

LAPORAN TUGAS BESAR I
IF3170 INTELIGENSI BUATAN

MINIMAX ALGORITHM AND ALPHA BETA PRUNING IN
ADJACENCY STRATEGY GAME



Oleh:

Wilson Tansil	(13521054)
Bill Clinton	(13521064)
Eugene Yap Jin Quan	(13521074)
Jimly Firdaus	(13521102)

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

2023

DAFTAR ISI

DAFTAR ISI.....	1
PEMBAHASAN.....	2
1. Penggunaan Objective Function.....	2
2. Implementasi Bot dengan Algoritma Minimax Alpha Beta Pruning.....	2
3. Implementasi Bot dengan Local Search.....	4
4. Hasil Pertandingan.....	4
5. Link Video Pertandingan.....	9
6. Link Source Code.....	9
PEMBAGIAN TUGAS.....	10

PEMBAHASAN

1. Penggunaan Objective Function

Secara umum, objective function yang kami implementasikan berhubungan dengan banyak petak/tanda yang dimiliki bot atau lawan. *Objective function* ini digunakan untuk perbandingan nilai *state* saat ini dengan *neighbor state*.

Pada algoritma *minimax*, *objective function* digunakan dalam proses pencarian pada pohon *minimax*. Pada level *max*, pemilihan *node* dilakukan berdasarkan nilai objektif terbesar, sedangkan pada level *min*, pemilihan *node* dilakukan berdasarkan nilai objektif terkecil. Untuk implementasi algoritma *minimax*, *objective function* yang kami digunakan dihitung berdasarkan rumus ($\langle \text{jumlah petak bot} \rangle - \langle \text{jumlah petak lawan} \rangle + \langle \text{jumlah poin terjamin untuk bot} \rangle$). Jumlah poin terjamin pada rumus ini dihitung berdasarkan perbedaan jumlah petak bot yang tidak bisa diubah dengan jumlah petak lawan yang tidak bisa diubah.

Pada algoritma *local search* (dalam tugas ini, yang diimplementasikan adalah *simulated annealing*), *objective function* digunakan untuk menentukan apakah suatu *move* bisa dikatakan sebagai *good move* atau *bad move*. Suatu *move* dikategorikan sebagai *good move* jika menyebabkan penambahan tanda untuk bot dengan jumlah lebih dari 1 atau dengan kata lain berarti memakan tanda musuh minimal 1 butir, sedangkan suatu *move* dikatakan *bad move* jika hanya menyebabkan penambahan tanda untuk bot sebanyak 1 butir atau dengan kata lain hanya menambahkan tanda sendiri dan tidak memakan tanda lawan. Namun, perlu dicatat bahwa suatu *bad move* dalam *simulated annealing* dapat tetap dilaksanakan dengan syarat bahwa *move probability*-nya melebihi *threshold* yang telah ditetapkan sebelumnya.

2. Implementasi Bot dengan Algoritma Minimax Alpha Beta Pruning

Implementasi bot ini menerapkan prinsip-prinsip dasar algoritma *minimax alpha beta pruning*. Bot ini membentuk pohon *minimax* yang bergantian antara level *max* dan *min*. Bot ini kemudian mengevaluasi pohon secara rekursif, memperbarui nilai *alpha* dan *beta* dari level *leaf* menuju *root*. Pada level setiap level, pencarian akan dilakukan *pruning* jika ditemukan kondisi $\beta \leq \alpha$. Pada level *maximizing*, nilai *alpha* akan diperbarui

jika ditemukan nilai skor yang lebih besar, sedangkan pada level *minimizing*, nilai *beta* akan diperbarui jika ditemukan nilai skor yang lebih kecil.

Pada implementasi ini, sebuah pohon *minimax* disusun ketika bot pemain hendak memilih langkah. Langkah terpilih pada giliran tersebut adalah *child node* dari *root* yang memiliki nilai tertinggi.

Berdasarkan peraturan dan batasan permainan ini, kami menetapkan beberapa *heuristic* untuk membantu pemilihan langkah berikutnya. Di bawah ini adalah penjelasan singkat terkait *heuristic* tersebut.

1. Pada pencarian langkah terbaik, bot memberikan batasan minimal dan maksimal terhadap kedalaman pohon *minimax*. Batasan minimal bersifat konstan, sedangkan batas maksimal kedalaman pohon bersifat variabel. Batas maksimal ini dihitung menggunakan rumus $\text{depth} = \text{floor}(\text{minDepth} + (\text{maxDepth} - \text{minDepth}) * (1 - \min(1, \max(0, \text{emptyCoordinates} / (\text{maxCol} * \text{maxRow}))))$;
2. Pembangkitan *successor* dilakukan dengan melihat petak-petak yang kosong (belum ditempati oleh pemain/lawan). Jika terdapat lebih dari satu opsi langkah pada papan, *successor* yang akan ditambahkan ke dalam pohon *minimax* adalah *successor* yang memilih langkah pada sebuah petak kosong yang dikelilingi oleh minimal satu petak lawan.
3. Pada penelusuran pohon, *node-node* yang dievaluasi menggunakan urutan tertentu untuk mengefisienkan proses *pruning*. Urutan ini diperoleh dengan cara menetapkan prioritas *node*, yang dihitung berdasarkan jumlah petak lawan yang mengelilingi sebuah petak kosong terpilih. Pengurutan ini menggunakan *PriorityQueue* (library Java).

Mekanisme evaluasi menggunakan *priority queue* ini adalah dengan menyimpan tuple argumen yang akan digunakan dalam pemanggilan fungsi rekursif *minimax* pada setiap *node queue* (beserta prioritasnya). Setelah menambahkan semua *child node* dari sebuah *node* pohon pencarian, bot akan memanggil fungsi *minimax* dengan melakukan *pop* terhadap *queue*, kemudian melakukan pemanggilan menggunakan argumen tersimpan pada hasil *pop*. Oleh karena itu, proses pembentukan pohon berlangsung secara dua tahap, yaitu tahap seleksi dan pemasukan *node* ke dalam *queue*, kemudian tahap pemanggilan fungsi *minimax*.

3. Implementasi Bot dengan Local Search

Untuk *local search*, kami mengimplementasikan bot dengan algoritma *simulated annealing*. Alasan pemilihan ini karena *simulated annealing* memiliki kemampuan untuk menghindari *local optima* dan mampu mencapai *global optima* melalui lebih banyak kemungkinan move karena *bad move* boleh dilakukan dengan syarat bahwa *move probability*-nya melebihi *threshold* yang ditetapkan.

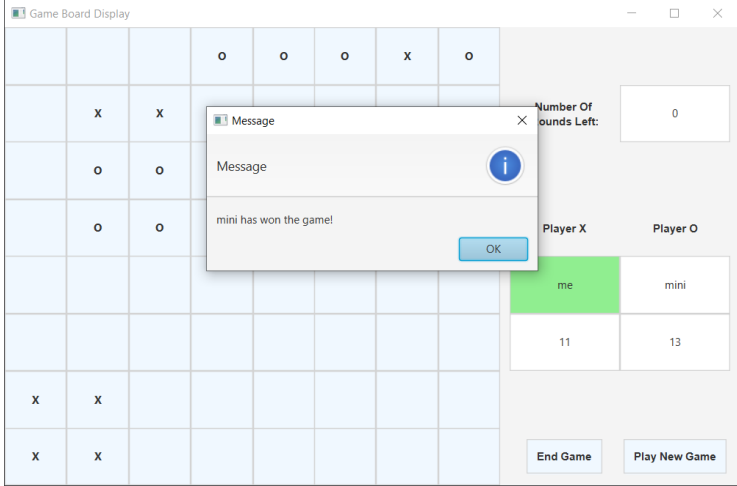
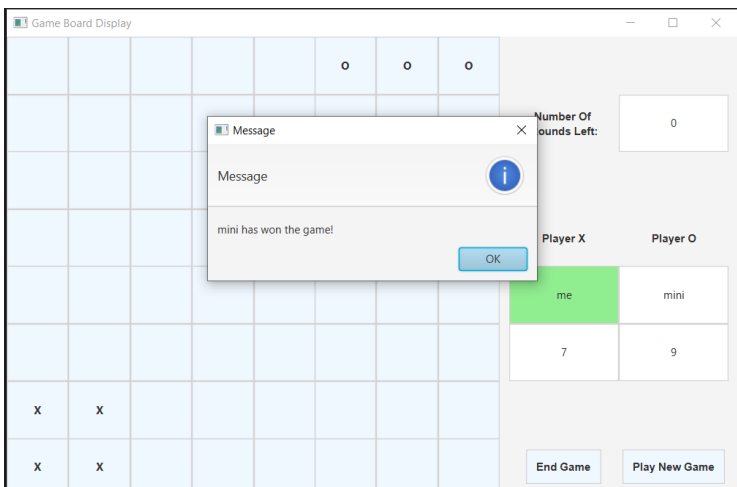
Pengimplementasiannya dimulai dengan pembangkitan seluruh *successor* yang mungkin. Setelah itu, dilakukan pemilihan *successor random* berupa koordinat dalam *board*. Koordinat ini kemudian akan dievaluasi nilai *state*-nya untuk kemudian dibandingkan dengan nilai *current state*, yaitu nilai sebelum dipilihnya koordinat tersebut. Jika nilai *state* ketika dipilih koordinat tersebut lebih tinggi dari nilai *current state* ditambah 1, koordinat tersebut jadi akan dipilih, sedangkan jika nilai *state* tersebut kurang dari sama dengan nilai *current state* ditambah 1, akan dihitung *move probability*-nya menggunakan rumus $P = e^{-\Delta E/T}$ dengan ΔE adalah *neighbor state value* - *current state value*. Jika nilai P melebihi *threshold* yang ditetapkan di awal, maka koordinat tersebut dipilih, sedangkan jika nilai P tidak melebihi *threshold* yang ditetapkan di awal, pemilihan *successor* secara random diulangi. Untuk laju pengurangan temperatur dan *threshold*, kami menetapkan berturut-turut sebesar 0.9 dan 0.5 berdasarkan hasil percobaan untuk menghasilkan solusi yang paling optimal. Langkah-langkah tersebut nantinya akan diulangi sebanyak jumlah round yang ditetapkan di awal permainan.

4. Hasil Pertandingan

a) Bot Minimax vs Manusia (5 kali)

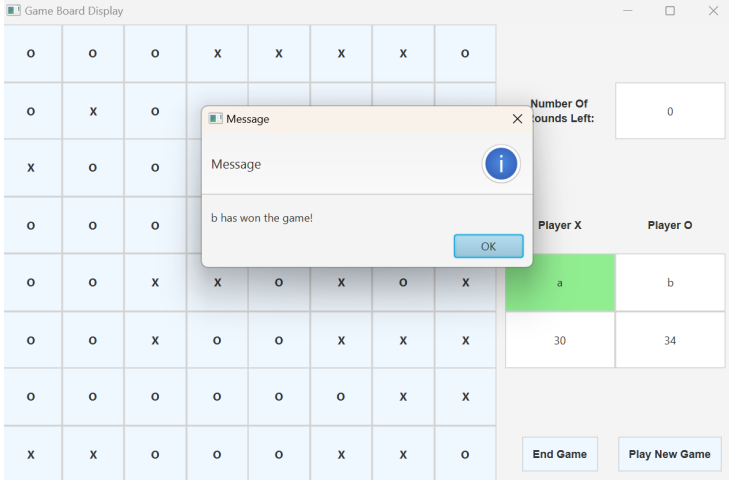
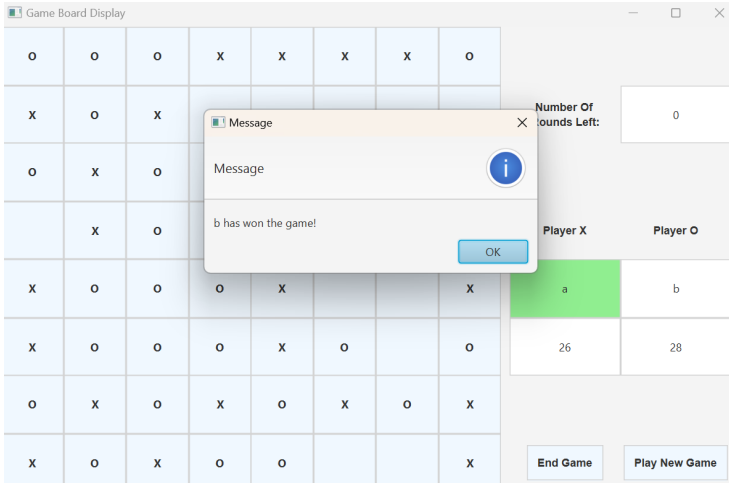
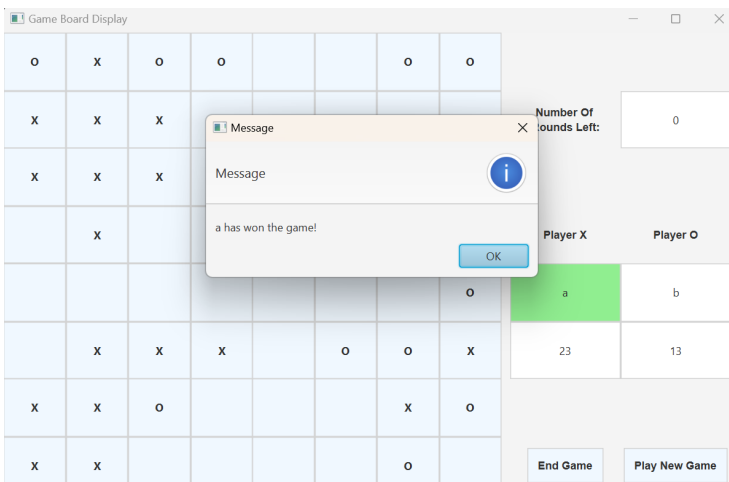
Total Ronde	Papan Akhir	Hasil (W/L)
----------------	-------------	----------------

28		W
23		W
14		L

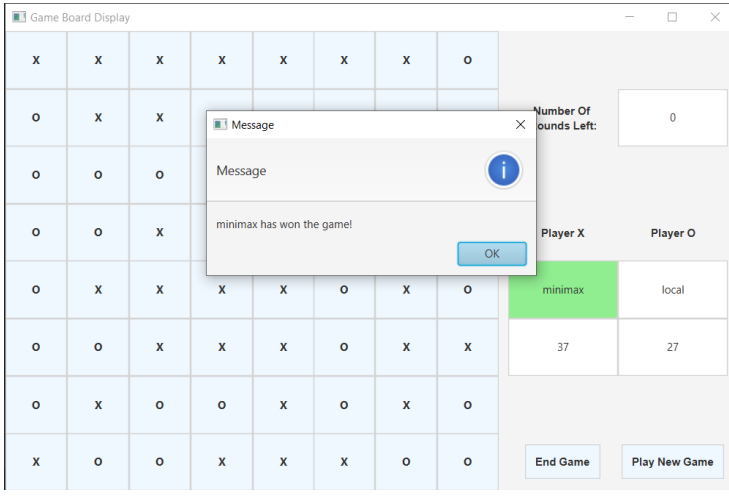
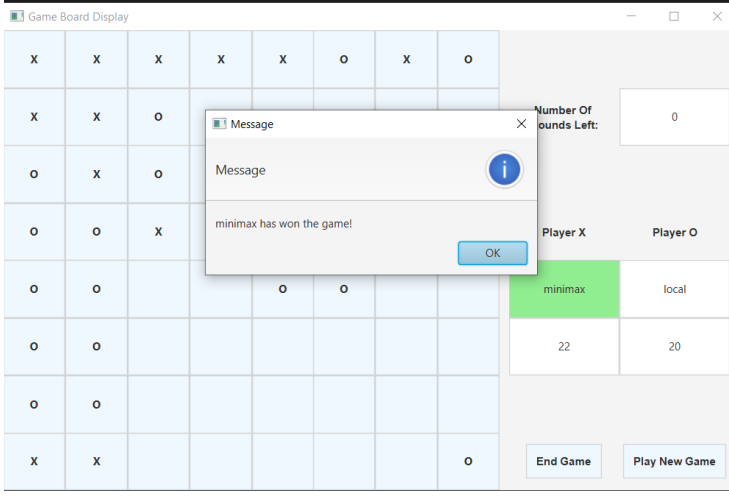
8		W
4		W
Persentase Kemenangan		80%

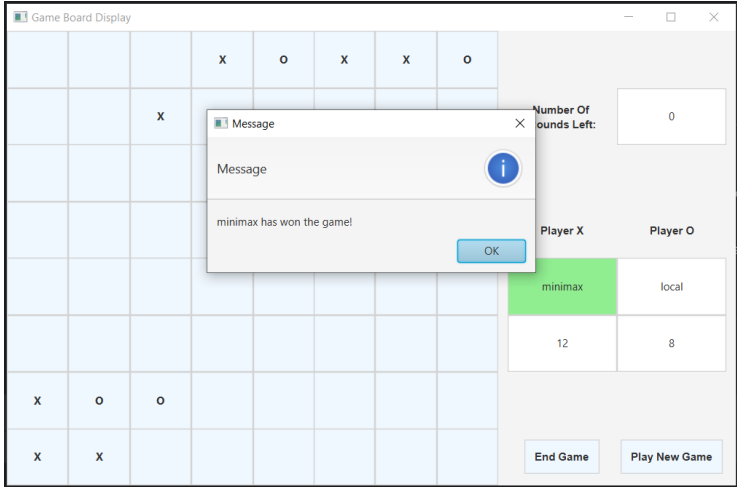
b) Bot Local Search vs Manusia (3 kali)

Total Ronde	Papan Akhir	Hasil (W/L)
----------------	-------------	----------------

28		W
23		W
14		L
Persentase Kemenangan		66.67%

c) Bot Minimax vs Bot Local Search (3 kali)

Total Ronde	Papan Akhir	Pemenang
28		minimax
17		minimax

6		minimax
Persentase Kemenangan Minimax		100%
Persentase Kemenangan Local Search		0%

5. Link Video Pertandingan

Hasil video pertandingan dapat diakses pada link:

<https://drive.google.com/drive/folders/1UUguMHKv9EB60o5dgjA7D-Frx64Bxs9i?usp=sharing>

6. Link Source Code

Source code program dapat diakses pada link:

<https://github.com/Jimly-Firdaus/Tubes-1-AI-Kelompok-6>

PEMBAGIAN TUGAS

Pembagian tugas untuk tugas besar I IF3170 Intelijensi Buatan ini adalah sebagai berikut.

Anggota	NIM	Tugas
Wilson Tansil	13521054	Bot Minimax
Bill Clinton	13521064	Bot Local Search
Eugene Yap Jin Quan	13521074	Bot Minimax
Jimly Firdaus	13521102	Bot Local Search