Algoritma MiniMax untuk Mencari Solusi dalam *Game Tree* dari Permainan *Connect Four*

Gerard Edwin Theodorus (13507079)

Program Studi Teknik Informatika Institut Teknologi Bandung
Alamat : Jl. Ganesha 10, Bandung
e-mail: geedatz@gmail.com

ABSTRAK

***Game tree* merupakan konsep penting dalam pembuatan Inteligensia Buatan. *Game Tree* menggambarkan semua kemungkinan posisi dan langkah-langkah yang dapat dicapai dalam sebuah *game* atau permainan. Permainan yang dimaksud adalah permainan 2 orang dan hanya dapat berakhir dengan salah satu pemain menang atau seri. Contohnya permainan catur, tic-tac-toe, checkers, connect four, dan sebagainya. Semakin rumit dan besar suatu *game*, maka *game tree* dari *game* tersebut juga akan semakin kompleks dan dalam. Simpul dalam *game tree* menggambarkan kondisi permainan, sisi menggambarkan langkah yang memungkinkan dan simpul daun menggambarkan kondisi akhir permainan. Karena itu dibutuhkan suatu fungsi pencarian untuk memperoleh langkah terbaik agar memperoleh solusi yang terbaik (menang atau minimal seri). Namun, untuk permainan-permainan yang skalanya besar seperti catur, *game tree* lengkapnya memiliki ukuran yang sangat besar, sehingga umumnya hanya dilakukan pencarian hingga beberapa tingkat ke bawah. Algoritma minimax adalah algoritma pencarian dalam *game tree* yang menggunakan konsep meminimalkan hasil maksimal yang dapat diperoleh oleh pemain lain dan memaksimalkan hasil minimum yang diperoleh oleh diri sendiri. Dalam makalah ini akan dibahas mengenai konsep MiniMax, pengembangannya dengan *Alpha-Beta pruning*, dan contoh aplikasinya dalam permainan *Connect Four*.**

Kata kunci: MiniMax, *game tree*, Inteligensia Buatan, *alpha-beta pruning, Connect Four*

1. PENDAHULUAN

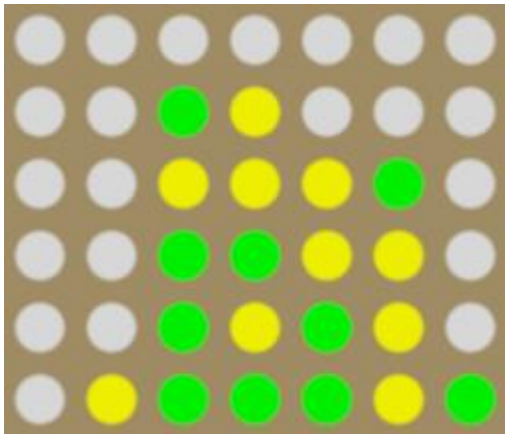
1.1 *Game Tree*

Game Tree adalah sebuah graf berarah (dari simpul akar ke simpul daun) dari sebuah permainan. *Game tree* lengkap dari sebuah permainan menggambarkan permainan mulai dari posisi awal hingga posisi akhir ketika permainan berakhir.

Simpul dari *game tree* menggambarkan kondisi permainan atau kondisi papan, sisi menggambarkan langkah yang dapat diambil dari suatu kondisi untuk mencapai kondisi lain, sedangkan simpul daun adalah simpul tujuan atau goal yang merupakan akhir permainan yang menyatakan kondisi pemain-1 menang, pemain-2 menang atau seri. *Game tree* yang dibahas pada makalah ini adalah *game tree* pada permainan yang memiliki ciri-ciri yaitu : dimainkan oleh dua orang pemain yang berbeda pihak dan *zero sum* (ada yang menang dan ada yang kalah atau seri, tidak ada kemenangan bersama).

1.2 *Connect Four*

Connect Four adalah sebuah permainan yang dimainkan oleh dua orang pemain. Papan permainan berukuran 7x6 dan diletakkan secara vertikal. Setiap pemain secara bergiliran meletakkan keping dengan warnanya pada salah satu dari 7 kolom pada papan. Pemain pertama yang meletakkan 4 buah keping secara berurutan baik secara vertikal, horizontal, ataupun diagonal.



Gambar 1. Contoh permainan *Connect Four* yang sedang berlangsung

1.3 Algoritma MiniMax

Minimax adalah salah satu algoritma pencarian yang dapat diterapkan dalam pencarian solusi dalam sebuah *game tree*. Algoritma minimax dibuat berdasarkan teorema :

"Untuk setiap permainan zero-sum dengan dua pemain dan strategi yang terbatas, ada sebuah nilai V dan strategi tertentu untuk setiap pemain sedemikian sehingga dengan strategi dari pemain-2, pemain-1 dapat memperoleh nilai terbaik V dan sebaliknya dengan strategi dari pemain-1, pemain-2 dapat memperoleh nilai terbaik –V."

Artinya, pemain-1 harus memperoleh nilai V untuk memenangkan permainan dan pemain-2 harus memperoleh nilai –V untuk memenangkan permainan. Nilai 0 menunjukkan bahwa permainan seri, dan nilai-nilai lain di antara –V dan V menunjukkan pemain yang lebih dominan atau menguasai permainan (peluang untuk menang lebih besar).

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. Namun, untuk permainan-permainan yang kompleks dan memiliki *game tree* yang berukuran sangat besar, akan sulit untuk memproses keseluruhan dari *game tree*. Pada kasus seperti ini, biasanya algoritma minimax hanya digunakan hingga beberapa level ke bawah saja dan langkah yang diambil ditentukan berdasarkan hasil evaluasi simpul-simpul pada level paling bawah. Evaluasi didasarkan pada nilai dari setiap kondisi pada level tersebut, apakah lebih memungkinkan pemain-1 untuk menang, lebih memungkinkan pemain-2 untuk menang atau seri.

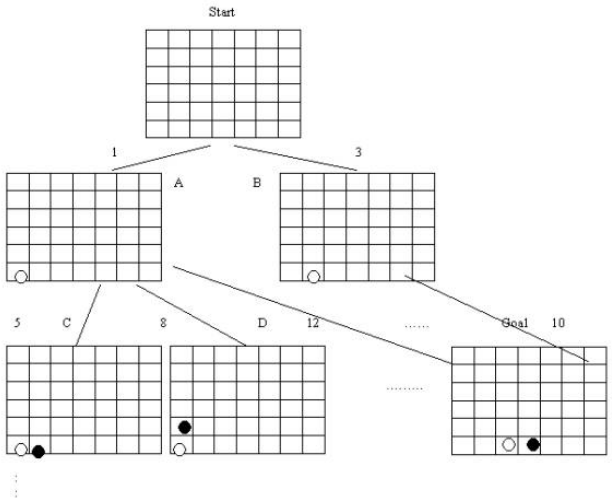
Algoritma minimax ini menggunakan asumsi bahwa pemain lawan juga memilih langkah yang maksimum untuknya.

Untuk membantu memangkuskan algoritma minimax, dapat digunakan beberapa algoritma seperti *alpha-beta pruning*, Negascout dan MTD(f). Yang akan dibahas pada makalah ini adalah *alpha-beta pruning*.

1.4 *Alpha-Beta Pruning*

Adalah algoritma yang mengurangi jumlah simpul yang harus diproses pada algoritma minimax. Algoritma ini "memotong" simpul yang sedang diproses jika ditemukan bahwa langkah yang dipilih tersebut lebih buruk dari langkah yang diproses sebelumnya.

Kompleksitas pencarian dari algoritma alpha-beta ini adalah $O(b^{d/2})$ dibandingkan dengan kompleksitas algoritma minimax biasa yaitu $O(b^d)$. Jadi, jika pada algoritma minimax diproses 400 simpul, maka dengan adanya algoritma alpha-beta, simpul yang diproses hanya 20 buah saja.



Gambar 2. Sebagian dari *game tree* untuk permainan *connect four*

2. METODE PENCARIAN SOLUSI

2.1 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 dan mencegah pemain-2 untuk memperoleh nilai seminimal mungkin

min adalah pemain-2 yang berusaha memperoleh nilai seminimal mungkin dan mencegah pemain-1 untuk memperoleh nilai semaksimal mungkin

2.2 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 dipotong).

Nilai alpha diinisialisasi dengan -∞ dan beta diinisialisasi dengan ∞. Hal ini untuk memastikan bahwa algoritma ini pasti mengembalikan suatu nilai.

Pseudocode untuk algoritma minimax yang ditambahkan *alpha beta pruning* ini yaitu :

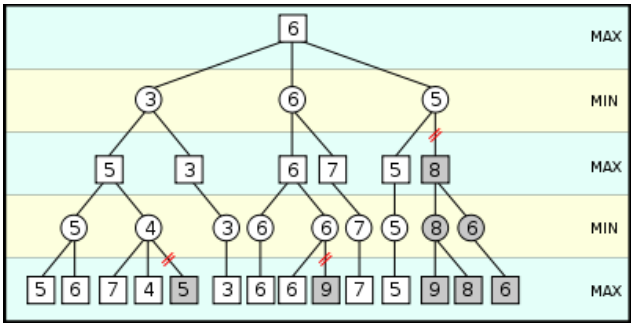
```
alpha-beta (pemain,simpul,alpha,beta)
    if (simpul daun)
        return pemenang

    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 di atas bahwa algoritma alpha beta merupakan algoritma rekursif dan karena dalam permainan kedua pemain secara bergantian menjalankan langkah yang dipilih, ketika pemain max yang sedang diproses, maka algoritma akan dipanggil untuk pemain min.

Fungsi alpha-beta ini dipanggil di suatu simpul untuk menentukan langkah terbaik yang harus dipilih pada simpul tersebut. Algoritma alpha-beta ini akan lebih cepat jika simpul-simpul anak telah terurut (terurut menurun untuk max dan terurut menaik untuk min).

Alpha-beta ini merupakan perbaikan dari fungsi minimax, sehingga akan menghasilkan nilai yang sama dengan nilai yang diperoleh oleh fungsi minimax biasa atau dengan kata lain algoritma alpha-beta ini memang digunakan untuk mempercepat algoritma minimax.

Algoritma alpha-beta ini merupakan algoritma yang berbasis *depth-first-search*, sehingga biasanya digunakan secara *iterative deepening*, sehingga jika algoritma ini di-*interrupt* sebelum selesai dieksekusi, langkah yang dipilih masih merupakan langkah yang cukup baik. Selain itu, dengan *iterative deepening*, pemrosesan akan lebih cepat karena jika terjadi pemotongan pada level atas, simpul tersebut tidak perlu lagi diperiksa hingga ke anak-anaknya, sehingga jumlah simpul yang diproses akan berkurang.



Gambar 3. Contoh *game tree* yang diproses dengan algoritma alpha-beta

3. APLIKASI PADA PERMAINAN *CONNECT FOUR*

3.1 Kompleksitas Permainan

Kompleksitas dari permainan *Connect Four* ini yaitu :

- Ukuran papan : 42 sel
- Kompleksitas status (jumlah status permainan yang dapat dicapai) = 10^{14}
- Kompleksitas *game tree* (jumlah simpul pada *game tree*) = 10^{21}
- Panjang permainan rata-rata = 36 langkah (= 36 level pada *game tree*)

Seperti terlihat dari kompleksitas permainan ini, Connect Four memiliki *game tree* yang cukup besar, sehingga tidak memungkinkan untuk memproses seluruh simpul pada game tree dari awal hingga akhir (meskipun menggunakan algoritma alpha-beta). Karena itu algoritma yang diterapkan pada pencarian *game tree* untuk permainan ini hanya akan memproses untuk beberapa level ke bawah.

3.2 Aplikasi Algoritma Minimax

Karena algoritma minimax yang diterapkan hanya memproses hingga beberapa level saja, diperlukan suatu fungsi evaluasi posisi yang menilai kondisi papan saat itu. Dalam makalah ini digunakan rentang nilai untuk permainan ini adalah antara -512 sampai 512, dengan nilai -512 menyatakan kemenangan pemain-1, nilai 512 menyatakan kemenangan pemain-2 dan nilai 0 menyatakan permainan berakhir seri.

Fungsi evaluasi ini dibuat berdasarkan posisi keping pada suatu "segmen" yaitu 4 sel pada papan yang memungkinkan pemain untuk membuat 4 keping berurutan, misalnya satu keping pemain dan tiga sel kosong. Nilai dari sebuah "segmen" yaitu :

- "0" jika tidak ada keping pada segmen

- "±1" jika ada satu keping pada "segmen"
- "±10" jika terdapat dua buah keping berwarna sama dalam "segmen"
- "±50" jika terdapat tiga buah keping berwarna sama dalam "segmen" tersebut

Tanda + (plus) dan – (minus) bergantung pada warna keping yang diperiksa. Untuk keping pemain-1 (max) nilai yang dikembalikan adalah positif, sedangkan untuk keping pemain-2 nilai yang dikembalikan adalah negatif.

Nilai yang dikembalikan dari fungsi evaluasi kondisi sebuah papan (simpul *game tree*) adalah hasil penjumlahan dari tiga buah bilangan, yaitu :

- Nilai netral atau 0
- Nilai bonus giliran, bernilai 16 (16 untuk pemain-1 dan -16 untuk pemain-2)
- Jumlah keseluruhan dari semua "segmen" yang mungkin

Hasil penjumlahan ini mungkin dikurangi untuk menjaga agar nilai evaluasi tetap dalam batas (antara -511 sampai 511).

Selanjutnya setelah evaluasi nilai dari kondisi papan, algoritma minimax dengan alpha-beta dijalankan seperti biasa untuk memperoleh langkah terbaik yang dapat diambil pada saat itu berdasarkan hasil evaluasi beberapa level ke depan.

Pseudocode untuk algoritma pencarian ini sama dengan pseudocode untuk algoritma alpha-beta, hanya ditambahkan fungsi evaluasi, dan nilai yang dikembalikan oleh suatu simpul daun adalah hasil evaluasi dari kondisi simpul tersebut :

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

    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)
```

Pseudocode untuk fungsi evaluasi kurang lebih sebagai berikut :

```
eval (pemain,simpul)
    sum = 0
    sum = sum + 16 // bonus giliran
    for each (segmen horisontal)
        hitung jumlah keping pemain pada segmen
        sum = sum + nilai jumlah keping
    for each (segmen vertikal)
        hitung jumlah keping pemain pada segmen
        sum = sum + nilai jumlah keping
    for each (segmen diagonal)
        hitung jumlah keping pemain pada segmen
        sum = sum + nilai jumlah keping
    if (pemain = min) sum = -sum
    return sum
```

4. KESIMPULAN

Kesimpulan yang dapat diperoleh dari pembahasan ini, yaitu :

1. Algoritma minimax dan alpha-beta dapat digunakan untuk mencari langkah terbaik yang dapat dipilih pada sebuah permainan *zero sum*.
2. Kompleksitas permainan papan pada umumnya besar, sehingga pencarian *game tree* terbatas hanya beberapa level saja.
3. Pencarian minimax dan alpha-beta pada permainan yang kompleks belum tentu menghasilkan solusi terbaik karena keterbatasan pemrosesan *game tree* (tidak dapat melihat sampai akhir).
4. Umumnya semakin jauh langkah ke depan yang dilihat (semakin banyak level yang diproses) langkah yang dipilih akan semakin baik.
5. Algoritma minimax dan alpha-beta dapat diterapkan dalam permainan *Connect Four*.

REFERENSI

[1] Ron Adams, Erik Ibsen, Chen Zhang, "A Connect Four Playing AI Agent: Algorithm and Creation Process", 2003.

[2] Yosen Lin, "Game Trees", 2003.

[3] Ronald L. Rivest, "Game Tree Searching by Min/Max Approximation", 1995

[4] Connect 4 Game, http://miss372.a-gi.net/533/progress_report.htm, tanggal akses : 2 Januari 2010

[5] Game tree, http://en.wikipedia.org/wiki/Game_tree, tanggal akses : 2 Januari 2010

[6] Minimax, http://en.wikipedia.org/wiki/Minimax, tanggal akses : 2 Januari 2010

[7] Alpha-beta pruning, http://en.wikipedia.org/wiki/Alpha-beta_pruning, tanggal akses : 2 Januari 2010

[8] Minimax Game Trees, http://www.aihorizon.com/essays/basiccs/trees/minimax.htm, tanggal akses : 2 Januari 2010