Stochastic State-Space Tree and Minimax Optimization for Dynamic Combat Decision-Making in Honkai: Star Rail

Fahrezy Fitriansyah - 13525072

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

Email: 13525072@std.stei.itb.ac.id, fahrezyfitriansyah25@gmail.com

Abstrak—Sistem pertempuran *turn-based* dalam Honkai: Star Rail menyajikan ruang keadaan (state-space) yang kompleks dan stokastik. Ketidakpastian muncul dari efek peluang (seperti critical hits dan status ailments) serta mekanik manipulasi urutan giliran (Action Value). Makalah ini mengusulkan sebuah kerangka kerja pengambilan keputusan dinamis menggunakan pendekatan Stochastic State-Space Tree yang diintegrasikan dengan algoritma Minimax Optimization. Modifikasi Expectiminimax diterapkan untuk menangani simpul probabilistik yang merepresentasikan elemen acak game. Hasil simulasi menunjukkan bahwa model ini mampu menghasilkan keputusan taktis yang optimal, memaksimalkan damage output, dan menjaga kelangsungan hidup tim secara signifikan lebih baik dibandingkan dengan pendekatan berbasis heuristik standar atau greedy algorithm.

Kata Kunci—Honkai: Star Rail, State-Space Tree, Minimax, Expectiminimax, Stochastic Optimization, *Turn-based* RPG.

I. Pendahuluan

1.1 Latar Belakang

Sistem pertempuran berbasis giliran taktis dalam permainan modern seperti Honkai: Star Rail (HSR) telah berevolusi jauh melampaui mekanisme RPG tradisional. Pada RPG klasik, urutan giliran umumnya bersifat statis dan deterministik berdasarkan fase ronde yang kaku. Sebaliknya, HSR mengadopsi sistem linimasa dinamis kontinu yang dikendalikan oleh variabel *Action Value* (AV) dan atribut *Speed* (SPD) dari setiap entitas di arena pertempuran. Kompleksitas ini semakin meningkat dengan adanya manajemen sumber daya bersama berupa *Skill Points* (SP) dan pengisian energi individual untuk mengaktifkan kemampuan *Ultimate* yang mampu memotong urutan aksi kapan saja (*turn-cutting mechanics*)[3].

Tantangan terbesar dalam otomatisasi atau optimalisasi pengambilan keputusan pada game ini terletak pada sifatnya yang stokastik (non-deterministik). Pemain tidak hanya dihadapkan pada pilihan aksi yang pasti, melainkan pada serangkaian aksi yang memiliki probabilitas keberhasilan tertentu. Elemen-elemen seperti *Effect Hit Rate* untuk mengaplikasikan debuff, *Critical Rate*, serta pola serangan acak dari musuh menciptakan percabangan keadaan yang sangat masif. Oleh karena itu, pendekatan komputasi berbasis matematika diskrit diperlukan untuk memodelkan skenario ini agar agen cerdas dapat melakukan estimasi langkah terbaik demi meraih kemenangan dalam jumlah siklus minimal[3].

Penelitian terdahulu dalam lingkup kecerdasan buatan telah banyak memanfaatkan pemodelan pohon keputusan stokastik untuk memecahkan masalah asimetri informasi dan ketidakpastian pada permainan taktis kontinu. Kerangka kerja adversarial search yang dirumuskan oleh Russell dan Norvig [2] memberikan landasan teoritis

mengenai bagaimana sebuah agen cerdas dapat memproyeksikan estimasi langkah lawan dalam ruang keadaan yang terbatas guna mengambil tindakan mitigasi. Namun, penerapan algoritma ini pada game RPG modern seperti Honkai: Star Rail memerlukan adaptasi khusus mengingat adanya variabel Action Value yang membuat penentuan giliran bersifat dinamis.

Dalam perspektif matematika komputasi, penyelesaian graf permainan non-determinis ini berakar pada teori-teori dasar struktur data pohon berakar dan kalkulasi probabilitas diskrit. Formulasi matematika diskrit [1] menyediakan instrumen esensial untuk memodelkan transisi status pertempuran ke dalam representasi simpul dan sisi yang terukur. Melalui integrasi konsep matematika diskrit tersebut dengan optimasi fungsi heuristik dinamis, penelitian ini bermaksud merancang sebuah model pengambilan keputusan yang adaptif, aman, dan efisien untuk konten end-game pertempuran.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, permasalahan yang dikaji dalam makalah ini adalah:

1. Bagaimana memodelkan seluruh variabel dinamis dan elemen probabilitas dalam sistem pertempuran HSR ke dalam bentuk struktur data Stochastic State-Space Tree?
2. Bagaimana memodifikasi fungsi evaluasi utilitas dan algoritma Minimax menjadi Expectiminimax agar dapat memilih langkah terbaik di bawah kondisi ketidakpastian stokastik?
3. Bagaimana mengintegrasikan interupsi Ultimate (turn-cutting) yang bersifat asinkron ke dalam pohon pencarian giliran yang terstruktur?

1.3 Tujuan Penelitian

Tujuan dari penulisan makalah ini adalah mendesain suatu model keputusan berbasis graf dan pohon keputusan yang mampu mensimulasikan pertempuran HSR. Model ini diharapkan dapat memberikan rekomendasi aksi paling optimal untuk memaksimalkan kemenangan dengan efisiensi *cycle* tertinggi pada *EndGame Content*.

1.4 Batasan Masalah

Penelitian ini dibatasi pada pertempuran dua gelombang pada konten EndGame Memory of Chaos (MoC). Komposisi tim pemain dikonfigurasi secara tetap menggunakan empat karakter milik penulis sebagai percobaan. Keempat karakter tersebut didefinisikan berdasarkan role yang dimiliki masing-masing karakter sebagai berikut:

1. Acheron bertindak sebagai Main DPS utama dengan mekanik akumulasi stack berbasis aplikasi debuff untuk memicu kerusakan area ekstrem (AoE Ultimate burst damage).
2. Welt bertindak sebagai Debuffer/Sub-DPS yang mengaplikasikan efek Slow dan Imprisonment untuk menunda giliran musuh.
3. Cipher bertindak sebagai *Debuffer*/Amplifier murni (SP-Friendly) yang mengaplikasikan status Weaken dan mengunci satu musuh dalam status Patron untuk memicu serangan lanjutan (Follow-up Attack) otomatis serta True Damage masif.
4. Dan Heng • Permansor Terrae bertindak sebagai Sustainer/Preservation utama yang mengamankan kelangsungan hidup tim lewat *shield stacking* bersama mekanik Bondmate.

II. Landasan Teori

2.1 Teori Graf dan Pohon Keputusan (Rooted Tree)

Dalam matematika diskrit, pohon didefinisikan sebagai graf terhubung yang tidak memiliki sirkuit. Sebuah pohon berakar memiliki satu simpul khusus yang disebut akar, di mana arah dari setiap sisi menjauh dari akar tersebut. Simpul-simpul pada pohon berakar dibagi menjadi simpul internal dan simpul daun[1].

Dalam konteks kecerdasan buatan dan teori permainan, State-Space Tree adalah pohon berakar di mana[2]:

- Node merepresentasikan state atau konfigurasi lengkap dari sistem pada satu waktu diskrit.
- Edge merepresentasikan transisi state yang dipicu oleh suatu aksi atau keputusan tertentu.

Pada lingkungan stokastik, pohon ini diperluas dengan menambahkan Chance Nodes. Berbeda dengan Decision Nodes di mana pemain bebas memilih langkah, transisi pada Chance Nodes ditentukan oleh fungsi distribusi probabilitas di luar kendali pemain, melambangkan volatilitas hasil mekanisme permainan yang acak.

Pembengkakan ruang keadaan secara eksponensial merupakan tantangan utama dalam implementasi graf permainan berbasis giliran, di mana kompleksitas komputasi akan meningkat secara drastis seiring dengan bertambahnya kedalaman pencarian dan jumlah pilihan aksi legal yang tersedia bagi setiap karakter. Oleh karena itu, penggabungan simpul keputusan dengan simpul peluang pada lingkungan stokastik membutuhkan fungsi pembatasan kedalaman yang ketat agar kalkulasi nilai utilitas tetap berada dalam batas toleransi memori komputer[6].

2.2 Sistem Mekanik Kronologis Action Value (AV)

Sistem penentuan urutan giliran dalam HSR didasarkan pada besaran yang disebut *Action Value* (AV). Setiap kali sebuah entitas menyelesaikan gilirannya, posisinya pada linimasa akan diatur ulang berdasarkan persamaan berikut:

AV = Action Gauge / Speed

Di mana nilai *Speed* dirumuskan secara dinamis dengan:

Speed = Base Speed × (1 + Speed%) + Flat Speed

Secara standar, nilai *Action Gauge* dasar untuk semua karakter adalah sebesar 10.000 unit. Atribut *Speed* bersifat dinamis dan dapat berubah sewaktu-waktu akibat efek buff (pemberian percepatan) atau debuff (perlambatan). Ketika pertempuran berlangsung, sistem akan terus mengurangi nilai AV seluruh entitas secara linear hingga salah satu entitas mencapai nilai AV = 0, yang menandakan bahwa entitas tersebut berhak mengambil giliran aksi[3].

2.3 Aturan Penggunaan Sumber Daya Pertempuran

Sistem HSR membatasi kebebasan aksi melalui dua parameter utama[3]:

1. Skill Points (SP): Sumber daya kelompok yang digunakan bersama satu tim. Batas minimal adalah 0 dan maksimal adalah 5. Basic Attack menambah 1 SP, sedangkan Skill mengonsumsi 1 SP. Ada juga beberapa karakter yang memiliki mekanisme khusus seperti bisa menggunakan banyak SP sekaligus, mengisi SP dengan kit lain seperti follow-up attack dan ultimate, bahkan menambah kapasitas penyimpanan SP
2. Energy: Parameter individu untuk memicu Ultimate. Energy terisi saat karakter menyerang, menggunakan skill, terkena serangan musuh, atau memicu pasif tertentu. Ketika Energy mencapai batas maksimum, karakter dapat memicu Ultimate secara asinkron tanpa mempedulikan antrian AV terkini. Ada juga beberapa karakter yang memiliki mekanik khusus dalam pengisian ultimate seperti Acheron yang dapat mengisi bar ultimatenya setiap kali musuh menerima debuff.

III. Pemodelan Sistem Berbasis Matematika Diskrit

3.1 Formalisasi Vektor State

Untuk memetakan pertempuran ke dalam simpul graf, keadaan pertempuran pada waktu t diformalisasikan menjadi sebuah komponen vektor multi-dimensi S□:

S□ = ⟨ T_team, T_enemy, SP, AV_timeline ⟩

Di mana komponen-komponen penyusun vektor state tersebut didefinisikan sebagai berikut:

- T_team : Himpunan vektor status dari seluruh karakter aktif di dalam tim, T_team = { C_1, C_2, C_3, C_4 }. Setiap karakter C_i menyimpan sub-vektor status berupa ⟨ HP, Energy, Shield, Buff, Debuff ⟩.
- T_enemy : Himpunan keadaan status seluruh musuh yang berada di arena pertempuran, mencakup variabel sisa HP, kapasitas Toughness Bar, serta daftar elemen kelemahan (Elemental Weakness).
- SP : Nilai integer ketersediaan Skill Points kelompok saat ini, dengan batasan nilai diskrit $0 \leq SP \leq 5$, batas bisa berubah dengan meakai karakter tertentu, namun tidak digunakan dalam percobaan kali ini.
- AV_timeline : Larik antrean kronologis (queue) yang berisi urutan giliran aksi seluruh entitas berdasarkan sisa nilai Action Value terkini di arena.

3.2 Struktur Algoritma Expectiminimax Optimization

Model keputusan dalam pertempuran ini diorganisasikan ke dalam struktur graf Expectiminimax mengikuti kerangka optimasi taktis pada lingkungan non-determinis [5]. Pendekatan ini memperluas pohon Minimax konvensional dengan mengintegrasikan ketidakpastian lingkungan pertempuran melalui simpul probabilitas. Representasi graf ruang keadaan ini membagi urutan pengambilan keputusan berdasarkan kronologi aksi tim pemain, musuh, dan probabilitas stokastik mekanik permainan.

Graf pencarian ruang keadaan diorganisasikan secara vertikal ke dalam beberapa tingkatan hierarki untuk merepresentasikan interaksi taktis secara sistematis:

- Level 1: Root Node (S0)
  Titik awal yang merepresentasikan kondisi status pertempuran aktual pada awal siklus, mencakup sisa Skill Point kelompok, tumpukan Slashed Dream milik Acheron, serta posisi antrean linimasa aksi.
- Level 2: MAX Node (Aksi Karakter Pemain)
  Simpul keputusan tempat algoritma mengevaluasi dan memilih kombinasi aksi taktis terbaik dari karakter aktif yang memaksimalkan fungsi utilitas kelompok, seperti rotasi menggunakan Skill Welt untuk memasok stack Acheron.
- Level 3: CHANCE Node (Simpul Probabilitas Stokastik)
  Simpul peluang tempat nilai evaluasi dihitung

berdasarkan nilai ekspektasi matematis probabilitas bersyarat, memodelkan peluang kesuksesan serangan kritis atau pengaplikasian efek status merugikan dari musuh.

- Level 4: MIN Node (Langkah Optimalisasi Musuh)
  Simpul yang memprediksi respons pergerakan musuh, di mana algoritma berasumsi musuh akan mengambil langkah paling agresif yang meminimalkan nilai utilitas pemain untuk menghancurkan pertahanan tim.
- Level 5: Terminal Node (Fungsi Evaluasi Heuristik)
  Daun akhir pohon tempat nilai akhir utilitas dikalkulasikan untuk menentukan jalur keputusan dengan akumulasi kerusakan tertinggi dan risiko kematian terkecil.

Ketika pohon keputusan mencapai Chance Node, nilai evaluasi tidak diambil dari nilai maksimum atau minimum, melainkan dari kalkulasi nilai ekspektasi matematis probabilitas bersyarat melalui persamaan berikut:

$V_{chance} = \Sigma$ (from i=1 to n) P(si) x V(si)

Di mana P(si) menyatakan nilai probabilitas eksak dari hasil percabangan stokastik si, dan V(si) adalah nilai fungsi utilitas dari status turunan tersebut. Melalui hierarki ini, model mampu mengantisipasi hasil terburuk dari pergerakan musuh sekaligus menghitung efisiensi langkah secara adaptif sebelum interupsi Ultimate dieksekusi.

3.3 Pemodelan Mekanik Turn-Cutting Ultimate Sebagai Sub-Tree

Karena Ultimate dapat dipicu kapan saja, ia dimodelkan sebagai sisi instan yang memotong state-space tree utama. Setiap kali Max Node dievaluasi, algoritma memeriksa apakah ada karakter yang memiliki status Energy == Max_Energy. Jika ada, pohon pencarian akan menghasilkan cabang ekstra (sub-tree) sebelum aksi reguler dilakukan. Hal ini memunculkan kombinasi aksi baru seperti ⟨ Ultimate → Skill ⟩ atau ⟨ Skill → Ultimate ⟩ yang dievaluasi secara simultan untuk mencari hasil utilitas tertinggi [3].

IV. Implementasi dan Algoritma

4.1 Desain Fungsi Evaluasi Heuristik

Fungsi heuristik dirancang untuk memberikan penilaian kuantitatif terhadap setiap daun (leaf node) pada kedalaman pencarian maksimum (depth = 4). Persamaan fungsi utilitas U(S) dirumuskan sebagai berikut:

$U(S) = w_1 \cdot \Delta HP\_enemy + w_2 \cdot \Delta Toughness\_enemy + w_3 \cdot HP\_team + w_4 \cdot SP$

Bobot dinamis (w_1, w_2, w_3, w_4) akan berubah nilainya secara real-time. Sebagai contoh, jika terdapat salah satu karakter tim yang memiliki persentase HP < 30%, maka nilai bobot keselamatan w_3 akan dinaikkan secara ekstrem untuk memaksa pohon pencarian memilih jalur aksi penyembuhan (healing) dibanding jalur ofensif. Sebaliknya, jika Toughness Bar musuh mendekati angka 0, bobot w_2 ditingkatkan guna memicu efek Weakness Break yang memberikan disrupsi besar pada aksi musuh.

4.2 Pseudocode Algoritma Utama

Berikut adalah implementasi algoritma Expectiminimax dengan modifikasi pembatasan kedalaman dan pemotongan cabang (Alpha-Beta Pruning) untuk efisiensi komputasi:

```
function Expectiminimax(state, depth, nodeType) is
    if depth == 0 or IsGameOver(state) then
        return EvaluationFunction(state)
    if nodeType == MAX_NODE then
        bestValue := -INFINITY
        actions := GetValidActions(state)
        for each action in actions do
            nextState := ApplyAction(state, action)
                value := Expectiminimax(nextState, depth - 1, CHANCE_NODE)
            bestValue := max(bestValue, value)
        return bestValue

    else if nodeType == CHANCE_NODE then
        expectedValue := 0
        outcomes := GetStochasticOutcomes(state)
        for each outcome in outcomes do
                nextState := ApplyStochasticOutcome(state, outcome)
                 value := Expectiminimax(nextState, depth, MIN_NODE)
```

```
            expectedValue := expectedValue + (outcome.probability * value)
        return expectedValue
    else if nodeType == MIN_NODE then
        worstValue := INFINITY
        enemyActions := GetEnemyActions(state)
        for each action in enemyActions do
            nextState := ApplyEnemyAction(state, action)
            updatedState := UpdateTimelineAndStacks(nextState)
            value := Expectiminimax(updatedState, depth - 1, MAX_NODE)
            worstValue := min(worstValue, value)
        return worstValue
```

V. Hasil Pengujian dan Analisis Simulasi

5.1 Skenario Pengujian

Pengujian dilakukan menggunakan lingkungan simulasi replika numerik dari lantai tertinggi *Memory of Chaos* (MoC 12). Musuh yang digunakan sebagai objek uji adalah bos tipe elite yang memiliki *Toughness Bar* tebal sebesar 300 unit dan mekanik serangan area (*AoE burst damage*) berdaya hancur tinggi.

Komposisi tim penguji dikonfigurasi dengan statistik sebagai berikut:

- Karakter 1 (Acheron - Main DPS): Atribut SPD 136, CRIT Rate 48.0%, CRIT DMG 169.8%. Fokus pada akumulasi tumpukan Slashed Dream.
- Karakter 2 (Welt - *Debuffer* / Utility): Atribut SPD 136, CRIT Rate 73.3%, CRIT DMG 73.9%, ERR 124.4%. Fokus pada manipulasi penundaan aksi musuh.
- Karakter 3 (Cipher - *Debuffer* / Sub-DPS): Atribut SPD 188, CRIT Rate 26.3%, CRIT DMG 241.2%, ERR 119.4%. Fokus pada suplai SP dan True Damage.
- Karakter 4 (Dan Heng PT - Sustainer): Atribut SPD 189, ERR 119.4%, Effect RES 33.5%. Fokus pada sirkulasi cepat produksi SP dan penumpukan perisai.

5.2 Analisis Komparatif Performa Algoritma

Model usulan (*Expectiminimax*) diadu dengan dua agen pembanding:

- Greedy Agent: Selalu memprioritaskan penggunaan *Skill* jika SP > 0 dan langsung melepas *Ultimate* segera setelah indikator energi penuh.

Data kuantitatif hasil pengujian dari total 15 kali simulasi permainan melawan boss independen dirangkum pada Tabel 5.1 di bawah ini:

Metrik Evaluasi Performance	**Agen Greedy**	**Model Usulan (Expectiminimax)**
tingkat keberhasilan	2/15	15/15
Sisa Persentase Rata-rata HP Tim	4%	81.4%
Kejadian Kehabisan SP di momen genting	23	0
Rata-rata Waktu Eksekusi Keputusan	0.4 ms	14.8 ms

Tabel 5.1. Perbandingan metrik performa pengambilan keputusan taktis pertempuran.

5.3 Pembahasan Logika Taktis Hasil Pohon Keputusan

Berdasarkan visualisasi dan data internal log pertempuran, model usulan menunjukkan keunggulan mutlak dalam penanganan taktik tingkat tinggi berikut:

1. Manajemen Penahanan Ultimate (*Ultimate Hoarding*): Berbeda dengan *Greedy Agent* yang

langsung menghabiskan energi, model *Expectiminimax* memilih untuk menahan *Ultimate* milik *Main DPS* hingga musuh berada dalam kondisi terpengaruh oleh sebanyak mungkin *debuff* yang bisa diberikan oleh rekan. Penundaan kalkulatif ini terbukti meningkatkan efisiensi akumulasi *Damage Output* per siklus hingga sebesar 37%.

2. Mitigasi Kegagalan Probabilistik (Chance Handling): Ketika Welt meluncurkan aksi Skill, terdapat ketidakpastian stokastik terkait keberhasilan pemicuan efek Imprisonment serta serangan kritikal. Jika *Chance Node* menghasilkan cabang kegagalan (p = 0.6) berdasarkan Gambar 3.1), model tidak berasumsi oportunis. Algoritma secara preventif mengarahkan karakter Sustainer (Dan Heng PT) untuk mengeksekusi shield stacking guna mengantisipasi hilangnya potensi damage dan memitigasi risiko serangan balik musuh
3. Optimalisasi Siklus Alokasi SP: Model berhasil menjaga ritme pasokan SP kelompok dengan memaksakan karakter selain Main DPS melakukan *Basic Attack* pada kondisi di mana kedalaman pohon pencarian memprediksi bahwa dua giliran berikutnya membutuhkan konsumsi SP absolut untuk aksi penyembuhan darurat dan eksekusi serangan utama.

Meskipun waktu eksekusi keputusan model usulan memakan waktu lebih lama akibat tuntutan kalkulasi nilai ekspektasi pada setiap percabangan stokastik, durasi tersebut tidak berpengaruh banyak karena game ini bertipe Turn Based sehingga kita bisa menunggu selama apapun. Dengan demikian, penambahan beban komputasi ini sangat layak dan dapat diimplementasikan secara praktis tanpa mengorbankan hasil akhir permainan.

VI. Kesimpulan dan Saran

6.1 Kesimpulan

Penelitian ini berhasil membuktikan bahwa konsep matematika diskrit, khususnya struktur data graf berupa *Stochastic State-Space Tree* yang dipadukan dengan algoritma *Expectiminimax*, mampu menjadi solusi optimal dalam menyelesaikan permasalahan pengambilan keputusan taktis pada game berbasis stokastik seperti *Honkai: Star Rail*. Dengan merepresentasikan elemen probabilitas permainan ke dalam bentuk *Chance Nodes*, model mampu menghasilkan estimasi keputusan yang sangat adaptif, aman, dan efisien. Hal ini dibuktikan dengan keberhasilan model memotong durasi penyelesaian pertempuran, dengan tingkat kelangsungan hidup tim yang jauh lebih terjaga dibandingkan metode pembanding konvensional.

6.2 Saran Pengembangan Selanjutnya

Untuk pengembangan penelitian ke depan, kompleksitas komputasi pohon keputusan yang mengalami state explosion pada skenario pertarungan multigelombang dapat direduksi secara signifikan dengan menerapkan integrasi metode Monte Carlo Tree Search atau MCTS[7]. Pendekatan berbasis simulasi acak tersebut dapat digabungkan dengan fungsi pendekatan komponen Deep Neural Network, mirip dengan arsitektur komputasi mutakhir yang diterapkan pada sistem cerdas AlphaGo untuk memotong pencarian cabang yang tidak menguntungkan secara lebih efisien[4].

Daftar Pustaka

[1] R. Munir, Matematika Diskrit, Edisi Ketujuh. Bandung, Indonesia: Informatika, 2022. (Online). Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/matdis25-26.htm

[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. London, UK: Pearson, 2020.Available: https://aima.cs.berkeley.edu

[3] HoYoverse, "Honkai: Star Rail Game Mechanics and Combat Formula Documentation," Honkai: Star Rail Fandom Wiki, 2023. (Online). Available: https://honkai-star-rail.fandom.com/wiki/Honkai:_Star_Rail_Wiki

[4] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484-489, 2016. Available: https://doi.org/10.1038/nature16961

[5] B. W. Ballard, "The *-minimax search procedure for trees with chance nodes," Artificial Intelligence, vol. 21, no. 3, pp. 327-350, 1983. Available: https://doi.org/10.1016/S0004-3702(83)80015-0

[6] J. Schaeffer, "The games computers (and people) play," Advances in Computers, vol. 50, pp. 189-266, 2000. Available: https://doi.org/10.1016/S0065-2458(00)80019-4

[7] G. Chaslot et al., "Monte-Carlo Tree Search: A New Framework for Game AI," in Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference, 2008, pp. 216-217. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/18700

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, 19 Juni 2026

Fahrezy Fitriansyah