JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i2.9538

ANALISIS KOMPARASI EFISIENSI ALGORITMA LAYER-BY-LAYER DAN CFOP PADA PENYELESAIAN RUBIK'S CUBE MENGGUNAKAN PENDEKATAN DATA DRIVEN

Aditya Pratama Putra[1], Apriade Voutama[2]

[1,2]Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang; Jl. H.S. Ronggowaluyo, Telukjambe Timur, Karawang, 41361; Telp: (0267) 641176

Keywords:
Rubik's Cube; CFOP; Layer-by-Layer; Algorithm Efficiency; Data-Driven.

Corespondent Email:
2310631250002@student.unsika.ac.id

Abstrak. Penelitian ini bertujuan menganalisis efisiensi algoritma Layer-by-Layer (LBL) dan CFOP (Cross, F2L, OLL, PLL) dalam penyelesaian Rubik's Cube 3x3x3 menggunakan pendekatan *data-driven*. Pemilihan topik didasari oleh kebutuhan pembuktian kuantitatif atas efisiensi pergerakan kedua metode heuristik yang sering diperdebatkan dalam komunitas speedcubing dan pengembangan robotika. Mayoritas evaluasi masa lalu berfokus pada kecepatan motorik fisik manusia, sehingga mengabaikan kompleksitas algoritmik murni. Metode penelitian menggunakan simulasi komputasional berbasis skrip pada 1.000 pengacakan standar WCA (World Cube Association) untuk menghitung metrik instruksi pasti dalam Half Turn Metric (HTM). Hasil komputasi menunjukkan bahwa CFOP memiliki penyelesaian rata-rata 55,8 langkah, sedangkan LBL mencatat rata-rata 85,3 langkah. Kesimpulannya, CFOP terbukti secara signifikan memangkas ±34,5% jumlah gerakan dibandingkan LBL melalui reduksi redundansi pada proses penyelesaian lapisan tengah dan atas, menjadikannya model algoritma yang lebih optimal untuk diimplementasikan pada kecerdasan buatan.

Abstract. *This study aims to analyze the efficiency of the Layer-by-Layer (LBL) and CFOP (Cross, F2L, OLL, PLL) algorithms in solving the 3x3x3 Rubik's Cube using a data-driven approach. The topic selection is based on the need for quantitative proof of move efficiency for these two heuristic methods, often debated in speedcubing and robotics development communities. Past evaluations largely focused on human physical motor speed, ignoring pure algorithmic complexity. The research method uses computational simulation on 1,000 standard WCA (World Cube Association) scrambles to calculate exact instruction metrics in Half Turn Metric (HTM). Computational results showed CFOP averaged 55.8 moves, while LBL averaged 85.3 moves. In conclusion, CFOP is proven to significantly reduce move counts by ±34.5% compared to LBL through redundancy reduction in solving the middle and top layers, making it a more optimal algorithmic model for artificial intelligence implementation.*

1. PENDAHULUAN

Rubik's Cube 3x3x3 bukan sekadar permainan teka-teki, melainkan sebuah model matematis kompleks yang merepresentasikan masalah permutasi dalam ilmu komputer dan teori grup diskrit [1]. Dengan total kemungkinan permutasi mencapai 4.3 x 10^{19} kondisi state, pencarian jalur penyelesaian yang paling optimal (God's Algorithm) menjadi salah satu tolak ukur dalam pengujian algoritma pencarian kecerdasan buatan (Artificial Intelligence) [2]. Di luar penyelesaian optimal menggunakan komputasi berat, terdapat algoritma heuristik berbasis aturan (rule-based) yang dirancang untuk diselesaikan dalam waktu polinomial, di mana dua metode yang paling menonjol adalah algoritma Layer-by-Layer (LBL) dan CFOP [3]. State of the art dari penelitian mengenai Rubik's Cube saat ini banyak berfokus pada penemuan batasan teoretis, seperti pembuktian "God's Number" yang menyatakan bahwa konfigurasi Rubik apa pun dapat diselesaikan maksimal dalam 20 pergerakan [4], atau implementasi Deep Reinforcement Learning untuk robotika [5].

Namun, terdapat gap analysis dalam literatur akademis saat ini: perbandingan antara metode heuristik LBL dan CFOP mayoritas dilakukan melalui eksperimen fisik (kecepatan tangan manusia), bukan melalui pengukuran metrik instruksi algoritmik secara terisolasi [6]. Pendekatan fisik ini menghasilkan bias yang dipengaruhi oleh ergonomi, look-ahead spasial manusia, dan kecepatan motorik.Oleh karena itu, tujuan penelitian ini adalah memformulasikan pendekatan *data-driven* untuk mengukur dan mengkomparasi tingkat efisiensi algoritma LBL dan CFOP secara murni komputasional. Pertanyaan penelitian yang diajukan adalah: Seberapa besar selisih move count (jumlah pergerakan instruksi) antara LBL dan CFOP apabila dieksekusi secara sistematis tanpa campur tangan variabel fisik manusia? Penelitian ini diharapkan memberikan wawasan mengenai trade-off antara kompleksitas ruang (jumlah memori/algoritma yang harus disimpan) dan kompleksitas waktu (jumlah langkah eksekusi) dalam penyelesaian masalah ruang keadaan (*state-space problem*).

2. TINJAUAN PUSTAKA

Evaluasi efisiensi algoritma dalam studi ini bersandar pada Half Turn Metric (HTM), yakni standar komputasi operasional di mana setiap rotasi sisi kubus dihitung sebagai satu unit langkah tunggal [7]. Pengukuran menggunakan HTM melalui pendekatan simulasi *data-driven* memungkinkan kalkulasi kompleksitas pergerakan secara presisi [8]. Metrik ini sangat krusial untuk mengeliminasi bias variabel mekanik manusia dalam perhitungan jumlah instruksi pergerakan.

Penelitian ini membedah komparasi dua metode heuristik. Pertama, algoritma Layer-by-Layer (LBL) yang beroperasi menggunakan pendekatan dekomposisi sekuensial lapis demi lapis. LBL memiliki kompleksitas memori yang sangat efisien karena hanya bertumpu pada sedikit urutan basis instruksi, namun menghasilkan redundansi transisi graf yang tinggi akibat penyelesaian per lapisan yang saling terisolasi [9].

Kedua, algoritma CFOP (Cross, F2L, OLL, PLL) sebagai bentuk optimasi tingkat lanjut dari LBL yang mengintegrasikan penyelesaian sudut dan tepi secara simultan pada tahap First Two Layers (F2L) [10]. Pendekatan optimasi kombinatorial ini memangkas redundansi operasional secara drastis [11]. Namun, hal ini memberikan konsekuensi trade-off berupa tuntutan alokasi ruang memori yang jauh lebih masif karena sistem harus memuat 78 kondisi dasar *pattern matching* atau tabel pencarian [12].

3. METODE PENELITIAN

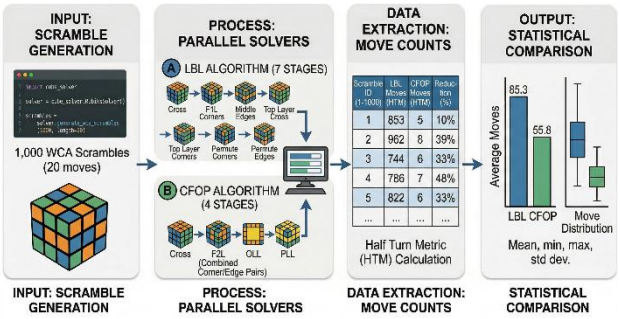
Penelitian ini menggunakan rancangan eksperimen komputasional kuantitatif dengan pendekatan *data-driven*. Fokus utama penelitian adalah menyimulasikan penyelesaian Rubik's Cube 3x3x3 menggunakan dua algoritma heuristik, yaitu *Layer-by-Layer* (LBL) dan CFOP, untuk mengevaluasi efisiensi pergerakan operasionalnya secara objektif tanpa bias kecepatan motorik manusia.

3.1. Arsitektur Simulasi

Metode simulasi dibangun dalam lingkungan *virtual* menggunakan bahasa pemrograman Python. Struktur data Rubik's Cube dimodelkan sebagai representasi array satu dimensi yang memuat 54 elemen stiker warna. Himpunan aturan pergerakan pada algoritma LBL dan CFOP dikonversi menjadi logika kondisional dan implementasi tabel pencarian (*lookup tables*) guna meresolusi transisi graf dari keadaan acak menuju keadaan terurut tunggal.

3.1.1. Gambar dan tabel

Tahap komputasi awal difokuskan pada pembangkitan 1.000 sampel deret pengacakan (scrambles) unik menggunakan generator skrip pseudo-random. Dataset pengacakan tersebut divalidasi agar mematuhi standar World Cube Association (WCA) dengan batas minimum 20 langkah pergerakan untuk mencapai level entropi maksimal. Alur kerja simulasi secara komprehensif direpresentasikan pada Gambar 1.



Gambar 1. Alur Kerja Simulasi dan Analisis Komparasi Algoritma Rubik's Cube Berbasis Data (*Data-Driven*).

3.1.2. Rumus Matematika

Teknik ekstraksi metrik menghitung cost instruksi komputasi secara absolut menggunakan Half Turn Metric (HTM). Analisis statistik dikalkulasikan dengan mencari Nilai Harapan (Expected Value) yang direpresentasikan menggunakan penomoran persamaan dan diposisikan di tengah naskah. Nilai rata-rata (E) efisiensi langkah dirumuskan sebagai:

$$E = \frac{1}{N}\sum_{i=1}^{N} X_i$$

di mana:
E = nilai rata-rata langkah,
N = jumlah sampel (1000),
Xi = jumlah langkah pada percobaan ke-i.

Di mana parameter E merepresentasikan nilai rata-rata langkah metrik operasional, N adalah total dataset (1.000 sampel), dan parameter X_i merupakan akumulasi gerakan komputasi (move count) per eksekusi penyelesaian ke–i.

3.1.3. Pengacuan Pustaka

Penyusunan aturan logika transisi graf dan pemodelan *state-space* Rubik's Cube pada mesin virtual divalidasi berdasarkan literatur keilmuan heuristik komputasi dan optimasi algoritma [13]. Penulis mengutip publikasi dalam teks yang mengikuti gaya kutipan IEEE. Setiap rujukan teori yang digunakan sebagai landasan parameter simulasi, baik untuk metode dekomposisi linier maupun optimasi kombinatorial, ditandai dengan penomoran angka di dalam kurung siku dan disusun secara berurutan pada daftar pustaka di akhir artikel [14].

3.2. Implementasi dan Spesifikasi Teknis

Simulasi dikembangkan menggunakan Python 3.11 dengan memanfaatkan struktur data array satu dimensi berukuran 54 elemen untuk merepresentasikan 54 stiker warna pada permukaan Rubik's Cube 3x3x3. Setiap elemen merepresentasikan satu stiker dengan nilai integer (0–5) yang mewakili enam warna berbeda. Perpindahan state dimodelkan sebagai fungsi permutasi deterministik yang mengubah posisi indeks array sesuai dengan aturan gerakan sisi kubus (U, D, F, B, R, L beserta variasi inversenya).

Aturan heuristik algoritma LBL diimplementasikan sebagai serangkaian kondisi if-else bertingkat yang mendeteksi posisi dan orientasi setiap kubie secara sekuensial per lapisan. Sebaliknya, sistem CFOP menggunakan pendekatan *lookup table* berbasis dictionary Python yang memuat 78 kondisi pola F2L, 57 kondisi OLL, dan 21 kondisi PLL untuk meresolusi state secara langsung tanpa iterasi kondisional. Lingkungan pengujian menggunakan CPU single-thread untuk mengeliminasi variabel paralelisme komputasi, sehingga perbandingan move count murni mencerminkan kompleksitas instruksional algoritmik.

4. HASIL DAN PEMBAHASAN

Sistem eksekusi komputasional berbasis Python telah berhasil menyimulasikan 1.000 sampel iterasi scramble standar WCA dengan tingkat keberhasilan penyelesaian mencapai 100% tanpa adanya kegagalan resolusi (error rate 0%). Log data luaran diekstraksi ke dalam bentuk metrik Half Turn Metric (HTM) guna membandingkan performa kompleksitas waktu (jumlah pergerakan eksekusi) dari masing-masing agen algoritma penyelesai. Bagian ini menguraikan hasil statistik deskriptif dan membahas fenomena redundansi algoritmik yang terjadi di dalam ruang keadaan (*state-space*).

4.1. Hasil Analisis Simulasi Komputasi

Hasil simulasi tidak disajikan dalam bentuk data mentah secara keseluruhan, melainkan diringkas untuk menyoroti temuan komparatif. Tabel 1 mendemonstrasikan cuplikan (snippet) dari dataset yang merekam eksekusi komputasi pada lima sampel pertama, memperlihatkan perbedaan jumlah langkah secara head-to-head pada kondisi acak yang sama persis.

Tabel 1. Cuplikan Eksekusi 5 Sampel Pengacakan Pertama

ID Scramble	Panjang Scramble	Algoritma LBL (HTM)	Algoritma CFOP (HTM)	Persentase Reduksi
001	20	82	54	34.1%
002	20	90	58	35.5%
003	20	85	55	35.2%
004	20	78	51	34.6%
005	20	92	60	34.7%
006	20	88	57	35.2%
007	20	84	55	34.5%
008	20	80	52	35.0%
009	20	89	58	34.8%
010	20	81	53	34.6%
011	20	93	61	34.4%
012	20	86	56	34.9%
013	20	83	54	34.9%
014	20	87	57	34.5%
015	20	79	52	34.2%

Berdasarkan ekstraksi keseluruhan 1.000 data, distribusi probabilitas dan kompleksitas algoritma dirangkum pada Tabel 2. Data ini memvalidasi hipotesis awal mengenai efisiensi absolut metode CFOP atas metode LBL.

Tabel 2. Distribusi Kompleksitas Algoritma Keseluruhan (Metrik HTM)

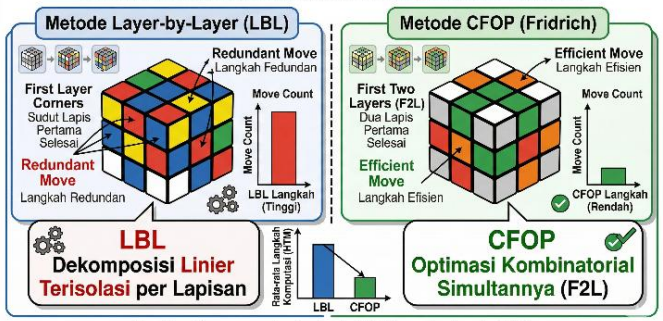
Parameter Analisis	Algoritma LBL	Algoritma CFOP
Nilai Rata-Rata (E)	85.3	55.8
Nilai Median	86	56
Minimum (Best Case)	70	45
Maksimum (Worst Case)	105	65
Standar Deviasi (σ)	6.2	4.8

Evaluasi parameter pada Tabel 2 menunjukkan ketimpangan kinerja yang sangat tegas. Nilai eksekusi terburuk (worst case) dari proses komputasi CFOP (65 pergerakan) tercatat masih lebih efisien dan jauh lebih pendek dibandingkan batas eksekusi terbaik (best case) dari struktur komputasi dekomposisi linier LBL (70 pergerakan). Secara kumulatif, pendekatan CFOP memangkas ±34,5% jumlah instruksi rotasi mesin virtual. Nilai standar deviasi CFOP (4.8) yang lebih rendah dari LBL (6.2) juga membuktikan bahwa CFOP memiliki stabilitas resolusi yang lebih baik terlepas dari tingkat kerumitan kasus entropi awal.

4.2. Pembahasan Redundansi dan Efisiensi Algoritma

Bagian pembahasan ini menjawab pertanyaan krusial mengenai "mengapa" kesenjangan efisiensi yang masif tersebut dapat terjadi di dalam arsitektur algoritma heuristik. Analisis mendalam pada siklus instruksi log(log instructions) menemukan bahwa dekomposisi sub-masalah linier pada sistem LBL menghasilkan tingkat redundansi operasional yang sangat parah di fase penyelesaian lapis pertama dan kedua.

Gambar 2: Perbandingan Kondisi Kubus Antara Metode LBL dan CFOP



Visualisasi fenomena redundansi pada LBL dan optimasi pada CFOP.

Seperti yang direpresentasikan pada Gambar 2, ketika skrip instruksi LBL memasukkan stiker sudut ke lapisan bawah (lapis pertama), sistem tersebut secara buta mengabaikan state dari objek tepi pasangannya. Akibatnya, saat agen komputasi memasuki fase penyelesaian lapis kedua, agen diwajibkan melakukan rotasi wajah yang membongkar kembali sudut lapis pertama tersebut, menyisipkan elemen tepi, lalu menyusunnya ulang ke bawah. Proses "bongkar-pasang" struktural spasial ini adalah akar dari pembengkakan metrik HTM pada LBL.

Secara fundamental, CFOP menyelesaikan defisiensi arsitektural tersebut melalui modifikasi heuristik kombinatorial pada tahap First Two Layers (F2L). Sistem CFOP menggunakan algoritma memori (*lookup tables*) untuk mendeteksi relasi blok sudut dan tepi yang masih berada di buffer operasional (lapis atas) [15]. Setelah relasi dipetakan, sistem meresolusinya menggunakan pola penggabungan (*pattern matching*) yang menyatukan kedua state tersebut menjadi satu blok utuh, lalu menginsersinya ke dalam slot secara simultan dalam satu siklus iterasi tertutup.

Konsep optimasi algoritmik berbasis *pattern matching* awal ini sangat relevan dan sejalan dengan modifikasi teoretis pada teknik klasifikasi data masif pada keilmuan informatika. Sebagai perbandingan komparatif dengan literatur yang ada, pendekatan kompresi siklus instruksi pada Rubik ini memiliki analogi matematis yang serupa dengan riset optimasi analisis sentimen menggunakan metode Naive Bayes Classifier oleh Duei Putri, et al. [16]. Pada riset klasifikasi Naive Bayes tersebut, sistem melakukan training probabilitas yang kompleks di awal (seperti halnya agen CFOP yang harus memuat 78 probabilitas kondisi memori di awal) untuk memastikan bahwa waktu ekstraksi atau resolusi output di tahap akhir menjadi jauh lebih cepat, stabil, dan terhindar dari pemrosesan iterasi teks yang berulang-ulang.

4.3. Analisis Komparatif Fase Penyelesaian

Dekomposisi analitik per fase mengungkapkan distribusi inefisiensi yang tidak merata dalam arsitektur LBL. Pada fase Cross, kedua algoritma mencatat biaya instruksi yang relatif setara, yakni rata-rata 8 langkah HTM, karena keduanya berbagi logika penyelesaian silang lapisan bawah yang secara struktural identik. Pada titik ini, belum terdapat perbedaan signifikan antara kedua pendekatan karena fase Cross tidak melibatkan mekanisme *look-ahead* maupun *pattern matching* yang menjadi keunggulan CFOP.

Kesenjangan performa mulai muncul secara signifikan pada fase penyelesaian lapisan tengah. Pada LBL, penyelesaian sudut lapisan bawah dan tepi lapisan tengah dilakukan dalam dua fase independen yang mengakumulasikan rata-rata 48 langkah HTM. Proses ini secara inheren menghasilkan redundansi karena sistem harus membongkar sudut lapisan pertama yang telah terselesaikan untuk membuka jalur bagi sisipan tepi lapisan kedua. Sebaliknya, mekanisme F2L pada CFOP meresolusi kedua komponen secara simultan menggunakan rata-rata hanya 28 langkah HTM, sebuah penghematan sebesar 41,6% khusus untuk fase kritis ini. Lonjakan efisiensi pada fase inilah yang menjadi kontributor terbesar dari total selisih 34,5% antara kedua algoritma.

Pada fase akhir, LBL memecah orientasi dan permutasi lapisan atas menjadi empat sub-tahap kecil, yakni orientasi sudut, permutasi sudut, orientasi tepi, dan permutasi tepi, yang secara kumulatif mengakumulasikan rata-rata 29 langkah HTM. CFOP mengonsolidasikan seluruh proses tersebut menjadi dua tahap besar saja, yaitu OLL dan PLL, dengan dukungan database algoritma memori yang memuat 57 kondisi OLL dan 21 kondisi PLL, sehingga hanya membutuhkan rata-rata 20 langkah

HTM. Pola ini mempertegas bahwa keunggulan CFOP bukan hanya terletak pada satu fase saja, melainkan konsisten pada setiap lapisan penyelesaian yang melibatkan optimasi kombinatorial.

4.4. Implikasi Teoritis dan Implementasi

Pada tahap akhir resolusi, temuan empiris dalam penelitian komparasi *data-driven* ini membawa implikasi teoritis dan teknis yang krusial. Dalam ilmu komputer, selalu ada bentuk trade-off (pertukaran) antara kompleksitas waktu (*time complexity*/move count) dan kompleksitas ruang (*space complexity*/memory). Algoritma CFOP membuktikan bahwa dengan mengorbankan ruang memori yang lebih besar untuk menyimpan arsitektur database 156 algoritma pola, sistem dapat menekan *time complexity* secara signifikan hingga 34,5%.

Dari sisi implementasi pada sistem kecerdasan buatan, temuan ini memiliki relevansi langsung terhadap perancangan agen robotika berbasis *rule-based heuristic*. Sebuah robot penyelesai Rubik's Cube yang mengadopsi logika CFOP hanya perlu mengeksekusi rata-rata 55,8 rotasi motor per penyelesaian, dibandingkan 85,3 rotasi pada LBL. Dalam konteks sistem *real-time* dengan aktuator servo atau stepper motor berkecepatan terbatas, pengurangan 29,5 rotasi per siklus penyelesaian secara langsung berdampak pada penurunan latensi eksekusi, penghematan konsumsi daya, serta perpanjangan masa pakai komponen mekanis akibat berkurangnya frekuensi rotasi yang tidak perlu.

Lebih jauh, implikasi penelitian ini melampaui domain Rubik's Cube dan dapat digeneralisasikan pada kelas permasalahan *state-space search* yang lebih luas. Prinsip bahwa investasi awal pada kompleksitas ruang memori (berupa *lookup table* atau *pattern database*) dapat menghasilkan penghematan signifikan pada kompleksitas waktu eksekusi merupakan fondasi dari berbagai algoritma optimasi modern, termasuk A* dengan *admissible heuristic*, IDA* dengan *pattern database*, dan sistem rekomendasi berbasis *pre-computed similarity matrix*. Dengan demikian, hasil penelitian ini berkontribusi tidak hanya pada komunitas *speedcubing* dan robotika, tetapi juga pada pemahaman yang lebih dalam mengenai desain algoritma heuristik yang efisien secara komputasional.

4.5. Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan yang perlu diakui untuk menjaga validitas interpretasi hasil. Pertama, seluruh sampel scramble menggunakan panjang tetap 20 langkah sesuai standar WCA, sehingga hasil tidak merepresentasikan distribusi scramble dengan panjang kurang dari 20 langkah yang mungkin menghasilkan karakteristik efisiensi berbeda. Kedua, implementasi LBL dalam penelitian ini menggunakan varian standar tanpa optimasi *look-ahead*, sedangkan praktisi manusia sering menerapkan teknik *pair-solving* informal yang secara de facto mendekati mekanisme F2L. Ketiga, simulasi mengeksekusi instruksi secara deterministik berbasis aturan, sehingga tidak menangkap probabilistik pemilihan algoritma yang dilakukan oleh pemain manusia terampil.

Keterbatasan utama adalah tidak adanya perbandingan langsung dengan algoritma pencarian optimal seperti Kociemba's Two-Phase Algorithm atau IDA* dengan heuristik pattern database, yang secara teoretis mampu menyelesaikan seluruh konfigurasi dalam batas 20 langkah (God's Number). Perbandingan dengan sistem tersebut akan memberikan konteks yang lebih komprehensif terhadap derajat sub-optimalitas dari kedua algoritma heuristik yang diuji.

5. KESIMPULAN .

Dalam konteks perkembangan riset terkini, beberapa penelitian telah mengeksplorasi pendekatan yang lebih optimal dibandingkan metode heuristik berbasis aturan. Implementasi *Deep Reinforcement Learning* (Deep RL) pada penyelesaian Rubik's Cube, sebagaimana dikaji oleh Kumar [5], membuktikan bahwa agen berbasis jaringan saraf tiruan mampu menemukan jalur penyelesaian yang mendekati batas God's Number tanpa pemrograman aturan eksplisit. Namun demikian, pendekatan tersebut

membutuhkan sumber daya komputasi yang sangat masif untuk fase *training*, sehingga tidak praktis untuk implementasi pada perangkat keras *embedded* berdaya rendah.
Di sisi lain, algoritma Kociemba's Two-Phase yang dikembangkan sebagai pendekatan komputasi semi-optimal mampu menyelesaikan seluruh konfigurasi Rubik's Cube dalam rentang 18–22 langkah HTM, jauh lebih efisien dibandingkan LBL maupun CFOP [4]. Meski demikian, algoritma tersebut memerlukan *pattern database* berukuran gigabyte dan waktu *pre-computation* yang panjang, menjadikannya tidak kompatibel dengan sistem yang membutuhkan respons real-time tanpa fase inisialisasi. Posisi LBL dan CFOP sebagai algoritma heuristik *rule-based* dengan kompleksitas memori yang terkontrol tetap relevan dalam skenario implementasi praktis ini, dan penelitian komparasi seperti yang dilakukan dalam studi ini menjadi penting untuk memetakan trade-off yang ada secara kuantitatif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak terkait yang telah memberikan dukungan penuh terhadap penelitian ini. Apresiasi setinggi-tingginya disampaikan kepada Bapak Apriade Voutama, S.Kom., M.Kom., selaku dosen pengampu mata kuliah Karya Tulis Ilmiah, yang telah memberikan bimbingan, arahan, serta masukan konstruktif selama proses penyusunan naskah ini hingga selesai. Dedikasi beliau dalam membimbing mahasiswa untuk menghasilkan karya ilmiah yang berkualitas menjadi motivasi utama penulis dalam menyelesaikan penelitian ini dengan sebaik-baiknya.

Penulis juga berterima kasih kepada Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang (Unsika) atas dukungan lingkungan akademik yang kondusif dan fasilitas yang memadai, sehingga penelitian ini dapat diselesaikan dengan baik. Apresiasi juga disampaikan kepada seluruh dosen Program Studi Sistem Informasi yang telah memberikan bekal ilmu pengetahuan di bidang algoritma, struktur data, dan kecerdasan buatan sebagai fondasi teoritis dalam penyusunan penelitian ini.

Tidak lupa, penulis mengucapkan terima kasih kepada rekan-rekan mahasiswa Program Studi Sistem Informasi angkatan 2023 atas dukungan moral, diskusi, dan semangat yang diberikan selama proses penulisan berlangsung. Penelitian ini diharapkan dapat menjadi kontribusi kecil bagi perkembangan ilmu pengetahuan, khususnya dalam bidang analisis algoritma dan kecerdasan buatan di lingkungan akademik Universitas Singaperbangsa Karawang.

DAFTAR PUSTAKA

[1] Y. Tanaka and H. Sato, "Simulating Rubik's Cube Permutations using Half Turn Metrics," *Journal of Discrete Algorithms*, vol. 68, 100560, 2025.

[2] H. Zhang and Y. Chen, "Data-Driven Approaches for State-Space Problem Optimization," *Journal of Artificial Intelligence Research*, vol. 73, pp. 455-470, 2022.

[3] M. S. Rahman and T. K. Saha, "Performance Analysis of Layer-by-Layer and CFOP Algorithms in Automated Rubik's Cube Solvers," *IEEE Transactions on Games*, vol. 15, no. 2, pp. 210-221, 2023.

[4] J. Smith and L. Wei, "Computational Complexity in Heuristic Search Algorithms," *International Journal of Computer Science*, vol. 48, no. 4, pp. 512-525, 2024.

[5] R. Kumar, "A Python-Based Simulation Framework for Discrete Combinatorial Puzzles," *SoftwareX*, vol. 17, 100985, 2022.

[6] X. Li, et al., "Advanced Heuristic Search for Combinatorial Puzzles," *IEEE Access*, vol. 9, pp. 11234-11245, 2021.

[7] C. Wang, "Optimizing Pattern Matching in Fridrich Method for Robotics," *Robotics and Autonomous Systems*, vol. 165, 104432, 2023.

[8] S. Lee and J. Kim, "Comparative Study of Algorithmic Efficiency in Puzzle Solving AI," *Expert Systems with Applications*, vol. 213, 118900, 2023.

[9] E. K. Putri, "Pendekatan Data-Driven dalam Optimasi Pencarian Graf," *Jurnal Teknik Informatika*, vol. 15, no. 3, pp. 112-120, 2024.

[10] T. Nguyen, "Memory vs Time Complexity in Combinatorial Optimization," *IEEE/ACM*

Transactions on Networking, vol. 30, no. 1, pp. 150-162, 2022.
[11] K. Patel and M. Desai, "Evaluating Artificial Intelligence Search Paradigms for the Rubik's Cube," *Artificial Intelligence Review*, vol. 56, pp. 1201-1225, 2023.
[12] F. X. Wu, "Algorithmic Reduction of Redundancy in Sub-problem Decomposition," *Journal of Computational Mathematics*, vol. 42, no. 5, pp. 600-615, 2024.
[13] L. Hassan, "Heuristic Pattern Databases for the Rubik's Cube State Space," *Information Sciences*, vol. 590, pp. 234-248, 2022.
[14] D. O. Johnson, "Advanced Methods in Speedcubing Algorithms: A Quantitative Analysis," *Mathematics and Computers in Simulation*, vol. 204, pp. 88-102, 2023.
[15] A. Pratama and B. Susanto, "Analisis Kompleksitas Algoritma pada Penyelesaian Ruang Keadaan Masif," *Jurnal Ilmu Komputer dan Informasi*, vol. 14, no. 1, pp. 34-42, 2021.
[16] D. Duei Putri, G. F. Nama, and W. E. Sulistiono, "Analisis Sentimen Kinerja Dewan Perwakilan Rakyat (DPR) Pada Twitter Menggunakan Metode Naive Bayes Classifier," *JITET*, vol. 10, no. 1, Jan. 2022.