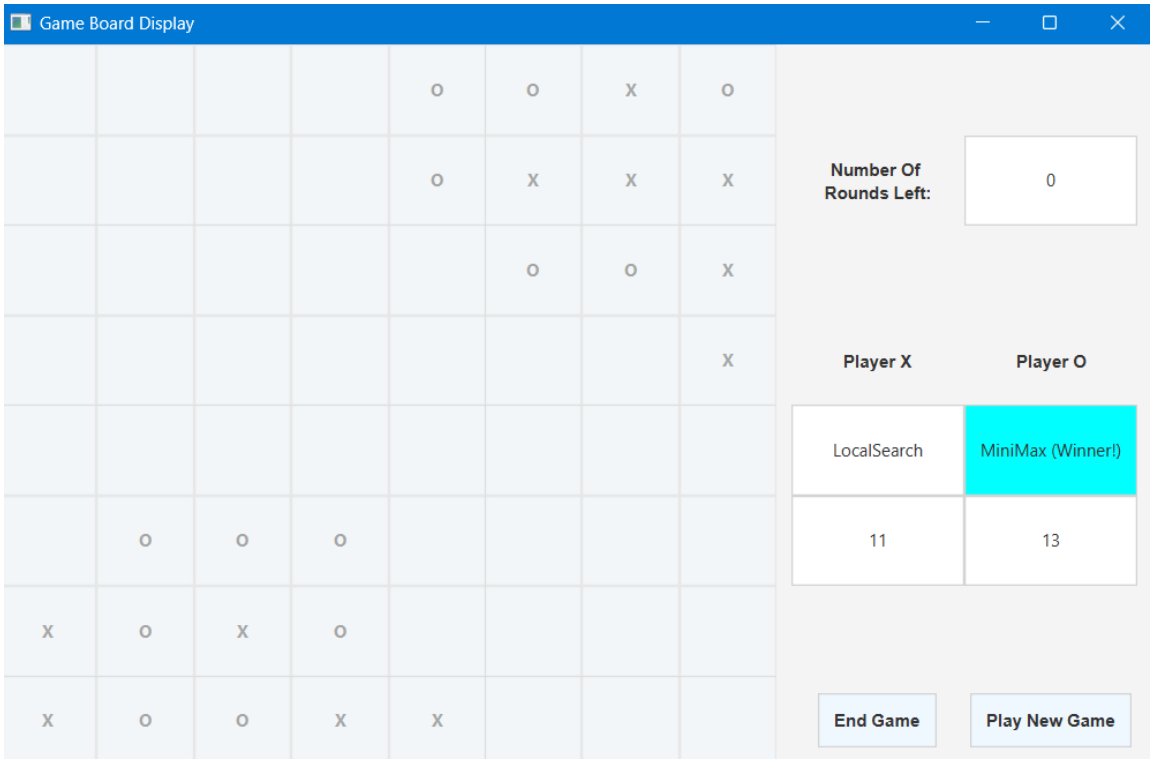
Pertandingan 2: 16 ronde



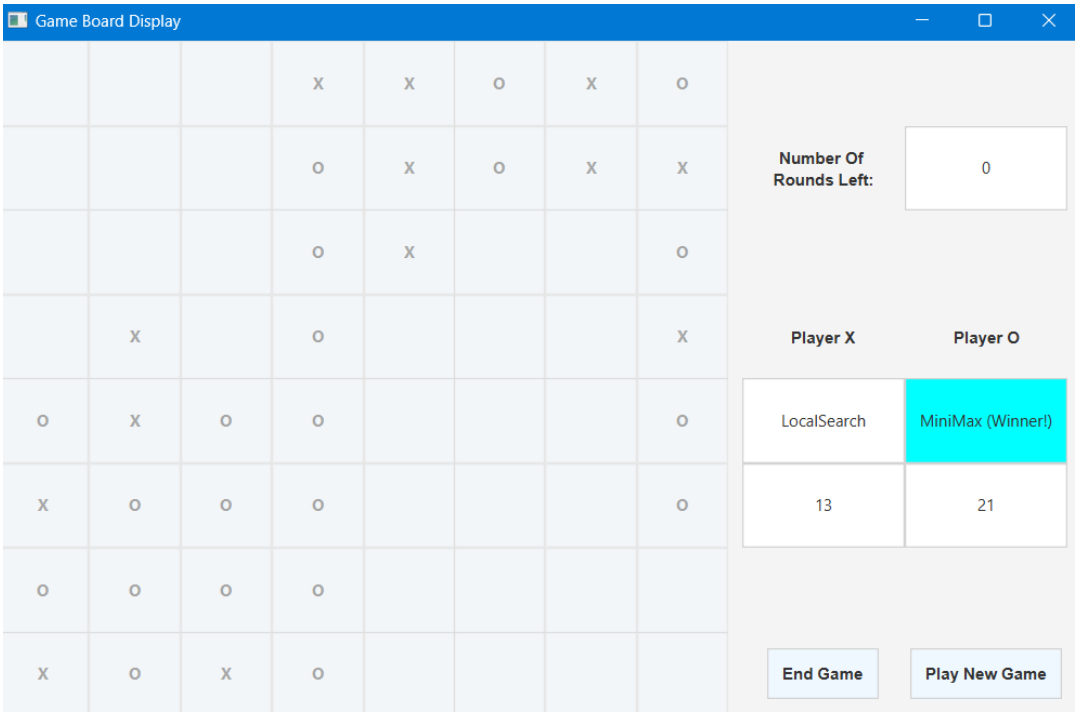
Pertandingan 3: 28 ronde

Bot LocalSearch pun juga demikian, memiliki kecenderungan untuk memenangi pertandingan ketika melawan manusia. Bot ini memeriksa nilai pada setiap kotak dengan mempertimbangkan keuntungan adanya marka lawan yang bisa ditukar menjadi marka milik mereka, sehingga nilai pada setiap kotak berkisar antara 1 (tidak ada marka lawan yang berdekatan) hingga 5 (ada 4 marka lawan yang berdekatan sehingga bisa ditukar menjadi marka milik bot.

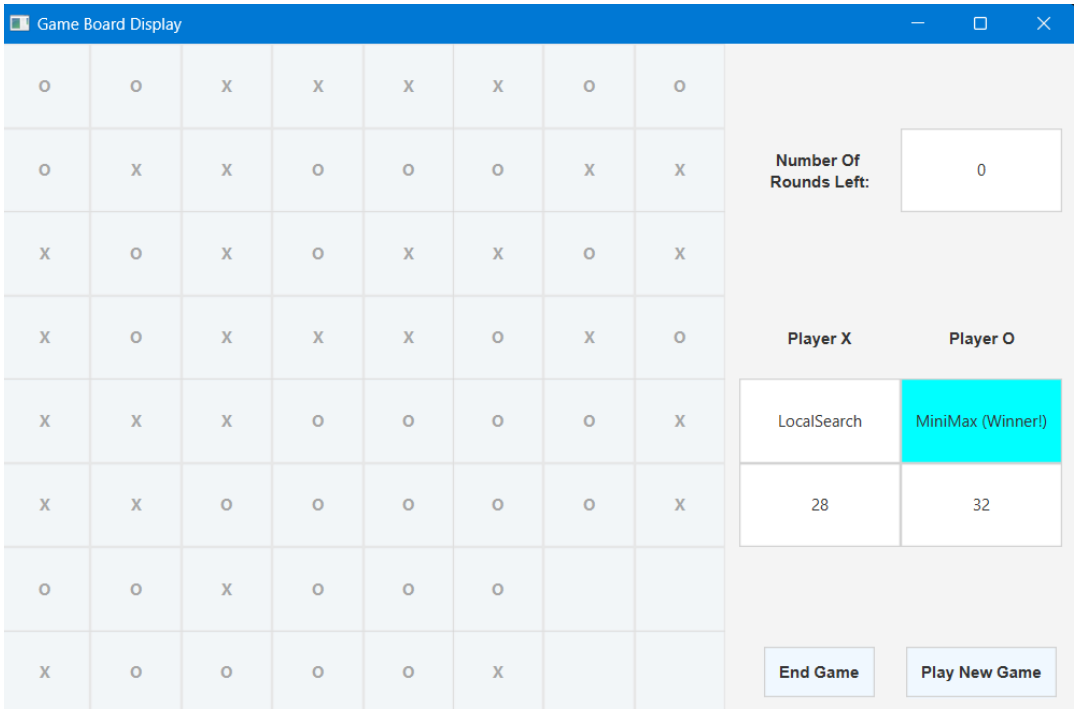
3. MiniMax vs Local Search



Pertandingan 1: 8 ronde



Pertandingan 2: 13 ronde



Pertandingan 3: 26 ronde

Pada pertandingan kedua bot, yakni, antara LocalSearch dan MiniMax, terlihat pada ketiga pertandingan bot minimax mengungguli pertandingan. Hal ini dikarenakan algoritma yang digunakan pada perhitungan bot MiniMax memiliki kompleksitas yang lebih detail dibanding pada LocalSearch.

V. Kontribusi Anggota

Nama	NIM	Kontribusi
Fazel Ginanda	13521098	Membuat objective function, algoritma minimax, dan local search, serta menyusun laporan
Akhmad Setiawan	13521164	Membuat objective function, algoritma minimax, dan local search, serta menyusun laporan
Irgiansyah Mondo	13521167	Membuat objective function, algoritma minimax, dan local search, serta menyusun laporan

Lampiran

Pranala Github: https://github.com/akhmadst1/Tubes1_13521098_AI