Aplikasi *Exhaustive Search* dan Minimax dalam Pembuatan Algoritma AI Permainan Sekuensial

Fairuz Astra Pratama / 13514104
Program *Studi Informatika*
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
pratamafairuz@gmail.com

***Abstrak*—Algoritma Minimax adalah salah satu algoritma yang sering digunakan dalam pembuatan AI permainan sekuensial dimana pemain bermain secara bergiliran. Secara singkat, cara kerja algoritma ini adalah menganalisa semua kemungkinan pergerakan secara exhaustive dan memberi nilai skenario terburuk yang dapat terjadi atas pemilihan pergerakan tersebut.**

***Kata Kunci* —Minimax, Exhaustive Search, Permainan Sekuensial, AI**

I. PENDAHULUAN

Pada masa kecil dulu, kebanyakan dari kita mengenal permainan permainan seperti catur, *checkers*, dan Tic Tac Toe. Pada permainan permainan ini kedua pemain akan saling melakukan pergerakannya secara bergantian, saling merespon pergerakan lawan, dan mencoba untuk menang.

Konsep ini sering disebut disebut permainan sekuensial, dan ada beberapa contoh dunia nyata yang menggunakan konsep ini, memberikan perbandingan antara permainan kompetitif dengan skenario nyata.

Sebagai contoh, coba bandingkan permainan catur dengan dua perusahaan minyak yang saling bersaing. Pada catur setelah salah seorang pemain melakukan pergerakan, lawannya akan mengambil langkah balasan dengan mempertimbangkan pergerakan sebelumnya; pemain pertama pun akan memperhatikan hal ini, dan menggunakannya dlm melakukan pergerakan selanjutnya untuk mencoba membunuh pion raja lawan.

Pada perusahaan minyak, hal yang serupa pun terjadi; perusahaan satu akan memasang harga minyak baru, perusahaan dua melihat hal ini, dan akan mengubah harga mereka untuk bersaing dengan perusahaan satu yang juga akan membalas pergerakannya dan seterusnya.

Kedua perusahaan akan terus melakukan hal ini agar mereka dapat "menang" dalam "permainan" ini, yaitu dengan mendapatkan keuntungan setinggi mungkin, karenanya mereka akan terus berubah dan beradaptasi terhadap pergerakan pasar dan satu sama lain.

Setelah melihat kesamaan keduanya, dan mengingat telah dibuat AI dalam permainan sekuensial seperti catur yang dapat mengalahkan manusia, tentunya dapat dibuat semacam AI yang akan dapat membantu manusia dalam mengambil keputusan bisnis.

Tetapi sebelum membuat AI itu, perlu dipahami seperti apakah cara kerja AI pada permainan sekuensial itu ?, dan apakah cara kerjanya dapat digunakan di skenario dunia nyata, hal itulah yang akan dicoba dibahas di makalah ini, dan akan digunakan AI game Tic Tac Toe yang sederhana sebagai contoh utama.

II. *COMBINATORIAL GAME THEORY*

Pada pendahuluan, telah sedikit dibahas mengenai *The Game Theory* yang meneliti situasi kompetitif dan bagaimana para peserta akan bereaksi terhadap aksi satu sama lain. Namun pada pembuatan AI game sederhana seperti Tic Tac Toe, akan digunakan cabang dari teori tersebut yang bernamakan *Combinatorial Game Theory*

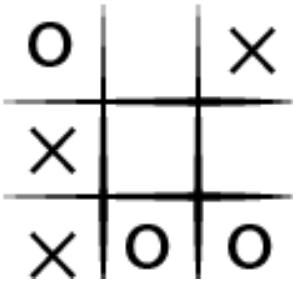
2.1 Perbedaan dengan *Game Theory*

Walaupun keduanya membahas hal yang relatif sama, perbedaan yang mencolok pada *Combinatorial Game Theory* jumlah informasi yang tersedia pada pemain. Pada Teori Game yang umum, terdapat informasi yang tidak diketahui suatu pemain (tetapi diketahui lawan) dan juga terdapat unsur peluang; membuat algoritma AI harus mempertimbangkan unsur ketidakpastian didalamnya.

Sedangkan pada CGT, pemain (atau AI) memiliki informasi yang sempurna tentang kondisi permainan; hal ini berarti kedua pemain mempunyai akses kepada semua event atau aksi yang diambil hingga saat ini. Permainan catur tergolong kategori ini karena kedua pemain dapat melihat semua isi papan, sedangkan pada permainan seperti kartu, isi kartu pada tangan lawan disembunyikan.

2.2 Konsep Dasar

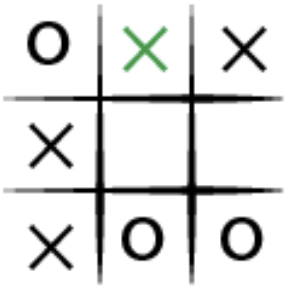
CTG memandang permainan sebagai sekumpulan alternatif pergerakan yang tersedia pada pemain. Sebagai contoh sederhana, perhatikan gambar papan Tic Tac Toe berikut. Untuk mempermudah, tiap kotak diidentifikasi dengan koordinat (x,y) seperti diagram kartesian. Dengan (1,1) merupakan kotak di pojok kiri bawah, dan (3,3) merupakan kotak di pojok kanan atas.



Gambar 2.2.1 Papan permainan Tic Tac Toe
[1]http://neverstopbuilding.com/minimax, akses 06/12/2015

Misal papan diatas adalah kondisi ditengah tengah permainan Tic Tac Toe dan sekarang adalah giliran pemain "X". Dia akan mempunyai 3 kemungkinan pergerakan yang dapat dia ambil, yaitu menaruh X pada kotak (2,3), (2,2), atau (3,2). Pergerakan manapun yang dia ambil akan merubah kondisi permainan.

Konsep CTG lainnya adalah setiap kondisi permainan baru yang diakibatkan aksi salah satu pemain dapat dipandang sebagai permainan baru yang terpisah. Kondisi game lama serta aksi pemain yang telah dilakukan tidak akan berpengaruh terhadap kondisi game saat ini. Selain itu, pengambilan keputusan pemain hanya dipengaruhi oleh kondisi permainan sekarang, bukan yang terdahulu.



Gambar 2.2.2 Papan baru jika pemain "X" memilih kotak (2,3)
[1]http://neverstopbuilding.com/minimax, akses 06/12/2015

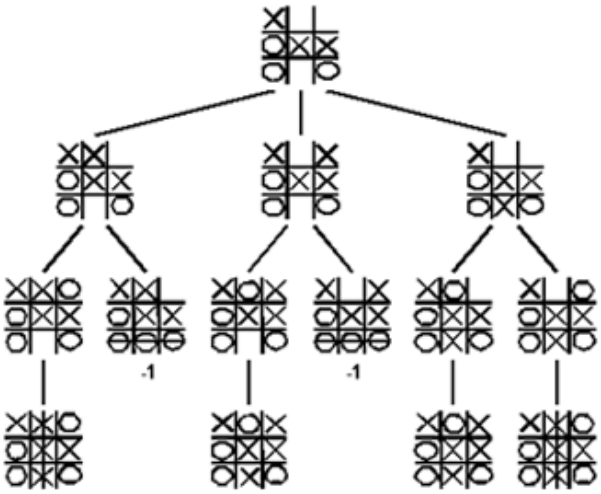
Sebagai contoh perhatikan gambar diatas. Saat pemain "X" melakukan aksi untuk meletakan tanda di kotak (2,3), kondisi game pun berubah ke seperti gambar. Pemain "O" (yang asumsikan saja merupakan AI) tidak perlu memperhatikan kondisi papan sebelumnya ataupun mempertanyakan kenapa kenapa pemain "X" tidak memilih kotak (2,2) dan memenangkan permainan. Dia hanya perlu informasi yang disediakan kondisi papan sekarang yang memberikannya dua pilihan pergerakan: (2,2), atau (3,2); lalu memilih satu pergerakan darinya.

2.3 *The Game Tree*

Dari pembahasan sebelumnya, diketahui bahwa kondisi permainan dapat dipandang secara independen dan dihubungkan oleh pergerakan pemain. Berdasarkan konsep ini, dapat dibentuk sebuah graf berarah yang menunjukan setiap kemungkinan kondisi permainan yang mungkin serta aksi yang menghubungkannya. Graf pohon inilah yang disebut sebagai *Game Tree*

Pada graf ini, setiap simpul merepresentasikan kondisi permainan, dan tiap sisi merepresentasikan pergerakan yang menyebabkan kondisi itu berubah dari simpul parrent-nya. Setiap simpul akan bercabang berdasarkan jumlah pergerakan yang dapat dilakukan pada kondisinya, dan setiap cabang akan berisi alternatif kondisi yang dapat dicapai.

Selain itu, setiap simpul daun merepresentasikan kondisi game akhir (pada Tic Tac Toe, antara semua kotak penuh atau terdapat 3 simbol sama yang bersisian). Sedangkan simpul dalam merepresentasikan kondisi permainan yang belum selesai.



Gambar 2.2.1 Contoh pohon permainan partial TIcTacToe
[2]https://www.cs.cmu.edu/~adamchik/15121/lectures/Game%20Trees/Game%20Trees.html, diakses pd 06/12/2015

Gambar diatas adalah bagian dari pohon permainan Tic Tac Toe, yang menunjukan semua kemungkinan pergerakan yg dapat terjadi dari kondisi awal yg terlihat di akar pohon. Pohon permainan Tic Tac Toe penuh, mengandung sekitar 255.000 buah simpul daun.

Untuk permainan sederhana seperti Tic Tac Toe AI dapat memiliki akses ke pohon permainan penuh yang akan digunakannya dalam menerka pergerakan terbaik selanjutnya, sementara untuk permainan kompleks seperti catur, pada normalnya hanya akan menggunakan sebagian dari pohon permainan yang dihasilkan.

Dari konsep ini AI dapat menggunakan metode yang dinamai Minimax dalam menelusuri / membangun pohon permainan yang akan membantunya memilih pergerkana terbaik yang dapat dilakukan pada tiap status permainan yang mungkin.

III. ALGORITMA *MINIMAX*

Minimax, atau dapat disebut sebagai Minmax adalah aturan yang dapat digunakan di statistika dan berbagai cabang pengetahuan lain untuk menentukan nilai terbaik dari suatu skenario terburuk.

3.1 Konsep Dasar

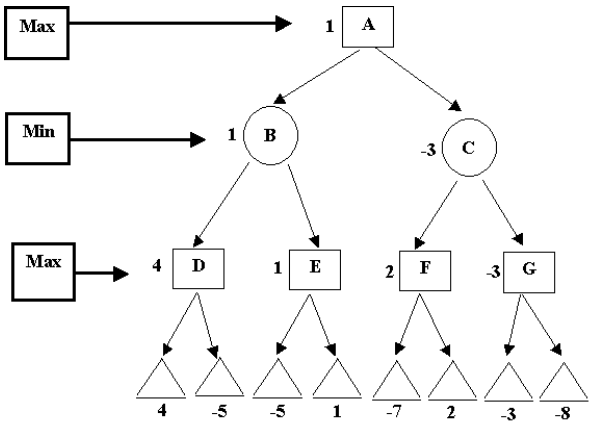
Pada permainan secara umumnya, tujuan kita adalah mendapatkan "poin" setinggi mungkin. Oleh karena itu, pada giliran kita, kita akan berusaha memilih gerakan yang dapat menghasilkan poin tertinggi. Nilai ini disebut dengan nilai Maximin.

Disisi lain, lawan kita, juga memiliki tujuan yang sama dengan kita. Dengan kata lain, pada giliran mereka, mereka akan berusaha memilih gerakan yang dapat menghasilkan poin yang terendah untuk kita. Nilai ini disebut dengan nilai Minimax.

Pada giliran pemain, akan tersedia sekelompok pilihan yang dapat diambilnya. Tujuan kita adalah memberi nilai ke tiap alternatif agar kita dapat memilih alternatif yang terbaik. Apabila pengambilan pilihan tertentu berakibat berakhirnya permainan, maka nilai pilihan itu dapat diketahui (misal pemberian poin 3 jika pemain menang, -3 jika lawan menang, dst)

Namun apabila tidak, maka giliran akan menjadi giliran lawan dengan kondisi permainan yang baru sesuai aksi pemain. Pada giliran lawan, maka hal yang dilakukan sama, berusaha memberi nilai dari alternatif pergerakan yang dapat diambil. Hanya saja, pada giliran lawan, mereka pasti akan berusaha mengambil nilai yang terendah bagi kita, kebalikan dari yang kita inginkan.

Pemprosesan pun akan terus terjadi secara rekursif dan bergantian, sampai pada akhirnya pasti tercapai kondisi dimana semua alternatif aksi akan berakhir pada berakhirnya permainan, dan dapat diketahui dengan mudah nilai pergerakannya.



Gambar 3.1.1 Contoh algoritma minimax, [3]https://wiki.scratch.mit.edu/wiki/Game_Tree, akses pd 06/12/2015

Pada gambar diatas simbol kotak melambangkan giliran pemain, lingkaran giliran lawan, dan segitiga adalah kondisi akhir permainan. Dimana angka disampingnya melambangkan nilai status tersebut.

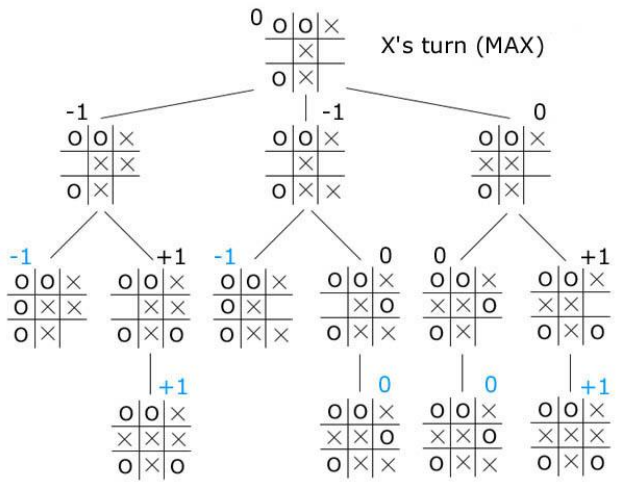
Dari kondisi ini, maka pemprosesan dapat dilakukan dari bawah pohon hingga ke akarnya, jika pada level tersebut merupakan giliran pemain, maka ambili nilai yang menuju ke alternatif dengan nilai yang terbesar. Sedangkan apabila merupakan giliran lawan, ambil nilai yang menuju ke alternatif dengan nilai yang terendah.

Sesampainya di akar, dapat diketahui nilai akhir dari setiap alternatif pergerakan jika pergerakan pemain lawan optimum dan pergerakan kita selanjutnya optimum juga. Dari informasi ini, dilakukanlah apa yang telah algoritma lakukan dari awal dan memilih nilai yang terbaik.

3.2 Aplikasi Minimax di Permainan Tic Tac Toe

Pada permainan Tic Tac Toe, penilaian status akhir akan jauh lebih sederhana, dikarenakan hanya ada 3 kemungkinan akhir: Pemain menang, kalah, atau seri. Perangkingan pun dapat dengan mudah dilakukan dengan memberi nilai yang terurut membesar. Ex: menang = 1, seri = 0, dan kalah = -1.

Pemprosesan secara umumnya dapat dilakukan secara serupa dengan algoritma Minimax secara umumnya; hanya saja untuk satu / dua pergerakan pertama dapat dilakukan secara acak, dikarenakan pergerakan awa kurang berpengaruh ke hasil akhir permainan, serta pemprosesan tile awal berarti membangun seluruh pohon permainan Tic Tac Toe (lebih dari 250000 node daun)



Gambar 3.2.1 Contoh algoritma minimax pada Tic Tac Toe [4]https://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html, akses pd 06/12/2015

Gambar diatas merupakan ilusrasi pemanggilan algo Minimax pada pertengahan permainan. Pemain memiliki 3 pilihan pergerakan, berikut penjabarannya:

- Jika menaruh "X" di 3,2; pada saat giliran musuh :
  - Jika menaruh "O" di 1,2; pemain kalah (-1)
  - Jika menaruh "O" di 3,1; pada saat giliran pemain
    - Jika menaruh "X" di 1,2; pemain menang (+1)

        - Pemain mengambil satu satunya pilihan di 1,2; bernilai +1
    - Jadi, jika menaruh “O” di 3,1; akan bernilai +1
    - Musuh mengambil yg paling merugikan di 1,2; bernilai -1
- Jadi, jika menaruh “X” di 3,2; akan bernilai -1

- Jika menaruh “X” di 3,1; pada saat giliran musuh :
    - Jika menaruh “O” di 1,2; pemain kalah (-1)
    - Jika menaruh “O” di 3,2; pada saat giliran pemain
        - Jika menaruh “X” di 1,2; keadaan seri (0)
        - Pemain mengambil satu satunya pilihan di 1,2; bernilai 0
    - Jadi, jika menaruh “O” di 3,1; akan bernilai 0
    - Musuh mengambil yg paling merugikan di 1,2; bernilai -1
- Jadi, jika menaruh “X” di 3,1; akan bernilai -1

- Jika menaruh “X” di 1,2; pada saat giliran musuh :
    - Jika menaruh “O” di 3,1; pada saat giliran pemain
        - Jika menaruh “X” di 3,2; pemain menang (+1)
        - Pemain mengambil satu satunya pilihan di 3,2; bernilai +1
    - Jadi, jika menaruh “O” di 3,1; akan bernilai +1

    - Jika menaruh “O” di 3,2; pada saat giliran pemain
        - Jika menaruh “X” di 3,1; keadaan seri (0)
        - Pemain mengambil satu satunya pilihan di 3,1; bernilai 0
    - Jadi, jika menaruh “O” di 3,2; akan bernilai 0
    - Musuh mengambil yg paling merugikan di 3,2; bernilai 0
- Jadi, jika menaruh “X” di 1,2; akan bernilai 0

- Pemain akan mengambil alternatif yang paling menguntungkan di 1,2; bernilai 0

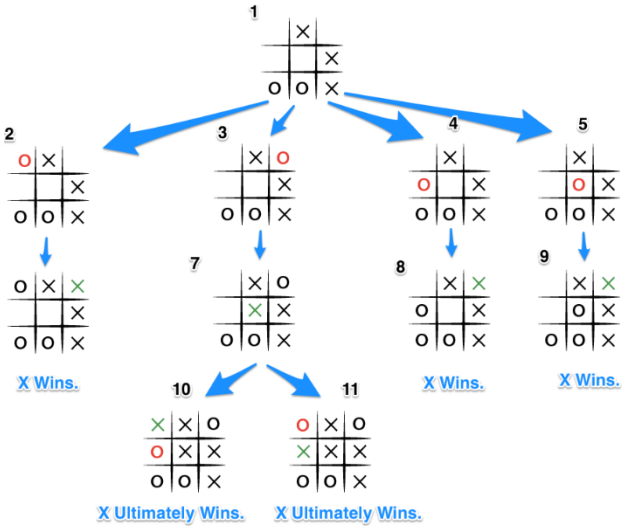
Setelah AI membangun pohon permainan dan nilai Minimax setiap simpul, maka untuk pergerakan berikutnya tidak perlu membuat pohon baru. Setelah mengambil pergerakan yang dirasa paling baik, AI hanya perlu menunggu gerakan yang dibuat lawan (dapat melakukan tindakan non-optimum).

Karena pada pohon permainan semua alternatif kejadian sudah ada, AI akan hanya perlu melihat nilai Minimax pada pohon yang telah ditentukan pada pemprosesan sebelumnya dan menggunakannya dalam memilih pergerakan berikutnya.

Sebagai contoh sederhana, apabila setelah menaruh “X” di 1,2; musuh menaruh “O” di 3,1; AI tidak perlu lagi menghitung nilai minimax kembali dari setiap alternatif aksi pada gilirannya, dia hanya cukup menelusuri pohon status permainan berdasarkan aksi lawan, dan melihat nilai minimax dari setiap alternatif aksi dari situ.

Dengan menggunakan cara ini, pembangunan pohon dapat dilakukan sekali saja pada tiap permainan atau bahkan disimpan dalam file eksternal yang dibaca saat permainan dimulai, sehingga AI hanya perlu menelusuri pohon sesuai alur permainan dan memilih tindakan terbaik berdasarkan nilai Minimax yg ada di pohon.

Namun terdapat kelemahan pada penggunaan algoritma Minimax untuk permainan Tic Tac Toe ini. Jika, entah bagaimana caranya, AI dihadapkan pada kondisi dimana gerakan apapun yang dia ambil, lawan akan selalu menang, walaupun terdapat cara untuk menunda kekalahan, AI belum tentu akan mengambil cara itu dikarenakan nilai setiap status sama (-1).



Gambar 3.2.2 Kondisi dimana pemain pasti kalah
[1]http://neverstopbuilding.com/minimax, akses 07/12/2015

Seperti yang dapat dilihat pada gambar diatas, tindakan apapun yang diambil AI, lawan tetap akan menang. Dikarenakan kebanyakan algoritma akan mengambil nilai terbaik pertama jika diberikan beberapa nilai yang sama, maka kemungkinan besar, AI akan mengambil alternatif ke-2, dan lawan akan menang pada giliran berikutnya.

Namun, apabila AI mengambil alternatif ke-3, lawan baru akan menang setelah 3 giliran lagi. Perilaku AI yang “berjuang” dan menunda kekalahan ini lebih menyerupai perilaku manusia. Untuk membuat AI bertingkah laku seperti ini, maka hanya perlu dilakukan modifikasi sedikit ke algoritma Minimax yang standar, untuk membuat pemain ideal yang juga berusaha memperlama permainan.

Agar tidak merubah nilai minimax yang dapat mempengaruhi perilaku mesin, maka sebaiknya untuk setiap simpul status, selain disimpan nilai Minimax, juga disimpan jumlah giliran antara status ini dan status yang menghasilkan nilai minimax tersebut. Besar nilai jumlah giliran ini akan 0 pada status final, dan untuk setiap rekursi pemilihan ke akar pohon akan ditambah satu.

Pada saat pemilihan, jika ditemukan semua pilihan berakhir dengan kekalahan atau seri, maka pilih alternatif dengan nilai giliran yang tertinggi. Disisi lain, jika semua pilihan berakhir dengan kemenangan, pilihlah alternatif dengan nilai giliran yang terendah.

3.3 Optimisasi Minimax

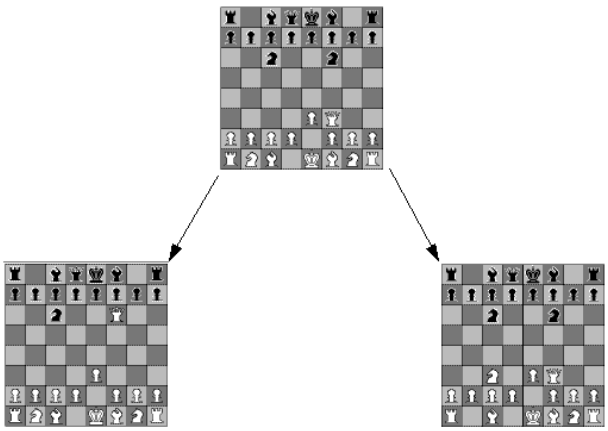
Sebelumnya telah dibahas konsep dasar dr algoritma Minimax untuk permainan sederhana Tic Tac Toe, namun algoritma ini sangat susah dimplementasikan untuk permainan lain seperti catur yang jauh lebih kompleks, ada beberapa perubahan yang dapat dilakukan.

- Penilaian sebelum kondisi akhir

Secara umum pemberian nilai minimax dilakukan di akhir permainan, namun hal ini berarti bahwa pohon permainan harus dievaluasi hingga akhir permianan sebelum dapat memberikan nilai kondisi sehingga memakan banyak waktu. Oleh karena itu, penilaian dapat dilakukan di kondisi pertengahan permainan berdasarkan heuristik permainan.

Pada permainan Othello pemberian nilai dapat dilakukan dengan menghitung jumlah pin pemain dikurangi jumlah pin lawan. Pada permainan catur, setiap pin dapat diberi nilai masing2, dimana pin seperti ratu bernilai lebih tinggi daripada pion dan raja bernilai sangat tinggi; pemberian nilai dapat berdasarkan jumlah nilai pion pemain dikurangi jumlah nilai pion lawan.

Pemberian nilai pun juga dapat didasarkan atas strategi permainan yang telah dibuat oleh ahli permainan tersebut. Sebagai contoh sederhana, kondisi permainan Othello dimana salah satu pin pemain berada di ujung papan akan mempunyai nilai tambahan besar karena pin tersebut tidak akan dapat diambil alih dan dapat mengambil alih jumlah besar pin lawan nantinya.



Gambar 3.3.1 Contoh Minimax pada permainan catur [9]http://www.ics.uci.edu/~pazzani/171.html, diakses pd 07/12/2015

- Tidak mencari hingga ke kondisi akhir

Pada permainan kompleks seperti catur, ada banyak sekali kombinasi kondisi papan permainan yang mungkin. Sebagai gambaran, misalkan secara rata rata permainan catur terjadi selama 40 giliran, dan pada tiap giliran terdapat 30 pergerakan yang dapat diambil. Hal ini berarti terdapat sekitar 30^40 simpul status yang harus dibangkitkan tiap permainan.

Oleh karena itu, untuk mempercepat algoritma, minimax sebaiknya hanya perlu membangkitkan semua status permainan di sekitar 5-6 giliran berikutnya saja, dan memberi nilai pada kondisi permainan berdasarkan heuristik yg dibahas sebelumnya.

Tentu saja, cara ini berarti bahwa AI akan melakukan tindakan yang sub-optimal, tapi cara kerja ini mirip dengan cara kerja manusia yang hanya memikirkan beberapa giliran kedepan. Selain itu, metode ini juga memberikan cara mengatur tingkat kesulitan AI, dimana AI yang lebih susah akan meningkatkan kedalaman pembangkitan pohon status (hingga puluhan giliran berikutnya) untuk meningkatkan performasi AI dengan mengorbankan kecepatan pemprosesan.

- Menggunakan *Alpha Beta Pruning*

Pada dasarnya *Alpha Beta Pruning* adalah cara pembangkitan pohon status yang lebih efisien dengan menerapkan prinsip *Branch and Bound*. Cara ini akan "memangkas" sisi (alternatif aksi) yang dirasa tidak penting karena berapapun nilai yang dihasilkan tidak akan merubah hasil pencarian.

Sebagai contoh sederhana, misalkan pada giliran kita terdapat sepasang alternatif aksi (diberi nama alternatif A dan B).. Setelah ditelusuri ternyata alternatif A akan menghasilkan nilai minmax 5. Karenanya, alternatif B harus menghasilkan nilai minimax yang lebih besar dari 5 agar tetap relevan (dapat merubah hasil pencarian).

Setelah dijalankan alternatif kedua itu, ternyata kondisi permainan berubah ke giliran lawan dengan sepasang alternatif aksi juga. Setelah ditelusuri, alternatif pertama menghasilkan nilai minimax 2. Karena ini giliran lawan, (pasti memilih nilai minimum), maka dapat disimpulkan bahwa nilai minimax alternatif B akan bernilai kurang dari sama dengan 2.

Kita masih belum tahu berapa nilai minimax yang dihasilkan alternatif dua tersebut, namun informasi ini tidak relevan. Untuk menghasilkan nilai yang relevan, aksi ini harus menghasilkan nilai yang cukup kecil sehingga dipilih oleh lawan (lebih kecil dari alternatif pertama yang bernilai 2), dan cukup besar sehingga dipilih oleh pemain (lebih besar dari alternatif A yang bernilai 5). Karena tidak mungkin, maka alternatif kedua giliran lawan dapat dilewatkan, dan alternatif B diberi nilai minimax sebesar 2.

Detail lebih lanjut mengenai cara terbaik penggunaan konsep ini tidak akan dibahas di makalah ini, namun penjelasan yang cukup baik tersedia di http://will.thimbleby.net/algorithms/doku.php?id=minimax_search_with_alpha-beta_pruning.

IV. KESIMPULAN

Teori Minimax dapat digunakan untuk mencari tahu nilai terbaik yg dapat dilakukan dalam kondisi terburuk, dan merupakan konsep yang sangat penting dalam pembuatan AI dalam game sekuensial sekaligus landasan bagi pembuatan algoritma pengambilan keputusan.

Hanya saja algoritma ini mempunyai kelemahan berupa lambatnya pemprosesan dikarenakan pada realisasinya menggunakan algoritma *Exhaustive Search* yang membangkitkan semua kemungkinan pergerakan sehingga diperlukan modifikasi besar terhadap algoritma jika ingin digunakan di kasus yang sangat kompleks seperti pengambilan keputusan bisnis.

REFERENSI

[1]http://neverstopbuilding.com/minimax, akses 06/12/2015

[2]https://www.cs.cmu.edu/~adamchik/15121/lectures/Game%20Trees/Game%20Trees.html, diakses pd 06/12/2015

[3]https://wiki.scratch.mit.edu/wiki/Game_Tree, akses pd 06/12/2015

[4]https://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html, akses pd 06/12/2015

[5]http://gatton.uky.edu/faculty/sandford/401_f12/sequential.pdf, diakses pada 05/05/2016

[6]http://kwanghui.com/mecon/value/Segment%205_2.htm, diakses pada 05/12/2015

[7]http://www.helsinki.fi/~kulikov/matem/BeckCombinatorialGames.pdf, diakses pada 05/12/2015

[8]https://wiki.scratch.mit.edu/wiki/Game_Tree, akses pd 05/12/2015

[9]http://www.ics.uci.edu/~pazzani/171.html, diakses pd 07/12/2015

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, 7 Mei 2015

Fairuz Astra Pratama - 13514104