

**Laporan Tugas Besar 1 IF3170 Inteligensi Buatan**  
**Implementasi *Minimax Algorithm* dan *Local Search* pada Permainan**  
***Dots and Boxes***



**Disusun oleh:**

13520068 - Muhammad Naufal Satriandana

13520110 - Farrel Ahmad

13520133 - Jevant Jedidia Augustine

13520160 - Willy Wilsen

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**

**2022**

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Objective Function</b>	<b>3</b>
<b>Minimax dan Alpha-Beta Pruning</b>	<b>4</b>
<b>Local Search</b>	<b>6</b>
<b>Hasil Pertandingan</b>	<b>8</b>
Bot Minimax vs. Manusia	8
Bot Local Search vs Manusia	9
Bot Minimax vs Bot Local Search	12
Statistik Permainan	14
<b>Saran</b>	<b>15</b>
<b>Kontribusi</b>	<b>16</b>

# Objective Function

Pada permainan Dots and Boxes ini, fungsi objektif yang dibuat untuk digunakan dalam algoritma *Local Search* dan *Minimax* adalah sebagai berikut.

- Poin permainan berawal dengan nilai 0
- Jika pemain mendapatkan kotak, maka poin permainan bertambah sebanyak 1
- Jika lawan mendapatkan kotak, maka poin permainan berkurang sebanyak 1
- Apabila poin permainan berakhir dengan nilai positif, maka pemain menang
- Apabila poin permainan berakhir dengan nilai negatif, maka lawan menang

Fungsi objektif tersebut dibuat karena pemenang dari permainan *Dots and Boxes* ditentukan dari poin akhir kedua pihak. Pihak dengan poin terbanyak akan memenangkan permainan. Fungsi objektif ini menjadi representasi nilai *state* bagi pemain. Semakin besar dan positif maka semakin besar *advantage* pemain terhadap lawan sehingga kemenangan bagi pemain semakin pasti.

## *Minimax dan Alpha-Beta Pruning*

Proses pencarian dengan menggunakan *Minimax Alpha Beta Pruning* akan dimulai dengan penetapan *state* awal yang akan dijadikan sebagai akar dari pohon pencarian. *State* awal ini bisa berupa *board* kosong atau *board* yang sedang dimainkan. Kemudian akan ditetapkan berapa giliran ke depan yang akan diprediksi. Jumlah giliran yang banyak akan memperlambat komputasi dan sebaliknya.

*State* awal akan dibangun anak-anaknya. Jumlah anak dari sebuah *state* didasarkan jumlah garis yang dapat dipilih pada *state* tersebut. Anak-anak yang telah dibangun akan dibangun juga anak-anaknya. Hal tersebut akan terus dilakukan sampai kedalaman pohon sama dengan jumlah giliran ke depan yang akan diprediksi. Apabila pembangunan anak sudah selesai dilakukan, maka setiap daun atau terminal dari pohon pencarian akan dicari *state value*-nya dengan menggunakan *objective function*. Pada tiap kedalaman juga akan ditandai dengan *min* atau *max*. Penandaan tiap kedalaman dengan nilai *min* atau *max* dilakukan berselang-seling dari akar menuju terminal, dimulai dengan akar dari pohon ditandai dengan *max*, anak dari akar ditandai *min*, anak dari anak dari akar ditandai *max*, dan seterusnya. Akan tetapi, apabila pada suatu *node* non-terminal salah satu pihak (*min* atau *max*) terbentuk sebuah kotak alhasil dari pemilihan garis pada *board*, maka semua anak dari *node* tersebut akan memiliki tanda yang sama dengan orang tuanya (artinya penandaan tidak dilakukan berseling pada kasus ini). Apabila penandaan sudah selesai dilakukan, maka dimulai dari terminal, berdasarkan tanda dari kedalaman tertentu akan dipilih anak dengan *state value* terkecil atau terbesar. *Min* berarti orang tua akan menginherit nilai terkecil yang dimiliki anaknya dan *max* berarti orang tua akan menginherit nilai terbesar yang dimiliki anaknya. Proses ini akan dilakukan dari orang tua daun/terminal hingga akar pohon pencarian, dimana pada akar akan didapatkan jalur menuju sebuah terminal yang sudah dioptimasi dengan algoritma Minimax.

Untuk meningkatkan efisiensi dari algoritma Minimax, akan digunakan *alpha beta pruning*. Dengan menggunakan *alpha beta pruning*, maka tidak semua terminal harus dibangun dan dicari nilainya. Apabila sebuah *node* sudah diprediksi tidak akan memberikan nilai yang

akan dipilih oleh orang tuanya, maka *node* tersebut akan di-*prune* untuk menghemat biaya komputasi.

## *Local Search*

Untuk proses pencarian dengan algoritma local search akan digunakan varian hill-climbing with sideways move. Penggunaan algoritma local search dengan varian ini didasarkan bahwa akan ada banyak node tetangga yang memiliki nilai yang sama dengan node orangtuanya, sehingga apabila menggunakan varian hill-climbing biasa, akan sering didapatkan kasus dimana tidak ditemukan langkah optimal sama sekali. Dengan hill-climbing with sideways move, kemungkinan ditemukan langkah optimal menjadi lebih besar yang menyebabkan algoritma dapat berjalan secara efektif. Akibat dari natur local search, maka hanya bisa dicari 1 langkah optimal kedepan, dibandingkan dengan minimax alpha beta pruning dimana algoritma dapat memprediksi berapapun jumlah langkah kedepan.

Algoritma local search dimulai dengan menentukan state awal yang akan diolah untuk ditemukan langkah optimal selanjutnya. Berdasarkan jumlah garis yang bisa dipilih pada board, maka akan dibangun anak (successor) dengan jumlah yang sama dengan jumlah garis yang bisa dipilih. Dalam implementasi ini, dibagi menjadi 2, yakni successor yang dibangkitkan dari row dan dari column. State dari setiap successor merupakan state orang tua yang ditambahkan dengan salah satu garis yang dapat dipilih dari state orang tua. Kemudian pada setiap successor, akan diterapkan objective function untuk menemukan state value-nya, hal yang sama dilakukan pada state orang tua. Untuk mengetahui pengambilan langkah selanjutnya yang paling optimal, akan dibandingkan state value dari orang tua dan anak-anaknya. Pemilihan anak (neighbor) akan didasarkan dengan algoritma berikut (prioritas berurut):

- Anak yang dipilih adalah anak yang memiliki state value terbesar (atau terkecil, tergantung giliran) dan lebih besar dari state value orang tua.
- Apabila tidak terdapat anak yang memiliki state value yang lebih besar dari orang tua, maka akan diambil anak yang memiliki state value yang sama dengan state value orang tua.
- Apabila terdapat beberapa anak dengan state value yang sama dan nilai tersebut adalah nilai terbesar dari semua anak dan nilainya adalah lebih besar atau sama dengan state value orang tua, maka akan dilakukan pengambilan acak dari anak-anak yang memiliki state value yang sama tersebut.

- Apabila tidak ada anak dengan state value yang lebih besar atau sama dengan state value orang tua, maka tidak ada langkah optimal yang dapat diambil.

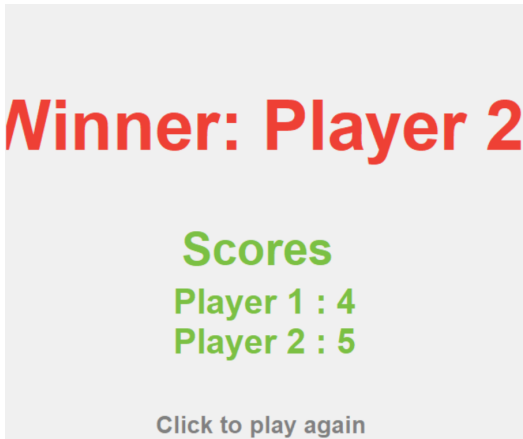
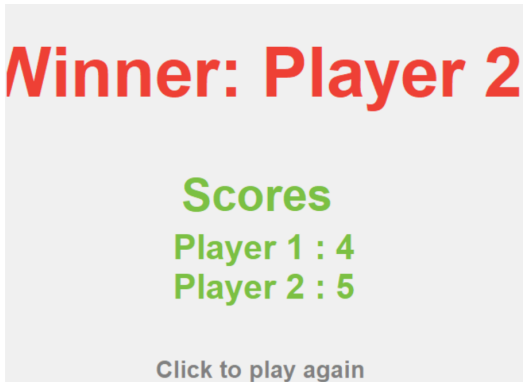
Berdasarkan algoritma tersebut, maka akan ada 2 kemungkinan hasil. Akan ditemukan sebuah langkah optimal yang dapat diambil dari state orang tua, atau tidak ada langkah optimal yang dapat diambil dari state orang tua. Akan tetapi, dalam implementasinya, walaupun tidak ada state anak yang bisa dipilih, tetap akan diambil sebuah neighbor, karena tidak mungkin permainan berhenti di tengah. Untuk mengimplementasikan hal tersebut, pengambilan neighbor cukup dari melihat state value yang maksimum (atau minimum, tergantung giliran).

# Hasil Pertandingan

Bot *Minimax* vs. Manusia

Player 1 : Bot Minimax

Player 2 : Manusia

Pertandingan ke-	Screenshot	Keterangan
1		Manusia menang dengan skor 4:5
2		Manusia menang dengan skor 4:5



3		Manusia menang dengan skor 4:5
4		Bot Minimax menang dengan skor 6:3
5		Bot Minimax menang dengan skor 5:4

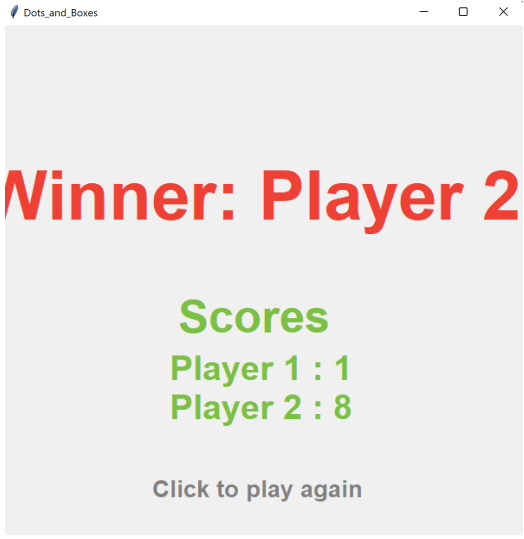
## Bot *Local Search* vs Manusia

Player 1: Bot Local Search

Player 2: Manusia

Pertandingan ke-	Screenshot	Keterangan
1		Manusia menang dengan skor 3:6
2		Manusia menang dengan skor 4:5


3		Bot Local Search menang dengan skor 5:4
4		Manusia menang dengan skor 4:5

5		Manusia menang dengan skor 1:8
---	---	--------------------------------

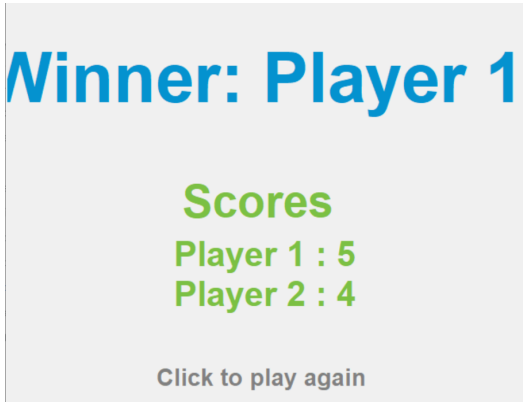
## Bot Minimax vs Bot Local Search

Player 1: Bot Minimax

Player 2: Bot Local Search

Pertandingan ke-	Screenshot	Keterangan
1		Bot Minimax menang dengan skor 5:4

2	<div><div>Winner: Player 2</div><div>Scores Player 1 : 3 Player 2 : 6</div><div>Click to play again</div></div>	Bot Local Search menang dengan skor 3:6
3	<div><div>Winner: Player 2</div><div>Scores Player 1 : 1 Player 2 : 8</div><div>Click to play again</div></div>	Bot Local Search menang dengan skor 1:8
4	<div><div>Winner: Player 2</div><div>Scores Player 1 : 4 Player 2 : 5</div><div>Click to play again</div></div>	Bot Local Search menang dengan skor 4:5

5		Bot Minimax menang dengan skor 5:4
---	---	------------------------------------

## Statistik Permainan

Bot Minimax vs Manusia & Bot Local Search vs Manusia

Jenis Bot	Jumlah Menang	Total Pertandingan	Win Rate
Minimax	2	5	40%
Local Search	1	5	20%

Bot Minimax vs Bot Local Search

Jenis Bot	Jumlah Menang	Total Pertandingan	Win Rate
Minimax	3	5	60%
Local Search	2	5	40%

## Saran

Terdapat beberapa saran perbaikan terhadap kode program yang dibuat. Saran yang utama adalah untuk mencari *objective function* yang lebih efektif dalam mencari *state value* dari *state* yang ada. *Objective function* yang digunakan pada tugas besar kali ini adalah jumlah dari kotak player 1 dikurangi dengan jumlah kotak player 2. Dengan menggunakan perhitungan yang lebih detail dan mendalam, maka dapat ditemukan *objective function* yang lebih baik daripada *objective function* yang kami buat.

Saran khusus untuk algoritma minimax dengan alpha beta pruning adalah menentukan kedalaman yang lebih efektif dalam melakukan prediksi langkah yang akan diambil selanjutnya. Kedalaman yang digunakan pada tugas besar kali ini adalah 4 karena kecepatan algoritma dapat dikatakan seimbang dengan hasil yang diprediksi pada kedalaman ini. Dengan melakukan analisis pada kecepatan dan hasil prediksi pada algoritma ini, maka dapat ditentukan kedalaman yang lebih baik daripada kedalaman yang kami tentukan.

Saran khusus untuk algoritma local search adalah dengan membuat heuristik yang lebih cerdas. Heuristik yang diimplementasikan pada bot ini hanya sebatas memprediksi satu langkah ke depan yaitu jangan memilih garis yang membuat kotak tersebut dapat diambil oleh lawan pada langkah berikutnya dan hanya terbatas pada satu kotak. Heuristik yang lebih cerdas seharusnya dapat memprediksi langkah musuh lebih dari satu langkah ke depan.

# Kontribusi

NIM	Nama	Kontribusi
13520068	Muhammad Naufal Satriandana	Membuat laporan dan membuat bot Local Search
13520110	Farrel Ahmad	Membuat laporan dan membuat bot Local Search
13520133	Jevant Jedidia Augustine	Membuat laporan dan membuat bot Minimax
13520160	Willy Wilsen	Membuat laporan dan membuat bot Minimax