Penerapan Pohon Pencarian Monte Carlo Tree Search dalam Pembuatan Chess Bot

Mohammad Nugraha Eka Prawira - 13522001[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522001@std.stei.itb.ac.id

***Abstrak*— Pada permainan catur online dapat digunakan alat/bot untuk membantu kita memenangkan permainan tersebut dengan menggunakan chess bot. Chess bot dapat membantu kita menentukan langkah terbaik yang dapat kita ambil sesuai dengan kondisi yang terjadi. Salah satu algoritma yang digunakan pada pembuatan chess bot adalah Monte Carlo Tree Search yang berbasis pada teori pohon pencarian dalam Matematika Diskrit. Pada makalah ini akan dibahas bagaimana penerapan teori pohon dan Monte Carlo Tree Search dalam pembuatan chess bot.**

***Kata kunci*—chess bot, pohon pencarian, pohon, Monte Carlo Tree Search, matematika diskrit.**

I. PENDAHULUAN

Catur adalah permainan papan strategi dua orang yang dimainkan pada sebuah papan kotak-kotak yang terdiri dari 64 kotak, yang disusun dalam petak 8×8, yang terbagi sama rata (masing-masing 32 kotak) dalam kelompok warna putih dan hitam.
.

Gambar 1.1 Contoh Catur
(sumber :chess.com)
diakses pada 30 November 2023

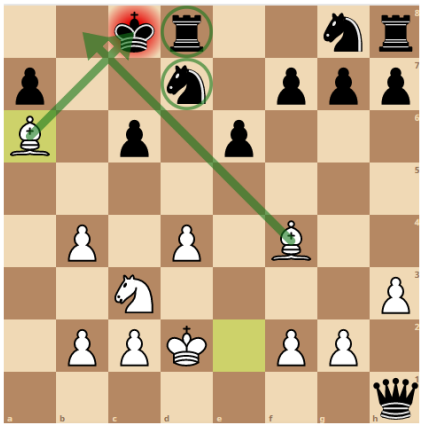
Saat permainan ini dimulai, tiap pemain diberi sejumlah *pion* yang memiliki keunikannya masing masing. Keunikan tersebut berupa bagaimana cara pion tersebut bergerak dan mengalahkan pion lawan lainnya. Setiap pion tersebut dengan keunikannya masing-masing diberikan sebuah nama yang umum digunakan dalam dunia catur, walaupun sering juga penamaannya berubah mengikuti budaya dari daerah dimana permainan tersebut dilakukan. Pada mulanya, setiap pemain akan memiliki 16 buah catur,yaitu satu Raja, satu Menteri, dua Benteng, dua Kuda, dua Gajah, dan delapan bidak Pion yang disusun seperti pada Gambar 1.1 , serta satu buah pion catur hanya bisa menempati satu petak.
.

Setiap pion tersebut memiliki Gerakan masing-masing. Dengan ketentuan sebagai berikut :

- Raja : dapat bergerak satu petak ke segala arah, kecuali jika dihalangi pon lain.
- Benteng : dapat bergerak sepanjang petak secara horizontal dan vertical, tetapi tidak dapat melompati pion catur lain.
- Gajah : dapat bergerak sepanjang petak secara diagonal, tetapi tidak dapat melompati pion catur lain.
- Menteri : memiliki kombinasi Gerakan dari benteng dan gajah, tetapi tidak dapat melompati buat catur lain.
- Kuda : dapat bergerak ke segala arah dengan Gerakan seperti huruf L, dengan memanjang satur petak dan melebar dua petak. Hanya kuda yang dapat melompati pion lain.
- Bidak : dapat bergerak maju (ke arah lawan) satu petak atau (khusus pada langkah awal) hingga dua petak ke petak yang tidak ditempati (titik hitam). Bidak hanya dapat bergerak untuk memakan buah catur lawan satu petak secara diagonal dan tidak dapat memakan buah yang ada di depannya (X hitam). Bidak tidak dapat berjalan mundur.

Dengan keunikan geraknya masing masing pion, dapat ditentukan bahwa pion terkuat adalah Menteri, karena dapat bergerak ke arah manapun, dan bidak Pion karena hanya dapat bergerak satu arah.

Permainan berlangsung dengan pemain yang memiliki bidak catur bewarna putih akan memulai Langkah pionnya pertama kali dan bergantian dengan pemain bidak hitam hingga permainan selesai. Setiap giliran pemain hanya boleh menggerakan satu buah pion saja. Apabila pada saat pemain memindahkan pionnya ke kotak yang berisi pemain lawan, maka pion lawan tersebut dapat disingkirkan dari atas papan catur. Setiap giliran pemain mutlak hukumnya untuk menggerakan salah satu pion catur miliknya, bahkan bila Gerakan tersebut dapat merugikan pemain tersebut. Setiap pemain sedapat mungkin tidak melakukan Langkah apapun yang akan membuat atau membiarkan pion Raja pemain tersebut masuk pada keadaan *sekak* melainkan memilih strategi untuk melakukan hal tersebut terhadap pion Raja pemain lawan. Apabila pemain tidakdapat lagi menggerakan raja(sesuai aturan) agar keluar dari keadaan sekak, maka Raja pemain tersebut masuk ke dalam keadaan sekakmat dan permainan berakhir dengan kemenangan pemain yang menyekakmat.

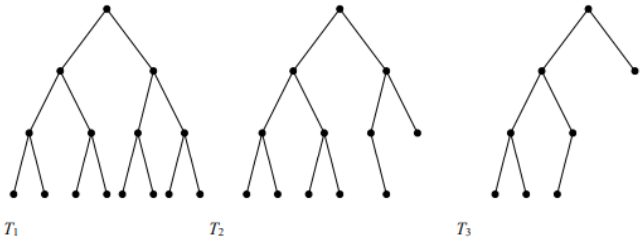


Gambar 1.2 Contoh Skakmat
(sumber :chess.com)
diakses pada 30 November 2023

Oleh karena banyaknya kemungkinan Gerakan yang terjadi dalam sebuah permainan catur, maka setiap pemain diharuskan membuat sebuah strategi yang seoptimal mungkin untuk dapat mengalahkan satu sama lain. Setiap Gerakan yang terjadi memiliki banyak sekali kemungkinan yang meningkat secara eksponensial selama permainan itu berlangsung. Idealnya jika salah satu pemain dapat memprediksi setiap Gerakan yang mungkin terjadi ke depannya, maka pemain tersebut dapat meingkatkan kemungkinan menangnya. Karena catur sendiri memiliki milyaran kemungkinan, maka akan sulit untuk membuat sebuah algoritma untuk menghitung setiap potensi Gerakan yang terjadi, maka solusi menentukan Gerakan terbaik adalah dengan cara menggunakan data statistic. Jika kita menggunakan sebuah Gerakan yang mengarahkan kita ke 99 kemenangan dari 100 game acak, maka kita dapat menilai Gerakan tersebut sebagai Gerakan yang bagus dan memiliki kecil kemungkinan membuat kita kalah. Inilah ide dari MTCS yang digunakan oleh Chess Bot secara umum. Chess bot hadir untuk membantu kita menentukan gerakan Gerakan terbaik yang dapat kita ambil untuk membantu kita memenangkan sebuah permainan.

II. Dasar Teori

A. Pohon



Gambar 2.1 Contoh Pohon
(sumber : belajarstatistk.com)
diakses pada 30 November 2023

Pohon dalam matematika diskrit adalah struktur hierarkis yang digunakan untuk merepresentasikan hubungan antar elemen. Secara umum, pohon terdiri dari node-node yang terhubung satu sama lain melalui sisi atau tepi. Setiap pohon memiliki satu node khusus yang disebut sebagai root, yang menjadi awal dari struktur hierarkis ini. Node-node yang terhubung langsung ke node root disebut sebagai child, sementara node yang berada pada ujung struktur dan tidak memiliki child disebut sebagai leaf.

Pohon memiliki properti-properti seperti tingkat (level), kedalaman (depth), dan tinggi (height). Tingkat suatu node didefinisikan sebagai jarak antara node tersebut dan root, sedangkan kedalaman adalah jarak dari suatu node ke root. Tinggi dari pohon adalah tingkat maksimum dari semua node di dalamnya.

Pohon juga dapat dipecah menjadi subtrees, yaitu bagian-bagian yang independen yang masih mempertahankan struktur pohon. Dalam matematika diskrit, pohon sering digunakan sebagai representasi visual untuk algoritma-algoritma, seperti pohon pencarian biner atau pohon merah-hitam dalam struktur data.

Dalam analisis algoritma, pohon digunakan untuk memodelkan kompleksitas waktu dan ruang. Misalnya, pohon keputusan dapat digunakan untuk merepresentasikan algoritma pengambilan keputusan, di mana setiap node internal mewakili pengujian terhadap suatu fitur, dan cabang-cabangnya menggambarkan hasil pengujian tersebut.

Dengan memahami abstraksi pohon dalam matematika diskrit, kita dapat mengaplikasikan konsep ini dalam berbagai konteks, termasuk permasalahan struktur data, algoritma pencarian, dan analisis kompleksitas algoritma.

B. Sifat-sifat Pohon

Misalkan G = (V, E) adalah graf sederhana, tak-berarah, dan memiliki n simpul, maka fakta-fakta ini benar dan ekivalen:

i) G adalah pohon.

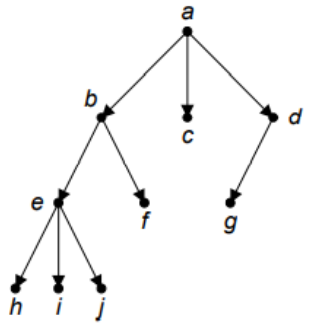
ii) Setiap pasang simpul di G terhubung dengan tepat satu lintasan.

iii) G terhubung dan memiliki n-1 buah sisi.

iv) G tidak memiliki sirkuit dan penambahan satu sisi pada graf akan membuat tepat satu sirkuit

v) G terhubung dan semua sisinya adalah jembatan

C. Pohon Berakar



Gambar 2.2 Contoh Pohon Berakar
(sumber : belajarstatistk.com)
diakses pada 30 November 2023

Pohon berakar adalah struktur pohon khusus yang memiliki sebuah simpul yang dianggap sebagai akar. Sisi-sisinya diarahkan, membentuk graf berarah yang merepresentasikan hierarki antar elemen. Dalam gambar 2.2, kita memperoleh gambaran visual mengenai pohon berakar, dan untuk memahami konsep ini dengan lebih mendalam, kita akan mendefinisikan beberapa terminologi kunci pada graf berarah.

i) Parent (Orang Tua):
Sebuah simpul v memiliki tepat satu simpul orang tua, disimbolkan sebagai u, jika terdapat sisi yang langsung mengarah menuju v. Dengan kata lain, u adalah orang tua dari v. Sebagai contoh, simpul e adalah orang tua dari h, i, dan j.

ii) Children (Anak):
Jika u adalah orang tua dari v, maka v adalah anak dari u. Artinya, setiap sisi yang keluar dari u menuju v menunjukkan bahwa v adalah anak dari u. Sebagai contoh, simpul j adalah anak dari e.

iii) Path (Lintasan):
Lintasan dari simpul u ke simpul v adalah urutan simpul yang dilewati untuk berjalan dari u ke v. Lintasan sederhana merupakan lintasan tanpa simpul yang diulang. Sebagai ilustrasi, lintasan dari a ke i adalah a, b, e, i, dan panjang lintasannya adalah tiga.

iv) Level (Tingkat):
Tingkat simpul v adalah panjang lintasan dari akar ke v. Misalnya, jika simpul b berada pada tingkat satu, ini berarti lintasan dari akar ke b memiliki panjang satu.

v) Sibling (Saudara):
Saudara adalah simpul-simpul yang memiliki orang tua yang sama. Dengan kata lain, simpul-simpul ini berada pada tingkat yang sama dalam pohon. Sebagai contoh, simpul e adalah saudara dari b.

vi) Subtree (Upapohon):
Subtree atau upapohon adalah bagian dari pohon berakar yang juga merupakan pohon berakar. Ini mencakup sebuah simpul dan semua anak-anaknya.

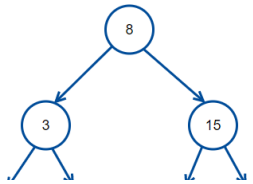
vii) Degree (Derajat):
Derajat sebuah simpul adalah jumlah anak pada simpul tersebut. Sebagai contoh, derajat i adalah nol, karena i adalah daun, sedangkan derajat g adalah tiga karena g memiliki tiga anak.

viii) Leaf (Daun):
Daun adalah simpul yang tidak memiliki anak (berderajat nol). Sebagai contoh, simpul i adalah daun karena tidak memiliki anak.

ix) Internal Nodes (Simpul Dalam):
Simpul dalam adalah simpul yang memiliki anak. Dalam konteks gambar 2.2, simpul b dan d adalah simpul dalam karena keduanya memiliki anak.

x) Depth (Kedalaman):
Kedalaman adalah panjang lintasan terpanjang dari akar ke daun yang ada di pohon. Sebagai contoh, jika lintasan terpanjang dari akar ke suatu daun adalah tiga simpul, maka kedalaman graf ini adalah tiga.

D. Pohon Biner



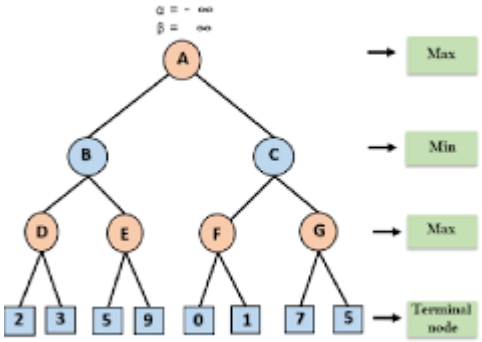
Gambar 2.1 Contoh Pohon Biner
(sumber : belajarstatistk.com)
diakses pada 30 November 2023 pukul 21.22

Pohon n-ary adalah sebuah pohon berakar yang anak dari setiap simpulnya paling banyak n buah. Pohon n-ary yang teratur atau penuh adalah pohon yang setiap simpulnya memiliki tepat n anak atau tidak memiliki anak sama sekali.

Pohon biner atau binary tree adalah pohon n-ary terurut dengan n sama dengan dua. Pohon tipe ini sangat banyak digunakan di dunia komputasi.

Simpul pohon biner paling banyak memiliki dua buah anak, dibedakan dengan anak kiri (atau left child) dan anak kanan (atau right child). Pohon biner yang penuh adalah pohon biner yang setiap simpulnya memiliki tepat dua anak atau tidak memiliki anak sama sekali. Pohon biner seimbang adalah pohon biner yang memiliki mutlak beda ketinggian subtree kanan dan kirinya maksimal satu.

E. Pohon Pencarian (Tree Search)



Gambar 2.3 Contoh Pohon Pencarian
(sumber : belajarstatistk.com)
diakses pada 30 November 2023

Pohon pencarian (search tree) adalah suatu struktur pohon yang digunakan untuk merepresentasikan dan memodelkan proses pencarian atau traversal dalam algoritma pencarian. Pohon ini memiliki sifat khusus yang memudahkan penyusunan dan pencarian informasi. Dalam konteks ini, kita akan memahami beberapa konsep dasar terkait dengan pohon pencarian.

i) Node pada Pohon Pencarian

Node dalam pohon pencarian merepresentasikan suatu keadaan atau konfigurasi tertentu dari masalah yang sedang diselesaikan. Setiap node dapat memiliki anak-anaknya sendiri yang merepresentasikan kemungkinan langkah atau keadaan selanjutnya. Dalam konteks pohon berakar, simpul akar adalah representasi awal dari masalah, sementara simpul-simpul lain adalah langkah-langkah atau keadaan selanjutnya yang mungkin diambil.

ii) Parent dan Children pada Pohon Pencarian

Setiap node pada pohon pencarian memiliki orang tua (parent) yang merupakan simpul yang langsung menuju ke simpul tersebut. Sebaliknya, setiap node dapat memiliki anak-anak (children) yang merepresentasikan langkah-langkah atau keadaan selanjutnya yang dapat diambil dari simpul tersebut.

iii) Depth-First Search (DFS) dan Breadth-First Search (BFS)

- Depth-First Search (DFS):

  DFS melakukan pencarian sejauh mungkin ke dalam pohon sebelum beralih ke cabang lain. Ini dapat dilakukan secara rekursif atau dengan menggunakan tumpukan (stack). DFS sering digunakan dalam konteks pencarian jalur atau solusi dalam struktur pohon pencarian.
- Breadth-First Search (BFS):

  Sebaliknya, BFS melakukan pencarian secara berlapis, mengunjungi semua node pada tingkat yang sama sebelum bergerak ke tingkat berikutnya. Ini dapat dilakukan dengan menggunakan antrian (queue). BFS berguna dalam menemukan solusi terpendek atau mencari solusi pada tingkat tertentu dalam pohon pencarian.

F. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) adalah metode pohon pencarian yang efektif dan sering digunakan dalam pengambilan keputusan pada permainan, terutama dalam konteks kecerdasan buatan. MCTS memadukan konsep pohon pencarian dengan pendekatan Monte Carlo untuk mengeksplorasi ruang keputusan secara adaptif. Dalam penerapannya pada pohon pencarian, MCTS melibatkan beberapa tahapan utama:

i) Seleksi (Selection):

Pada tahap ini, MCTS memilih simpul dari pohon yang akan dieksplorasi lebih lanjut. Seleksi dilakukan berdasarkan suatu kebijakan, seperti Upper Confidence Bound (UCB), yang mempertimbangkan nilai dan eksplorasi dari setiap simpul.

ii) Ekspansi (Expansion):

Setelah simpul terpilih, tahap ekspansi melibatkan penambahan anak-anak baru ke simpul tersebut. Anak-anak ini merepresentasikan langkah-langkah atau

keadaan selanjutnya yang mungkin diambil dari simpul tersebut.

iii) Simulasi (Simulation):

MCTS melakukan simulasi permainan dengan memainkan langkah-langkah acak dari simpul ekspansi ke suatu keadaan akhir atau batasan yang ditetapkan. Simulasi ini memberikan perkiraan nilai dari simpul tersebut.

iv) Backpropagation:

Setelah simulasi selesai, nilai yang diperoleh dari simulasi tersebut dikembalikan ke simpul-simpul yang terlibat dalam seleksi dan ekspansi. Ini memungkinkan pembaruan nilai-nilai dan penghitungan jumlah kunjungan (visit count) pada setiap simpul.

MCTS memungkinkan pohon pencarian untuk secara adaptif menyesuaikan diri dengan perubahan kondisi permainan dan memfokuskan eksplorasi pada langkah-langkah yang memiliki potensi kemenangan lebih tinggi. Keunggulan utama MCTS terletak pada kemampuannya untuk beroperasi tanpa pengetahuan eksplisit dalam domain tertentu, membuatnya cocok untuk berbagai jenis permainan.

Ini lebih baik daripada algoritma pencarian pohon tradisional karena tidak mengeksplorasi setiap keadaan yang mungkin untuk membuat keputusan karena memiliki jutaan triliun kemungkinan keadaan yang tidak mungkin disimpan dan diproses.

Dalam konteks pengembangan chess bot, MCTS dapat diintegrasikan dengan pohon pencarian untuk meningkatkan keputusan adaptif dan dinamis pada setiap iterasi permainan. Dengan kombinasi konsep pohon pencarian dan MCTS, kita dapat merancang sistem kecerdasan buatan yang mampu belajar dan beradaptasi dalam permainan catur online yang kompleks.

G. Upper Confidence Bound (UCB)

UCB merupakan faktor yang menentukan simpul mana yang akan dievaluasi selanjutnya untuk memaksimalkan probabilitas kemenangan dari keadaan yang diberikan. Secara umum, UCB terdiri dari dua faktor utama, yaitu Eksploitasi (Exploitation) dan Eksplorasi (Exploration). Formulanya dapat dituliskan sebagai berikut:

$$UCB = V + 2\,x\,\sqrt{\frac{\ln N}{n_i}}$$

Di mana:
V adalah skor kemenangan dari simpul saat ini.
N adalah jumlah kali simpul induk telah dikunjungi.
n_i adalah jumlah kali simpul anak telah dikunjungi.

Formula ini mencerminkan keseimbangan antara Eksploitasi dan Eksplorasi:

i) Eksploitasi (Qt(a)):

Diwakili oleh Qt(a), ini mengukur seberapa sukses tindakan 'a' ketika dipilih. Semakin tinggi tingkat keberhasilan, semakin besar nilai UCB, menekankan aspek eksploitasi dalam pengambilan keputusan.

ii) Eksplorasi (Nt(a)):

Dilambangkan sebagai Nt(a), ini merepresentasikan jumlah kali tindakan 'a' telah diambil. Frekuensi rendahnya tindakan 'a' menghasilkan nilai UCB yang lebih tinggi, mendorong algoritma untuk menjelajahi kemungkinan terkait tindakan tersebut.

Parameter 'c' diperkenalkan sebagai konstanta eksplorasi, menentukan seberapa sering algoritma harus menjelajahi kemungkinan baru. Jika sistem melibatkan banyak keadaan, disarankan untuk menjaga 'c' tetap rendah untuk mengurangi kompleksitas komputasional. Eksperimen dengan nilai 'c' dianjurkan untuk menemukan pengaturan optimal.

Rumus UCB lengkap adalah:

$$UCB = V + 2\,x\,\sqrt{\frac{\ln N}{n_i}}$$

III. ANALISIS PENERAPAN MCTS

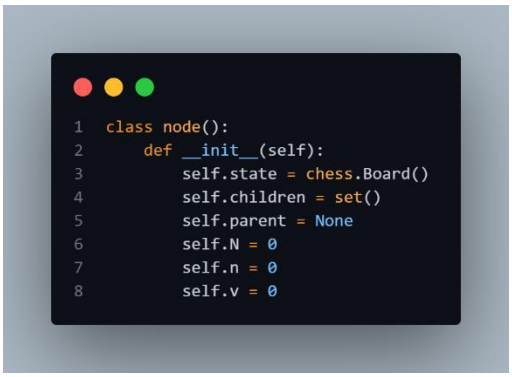
Pada bagian ini akan diimplementasikan Monte Carlo Tree Search pada bentuk kode pemrograman Bahasa Python yang sederhana dan umum digunakan (Bukan implementasi kode murni penulis melainkan umum digunakan).

A. UCB

Gambar 3.1 Kode Python UCB
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

konstanta $\frac{1}{10^6}$ dan $\frac{1}{10^{10}}$ ditambahkan ke dalam formula ini untuk mencegah terjadinya pembagian oleh nol.

B. Inisiasi



```python
class node():
    def __init__(self):
        self.state = chess.Board()
        self.children = set()
        self.parent = None
        self.N = 0
        self.n = 0
        self.v = 0
```

Gambar 3.2 Kode Python Inisiasi
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

i) State (Keadaan)

State merujuk pada posisi terkini dari papan catur. Dalam konteks ini, state mencakup susunan semua bidak di papan catur pada suatu saat dalam permainan.

ii) Children (Anak-anak)

Children merujuk pada himpunan semua kemungkinan keadaan atau posisi yang dapat dihasilkan dari langkah-langkah legal yang dapat diambil dari node (simpul) saat ini. Dalam catur, children akan mewakili berbagai kemungkinan langkah yang dapat diambil oleh pemain pada suatu giliran tertentu.

iii) Parent (Induk)

Parent merujuk pada node (simpul) yang menjadi "induk" atau "orang tua" dari simpul saat ini dalam struktur pohon. Dalam konteks MCTS (Monte Carlo Tree Search), ini adalah simpul yang membawa keadaan atau posisi sebelum langkah tertentu diambil.

iv) N

N adalah jumlah kali simpul induk (parent node) telah dikunjungi. Dalam algoritma UCB (Upper Confidence Bound), nilai N digunakan untuk menghitung eksplorasi dalam pemilihan langkah selanjutnya.

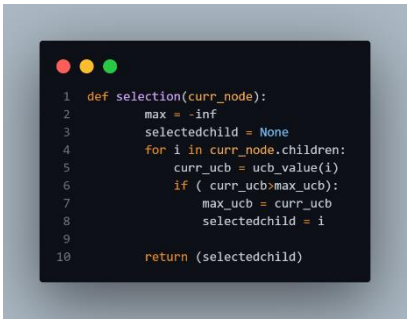
v) n

n adalah jumlah kali simpul saat ini (current node) telah dikunjungi. Seperti N, nilai n juga digunakan dalam algoritma UCB untuk menghitung eksplorasi dan membantu dalam pengambilan keputusan untuk memilih langkah.

vi) v

v adalah faktor eksploitasi dari simpul saat ini. Dalam algoritma UCB, v digunakan untuk mengukur seberapa baik atau buruk suatu langkah berdasarkan hasil kunjungan sebelumnya. Faktor ini membantu algoritma memilih langkah yang paling menjanjikan berdasarkan sejarah kunjungan dan hasil yang telah dicapai.

C. Selection



```python
def selection(curr_node):
        max = -inf
        selectedchild = None
        for i in curr_node.children:
            curr_ucb = ucb_value(i)
            if ( curr_ucb>max_ucb):
                max_ucb = curr_ucb
                selectedchild = i

        return (selectedchild)
```

Gambar 3.3 Kode Python Selection
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

Dalam fungsi ini, kita secara sederhana melakukan iterasi melalui semua anak dari keadaan (state) yang diberikan dan memilih yang memiliki nilai UCB tertinggi.

Fungsi seleksi merupakan langkah penting dalam algoritma MCTS (Monte Carlo Tree Search) yang digunakan untuk memilih langkah atau aksi selanjutnya dalam permainan. Pada setiap iterasi, algoritma MCTS memutuskan anak simpul yang akan dijelajahi lebih lanjut, dan pemilihan ini didasarkan pada nilai UCB (Upper Confidence Bound). Dengan proses sebagai berikut :

i) Iterasi melalui semua anak (children) dari simpul saat ini.
ii) Menghitung nilai UCB untuk setiap anak menggunakan rumus tertentu, yang umumnya melibatkan faktor eksploitasi dan eksplorasi.
iii) Memilih anak yang memiliki nilai UCB tertinggi sebagai langkah selanjutnya.

Fungsi seleksi bertujuan untuk menentukan langkah berikutnya dengan mempertimbangkan keseimbangan antara eksplorasi (mengeksplorasi langkah-langkah yang belum banyak dijelajahi) dan eksploitasi (memilih langkah yang memiliki hasil yang baik berdasarkan kunjungan sebelumnya).

Dengan memilih anak dengan nilai UCB tertinggi, algoritma berusaha untuk memilih langkah yang memiliki potensi terbaik untuk memberikan hasil yang baik. Pemilihan ini dilakukan dengan memperhitungkan seberapa sering simpul tersebut telah dikunjungi (faktor eksploitasi) dan seberapa banyak algoritma ingin menjelajahi langkah tersebut (faktor eksplorasi).

Selain itu, algoritma MCTS secara adaptif memilih langkah berdasarkan perubahan nilai UCB setiap saat,

memungkinkan penyesuaian strategi selama proses pencarian.

D. Expansion

```
def expansion(cur_node):
        if (cur_node.children.empty()):
            return cur_node
        max_ucb = -inf
        selected_child = None
        for i in cur_node.children:
            curr_ucb = ucb_value(i)
            if ( curr_ucb>max_ucb):
                max_ucb = curr_ucb
                selected_child = i
        return node.expansion(selected_child)
```

Gambar 3.4 Kode Python Expansion
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

Dalam fungsi ini, kita terus memanggil anak dengan prioritas tertinggi hingga mencapai simpul daun (leaf node) saat ini.

Fungsi ekspansi adalah langkah berikutnya dalam algoritma MCTS (Monte Carlo Tree Search), yang bertanggung jawab untuk mengekspansi pohon pencarian dengan menambahkan simpul anak baru dari simpul saat ini. Dengan proses sebagai berikut :

i) Memeriksa apakah simpul saat ini sudah mencapai kondisi daun (leaf node) atau belum.
ii) Jika simpul saat ini masih bukan daun, pilih anak dengan prioritas tertinggi dan lanjutkan ke anak tersebut.
iii) Ulangi langkah 1 dan 2 hingga mencapai simpul daun.
iv) Tambahkan satu atau lebih simpul anak baru ke simpul daun.

Fungsi ekspansi dimulai dengan memeriksa apakah simpul saat ini sudah mencapai kondisi daun atau masih dapat diekspansi lebih lanjut. Jika simpul saat ini belum mencapai kondisi daun, langkah berikutnya adalah memilih anak dengan prioritas tertinggi.

Pemilihan anak dengan prioritas tertinggi dapat melibatkan nilai UCB yang dimiliki. Proses ini terus diulang hingga mencapai simpul daun, yaitu simpul yang belum memiliki anak. Pada tahap ini, satu atau lebih simpul anak baru ditambahkan ke pohon pencarian.

Dalam konteks Chess Bot yang menggunakan algoritma MCTS, fungsi ekspansi diterapkan untuk memperluas pohon pencarian berdasarkan langkah-langkah yang mungkin. Saat mencari langkah terbaik untuk dipilih, Chess Bot perlu memperluas pohon pencarian dengan menambahkan langkah-langkah baru ke simpul daun. Proses ini membantu meningkatkan keragaman dan eksplorasi langkah-langkah yang mungkin, memberikan gambaran yang lebih lengkap tentang ruang pencarian langkah dalam permainan catur.

E. Rollout

```
def rollout(curr_node):
        if(curr_node.game_over()):
            if(won):
                return (1,curr_node)
            elif(lose):
                return (-1,curr_node)
            else:
                return (0.5,curr_node)
        curr_node.children = generate_all_states(cur_node)
        random_child = random.choice(curr_node.children)
        return rollout(random_child)
```

Gambar 3.5 Kode Python Rollout
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

Setelah mendapatkan simpul daun dari fungsi ekspansi (Expansion), akan dipanggil fungsi penjelajahan (rollout), yang akan membuat langkah-langkah acak dari simpul yang diterima dari ekspansi hingga mencapai akhir permainan dan akan mengembalikan simpul daun. Catatan - Dari titik ini tidak mungkin ada keadaan baru karena permainan sudah berakhir. Kita akan mengembalikan simpul ini dengan imbalan (+1) jika menang di akhir atau (-1) jika kalah atau (+0.5) jika seri.

Fungsi penjelajahan adalah bagian penting dari algoritma MCTS (Monte Carlo Tree Search), yang bertanggung jawab untuk mengevaluasi hasil dari suatu simpul dengan menjalankan simulasi permainan acak dari simpul tersebut hingga akhir permainan. Proses rolling adalah sebagai berikut :

i) Memulai dari simpul daun yang dihasilkan oleh fungsi ekspansi.
ii) Melakukan langkah-langkah acak dari simpul tersebut hingga mencapai kondisi akhir permainan.
iii) Mengembalikan simpul daun sebagai hasil penjelajahan.

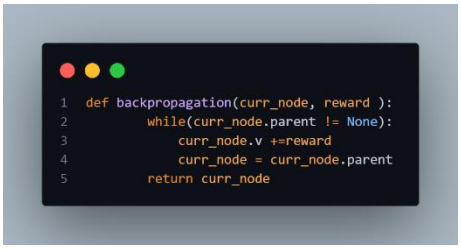
Fungsi penjelajahan dimulai dengan simpul daun yang telah dihasilkan oleh fungsi ekspansi. Selanjutnya
langkah-langkah acak dilakukan dari simpul tersebut hingga permainan mencapai akhir. Ini menggambarkan pendekatan simulasi untuk menentukan hasil dari suatu posisi dalam permainan catur.

Ketika permainan mencapai kondisi akhir, fungsi penjelajahan mengembalikan simpul daun sebagai hasil. Hasil ini kemudian digunakan untuk memberikan return nilai ke simpul-simpul dalam perjalanan kembali ke simpul awal.

Dalam konteks Chess Bot yang menggunakan algoritma MCTS, fungsi penjelajahan memberikan perkiraan hasil dari suatu posisi dalam permainan catur dengan melakukan simulasi langkah-langkah acak. Hasil ini kemudian digunakan untuk memperbarui informasi di sepanjang jalur yang diambil oleh algoritma MCTS.

Dengan memasukkan elemen penjelajahan ke dalam proses pengambilan keputusan, Chess Bot dapat lebih baik mengeksplorasi dan mengevaluasi potensi hasil dari berbagai langkah yang mungkin.

F. Backpropagation



Gambar 3.6 Kode Python Backpropagation
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

Ketika kita menerima simpul dan return nilai akhir dari fungsi penjelajahan (rollout), kita akan menelusuri imbalan tersebut hingga ke akar pohon, yang pada gilirannya akan memperbarui nilai UCB (Upper Confidence Bound) dari setiap simpul di jalur tersebut. Dengan proses sebagai berikut :

i) Memulai dari simpul daun yang telah menerima hasil dari penjelajahan.
ii) Menelusuri imbalan dari simpul tersebut hingga ke akar pohon.
iii) Memperbarui nilai UCB dari setiap simpul di jalur tersebut.

Fungsi pemrosesan balik dimulai dengan simpul daun yang telah menerima hasil dari penjelajahan. Imbalan ini kemudian ditelusuri kembali melalui jalur yang diambil selama penjelajahan hingga mencapai akar pohon.

Selama proses ini, nilai UCB dari setiap simpul di jalur tersebut diperbarui berdasarkan hasil yang diterima. Pemrosesan balik bertujuan untuk memperbarui informasi di seluruh jalur yang diambil oleh algoritma MCTS, sehingga keputusan yang diambil oleh algoritma tersebut dapat lebih tepat dan terinformasi.

Dalam konteks Chess Bot yang menggunakan algoritma MCTS, fungsi pemrosesan balik memainkan peran penting dalam meningkatkan informasi di pohon pencarian. Informasi ini mencakup hasil dari simulasi permainan yang telah dilakukan selama penjelajahan.

Pemrosesan balik memastikan bahwa hasil dari penjelajahan disebarkan ke seluruh pohon pencarian, membantu algoritma MCTS untuk lebih baik menilai potensi hasil dari berbagai jalur. Dengan demikian, proses pemrosesan balik berkontribusi pada pengambilan keputusan yang lebih baik dan lebih terinformasi oleh Chess Bot.

G. Implementasi dalam permainan catur

Kita dapat menggunakan library pychess dalam python yang memiliki fungsi bawaan yang dapat kita gunakan seperti chess.Board(), chess.Board().legal_moves dan lain sebagainya. Salah satu fungsi main yang dapat digunakan adalah sebagai berikut :

Gambar 3.7 Kode Python main
(sumber : VisualStudioCode)
diakses pada 1 Desember 2023

IV. KESIMPULAN

Berdasarkan penjelasan di atas, dapat disimpulkan bahwa teori pohon pencarian dapat dikombinasikan dengan ilmu pengetahuan lain untuk membantu kita dalam menyelesaikan berbagai permasalahan yang ada. Salah satunya adalah kita dapat mengkombinasikan antara teori pohon pencarian yaitu Monte Carlo Tree Search dengan ilmu statistik untuk membangun sebuah chess bot yang dapat kita gunakan untuk menentukan strategi terbaik dalam bermain catur. Meskipun keindahan dari sebuah permainan catur sebenarnya ada di bagian mengatur strategi bermain, tetapi chess bot tidaklah masalah apabila kita gunakan secara tepat dan dengan tujuan yang baik.

Metode pembuatan chess bot saat ini mungkin saja masih bisa dikembangkan dengan menggabungkannya dengan ilmu disiplin lain dan dengan studi yang lebih lanjut, bisa saja penerapan pohon pencarian Monte Carlo ini dimanfaatkan untuk menyelesaikan berbagai permasalahan lain yang lebih kompleks.

V. UCAPAN TERIMA KASIH

Dengan rendah hati, penulis ingin menyampaikan terima kasih kepada berbagai pihak yang telah berkontribusi dalam penyelesaian makalah ini. Keberhasilan penulisan ini tidak hanya merupakan hasil usaha individu penulis, tetapi juga dukungan dan bimbingan dari beberapa pihak lain.

Pertama-tama, penulis ingin bersyukur kepada Tuhan Yang Maha Esa atas limpahan rahmat dan rezekinya yang senantiasa mengiringi penulis dalam perjalanan penulisan makalah ini.

Tidak lupa, penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., yang telah memberikan pengajaran dalam perkuliahan IF2120 Matematika Diskrit. Bimbingan dan pemahaman yang diberikan telah membantu penulis dalam menyusun makalah ini. Serta Bapak Dr. Ir. Rinaldi, M.T. yang atas

penyediaan sarana websitenya yang sangat bermanfaat selama proses pembelajaran. Sarana tersebut telah menjadi sumber referensi yang berharga bagi penulis. Terima kasih juga disampaikan kepada Nasywaa Anggun Athiefah, atas dukungannya dalam membantu penulis mengerjakan makalah ini. Harapan penulis, makalah ini dapat memberikan manfaat dan kontribusi positif. Terima kasih.

REFERENSI

Carlo, M. (n.d.). Retrieved from https://www.chessprogramming.org/Monte-Carlo_Tree_Search

Games, A. a. (n.d.). Retrieved from https://www.youtube.com/watch?si=YiX29xUP6gKWI3c2&v=lhFXKNyA0QA&feature=youtu.be

Munir, R. (n.d.). Retrieved from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/23-Pohon-Bag2-2023.pdf

Munir, R. (2023). Retrieved from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/22-Pohon-Bag1-2023.pdf

Wikipedia. (n.d.). Retrieved from https://id.wikipedia.org/wiki/Catur

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, 1 Desember 2023

Mohammad Nugraha Eka Prawira - 13522001