Penyelesaian Game Tic-Tac-Toe Dengan Algoritma Minimax

Yudy Valentino/13517128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517128@std.stei.itb.ac.id

***Abstrak*—Tic Tc Toe adalah permainan papan yang dimainkan oleh dua orang dengan menggunakan dua buah simbol x atau o, dan permainan tic-tac-toe dapat diselesaikan dengan algoritma minimax.**

Kata Kunci—Algoritma; Minimax; Tic-Tac-Toe; Gameboard

I. PENDAHULUAN

Perkembangan Zaman di era globalisasi saat ini sungguh pesat, hampir tiap sektor dari kehidupan manusia disentuh oleh yang namanya teknologi, hal ini semakin marak dan semakin meluas dengan revolusi industri 4.0 yang sedang terjadi. Salah satu aspek yang sangat mempengaruhi dan mempermudah kehidupan manusia adalah kecerdasan buatan *(Artificial Intelligence)*, kecerdasan buatan ini merupakan salah satu inovasi dari akibat berkembangannya ilmu pengetahuan terutama perkembangan komputer modern di sekitar tahun 1950.

Kecerdasan buatan memudahkan manusia dalam melakukan segala sesuatu, kecerdasan buatan menyentuh hampir dari seluruh aspek hidup manusia dengan salah satu contohnya adalah dibidang entertainment / hiburan. Hiburan bagi manusia dapat bermacam-macam, bisa film, musik, permainan, dll. Jika pada masa lampau, permainan hanya bisa dimainkan jika jumlah pemainnya cukup, atau menggunakan sesuatu alat, dengan kecerdasan buatan semuanya itu mungkin untuk dilakukan, salah satunya adalah permainan tic-tac-toe.

Tic-Tac-Toe adalah permainan papan yang dimainkan oleh dua orang dengan menggunakan strategi untuk mencapai kemenangan. Pemain bebas menggunakan simbol x atau o sebagai bidak yang akan melangkah di papan permainan sebesar n x n. dengan kondisi menang yaitu pada saat terdapat tiga buah bidak yang sama di posisi sejajar, baik itu vertikal, horizontal, ataupun diagonal.

Gambar 1.1 Tic-Tac-Toe

Meskipun tidak ada yang mengetahui masa depan secara pasti, tetapi kecerdasan buatan mempunyai dampak bagi kehidupan manusia, karena kecerdasaan buatan itu menyediakan solusi masalah kehidupan yang nyata. Kecerdasan buatan sangat menarik untuk dipahami dan dipelajari terutama dengan pernananya yang tidak lepas dari kehidupan manusia.

II. LANDASAN TEORI

A. *ALGORITMA DFS*

Algoritma DFS(Depth First Search) adalah algoritma rekursif yang memanfaatkan konsep dari backtracking. Algoritma ini akan mencari semua penelusuran secara vertikal selama itu masih memungkinkan untuk dilakukan, jika tidak menemui node yang tepat ia akan kembali ketempat sebelumnya dengan menggunakan baacktracking.

Backtracking yang dilakukan adalah dalam jalur node yang sama secara vertikal, sehingga dari sebuah node awal akan dilakukan pengecekan hingga mencapai tahap akhir kemudian beralih jika tidak seusai. Sedangkan rekursif yang dilakukan pada algoritma DFS ini adalah teori dari stack. Secara garis besar cara kerja dari stack yang digunakan dalam alogritma ini adalah setelah dipilih sebuah node awal, dilakukan semua push node yang sejajar secara vertikal terhadap node awal tersebut kedalam stack, kemudian dipilih satu node dengan konsep LIFO (Last In First Out) dilakukan pop dan dipilih node dari stack untuk ditelusuri jalurnya. Ulangi prose tersebut hingga stack nya menjadi kosong.

Untuk memastikan tidak ada infinite loop pada proses pencarian maka dilakukan penandaan pada node yang sudah diperiksa dan tidak menghasilkan solusi yang sesuai. Algoritma DFS ini juga dapat menggunakan alpha beta tunning untuk mereduksi jumlah pencarian node yang seharusnya dilakukan.

Pseudo code dari Algoritma DFS :

```
DFS-iterative (G, s):     //G adalah graf dan s adalah source vertex
let S be stack
    S.push( s )           //sisipkan s ke stack
    mark s as visited.
    while ( S is not empty):
```

```
        //Pop verteks dari stack untuk menelusuri next
        v = S.top( )
        S.pop( )
        //Push semua neighbour dari v pada stack yang belum
ditelusuri
        for all neighbours w of v in Graph G:
          if w is not visited :
                S.push( w )
               mark w as visited
    DFS-recursive(G, s):
      mark s as visited
      for all neighbours w of s in Graph G:
        if w is not visited:
          DFS-recursive(G, w)
```

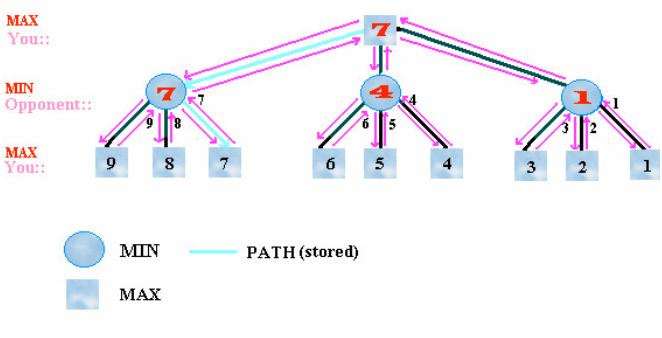
Tabel 1 Pseudocode Algoritma DFS

B. ALGORITMA MINIMAX

Algoritma Minimax adlah algoritma yang sering dipakai untuk menentukan sebuah pilihan, sehingga pilihan yang kita tentukan tersebut tidak kehilangan nilai maksimalnya. Kecerdasan buata (Artificial Intelligence) adalah salah satu bidang ilmu komputer yang paling banyak menerapkan algoritma ini, baik dalam menentukan pilihan hiburan untuk manusia, menentukan jenis makanan yang akan dimakan esok hari, bahkan hingga menentukan alur permainan sebuah game seperti tic-tac-toe, go, checkers,dll.

Algoritma ini berkembang dengan adanya teori *zero-sum* dimana jika sesorang mempunyai sebuah pilihan dengan nilai pilihan tersebut tergantung dalam konteks yang sedang dihadapinya maka ada pemain lain yang akan kehilangan nilai dari pilihannya sesuai dengan nilai yang didapatkan oleh orang pertama.

Algoritma minimax merupakan salah satu algoritma dasar pencarian DFS, dimana setelah dilakukan pemilihan node awal, ia akan terus membangkitkan simpul dengan node lebih dalam hingga tercapainya sebuah solusi, jika tidak ia akan melakukan backtracking. Dalam reprensentasi algoritma DFS di dalam pohon pencarian hasil dari algoritma minimax, algoritma minimax mempunyai dua buah node didalam pohono pencariannya yaitu max dan min, dimana max untuk mendapatkan nilai tertinggi dan min kebalikannya. Dalam algoritma minimax, langkah sebuah pemain ditentukan berdasarkan dari pilihan pemain lainnya.



Gambar 2.1 Pohon Pencarian Algoritma Minimax

Dengan demikian algoritma minimax mempunyai fungsi heuristik untuk melakukan evaluasi hasil dari pilihan yang dipilih, sebagai contoh dalam permainan tic-tac-toe digunakan nilai 1,0,-1 untuk fungsi heuristiknya.

Pseudocode dari algoritma minimax:

```
function MinMax (game : GamePosition)
return MaxMove (game)

function MaxMove (game : GamePosition, alpha : integer,
beta: integer)
if (GameEnded(game)) then
        return EvalGameState(game)
else
        best_move ← ()
        moves ← GenerateMoves(game)
        ForEach moves do
                move ← MinMove(ApplyMove(game),
alpha, beta)
                if (Value(move) > Value(best_move))
then
                        best_move ← move
                        alpha ← Value(move)
                endif
                if (beta>alpha)
                        return best_move
                endif
        endfor
        return best_move
endif

function MinMove (game : GamePosition, alpha : integer,
beta : integer)
if (GameEnded(game)) then
        return EvalGameState(game)
else
        best_move ← ()
        moves  GenerateMoves(game)
        ForEach moves do
                move ← MaxMove(ApplyMove(game),
alpha, beta)
                if (Value(move) > Value(best_move))
then
                        best_move ← move
                        beta ← Value(move)
                endif
                if (beta<alpha)
                        return move
                endif
        endfor
        return best_move
endif
```

Tabel 2 Pseudocode Algortima Minimax

C. *TIC-TAC-TOE*

Tic-Tac_toe merupakan permainan papan yang berukuran 3x3. Tic-tac-toe ini dimainkan oleh dua orang dengan menggunakan bida x atau o untuk mengisi papan permainan. Kondisi menang yang dicapai adalah jika salah satu bidak mempunyai 3 buah bidaknya yang sejajar baik itu vertikal, horizontal, ataupun diagonal.

III. ALGORITMA MINIMAX DALAM TIC-TAC-TOE

Untuk menyelesaikan permasalahan tic-tac-toe dengan menggunakan algoritma minimax, berikut adalah kode dari fungsi minimax yang ditulis dalam Bahasa python

```
def minimax(state, depth, player):
    if player == MAX:
        best = [-1, -1, -999]
    else:
        best = [-1, -1, +999]

    if depth == 0 or game_over(state):
        score = evaluate(state)
        return [-1, -1, score]

    for cell in empty_cells(state):
        x, y = cell[0], cell[1]
        state[x][y] = player
        score = minimax(state, depth - 1, -player)
        state[x][y] = 0
        score[0], score[1] = x, y

        if player == MAX:
            if score[2] > best[2]:
                best = score  # max value
        else:
            if score[2] < best[2]:
                best = score  # min value

    return best
```

Tabel 3 Algoritma Minimax dalam Python

Untuk penjelasan kode dari algoritma minimax tersebut adalah sebagai berikut:

Kondisi awal yang perlu diketahui adalah MAX bernilai 1 sedangkan MIN bernilai -1, dimana untuk node MAX dan MIN dapat berupa x ataupun o, dengan papan permainan yang dipakai disini adalah 3x3.

Parameter dari masukan fungsi minimax ini adalah state, depth, player. State adalah kondisi terkini yang ada di papan permainan, kondisi ini menunjukkan node pada pohon pencarian algoritma minimax, depth adalah kedalam dari pohon pencarian yang dilakukan sedangkan player adalah pemain yang sedang memainkan permainan tic-tac-toe yaitu bisa menjadi pemain dengan node max ataupun dengan node min.

```
    if player == MAX:
        best = [-1, -1, -999]
    else:
        best = [-1, -1, +999]
```

Saat memulai game, baik pemain maupun komputer akan diset dengan kasus score terburuk, dalam kasus ini dipilih untuk kasus score terburuk adalah 999. Jika pemain adalah node MAX maka score terburuknya diset ke angka -999 sedangkan jika pemain merupakan node MIN maka score akan diset ke angka 999. Pada saat permulaan permainan, langkah terbaik untuk pemain dan komputer adalah -1,-1 dikarenakan belum ada yang meletakkan bidaknya dan kondisi papan permainan masih kosong.

Setelah dilakukan inisiasi langkah pertama saat memulai permainan, dilakukan pengecekan apakah kondisi yang ada di papan permainan sekarang merupakan kondisi akhir bagi kedua player baik itu pemain maupun komputer atau ketikan depth dari kondisi papan sudah mencapai 0, depth / kedalaman yang dipakai disini adalah depth 0 saat sudah mencapai kondisi akhir, dan naik satu node akan mengurangi nilai node tersebut. Jika sudah kondisi akhir maka akan dipanggil fungsi evaluate untuk menghitung dari score yahng didapatkan

Berikut adalah potongan kode dari kode fungsi keseluruhan:

```
    if depth == 0 or game_over(state):
        score = evaluate(state)
        return [-1, -1, score]
```

Untuk fungsi rekursif dari algoritma minimax pada game tic-tac-toe ini dipanggil saat melakukan pengecekan didalam kotak kosong di papan permainan. Berikut adalah potongan kode dari keseluruhan fungsi:

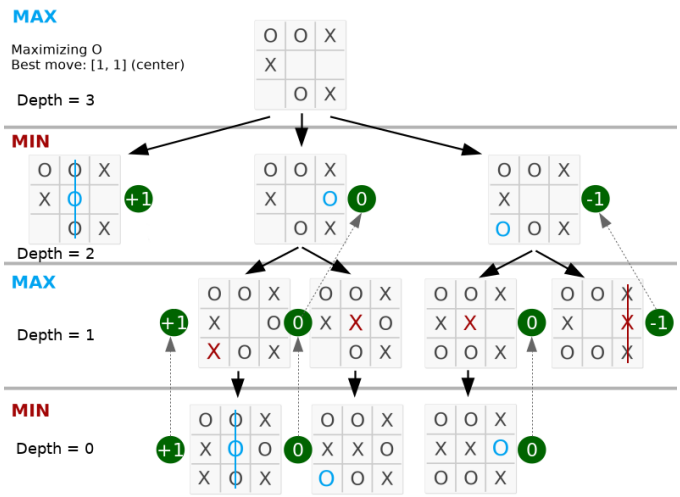
```
for cell in empty_cells(state):
        x, y = cell[0], cell[1]
        state[x][y] = player
        score = minimax(state, depth - 1, -player)
        state[x][y] = 0
        score[0], score[1] = x, y
```

Keterangan dari potongan kode diatas adalah sebagai berikut, x dan y merupakan reprensentasi dari baris dan kolom di dalam papan permainan, kemudian dilakukan pengecekan jika papan permainan pada kondisi indeks baris dan kolomnya x dan y, jika tersedia maka akan menerima MAX atau MIN dari player yang sedang bermain saat ini. Lalu dilakukan pemanggilan secara rekursif untuk menghitung score yang ada dengan state adalah kondisi papan saat ini didalam rekursi, depth / kedalamannya akan dikurangi 1 karena dalam pohon pencarian akan naik ke node parentnya dan menerima kode player jika player MAX maka 1 atau jika MIN maka akan -1.

```
        if player == MAX:
            if score[2] > best[2]:
                best = score  # max value
        else:
            if score[2] < best[2]:
                best = score  # min value
```

Kemudian tahap terakhir dari fungsi minimax ini adalah membandingkan score dengan kondisi best yang diinisiasi di awal pemanggilan fungsi minimax. Jika player adalah MAX maka nilai yang dikembalikan akan nilai yang lebih besar, sedangkan jika player MIN maka nilai yang dikembalikan akan nilai yang lebih kecil. Dan fungsi ini akan mengembalikan nilai yang terbaik

Algortima minimax ini menghasilkan pohon pencarian yang akan ditelusuri secara rekursif, berikut salah satu upapohon yang ada di pohon penelusuran dari algoritma minimax:



Keterangan dari gambar tersebut adalah saat bidak (x atau o) diletakkan dan menghasilkan keuntungan (dalam kasus game tic-tac-toe adalah kemenangan) maka proses akan langsung dihentikan. Contoh diatas merupakan sebagian kecil node yang ada di pohon pencarian dikarenakan total node dari keseluruhan game yang mungkin ada sekitar 549.946 node. Dari contoh diatas langkah terbaik ada di depth / kedalaman dua. Dengan alpha beta punning maka pohon penelusuran yang mempunyai banyak node dapat dipotong di tempat yang tidak perlu / yang sudah pasti tidak menghasilkan solusi yang terbaik

Untuk menguji algoritma minimax yang diimplementasikan dengan Bahasa pemrograman python, penulis memasukkan beberapa contoh hasil kompilasi program yang telah dibuat, contoh kompilasi program adalah sebagai berikut :

```
---------------
| O || X || X |
---------------
| X || X || O |
---------------
| O || O || X |
---------------
DRAW!
```

Contoh dari table tersebut adalah pemain memilih bidak x dan jalan terlebih dahulu, maka hasil akhir yang didapatkan adalah seri antara pemain dan komputer. Contoh lain dari kompilasi program adalah sebagai berikut:

```
Computer turn [O]
---------------
| X || X || O |
---------------
| X || O ||   |
---------------
| O ||   ||   |
---------------
YOU LOSE!
```

Contoh dari table tersebut adalah pemain memilih bidak x dan jalan terlebih dahulu, maka hasil akhir yang didapatkan adalah pemain kalah dari komputer. Contoh dari kompilasi program yang lain adalah sebagai berikut :

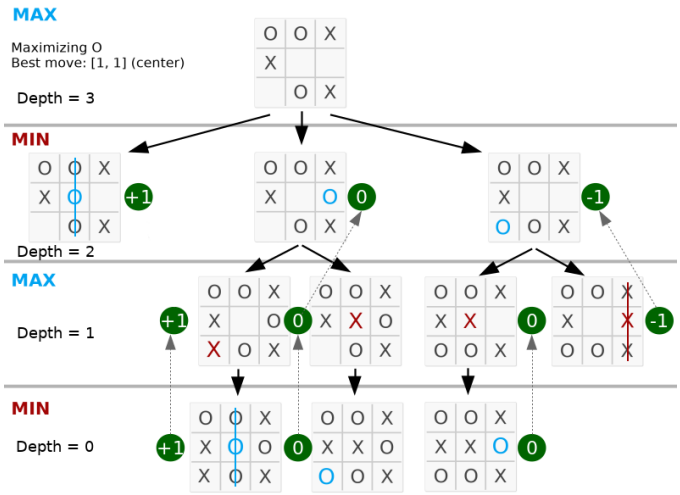
```
Human turn [X]
---------------
| X || O ||   |
---------------
| O ||   ||   |
---------------
|   ||   ||   |
---------------
Use numpad (1..9): 1
Bad move
Use numpad (1..9):
```

Contoh dari table tersebut adalah pemain memilih bidak x dan jalan setelah komputer, jika pemain memilih kotak yang sudah ada bidak lain maka aka nada pemberitahuan bahwa bad moves yang mengindikasikan bahwa pemain perlu memasukkan ke kotak yang lain.

IV. ANALISIS DARI PROGRAM

Berdasarkan beberapa percobaan memainkan permainan tic-tac-toe, pemain tidak bisa mendapatkan kemenangan melawan komputer hal ini dapat ditinjau dari komponen secara garis besar algoritma minimax itu sendiri yaitu :

1. Initial State : Keadaan papan sebelum dijalankan
2. Player : Kesempatan pemain untuk meleteakkan bidak pada kotak yang kosong di papan permainan
3. Action : Himpunan pergerakan yang dapat dilakukan oleh player
4. Result : Keadaan papan setelah aksi action dilakukan
5. Terminal Test : Fungsi pengecekan tujuan , apakah kondisi menang sudah diperoleh atau belum
6. Utility Function : nilai evalulasi kemungkinan kemenganan



Analisis mendapatkan solusi yang terbaik untuk pergerakan komputer berdasarkan komponen dari algoritma minimax adalah:

1. Kondisi initial state pada gambar diatas adalah saat kotak no 4, 5, dan 6 belum terisi
2. Player mendapatkan kesempatan untuk meletakkan bidaknya di kotak yang belum terisi
3. Action / himpunan pergerakan dari komputer menurut gambar diatas ada 3 yaitu komputer dapat meletakkan bidak di kotak no 4, 5, atau 6
4. Result yang didapatkan dari kemungkinan tiga pergerakan tersebut adalah sebagai berikut:
    a. Jika komputer meletakkan bidak 0 di kotak kosong no 4 maka permainan langsung diberhentikan dengan status kemenangan untuk komputer
    b. Jika komputer meletakkan di kotak no 5 maka nilai dari best score yang didapatkan dari state saat itu adalah 0, karena proses yang dilakukan tidak memberikan keuntungan bagi komputer
    c. Jika komputer meletakkan bidak o di kotak no 6 maka best score yang dihasilkan akan mengembalikan nilai -1 karena pergerakan komputer memberikan keuntungan buat pemain dan kerugian untuk komputer

Kemudian dilakukan secara backtrack dengan menggunakan fungsi rekursif yang dipanggil di algoritma minimax

5. Terminal Test dilakukan di setiap komputer sudah meletakkan bidak o, apakah salah satu dari delapan kondisi menang sudah dicapai atau belum
6. Utility Function dilakukan untuk melakukan return dari nilai score yang didapatkan setelah tercapainya salah satu kondisi menang

Berdasarkan analisis yang dilakukan oleh penulis, komputer yang menggunakan algoritma minimax sebagai dasar ia meletakkan bidaknya tidak akan pernah kalah, karena komputer akan selalu meminimalisir kemungkinan pemain untuk menang dan melakukan blok di setiap kemungkinan pemain akan menang, hasil dari pergerakkan tersebut adalah seri, namu jika pemain salah memasukkan langkah maka minimax akan langsung memanfaatkan kesempatan itu untuk memperoleh kemenangan.

KESIMPULAN

Kesimpulan yang didapat melalui penulisan masalah ini adalah, algoritma minimax merupakan algoritma yang mangkus dan sangat bagus dalam menentukan pilihan di permainan-permainan berbasis kecerdasan buatan dengan player yang lebih dari dua, jika data atau pohon penelusuran yang dihasilkan tidak begitu luas dan besar. Namun untuk permainan yang memerlurkan data yang sangat besar sehingga pohon penelusurannyapun akan semakin besar, algoritma minimax dapat dilakukan penambahan alpha-beta punning untuk memaksimalkan sistem pencarian minimax yang menggunakan rekursif, dengan memotong node-node dari pohon penelusuran yang sudah pasti tidak akan ditelusuri atau dapat membatasi kedalam dari suatu pohon penelusuran. Fungsi heuristik yang digunakan juga berpengaruh besar untuk menentukan keputusan yang diambil oleh kecerdasan buatan, dan untuk kasus algoritma minimax di permainan tic-tac-toe ini pemain tidak dapat menang melawan kecerdasan buatan yang diwakilkan oleh komputer

UCAPAN TERIMA KASIH

Pertama – tama penulis mengucapkan terimakasih kepada Tuhan Yang Maha Esa atas berkat-Nya, penulis dapat menyelesaikan tugas makalah ini. Penulis juga mengucapkan terimakasih kepada seluruh dosen pengampu mata kuliah IF2211 Strategi Algoritma dalam membimbing penulis dalam menyelesaikan tugas makalah ini, dan yang terakhit penulis mengucapkan terimakasih kepada teman-teman peserta mata kuliah IF2211 Strategi Algotrima yang selalu mendukung dalam penyelesaian makalah ini.

DAFTAR REFERENSI

[1] https://indonesia.hackerearth.com/konsep-depth-first-search-dfs/

[2] http://catatankecilanaknegeri.blogspot.com/2014/12/algoritma-minimax-pada-permainan-tic.html

[3] https://www.academia.edu/18735022/ALGORITMA_MINIMAX

[4] http://freakermedia.blogspot.com/2016/01/pengertian-algoritma-minimax.html

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/BFS-dan-DFS-(2019).pdf

[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Runut-balik-(2018).pdf

[7] https://www.geeksforgeeks.org/backtracking-algorithms/

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, 26 April 2019

Yudy Valentino / 13517128