Menentukan Langkah Terbaik pada Permainan Catur dengan Menggunakan Algoritma Runut-Balik

Giant Andreas Tambunan - 13519127
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): giantandreas0311@gmail.com

***Abstrak*—Permainan catur merupakan permainan papan berbasis giliran yang bisa dikatakan kompleks yang dimainkan oleh dua orang. Catur merupakan salah satu permainan papan yang paling populer di dunia pada saat ini. Permainan catur merupakan sebuah permainan berbasis strategi abstrak yang pada setiap langkah pada gilirannya memiliki banyak sekali pilihan solusi yang dapat dilakukan. Pada makalah ini akan dibahas bagaimana pengimplementasian algoritma runut-balik pencarian pilihan langkah yang terbaik yang dapat diambil dari sekian banyak pilihan solusi.**

Keywords—catur; backtracking; minimax; alpha-beta, fungsi evaluasi.

I. Pendahuluan

Catur merupakan salah satu permainan papan yang paling populer di dunia pada saat ini. Permainan catur merupakan sebuah permainan berbasis strategi abstrak yang telah ada sejak ada ke-7. Kejuaraan atau perlombaan catur sendiri sudah ada sejak berabad abad lalu dan sudah diakui dunia pada sejak tahun 1886. Seiring berjalannya waktu, permainan catur mulai diakui dunia dan dijadikan salah satu cabang olahraga yang dilombakan dalam perlombaan olahraga.

Dalam permainan catur, setiap pemain memiliki 16 buah catur yang masing masing jenisnya memiliki pergerakan yang unik. Buah-Buah catur tersebut terdiri dari Raja, Mentri/Ratu, Gajah, Benteng dan Bidak/Pion. Buah catur tersebut disusun di atas papan catur yang terdiri dari 64 kotak dengan posisi tertentu. Tujuan dari permainan catur adalah *checkmate* yaitu dengan 'menangkap' Raja pemain lawan. Dengan memperhitungkan banyaknya buah catur, jenis buah catur, jenis pergerakan buah catur, dan kemungkinan posisi, tentunya untuk memainkan satu langkah saja memiliki banyak sekali kemungkinan kejadian. Jika dihitung secara matematis, untuk satu langkah saja (putih dan hitam menjalankan buah caturnya satu kali) akan menghasilkan puluhan bahkan ribuan kemungkinan kejadian yang berbeda.

Pemilihan langkah yang paling optimal dari sekian banyak kemungkinan kejadian tentunya bukan suatu persoalan yang mudah. Dalam hal ini, makalah ini akan membahas konsep pemilihan langkah yang paling optimal dalam permainan catur dengan menggunakan algoritma runut-balik (backtracking).

Gambar 1. Catur.

Sumber : https://en.wikipedia.org/wiki/Chess

II. Landasan Teori

Untuk dapat memahami bagaimana konsep pemilihan langkah terbaik pada permainan catur dengan menggunakan algoritma backtracking dibutuhkan beberapa pemahaman pada beberapa bidang berikut.

A. Catur

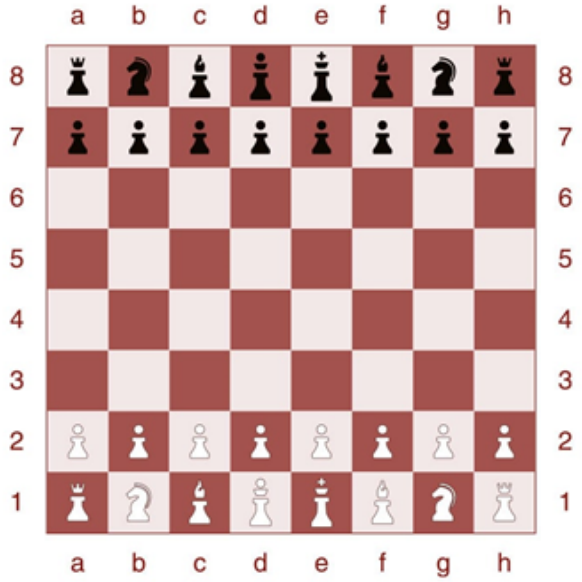
Catur adalah permainan papan strategi yang dimainkan oleh dua orang pada sebuah papan kotak-kotak yang terdiri dari 64 (8x8) kotak yang dibagi sama rata menjadi kelompok yang berwarna hitam dan putih. Permainan ini sangatlah populer dan dimainkan oleh jutaan orang di seluruh dunia. Catur sudah ada sejak abad ke-7, bernama chaturanga dan diyakini berasal dari India. Chaturanga juga diperkirakan merupakan asal muasal permainan papan strategi yang ada di Asia Timur seperti shogi (catur Jepang), xiangqi (catur Cina) dan janggi (catur Korea).

Setiap pemain pada pemain pada permainan catur memiliki buah-buah catur yang berjumlah 16 buah untuk setiap pemain. Buah-Buah catur tersebut terdiri dari satu buah raja, satu buah menteri/ratu, dua buah gajah, dua buah gajah, dua buah benteng dan delapan buah bidak/pion. Setiap jenis dari buah catur tersebut memiliki cara bergerak yang unik dan berbeda-beda. Setiap jenis buah catur tersebut juga memiliki nilai/value yang berbeda-beda juga, yang memiliki nilai paling

tinggi adalah menteri dan yang memiliki nilai paling rendah adalah pion.

Permainan catur dapat dimenangkan dengan skakmat (*checkmate*) raja pemain lawan dimana raja pemain lawan 'tertangkap' dan berada dalam keadaan yang terancam (diskak) dan tidak ditemukan lagi langkah yang dapat melepaskan raja dari ancaman tersebut. Seorang pemain juga dapat dinyatakan menang jika lawan main pemain tersebut menyerah atau kehabisan waktu jika permainan catur tersebut memiliki batasan waktu. Sebuah permainan catur juga dapat berakhir remis atau bisa disebut juga seri. Sebuah permainan catur dapat berakhir remis jika salah satu pemain tidak lagi bisa melakukan langkah (dengan keadaan raja tidak terancam/diskak), pengulangan langkah yang sama berulang kali, persetujuan antara kedua pemain, ataupun kedua pemain tidak lagi memiliki buah catur yang memungkinkan terjadinya skakmat.

Permainan catur dimainkan pada sebuah papan yang terdiri dari 8x8 kotak yang dibagi-bagi menjadi 2 kelompok yaitu yang berwarna putih dan berwarna hitam. Setiap kolom dan baris dari papan catur diberi nama. Untuk baris dari setiap kolom diberi nama 1-8 sesuai urutannya (dalam perspektif pemain yang memegang buah catur putih) dari bawah ke atas dan untuk kolom diberi nama a-h dari kiri ke kanan sesuai urutannya (dalam perspektif pemain putih). Penamaan ini dilakukan untuk memudahkan pemanggilan posisi buah catur dan penulisan/dokumentasi pada sebuah permainan catur. Misalnya untuk memanggil kotak yang ada pada kolom ketiga dan baris keempat (dari sudut pandang pemain putih) dapat dengan mudah dituliskan menjadi e4. Pada awal permainan catur, setiap buah catur yang dimiliki pemain disusun dengan aturan tertentu menurut jenis dan warna nya.



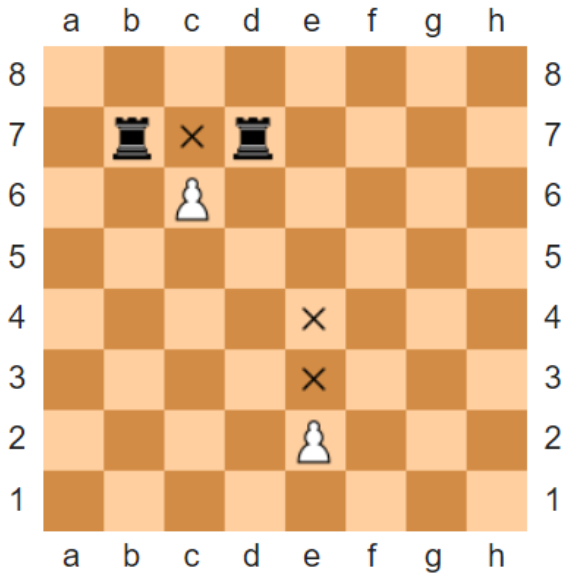
Gambar 2.Posisi Awal Buah Catur pada papan catur

Sumber : https://www.regencychess.co.uk/how_to_set_up_a_chessboard.html

Langkah dalam permainan catur dilakukan secara bergantian dan tidak boleh melewatkan giliran. Pada satu giliran, pemain hanya dapat menggerakkan satu buah catur yang dimilikinya sesuai aturan jenis catur yang digerakkan, kecuali pemain melakukan *castling* dimana pemain langsung menggerakkan raja dan kastel yang dimilikinya. Pemain dapat memindahkan buah caturnya ke kotak yang kosong maupun sudah ditempati oleh buah catur musuh. Jika pemain menggerakkan buah caturnya ke kotak yang sudah ditempati buah catur musuh berarti buah catur tersebut 'ditangkap' dan dieliminasi dari papan catur.

Setiap jenis buah catur pada permainan catur memiliki cara bergerak/berpindah yang berbeda beda. Untuk masing masing jenis buah catur tersebut dijelaskan sebagai berikut:

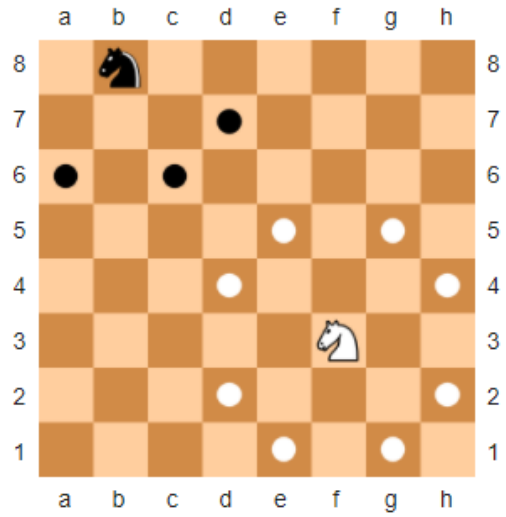
- Sebuah bidak/pion hanya dapat melangkah satu kotak ke arah depan, dan dapat juga dua kotak kearah depan pada saat pertama pertama kali bidak tersebut digerakkan dan tidak ada buah catur lain yang menghalangi. Sebuah bidak juga dapat memakan buah catur pemain lawan dengan cara bergerak diagonal satu langkah ke depan sebelah kanan atau depan sebelah kiri. Bidak juga dapat memakan bidak lawan dengan aturan khusus yang disebut *en passant.*



Gambar 3. Cara Bergerak Bidak/Pion

Sumber : https://en.wikipedia.org/wiki/Chess

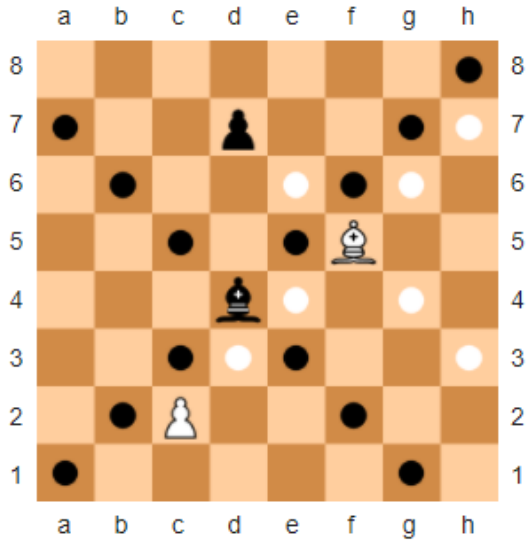
- Kuda merupakan satu-satunya buah catur yang dapat melompati buah catur lainnya. Kuda memiliki cara melangkah yang unik yaitu bergerak sejauh dua kotak secara vertikal dan satu kotak secara horizontal atau dua kotak secara horizontal dan satu kotak secara vertikal.



Gambar 4. Cara Bergerak Kuda

Sumber : https://en.wikipedia.org/wiki/Knight_(chess)

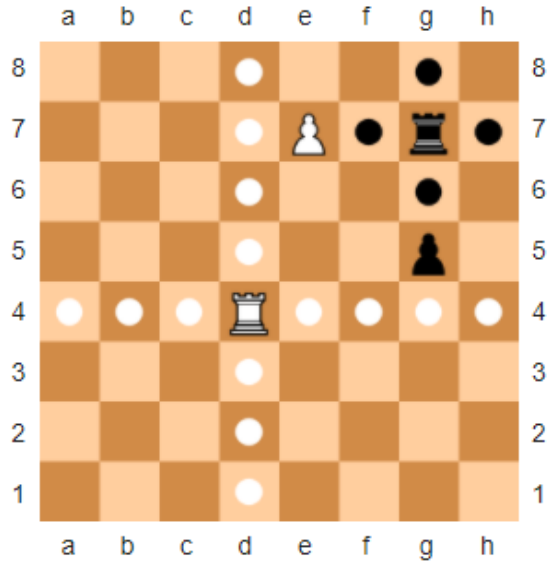
- Sebuah gajah dapat bergerak bebas tanpa batasan jarak secara diagonal.



Gambar 5. Cara Bergerak Gajah

Sumber : https://en.wikipedia.org/wiki/Bishop_(chess)

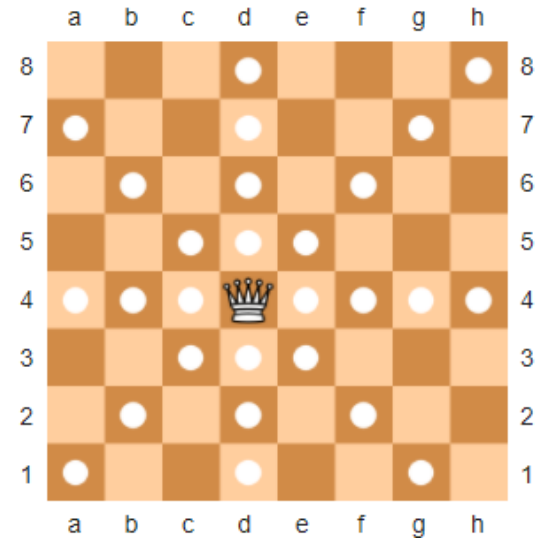
- Sebuah benteng/kastil dapat bebas tanpa batasan jarak secara vertikal ataupun horizontal



Gambar 6. Cara Bergerak Benteng/Kastil

Sumber : https://en.wikipedia.org/wiki/Rook_(chess)

- Sebuah ratu/menteri dapat bergerak secara bebas tanpa batasan jarak mengikuti garis lurus secara vertikal, horizontal maupun diagonal



Gambar 7. Cara Bergerak Ratu/Menteri

Sumber : https://en.wikipedia.org/wiki/Queen_(chess)

- Raja dapat bergerak satu kotak secara vertikal, horizontal, maupun diagonal dengan syarat kotak tempat tujuan pergerakan tidak dalam zona yang sedang dikuasai buah catur musuh. Jika raja dalam keadaan diskak maka pemain tidak dapat menjalankan buah caturnya yang tidak membuat raja menjadi dalam keadaan tidak diskak. Raja juga dapat melakukan *castling* atau rokade, raja akan berpindah 2 atau 3 kotak ke arah kiri posisi awal raja dan benteng akan berpindah ke arah berlawanan (kanan atau kiri) dari arah pergerakan raja. Rokade atau castling hanya bisa dilakukan jika raja dan benteng

belum pernah bergerak sebelumnya, tidak ada buah catur di antara raja dan benteng, dan raja tidak dalam keadaan terancam (di skak).

Gambar 8. Cara Bergerak Raja

Sumber : https://en.wikipedia.org/wiki/King_(chess)

B. Algoritma Runut-Balik (Backtracking)

Runut-Balik atau Backtracking dapat diartikan dalam dua hal. Yang pertama backtracking dapat diartikan sebagai salah satu fase atau proses yang terjadi dalam algoritma *Depth First Search.* Yang kedua, Backtracking dapat diartikan menjadi salah satu metode pemecahan masalah yang mangkus, terstruktur dan sistematis, baik dalam memecahkan permasalahan optimasi maupun non optimasi. Dalam makalah ini, backtracking yang dimaksud adalah backtracking dengan artian yang kedua.

Algoritma runut-balik atau backtracking merupakan perbaikan dari algoritma *Exhaustive Search*. Tidak seperti *exhaustive search yang* yang mengeksplorasi semua pilihan kemungkinan solusi, backtracking hanya mengeksplorasi pilihan yang mengarah ke arah solusi dan pilihan yang tidak mengarah ke solusi tidak lagi dipertimbangkan. Dalam hal ini algoritma backtracking akan memangkas (*pruning*) semua simpul pilihan yang tidak mengarah ke simpul solusi. Backtracking pertama kali diperkenalkan oleh D. H. Lehmer tahun 1950, dan kemudian disaikan dalam uraian umum oleh R. J. Walker, Golomb, dan Baumert.

Dalam menyelesaikan suatu persoalan dengan mangkus, backtracking atau runut-balik memiliki beberapa properti umum yang sering digunakan.

1) Solusi Persoalan

Solusi persoalan permasalahan dinyatakan dalam sebuah vektor yang berisi n-tuple

$X = (x1, x2, ..., xn)$

$xi \in Si$

dan umumnya

$S1 = S2 = ... = Sn$

2) Fungsi Pembangkit nilai xk

Dinyatakan dengan predikat T()

$T(x[1], x[2], ..., x[k-1])$

dinyatakan untuk membangkitkan nilai untuk xk, dimana xk merupakan komponen solusi

3) Fungsi Pembatas (bounding function)

Dinyatakan sebagai predikat

$B(x1, x2, ..., xk)$

yang bernilai *true* jika mengarah ke solusi dan bernilai *false* jika tidak mengarah ke solusi. Mengarah ke solusi berarti tidak melanggar pembatas (*constraint*). Jika bernilai true, maka solusi akan dilanjutkan sedangkan jika false solusi akan dibuang dan tidak dilanjutkan lagi.

III. Pembahasan dan Analisis

Sebelum dapat menggunakan algoritma runut-balik untuk menemukan langkah terbaik dalam permainan catur, kita harus terlebih dahulu mendefinisikan properti-properti yang diperlukan dalam pelaksanaan algoritma backtracking atau runut-balik. Untuk setiap state di dalam game harus diubah menjadi angka sehingga dapat dinilai oleh fungsi pembatas.

A. Fungsi Evaluasi dan Inisiasi

Untuk mengubah *state game* menjadi angka perlu didefinisikan terlebih dahulu fungsi evaluasi. Fungsi evaluasi akan mengevaluasi data *state game* yang ada di atas papan catur secara keseluruhan dan mengubah data data tersebut menjadi angka numerik sehingga dapat dikenali, dibandingkan dan dinilai oleh fungsi-fungsi lainnya dalam backtracking.

Fungsi evaluasi untuk catur dibuat untuk dapat menimbang keseimbangan materi (buah catur), istilah posisi, keselamatan raja, banyaknya buah catur tersisa, mobilitas, control, keselamatan raja, pusat kendali, struktur bidak, tropisme raja, dan sebagainya menjadi suatu data angka agar dapat dikenali oleh komputer.

Secara umum, fungsi evaluasi tentunya dapat berbentuk:

stateScore = c1 * materi + c2 * mobilitas
+ c3 * keselamatan raja + c4 * pusat kendali
+ c5 * struktur bidak + c6 * tropisme raja

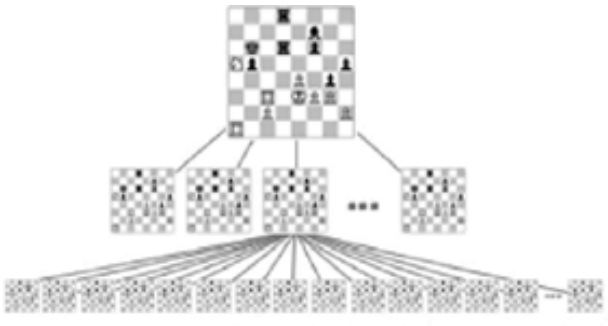
dengan c1, c2, c3,...dst merupakan konstanta yang dapat diubah-ubah untuk menyesuaikan keakuratan dari fungsi tersebut. Nilai materi bisa diperoleh dengan memberi nilai satuan kepada masing-masing jenis buah catur. Untuk yang konvensional adalah : Ratu = 9, Benteng = 5, Gajah dan Kuda = 3 dan pion = 1. Sedangkan raja diberi nilai besar yang sewenang wenang dan biasanya lebih besar dari jumlah nilai buah catur lainnya. Selain itu rasio jumlah material yang dimiliki oleh kedua pemain juga dipertimbangkan. Skor mobilitas adalah jumlah gerakan legal yang tersedia untuk

pemain, atau secara bergantian jumlah dari jumlah ruang yang diserang atau dipertahankan oleh setiap bidak, termasuk ruang yang ditempati oleh bidak lawan atau teman. Skor keselamatan raja adalah beberapa bonus penalty yang dinilai untuk lokasi raja dan konfigurasi bidak yang berada di dekat raja baik itu bidak lawan ataupun bidak milik sendiri. Skor pusat kendali diperhitungkan melalui konfigurasi dari beberapa buah catur yang menempati ruang tengah papan catur. Struktur bidak adalah seperangkat hukuman dan bonus untuk berbagai kekuatan dan kelemahan dalam struktur bidak, seperti penalti untuk bidak ganda dan bidak terisolasi. Tropisme raja adalah bonus untuk kedekatan (atau penalti jarak) bidak tertentu, terutama ratu dan ksatria, dengan raja lawan. Untuk perhitungan perhitungan lain dan perhitungan yang lebih mendetail mengenai fungsi evaluasi tidak dibahas lebih lanjut pada makalah ini.

Tentunya fungsi evaluasi tersebut tentunya tidak selalu merepresentasikan nilai sebenarnya yang terjadi di atas papan catur mengingat banyak sekali pertimbangan dan strategi yang mungkin dilakukan dalam permainan catur. Fungsi evaluasi tersebut masih dapat di ubah-ubah agar dapat menghasilkan nilai *state game* dengan lebih akurat. Namun hal itu tentunya membutuhkan perhitungan dan komputasi yang lebih kompleks.

Nilai-nilai dari *state game* tersebut kemudian akan dijadikan nilai dari setiap node pada tree yang ada pada algoritma runut-balik atau backtracking. Setiap node akan merepresentasikan satu *state game* pada satu waktu, yang berarti jika pemain melangkahkan buah caturnya berarti *state game* pasti berubah. Dari hal tersebut kita dapat menarik suatu kesimpulan bahwa pada kasus permainan catur, kita membutuhkan fungsi yang membuat tree awal yang nanti nya akan diaplikasikan algoritma backtracking pada tree tersebut. Dalam hal ini, tree dibangun dari kumpulan node-node *state game* baru yang dibangun dari setiap kemungkinan langkah yang dapat dilakukan oleh pemain.

Setiap kali pemanggilan *state game* baru dilakukan evaluasi pada state tersebut untuk menentukan skor dari state tersebut. Pemanggilan evaluasi *state game* baru itu sendiri tidak dilakukan langsung saat sebuah langkah yang mungkin dilakukan, melainkan dipanggil di pada state yang ada pada daun (ujung) dari tree yang dibangun. Sedangkan kedalaman dari daun yang akan dicari tersebut haruslah kita batasi karena jika tidak dibatasi akan menimbulkan pencarian *game state* yang tidak terhingga. Node dari tree yang dibangun akan menyimpan state dari game itu, sedangkan lintasan yang dibangkitkan akan menyimpan langkah yang dilakukan pada saat itu, misalnya king e1 to e2.



Gambar 9. Pembuatan Tree Pilihan Solusi pada permainan Catur

Sumber : https://stanford.edu/~cpiech/cs221/apps/deepBlue.html

Setelah tree dengan kedalaman tertentu tersebut berhasil dibangun, barulah dilakukan pencarian solusi yang paling optimal. Berbeda dengan algoritma pencarian solusi backtracking pada umumnya yang mencari solusi dari akar menuju daun, algoritma pencarian backtracking pada tree yang telah dibangun dimulai dari daun menuju akar. Pembangkitan juga mengikuti aturan yang berbeda karena pada permainan catur terdiri dua pemain sehingga perhitungan yang dilakukan harus memperhitungkan perspektif dari kedua pemain. Jenis dari algoritma backtracking yang unik ini disebut dengan *minimax*.

B. Minimax dan Teori Game

Minimax merupakan algoritma backtracking yang digunakan untuk mencari pilihan keputusan yang melibatkan lebih dari satu pemain dengan mengamsumsikan pemain lain juga mengambil pilihan yang paling optimal. Algoritma minimax banyak digunakan pada permainan yang berbasis giliran dan lebih dari 1 pemain, dan juga banyak diimplementasikan pada Artificial intelligence (AI) pada permainan tersebut seperti tic-tac-toe, go, checker, othello, catur dan lain sebagainya.

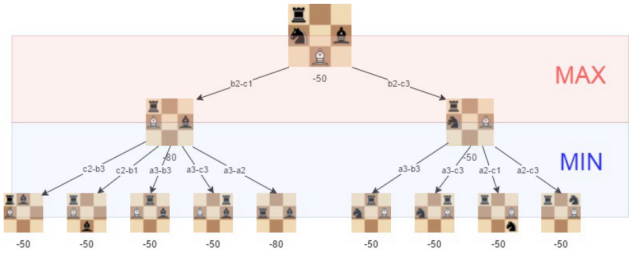
```
function minimax(node, depth, maximizingPlayer)
    if (depth = 0 or node is a terminal node) then
        return the heuristic value of node
    if (maximizingPlayer) then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player = FALSE*)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Gambar 10. Pseudocode minimax algorithm

Secara sederhana, minimax algorithm dapat diartikan sebagai suatu algoritma pencarian lintasan menuju nilai terbaik yang ada pada suatu decision tree (pohon keputusan) sampai dengan kedalaman (tinggi) tertentu. Pencarian tersebut dilakukan dengan pola tertentu dan nilai-nilai yang di maksud adalah state game (posisi dan kedudukan/dominasi) yang sudah dikalkulasi menjadi angka dengan cara tertentu (tergantung permainan yang sedang ditinjau). Dalam teori permainan, minimax algorithm digunakan untuk memutuskan pengambilan keputusan untuk mendapat keputusan yang paling menguntungkan dengan banyak pertimnbangan. Dalam

permainan catur sendiri, pertimbanganpertimbangan tersebut dapat berupa posisi, nilai/banyak buah catur tersisa, state game dsb. Pertimbangan-pertimbangan tersebut kemudian akan diubah menjadi suatu nilai numerik yang dapat dibandingkan dengan suatu fungsi evaluasi. Fungsi evaluasi dalam catur sendiri terdiri dari keseimbangan materi yang mendominasi evaluasi, ditambah satu set istilah posisi yang biasanya berjumlah tidak lebih dari nilai bidak, meskipun di beberapa posisi istilah posisi bisa menjadi jauh lebih besar, seperti saat skakmat sudah dekat.

Algoritma minimax secara rekursif akan membangun tree yang berisi dari semua *state game* baru yang dapat dibangkitkan. Tree yang dibangun berdasarkan kedalaman yang ditentukan dan kemudian semua state game pada tree di evaluasi nilainya. Algoritma minimax kemudian mengevaluasi setiap daun dan mengambil nilai yang tertinggi terendah pada daun daun tersebut dan mengembalikannya sebagai nilai dari state game yang ada diatasnya. Minimax mengembalikan nilai tertinggi jika pemain sedang melakukan langkah pada saat itu, dan mengembalikan nilai terendah untuk memperkirakan pilihan terbaik yang mungkin bisa dilakukan oleh lawan main.



Gambar 11. Ilustrasi dari minimax mengambil keputusan melangkah dengan kedalaman 2

Sumber : https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/

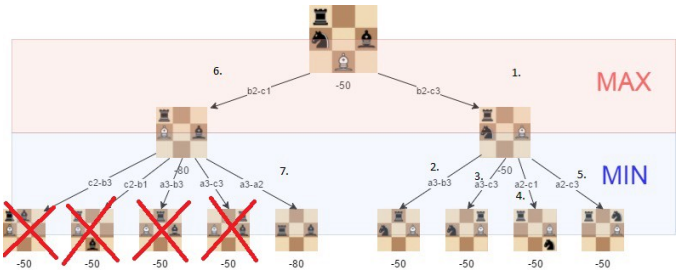
Pada daun yang ada pada kedalaman 2, state game dihitung dan di assign sebagai nilai dari node (daun) tersebut. Node-node pada kedalaman 1 ke kedalaman 2 merupakan kemungkinan langkah yang dapat diambil oleh lawan main, sehingga dengan minimax algorithm nilai node pada kedalaman 1 di assign dari nilai daun hasil percabangan tersebut yang memiliki nilai terendah. Sedangkan dari node akar ke node dengan kedalaman 1 merupakan langkah yang harus kita ambil sehingga dengan minimax diassign nilai tertinggi dari pilihan yang kita punya. Sehingga didapat solusi dengan nilai -50. Dan dari tree hasil minimax diatas, kita mendapati bahwa langkah yang paling optimal saat ini adalah e2-e3.

Dari ilustrasi di atas kita bisa lihat dengan catur dengan banyak kotak 3x3 ditemukan ada 11 kotak yang perlu di cek, bisa dibayangkan bahwa metode pemilihan keputusan dalam permainan catur dengan menggunakan minimax masih sangat memerlukan waktu dan komputasi yang banyak. Setiap kali pemanggilan minimax akan mengeluarkan sejumlah papan catur dengan state game yang berbeda beda, sehingga kedalaman pemanggilan sangat terbatas agar tidak terlalu banyak state game yang perlu dihitung. Namun kedalaman yang besar diperlukan untuk menemukan pilihan langkah yang lebih baik dan optimal. Untuk itu, pada setiap node akan dilakukan pembatasan (pemangkasan) dengan menggunakan fungsi pembatas yang disebut *alpha-beta pruning* sehingga algoritma backtracking ini bisa berjalan dengan lebih mangkus.

C. *Alpha-Beta Pruning*

Alpha-beta pruning merupakan suatu metode pembatasan (fungsi pembatas) yang dapat mengoptimasi algoritma minimax. Dengan pembatasan ini, maka tidak semua state game perlu dievaluasi yang membuat algoritma bisa berjalan dengan lebih mangkus. Akibatnya pencarian solusi yang dilakukan dapat dilakukan dengan lebih dalam. Biarpun begitu, tetap saja pembatasan perlu dilakukan karena banyaknya pilihan yang dapat muncul pada setiap pertambahan kedalaman yang membuat komputasi dan perhitungan semakin kompleks.

Alpha-beta pruning memaksa algoritma berhenti mengevaluasi pada node suatu pohon jika sudah ditemukan kasus yang lebih buruk dari pada node yang sedang diperiksa sekarang.



Gambar 12. Alpha-Beta Pruning

Sumber : : https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/

Pada gambar diatas, node yang ditandai dengan tanda silang merah tidak lagi di evaluasi node nya maupun node yang mungkin muncul dari node tersebut berdasarkan fungsi pembangkit. Hal tersebut dikarenakan sudah ditemukan nilai yang lebih buruk dari node node tersebut yaitu pada node 7 yang bernilai -80. Alpha-beta pruning memaksa algoritma untuk melewatkan node node yang disilang dan langsung meng-assign node 6 dengan nilai -80. Algoritma minimax dengan alpha-beta pruning dapat dipastikan menghasilkan solusi yang sama dengan minimax tanpa alpha beta pruning karena node yang disilang dapat dianggap merupakan node yang tidak menuju solusi.

IV. KESIMPULAN

Pemilihan langkah dalam permainan catur menggunakan algoritma backtracking akan selalu menghasilkan pilihan/solusi yang terbaik secara lokal selama fungsi evaluasi yang didefinisikan tepat dan sesuai. Solusi yang dihasilkan tidak selalu menghasilkan pemilihan langkah yang optimum secara global dikarenakan keterbatasan pencarian semua kemungkinan solusi yang ada akibat banyaknya hal yang perlu dipertimbangkan di dalam permainan catur. Banyaknya pilihan solusi tersebut perlu dibatasi karena keterbatasan omputasi dan perhitungan baik dengan bantuan komputer maupun manual.

Akibatnya solusi yang dihasilkan juga hanya akan optimum sampai batas yang telah ditentukan tersebut. Pemilihan langkah catur dengan menggunakan algoritma minimax dan alpha-beta pruning bersifat mangkus dan efisien sehingga banyak diimplementasikan pada chess AI.

Video Link at Youtube

-

Ucapan Terima Kasih

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya lah penulis dapat menyelesaikan makalah yang berjudul "Menentukan Langkah Terbaik pada Permainan Catur dengan Menggunakan Algoritma Runut-Balik" ini dengan baik. Penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya atas bimbingan dan juga pengajaran yang diberikan oleh Bapak Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. selaku dosen pengampu kelas saya pada mata kuliah IF2211 Strategi Algoritma selama semester II tahun akademik 2020-2021 yang memungkinkan saya dapat menyelesaikan makalah ini. Penulis juga ingin mengucapkan terima kasih kepada teman-teman yang selalu mensupport dalam penulisan makalah ini dan semua pihak yang telah mendukung dan memberi semangat kepada penulis dalam proses penulisan makalah ini. Penulis juga ingin meminta maaf untuk semua kesalahan selama proses pembuatan makalah ini dan selama proses pengajaran. Akhir kata, penulis berharap makalah ini dapat memberikan manfaat dan pengetahuan yang bertambah kepada pembacanya.

References

[1] FIDE Handbook: Law of Chess, diakses pada tanggal 10 Mei 2021 pada https://handbook.fide.com/chapter/E012018

[2] Munir, Rinaldi, Algoritma Runut-Balik. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm diakses pada tanggal 10 Mei 2021

[3] Minimax Algorithm in Game Theory, https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/ , diakses pada tanggal 10 Mei 2021

[4] Michael Maschler, Eilon Solan & Shmuel Zamir (2013). "Game Theory". Cambridge University Press.

[5] The Anatomy of a Chess AI, https://medium.com/the-innovation/the-anatomy-of-a-chess-ai-2087d0d565#:~:text=The%20minimax%20algorithm%20takes%20advantage,other%20tries%20to%20minimize%20it. , diakses pada tanggal 11 Mei 2021

[6] https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/ , diakses pada tanggal 11 Mei 2021

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, 10 Mei 2021

Giant Andreas Tambunan 13519127