ISSN : 1411-1624
e-ISSN: 2621-8089

Studi Penggunaan Algoritma Flood Fill Untuk Mencari Jalan Tercepat Dalam Labirin Menggunakan Multi Agen

Semuil Tjiharjadi*
Universitas Kristen Maranatha
Jalan Suria Sumantri No.65 Bandung, Indonesia
Sur-el : semuiltj@gmail.com
***) Corresponden Author**

Received: 16 Juni 2025 Reviewed: 19 Juni 2025 Accepted: 3 Juli 2025

Abstract : *This study aims to establish a solid foundation for the development of advanced automated pathfinding systems by implementing Multi-Agent Pathfinding (MAPF) methods within maze environments. Although the Flood Fill algorithm has long been recognized and widely utilized for maze-solving tasks by single agents due to its simplicity and efficiency, its direct application in multi-agent contexts remains limited and presents certain challenges. To address these challenges, this research reviews existing studies on MAPF and conventional maze-solving techniques, identifying critical gaps and opportunities for enhancement. Based on this review, the Flood Fill algorithm, which has traditionally been employed for single-agent navigation, is reconsidered for scenarios involving multiple agents navigating the same maze concurrently, with necessary modifications proposed to ensure its suitability for multi-agent pathfinding. Furthermore, a comparative analysis of relevant algorithms from the literature demonstrates that the Flood Fill algorithm outperforms other commonly used maze-solving methods in terms of speed, efficiency, and robustness. These findings underscore the potential for further improvements to Flood Fill, positioning it as a practical and innovative solution for more effective multi-agent robotic and automation systems.*

Keywords: *multi-agents, maze, path finding, flood-fill*

Abstrak : *Penelitian ini bertujuan untuk menyediakan landasan yang kuat bagi pengembangan sistem pencarian jalur otomatis tingkat lanjut melalui penerapan metode Pencarian Jalur Multi-Agen (Multi-Agent Pathfinding/MAPF) pada lingkungan labirin. Meskipun algoritma Flood Fill telah lama dikenal dan banyak digunakan untuk penyelesaian labirin oleh agen tunggal karena kesederhanaan dan efisiensinya, penerapannya secara langsung pada konteks multi-agen masih terbatas dan menghadapi sejumlah tantangan. Untuk menjawab tantangan tersebut, penelitian ini mengkaji berbagai studi terkait MAPF dan teknik pemecahan labirin konvensional, serta mengidentifikasi kesenjangan penting dan peluang peningkatan kinerja. Berdasarkan tinjauan tersebut, algoritma Flood Fill yang secara tradisional digunakan untuk navigasi agen tunggal dipertimbangkan kembali penerapannya pada skenario dengan beberapa agen yang menavigasi labirin yang sama secara bersamaan, dengan usulan modifikasi untuk memastikan algoritma ini dapat berfungsi secara efektif dalam konteks multi-agen. Selain itu, analisis komparatif terhadap algoritma relevan yang diulas menunjukkan bahwa algoritma Flood Fill mampu mengungguli algoritma pemecahan labirin lainnya yang umum digunakan dalam hal kecepatan, efisiensi, dan ketahanan. Temuan ini menegaskan potensi pengembangan lebih lanjut Flood Fill sebagai solusi praktis dan inovatif untuk mendukung sistem robotika dan otomasi multi-agen yang lebih efektif.*

Kata kunci*: multi-agen, labirin, pencarian jalur, flood-fill*

1. PENDAHULUAN

Pencarian jalur (pathfinding) merupakan salah satu aspek krusial dalam pengembangan sistem robotika dan otomasi, khususnya dalam konteks eksplorasi lingkungan tidak dikenal seperti labirin. Efektivitas sistem pencarian jalur sangat bergantung pada algoritma yang digunakan untuk menavigasi rintangan dan

ISSN : 1411-1624
e-ISSN: 2621-8089

menemukan jalur optimal dari titik awal menuju tujuan. Salah satu algoritma yang telah dikenal luas dan terbukti efisien dalam konteks agen tunggal adalah *Flood Fill*. Algoritma ini menawarkan keunggulan dalam hal kesederhanaan implementasi, efisiensi waktu komputasi, dan kemampuan menjelajah lingkungan tanpa perlu pemetaan awal secara menyeluruh [1]. Namun demikian, *Flood Fill* pada dasarnya dirancang untuk skenario agen tunggal, sehingga penerapannya pada sistem multi-agen menimbulkan sejumlah keterbatasan, terutama dalam aspek koordinasi, konflik jalur, dan redundansi eksplorasi.

Seiring dengan meningkatnya kebutuhan akan sistem otonom yang melibatkan banyak agen, seperti robot eksplorasi, kendaraan logistik otomatis, dan sistem pencarian dan penyelamatan, maka diperlukan adaptasi algoritma yang sebelumnya hanya optimal untuk agen tunggal agar dapat bekerja secara efektif dalam skenario multi-agen. Beberapa pendekatan algoritmik telah dikembangkan untuk menangani permasalahan *Multi-Agent Pathfinding (MAPF)*, antara lain algoritma *Cooperative A** [2], *Conflict-Based Search (CBS)* [3], dan *Push and Swap* [4]. Meskipun algoritma-algoritma tersebut menawarkan solusi yang menjanjikan dalam aspek koordinasi dan penghindaran konflik antaragen, kompleksitas komputasi yang tinggi dan kebutuhan sumber daya yang besar sering menjadi hambatan dalam implementasi nyata, khususnya dalam sistem dengan keterbatasan perangkat keras dan waktu pemrosesan.

Studi-studi sebelumnya telah mengevaluasi performa berbagai algoritma MAPF dalam berbagai konteks. Silver (2005) mengusulkan *Cooperative A** sebagai solusi koordinatif untuk navigasi multi-agen, namun performanya menurun secara signifikan seiring bertambahnya jumlah agen dan kompleksitas lingkungan [2]. Sharon et al. (2015) memperkenalkan *Conflict-Based Search (CBS)* yang lebih efisien dalam penghindaran konflik, tetapi tetap memiliki keterbatasan dalam dinamika lingkungan yang berubah-ubah [3]. Sementara itu, algoritma *Flood Fill* relatif kurang mendapat perhatian dalam konteks MAPF karena keterbatasan awalnya sebagai algoritma untuk satu agen. Namun, karakteristiknya yang ringan dan cepat menunjukkan potensi besar jika dimodifikasi untuk menangani skenario multi-agen.

Berdasarkan identifikasi tersebut, terdapat kebutuhan untuk mengevaluasi kembali algoritma *Flood Fill* dalam konteks MAPF, serta mengembangkan versi modifikasi yang mampu mengatasi keterbatasannya. Penelitian ini bertujuan untuk memodifikasi algoritma *Flood Fill* agar dapat diterapkan secara efektif dalam lingkungan labirin yang melibatkan lebih dari satu agen, dengan mempertahankan keunggulan algoritma asli sekaligus menambahkan mekanisme koordinasi antaragen.

Tujuan utama dari penelitian ini adalah mengevaluasi kemungkinan algoritma *Flood Fill* dapat dimodifikasi untuk skenario multi-agen, serta apa yang menjadi kendala dan keterbatasan algoritma *Flood Fill* sehingga perlu dimodifikasi. Lalu dari keterbatasan algoritma *Flood Fill* didapatkan gagasan yang diperlukan untuk meningkatkan algoritma ini sehingga

ISSN : 1411-1624
e-ISSN: 2621-8089

memiliki karakteristik baru yang dapat mengatasi hambatan yang selama ini menghalanginya untuk dikembangkan dalam pencarian jalur agen banyak.

Kontribusi penelitian ini mencakup tiga aspek utama: (1) evaluasi kinerja algoritma tersebut dibandingkan algoritma MAPF lainnya yang telah ada; serta (2) penekanan pada efisiensi dan kesederhanaan algoritma sebagai solusi praktis untuk sistem berbasis robot otonom dalam kondisi terbatas; (3) modifikasi yang diperlukan algoritma *Flood Fill* untuk mendukung sistem multi-agen dalam lingkungan labirin;. Dengan pendekatan ini, diharapkan algoritma *Flood Fill* yang telah dimodifikasi dapat menjadi alternatif yang unggul dalam pengembangan sistem pencarian jalur multi-agen yang efektif dan adaptif.

2. METODOLOGI PENELITIAN

2.1 Pendekatan dan Model Penelitian

Penelitian ini menggunakan metode kualitatif deskriptif berbasis studi literatur sistematis, dengan tujuan mengumpulkan, mengevaluasi, dan membandingkan algoritma pathfinding dalam lingkungan grid/maze, khususnya dalam konteks sistem *Multi-Agent Pathfinding* (MAPF). Fokus utama adalah pada *Flood Fill* dan potensinya yang dimodifikasi untuk skenario multi-agen.

Proses penelitian mencakup:

1. Penelusuran artikel dari database bereputasi *(IEEE Xplore, ScienceDirect, SpringerLink, Google Scholar*) dengan kata kunci: "*Flood Fill maze*", "*multi-agent pathfinding*", "*Flood Fill MAPF*".
2. Penyaringan artikel (2014–2024) berdasarkan relevansi:
   a. Implementasi atau analisis algoritma dalam *labirin/grid,*
   b. Membahas konteks *single-agent* maupun multi-agent.

2.2 Teknik Pengumpulan dan Seleksi Literatur

Data dikumpulkan melalui teknik *documentary study*, melibatkan seleksi artikel berdasarkan kriteria : periode, konteks, dan keluaran. Berikut adalah beberapa artikel yang dianalisis: Semuil & Setiawan (2016) : *Flood Fill* pada *robot maze single-agent* [1], Tjiharjadi et al. (2017, 2022): Penggunaan dan optimasi *Flood Fill* [5], termasuk modifikasi untuk lingkungan multi-agen [6], Elshamarka & Saman (2012): *Flood Fill* dalam *robot maze* [7], Sharon et al. (2015): CBS untuk *MAPF optimal* [8], Linardakis et al. (2024): A*nalisis Multi-robot maze* dengan *cost-utility* [9], Qiu (2020): *Integrasi tradisional pathfinding* dan *reinforcement learning* [10], Bertolini, 2022[11].

2.3 Kerangka Teoritis

Penyusunan landasan teoritis dalam Penelitian ini, menggunakan beberapa konsep yang menjadi dasar dari membangun kerangka teoritis untuk algoritma yang lebih baik pada pencarian jalur di labirin menggunakan agen banyak. Algoritma *Flood Fill* dasar perbandingan utama dari berbagai algoritma pencarian jalur lainnya, untuk meningkatkan

ISSN : 1411-1624
e-ISSN: 2621-8089

efisiensi dalam menemukan jarak terpendek di dalam labirin. Lalu penggunaan *MAPF* klasik seperti: *Cooperative* A*, CBS (Sharon et al. 2015) menjadi uji banding yang efektif untuk membandingkan kedua algoritma terbaik dalam pencarian jalur.

Selain itu penggunaan pendekatan modern *MAPF*, seperti Bio-inspirasi (PSO/ACO), belajar mesin dan DRL serta penggunaan alternatif multi-agen efisien, seperti *Flow field*, memperkaya kerangka teoritis yang digunakan dalam pengembangan penelitian ini.

2.4 Rancangan Analisis Literasi

Analisis dilakukan secara sistematis dalam dua tahap:

a. **Identifikasi fitur**
   Keunggulan dan kelemahan masing-masing algoritma yang digunakan dalam perbandingan. Di sini digunakan juga berbagai data algoritma pencarian jalur pada labirin yang popular sekalipun itu agen tunggal.
b. **Perbandingan kualitatif**
   Hasil identifikasi fitur tersebut kemudian disajikan dalam bentuk tabel dan grafik untuk memperjelas komparasi dari semua algoritma yang diuji. Berdasarkan komparasi ini, maka dapat mempermudah rekomendasi modifikasi yang diperlukan.

2.5 Formulasi Rekomendasi

Pada tahapan ini, dirancang susunan rekomendasi meliputi: sintesis hasil dan algoritma yang paling cocok. Lalu dibuat formulasi kerangka kerja untuk modifikasi algoritma Flood Fill yang diperlukan dengan berbagai pertimbangan yang ditemukan dari identifikasi dan komparasi berbagai algoritma yang digunakan sebagai acuan perbandingan. Hasil pada tahapan ini adalah sebuah formulasi rekomendasi yang akan diwujudkan pada penelitian lanjutan yang merealisasikan algoritma tersebut dalam membangun algoritma baru.

3. HASIL DAN PEMBAHASAN

3.1 Perbandingan Algoritma

Berbagai algoritma ditinjau untuk mendapatkan karakteristik setiap algoritma, sehingga dapat dianalisa berdasarkan keunggulan dan kelemahan masing-masing algoritma.

3.1.1 *Flood Fill* (Standar)

Flood Fill bekerja dengan menyebarkan nilai dari titik target (biasanya nilai 0) ke seluruh grid secara rekursif atau iteratif. Setiap *cell* tetangga diberi nilai +1 dari cell sebelumnya hingga seluruh maze terisi nilai jarak dari target. Agen kemudian bergerak menuju cell dengan nilai terkecil. *Rumus algoritma* ini adalah:

$$V(x,y) = min\{V(x',y')\} + 1 \quad (1)$$

di mana (x′,y′) adalah tetangga cell (x,y).

Keunggulan utama algoritma ini adalah runtime yang sangat cepat dan kompleksitas rendah, menjadikannya pilihan populer untuk robot *maze single-agent*. Namun, algoritma ini tidak mendukung koordinasi antar-agen dan cenderung menghasilkan jalur yang kurang efisien di area terbuka karena tidak mempertimbangkan optimasi global [12].

ISSN : 1411-1624
e-ISSN: 2621-8089

3.1.2 Dijkstra

Dijkstra bekerja dengan menyebarkan nilai jarak minimum dari start ke semua titik, memilih node dengan jarak minimum yang belum dikunjungi. Rumus algoritma Dijkstra adalah :

$$d(v) = \min\{d(u) + w(u,v)\} \quad (2)$$

di mana w(u,v) = cost edge u ke v.

Algoritma ini menjamin jalur terpendek, tetapi memiliki runtime lambat pada grid besar dan tidak dirancang untuk multi-agen [13].

3.1.3 A / Cooperative A**

A* efektif untuk *single-agent* dengan jalur efisien, sedangkan *Cooperative A** mendukung koordinasi multi-agen dengan baik. Kelemahannya adalah kurang skalabel untuk jumlah agen besar karena overhead perhitungan meningkat drastis. Rumus dari A* ini merupakan gabungan dari Dijkstra dengan Heuristik (biasanya menggunakan *Manhattan distance*), sehingga rumusnya menjadi [14]:

$$f(n) = g(n) + h(n) \quad (3)$$

$g(n)$: cost dari start ke node n;
$h(n)$: estimasi cost ke goal.

3.1.4 Conflict-Based Search (CBS)

CBS memberikan solusi jalur optimal dengan manajemen konflik yang baik, tetapi membutuhkan waktu komputasi tinggi sehingga kurang praktis untuk real-time MAPF di lingkungan besar [3].

Cara kerja algoritma ini adalah dengan mencari jalur individual optimal → deteksi konflik → pecah node menjadi constraint (misal agent A hindari posisi X di t).

3.1.5 Push and Swap (PAS)

Algoritma ini andal untuk koordinasi desentralisasi, namun rumit diimplementasikan dan berpotensi mengalami deadlock jika tidak disertai kontrol tambahan. Dalam algoritma *Push and Swap*, agen saling menukar posisi atau mendorong agen lain jika jalur terhalang [4].

3.1.6 Flow Field

Flow Field menawarkan *runtime* sangat cepat dan cocok untuk banyak agen. Kelemahannya adalah ketidakoptimalan jalur individu dan ketergantungan pada desain *field* yang efektif. *Flow Field* bekerja dengan menyiapkan medan vektor, lalu setiap agen mengikuti gradien menuju target [11].

3.1.7 Breath-First Search (BFS)

Memberikan jalur terpendek pada grid seragam tetapi boros memori sehingga kurang cocok untuk maze besar. BFS bekerja dengan ekspansi ke node tetangga secara tingkatan [15].

3.1.8 Depth-First Search (DFS)

Mudah diimplementasikan dan cepat, namun jalur yang dihasilkan tidak efisien karena bisa sangat panjang. DFS bekerja dengan ekspansi node sedalam mungkin sebelum terpaksa backtrack [16].

3.1.9 Tremaux

Tremaux mampu menjamin robot dapat keluar dari maze tanpa jalur optimal. Cocok untuk robot sederhana tetapi tidak efisien. Cara kerjanya dengan memberikan tanda pada persimpangan saat dilewati. Jika persimpangan

ISSN : 1411-1624
e-ISSN: 2621-8089

telah ditandai dua kali, maka Tremaux akan melakukan backtrack [17].

3.1.10 Pledge

Algoritma Pledge mampu bekerja secara sederhana dan efektif untuk robot tanpa peta, tapi jalurnya panjang dan tidak optimal. Pledge bekerja dengan mengikuti dinding sambil menghitung sudut untuk menghindari terjadinya putaran looping [18].

3.1.11 Genetic Algorithm(GA)

Algoritma Genetic bersifat adaptif dan mampu menemukan jalur mendekati optimal, namun memerlukan waktu komputasi tinggi dan parameterisasi yang kompleks, sehingga tidak praktis untuk aplikasi *real-time MAPF*. GA bekerja menggunakan populasi solusi dihasilkan dihasilkan dengan *crossover* dan mutase, lalu fitness dihargai sama dengan biaya pada jalur [19].

3.1.12 Ant Colony Optimization (ACO)

Algoritma Ant Colony Optimization juga bersifat adaptif dan mampu mencari jalur yang optimal, namun sama seperti GA, ACO memerlukan waktu komputasi tinggi dan parameterisasi yang kompleks. ACO bekerja dengan menjelajah acak, lalu meninggalkan jejak feromon. Setiap jalur terbaik akan memperkuat feromon. Kelemahannya adalah memerlukan iterasi yang banyak [20].

3.2 Hasil Sintesis Studi Literatur

Penelitian ini menyajikan hasil sintesis dari berbagai literatur terkait algoritma *pathfinding* yang digunakan dalam penyelesaian maze, baik untuk agen tunggal maupun multi-agen. Perbandingan dilakukan berdasarkan empat kriteria utama, yakni runtime, efisiensi jalur *(steps/path cost*), koordinasi agen (jika mendukung multi-agen), dan kompleksitas algoritma. Data dikumpulkan melalui studi pustaka sistematis dari jurnal nasional dan internasional bereputasi, prosiding konferensi, serta buku ajar standar (Cormen et al., 2009 [21]; Sharon et al., 2015 [3]; Silver, 2005 [22]; Luna & Bekris, 2011[23]; Dorigo & Stützle, 2004[24]).

Data dalam tabel dan grafik disusun berdasarkan studi literatur sistematis. Tidak dilakukan uji coba perangkat langsung (eksperimen lab atau simulasi digital). Pengujian dilakukan dengan langkah-langkah berikut:

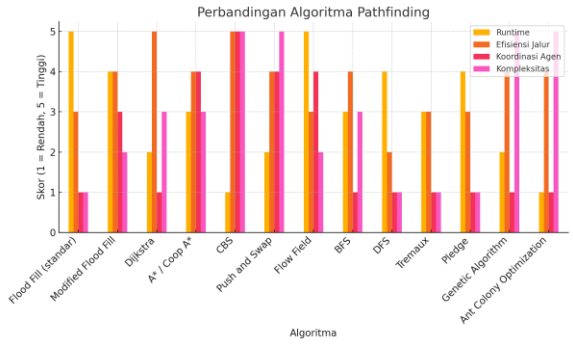
1. Pengumpulan data sekunder: Seluruh performa algoritma diekstraksi dari hasil uji yang dilaporkan dalam literatur. Di antaranya:
   a. *Runtime* dan *path cost* diambil dari laporan Sharon et al. (2015) untuk CBS[3],
   b. *Performa Flood Fill* dikutip dari Semuil & Setiawan (2016) [1] dan Tjiharjadi et al. (2022) [6],
   c. *Flow Field* dari Bertolini (2022) [11].
2. Normalisasi skor: Performa diklasifikasikan ke dalam skala kualitatif (1–5) agar dapat dibandingkan secara naratif.
3. Visualisasi data: Data dirangkum dalam tabel komparatif dan divisualisasikan melalui grafik batang agar perbedaan karakteristik algoritma lebih mudah dilihat.

ISSN : 1411-1624
e-ISSN: 2621-8089

Sebagai hasilnya, setiap algoritma dinilai secara kualitatif dengan skala 1 (rendah/buruk) hingga 5 (tinggi/baik). Data penilaian diperoleh dari hasil pengukuran dan laporan performa yang tercantum dalam literatur, bukan dari uji coba perangkat lunak mandiri, mengingat penelitian ini berbasis studi literatur. Hasil yang didapat terlihat pada tabel 1 Analisa Algoritma *Pathfinding* dan gambar 1 Perbandingan Algoritma Pathfinding.

Tabel 1. Analisa Algoritma *Pathfinding*

Algoritma	*Run-time*	*Efisi-ensi Jalur*	*Koor-dinasi Agen*	*Kom-pleksitas*
Flood Fill	5	3	1	1
Dijkstra	2	5	1	3
A*	3	4	4	3
CBS	1	5	5	5
PAS	2	4	4	5
Flow Field	5	3	4	2
BFS	3	4	1	3
DFS	4	2	1	1
Tremaux	3	3	1	1
Pledge	4	3	1	1
GA	2	4	1	5
ACO	1	4	1	5



Gambar 1. Perbandingan algoritma *Pathfinding*

3.3 Modified Flood Fill

Berdasarkan perbandingan seluruh algoritma tersebut, terlihat bahwa setiap algoritma memiliki keunggulan dan kelemahan masing-masing. Namun untuk mengembangkan algoritma yang tepat untuk pencarian jalur dalam labirin menggunakan banyak agen. Maka diperlukan algoritma yang sederhana, dapat bekerja mandiri, memiliki komputasi yang ringan dan cepat tapi mampu menghasilkan solusi yang diinginkan tanpa berakhir pada jalan buntu. Untuk itulah dipilih algoritma Flood Fill yang selama ini digunakan sebagai algoritma untuk agen tunggal dalam pencarian jalur di labirin, untuk dikembangkan sehingga memiliki kemampuan yang tepat untuk agen banyak. Karakter programnya yang ringan dan cepat, dinilai memiliki keunggulan dibandingkan algoritma lainnya.

Modifikasi pada Flood Fill dilakukan dengan menambahkan mekanisme koordinasi desentralisasi sederhana (seperti pembobotan grid dan pertukaran informasi target antar-agen). Hasil sintesis menunjukkan bahwa modifikasi ini mampu meningkatkan efisiensi jalur serta mengurangi konflik antar-agen tanpa menambah kompleksitas secara signifikan.

Cara kerja yang diusulkan adalah dengan menambahkan pembobotan dinamis seperti nilai penalti pada sel yang sudah dilewati agen lain atau integrasi dengan *flow field* local. Agen juga berbagi informasi jalur atau lokasi target.

Berikut ini adalah modifikasi rumus *Flood Fill* yang dikembangkan:

$$V'^{(x,y)} = V(x,y) + W(x,y) \qquad (4)$$

dengan W(x,y) adalah bobot tambahan (misalnya area padat atau dilalui agen lain).

Keunggulan yang diharapkan pada algoritma ini adalah memiliki algoritma yang ringan dan cepat, serta memiliki koordinasi sederhana sehingga dapat diterapkan pada cukup banyak agen. Namun untuk mewujudkannya

ISSN : 1411-1624
e-ISSN: 2621-8089

maka dibutuhkan pengaturan pembobotan agar optimal yang dapat dilakukan dengan uji coba program saat sudah diwujudkan.

Selain itu karena jalur labirin yang sempit dan hanya dapat dilalui oleh satu robot, maka untuk menghindari kemacetan pada saat pencarian jalur agen banyak, dapat dikembangkan pula *task swapping* (pertukaran tugas) dan *task reallocation* (realokasi tugas) yang sering dikenal dalam MAPF (*Multi-agent pathfinding*) dengan nama: *agent role switching* dan penggunaan *path reservation swapping* (pertukaran reservasi jalur).

Melalui penggunakan *task swapping* dan *task reallocation*, maka agen banyak dapat bertukar tugas untuk meningkatkan efisiensi sistem, seperti mengurangi total waktu tempuh, konflik jalur, atau beban energi.

Terdapat banyak konsep algoritma yang dapat dipertimbangkan untuk mengembangkan kemampuan khusus ini, seperti:

1. *Market-Based Task Allocation,* dengan konsep setiap agen / robot menawar utnuk tugas berdasarkan biaya minimum atau efisiensi. Tugas dapat dialokasikan ulang bila ada robot lain yang dapat menyelesaikannya dengan lebih baik [25]. Contoh *algoritma MBTA* adalah: *Contract Net Protocol* (CNP) dan *Consensus-Based Bundle Algorithm* (CBBA) [26].
2. *Role Exchange / Dynamic Task Reallocation*, dengan konsep robot dapat secara dinamis bertukar peran atau tugas untuk mengoptimalkan jalur atau menyelesaikan konflik, misalnya dalam MAPF. Contoh *algoritma role exchange* adalah: *Push and Swap* yang memungkinkan agen dapat saling menukar posisi untuk menghindari deadlock [27], serta *algoritma Push and Rotate* yang mengembangkan *push and swap* dengan rotasi grup agen [28].
3. *Task swapping* dalam eksplorasi agen banyak: yang menukar tugas eksplorasi robot, saat ada robot lain yang lebih dekat atau efisien secara energi [29]. Contohnya adalah *algoritma Dynamic role / task reassignment.*
4. *Task Swapping in Scheduling*, menggunakan konsep pertukaran tugas agen banyak untuk meningkatkan throughput produksi [30].

Proses *task swapping / task reallocation* diimplementasikan untuk meminimalkan waktu penyelesaian tugas dengan menghindari konflik atau *deadlock* serta meningkatkan efisiensi energi atau jarak tempuh robot dikondisi nyata.

Secara keseluruhan perbandingan karakter hasil modifikasi algoritma flood fill dibandingkan algoritma lainnya dapat terlihat pada tabel 2.

Tabel 2. Perbandingan Algoritma Modifikasi Flood Fill dengan algoritma Pathfinding pada Maze (Single-Agent dan Multi-Agent)

Algoritma	***Runtime***	***Efisiensi Jalur***	***Koordinasi Agen***	***Komplek-sitas***	***Kelebihan***	***Kekurangan***
Modified Flood Fill	Cepat	Tinggi	Desentralisasi	Rendah–Sedang	Adaptif, ringan, mendukung multi-agent	Butuh aturan tambahan untuk koordinasi optimal
Flood Fill	Sangat	Moderat	Tidak ada	Rendah	Implementasi	Tidak mendukung

ISSN : 1411-1624
e-ISSN: 2621-8089

(standar)	cepat				sederhana, cepat untuk single-agent	multi-agent, boros langkah di area terbuka
Dijkstra	Lambat	Sangat tinggi	Tidak ada	Sedang	Jalur terpendek dijamin	Lambat di grid besar
A / Cooperative A**	Sedang	Tinggi	Terpusat	Sedang–Tinggi	Efisien, jalur baik, bisa multi-agent (coop)	Tidak skalabel banyak agen (coop)
Conflict-Based Search (CBS)	Lambat (optimal)	Sangat tinggi	Terpusat	Tinggi	Jalur optimal, konflik minimal	Overhead besar untuk banyak agen
Push and Swap	Sedang	Tinggi	Desentralisasi	Tinggi	Efektif untuk interaksi kompleks	Implementasi rumit, risiko deadlock
Flow Field	Sangat cepat	Moderat	Desentralisasi	Rendah	Efisien secara global untuk banyak agen	Jalur agen individual tidak optimal
Breadth-First Search (BFS)	Sedang	Tinggi	Tidak ada	Sedang	Jalur terpendek di grid uniform	Boros memori
Depth-First Search (DFS)	Cepat	Rendah	Tidak ada	Rendah	Mudah diimplementasi	Jalur panjang, tidak optimal
Tremaux	Sedang	Moderat	Tidak ada	Rendah	Menjamin keluar dari maze (tanpa optimasi jalur)	Jalur tidak optimal
Pledge Algorithm	Cepat	Rendah–Moderat	Tidak ada	Rendah	Dapat menyelesaikan maze tanpa map	Jalur panjang
Genetic Algorithm (GA)	Variabel	Variabel	Tidak ada	Tinggi	Adaptif, bisa menemukan jalur unik	Waktu komputasi tinggi, tidak deterministik
Ant Colony Optimization (ACO)	Lambat	Tinggi	Tidak ada	Tinggi	Adaptif, jalur mendekati optimal	Perlu banyak iterasi

4. KESIMPULAN

Berdasarkan hasil sintesis, modifikasi *Flood Fill* dengan menambahkan mekanisme koordinasi desentralisasi seperti pembobotan grid dan pertukaran informasi target antar-agen, menunjukkan bahwa modifikasi ini mampu meningkatkan efisiensi jalur serta mengurangi konflik antar-agen tanpa menambah kompleksitas secara signifikan.

Selain itu algoritma ini memiliki keunggulan yang menonjol pada penerapan MAPF, disebabkan beberapa karakternya, yaitu: memiliki *runtime* cepat yang diwarisi dari *algoritma Flood Fill standar*. Selain itu algoritma ini bersifat desentralisasi dan ringan, sehingga tidak memerlukan control pusat dan mudah diimplementasikan pada robot kecil. Algoritma ini juga memiliki koordinasi sederhana sehingga melalui modifikasi akan memungkinkan agen saling berbagi Informasi jalur atau target sehingga meminimalkan konflik secara lokal. Lalu dari skalabilitas, algoritma modifikasi *flood fill* jauh lebih baik dibandingkan *CBS dan Cooperative A** yang semakin berat secara significant saat jumlah agen meningkat. Dibandingkan algoritma lain seperti CBS, ACO, GA yang sering menuntut komputasi berat, parameterisasi kompleks, sehingga kurang cocok untuk implementasi praktis nyata pada robot labirin agen banyak.

ISSN : 1411-1624
e-ISSN: 2621-8089

Beberapa modifikasi lain yang dapat dipertimbangkan untuk menambah kemampuan *Flood Fill*, adalah: *Integrasi flow field local*, sehingga setiap agen dapat menilai arah dengan bobot grid hasil *Flood Fill* dan *gradien vektor* lokal. Dapat pula diberikan prioritas jalur untuk agen yang diprioritaskan sehingga agen lain dapat menghindar. Keunggulan berbagi Informasi target oleh agen pertama yang mencapai target akan memudahkan agen lain dalam proses pencarian jalur dari rute yang lain dan ini akan membantu proses penghindaran konflik secara adaptif yang memberikan mekanisme deteksi dini konflik jalur dengan radius tertentu dan rencana jalur alternatif lokal. Secara keseluruhan, penggunaan modifikasi ini dapat menambah fitur yang lebih unggul tanpa menambah kompleksitas algoritma secara signifikan dan tetap menjaga kecepatan algoritma *Flood Fill.*

DAFTAR PUSTAKA

[1] S. Tjiharjadi and E. Setiawan, "Design and implementation of a path finding robot using flood fill algorithm," *Int. J. Mech. Eng. Robot. Res.*, 2016, doi: 10.18178/ijmerr.5.3.180-185.

[2] D. Silver, "*Cooperative pathfinding," in AI Game Programming Wisdom 3*, 3rd ed., S. Rabin, Ed. Boston: Charles River Media, 2005, pp. 99–111.

[3] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[4] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," *in Proc. 22nd Int. Joint Conf. on Artificial Intelligence* (IJCAI), 2011, pp. 294–300.

[5] S. Tjiharjadi, M. C. Wijaya, and E. Setiawan, "Optimization maze robot using A* and flood fill algorithm," *Int. J. Mech. Eng. Robot. Res.*, vol. 6, no. 5, 2017, doi: 10.18178/ijmerr.6.5.366-372.

[6] S. Tjiharjadi, S. Razali, H. A. Sulaiman, and G. Fernando, "Design of Multi-Agent Pathfinding Robot Using Improved Flood Fill Algorithm in Maze Exploration," *Int. J. Mech. Eng. Robot. Res.*, vol. 11, no. 8, pp. 631–638, 2022, doi: 10.18178/ijmerr.11.8.631-638.

[7] I. Elshamarka and A. Bakar Sayuti Saman, "Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm," *Int. J. Comput. Appl.*, vol. 56, no. 5, pp. 8–13, 2012, doi: 10.5120/8885-2882.

[8] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[9] M. Linardakis, I. Varlamis, and G. Th. Papadopoulos, "Multi-robot maze exploration using an efficient cost-utility method," **Proc. 13th Hellenic Conf. on Artificial Intelligence (SETN 2024)**, Piraeus, Greece, Jul. 2024, pp. 1–12. [Online]. Available: https://arxiv.org/abs/2407.14218

[10] H. Qiu, "*Multi-agent navigation based on deep reinforcement learning and traditional pathfinding algorithm*," *arXiv preprint arXiv:2012.09134*, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2012.09134

[11] B. Bertolini, "Flow field-based multi-agent pathfinding: A survey," **Robotics and Autonomous Systems**, vol. 150, p. 104054, 2022.

[12] A. Jabbar, "Autonomous Navigation of Mobile Robot Based on Flood Fill Algorithm," *Iraqi J. Electr. Electron. Eng.*, vol. 12, no. 1, pp. 79–84, 2016, doi: 10.37917/ijeee.12.1.8.

[13] Y. Murata and Y. Mitani, "A Fast and Shorter Path Finding Method for Maze Images by Image Processing Techniques and Graph Theory," *J. Image Graph.*, vol. 2, no. 1, pp. 89–93, 2014, doi: 10.12720/joig.2.1.89-93.

[14] J. C. Young, A. Suryadibrata, R. Luhulima, and U. M. Nusantara, "Review of Various A * Pathfinding Implementations in Game Autonomous Agent," *Int. J. New Media Technol.*, 2019.

[15] K. Sharma and C. Munshi, "A comprehensive and comparative study of

ISSN : 1411-1624
e-ISSN: 2621-8089

maze-solving techniques by implementing graph theory," *IOSR J. Comput. Eng.*, vol. 17, no. 1, pp. 24–29, 2015, doi: 10.9790/0661-17142429.

[16] K. Sharma and C. Munshi, "A comprehensive and comparative study of maze-solving techniques by implementing graph theory," *IOSR J. Comput. Eng.*, vol. 17, no. 1, pp. 24–29, 2015, doi: 10.9790/0661-17142429.

[17] I. R. Fahmi and D. J. Suroso, "A Simulation-Based Study of Maze-Solving-Robot Navigation for Educational Purposes," *J. Robot. Control*, vol. 3, no. 1, pp. 48–54, 2022, doi: 10.18196/jrc.v3i1.12241.

[18] I. R. Fahmi and D. J. Suroso, "A Simulation-Based Study of Maze-Solving-Robot Navigation for Educational Purposes," *J. Robot. Control*, vol. 3, no. 1, pp. 48–54, 2022, doi: 10.18196/jrc.v3i1.12241.

[19] J. Xin, J. Zhong, J. Sheng, P. Li, and Y. Cui, "Improved genetic algorithms based on data-driven operators for path planning of unmanned surface vehicle," *Int. J. Robot. Autom.*, 2019, doi: 10.2316/J.2019.206-0315.

[20] J. Xin, J. Zhong, J. Sheng, P. Li, and Y. Cui, "Improved genetic algorithms based on data-driven operators for path planning of unmanned surface vehicle," *Int. J. Robot. Autom.*, 2019, doi: 10.2316/J.2019.206-0315.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[22] D. Silver, "*Cooperative pathfinding*," in *AI Game Programming Wisdom 3*, S. Rabin, Ed. Hingham, MA, USA: Charles River Media, 2005, pp. 99–111.

[23] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *Proc. 22nd Int. Joint Conf. on Artificial Intelligence* (IJCAI)*, Barcelona, Spain, 2011, pp. 294–300.

[24] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.

[25] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[26] B. Ponda, L. B. Johnson, E. Frazzoli, and J. P. How, "Consensus-based bundle algorithm for multi-UAV task assignment," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1467–1500, Aug. 2011, doi: 10.1002/rnc.1752.

[27] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *Proc. 22nd Int. Joint Conf. on Artificial Intelligence* (IJCAI)*, Barcelona, Spain, 2011, pp. 294–300.

[28] B. de Wilde, G. C. de Croon, and M. A. Dorigo, "Push and rotate: Cooperative multi-agent pathfinding," in *Proc. 24th Int. Joint Conf. on Artificial Intelligence* (IJCAI)*, Buenos Aires, Argentina, 2015, pp. 2045–2051.

[29] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. 2nd Int. Conf. on Autonomous Agents*, Minneapolis, MN, USA, 1998, pp. 47–53.

[30] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys*, vol. 26, no. 1, pp. 87–119*, Mar. 1994, doi: 10.1145/174666.174668.