Jurnal Ekselenta
e- ISSN: 3089-2163
Vol. 1 No. 2. Mar 2025

EKSPLORASI BERBASIS SIMULASI TERHADAP ALGORITMA *PATHFINDING* A* UNTUK NAVIGASI BERBASIS GRID

Silviana Windasari[1] , Abdurohman[2]
[1]Prodi Teknik Elektro, Universitas Sains Indonesia, Bekasi
[2]Prodi Teknik Elektro, Universitas Sains Indonesia, Bekasi

Email : silviana.windasari@lecturer.sains.ac.id, abdurohman.abdurohman@.sains.ac.id

Pencarian jalur (*pathfinding*) merupakan tugas mendasar dalam bidang robotika, kecerdasan buatan (AI), dan pengembangan permainan, yang penting untuk memungkinkan agen bernavigasi di dalam lingkungan yang terbatas dan terstruktur. Studi ini menyajikan analisis berbasis simulasi terhadap algoritma pencarian jalur A* dalam lingkungan grid 25x25, yang diimplementasikan menggunakan Python dan Pygame. Simulasi memungkinkan pengguna untuk menentukan titik awal dan akhir, menempatkan rintangan, serta mengamati bagaimana algoritma A* secara dinamis menghitung jalur optimal. Berbagai skenario pengujian dilakukan untuk menganalisis kinerja algoritma terhadap variasi kepadatan rintangan, dengan metrik yang mencakup panjang jalur, jumlah node yang dieksplorasi, dan waktu eksekusi. Hasil menunjukkan kekuatan, efisiensi, dan keterbatasan algoritma A*, serta diperkuat dengan perbandingan kuantitatif terhadap algoritma Dijkstra dan Breadth-First Search (BFS) untuk mengevaluasi performa relatifnya.

Kata Kunci: algoritma A*, pencarian jalur, simulasi, Pygame, navigasi berbasis grid.

Abstract

Pathfinding is a fundamental task in robotics, artificial intelligence, and game development, essential for enabling agents to navigate within constrained and structured environments. This study presents a simulation-based analysis of the A pathfinding algorithm within a 25x25 grid environment, implemented using Python and Pygame. The simulation allows users to define start and end points, place obstacles, and observe how A* dynamically computes the optimal path. Multiple test scenarios are conducted to analyze the algorithm's performance in relation to obstacle density, with metrics including path length, number of explored nodes, and execution time. The results demonstrate the robustness, efficiency, and limitations of the A* algorithm, and are supported with quantitative comparisons to Dijkstra's and Breadth-First Search (BFS) algorithms to further evaluate its relative performance. This paper presents a simulation-based exploration of the A* pathfinding algorithm in a grid-based environment using Python and Pygame. The simulation allows users to interactively set start and end points, place barriers, and observe how the A* algorithm efficiently navigates the grid to find the optimal path around obstacles. Various scenarios are simulated to evaluate the algorithm's performance under different obstacle densities. The results highlight the robustness and limitations of the A* algorithm in grid-based pathfinding.*

Keywords: *A* algorithm, pathfinding, simulation, Pygame, grid-based navigation*

1. Pendahuluan

Algoritma *pathfinding* memainkan peran penting dalam berbagai aplikasi, mulai dari robotika dan kecerdasan buatan (*Artificial Intelligence/AI*) hingga pengembangan permainan video. Pencarian jalur yang efisien

memungkinkan sistem untuk menavigasi lingkungan sambil menghindari rintangan. Salah satu algoritma yang paling dikenal dalam pencarian jalur adalah *A-Star* (A*), yang terkenal karena kemampuannya dalam menyeimbangkan antara optimalitas dan efisiensi dengan menggabungkan biaya aktual dan estimasi untuk mengarahkan proses pencarian (Daohong, 2023; Sara et al., 2022; Sharmad et al., 2022; Yunbo, 2024; Zuoyu & Xinduo, 2024).

Algoritma seperti Dijkstra, *Breadth-First Search* (BFS), dan *Depth-First Search* (DFS) telah banyak digunakan untuk permasalahan pencarian jalur (Y. K. A., 2020; Daohong, 2023; E. T. P. & B., 2019; Zuoyu & Xinduo, 2024). Namun, A* menonjol karena penggunaan fungsi heuristik untuk membimbing pencarian menuju tujuan secara lebih efisien. Artikel ini mengeksplorasi algoritma A* melalui pendekatan berbasis simulasi menggunakan grid 25x25 yang diimplementasikan dalam Python dan Pygame. Pengguna dapat menempatkan rintangan, menetapkan titik awal dan tujuan, serta mengamati bagaimana algoritma menghitung jalur optimal melalui grid.

Struktur artikel ini adalah sebagai berikut: Bagian 2 mengulas karya terkait, Bagian 3 menjelaskan metodologi dan pengaturan simulasi, Bagian 4 menyajikan desain dan fungsionalitas simulasi, Bagian 5 membahas hasil dari berbagai skenario *pathfinding*, dan Bagian 6 memberikan kesimpulan serta saran untuk pengembangan lebih lanjut.

2. Tinjauan Pustaka

Pathfinding telah diteliti secara luas di berbagai bidang seperti robotika, AI, dan sistem otonom (Ángel et al., 2021; Başara, 2024; S. A. N. et al., 2022; Shahid, 2024). Karya awal mencakup algoritma Dijkstra, yang dikembangkan pada tahun 1956, yang menjamin jalur terpendek tetapi memiliki kompleksitas komputasi tinggi karena eksplorasi menyeluruh terhadap semua node dalam graf (Adeel, 2013; Alec et al., 2022; Harika, 2013; O. & D., 2020; Sidharth et al., 2022). BFS dan DFS adalah pendekatan yang lebih sederhana dan mengeksplorasi semua jalur secara *uninformed*, yang sering kali menghasilkan performa suboptimal ketika terdapat banyak rintangan (S. A., 1993; Fahed & Mohammed, 2020; Langyu, 2024; Roni et al., 2010).

Algoritma A*, yang diperkenalkan oleh Hart, Nilsson, dan Raphael pada tahun 1968(Nathan, 2018; H. P. et al., 1968; Xiao & Hao, 2011), menggabungkan keunggulan penelusuran graf dari Dijkstra dengan fungsi heuristik untuk memperkirakan biaya tersisa menuju tujuan. A* dikenal dapat menemukan solusi optimal secara efisien selama heuristik yang digunakan bersifat *admissible* (C. N. & M., 2013; Xiao & Hao, 2011; Zuoyu & Xinduo, 2024). Selama bertahun-tahun, A* telah diterapkan di berbagai bidang, mulai dari perencanaan jalur robotika, pengembangan permainan, hingga perutean jaringan (Antti, 2005; Daohong, 2023; Dongke et al., 2020; Huilai et al., 2010; Peter et al., 2011; Xiao & Hao, 2011; Xiaoli & Ping, 2009; Zuoyu & Xinduo, 2024). Berbagai variasi dari A*, seperti *Weighted A* telah diteliti untuk mempercepat waktu perencanaan dengan mengorbankan tingkat optimalitas(Archetti et al., 2022; J. & Wheeler, 2008; Wilt & Ruml, 2021). Telah diimplementasikan robot roda Omni berbasis Arduino Mega dengan integrasi desain mekanik, simulasi rangkaian listrik, dan penerapan AI untuk navigasi pada arena terstandarisasi Windasari, S. (2024). Metode BPSO diusulkan untuk mengoptimalkan kontrol PID secara adaptif dan efisien, menghasilkan solusi lebih stabil dibanding metode konvensional Suwoyo, H., Abdurohman, A., *et al.* 2022). Telah diusulkan desain multi-koil untuk sistem wireless power transfer yang menunjukkan peningkatan efisiensi rata-rata 7%

dibandingkan desain koil tunggal, dengan efisiensi maksimum mencapai 82% (Dama, M., *et al.* 2019). IoT merupakan konsep komunikasi berbasis internet yang menghubungkan perangkat melalui sensor, gateway, dan cloud, dengan cakupan lebih luas dibandingkan M2M (Bayu, M., et al. 2024).

Metode BPSO diusulkan untuk mengoptimalkan kontrol PID secara adaptif daSn efisien, menghasilkan solusi lebih stabil dibanding metode konvensional (Suwoyo, H., Abdurohman, A., *et al.* 2022) . Telah diimplementasikan estimasi SoC berbasis logika fuzzy pada Arduino dengan akurasi tinggi dan error minimal, mengungguli metode tegangan dan kesetimbangan kimia dalam kondisi pengaruh suhu 17) Ashidqi, M. et al. (2021). Telah dirancang dan diimplementasikan interkoneksi jaringan berbasis VPN yang terintegrasi dengan IPv6, sehingga meningkatkan efisiensi pengalamatan dan keamanan transmisi data antarjaringan (Frihadi. A. 2024). Telah disimulasikan arsitektur pusat data berbasis SDN dan AI pada platform NS-3, sehingga diperoleh peningkatan keamanan dan efisiensi tanpa mengurangi performa jaringan (A. Affandi., R. 2024). Telah disimulasikan arsitektur pusat data berbasis SDN dan AI pada platform NS-3, sehingga diperoleh peningkatan keamanan dan efisiensi tanpa mengurangi performa jaringan (A. Affandi., R. 2024).

Simulasi yang dikembangkan dalam penelitian ini memperluas pekerjaan-pekerjaan tersebut dengan menyediakan lingkungan interaktif, di mana pengguna dapat mengatur rintangan dan mengamati perilaku algoritma A* secara waktu nyata. Dengan menganalisis berbagai konfigurasi rintangan, kami menunjukkan bagaimana algoritma ini beradaptasi terhadap berbagai kondisi lingkungan.

3. Methodology

3.1 A* Algorithm Overview

Algoritma A* adalah algoritma pencarian jalur dan penelusuran graf yang banyak digunakan dalam AI, robotika, dan permainan (Y. K. A., 2020; Archetti et al., 2022; Daohong, 2023; J. & Wheeler, 2008; E. T. P. & B., 2019; Sharmad et al., 2022; Wilt & Ruml, 2021). Algoritma ini menghitung jalur terpendek antara dua node dalam sebuah grid atau graf, dengan menggabungkan biaya aktual ($g(n)$) dari node awal ke node saat ini dan estimasi biaya ($h(n)$) dari node saat ini ke tujuan. Fungsi total biaya, $f(n)$, didefinisikan sebagai:

$$f(n) = g(n) + h(n)$$

di mana *f(n)* adalah estimasi total biaya solusi termurah melalui node *n*, *g(n)* adalah biaya aktual dari node awal ke node *n*, dan *h(n)* adalah estimasi heuristik dari node *n* ke node tujuan.

Fungsi heuristik *h(n)* memiliki peran penting dalam menentukan efisiensi algoritma A*. Heuristik yang umum digunakan dalam lingkungan grid adalah *Manhattan Distance*, yang didefinisikan sebagai:

$$h(n) = |x_1 - x_2| + |y_1 - y_2|$$

Heuristik ini cocok digunakan pada lingkungan yang hanya mengizinkan gerakan empat arah (atas, bawah, kiri, kanan). Sebaliknya, jika gerakan diagonal diperbolehkan, maka dapat digunakan *Euclidean Distance*, yang dihitung sebagai:

$$h(n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Algoritma A* menggunakan dua himpunan: *open set*, yang berisi node-node yang akan dieksplorasi, dan *closed set*, yang menyimpan node-node yang sudah dieksplorasi. Pada setiap langkah, algoritma memilih node dengan nilai $f(n)$ terendah dari *open set* dan mengeksplorasi tetangganya, hingga tujuan tercapai atau *open set* kosong(Çakır & Ulukan, 2020; Maxim et al., 2003; Xiao & Hao, 2011).

3.2 A* Algorithm

Algoritma A* bekerja dengan langkah-langkah sebagai berikut (Çakır & Ulukan, 2020; Daohong, 2023; M., 2018; Qing, 2018):

1. **Inisialisasi**: Node awal dimasukkan ke dalam *open set*. Pada setiap iterasi, node dengan nilai $f(n)$ terendah dipilih untuk dieksplorasi.

2. **Ekspansi Node**: Jika node yang dipilih adalah tujuan, algoritma membangun ulang jalur dari awal ke tujuan dengan menelusuri kembali pointer induk. Jika tidak, algoritma mengevaluasi setiap tetangga dari node saat ini, dan melewati node yang sudah berada di *closed set*.

3. **Pembaruan Biaya**: Untuk setiap tetangga, algoritma menghitung nilai $g(n)$ sementara, yaitu biaya dari node awal ke tetangga:

$$g(n) = g(n_{sebelumnya}) + d$$

di mana d adalah jarak antara node saat ini dengan node tetangganya (biasanya bernilai 1 dalam grid). Jika nilai ini lebih kecil dibandingkan dengan nilai yang telah tercatat sebelumnya, maka node tetangga diperbarui dan dimasukkan ke dalam *open set*.

4. **Terminas**i: Jika *open set* kosong dan tujuan belum ditemukan, maka algoritma menyimpulkan bahwa tidak ada jalur yang tersedia.

Berikut adalah *pseudocode* dari algoritma A*:

```
FUNGSI A_Star(start, goal)
  open_set ← {start}
  came_from ← peta kosong

  g_score[start] ← 0
  f_score[start] ← heuristic(start, goal)

  SELAMA open_set tidak kosong LAKUKAN
    current ← node dengan f_score terendah
              dalam open_set

    JIKA current = goal MAKA
       KEMBALIKAN bangun_jalur(came_from,
                  current)
    AKHIR JIKA

    HAPUS current dari open_set

    UNTUK SETIAP neighbor dari current
    LAKUKAN
      tentative_g_score ← g_score[current]
                          + jarak
                        (current, neighbor)

      JIKA neighbor TIDAK ADA dalam g_score
      ATAU tentative_g_score <
      g_score[neighbor]
      MAKA
        came_from[neighbor] ← current
        g_score[neighbor] ←
                          tentative_g_score
        f_score[neighbor] ←
                           g_score[neighbor]
                         +heuristic
                          (neighbor, goal)

        JIKA neighbor TIDAK ADA dalam
        open_setMAKA
          TAMBAHKAN neighbor KE open_set
        AKHIR JIKA
      AKHIR JIKA
    AKHIR UNTUK
  AKHIR SELAMA

  KEMBALIKAN gagal // jalur tidak ditemukan
AKHIR FUNGSI
```

Gabungan dari *g(n)* dan *h(n)* memungkinkan A* untuk mengeksplorasi grid secara cerdas, meminimalkan jumlah node yang dikunjungi sambil memastikan jalur terpendek ditemukan. Performa algoritma ini dapat ditingkatkan lebih lanjut menggunakan variasi seperti *Bidirectional* A*.

4. Simulation Design

Bagian ini menjelaskan struktur dan operasi dari simulasi pencarian jalur menggunakan algoritma A*. Termasuk di dalamnya konfigurasi lingkungan grid, model interaksi pengguna, serta visualisasi perilaku algoritma dalam merespons berbagai tata letak rintangan.

4.1 Pengaturan Grid dan Interaksi Pengguna

Simulasi berbasis pada grid dua dimensi berukuran 25×25, di mana setiap sel dapat merepresentasikan ruang kosong, rintangan, titik awal (*start*), atau titik tujuan (*goal*). Pengguna berinteraksi dengan simulasi melalui antarmuka berbasis teks yang dipicu dengan menekan tombol **SPACE**. Setelah diaktifkan, pengguna dapat mengetik atau menempelkan perintah terstruktur untuk mengatur skenario.

Format perintah adalah sebagai berikut:

`S=<posisi>`: Menetapkan posisi awal, contoh: `S=A1`

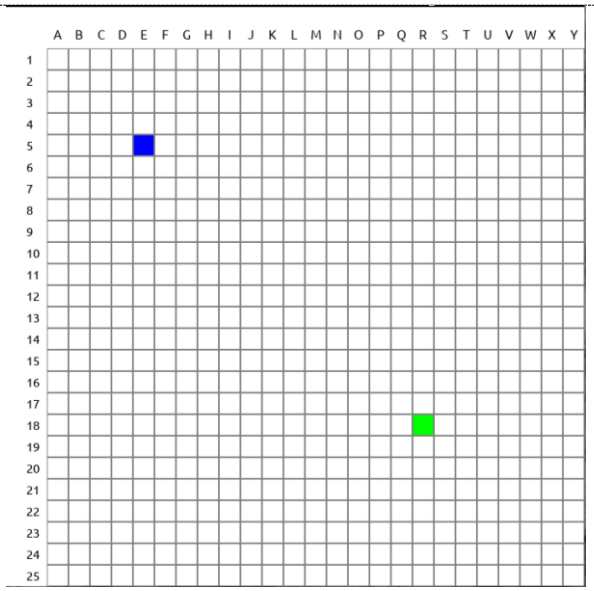
`G=<posisi>`: Menetapkan posisi tujuan, contoh: `G=Y25`

`O=<pos1>,<pos2>,...`: Menambahkan rintangan di beberapa posisi, contoh: `O=B2,C2,C3`

`R=<pos1>,<pos2>,...`: Menghapus rintangan tertentu, contoh: `R=C3,D4`

Setelah memasukkan string perintah, pengguna menekan **ENTER** untuk menerapkan perubahan pada grid. Pengguna kemudian dapat menjalankan algoritma A* dengan menekan tombol **R**. Algoritma berjalan secara *real-time* dan divisualisasikan langsung pada kanvas grid.

Gambar 1 menggambarkan kondisi awal simulasi, di mana posisi awal dan tujuan telah ditentukan, tetapi belum ada rintangan.



Gambar 1. Pengaturan Awal Grid dengan Titik Awal (Biru) dan Titik Tujuan (Hijau), Tanpa Rintangan.

4.2 Visualisasi Pencarian Jalur

Untuk mendukung pemahaman intuitif tentang perilaku algoritma A*, simulasi menggunakan sistem visualisasi warna yang dinamis. Sistem ini mencerminkan status node yang berubah selama proses pencarian, memungkinkan pengguna untuk mengamati bagaimana algoritma mengevaluasi berbagai jalur dan merespons rintangan.

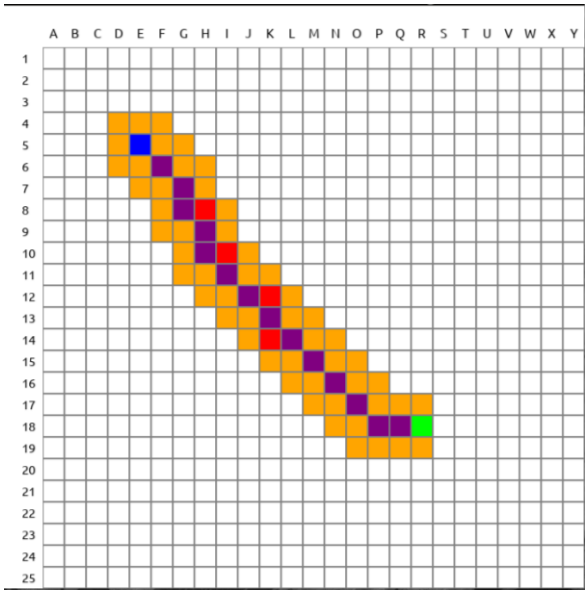
Skema warna yang digunakan dalam visualisasi adalah sebagai berikut:

- **Biru**: Titik awal (*start node*)
- **Hijau**: Titik tujuan (*goal node*)
- **Merah**: Rintangan atau penghalang yang harus dihindari
- **Oranye**: Node yang telah dikunjungi dan sedang dieksplorasi oleh algoritma
- **Ungu**: Jalur akhir yang dipilih setelah tujuan berhasil dicapai

Umpan balik visual ini diberikan secara waktu nyata, memberikan wawasan kepada pengguna tentang strategi pencarian jalur, termasuk bagaimana algoritma menavigasi sekitar rintangan dan memprioritaskan jalur

berdasarkan akumulasi biaya dan estimasi heuristik.

Contoh visualisasi ini ditunjukkan pada Gambar 2, di mana rintangan jarang ditempatkan pada grid. Eksplorasi jalur ditampilkan dengan warna oranye, dan jalur optimal akhir ditandai dengan warna ungu.



Gambar 2. *Visualisasi Algoritma A** Saat Mengeksplorasi Node dengan Rintangan yang Jarang.

4.3 Fungsi Kontrol dan Umpan Balik *Real-Time*

Simulasi menyediakan berbagai pintasan keyboard untuk memodifikasi lingkungan dan menjalankan kembali algoritma. Tabel berikut merangkum fungsi-fungsi yang tersedia:

Table 1. Kontrol Keyboard dan Deskripsi Fungsional untuk Simulasi Algoritma A*

Tombol	Function
SPASI	Mengaktifkan mode input
ENTER	Menerapkan input yang diketik/tempel
CTRL + V	Menempelkan string perintah dari clipboard
R	Menjalankan algoritma pencarian jalur A*
P	Menghapus jalur saja (rintangan, start, dan goal tetap)

Setelah algoritma dijalankan, simulasi juga menampilkan metrik kinerja di konsol, mencakup:

- *Path Length*: Jumlah langkah dalam jalur akhir
- *Nodes Explored*: Total sel grid yang dikunjungi.
- *Obstacle Count*: Total rintangan yang ditempatkan
- *Execution Time:* Durasi waktu proses algoritma dalam detik

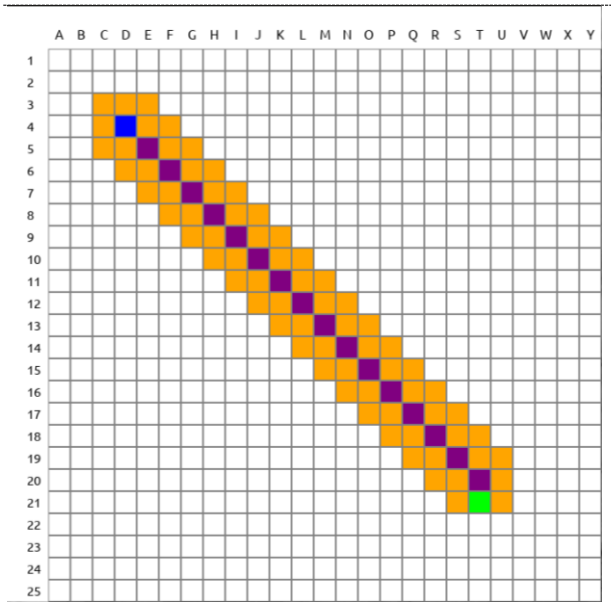
Fitur ini memungkinkan pengguna membandingkan performa pencarian jalur secara kuantitatif untuk berbagai konfigurasi grid.

5. Hasil

5.1 Eksperimen 1: Pencarian Jalur Dasar Tanpa Rintangan

Eksperimen awal ini bertujuan untuk menentukan performa dasar algoritma A* dalam lingkungan ideal tanpa rintangan apa pun. Dalam skenario pertama, tidak ada rintangan yang ditempatkan di grid. Algoritma A* menemukan jalur terpendek berupa garis lurus dari titik awal ke titik tujuan. Simulasi ini mengonfirmasi bahwa, tanpa adanya hambatan, A* bekerja secara optimal dengan eksplorasi minimum.

Gambar 3 berikut menggambarkan jalur langsung yang diidentifikasi oleh A* ketika tidak ada rintangan antara node awal dan node tujuan:



Gambar 3. Jalur Lurus Terpendek dari Titik Awal ke Titik Tujuan dalam Lingkungan Tanpa Rintangan.

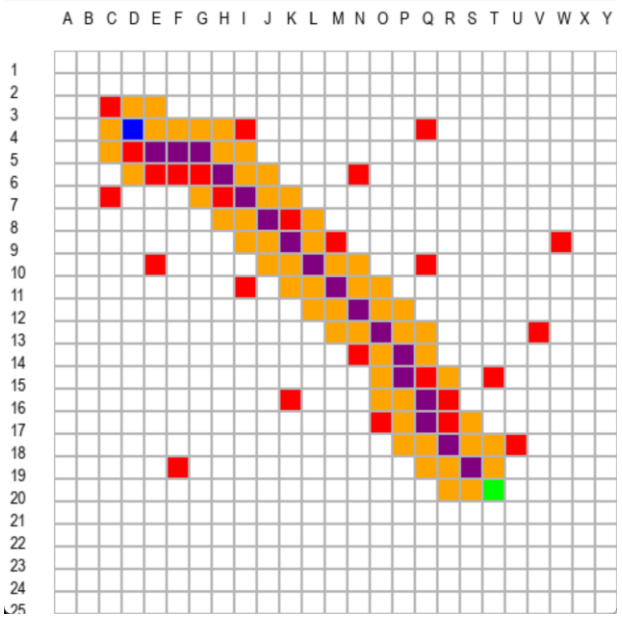
Metrik kinerja pada eksperimen ini:

- Jumlah node yang dieksplorasi: 17
- Panjang jalur: 16 langkah
- Waktu eksekusi: 0,36 detik

5.2 Eksperimen 2: Pencarian Jalur dengan Rintangan yang Jarang

Eksperimen kedua dilakukan untuk menilai bagaimana algoritma beradaptasi terhadap penghalang minor. Dalam skenario ini, rintangan ditempatkan secara jarang di grid. Algoritma A* berhasil menavigasi di sekitar rintangan ini, memilih jalur yang sedikit lebih panjang tetapi tetap meminimalkan jumlah eksplorasi. Node yang dieksplorasi namun tidak termasuk dalam jalur akhir ditampilkan dalam warna oranye.

Visualisasi dari skenario ini ditunjukkan pada Gambar 4, di mana penghalang jarang berhasil dihindari oleh pencarian A*:



Gambar 4. Jalur yang Dibentuk oleh Algoritma A* Saat Menghadapi Rintangan yang Jarang pada Grid.

Metrik kinerja pada skenario in:

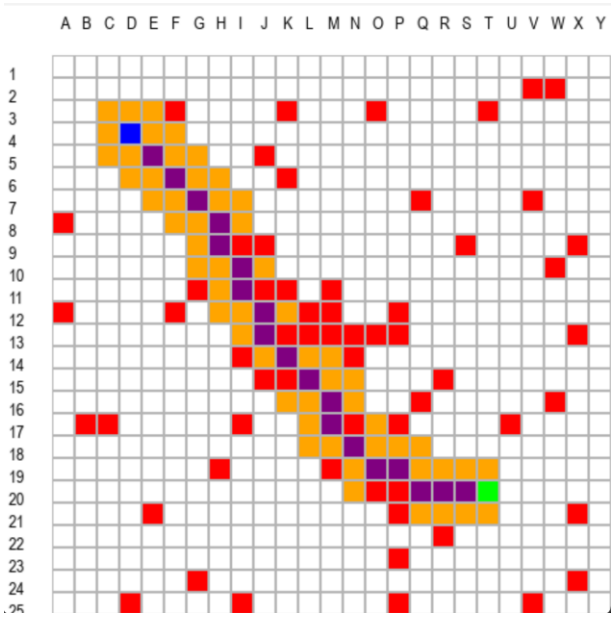
- Jumlah rintangan: 26
- Jumlah node yang dieksplorasi: 19
- Panjang jalur: 18 langkah
- Waktu eksekusi: 0,52 detik

5.3 Eksperimen 3: Pencarian Jalur dengan Rintangan yang Padat

Pada eksperimen ketiga, dianalisis bagaimana algoritma bekerja dalam kondisi kompleksitas maksimum. Skenario ini melibatkan kepadatan rintangan tinggi yang terkonsentrasi di pusat grid, menjadikannya jauh lebih menantang bagi algoritma untuk menemukan jalur yang valid.

Dalam pengujian akhir ini, tujuannya adalah mengevaluasi seberapa baik algoritma A* menangani tingkat kerumitan dan hambatan yang tinggi. Rintangan yang ditempatkan secara padat di sekitar pusat grid memaksa algoritma untuk mengeksplorasi bagian besar grid sebelum menemukan jalur yang layak. Proses eksplorasi secara visual menjadi lebih luas karena banyaknya penghalang, namun algoritma tetap berhasil mengidentifikasi jalur terpendek yang tersedia.

Visualisasi dari skenario ini ditampilkan pada Gambar 5 berikut, yang menunjukkan bagaimana algoritma harus berkeliling melewati inti penghalang yang padat sebelum mencapai titik tujuan:



Gambar 5. Jalur yang Dibentuk oleh Algoritma A* Saat Menghadapi Rintangan Padat pada Grid.

Metrik kinerja dari eksperimen ini:

- Jumlah rintangan: 60
- Jumlah node yang dieksplorasi: 21
- Panjang jalur: 20 langkah
- Waktu eksekusi: 0,61 detik

6. Diskusi

6.1 Efisiensi Algoritma

Untuk mengevaluasi lebih lanjut performa algoritma A*, dilakukan analisis perbandingan dengan algoritma Dijkstra dan *Breadth-First Search* (BFS) menggunakan lingkungan grid yang sama.

Analisis ini memberikan wawasan tentang bagaimana masing-masing algoritma bekerja di bawah kondisi serupa, khususnya dalam hal jumlah node yang dieksplorasi, panjang jalur, dan waktu eksekusi. Tabel berikut merangkum hasil perbandingan performa:

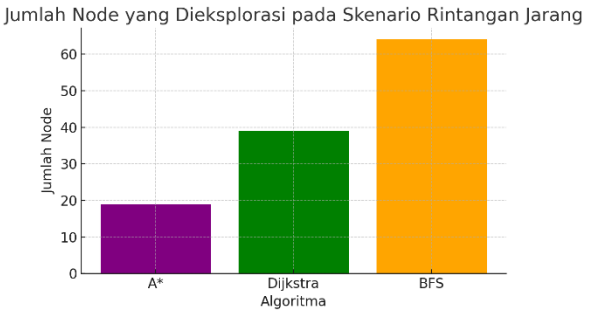
Tabel 2. Perbandingan Kinerja Algoritma A*, Dijkstra, dan BFS dalam Lingkungan Grid.

Alg.	Skenario	Node yang Dieksplorasi	Panjang Jalur (langkah)	Waktu Eksekusi (detik)
A*	Tanpa Rintangan	17	16	0,36
A*	Rintangan Jarang	19	18	0,52
A*	Dense Obstacles	21	20	0,61
Dijkstra	Rintangan Padat	39	18	0,73
BFS	Rintangan Jarang	64	18	1,12

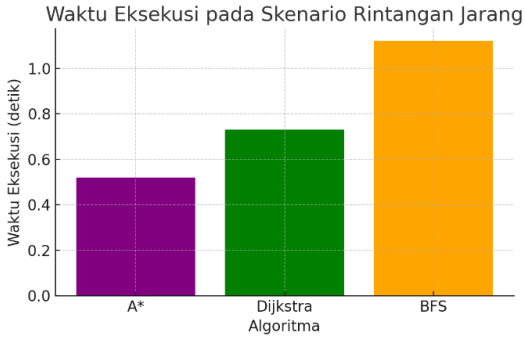
Hasil ini menunjukkan bahwa algoritma A* jauh lebih efisien dalam hal jumlah node yang dieksplorasi dan waktu eksekusi, berkat pencariannya yang dipandu oleh fungsi heuristik. Algoritma Dijkstra mengeksplorasi lebih banyak node karena ia mengevaluasi semua jalur terpendek secara merata, sementara BFS, yang bersifat *uninformed*, melakukan eksplorasi yang paling luas.

Secara kuantitatif, peningkatan waktu eksekusi dan jumlah node yang dieksplorasi berkorelasi dengan banyaknya rintangan yang ditempatkan. Dari waktu dasar 0,36 detik tanpa rintangan, waktu meningkat menjadi 0,61 detik ketika rintangan padat digunakan. Hal ini menunjukkan adanya hubungan linier hingga eksponensial antara kompleksitas rintangan dengan biaya komputasi, yang menegaskan pentingnya penggunaan heuristik adaptif dalam skenario yang lebih kompleks.

Kemudian lebih lanjut Gambar 6 dan Gambar 7 berikut, menampilkan perbandingan eksplorasi dan waktu eksekusi untuk skenario rintangan jarang oleh A*, Dijkstra, dan BFS pada Skenario Rintangan Jarang.



Gambar 6. Perbandingan Jumlah Node yang Dieksplorasi oleh A*, Dijkstra, dan BFS pada Skenario Rintangan Jarang.



Gambar 7. Perbandingan Waktu Eksekusi oleh A*, Dijkstra, dan BFS pada Skenario Rintangan Jarang.

Eksperimen-eksperimen ini menunjukkan bahwa algoritma A* sangat efisien dalam menyelesaikan masalah pencarian jalur berbasis grid. Dengan menggabungkan biaya aktual $g(n)$ dan estimasi heuristik $h(n)$ secara cerdas, A* melampaui algoritma pencarian tak terinformasi seperti BFS dan DFS dalam hal waktu dan jumlah node yang dieksplorasi. Bahkan ketika rintangan hadir, A* tetap menemukan jalur optimal dengan eksplorasi yang lebih sedikit dibandingkan Dijkstra pada kondisi serupa.

6.2 Limitasi

Meskipun A* sangat efektif di lingkungan statis, algoritma ini menghadapi keterbatasan dalam skenario dinamis atau *real-time*. Simulasi saat ini mengasumsikan bahwa rintangan bersifat statis, yang menyederhanakan masalah tetapi membatasi penerapannya dalam robotika dunia nyata atau sistem navigasi otonom.

Pekerjaan selanjutnya dapat mengeksplorasi algoritma pencarian jalur dinamis yang mampu beradaptasi secara waktu nyata, seperti D* atau *Rapidly-exploring Random Trees* (RRT). Selain itu, pendekatan hibrida, misalnya menggabungkan A* dengan teknik pembelajaran mesin untuk memprediksi perilaku rintangan, berpotensi meningkatkan pencarian jalur dalam lingkungan yang berubah-ubah.

7. Kesimpulan

Tulisan ini menyajikan eksplorasi berbasis simulasi terhadap algoritma pencarian jalur A* dalam lingkungan berbasis grid. Melalui input interaktif dari pengguna dan visualisasi waktu nyata, simulasi ini menunjukkan ketangguhan A* dalam menavigasi rintangan dan menemukan jalur optimal secara efisien. Eksperimen yang dilakukan menunjukkan efektivitas algoritma ini dalam lingkungan sederhana maupun kompleks, meskipun terdapat tantangan dalam menerapkannya pada skenario yang dinamis.

Pekerjaan di masa depan sebaiknya berfokus pada integrasi elemen dinamis ke dalam simulasi, seperti rintangan bergerak, serta mengeksplorasi alternatif heuristik untuk menghadapi lingkungan yang lebih kompleks.

Referensi

A., S. (1993). *Optimal and Efficient Path Planning for Unknown and Dynamic Environmen ts*.

A., Y. K. (2020). A Review on Informed Search Algorithms for Video Games Pathfinding. *International Journal of Advanced Trends in Computer Science and Engin Eering*. https://doi.org/10.30534/ijatcse/2020/4293 2020

Adeel, J. (2013). *Understanding Dijkstra's Algorithm*. https://doi.org/10.2139/ssrn.2340905

Alec, Z. M. T., Celeste, J. G. P., Mark, C. R. B., Richard, C. R., & Dan, M. A. C. (2022). Enhancement of Dijkstra Algorithm for Finding Optimal Path. *International Journal of Research Publications*. https://doi.org/10.47119/ijrp10010216202 23299

Ángel, M., Abdulla, A.-K., David, M., & A., D. L. E. (2021). Trajectory planning for multi-

robot systems: Methods and applications. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2021.114660

Antti, A. (2005). *Extensions and applications of the A* algorithm*.

Archetti, A., Cannici, M., & Matteucci, M. (2022). Neural Weighted A*: Learning Graph Costs and Heuristics with Different iable Anytime A*. In *Lecture Notes in Computer Science* (pp. 596–610). Springer International Publishing. https://doi.org/10.1007/978-3-030-95467-3_43

Başara, E. (2024). Path Planning and Motion Control: Current Methods, Challenges, and Fut ure Perspectives. *Human Computer Interaction*, *8*(1), 79. https://doi.org/10.62802/jv4qjh57

Çakır, E., & Ulukan, Z. (2020). A* Algorithm Under Single-Valued Neutrosophic Fuzzy Environment. In *Advances in Intelligent Systems and Computing* (pp. 302–310). Springer International Publishing. https://doi.org/10.1007/978-3-030-51156-2_36

Daohong, L. (2023). Research of the Path Finding Algorithm A* in Video Games. *Highlights in Science Engineering and Technology*. https://doi.org/10.54097/hset.v39i.6642

Dongke, R., Junfeng, M., Fulun, P., Hongguang, L., & Junfeng, M. (2020). SUMMARY OF RESEARCH ON PATH PLANNING BASED ON A* ALGORITHM. *The 8th International Symposium on Test Automation & Instrumentation (ISTAI 2020)*. https://doi.org/10.1049/icp.2021.1291

Fahed, J., & Mohammed, H. (2020). Exploiting Obstacle Geometry to Reduce Search Time in Grid-Based Pathf inding. *Symmetry*. https://doi.org/10.3390/sym12071186

Harika, R. (2013). *PATH FINDING - Dijkstra's and A* Algorithm's*.

Huilai, Z., Li, Z., Hua, L., Chaonan, W., Zening, Q., & Youtian, Q. (2010). Optimized Application and Practice of A* Algorithm in Game Map Path-Fi nding. *2010 10th IEEE International Conference on Computer and Information Te Chnology*. https://doi.org/10.1109/CIT.2010.365

J., T., & Wheeler, R. (2008). Faster than Weighted A*: An Optimistic Approach to Bounded Suboptimal Search. *International Conference on Automated Planning and Scheduling*.

Langyu, S. (2024). Research on Path Planning Method based on Graph Search Algorithm. *Highlights in Science Engineering and Technology*. https://doi.org/10.54097/s0t5zj43

M., T. (2018). *Performance Evaluation of Competing Data Structures*.

Maxim, L., Geoffrey, J. G., & S., T. (2003). ARA*: Anytime A* with Provable Bounds on Sub-Optimality. *Neural Information Processing Systems*.

N., C., & M., T. (2013). *Analysis of Working of Dijkstra and A* to Obtain Optimal Path*.

N., S. A., W., B., M., A., S., O., & S., A. S. (2022). A Comprehensive Overview of Classical and Modern Route Planning Algori thms for Self-Driving Mobile Robots. *Journal of Robotics and Control (JRC)*. https://doi.org/10.18196/jrc.v3i5.14683

Nathan, R. S. (2018). AI education matters: teaching search algorithms. *SIGAI*. https://doi.org/10.1145/3284751.3284757

O., A. G., & D., R. A. (2020). Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs. *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*. https://doi.org/10.1109/ICMCECS47690.2020.240873

P., E. T., & B., C. (2019). Path Planning Algorithms and Their Use in Robotic Navigation Systems. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1207/1/012018

P., H., N., N., & B., R. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. https://doi.org/10.1109/TSSC.1968.30013

6

Peter, Y., Neil, B., R., H., & J., S. (2011). Any-Angle Path Planning for Computer Games. *Artificial Intelligence and Interactive Digital Entertainment Conferen Ce*. https://doi.org/10.1609/aiide.v7i1.12445

Qing, C. (2018). *Exploiting Problem Structure in Pathfinding*.

Roni, S., Tamar, K., Ariel, F., & R., H. (2010). Using Lookaheads with Optimal Best-First Search. *AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v24i1.7559

Sara, L. P., Fadel, R. A., Yahya, N. H., & Fikri, Z. (2022). A Review of Pathfinding in Game Development. *[CEPAT] Journal of Computer Engineering: Progress, Application and Tec Hnology*. https://doi.org/10.25124/cepat.v1i01.4863

Shahid, S. (2024). Artificial Intelligence in Path Planning for Autonomous Robots: A Revi ew. *Metaheuristic Optimization Review*. https://doi.org/10.54216/mor.020204

Sharmad, R. L., Grace, J., Jani, A., & L., I. I. (2022). A Systematic Review and Analysis of Intelligence-Based Pathfinding Alg orithms in the Field of Video Games. *Applied Sciences*. https://doi.org/10.3390/app12115499

Sidharth, P., Abhishek, J., A., D., & Irfan, A. S. (2022). An Exhaustive Approach Orchestrating Negative Edges for Dijkstra's Alg orithm. *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*. https://doi.org/10.1109/i2ct54291.2022.9824795

Wilt, C., & Ruml, W. (2021). When Does Weighted A* Fail? *Proceedings of the International Symposium on Combinatorial Search*, *3*(1), 137–144. https://doi.org/10.1609/socs.v3i1.18250

Xiao, C., & Hao, S. (2011). *A*-based Pathfinding in Modern Computer Games*.

Xiaoli, G., & Ping, G. (2009). A* Algorithm Analysis and Optimization: In Network Game Design. *2009 International Conference on Computational Intelligence and Softwa Re Engineering*. https://doi.org/10.1109/CISE.2009.536675

7

Yunbo, Z. (2024). Optimization of the mobile robot's path searching based on the A-star algorithm. *Applied and Computational Engineering*. https://doi.org/10.54254/2755-2721/67/20240463

Zuoyu, W., & Xinduo, F. (2024). A systematic review and analysis of A-star pathfinding algorithms and its variations. *International Conference on Signal Processing and Machine Learning*. https://doi.org/10.1117/12.3027132

Windasari, S. (2024). Designing An Omni Wheel Robot. Jurnal Ekselenta-Jurnal Ilmiah Fakultas Teknik, 1(1), 40-48.

Suwoyo, H., Abdurohman, A., Li, Y., Adriansyah, A., Tian, Y., & Hajar, M. H. I. (2022). The Role of Block Particles Swarm Optimization to Enhance The PID-WFR Algorithm. International Journal of Engineering Continuity, 1(1), 9-23.

Baskoro, B. (2024). Pemanfaatan IoT Sebagai Teknologi Terkini di Kehidupan Masyarakat. Jurnal Ekselenta-Jurnal Ilmiah Fakultas Teknik, 1(1).

Ashidqi, M. D., Anwar, M., Hermanu, C., Ramelan, A., & Adriyanto, F. (2021). Fuzzy Logic Implementation for Accurate Electric Car Battery SOC measurement. Jurnal Nasional Teknik Elektro dan Teknologi Informasi, 10(3), 257-264.

Dama, M., & Alaydrus, M. (2019, October). Analysis of multi coils in misalignment conditions in the WPT system. In 2019 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) (pp. 20-23). IEEE.

Frihadi,. A. (2024). Pemanfaatan IoT Sebagai Teknologi Terkini di Kehidupan Masyarakat. Jurnal Ekselenta-Jurnal Ilmiah Fakultas Teknik, 1(1),

Adi Affandi Rotib (2024). Pusat Data dan Layanan Cloud Center: Jaringan Protokol dan Manajement. Jurnal Ekselenta-Jurnal

Ilmiah Fakultas Teknik, 1(1),

Kusumo, probo; rokhmah, a. K. (2024). Perbaikan tata letak fasilitas departemen produksi cv. Decorus menggunakan systematic layout planning untuk meningkatkan produktivitas dan mengurangi. *Jurnal ekselenta*, *1*(1).

Kusumo, Probo; Rokhmah, A. K. (2024). Perancangan pengembangan meja kerja pengolahan lele yang ergonomis menggunakan metode rasional. *Jurnal Ekselenta*, *1*(1).

Kusumo, Probo; Rokhmah, A. K. (2024). Perancangan pengembangan produk toolbox dengan pendekatan ergonomi dan antopometri. *Jurnal Ekselenta*, *1*(1), 1–7.

Sari, E. M., Mulyani, R., & Saepullah, a. (2024). Mengukur partnering dalam design-bid-build (dbb). *Jurnal ekselenta*, *1*(1), 1–9.

Saepullah, A. (2024). Implementasi sistem barcode terintegrasi dengan sap erp pada sistem persediaan pt al 1. *Jurnal Ekselenta*, *1*(1), 1–6.

Saepullah, A. (2024). Analisis Kecacatan Produk Pada Hasil Pengelasan Dengan Menggunakan Metode FMEA (Failure Mode Effect Analysis). *Jurnal Ekselenta*, *1*(1).https://doi.org/10.32672/jse.v7i2.3853.

Kusumo, P., Setyaningrum, R., & Tjahyono, R. (2022). Design of an Ergonomic Crackers Dryer to Increase Production Productivity at Rahayu Krupuk SME. *Proceedings of the 4th Asia Pacific Conference on Research in Industrial and Systems Engineering*, 31–34.https://doi.org/10.1145/3468013.3468305.

Kusumo, P., Setyaningrum, R., & Tjahyono, R. (2021). Perancangan Pengering Kerupuk "Smart Fuse Water Dryer" Yang Ergonomis Untuk Meningkatkan Produktivitas Produksi Di Ukm Rahayu Kerupuk. *Jurnal Simetris*, *12* (2).

Karyadi (2011). Perancangan Sistem Pokayoke Bolt Modul Airbag Studi Kasus di Proses Shower Test Final Inspection 4W PT. Suzuki Indomobil Motor. https://repository.mercubuana.ac.id/id/eprint/19695.

Karyadi (2024). Implementasi framework PM3 sebagai alat bantu diagnostik untuk mengidentifikasi kebutuhan pengembangan organisasi PT.GBSI. http://repository.trisakti.ac.id/usaktiana/index.php/home/detail/detail_koleksi/1/THE/th_terbit/00000000000000052643/2013.

Rusman Karyadi; Hetharia Dorina (2025). Quality improvement through 8D methodology: an automotive industry case study. Vol. 17 No. 1 (1-11). https://publikasi.mercubuana.ac.id/index.php/oe/article/view/20205.