Penerapan Algoritma *Minimax* dengan *Alpha-Beta Pruning* sebagai Strategi Permainan Othello

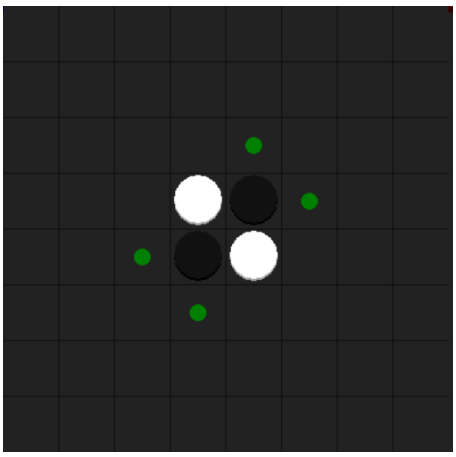
Adiyansa Prasetya Wicaksana - 13520044
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520044@std.stei.itb.ac.id

***Abstract*—Permainan Othello merupakan permainan papan berbasis giliran yang dimainkan oleh dua pemain. Untuk memainkan permainan Othello diperlukan strategi tertentu jika ingin mendapatkan hasil yang optimal. Strategi tersebut dapat diimplementasikan dengan algoritma. Algoritma yang dipilih adalah algoritma *minimax* dengan meminimalisasi langkah berdasarkan kemunkgin terburuk. Untuk optimasi lebih lanjut dari algoritma *minimax* digunakan *alpha-beta pruning* untuk memangkas langkah penelusuran yang dianggap tidak menuju solusi.**

Keywords—Othello; minimax; alpha-beta

I. PENDAHULUAN

Othello merupakan salah satu jenis permainan papan. Permainan papan menjadi permainan yang cukup digemari oleh banyak orang. Bahkan, saat ini banyak sekali permainan papan baru yang muncul dengan berbagai macam aturan dan konsep yang menarik. Othello dikembangkan dari permainan Reversi yang merupakan permainan papan yang diciptakan pada 1883 oleh antara Lewis Waterman atau John W. Mollet (keduanya saling menuduh) dan akhirnya pada 1971 permainan Othello dipatenkan oleh Goro Hasegawa. Nama Othello didasarkan kepada drama yang dibuat oleh Shakespeare [1].



Gambar 1.1 Posisi Awal Papan

Permainan Othello dimainkan pada papan ukuran 8 x 8 dengan 64 keping. Setiap keping memiliki kedua sisi yang berbeda warna yaitu hitam dan putih. Permainan dimulai dengan keping yang berada di kiri atas papan, lihat Gambar 1.1. yang berada di kiri atas pappan adalah keping putih sehingga permainan dimulai oleh pemain dengan keping putih. Permainan dilanjutkan dengan saling bergantian antara keping putih dan hitam.

Pemain dapat menjalankan giliran dengan menggerakan keping yang dimiliki ke arah horizontal, vertikal, ataupun diagonal. Pemain dapat membalikkan kepingan lawan menjadi kepingan yang dimiliki oleh pemain tersebut jika kepingan lawan diapit oleh kepingan yang pemain miliki. Pemain harus berlomba-lomba mendapatkan kepingan dengan warna yang pemain tersebut miliki untuk memenuhi papan. Permainan selesai jika papan sudah terisi penuh oleh kepingan atau salah satu pemain sudah tidak bisa melakukan gerakan. Permainan dimenangkan oleh pemain yang memiliki kepingan terbanyak.

Seperti permainan papan lainnya, Othello dengan aturannya sendiri memerlukan strategi dalam memainkannya. Beberapa algoritma yang dapat diterapkan dalam permainan ini adalah *greedy*, *branch and bound*, *brute force*, *A**, dan *backtracking*. Pada makalah ini, akan diterapkan algoritma *minimax* dengan *alpha-beta pruning* yang merupakan implementasi dari algoritma *backtracking* untuk menentukan langkah terbaik pada giliran tersebut.

II. DASAR TEORI

A. *Depth-First Search*

Depth-First Search (DFS) merupakan algoritma pencarian mendalam dalam penelusuran graf. DFS dalam penelusurannya melakukan penelusuran terhadap anak suatu simpul terlebih dahulu sebelum menelusuri simpul yang bertetangga. DFS merupakan versi berlawanan dari *Breadth-First Search* (BFS) yang mana penelusuran dilakukan terhadap semua simpul tetangga terlebih dahulu sebelum menelusuri anak dari simpulnya.

DFS memiliki implementasi lanjutannya, salah satunya adalah *backtracking* dan *iterative deepening search*. Pada *Iterative deepening search* adalah implementasi DFS dengan menambahkan parameter tambahan yaitu kedalaman. Pencarian dengan *iterative deepening search* akan dibatasi pada suatu kedalaman tertentu sesuai parameter. Hal ini dilakukan untuk penghematan waktu dan juga simpul yang

ditelusuri karena penelusuran DFS akan melakukan pencarian setiap simpul sampai ditemukan solusi.

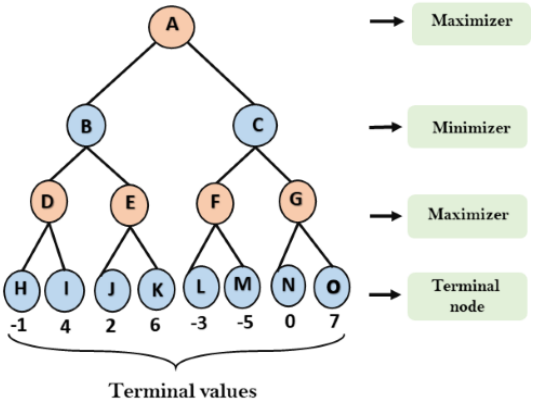
Pada algoritma *backtracking*, pencarian suatu simpul tidak diteruskan jika memenuhi suatu kondisi tertentu. Pada implementasinya, kasus ini dipangkas (tidak dilanjutkan) karena dianggap tidak akan mencapi suatu solusi sehingga akan mempercepat mencapai solusi karena tidak harus menelusuri semua simpulnya terlebih dahulu.

B. *Minimax*

Algoritma *Minimax* merupakan algoritma pengambilan keputusan untuk mengambil keputusan yang meminimalisasi kerugian dari keadaan terburuk (kerugian maksimal). *Minimax* biasanya digunakan untuk permainan yang memerlukan giliran antara dua pemain sehingga bisa dilakukan analisis terhadap gerakan pemain lawan [2].

Pada penggunaannya dalam permainan berbabis giliran/*turn based* salah satu pemain menggunakan MAX dan pemain lawannya menggunakan MIN. Pemain MAX akan memilih nilai yang paling maksimum, sedangkan pemain MIN akan memilih nilai yang paling minimum. Asumsi dari algoritma ini adalah MAX dan MIN akan memilih langkah yang optimal. Pengambilan keputusan akan berdasarkan terhadap MIN dan MAX tiap simpulnya.

Algoritma ini dapat divisualisasikan dengan pohon ruang status dengan tiap simpul memvisualisasikan kemungkinan langkah yang mungkin dipilih. Tiap simpul akan dilakukan evaluasi menjadi sebuah *value* berdasarkan *utility function* pada program untuk menilai sebuah simpulnya yang umumnya semakin besar *value*-nya maka akan semakin menguntungkan pemain tersebut. Simpul-simpul pada pohon ruang mempunyai kedalamannya (*depth*) masing-masing yang diawali dari *depth* 0.

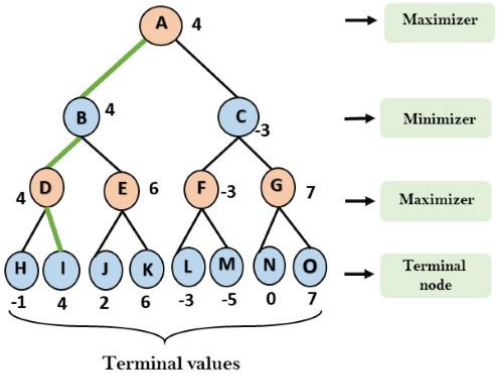


Gambar 2.1 Visualisasi Pohon Status

(Sumber: https://www.javatpoint.com/mini-max-algorithm-in-ai)

Pada permainan berbasis giliran/*turn-based* dengan dua pemain pada setiap kedalaman, pemain akan mencari nilai maksimum/minimum tergantung giliran dan akan dievaluasi dari seluruh anak simpul. Sebagai contoh, simpul A akan mencari nilai maksimum berdasarkan *value* dari simpul B dan C, sedangkan simpul B akan mencari nilai minimum berdasarkan *value* dari D dan E dan juga simpul C akan mencari nilai minimum dari simpul F dan G. Pola penulusan ini akan dilakukan sampai mencapai simpul daun yang mempunyai sebuah nilai. Pada akhirnya algoritma ini akan memberikan nilai dari evaluasi simpul pada akar yang merupakan *value* terbaik.

Dari Gambar 2.1., pada kedalaman 2 simpul D akan menghasilkan nilai max(-1, 4) → 4, simpul E akan menghasilkan nilai max(2, 6) → 6, simpul F akan menghasilkan nilai max(-3, -5) → -3, dan simpul G akan menghasilkan nilai max (0, 7) → 7. Proses tersebut akan dilakukan juga pada kedalaman yang lebih tinggi hingga dicapai akarnya dan pada akhirnya akan mendapat kan hasil 4.



Gambar 2.2 Hasil Akhir dari Visualisasi

(Sumber: https://www.javatpoint.com/mini-max-algorithm-in-ai)

```
function minimax(node, depth, maximizingPlayer) is
if depth ==0 or node is a terminal node then
    return static evaluation of node

if MaximizingPlayer then      // for Maximizer Player
    maxEva= -infinity
     for each child of node do
     eva= minimax(child, depth-1, false)
    maxEva= max(maxEva,eva)          //gives Maximum of the values
    return maxEva
else                          // for Minimizer player
     minEva= +infinity
     for each child of node do
     eva= minimax(child, depth-1, true)
```

```
    minEva= min(minEva, eva)          //gives minimum of the values
                return minEva
```

Berikut merupakan *pseudocode* dari algoritma *minimax* yang diambil dari https://www.javatpoint.com/mini-max-algorithm-in-ai .

C. *Alpha-Beta Pruning*

Algoritma *alpha-beta pruning* merupakan optimasi untuk *minimax*. Penulusuran simpul pada pohon ruang status algoritma *minimax* akan dilakukan banyak penelusuran simpul. Jumlah simpul akan bertambah secara eksponensial setiap penambahan kedalaman. Sehingga, agar algoritma dapat berjalan dengan lebih efisien diperlukan adanya optimasi.

Algoritma ini digunakan untuk memangkas (*pruning*) simpul-simpul yang dianggap tidak akan memberikan solusi optimal. Pada algoritma ini akan digunakan dua variable *alpha* dan *beta* (oleh karena itu diberi nama *alpha-beta pruning*) [3]. *Alpha*, mewakili nilai tertinggi sejauh ini yang diperoleh pemain MAX, sedangkan *beta* mewakili nilai terendah sejauh ini yang diperoleh pemain yang meminimalkan nilai.

Pada pemanggilan algoritma *minimax*, *alpha* dan *beta* akan diberi nilai masing-masing negatif tak hingga dan tak hingga. Setelah pemanggilan beriktunya, kedua nilai ini akan diperbarui ketika mendapatkan nilai evaluasi simpul yang lebih optimal. Kondisi agar pencarian tetap dilakukan adalah ketika α>=β. Ketika kondisi tersebut tidak dipenuhi maka pencarian di simpul tersebut akan dipangkas (*pruning*).

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is

if depth ==0 or node is a terminal node then
return static evaluation of node

if MaximizingPlayer then      // for Maximizer Player
   maxEva= -infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, False)
  maxEva= max(maxEva, eva)
  alpha= max(alpha, maxEva)
   if beta<=alpha
 break
 return maxEva

else                          // for Minimizer player
   minEva= +infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, true)
   minEva= min(minEva, eva)
   beta= min(beta, eva)
    if beta<=alpha
 break
return minEva
```

Berikut merupakan *pseudocode* dari algoritma *minimax* dengan optimasi *alpha-beta pruning* yang diambil dari https://www.javatpoint.com/mini-max-algorithm-in-ai . Terlihat pada optimasi *alpha-beta pruning* terdapat parameter tambahan pada fungsinya yaitu *alpha* dan *beta*. Untuk mempercepat penelusuran seperti yang telah disebutkan sebelumnya ada pemangkasan pada pemangkasan yaitu ketika kondisi α>=β sudah tidak dipenuhi karena dianggap simpul tersebut tidak akan sampai kepada simpul optimal.

III. IMPLEMENTASI DAN PEMBAHASAN

Algoritma *minimax* dengan optimasi *alpha-beta pruning* dipilih sebagai strategi penyelesaian permainan Othello. Pemilihan algoritma ini dikarenakan *minimax* merupakan algoritma yang biasa dipakai pada permainan dua pemain yang saling bergiliran sehingga pada setiap giliran bisa dilakukan perhitungan untuk langkah optimal selanjutnya. Namun, algoritma *minimax* menggunakan DFS sebagai mekanisme penelusurannya yang berarti untuk pencarian setiap kemungkinan akan berambah secara eksponensial setiap adanya penambahakan kedalaman. Oleh karena itu, digunakan optimasi *alpha-beta pruning* untuk memangkas simpul yang dianggap tidak akan memberikan langkah yang optimal.

Dalam implementasi strategi *minimax* dengan *alpha-beta pruning* untuk permainan Othello, digunakan pohon ruang status dengan tiap simpulnya menggambarkan setiap kemungkinan pada kedalaman tersebut dan untuk kedalaman berikutnya menggambarkan langkah berikutnya lagi. Tiap simpul akan mempunyai nilainya berdasarkan nilai pada kedalaman selanjutnya dan terus menerus hingga mencapai simpul daun.

Sebelum dilakukan keputusan berdasarkan algoritma yang digunakan diperlukan penghasil semua langkah yang dapat dilakukan. Untuk memudahkan dilakukan iterasi terhadap setiap kemungkinan petak pada papan 8 x 8 dan akan digunakan fungsi validasi untuk memvalidasi gerakan tersebut. Hanya gerakan yang valid yang akan dijadikkan simpul pada pohon ruang status.

Setiap simpul akan dibangkitkan tetapi tidak mempunyai nilainya terlebih dahulu sesuai dengan algoritma *minimax*. Nilai dari simpul baru akan didefinisikan ketika simpul merupakan simpul daun yaitu merupakan *terminal nodes* atau sudah tidak bisa dilakukan gerakan lanjutan. Nilai dari simpul

tersebut akan dihitung berdasarkan posisi dari tiap kepingnya dan akan dijumlahkan berdasarkan posisinya pada papan.

5	3	3	3	3	3	3	5
3	1	1	1	1	1	1	3
3	1	1	1	1	1	1	3
3	1	1	1	1	1	1	3
3	1	1	1	1	1	1	3
3	1	1	1	1	1	1	3
5	3	3	3	3	3	3	5

Gambar 3.1 Nilai Posisi Tiap Keping

Setelah semua simpul terbangkitkan dan simpul daun sudah memiliki nilai, maka akan dilakukan *backtracking* dari simpul daun tersebut dan akan dilakukan *mini-maxing* di setiap simpulnya hingga mencapai simpul akar. Dari situ pengambilan keputusan akan dilakukan.

```
def alphaBeta(self,node,depth,alpha,beta,maximizing):
    global nodes
    nodes += 1
    boards = []
    choices = []

    for x in range(8):
      for y in range(8):
        if valid(self.array,self.player,x,y):
          test = move(node,x,y)
          boards.append(test)
          choices.append([x,y])

    if depth==0 or len(choices)==0:
      return ([finalHeuristic(node,maximizing),node])

    if maximizing:
      v = -float("inf")
      bestBoard = []
      bestChoice = []
      for board in boards:
        boardValue = self.alphaBeta(board,depth-1,alpha,beta,0)[0]
        if boardValue>v:
          v = boardValue
          bestBoard = board
          bestChoice = choices[boards.index(board)]
        alpha = max(alpha,v)
        if beta <= alpha:
          break
      return([v,bestBoard,bestChoice])
    else:
      v = float("inf")
      bestBoard = []
      bestChoice = []
      for board in boards:
        boardValue = self.alphaBeta(board,depth-1,alpha,beta,1)[0]
        if boardValue<v:
          v = boardValue
          bestBoard = board
          bestChoice = choices[boards.index(board)]
        beta = min(beta,v)
        if beta<=alpha:
          break
      return([v,bestBoard,bestChoice])
```

Berikut merupakan implementasi strategi permainan Othello dengan algoritma *minimax* optimasi *alpha-beta pruning* dalam bahasa Python dengan pemrograman berorientasi objek. Implementasi tersebut memerlukan inisialisasi *depth* pada awal kode. *Depth* bisa diatur dengan nilai *depth* lebih tinggi berarti penelusuran akan dilakukan secara lebih mendalam dan akan menghasilkan langkah yang lebih optimal.

IV. EKSPERIMEN DAN HASIL

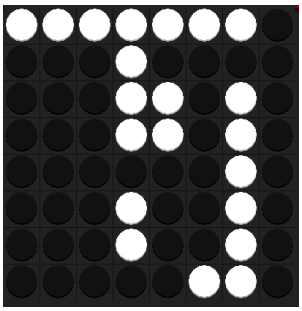
Untuk menguji hasil implementasi akan dilakukan pengujian. Pengujian akan dilakukan dengan nilai *depth* yaitu 4 agar waku penelusuran tidak terlalu lama. Subjek yang akan menjadi tes uji ada 2 yaitu manusia dan komputer lain secara daring.

A. Pengujian Dengan Manusia

Uji dengan manusia akan dilakukan dengan mempertandingkan antara penulis dan program. Penulis merupakan pemain yang awam dalam bermain Othello sehingga menjadi representatif sebagai pemain awam.

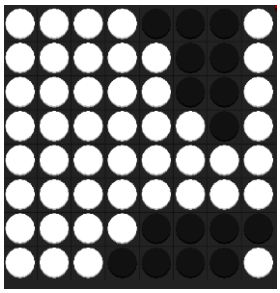
Pengujian akan dilakukan sebanyak 3 kali dengan Penulis memulai pertama kali dengan keping putih dan program akan menggunakan keping hitam.

1) Percobaan 1



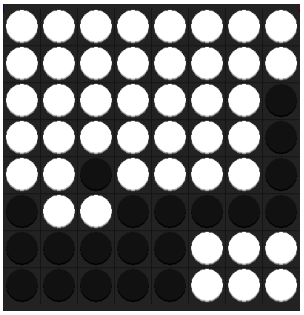
Gambar 4.1 Percobaan 1 Uji Manusia

2) Percobaan 2



Gambar 4.2 Percobaan 2 Uji Manusia

3) Percobaan 3



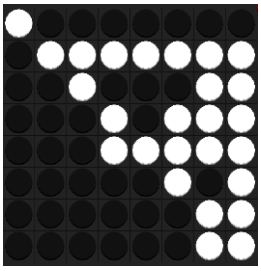
Gambar 4.3 Percobaan 3 Uji Manusia

Dari 3 percobaan yang dilakukan dengan Penulis, hasil yang didapatkan adalah Penulis menang 2 kali, sedangkan program menang 1 kali. Disini penulis mendapatkan giliran pertama dan menggunakan strategi mengambil yang paling ujung.

B. Pengujian dengan Komputer Lain

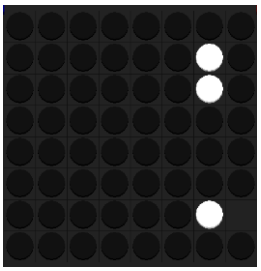
Pengujian akan dilakukan dengan komputer lain secara daring dari komputer Othello yang tersedia pada laman https://www.eothello.com/ . Langkah pengujian adalah dengan memodifikasi program yang tadi agar program tersebut bergerak duluan. Kepingan hitam adalah program implementasi *minimax* dengan *alpha-beta pruning*, sedangkan kepingan putih adalah komputer lain. Pengujian dilakukan dengan mengikuti langkah kepingan putih yang dihasilkan oleh komputer lain. Pengujian ini juga akan dilakukan sebanyak 3 kali.

1) Percobaan 1



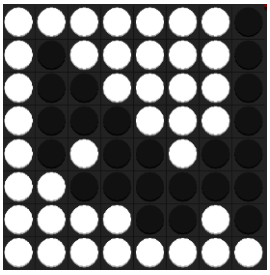
Gambar 4.4 Percobaan 1 Uji Komputer Lain

2) Percobaan 2



Gambar 4.5 Percobaan 2 Uji Komputer Lain

3) Percobaan 3



Gambar 4.6 Percobaan 3 Uji Komputer Lain

Dari 3 percobaan yang dilakukan dengan komputer lain, hasil yang didapatkan adalah program menang 2 kali, sedangkan komputer lain menang 1 kali.

C. Analisis

Berdasarkan pengujian yang dilakukan dengan manusia, hasilnya tentu tidak menggambarkan dengan jelas kemampuan dari algoritma ini. Dikarenakan pengambilan keputusan yang dilakukan oleh subjek ujinya, dalam hal ini merupakan Penulis, tidak berdasarkan perhitungan tertentu seperti program. Beda halnya dengan program yang menggunakan algoritma. Penulis pada saat pengujian mengalami perkembangan juga dalam pola berpikir bermain Othello, sehingga bisa memenangkan pada pengujian 2 dan 3. Penulis hanya mendasarkan pengambilan keputusan langkahnya dari mencari keping yang akan mengambil hasil paling optimis menurut Penulis itu sendiri.

Berbeda dengan pengujian yang dilakukan dengan manusia, Pengujian dengan komputer lain berarti program melawan program. Komputer lain tersebut merujuk kepada komputer Othello pada laman https://www.eothello.com/. Pada laman tersebut tidak dijelaskan bahwa komputernya menggunakan algoritma tertentu. Saat dilakukan pengujian, program algoritma *minimax* dengan optimasi *alpha-beta pruning* mendapatkan giliran pertama dan kepingan hitam, sedangkan komputer lain mendapatkan kepingan putih. Sehingga, program ini mendapatkan giliran pertama secara terus menerus dan Penulis mengamati bahwa pengambilan keputusan pertama kali dari program selalu konsisten, tetapi komputer lain menanggapi langkah pertama yang konsisten tersebut dengan berbeda-beda di setiap percobaan.

Pengujian dapat dilakukan dengan lebih banyak lagi dan dengan subjek uji yang lain untuk mendapatkan hasil yang

lebih valid. Algoritma *minimax* sangat bergantung terhadap fungsi evaluasi yang digunakan. Fungsi evaluasi bertugas untuk mengevaluasi setiap simpul dan menghasilkan nilai berdasarkan evaluasi tersebut, nilai yang dipakai pada program ini sesuai pada Gambar 3.1 . Dengan fungsi evaluasi yang berbeda, algoritma *minimax* juga dapat menghasilkan langkah yang berbeda juga. Fungsi evaluasi yang dipakai pada program ini dapat dioptimalkan lagi dengan melakukan analisis yang lebih jauh terhadap nilai dari posisi setiap kepingnya.

Terdapat beberapa strategi lain juga yang dapat diuji yaitu *maximum disc strategy*, *weighted square strategy*, dan *minimum disc strategy* [4].

V. KESIMPULAN

Permainan Othello merupakan permainan papan berbasis giliran / *turn-based*. Layaknya permainan papan lainnya seperti Catur, Go, ataupun Tic-Tac-Toe, Othello memerlukan strategi dalam memainkannya jika ingin mendapatkan hasil yang optimal. Dalam pembentukan strategi tersebut dapat digunakan berbagai algoritma untuk menghasilkan strategi tersebut. Pada makalah kali ini, Penulis memilih untuk menggunakan algoritma *minimax* karena algoritma tersebut merupakan algoritma untuk menentukan langkah berbasis meminimalisasi kerugian berdasarkan asumsi langkah lawan. Sehingga, Penulis merasa adanya kecocokan antara algoritma tersebut dengan permainan Othello yang merupakan permainan berbasis giliran.

Pada implementasinya, algoritma *minimax* memerlukan waktu yang cukup lama dalam penelusuran langkahnya dikarenakan pencarian dilakukan terhadap seluruh simpul yang dibangkitkan, sehingga diperlukannya optimasi untuk memangkas simpul yang sudah dianggap tidak menuju ke solusi dengan menggunakan optimasi *alpha-beta pruning*. Optimasi ini menambahkan parameter *alpha* dan *beta* sebagai nilai MAX dan MIN pada saat itu dan juga menambahkan kondisi untuk pemberhentian penelusuran pada langkah tersebut.

Pada pengujian yang dilakukan, algoritma *minimax* dengan *alpha-beta pruning* menggunakan pembatasan dalam pencariannya sampai kedalaman 4 agar penelusuran tidak memakan terlalu banyak waktu. Berdasarkan 6 pengujian yang dilakukan terhadap manusia dan juga komputer lain, strategi Othello dengan menggunakan algoritma *minimax* ini menang setengahnya. Pengujian memang masih terlalu sedikit karena dilakukan secara manual dan terbatas oleh waktu, tetapi langkah yang dilakukan dari program tersebut sudah sesuai dengan ekspetasi dari Penulis.

VIDEO LINK AT YOUTUBE

https://bit.ly/VideoMakalahStima13520044

ACKNOWLEDGMENT

Penulis ingin berterima kasih kepada Tuhan Yang Maha Esa karena berka izin-Nya, makalah ini dapat diselesaikan. Penulis juga ingin mengucapkan terimakasih terhadap semua pihak yang terlibat dalam membantu penulis menyelesaikan makalah "Penerapan Algoritma *Minimax* dengan *Alpha-Beta Pruning* sebagai Strategi Permainan Othello". Tentunya, terimakasih juga penulis tujukan terhadap Ibu Dr. Nur Ulfa Maulidevi, ST., M.SC. dan dosen pengampu mata kuliah IF2211 Strategi Algoritma lainnya, beserta asisten yang turut membantu pelaksanaan mata kuliah.

Penulis merasa bersyukur telah diberikan pengalaman yang menarik dalam pengerjaan makalah ini. Akhir kata, mohon maaf jika terdapat kesalahan makna ataupun kata yang ada pada makalah ini.

REFERENCES

[1] Othello Museum. *Brief History of Othello*.

[2] Author Tanpa Nama. *Mini-Max Algorithm in Artificial Intelligence*. Dikunjungi 21 Mei 2022 dari https://www.javatpoint.com/mini-max-algorithm-in-ai

[3] Author Tanpa Nama. *Alpha-Beta Pruning.* Dikunjungi 21 Mei 2022 dari https://www.javatpoint.com/ai-alpha-beta-pruning

[4] S. Rosenbloom, Paul. 1981. *A World-Championship-Level Othello Program.*

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2022

Adiyansa Prasetya Wicaksana 13520044