Implementasi Kecerdasan Buatan Pada Permainan Eggy Chess Dengan Pendekatan Algoritma Pencarian Heuristik

Muhammad Zahran Ramadhan Ardiana - 13523104[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13523104@std.stei.itb.ac.id muzaraar@gmail.com

***Abstract*—**
***Eggy Chess* adalah sebuah mini-game strategis berbasis papan yang terdapat dalam MMORPG Ragnarok Origin Global, yang memodifikasi permainan Tic-Tac-Toe klasik dengan menambahkan mekanisme ukuran bidak yang kompleks. Penelitian ini berfokus pada perancangan dan implementasi sebuah agen kecerdasan buatan (bot) yang mampu membuat keputusan optimal dengan memanfaatkan pendekatan algoritma pencarian heuristik. Logika bot didasarkan pada pencarian kedalaman satu (1-ply search), di mana setiap kemungkinan langkah dievaluasi menggunakan fungsi heuristik gabungan yang menilai berbagai aspek strategis, termasuk kontrol posisi, keunggulan material bidak, dan analisis ancaman ofensif serta defensif seperti manuver fork. Hasil implementasi menunjukkan bahwa bot yang dikembangkan tidak hanya mampu bermain secara kompeten dengan melakukan manuver taktis seperti membuat fork dan memblokir ancaman lawan, tetapi juga berhasil mengungkap adanya strategi kemenangan yang dapat dipaksakan (forced-win) bagi pemain pertama, yang mengindikasikan adanya ketidakseimbangan dalam desain permainan itu sendiri. Studi ini mendemonstrasikan efektivitas pendekatan heuristik dalam menciptakan lawan AI yang cerdas untuk permainan dengan aturan yang terdefinisi dengan baik, sekaligus berfungsi sebagai alat analisis untuk memvalidasi dan mengkritisi desain mekanika permainan.**

***Keywords*— Eggy Chess, Pencarian Heuristik, Algoritma Minimax, Pohon Permainan.**

I. PENDAHULUAN

Permainan strategis berbasis papan (*board game*) telah lama menjadi medium fundamental dalam penelitian dan pengembangan kecerdasan buatan (AI). Permainan seperti catur dan Go menjadi arena pengujian klasik untuk mengukur kemampuan mesin dalam berpikir logis, merencanakan, dan membuat keputusan dalam lingkungan yang terstruktur dan kompetitif. Sifat permainan ini yang memiliki aturan jelas, tujuan terukur, dan ruang keadaan (*state space*) yang terdefinisi dengan baik menjadikannya lingkungan yang ideal untuk menguji seberapa "cerdas" sebuah agen buatan dalam membuat keputusan, merencanakan strategi, dan mengantisipasi langkah lawan. Keberhasilan atau kegagalan sebuah pendekatan AI dalam domain ini dapat dianalisis secara objektif, menjadikan permainan papan sebagai tolok ukur penting dalam kemajuan ilmu komputer, khususnya pada bidang pencarian heuristik.

Di dalam dunia Massively Multiplayer Online Role-Playing Games (MMORPG) seperti Ragnarok Origin Global, seringkali diadakan berbagai mini-game event yang dirancang untuk memberikan variasi dan hiburan bagi para pemainnya. Salah satu mini-game yang muncul secara periodik pada event tertentu adalah Eggy Chess (atau terkadang disebut juga Dumpling Chess). Meskipun bersifat sementara dan dijadikan sebagai konten sampingan, banyak *player* yang mencoba untuk mendapatkan peringkat tinggi dalam event sampingan tersebut. Sehingga sangat menarik untuk diterapkan prinsip-prinsip AI untuk mencapai kemenangan dalam permainan tersebut.

Eggy Chess sendiri, meskipun tampak sederhana karena berbasis pada Tic-Tac-Toe, memperkenalkan kedalaman strategis yang signifikan melalui mekanismenya yang unik: sistem ukuran bidak. Aturan yang memperbolehkan pemain untuk menimpa bidak lawan yang ukurannya lebih kecil secara dasar mengubah permainan dari sekadar adu formasi menjadi pertarungan yang melibatkan manajemen sumber daya, kontrol papan, dan antisipasi. Kompleksitas tambahan tersebut yang membuatnya menjadi sebuah game studi kasus yang ideal. Dengan kata lain, permainan Eggy Chess dalam konteks makalah ini berfungsi sebagai sebuah tolok ukur untuk mengukur seberapa efektif dan sesuai penggunaan serangkaian fungsi heuristik yang diimplementasikan dalam menghadapi tantangan strategis yang dinamis.

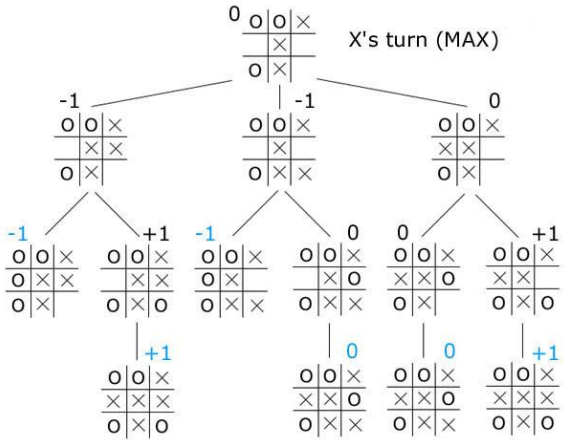
Kombinasi antara sifat temporer permainan ini di dalam Ragnarok Origin dan kompleksitas strategisnya melahirkan motivasi utama dari penelitian ini. Tujuan proyek ini terbagi menjadi dua. Pertama, untuk menciptakan sebuah lawan komputer (bot) yang kompeten dan menantang bagi para pemain Eggy Chess. Kedua, menggunakan permainan ini sebagai sebuah eksperimen untuk menjawab seberapa efektif heuristik yang dirancang dalam mengarahkan bot/AI untuk membuat Keputusan/langkah yang terbaik? Makalah ini secara mendalam akan menganalisis performa bot sebagai cerminan langsung dari keberhasilan atau kegagalan desain heuristik tersebut.

II. KAJIAN TEORI

A. *Game Tree*

Setiap permainan berbasis giliran (*turn based game)* dengan informasi sempurna dapat direpresentasikan sebagai

sebuah struktur data pohon permainan atau *game tree*. Pohon adalah jenis graf khusus yang terhubung dan tidak mengandung siklus. Dalam konteks permainan, setiap simpul (node) pada pohon melambangkan sebuah keadaan papan (*board state*) yang unik, dan setiap cabang (*edge*) yang menghubungkan dua simpul melambangkan sebuah langkah (*move*) yang mengubah satu keadaan ke keadaan lainnya. Simpul paling atas, yang tidak memiliki induk, disebut sebagai akar (*root*) dan merepresentasikan kondisi awal permainan. Dari *root*, semua kemungkinan keadaan lain dalam permainan dapat diakses melalui jalur-jalur yang ada. Karena setiap keadaan papan dalam Eggy Chess dapat memiliki banyak kemungkinan langkah lanjutan (kombinasi dari posisi dan ukuran bidak), maka pohon permainannya secara spesifik dapat diklasifikasikan sebagai Pohon N-ary, yaitu pohon di mana setiap simpulnya dapat memiliki paling banyak N anak. Algoritma AI pada dasarnya bekerja dengan cara menavigasi pohon ini untuk menemukan cabang atau langkah yang paling menguntungkan.



Gambar 2.1 Contoh Game Tree
(Sumber: https://www.yosenspace.com/posts/computer-science-game-trees.html)

B. *Algoritma Greedy Best-First Search (GBFS)*

Dalam menavigasi pohon permainan, berbagai algoritma pencarian dapat digunakan. Untuk kasus Eggy Chess di mana fokus utama adalah pada efektivitas fungsi heuristik untuk memilih langkah terbaik berikutnya, prinsip dari algoritma *Greedy Best-First Search* (GBFS) menjadi sangat relevan. GBFS adalah salah satu jenis algoritma pencarian terinformasi (informed search) yang menggunakan fungsi heuristik untuk memandu proses pencariannya. Sesuai dengan namanya, algoritma ini bersifat "rakus" (greedy) di mana selalu memilih untuk mengeksplorasi simpul yang tampaknya paling dekat dengan tujuan, berdasarkan nilai yang diberikan oleh fungsi heuristik saja, tanpa mempertimbangkan biaya atau jumlah langkah yang telah ditempuh untuk mencapai simpul tersebut. Logika ini sangat mirip dengan proses pengambilan keputusan bot Eggy Chess yang mengevaluasi semua kemungkinan langkah dan secara langsung memilih langkah yang menghasilkan keadaan dengan skor heuristik tertinggi.

Secara umum, cara kerja algoritma GBFS dapat diuraikan ke dalam beberapa tahapan utama sebagai berikut:

1. *Inisialisasi Priority Queue*

   Struktur data ini, seringkali diimplementasikan sebagai heap, secara otomatis mengurutkan elemen di dalamnya berdasarkan nilai prioritas tertentu. Pada GBFS, prioritas ini adalah nilai heuristik dari setiap keadaan. Langkah pertama adalah memasukkan keadaan awal permainan (simpul akar dari pohon permainan) ke dalam priority queue. Selain itu, sebuah himpunan atau daftar visited juga disiapkan untuk mencatat simpul-simpul yang sudah pernah dieksplorasi untuk menghindari penjelajahan berulang dan potensi perulangan tak terbatas (infinite loops), terutama pada graf yang mengandung siklus.

2. *Proses Iterasi dan Ekspansi Simpul*

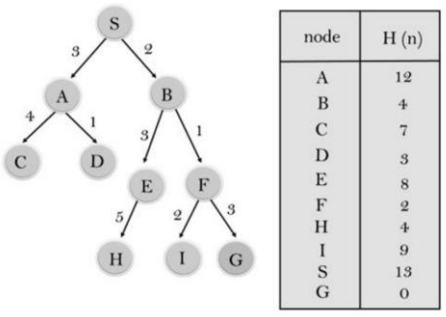
   Selama priority queue tidak kosong, algoritma akan masuk ke dalam sebuah loop iterasi. Pada setiap iterasi, ia akan mengambil simpul dengan prioritas tertinggi (nilai heuristik terbaik) dari semua simpul yang ada di dalam *prio queue*. Simpul kemudian "diekspansi", yang berarti algoritma akan membangkitkan semua kemungkinan keadaan berikutnya yang dapat dicapai dari simpul saat ini melalui satu langkah yang valid.

3. *Evaluasi dan Penambahan Simpul Baru*

   Setiap simpul anak yang berhasil dibangkitkan dari proses ekspansi akan dievaluasi menggunakan fungsi heuristik untuk mendapatkan skornya masing-masing sebelum ditambahkan ke priority queue untuk dipertimbangkan pada iterasi-iterasi selanjutnya.

4. *Terminasi dan Penemuan Solusi*

   Loop iterasi akan terus berlanjut hingga salah satu dari dua kondisi terminasi terpenuhi.
   - Simpul yang diambil dari *priority queue* adalah simpul tujuan (misalnya, keadaan papan di mana salah satu pemain menang), maka pencarian dianggap berhasil dan algoritma berhenti dengan mengembalikan jalur atau solusi yang ditemukan.
   - Jika *priority queue* menjadi kosong sementara simpul tujuan belum ditemukan, itu berarti tidak ada lagi jalur yang bisa dieksplorasi. Dalam kasus ini, algoritma berhenti dan menyimpulkan bahwa tidak ada solusi yang dapat ditemukan.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

Gambar 2.2 Contoh Tree GBFS dengan nilai Heuristic h(n)
(Sumber: [2])

Meskipun bot Eggy Chess kita tidak secara eksplisit mengimplementasikan *priority queue* untuk pencarian multi-langkah, logika pengambilan keputusannya pada setiap giliran mencerminkan prinsip GBFS secara menyeluruh. Bot mengevaluasi semua kemungkinan langkah (anak-anak dari simpul saat ini), menghitung skor heuristik untuk setiap keadaan hasil langkah tersebut dan memilih langkah yang menghasilkan skor heuristik tertinggi.

C. *Heuristic*

Fungsi evaluasi heuristik adalah komponen yang mendefinisikan kecerdasan dari AI berbasis pencarian. Heuristik adalah sebuah fungsi yang dirancang untuk memberikan estimasi atau nilai aproksimasi terhadap kualitas sebuah keadaan permainan.

Dalam permainan yang kompleks, sulit bagi komputer untuk menganalisis seluruh pohon permainan hingga akhir. Heuristik digunakan sebagai jalan pintas untuk menilai sebuah posisi tanpa harus melakukan pencarian mendalam. Fungsi ini mengubah fitur-fitur kualitatif dari seperti kontrol area tengah, jumlah bidak tersisa, atau adanya ancaman (dalam Eggy Chess) menjadi sebuah nilai numerik. Semakin baik sebuah fungsi heuristik dalam merepresentasikan peluang kemenangan, semakin baik juga keputusan yang akan diambil oleh bot.

- **Admissible Heuristic**: Sebuah heuristik dikatakan admissible jika ia tidak pernah melebih-lebihkan estimasi biaya atau jarak sebenarnya untuk mencapai tujuan. Sifat optimis ini adalah syarat penting bagi algoritma seperti A* untuk dapat menjamin penemuan solusi yang optimal. Namun, untuk algoritma seperti GBFS atau implementasi bot Eggy Chess kita yang hanya fokus pada pemilihan langkah terbaik berikutnya berdasarkan skor relatif, sifat admissibility tidak menjadi sebuah keharusan. Fokus utama dari heuristik yang dirancang adalah kemampuannya untuk secara konsisten memberikan skor lebih tinggi pada posisi yang lebih baik, dan skor lebih rendah pada posisi yang lebih buruk.

D. *Eggy Chess*

Gambar 2.3 Permainan Eggy Chess
(Sumber: https://www.youtube.com/watch?v=v-bavWZD094)

Eggy Chess merupakan sebuah mini-game strategis *turn based* yang terdapat dalam MMORPG Ragnarok Origin Global. Permainan ini memiliki aturan yang sederhana namun perlu langkah strategis untuk menang.

1. *Sistem Permainan dan Bidak*

   Permainan dimainkan oleh dua pemain di atas papan berukuran 3x3. Setiap pemain memiliki persediaan bidak dengan tiga ukuran berbeda: kecil, sedang, dan besar. Secara bergantian, pemain meletakkan satu bidak di petak yang tersedia. Aturan utamanya adalah pemain dapat menempatkan bidaknya di petak yang sudah terisi oleh bidak lawan, asalkan bidak miliknya berukuran lebih besar. Pemain tidak dapat menimpa bidak miliknya sendiri.
2. *Kondisi Kemenangan*

   Kemenangan dapat diraih melalui dua cara. Cara utama adalah dengan berhasil menjadi pemain pertama yang menyusun tiga bidak miliknya dalam satu baris lurus, baik secara horizontal, vertikal, maupun diagonal. Kedua, jika permainan mencapai kondisi di mana tidak ada lagi pemain yang dapat melakukan langkah valid (misalnya karena semua petak sudah terisi oleh bidak berukuran besar atau kehabisan bidan yang bisa digunakan untuk menimpa), maka kondisi akhir permainan tercapai. Pemenang pada skenario ini ditentukan berdasarkan jumlah bidak yang berada pada papan permainan.

III. IMPLEMENTASI

A. *Arstiektur Program*

Program ini dibangun menggunakan bahasa Java dengan pendekatan desain modular yang memisahkan tanggung jawab ke dalam beberapa package untuk meningkatkan keterbacaan dan kemudahan pengelolaan. Terdapat empat package utama:

- *Main*, yang berisi kelas utama dengan game loop dan logika interaksi pemain.
- *Unit*, yang mendefinisikan objek-objek inti permainan seperti Board dan Piece.
- *Utils*, yang menyediakan kelas-kelas pembantu untuk struktur data seperti Move dan EvaluatedMove.
- *Heuristic*, yang menampung seluruh implementasi dari berbagai fungsi evaluasi yang digunakan oleh bot.

B. *Representasi Masalah dan Objek Permainan*

Untuk memungkinkan analisis komputasional, permainan Eggy Chess perlu dimodelkan ke dalam struktur data dan objek yang logis sebagai berikut:

1. Unit Permainan (Board dan Piece)

   Inti dari representasi keadaan permainan terletak pada kelas Unit.Board. Kelas ini merepresentasikan papan permainan 3x3 menggunakan sebuah array dua dimensi Piece[][]. Setiap elemen dalam array ini adalah objek dari kelas Unit.Piece, yang menyimpan dua informasi penting: type ('X' atau 'O') yang menandakan pemilik bidak, dan size (1, 2, atau 3) yang menandakan ukurannya. Selain papan, kelas Board juga memiliki sebuah HashMap bernama pieceMap yang berfungsi untuk melacak jumlah bidak yang masih tersedia bagi setiap pemain. Metode-metode penting seperti isThreeInArrow(), isEnd(), dan clone() juga diimplementasikan di sini untuk memeriksa kondisi kemenangan dan menyalin keadaan papan untuk analisis.

2. Utilitas Data (Move dan EvaluatedMove)

   Untuk merepresentasikan aksi dan hasil evaluasi, digunakan kelas-kelas pembantu di dalam package Utils. Kelas Move, yang diimplementasikan sebagai record, adalah sebuah pembawa data sederhana yang menyimpan informasi sebuah langkah: koordinat tujuan (x, y) dan objek Piece yang akan diletakkan. Sementara itu, EvaluatedMove adalah kelas yang lebih kompleks yang membungkus sebuah objek Move beserta hasil analisisnya. Ia menyimpan skor dari setiap komponen heuristik secara terpisah (positionalScore, offensiveScore, dll.) serta totalScore yang sudah digabungkan. Kelas ini mengimplementasikan antarmuka Comparable untuk memungkinkan pengurutan langkah dari yang terbaik hingga terburuk.

C. *Logika Bot dan Pengembangan Simpul*

Inti dari kecerdasan dan proses pengambilan keputusan bot terpusat pada metode handleBotTurn yang diimplementasikan di dalam kelas Main. Metode ini adalah manifestasi dari penerapan algoritma pencarian pada pohon permainan dengan kedalaman satu. Pada setiap gilirannya, bot tidak hanya memilih langkah secara acak, melainkan melalui sebuah alur proses yang terstruktur untuk mengevaluasi dan memilih langkah yang dianggap paling optimal berdasarkan analisis heuristik. Alur kerja ini dapat diuraikan ke dalam beberapa tahapan utama sebagai berikut:

1. *Pengembangan Simpul (Node Expansion)*

   Proses dimulai dengan membangkitkan semua kemungkinan langkah yang valid dari keadaan papan saat ini (currentBoard). Tahapan ini secara konseptual adalah proses node expansion pada level pertama dari pohon permainan. Bot memanggil metode currentBoard.getAllMoves() yang bertanggung jawab untuk menganalisis papan dan mengembalikan sebuah List berisi semua objek Move yang dapat dilakukan. Setiap objek Move dalam daftar ini merepresentasikan satu cabang yang mungkin diambil dari simpul (keadaan) saat ini.

2. *Simulasi Langkah dan Pengaturan Perspektif*

   Setelah daftar semua kemungkinan langkah didapatkan, program melakukan iterasi untuk setiap Move. Di dalam setiap iterasi, sebuah papan hipotetis/percobaan dibuat dengan menyalin (board.clone()) keadaan papan saat ini. Kemudian, Move yang sedang dianalisis diterapkan pada papan hipotetis tersebut. Langkah ini sangat penting karena mensimulasikan keadaan papan setelah langkah diambil. Sebelum dievaluasi lebih lanjut, perspektif pada papan hipotetis ini diatur secara eksplisit (playerType diatur ke 'X' dan enemyType ke 'O') untuk memastikan bahwa fungsi-fungsi heuristik akan menilai papan dari sudut pandang bot.

3. *Evaluasi Heuristik Individual*

   Papan hipotetis yang telah dibuat kemudian menjadi subjek evaluasi. Setiap fungsi heuristik yang telah dirancang yaitu PositionalHeuristic, OffensiveHeuristic, DefensiveHeuristic, dan PieceAdvantageHeuristic dipanggil untuk menghitung skornya masing-masing. Proses ini mengubah keadaan papan yang kompleks menjadi nilai numerik yang dapat dibandingkan. Setiap skor merepresentasikan kualitas papan/keuntungan dari satu sudut pandang strategis yang spesifik, seperti keunggulan posisi, potensi serangan, tingkat ancaman dari lawan, atau keuntungan material.

4. *Agregasi Skor dan Penyimpanan Hasil*

   Setelah semua skor individual didapatkan, skor-skor tersebut digabungkan menjadi satu totalScore menggunakan formula bobot yang telah ditentukan (default perbandingan tiap heuristic 1:1:1:1). Proses ini memungkinkan penyesuaian "gaya bermain" bot dengan memberikan prioritas lebih pada aspek tertentu. Informasi lengkap mengenai langkah yang disimulasikan, beserta rincian skor dari setiap komponen heuristik dan skor totalnya, kemudian disimpan sebagai sebuah objek EvaluatedMove. Objek ini lalu ditambahkan ke dalam sebuah List<EvaluatedMove> untuk menampung semua hasil evaluasi state.

5. *Pemilihan Langkah Terbaik dan Eksekusi*

   Tahap akhir dari proses pengambilan keputusan adalah pemilihan langkah. Setelah iterasi selesai dan semua kemungkinan langkah telah dievaluasi dan disimpan, List<EvaluatedMove> diurutkan secara menurun berdasarkan totalScore. Bot kemudian secara deterministik memilih langkah pertama dari daftar terurut tersebut, yaitu langkah yang menghasilkan skor heuristik total tertinggi. Langkah terbaik inilah yang akhirnya dieksekusi pada papan permainan yang sebenarnya (currentBoard). Ini merupakan siklus keberlanjutan dari proses pencarian, di mana hasil dari analisis dan evaluasi langsung ditranslasikan menjadi sebuah aksi nyata di dalam permainan.

D. *Implementasi Fungsi Heuristik*

Kecerdasan dan gaya bermain bot sepenuhnya ditentukan oleh implementasi dari setiap fungsi heuristik.

1. *PositionalHeuristic*

   Heuristik ini memberikan nilai strategis pada penempatan bidak. Nilai-nilai diberikan sebagai berikut

   - posisi pinggir (EDGE_WEIGHT = 4)
   - posisi pojok (CORNER_WEIGHT = 2)
   - posisi tengah (CENTER_WEIGHT = 1)

   Ini akan menghasilkan bot dengan preferensi untuk mengontrol area pinggir papan. Nilai ini kemudian dikalikan dengan ukuran bidak, memberikan insentif lebih untuk menempatkan bidak besar di posisi yang bernilai tinggi. Alasan mengapa *Edge* > *Corner* > *Center* dikarenakan fungsi ini sangat ditujukan ketika papan masih kosong di mana untuk bermain secara aman terlebih dahulu. Sedangkan posisi *Center* dan *Corner* secara tidak langsung akan mendapat nilai yang besar pada *OffensiveHeuristic* dan *DeffensiveHeuristic* yang akan

dijelaskan selanjutnya.

2. *PieceAdvantageHeuristic*

Heuristik ini mengukur "keunggulan material" dengan menghitung total nilai dari bidak yang tersisa. Bobot untuk tiap bidak dinilai sebagai berikut:

- Besar (ukuran 3) = 9
- Sedang (ukuran 2) = 4
- Kecil (ukuran 1) = 3

Skor akhir adalah selisih antara total nilai bidak pemain dan lawan, mendorong bot untuk menggunakan bidak kecil terlebih dahulu dan menyimpan bidak besarnya untuk di kasus-kasus penting seperti memblokir atau menyerang.

3. *OffensiveHeuristic*

Heuristik ini dirancang untuk mengukur potensi serangan. Heuristik ini memiliki dua mekanisme.

- Secara statis mengevaluasi papan dan memberikan skor tinggi (+10000) untuk setiap ancaman bisak dua-berjejer yang sudah ada. “dua-berjejer” di sini maksudnya adalah ketika salah satu kolom itu kosong atau merupakan bidak yang sudah diisi lawan dan berukuran lebih kecil dari bidak terbesar yang kita punya.
- Melakukan simulasi satu langkah ke depan untuk mendeteksi potensi *fork* (satu langkah yang menciptakan dua ancaman sekaligus). Jika sebuah langkah dapat menciptakan *fork*, posisi tersebut akan mendapat bonus skor yang besar (+5000), membuat bot secara proaktif mencari manuver kemenangan.

Heuristik ini tidak diaktifkan ketika mendapat giliran pertama atau papan masih kosong. Jika tidak, maka bot otomatis akan mengambil posisi tengah di mana terdapat 4 percabangan serangan, tetapi tidak aman untuk awal permainan.

4. *DefensiveHeuristic*

Heuristik ini adalah cerminan dari OffensiveHeuristic. Ia memberikan penalti yang sangat besar (-100000 untuk satu ancaman) jika lawan sudah memiliki ancaman kemenangan di papan (karena mau tidak mau harus blokir serangan itu atau menimpa bidak lawan yang memungkinkan). Selain itu, heuristik ini mampu melihat satu langkah ke depan untuk mengantisipasi fork dari lawan. Jika bot mendeteksi bahwa lawan dapat menciptakan fork di giliran berikutnya, posisi tersebut akan dikenai penalti masif (-20000), memaksa bot untuk memprioritaskan langkah blokir di atas segalanya. Sama seperti *OffensiveHeuristic*, heuristik ini dinon-aktifkan di awal giliran agar lebih berfokus ke *positional* dan *pieceAdvantage heuristic*.

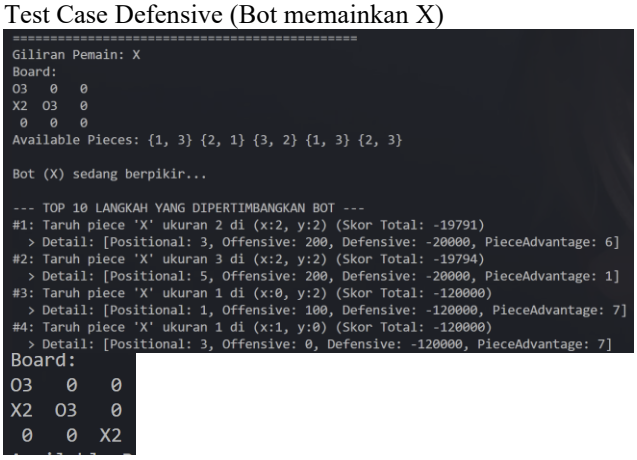
E. Batasan

Meskipun fungsional, implementasi ini memiliki beberapa batasan yang penting untuk diketahui

- Kedalaman Pencarian Dangkal: Bot hanya melakukan pencarian satu langkah ke depan (1-ply search). Tidak dapat merencanakan strategi multi-langkah atau mengantisipasi jebakan yang membutuhkan lebih dari satu langkah untuk dieksekusi.
- Sistem Non-Pembelajar: Kecerdasan bot sepenuhnya statis dan berbasis aturan yang telah diprogram secara manual (*hard-coded*). Bot tidak memiliki kemampuan untuk belajar dari permainan sebelumnya atau beradaptasi dengan gaya bermain lawan. Efektivitasnya bergantung sepenuhnya pada kualitas desain heuristik dan bobot yang ditentukan oleh pengembang.
- Potensi Eksploitasi: Logikanya deterministic sehingga jika pemain manusia berhasil menemukan kelemahan atau bias dalam fungsi heuristik (misalnya, preferensi bot yang tidak biasa terhadap posisi pinggir), kelemahan tersebut dapat dieksploitasi secara konsisten.
- Kompleksitas Komputasi: Meskipun pencarian hanya satu langkah, proses untuk membangkitkan dan mengevaluasi setiap kemungkinan langkah pada setiap giliran dapat menjadi intensif secara komputasi, terutama di awal permainan saat pilihan langkah masih sangat banyak.

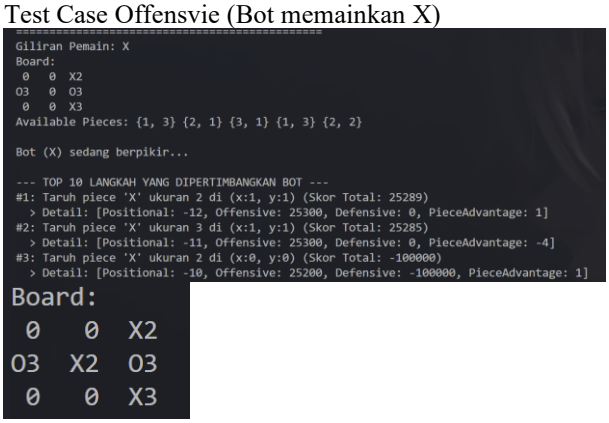
IV. TESTING DAN ANALYSIS

1. Test Case Defensive (Bot memainkan X)



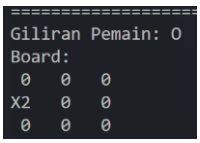
Bot berhasil melakukan blocking dengan bidak X berukuran 2 di mana tidak ada yang bisa menimpa karena bidak O berukuran 3 sudah habis.

2. Test Case Offensvie (Bot memainkan X)



Bot berhasil melakukan fork di mana bidak O berukuran 3 sebelumnya mencoba untuk mem-blokir kolom ke-3 untuk mencapai segaris.

3. Test Case Positional (Bot memainkan X)



Posisi sisi dengan bidak berukruan 2 merupakan posisi dengan kemungkinan menang terbanyak ketika di mainkan oleh giliran pertama.

V. Kesimpulan

Implementasi agen kecerdasan buatan untuk permainan Eggy Chess telah berhasil diselesaikan, membuktikan bahwa pendekatan berbasis heuristik sangat efektif untuk menciptakan lawan komputer yang kompeten. Hasil pengujian menunjukkan bahwa bot yang dikembangkan, meskipun hanya melakukan pencarian satu langkah ke depan, mampu menunjukkan perilaku taktis yang cerdas. Dengan pembobotan heuristik yang sederhana (rasio 1:1:1:1), bot secara konsisten dapat melakukan manuver kunci seperti serangan fork untuk menciptakan dua ancaman sekaligus, serta melakukan langkah defensif prioritas tinggi untuk memblokir potensi kemenangan pemain manusia.

Efektivitas bot lebih terasa ketika berperan sebagai pemain 'X' yang melangkah pertama, di mana ia terbukti sulit untuk dikalahkan (ketika pemain belum mengetahui cara eksploitasi bot). Lebih dari itu, performa optimal bot ini berhasil mengungkapkan karakteristik penting dari Eggy Chess itu sendiri: **Permainan ini cenderung tidak seimbang** (unbalanced) dan memberikan keuntungan signifikan bagi pemain pertama. Ditemukan adanya strategi kemenangan yang dapat dipaksakan (forced-win) dengan mengawali permainan menggunakan bidak berukuran 2 pada salah satu petak sisi (edge). Bot yang dikembangkan bisa berfungsi sebagai alat yang memvalidasi dan mengekspos adanya strategi dominan dalam desain permainan tersebut.

Namun, optimalisasi program masih perlu ditingkatkan mengingat adanya batasan-batasan dalam implementasi saat ini. Keterbatasan utama adalah kedalaman pencarian yang dangkal, tidak mampu merencanakan strategi jangka panjang. Untuk pengembangan di masa depan, sangat disarankan untuk mengimplementasikan algoritma Minimax dengan kedalaman multi-langkah yang dioptimalkan menggunakan Alpha-Beta Pruning. Selain itu, untuk mengatasi sifat permainan yang deterministik dan dapat diprediksi, penambahan elemen stokastik seperti memberikan kemungkinan bagi bot untuk memilih langkah terbaik kedua atau ketiga dapat membuat bot menjadi lawan tanding yang lebih dinamis dan menarik.

VI. Ucapan Terima Kasih

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat dan kelancaran yang diberikan sehingga penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga menyampaikan terima kasih kepada Pak Rinaldi Munir selaku dosen pengampu mata kuliah Strategi Algoritma 2025 Kelas 02, atas ilmu dan bimbingan yang telah diberikan selama perkuliahan berlangsung. Ilmu yang beliau sampaikan sangat membantu penulis dalam memahami materi kuliah serta menyelesaikan tugas makalah ini.

Tak lupa, penulis juga mengucapkan terima kasih kepada kedua orang tua yang telah memberikan dukungan selama proses perkuliahan. Penulis berharap makalah ini dapat memberikan kontribusi positif dalam memahami lebih dalam materi yang telah dipelajari.

References

[1] R. Munir, "Strategi Algotitma – Tahun Ajaran 2024/2025," Institut Teknologi Bandung (ITB). [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/stima24-25.htm. [Diakses: Jan. 23, 2025].

[2] Sangeeta, V. Rai, N. Pathak, and N. Sharma, "Search Strategies: Informed Search Strategies," in *Artificial Intelligence and Their Applications*, IIP Series, 2024. e-ISBN: 978-93-6252-717-2. [Online]. Available: https://iipseries.org/assets/docupload/rsl2024AF233C7BF02A178.pdf [Accessed: June. 24, 2025].

[3] A. A. Elnaggar, M. Abdel Aziem, M. Gadallah, and H. El-Deeb, "A Comparative Study of Game Tree Searching Methods," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 5, no. 5, 2014. [Online]. Available: https://www.researchgate.net/publication/262672371_A_Comparative_Study_of_Game_Tree_Searching_Methods [Accessed: Jan. 24, 2025].

[4] T. Eiter and T. Krennwallner, "Heuristic Evaluation Functions for General Game Playing," *KI - Künstliche Intelligenz*, vol. 25, no. 1, pp. 73–74, Mar. 2011, doi: 10.1007/s13218-010-0074-7. [Online]. Available: https://www.researchgate.net/publication/226059163_Heuristic_Evaluation_Functions_for_General_Game_Playing. [Accessed: Jan. 24, 2025].

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, 24 Juni 2025

Muhammad Zahran Ramadhan Ardiana
13523104