Mengoptimalkan Rute Ambulans untuk Efisiensi Menggunakan Algoritma A*

M. Zaidan Sa'dun R. - 13522146[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]135522146@std.stei.itb.ac.id

***Abstract*—Optimasi rute ambulans sangat penting untuk meningkatkan efisiensi respons darurat dan mengurangi waktu tanggap. Artikel ini menyajikan penerapan algoritma A* untuk mengoptimalkan rute ambulans di lingkungan perkotaan. Algoritma A*, yang dikenal efisien dalam pencarian jalur dan traversal graf, digunakan untuk mengidentifikasi rute yang paling efisien dengan mempertimbangkan data lalu lintas waktu nyata dan kondisi jalan. Pendekatan kami mencakup analisis komprehensif terhadap pola grid perkotaan, fungsi heuristik, dan perhitungan biaya untuk memastikan rute terpendek dan tercepat yang dipilih. Hasil penelitian menunjukkan peningkatan signifikan dalam waktu tanggap, yang memvalidasi efektivitas algoritma A* dalam skenario darurat nyata. Penelitian ini memberikan wawasan berharga dan kerangka kerja praktis bagi penyedia layanan darurat yang ingin meningkatkan efisiensi operasional mereka melalui teknik algoritmik canggih.**

Keywords—optimasi rute ambulans; algoritma A*; pencarian jalur; respons darurat; optimasi grid perkotaan; data lalu lintas; fungsi heuristik; optimasi rute; efisiensi operasional; layanan darurat.

I. PENDAHULUAN

Dalam situasi darurat, respons yang cepat dan efisien dari layanan ambulans dapat menjadi penentu antara hidup dan mati. Oleh karena itu, optimalisasi rute ambulans menjadi salah satu aspek yang sangat penting dalam manajemen layanan darurat. Dengan pertumbuhan populasi perkotaan dan meningkatnya kemacetan lalu lintas, menemukan rute tercepat dan terpendek bagi ambulans menjadi semakin menantang. Teknologi modern dan algoritma canggih kini memungkinkan kita untuk mengatasi tantangan ini dengan lebih efektif. Oleh karena itu, aspek permasalahan ini sangat mungkin diselesaikan. Pencarian rute tercepat dapat dilakukan menggunakan algoritma A*.

Algoritma A* adalah salah satu metode yang paling terkenal dan efisien untuk pencarian jalur dan traversal graf. Algoritma ini menggunakan fungsi heuristik untuk memperkirakan biaya dari titik awal ke titik tujuan, yang memungkinkan untuk menemukan rute optimal dengan lebih cepat dibandingkan metode lain. Dalam konteks optimasi rute ambulans, algoritma A* dapat digunakan untuk mempertimbangkan berbagai faktor seperti kondisi lalu lintas waktu nyata, jenis jalan, dan kemungkinan rintangan di sepanjang jalan.

Gambar 1. Illustrasi Rute Ambulans menggunakan A*

Makalah ini bertujuan untuk mengaplikasikan algoritma A* dalam konteks optimasi rute ambulans di lingkungan perkotaan. Kami akan mengeksplorasi bagaimana algoritma ini dapat diimplementasikan untuk meningkatkan efisiensi operasional layanan ambulans, dengan fokus pada pengurangan waktu tanggap dan peningkatan akurasi rute yang dipilih. Pendekatan ini melibatkan analisis mendalam terhadap pola grid perkotaan, pengembangan fungsi heuristik yang relevan, dan integrasi data lalu lintas waktu nyata. Namun, pada makalah ini kami menggunakan konsep secara dasar terlebih dahulu dan tidak secara langsung menerapkan pada perkotaan.

Studi ini juga akan membahas berbagai tantangan yang dihadapi dalam implementasi algoritma A* untuk optimasi rute ambulans. Ini termasuk penanganan data lalu lintas yang dinamis, penyesuaian algoritma untuk lingkungan perkotaan yang kompleks, dan pengujian terhadap skenario darurat yang beragam. Dengan mengatasi tantangan-tantangan ini, kami berharap kerangka awal pemecahan masalah kami dapat menyediakan kerangka kerja yang dapat diterapkan oleh penyedia layanan darurat untuk meningkatkan efisiensi dan efektivitas operasional mereka.

Selain itu, penelitian ini akan mengkaji bagaimana algoritma A* dapat diintegrasikan dengan sistem informasi geografis (GIS) dan teknologi pemetaan modern untuk menyediakan solusi yang lebih holistik. Dengan memanfaatkan data GPS dan sensor lalu lintas, kami akan menunjukkan bagaimana algoritma ini dapat beradaptasi dengan perubahan kondisi lalu lintas secara waktu nyata, sehingga memberikan rute yang paling optimal setiap saat.

Dalam makalah ini, penelitian ini tidak hanya menawarkan solusi teknis untuk optimasi rute ambulans, tetapi juga memberikan wawasan tentang bagaimana teknologi algoritmik dapat digunakan untuk meningkatkan layanan publik secara keseluruhan. Melalui penerapan algoritma A*, kami berharap dapat menyumbang pada peningkatan kualitas respons darurat, yang pada akhirnya dapat menyelamatkan lebih banyak nyawa.

II. DASAR TEORI

A. *Algoritma A**

Algoritma A* adalah salah satu algoritma pencarian jalur yang paling populer dan efektif yang digunakan dalam berbagai aplikasi mulai dari game komputer hingga robotika dan navigasi. Algoritma ini merupakan pengembangan dari algoritma Dijkstra, yang menemukan jalur terpendek antara dua titik dalam graf, dengan menambahkan fungsi heuristik untuk memperkirakan jarak yang tersisa ke tujuan. Dengan kombinasi ini, A* mampu mengoptimalkan pencarian jalur dengan lebih efisien, baik dalam hal waktu maupun sumber daya komputasi. Algoritma A* bekerja berdasarkan prinsip pencarian graf. Graf terdiri dari simpul-simpul (nodes) yang saling terhubung oleh sisi-sisi (edges). Setiap sisi memiliki bobot yang menunjukkan biaya atau jarak antara dua simpul yang terhubung. Dalam konteks optimasi rute ambulans, simpul-simpul dapat merepresentasikan lokasi-lokasi di peta, dan sisi-sisi mewakili jalan-jalan yang menghubungkan lokasi tersebut, dengan bobot yang bisa berupa jarak fisik, waktu tempuh, atau hambatan lainnya. Algoritma A* menggunakan dua fungsi utama, yaitu fungsi g(n) yang mengukur biata dari titik awal ke simpul n dan fungsi h(n) yang memperkirakan biaya dari simpul n ke tujuan. Fungsi evaluasi, $f(n)$, untuk algoritma pencarian A* adalah sebagai berikut:

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

Algoritma ini merupakan penggabungan algoritma *Uniform Cost* (UCS) dan *Greedy Best-First Search* (GBFS). UCS mendapatkan nilai f(n) menggunakan fungsi g(n) sedangkan GBFS mendapatkan nilai f(n) menggunakan fungsi h(n).

Keberhasilan algoritma A* sangat bergantung pada pemilihan fungsi heuristik yang tepat. Fungsi heuristik harus admissible, artinya tidak boleh melebihi biaya sebenarnya dari simpul saat ini ke tujuan. Beberapa contoh fungsi heuristik yang umum digunakan meliputi:

- Manhattan Distance, Digunakan untuk grid yang memungkinkan pergerakan hanya dalam arah vertikal atau horizontal.

  $h(n) = |x_{goal} - x_{current}| + |y_{goal} - y_{current}|$

- Euclidean Distance, Digunakan untuk pergerakan di ruang yang memungkinkan pergerakan diagonal.

  $h(n) = \sqrt{}\ ((x_{goal} - x_{current})2 + (y_{goal} - y_{current})2)$

- Chebyshev Distance, digunakan untuk grid yang memungkinkan pergerakan vertical, horizontal, dan diagonal.

  $h(n) = \max(|x_{goal} - x_{current}|, |y_{goal} - y_{current}|)$

Pemilihan fungsi heuristik yang tepat dapat mempercepat pencarian jalur dan mengurangi beban komputasi.

Algoritma A* secara garis besar dapat dijelaskan seperti berikut:

1. Masukkan node awal ke openlist
2. Loop langkah-langkah dibawah ini:
   a. Cari node (n) dengan nilai f(n) yang paling kecil dalam open list, dan node ini sekarang menjadi current node.
   b. Keluarkan current node dari openlist dan masukkan ke open list
   c. Untuk setiap tetangga dari current nodej ika tidak dapat dilalui atau sudah ada dalam close list , abaikan. Jika belum ada di open list buat current node parent dari node tetangga ini, simpan nilai f, g, dan h dari node ini. Jika sudah ada di openlist cek apabila node tetangga ini lebih baik menggunakan nilai g sebagai ukuran. Jika lebih baik ganti parent dari node ini di opentlist menjadi current node, lalu kalkulasikan ulang nilai g dan fdari node ini.
   d. Hentikan looping jika node tujuan telah ditambah ke openlist yang berarti rute ditemukan dan jika belum menemukan node akhir (tujuan) sementara openlist kosong atau berarti tidak ada rute.

B. *Perencanaan Rute*

Perencanaan rute (route planning) adalah proses menentukan jalur terbaik dari satu titik ke titik lain, yang dapat mencakup beberapa perhentian atau tujuan. Dalam konteks ambulans, perencanaan rute melibatkan penentuan jalur tercepat dan paling efisien dari lokasi ambulans saat ini ke lokasi darurat, dengan mempertimbangkan berbagai faktor seperti kondisi lalu lintas, rintangan jalan, dan kebijakan jalan tertentu (misalnya, jalan satu arah).

Langkah-langkah perencanaan rute:

1. Pemetaan Lokasi dan Jaringan Jalan

   Langkah pertama dalam perencanaan rute adalah pemetaan lokasi dan jaringan jalan dalam bentuk graf. Setiap lokasi atau persimpangan jalan direpresentasikan sebagai simpul, dan jalan yang menghubungkan lokasi tersebut direpresentasikan sebagai sisi dengan bobot tertentu.

2. Pengumpulan Data Lalu Lintas

   Data lalu lintas waktu nyata sangat penting dalam perencanaan rute untuk ambulans. Data ini mencakup informasi tentang kemacetan, kecepatan lalu lintas, kecelakaan, dan rintangan jalan lainnya. Data ini dapat diperoleh dari sensor lalu lintas, GPS, dan laporan pengguna jalan.

3. Penentuan Heuristik yang Tepat

Pemilihan fungsi heuristik yang tepat sangat penting untuk efisiensi algoritma A*. Heuristik yang umum digunakan dalam perencanaan rute adalah Manhattan Distance dan Euclidean Distance, tergantung pada grid atau jaringan jalan yang digunakan.

4. Implementasi Algoritma A*

   Setelah graf dan heuristik ditentukan, algoritma A* diimplementasikan untuk menemukan jalur terpendek dari lokasi awal ke tujuan. Algoritma ini mengevaluasi simpul terdekat berdasarkan biaya total (f(n)) dan memperbarui jalur hingga mencapai tujuan dengan biaya terendah.

5. Integrasi Data Lalu Lintas

   Integrasi data lalu lintas waktu nyata ke dalam perhitungan algoritma A* sangat penting untuk memastikan rute yang dihasilkan selalu optimal. Sistem harus dapat memperbarui graf secara dinamis berdasarkan perubahan kondisi lalu lintas.

C. Ambulans

Ambulans adalah kendaraan yang dirancang khusus untuk memberikan perawatan medis darurat dan transportasi pasien yang membutuhkan perhatian medis segera. Fungsi utama ambulans adalah untuk memberikan respons cepat terhadap situasi darurat medis, membawa peralatan medis penting, dan menyediakan transportasi yang aman ke fasilitas medis.

Terdapat beberapa jenis ambulans yang digunakan berdasarkan fungsinya, antara lain:

- Ambulans Darurat, dirancang untuk memberikan respons cepat terhadap panggilan darurat medis. Dilengkapi dengan peralatan medis darurat seperti defibrillator, oksigen, dan obat-obatan.
- Ambulans Transportasi**,** digunakan untuk memindahkan pasien yang tidak dalam kondisi kritis tetapi membutuhkan perawatan medis selama transportasi.
- Ambulans ICU (*Intensive Care Unit Ambulance*) Dilengkapi dengan peralatan medis canggih yang setara dengan ICU rumah sakit, digunakan untuk pasien dalam kondisi kritis yang memerlukan pengawasan intensif.

III. IMPLEMENTASI DAN PENGUJIAN

Untuk mengimplementasikan algoritma A* dalam optimasi rute ambulans, kita akan menggunakan bahasa pemrograman Python. Implementasi ini melibatkan beberapa langkah, termasuk pemodelan graf jalan, penerapan fungsi heuristik, dan integrasi data lalu lintas waktu nyata.

A. Struktur Data dan Pemodelan Graf

Graf jalan akan dimodelkan menggunakan struktur data yang cocok seperti dictionary, di mana setiap node (simpul) akan menyimpan informasi tentang node tetangganya dan bobot sisi (jalan) yang menghubungkannya. Berikut adalah contoh pemodelan graf sederhana,

```
class Graph:
    def __init__(self):
        self.nodes = set()
        self.edges = {}
        self.distances = {}

    def add_node(self, value):
        self.nodes.add(value)
        self.edges[value] = []

    def add_edge(self, from_node, to_node, distance):
        self.edges[from_node].append(to_node)
        self.edges[to_node].append(from_node)
        self.distances[(from_node, to_node)] = distance
    self.distances[(to_node, from_node)] = distance
```

*B. Implementasi Algoritma A**

Berikut adalah implementasi algoritma A* dalam Python:

```
import heapq
    def a_star_algorithm(graph, start, goal):
        open_set = []
        heapq.heappush(open_set, (0, start))
        came_from = {}
        g_score = {node: float('inf') for node in graph.nodes}
        g_score[start] = 0
        f_score = {node: float('inf') for node in graph.nodes}
        f_score[start] = heuristic(start, goal)

        while open_set:
            current = heapq.heappop(open_set)[1]

            if current == goal:
                path = reconstruct_path(came_from, current)
                total_cost = calculate_path_cost(graph, path)
                return path, total_cost

            for neighbor in graph.edges[current]:
```

```
            tentative_g_score = g_score[current] + graph.distances[(current, neighbor)]
            if tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = g_score[neighbor] + heuristic(neighbor, goal)
                if neighbor not in [i[1] for i in open_set]:
                    heapq.heappush(open_set, (f_score[neighbor], neighbor))

    return None, float('inf')

def heuristic(node, goal):
    # Menggunakan perbedaan nilai ASCII sebagai heuristik sederhana
    return abs(ord(node) - ord(goal))

def reconstruct_path(came_from, current):
    total_path = [current]
    while current in came_from:
        current = came_from[current]
        total_path.append(current)
    return total_path[::-1]

def calculate_path_cost(graph, path):
    total_cost = 0
    for i in range(len(path) - 1):
        total_cost += graph.distances[(path[i], path[i + 1])]
    return total_cost

def update_traffic_data(graph, traffic_data):
    for edge in graph.distances:
        from_node, to_node = edge
        if (from_node, to_node) in traffic_data:
            graph.distances[edge] = traffic_data[(from_node, to_node)]
```

Kode ini mengimplementasikan algoritma A* untuk menemukan jalur terpendek pada graf. Algoritma A* menggunakan struktur data heap untuk menyimpan node-node yang akan dievaluasi, dengan node awal dimulai dengan nilai f-score nol. Fungsi ini juga menggunakan dictionary untuk menyimpan node asal (came_from) dan nilai g-score serta f-score untuk setiap node. Saat mencari jalur, algoritma akan mengeluarkan node dengan f-score terendah dari heap dan mengevaluasi tetangganya. Jika g-score tentatif ke tetangga lebih rendah dari g-score yang diketahui, algoritma memperbarui nilai g-score, f-score, dan asal node, serta menambahkan tetangga ke heap jika belum ada di dalamnya. Fungsi heuristik sederhana menghitung perbedaan nilai ASCII antara node saat ini dan tujuan untuk memperkirakan jarak. Setelah mencapai tujuan, algoritma merekonstruksi jalur dari node asal ke tujuan menggunakan dictionary came_from dan menghitung total biaya jalur. Fungsi lain dalam kode memperbarui data lalu lintas pada graf berdasarkan kondisi terbaru untuk memastikan bahwa pencarian jalur memperhitungkan waktu tempuh yang dinamis.

C. Visualisasi Path

Berikut merupakan program untuk menampilkan visualisasi path dengan menggunakan library matplotlib dan network

```
import matplotlib.pyplot as plt
import networkx as nx

def visualize_graph(graph, path):
    G = nx.Graph()
    for node in graph.nodes:
        G.add_node(node)

    for (from_node, to_node), distance in graph.distances.items():
        G.add_edge(from_node, to_node, weight=distance)

    pos = nx.spring_layout(G)
    edge_labels = {(from_node, to_node): f"{distance}" for (from_node, to_node), distance in graph.distances.items()}

    plt.figure(figsize=(12, 8))
    nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=500, font_size=10, font_weight='bold')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

    path_edges = list(zip(path, path[1:]))
    nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='r', width=2)

    plt.title("Graph Visualization with A* Path Highlighted")
    plt.show()
```

Visualisasi dibuat dengan menampilkan node-node lalu edge-edge dengan angka yang menunjukkan jarak. Lalu edge yang bewarna merah merupakan jalur yang diambil ambulans untuk mencapai suatu tujuan rumah sakit yaitu J.

D. Main Program

Berikut merupakan main untuk menjalankan program dengan “A” adalah start node dan “J” adalah goal node

```
graph = Graph()
nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
for node in nodes:
    graph.add_node(node)

edges = [
    ('A', 'B', 5), ('A', 'C', 10), ('A', 'D', 3),
    ('B', 'E', 2), ('B', 'F', 8),
    ('C', 'G', 7), ('C', 'H', 4),
    ('D', 'I', 6), ('D', 'J', 9),
    ('E', 'H', 5), ('E', 'I', 3),
    ('F', 'G', 1), ('F', 'J', 2),
    ('G', 'I', 4), ('H', 'J', 3),
    ('I', 'J', 1), ('B', 'H', 4), ('G', 'H', 2)
]
for edge in edges:
    graph.add_edge(*edge)

# Mengupdate data lalu lintas
traffic_data = {
    ('A', 'B'): 6,  # Waktu tempuh yang diperbarui berdasarkan lalu lintas
    ('B', 'E'): 3,
    ('C', 'G'): 9,
    ('D', 'I'): 7,
    ('E', 'H'): 6,
    ('F', 'J'): 3,
    ('G', 'I'): 5,
    ('H', 'J'): 4
}
update_traffic_data(graph, traffic_data)

# Menjalankan algoritma A*
start = 'A'
goal = 'J'
path, total_cost = a_star_algorithm(graph, start, goal)
print("Path:", path)
print("Total Travel Time:", total_cost)

# Visualisasi graf dan jalur
visualize_graph(graph, path)
```

Main program diatas dibuat untuk membuat simulasi program yang menerapkan algoritma A* untuk mecapai rute paling optimala dalam menuju rumah sakit.

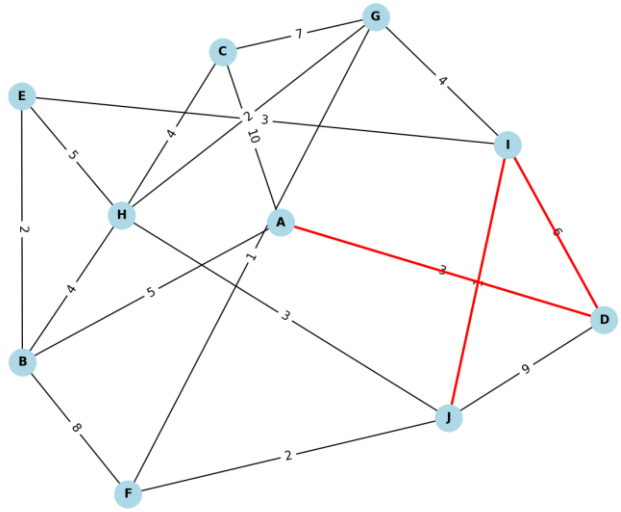
E. Hasil Pengujian

Berikut ini merupakan output program diatas

```
Path: ['A', 'D', 'I', 'J']
Total Travel Time: 11
```

Gambar 2. Output 1 program menggunakan A*

Output 1 merupakan hasil output pada terminal yang menujukkan node mana saja yang dipilih agar menemukan rute ambulans yang paling efisisen. Pada gambar diatas ditampilkan bahwa jalur ambulan yaitu A->D->I->J. Lalu Output 1 juga menampilkan total waktu perjalanan yang sudah merupakan waktu perjalanan tercepat untuk mencapai node goal yaitu J.



Gambar 3. Output 2 program menggunakan A*

Output 2 merupakan hasil visualisasi yang menggunakan library pada python. Output 2 ini ditujukan untuk menggambarkan secara jelas mana jalur yang diambil ambulans untuk mencapai rumah sakit. Berdasarkan visualisasi diatas, program kami sudah menunjukkan jalan yang tepat dengan jarak yang minimal dan waktu yang efisien.

IV. KESIMPULAN

Optimasi rute ambulans merupakan aspek kritis dalam meningkatkan efisiensi respons darurat dan mengurangi waktu tanggap. Algoritma A* menawarkan solusi yang efektif dengan menggabungkan biaya aktual $g(n)$ dan perkiraan heuristik $h(n)$ untuk menentukan jalur terpendek dan tercepat. Melalui implementasi algoritma A*, kita mampu:

a. Menentukan jalur optimal untuk ambulans dalam lingkungan perkotaan yang kompleks dengan mempertimbangkan berbagai faktor seperti kondisi lalu lintas waktu nyata dan kondisi jalan.

b. Menggunakan fungsi heuristik yang tepat untuk memperkirakan jarak dari node saat ini ke tujuan, yang membantu dalam mengarahkan ambulans melalui rute yang paling efisien.

c. Menghitung dan memperbarui biaya perjalanan berdasarkan data lalu lintas terkini, memastikan bahwa jalur yang dipilih selalu berdasarkan informasi yang paling mutakhir.

Penerapan algoritma A* dalam optimasi rute ambulans tidak hanya meningkatkan efisiensi operasional tetapi juga berpotensi menyelamatkan lebih banyak nyawa dengan mengurangi waktu tanggap. Dengan menggabungkan berbagai sumber data lalu lintas dan pemetaan digital, algoritma ini dapat secara efektif menyaring dan memilih rute terbaik untuk ambulans. Implementasi yang tepat dari algoritma ini dapat meningkatkan pengalaman dan kinerja layanan darurat, membuat proses pencarian jalur menjadi lebih terstruktur dan terinformasi. Di masa depan, pengembangan lebih lanjut pada fungsi heuristik dan integrasi data lalu lintas dapat semakin menyempurnakan hasil pencarian jalur dan mendukung keberhasilan sistem manajemen darurat dalam meningkatkan responsivitas dan efektivitasnya.

V. Ucapan Terima Kasih

Ucapan terima kasih penulis berikan kepada Tuhan Yang Maha Esa atas berka-Nya yang telah memungkinkan penulisan makalah berjudul “Mengoptimalkan Rute Ambulans untuk Efisiensi Menggunakan Algoritma A*” dapat diselesaikan dengan sempurna dan sesuai waktu yang diinginkan. Penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada para dosen pengampu Mata Kuliah Strategi Algoritma, khususnya Bapak Monterico Adrian, S.T., M.T. dan Bapak Dr. Rinaldi Munir, yang telah memberikan bimbingan, ilmu, dan inspirasi selama proses belajar. Selain itu, penulis tidak habisnya ingin menyampaikan rasa terima kasih kepada asisten mata kuliah, teman-teman seangkatan, serta keluarga penulis yang telah memberikan support dan dukungan yang begitu luar biasa selama proses berjalannya kuliah dan pembuatan makalh ini.

Penulis juga ingin mengucapkan terima kasih kepada semua sumber referensi dan literatur yang telah menjadi landasan utama dalam penulisan makalah ini. Setiap kontribusi dari sumber-sumber tersebut sangat berharga dalam memperkaya isi dan kualitas penelitian ini. Akhir kata, penulis berharap makalah ini dapat memberikan manfaat dan kontribusi positif bagi pengembangan ilmu pengetahuan, khususnya dalam bidang optimasi rute dan algoritma pencarian.

Referensi

[1] R. Munir, "Penentuan Rute (Route/Path Planning) (Bag.1)." Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/RoutePlanning-Bagian1-2021.pdf, pada 11 Juni 2024.

[2] R. Munir, "Penentuan Rute (Route/Path Planning) (Bag.2)." Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/RoutePlanning-Bagian2-2021.pdf, pada 11 Juni 2024.

[3] Python Software Foundation. (n.d.), "Python Documentation." Diakses dari https://docs.python.org/3/, pada 11 Juni 2024.

[4] Dechter, R., & Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. Journal of the ACM, 32(3), 505-536.

[5] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.

[6] Pohl, I. (1970). Heuristic search viewed as path finding in a graph. Artificial Intelligence, 1(3-4), 193-204.

[7] Stentz, A. (1994). Optimal and Efficient Path Planning for Partially-Known Environments. Proceedings of the IEEE International Conference on Robotics and Automation, 3310-3317.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Juni 2024

Muhammad Zaidan Sa’dun Robbani
13522146