Two Agent Games: Alpha Beta Pruning

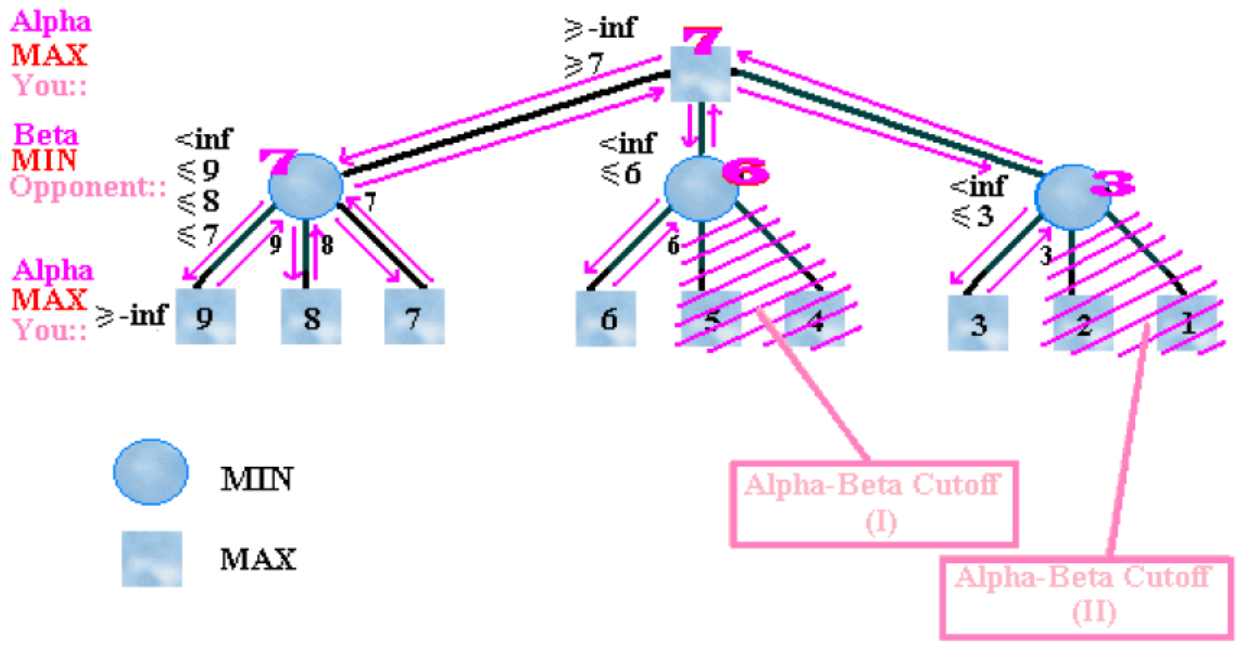
3.5 Alpha-Beta Pruning

Alpha-Beta Pruning adalah sebuah cara untuk mengurangi jumlah simpul yang dieksplorasi dalam algoritma MinMax. Dengan alpha-beta, waktu yang diperlukan dalam pencarian akan berkurang dengan cara membatasi waktu yang terbuang percuma pada saat mengevaluasi pohon permainan. Implementasi alpha-beta akan memberikan jalur terbaik dalam setiap kemungkinan permainan dalam pohon permainan yang terbentuk.

Dalam algoritma alpha-beta, urutan jalannya algoritma akan dimulai sama seperti algoritma minmax. Untuk simpul MIN, nilai yang dihitung dimuali dengan +infinity dan akan menurun seiring jalannya permaianan. Untuk simpul MAX, nilai akan dihitung mulai dengan –infinity dan akan menaik seiring berjalannya waktu.

Efisiensi prosedur alpha-beta akan sangat tergantung pada urutan simpul berikutnya dari sebuah simpul yang sedang dieksplorasi. Jika beruntung, maka sebuah simpul MIN akan dianggap sebagai simpul dengan urutan nilai dari rendah ke tinggi, dan simpul MAX dari tinggi ke rendah. Secara umum, dapat diperlihatkan bahwa dalam kondisi yang terbaik, alpha-beta akan membuka jumlah terminal nodes yang sama dengan MinMax dalam pohon permainan, dengan dua kali lipat tingkat kedalaman pohon.

Berikut adalah contoh algoritma alpha-beta:



Algoritma Alpha-Beta

Algoritma akan mendata dua nilai: alpha dan beta, yang akan merepresentasikan nilai minimum untuk MAX, dan nilai maksimum untuk MIN. Pada awalnya nilai alpha adalah –infinity, dan beta bernilai awal +infinity. Selama jalannya rekursi, perbedaan nilai alpha dan beta akan semakin mengecil.

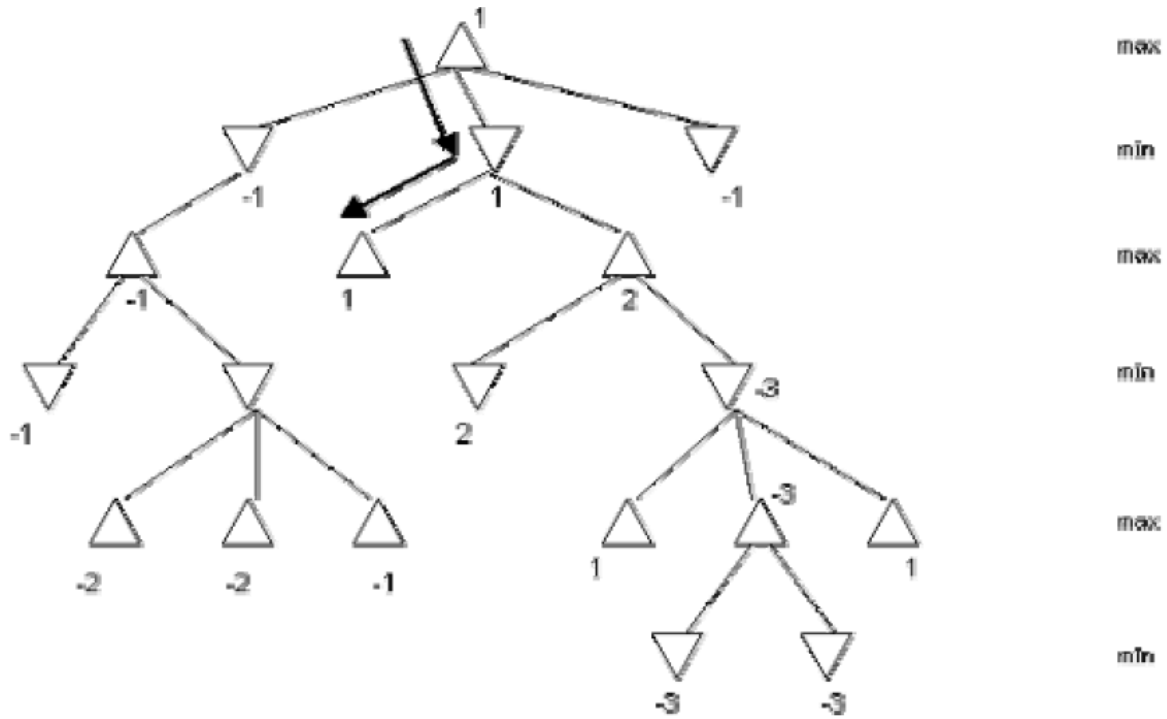
Ketika nilai beta menjadi lebih kecil dari alpha, berarti keadaan saat ini bukan merupakan hasil terbaik dari kedua pemain, dan dengan demikian tidak perlu dieksplorasi lebih jauh.

Pseudocode untuk algoritma alpha-beta adalah:

```
evaluate (node, alpha, beta)
    if node is a leaf
        return the heuristic value of node
    if node is a minimizing node
        for each child of node
            beta = min (beta, evaluate (child, alpha, beta))
            if beta <= alpha
                return beta
        return beta
    if node is a maximizing node
        for each child of node
            alpha = max (alpha, evaluate (child, alpha, beta))
            if beta <= alpha
                return alpha
        return alpha
```

3.6 Latihan

1. Asumsikan Anda dan rekan sedang memainkan sebuah permainan di atas papan. Sekarang adalah giliran Anda untuk bergerak, dan pohon di bawah memberikan situasi Anda. Nilai fungsi evaluasi pada simpul terminal (leaves), diberikan pada pohon. Perhatikan bahwa tidak semua simpul terminal berada pada level yang sama. Gunakan algoritma MinMax untuk menentukan langkah Anda. Gunakan pohon yang diberikan.



2. Gunakan pohon yang sama dengan nomor 1, sekarang lakukan algoritma alpha-beta, dan tentukan simpul mana saja yang tidak perlu diekspansi.

3. Ambil sebuah permaian TTT. Asumsikan bahwa X adalah pihak MAX. Gunakan fungsi utilitas untuk kemenangan X adalah 10, kekalahan -10, dan seri 0.

   a. Diberikan papan board1, X adalah yang akan melangkah berikutnya. Buatlah pohon permainan selengkapnya. Tentukan nilai utilitas dari setiap simpul terminal dan gunakan algoritma MinMax untuk menghitung langkah optimal bagi X.

   b. Diberikan papan board2, X adalah yang akan melangkah berikutnya, tunjukkan dalam pohon permainan sebuah pemotongan (cut off / pruning), untuk dua langkah permainan (satu untuk X, dan satu untuk O). Gunakan fungsi evaluasi sebagai berikut untuk setiap simpul:

      Eval(s) = 10X3(s) + 3X2(s) + X1(s) – (10O3(s) + 3O2(s) + O1(s))

      Dimana,
      Xn(s) adalah jumlah baris, kolom atau diagonal dalam state s, dengan X sejumlah n, dan tidak ada O, dan untuk On(s), adalah jumlah baris, kolom atau diagonal dalam state s, dengan O sejumlah n, dan tidak ada X.

      Gunakan algoritma MinMax untuk mengetahui langkah terbaik untuk X.

board1

X	O	X
O		O
	X	

board2

	X	
O		
X		O