ALGORITMA MINIMAX SEBAGAI AI (*ARTIFICIAL INTELEGENCE*) DALAM PERMAINAN TIC TAC TOE

Adi Suripiyanto (10111315)

Jurusan Teknik Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Indonesia

ABSTRAK

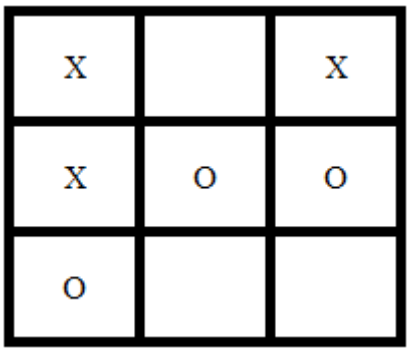
Permainan dalam komputer berkembang dengan sangat cepat, tidak hanya permainan yang mengandalkan grafik dan keindahan saja yang banyak dipandang. Game-game dasar dengan tingkat kesulitan dan keberagaman teknik penyelesaian juga mampu menghipnotis banyak penggunanya. Contoh permainan yang tidak banyak mengandalkan grafik adalah permainan Catur, Math Maze, Labirin, Tic Tac Toe, dan Game sejenisnya. Game telah berkembang dengan banyak mode. Seperti Tic Tac Toe game dengan Genre *Arcade,* Final Fantasy sebagai game *Adventure,* Fast and Furious sebagai game *Racing.* Dari semua game, Tic Tac Toe sangat ringan untuk dimainkan namun memerlukan taktik agar lawan lengah.

Dalam permainan tersebut tentu kita tidak selalu mempunyai lawan tanding, dengan di buatnya AI (*Artificial Intelegence)* kita dapat bermain kapanpun kita inginkan. Permainan Tic Tac Toe merupakan permainan sederhana yang dimainkan oleh dua orang. Game jenis *Arcade* ini memang selalu menarik dimainkan saat suntuk. Dalam membuat AI sebagai lawan tanding, Algoritma Minimax bisa dijadikan sebagai suatu solusi.

Kata Kunci : Algoritma *Minimax,* AI (*Artificial Intelegence*).

1. PENDAHULUAN

Tic Tac Toe merupakan game dimana dua player menaruh tanda silang "X" atau "O" pada matriks berukuran 3x3, dimana player akan dikatakan menang jika dia berhasil menaruh 3 tanda secara berurutan, baik vertical, horizontal atau diagonal. Dengan algoritma Minimax, algoritma yang merupakan basis dari semua permainan berbasis AI, seperti catur misalnya. Maka player tidak perlu mencari lawan main untuk bertanding, kita dapat bertanding dengan melawan komputer. Pada Algoritma minimax, pengecekan akan seluruh kemungkinan yang ada sampai akhir permainan. Pengecekan tersebut akan menghasilkan pohon permainan yang berisi semua kemungkinan agar AI bisa menang melawan player.



Gambar Tic Tac Toe melawan AI yang sedang berlangsung.

Algoritma Minimax merupakan dasar dari berbagai pembuatan AI diberbagai permainan. Namun jika AI yang dibentuk terlalu mudah, player tentu akan cepat merasa bosan, dalam hal ini algoritma Minimax merupakan dasar dari semua algoritma sebagai lawan tanding player, dan tingkat kompleksitas AI sebagai lawan bergantung pada metode dari algoritma yang digunakan.

2. METODOLOGI

2.1 Algoritma *MiniMax*

Minimax adalah suatu algoritma yang digunakan untuk menentukan skor dalam permainan setelah sejumlah langkah mempunyai nilai 0, 0 maksudnya permainan tidak dapat dilanjutkan lagi, atau telah mencapai akhir game, dengan pengambilan keputusan terbaik sesuai dengan fungsi evaluasi berdasarkan suatu ketentuan. Dalam pencarian satu lapis, dimana hanya ada langkah sequential dengan jumlah langkah yang diketahui, secara sederhana langkah yang ada dengan nilai terbaik akan terlihat setalah bermain dengan semua langkah yang mungkin. Langkah dengan evaluasi terbaik akan dipilih. Tapi untuk pencarian dua-lapis, saat lawan bergerak, hal menjadi lebih rumit. Lawan tentunya memilih langkah yang terbaik, dan pilihan tersisa adalah langkah dengan nilai yang lebih buruk dibanding langkah sebelumnya [1].

Algoritma minimax mencari solusi terbaik dengan melihat ke depan hingga ke akhir permainan dan memutuskan atau memilih langkah yang harus diambil saat itu untuk mencapai solusi tersebut. Evaluasi didasarkan pada nilai dari setiap kondisi pada level tersebut, apakah lebih memungkinkan player 1 untuk menang, lebih memungkinkan player 2 untuk menang atau seri

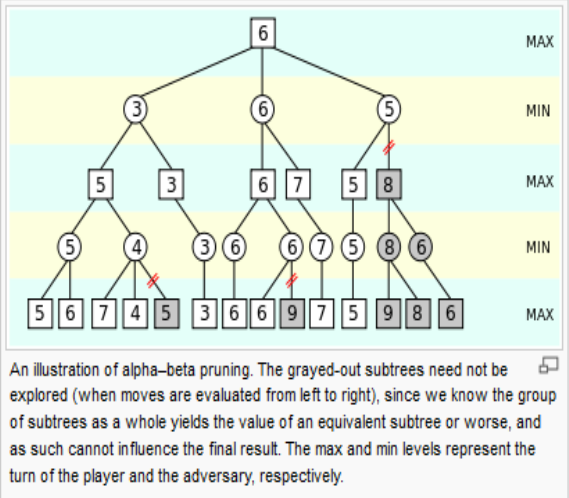
2.2 AI (*Artificial Intelegence*)

Kecerdasan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Konsep dari AI menggunakan *Heuristic* yang diambil dari bahasa Yunani yang berarti menemukan. *Heuristic* merupakan suatu strategi selektif, yang memandu proses pencarian yang kita lakukan disepanjang jalur yang memiliki kemungkinan sukses paling besar [2].

2.3 *Alpa-Beta Pruning*

Alpa Beta Pruning adalah algoritma pencarian yang bertujuan untuk mengurangi jumlah node atau kemungkinan yang akan dievaluasi lebih lanjut oleh algoritma minimax dalam proses pencarian solusi. Proses akan benar-benar berhenti mengevaluasi langkah ketika setidaknya satu kemungkinan telah ditemukan bahwa langkah tersebut lebih buruk dari langkah

yang diperiksa sebelumnya dan langkah tersebut tidak perlu dievaluasi lebih lanjut. Ketika diterapkan pada pohon minimax standar, ia mengembalikan langkah yang sama seperti minimax [3].



An illustration of alpha–beta pruning. The grayed-out subtrees need not be explored (when moves are evaluated from left to right), since we know the group of subtrees as a whole yields the value of an equivalent subtree or worse, and as such cannot influence the final result. The max and min levels represent the turn of the player and the adversary, respectively.

Ilustrasi alpa-beta pruning [3].

2.4 Pseudocode Algoritma Minimax

Secara umum, pseudocode untuk algoritma minimax yaitu:

```
minimax (simpul)
    if (simpul daun) then
        return pemenang
    else
        buat daftar semua simpul anak
        if (giliran max)
            return (nilai maksimum dari
                    semua simpul anak)
        else (giliran min)
            return (nilai minimum dari
                    semua simpul anak)
```

Keterangan:

Max adalah pemain 1 yang berusaha memperoleh nilai sebesar mungkin dengan mengambil langkah yang mendominasi secara vertical, horizontal atau diagonal dan mencegah pemain 2 agar memperoleh nilai seminimal mungkin.

Min adalah permain 2 yang berusaha memperoleh nilai sebesar mungkin dengan mengambil langkah yang mendominasi secara vertical, horizontal, atau diagonal dan mencegah pemain 1 untuk memperoleh nilai semaksimal mungkin.

2.4 Pseudocode Algoritma Alpha-Beta

Dalam algoritma alpha-beta, dicatat dua buah bilangan yaitu alpha dan beta, alpha adalah nilai dari langkah terbaik yang dapat dipilih, sedangkan beta adalah nilai dari langkah terbaik yang dapat dipilih oleh pemain lawan. Jika alpha>=beta, maka langkah yang dipilih oleh lawan dapat memaksa kita untuk mencapai posisi yang lebih buruk daripada langkah terbaik saat itu, sehingga langkah ini tidak perlu diperiksa(simpul dieliminasi).

Pseudocode untuk algoritma minimax yang ditambahkan *Alpa Beta Pruning* secara umum yaitu:

```
alpha-beta (pemain,simpul,alpha,beta)
    if (simpul daun)
        return pemenang

    buat daftar semua simpul anak
    if (pemain = max)
        for each (simpul anak)
            score = alpha-beta(min,anak,alpha,beta)
            if score > alpha then
            // ditemukan langkah yang lebih baik
                alpha = score
            if alpha >= beta then
            // potong
                return alpha
        return alpha
        // langkah terbaik (menghasilkan nilai
           maksimum)
    else (pemain = min)
        for each (simpul anak)
            score = alpha-beta(max,anak,alpha,beta)
            if score < beta then
            // ditemukan langkah yang lebih baik
               untuk lawan
                beta = score
            if alpha >= beta then
            // potong
                return beta
        return beta
        // langkah terbaik lawan (menghasilkan nilai
           maksimum)
```

Dapat dilihat dari pseudocode diatas bahwa algoritma alpha beta merupakan algoritma rekursif dank arena dalam permainan kedua pemain secara bergantian menjalankan langkah yang dipilih, ketika pemain max yang sedang diproses, maka algoritma akan dipanggil untuk pemain min.

Dapat dilihat dari pseudocode diatas bahwa algoritma tersebut merupakan rekursif dan karena dalam permainan kedua pemain antara AI dan player bergantian maka fungsi akan dipanggil baik setelah AI mengambil langkah, atau saat player mengambil langkah dan menentukan langkah terbaik yang akan diambil.

3. APLIKASI PADA PERMAINAN TIC TAC TOE

3.1 Aplikasi Algoritma Minimax

```
function AITurn(){
    if(giliran != AI) return;
    if(hitungKosong == 0) return;

    copyBoard();
    giliranAI = AI;

    // Algoritma Minimax
    var res,ci,cj,choose = -1000;

    for(var i = 0 ; i < kotakAI.length ; i++)
    {
        for(var j = 0 ; j < kotakAI[i].length ; j++)
        {
            if(kotakAI[i][j] == kosong)
            {
                kotakAI[i][j] = giliranAI;
                hitungKosong--;
                cekGiliran();
                res = search(1);
                kotakAI[i][j] = kosong;
                hitungKosong++;
                if(choose == -1000)
                {
                    choose = res;
                    ci = i;
                    cj = j;
                }
                else if(res > choose)
                {
                    choose = res;
                    ci = i;
                    cj = j;
                }
                cekGiliran();
            }
        }
    }
    // diulang sampai i==9 nilai res yang paling besar,yang dipilih. selama nilai res yang lebih besar tidak ditemukan, koordinat ci,cj tetap.
    currentGame[ci][cj] = AI;
    hitungKosong--;//mengurangi jumlah kotak, setiap fungsi ini dipanggil kotak-1

    setSign(ci,cj,AI,COLOR); //menampilkan giliran AI, X pada matriks.
    swapTurn();
}
```

Pada fungsi diatas, semua kemungkinan dicoba untuk mencari nilai terbaik pada langkah yang bisa diambil. Dalam permasalahan tersebut dibentuk suatu matriks dari i=0 sampai i=2, dan j=0 sampai j=2. Setiap giliran yang diambil baik AI maupun pemain, fungsi ini akan menggunakan copyBoard, dimana copyBoard ini dijadikan suatu pedoman bagi AI untuk mengetahui kotak mana saja yang telah terisi dalam matriks tersebut. Lalu memanggil fungsi ini lagi untuk menentukan langkah terbaik.

Saat pertama kali melakukan permisalan dengan suatu kondisi, AI akan menentukan apakah langkah tersebut baik atau tidak, jika baik berarti nilai lebih besar. Maka nilai itu akan dijadikan Pivot yang akan dibandingkan dengan perbandingan berikutnya.

Fungsi akan diulang sampe i=9 hingga mempunyai nilai terbesar. Lalu plotting posisi yang dipilih AI di simpan pada currentGame sebagai checkpoint suatu keadaan yang ada pada matriks. Hingga tercapai suatu kondisi dimana matriks tidak dapat terisi lagi dan kondisi tercapai, apakah AI yang memenangkan permainan dengan

ketentuan nilai terbesar didapat jika kotak terisi secara Diagonal, Vertical, atau Horizontal.

3.2 Aplikasi Algoritma Alpha Betha

```
function search(level)
{
    var res = determine(kotakAI);
    if(res == AI)/
    {
        return  100-level;
    }
    else if(res == player)
    {
        return level-100;
    }
    else if(res == "draw")
    {
        return 0;
    }

    var choose = -1000;
    var temp;

    for(var i = 0 ; i < kotakAI.length ; i++)
    {
        for(var j = 0 ; j < kotakAI[i].length ; j++)
        {
            if(kotakAI[i][j] == kosong)
            {
                kotakAI[i][j] = giliranAI;
                cekGiliran();
                hitungKosong--;
                //search merupakan fungsi rekursif,
                //dengan search bisa beganti jadi AI ata
                //untuk menentukan result yang menguntun
                temp = search(level + 1);
                cekGiliran();
                kotakAI[i][j] = kosong;//kotak yang tadi
                hitungKosong++;
                if(choose == -1000)
                {
                    choose = temp;
                }
                else if(giliranAI == AI)
                {
                    if(temp > choose) choose = temp;
                }
                else if(giliranAI == player)
                {
                    if(temp < choose) choose = temp;
                }
            }
        }
    }
    return choose;
}
```

Fungsi search akan dipanggil oleh fungsi Minimax, dimana fungsi ini berfungsi untuk mengeleminasi kemungkinan yang tidak diperlukan. Saat nilai terakhir ditemukan maka akan dikembalikan nilainya pada fungsi minimax.

4. KESIMPULAN

Kesimpulan yang dapat diperoleh dari pembahsan ini, adalah:

1. Algoritma Minimax dan alphe-betha dapat digunakan untuk mencari langkah terbaik yang dapat dipilih pada sebuah permainan.
2. Kompleksitas permainan bergantung pada banyaknya kotak atau cabang pada setiap kemungkinan, sehingga algoritma diatas untuk pencarian solusi terbatas hanya pada permainan Tic Tac Toe saja.
3. Pencarian minimax pada permainan yang kompleks belum tentu menghasilkan solusi terbaik karena keterbatasan pemrosesan. Karena algoritma ini pada dasarnya tidak membandingkan hasil dari tiap tiap solusi yang ada dan menentukan hasil yang terbaik, melainkan mencari tahap-tahap dengan nilai terbaik untuk menentukan hasil.
4. Algoritma minimax dapat digunakan dalam permainan Tic Tac Toe dengan matriks 3x3.

REFERENSI

[1] "Wikispaces," [Online]. Available: http://chessprogramming.wikispaces.com/Minimax. [Accessed Tuesday 1 2014].

[2] K. Gunadarma. [Online]. Available: http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=0CG8QFjAI&url=http%3A%2F%2Fwsilfi.staff.gunadarma.ac.id%2FDownloads%2Ffiles%2F4338%2F1-AI.pdf&ei=osziUtTpNuLMiAeeoICYAQ&usg=AFQjCNHjaewvdP9F6fG-VTUWtdoCKYy52g&sig2=C2sFg9EqbmdGRTU3jbH_Lg&b. [Accessed Tuesday 1 2014].

[3] "wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning. [Accessed Tuesday 1 2014].