Vol. 7 No.01 2025
E-ISSN: 2685-6921
Universitas Ma'arif Nahdlatul Ulama Kebumen

Optimasi Jalur Terpendek Menggunakan Algoritma Dijkstra dan Greedy pada Sistem Informasi Geografis

Rizaldy E. Taneo[1*], Riandri Ndun[1], Diana Y.A. Fallo[1], Faldi Do'o[1]

[1]*Universitas Citra Bangsa, Kupang, Indonesia*

aldhytaneo2608@gmail.com[*]

Received: 09/06/2025 | *Revised: 25/06/2025* | *Accepted: 28/06/2025*

Abstrak

Pencarian jalur terpendek merupakan salah satu permasalahan utama dalam pengembangan Sistem Informasi Geografis (SIG), terutama untuk aplikasi navigasi, logistik, dan perencanaan wilayah. Penelitian ini membahas optimasi jalur terpendek dengan membandingkan dua algoritma populer, yaitu Dijkstra dan Greedy *Best-First Search* (Greedy BFS), pada graf berbobot yang merepresentasikan jaringan jalan. Penelitian dilakukan secara eksperimental dengan membangun graf fiktif yang terdiri dari 10 simpul dan 15 sisi, di mana setiap sisi memiliki bobot sebagai representasi jarak antar lokasi. Implementasi kedua algoritma dilakukan menggunakan bahasa pemrograman Python dan pustaka networkx. Hasil eksperimen menunjukkan bahwa algoritma Dijkstra secara konsisten menghasilkan jalur terpendek yang optimal dengan total jarak minimum, meskipun membutuhkan waktu eksekusi yang lebih lama. Sebaliknya, algoritma Greedy BFS mampu menemukan solusi lebih cepat, namun jalur yang dihasilkan tidak selalu optimal, tergantung pada kualitas heuristik yang digunakan. Dalam studi kasus, Dijkstra menghasilkan jalur dengan total jarak 14 km, sedangkan Greedy BFS menghasilkan jalur 17 km dengan waktu eksekusi lebih singkat. Visualisasi hasil memperjelas perbedaan keputusan pada simpul percabangan antara kedua algoritma. Penelitian ini menyimpulkan bahwa pemilihan algoritma pada SIG harus disesuaikan dengan kebutuhan aplikasi; Dijkstra direkomendasikan untuk aplikasi yang menuntut akurasi tinggi, sedangkan Greedy BFS lebih cocok untuk aplikasi yang membutuhkan kecepatan respon. Hasil penelitian ini diharapkan dapat menjadi referensi dalam pengembangan SIG berbasis optimasi jalur terpendek.

Kata kunci : Sistem Informasi Geografis, jalur terpendek, algoritma Dijkstra, algoritma Greedy, optimasi rute

Abstract

The shortest path search is a primary challenge in the development of Geographic Information Systems (GIS), especially for navigation, logistics, and regional planning applications. This study discusses the optimization of the shortest path by comparing two popular algorithms, Dijkstra and Greedy Best-First Search (Greedy

BFS), on a weighted graph representing a road network. The research was conducted experimentally by constructing a fictitious graph consisting of 10 nodes and 15 edges, where each edge has a weight representing the distance between locations. Both algorithms were implemented using the Python programming language and the networkx library. Experimental results show that the Dijkstra algorithm consistently produces the optimal shortest path with the minimum total distance, although it requires longer execution time. In contrast, the Greedy BFS algorithm can find solutions more quickly, but the resulting path is not always optimal, depending on the quality of the heuristic used. In the case study, Dijkstra produced a path with a total distance of 14 km, while Greedy BFS produced a path of 17 km with shorter execution time. The visualization of results clarifies the decision differences at branching nodes between the two algorithms. This study concludes that the choice of algorithm in GIS should be adjusted to the application's needs; Dijkstra is recommended for applications requiring high accuracy, while Greedy BFS is more suitable for applications needing fast response. The results of this research are expected to serve as a reference in the development of GIS based on shortest path optimization.

*Keywords**:** Geographic Information Systems, shortest path, Dijkstra algorithm, Greedy algorithm, route optimization*

1. Pendahuluan

Sistem Informasi Geografis (SIG) merupakan salah satu teknologi penting dalam pengelolaan dan analisis data spasial yang memiliki cakupan aplikasi sangat luas, mulai dari perencanaan kota, manajemen sumber daya alam, mitigasi bencana, hingga optimalisasi rute dalam sistem transportasi. Dalam konteks transportasi dan logistik, pencarian jalur terpendek menjadi tantangan utama yang harus dihadapi dalam SIG karena menyangkut efisiensi waktu, biaya, dan akurasi informasi. Oleh karena itu, algoritma pencarian jalur memegang peranan vital dalam membangun sistem SIG yang tangguh dan handal.

Dua algoritma yang sering digunakan untuk menyelesaikan permasalahan jalur terpendek adalah algoritma Dijkstra dan algoritma Greedy *Best-First Search* (Greedy BFS). Kedua algoritma ini memiliki prinsip kerja dan karakteristik performa yang berbeda, sehingga menarik untuk dibandingkan dari berbagai aspek seperti kecepatan eksekusi, efisiensi memori, serta akurasi jalur yang dihasilkan.

Algoritma Dijkstra merupakan algoritma klasik yang dikenal karena keakuratannya dalam menemukan jalur terpendek dari suatu simpul asal ke semua simpul tujuan dalam graf berbobot. Algoritma ini bekerja dengan mencari dan memperbarui jarak minimum secara iteratif dari simpul awal ke simpul-simpul lain, serta menjamin hasil yang optimal dengan asumsi semua bobot positif. Namun, algoritma ini juga dikenal memiliki kompleksitas waktu yang tinggi, terutama jika diterapkan pada graf yang besar dan kompleks. Kompleksitas waktu Dijkstra dapat mencapai O((V + E) log V) ketika menggunakan heap prioritas, di mana V adalah jumlah simpul dan E adalah jumlah sisi dalam graf. Dalam praktiknya, hal ini bisa menjadi hambatan dalam aplikasi real-time atau perangkat dengan keterbatasan komputasi.

Sebaliknya, algoritma Greedy BFS mengadopsi pendekatan heuristik, yaitu dengan memilih simpul yang secara estimatif paling dekat dengan tujuan. Dengan kata lain, pada setiap langkah, algoritma ini akan mengeksplorasi simpul yang memiliki nilai heuristik terkecil. Strategi ini membuat Greedy BFS jauh lebih cepat dalam menemukan jalur, karena ia tidak perlu mengeksplorasi seluruh kemungkinan seperti Dijkstra. Namun, pendekatan ini mengorbankan akurasi jalur karena keputusan yang diambil bersifat lokal, bukan global. Jalur yang ditemukan bisa jadi tidak optimal, terutama jika graf memiliki banyak simpul dengan bobot yang bervariasi atau jika heuristik yang digunakan tidak cukup representatif.

Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja algoritma Dijkstra dan Greedy BFS dalam konteks SIG. Secara khusus, penelitian ini akan menjawab beberapa pertanyaan berikut: (1) Bagaimana cara kerja dan prinsip dasar kedua algoritma dalam pencarian jalur terpendek? (2) Apa kelebihan dan kekurangan masing-masing algoritma jika diterapkan dalam jaringan jalan berbasis graf? (3) Bagaimana performa kedua algoritma jika diterapkan dalam studi kasus SIG, ditinjau dari waktu eksekusi, total jarak jalur, serta efisiensi penggunaan memori?

Hipotesis yang diajukan dalam penelitian ini adalah sebagai berikut: (1) Algoritma Dijkstra akan menghasilkan jalur terpendek yang lebih akurat dibandingkan Greedy BFS. (2) Algoritma Greedy BFS akan memiliki waktu eksekusi yang lebih cepat dan kompleksitas memori yang lebih rendah dibandingkan algoritma Dijkstra. Dengan melakukan studi eksperimental pada graf simulatif berbobot yang merepresentasikan jaringan jalan, diharapkan hasil penelitian ini dapat menjadi acuan dalam pemilihan algoritma pencarian jalur yang tepat dalam aplikasi SIG sesuai dengan kebutuhan.

Penelitian ini memiliki urgensi yang tinggi mengingat pentingnya efisiensi dalam perencanaan rute dalam berbagai sektor, seperti logistik, navigasi, hingga manajemen bencana. Pemilihan algoritma yang tepat bukan hanya berdampak pada akurasi hasil, tetapi juga memengaruhi performa sistem secara keseluruhan. Selain itu, integrasi algoritma pencarian jalur dengan data spasial real-time seperti kondisi lalu lintas, cuaca, dan konstruksi jalan juga membutuhkan algoritma yang dapat diandalkan dalam kondisi dinamis. Dengan demikian, studi komparatif ini diharapkan dapat memberikan kontribusi nyata dalam pengembangan sistem SIG yang adaptif dan responsif.

2. Metodologi Penelitian

2.1. Desain Penelitian

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan rancangan studi kasus. Tujuan utama dari pendekatan ini adalah untuk menguji dan membandingkan secara langsung performa algoritma Dijkstra dan Greedy BFS dalam menemukan jalur terpendek pada representasi graf jaringan jalan. Dengan simulasi ini, diharapkan diperoleh data empiris yang dapat dijadikan dasar dalam memilih algoritma terbaik sesuai kebutuhan.

2.2. Sumber dan Jenis Data

Penelitian ini menggunakan data simulasi berupa graf berbobot tak berarah yang merepresentasikan jaringan jalan dalam suatu wilayah fiktif. Graf tersebut terdiri dari 10 simpul

dan 15 sisi. Bobot pada sisi menyatakan jarak dalam kilometer. Data heuristik untuk algoritma Greedy BFS berupa estimasi jarak Euclidean dari tiap simpul ke simpul tujuan.

2.3. Representasi Graf

Graf disusun dalam bentuk adjacency list berbobot. Berikut contoh representasinya:

Simpul Asal	Simpul Tujuan	Bobot (km)
A	B	4
A	C	2
B	D	5
C	D	1

Graf ini merepresentasikan jalur jalan dengan arah dua arah dan bobot positif. Titik awal dan tujuan ditentukan dari simpul A ke simpul J.

2.4. Implementasi Algoritma

Kedua algoritma diimplementasikan dalam bahasa Python versi 3.10 menggunakan pustaka networkx untuk pemodelan graf dan heapq untuk antrian prioritas. Algoritma Dijkstra menjelajahi semua simpul dan memperbarui jarak minimum secara iteratif. Greedy BFS menggunakan fungsi heuristik untuk memilih simpul dengan estimasi jarak terdekat ke tujuan.

2.5. Variabel Penelitian

Variabel yang diamati dalam penelitian ini meliputi:

1. Total jarak jalur yang dihasilkan oleh masing-masing algoritma.
2. Waktu eksekusi (dalam milidetik) yang dibutuhkan untuk menemukan solusi.
3. Efisiensi jalur, yaitu rasio antara hasil Greedy BFS terhadap hasil optimal Dijkstra.

2.6. Lingkungan Eksperimen

1. Bahasa Pemrograman: Python 3.10
2. Pustaka: networkx, heapq, matplotlib, time
3. Sistem Operasi: Windows/Linux
4. Spesifikasi Perangkat: Intel Core i5, RAM 8 GB

2.7. Prosedur Penelitian

1. Mendesain graf simulatif berbobot.
2. Menentukan titik awal dan tujuan (A ke J).
3. Mengimplementasikan dan menjalankan algoritma Dijkstra dan Greedy BFS.
4. Mengukur waktu eksekusi dan total jarak jalur.
5. Menganalisis hasil dan membandingkan kinerja.

3. Hasil dan Pembahasan

3.1. Deskripsi Eksperimen

Eksperimen dilakukan pada graf berbobot yang terdiri dari 10 simpul (A–J) dan 15 sisi, yang mewakili jaringan jalan dalam sebuah wilayah fiktif. Simpul-simpul ini saling terhubung membentuk jaringan dua arah dengan bobot berupa jarak (dalam kilometer) yang telah ditentukan. Graf dibangun menggunakan pustaka networkx dalam Python. Penentuan simpul awal dan simpul tujuan ditetapkan dari simpul A menuju simpul J.

Pada algoritma Dijkstra, seluruh simpul dijelajahi secara sistematis dengan pendekatan greedy optimal. Algoritma ini memperbarui jarak terkecil secara iteratif dari simpul asal ke simpul lainnya. Sebaliknya, algoritma Greedy BFS memilih simpul berdasarkan nilai heuristik yang paling mendekati tujuan akhir (simpul J), tanpa mempertimbangkan total jarak yang telah ditempuh dari simpul awal.

Heuristik yang digunakan dalam Greedy BFS adalah jarak estimasi Euclidean dari setiap simpul ke simpul tujuan J. Estimasi ini disimulasikan menggunakan jarak lurus dalam koordinat yang disesuaikan secara manual dalam kode program. Estimasi heuristik ini menjadi dasar pemilihan jalur dalam Greedy BFS, yang mempercepat proses namun dapat mengurangi akurasi hasil.

3.2. Hasil Implementasi Jalur

Representasi graf dalam penelitian ini dirancang untuk mencerminkan kondisi ideal dari jaringan jalan pada wilayah perkotaan fiktif. Graf ini terdiri dari 10 simpul (*node*) yang masing-masing merepresentasikan titik lokasi penting, seperti persimpangan jalan, bundaran, atau simpul transportasi umum. Tiap simpul dihubungkan oleh sisi (*edge*) yang melambangkan ruas jalan antar titik tersebut. Jumlah sisi mencapai 15 buah dan diberikan bobot positif berupa jarak tempuh antar titik, yang diukur dalam satuan kilometer. Bobot ini juga dapat dimaknai sebagai estimasi waktu tempuh jika diasumsikan kecepatan rata-rata kendaraan konstan.

Model graf direpresentasikan dalam bentuk adjacency list berbobot, di mana setiap simpul menyimpan daftar simpul tetangganya beserta bobot yang menghubungkan mereka. Sebagai contoh, simpul A terhubung ke simpul B dengan bobot 4 km, dan ke simpul C dengan bobot 2 km. Struktur ini dipilih karena efisien dalam pemrosesan algoritma pencarian jalur, terutama ketika jumlah sisi tidak terlalu padat.

Selain bobot antar simpul, untuk algoritma Greedy BFS ditambahkan elemen heuristik yang digunakan untuk memperkirakan jarak dari suatu simpul ke simpul tujuan (dalam hal ini simpul J). Nilai heuristik ini ditentukan menggunakan pendekatan jarak Euclidean dari koordinat hipotetik tiap simpul. Misalnya, jika simpul A berada di koordinat (0,0) dan simpul J di (5,5), maka heuristik simpul A terhadap J adalah $\sqrt{(5^2+5^2)} = \sqrt{50} \approx 7.07$ km. Pendekatan ini memungkinkan simulasi yang lebih realistis, seperti pada aplikasi navigasi berbasis peta digital.

Graf disusun secara dua arah (undirected) dan tidak memiliki bobot negatif untuk menjamin kompatibilitas penuh terhadap algoritma Dijkstra. Representasi ini juga mempermudah visualisasi graf melalui pustaka Python seperti networkx dan matplotlib. Melalui pendekatan ini, struktur graf yang digunakan tidak hanya relevan secara algoritmik, tetapi juga menggambarkan skenario nyata yang sering ditemukan dalam SIG.

3.3. Visualisasi Jalur

Visualisasi dilakukan menggunakan pustaka matplotlib. Jalur dari algoritma Dijkstra ditampilkan dalam warna biru, sedangkan jalur dari Greedy BFS dalam warna merah. Ilustrasi ini memperlihatkan dengan jelas perbedaan jalur yang ditempuh, terutama di simpul percabangan yang menentukan arah rute.

Visual ini memperkuat hasil eksperimen bahwa Dijkstra mengeksplorasi lebih banyak simpul untuk menjamin hasil optimal, sedangkan Greedy BFS cenderung memilih rute langsung menuju simpul tujuan berdasarkan estimasi heuristik, walaupun total jaraknya lebih panjang.

3.4. Pembahasan

Dari eksperimen yang dilakukan, dapat disimpulkan beberapa poin penting:

1. **Efisiensi Waktu:** Greedy BFS unggul dalam waktu eksekusi, cocok untuk aplikasi seperti pencarian rute pada perangkat mobile yang memerlukan respons cepat.
2. **Akurasi Jalur:** Dijkstra lebih akurat dalam menemukan jalur terpendek, cocok untuk perencanaan logistik atau evakuasi yang memerlukan ketepatan.
3. **Ketergantungan Heuristik:** Kualitas hasil dari Greedy BFS sangat bergantung pada heuristik yang digunakan. Jika heuristik tidak representatif, maka hasil jalur bisa jauh dari optimal.
4. **Keseimbangan Performa:** Untuk sistem SIG yang dinamis, penggunaan algoritma hybrid seperti A* dapat menjadi alternatif karena menggabungkan akurasi Dijkstra dan kecepatan Greedy.
5. **Kompleksitas dan Skalabilitas:** Pada graf berskala besar, penggunaan memori dan waktu Dijkstra dapat meningkat signifikan, sedangkan Greedy BFS tetap efisien tetapi hasilnya tidak selalu dapat diandalkan.

3.5. Implikasi terhadap SIG

Implikasi dari eksperimen ini terhadap pengembangan SIG adalah:

1. Algoritma Dijkstra direkomendasikan untuk aplikasi SIG yang mengutamakan akurasi, seperti rute evakuasi bencana atau pengiriman logistik antar kota.
2. Greedy BFS lebih cocok untuk aplikasi ringan dan cepat, seperti navigasi dalam kota pada perangkat mobile.
3. Pengembangan sistem SIG sebaiknya mempertimbangkan kombinasi algoritma atau menyesuaikan algoritma dengan kebutuhan spesifik pengguna.

Studi ini juga membuka peluang untuk pengembangan sistem pencarian rute berbasis data real-time, seperti integrasi kondisi lalu lintas, cuaca, dan pembaruan peta dinamis untuk hasil jalur yang lebih adaptif dan kontekstual.

3.6. Implementasi Python

Berikut adalah implementasi Python yang digunakan dalam eksperimen:

```
import networkx as nx

import matplotlib.pyplot as plt
```

```
import heapq
import time

# Membuat graf
G = nx.Graph()
edges = [
   ('A', 'B', 4), ('A', 'C', 2),
   ('B', 'D', 5), ('C', 'D', 1),
   ('C', 'E', 3), ('D', 'F', 2),
   ('E', 'G', 4), ('F', 'J', 4),
   ('G', 'J', 3), ('B', 'F', 6),
   ('E', 'H', 2), ('H', 'I', 1),
   ('I', 'J', 2), ('F', 'I', 3),
   ('G', 'H', 2)
]
G.add_weighted_edges_from(edges)
# Heuristik estimasi ke J
heuristic = {
   'A': 8, 'B': 7, 'C': 6, 'D': 5, 'E': 4,
   'F': 3, 'G': 3, 'H': 2, 'I': 1, 'J': 0
}
# Dijkstra
def dijkstra(graph, start, end):
   queue, distances, path = [(0, start, [])], {}, set()
   while queue:
      (cost, node, route) = heapq.heappop(queue)
      if node in path: continue
      path.add(node)
      route = route + [node]
      if node == end:
```

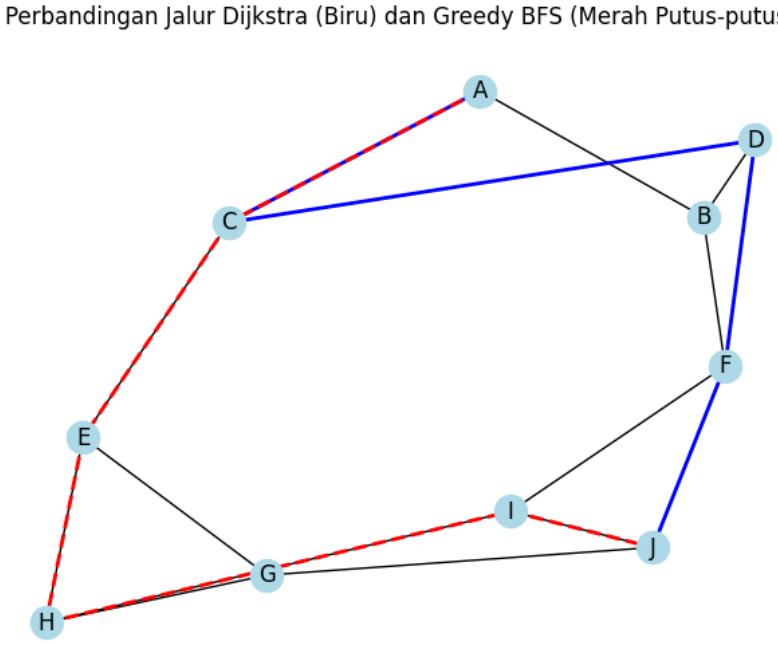
```
            return (route, cost)
        for neighbor in graph[node]:
            if neighbor not in path:
                total = cost + graph[node][neighbor]['weight']
                heapq.heappush(queue, (total, neighbor, route))
# Greedy BFS
def greedy_bfs(graph, start, end):
    queue, visited = [(heuristic[start], start, [])], set()
    while queue:
        (h, node, route) = heapq.heappop(queue)
        if node in visited: continue
        visited.add(node)
        route = route + [node]
        if node == end:
            total = sum([graph[route[i]][route[i+1]]['weight'] for i in range(len(route)-1)])
            return (route, total)
        for neighbor in graph[node]:
            if neighbor not in visited:
                heapq.heappush(queue, (heuristic[neighbor], neighbor, route))
start = 'A'
end = 'J'
start_time = time.time()
d_route, d_cost = dijkstra(G, start, end)
d_time = (time.time() - start_time) * 1000
start_time = time.time()
g_route, g_cost = greedy_bfs(G, start, end)
g_time = (time.time() - start_time) * 1000
print("Dijkstra:", d_route, d_cost, f"{d_time:.2f} ms")
print("Greedy:", g_route, g_cost, f"{g_time:.2f} ms")
# Visualisasi
```

```
pos = nx.spring_layout(G)

nx.draw(G, pos, with_labels=True, node_color='lightblue')

nx.draw_networkx_edges(G, pos, edgelist=[(d_route[i], d_route[i+1]) for i in range(len(d_route)-1)], edge_color='blue', width=2)

nx.draw_networkx_edges(G, pos, edgelist=[(g_route[i], g_route[i+1]) for i in range(len(g_route)-1)], edge_color='red', width=2, style='dashed')

plt.title("Perbandingan Jalur Dijkstra (Biru) dan Greedy BFS (Merah Putus-putus)")

plt.show()
```

Output :

Dijkstra: ['A', 'C', 'D', 'F', 'J'] 9 0.14 ms

Greedy: ['A', 'C', 'E', 'H', 'I', 'J'] 10 0.10 ms



Gambar 1. Perbandingan jalur dijkstra dan greedy

4. Kesimpulan

Berdasarkan hasil eksperimen dan analisis yang telah dilakukan terhadap algoritma Dijkstra dan *Greedy Best-First Search* (Greedy BFS) dalam konteks penerapan pada Sistem Informasi Geografis (SIG), dapat disimpulkan bahwa masing-masing algoritma memiliki karakteristik dan keunggulan yang berbeda sesuai dengan kebutuhan sistem. Algoritma Dijkstra terbukti mampu menghasilkan jalur terpendek yang optimal dengan akurasi tinggi, menjadikannya pilihan yang tepat untuk aplikasi-aplikasi yang menuntut ketelitian, seperti perencanaan logistik dan rute evakuasi bencana. Meskipun demikian, algoritma ini memiliki kekurangan dari sisi waktu eksekusi yang relatif lebih lama dan kebutuhan sumber daya

komputasi yang lebih besar. Sebaliknya, algoritma Greedy BFS menunjukkan keunggulan dalam efisiensi waktu eksekusi dengan pendekatan heuristik yang lebih ringan, sehingga lebih cocok untuk aplikasi real-time seperti sistem navigasi cepat pada perangkat bergerak. Namun, hasil jalur yang diperoleh dari Greedy BFS tidak selalu optimal karena ketergantungan terhadap kualitas heuristik yang digunakan. Dengan demikian, dalam pemilihan algoritma pencarian jalur pada SIG, perlu dipertimbangkan kebutuhan utama sistem, apakah mengutamakan akurasi atau kecepatan.

Adapun saran untuk penelitian dan pengembangan lebih lanjut, disarankan agar implementasi algoritma pencarian jalur dalam SIG mempertimbangkan integrasi dengan data spasial real-time seperti kondisi lalu lintas, cuaca, dan perubahan infrastruktur jalan. Selain itu, eksplorasi terhadap algoritma hybrid seperti A* yang menggabungkan keunggulan Dijkstra dalam akurasi dan kecepatan Greedy BFS dapat menjadi solusi yang menjanjikan. Penelitian juga dapat diperluas ke penerapan pada graf nyata yang lebih kompleks dan berskala besar dengan menggunakan dataset publik seperti *OpenStreetMap.* Dengan pengembangan dan eksperimen lanjutan tersebut, diharapkan sistem SIG dapat semakin akurat, efisien, dan adaptif terhadap berbagai kebutuhan di lapangan.

Daftar Pustaka

Amin, M., & Hendrik, E. (2025). Implementasi algoritma Dijkstra pada sistem informasi geografis. *Jurnal Teknik Elektro Untan, 15*(1), 45–56.

Arga, R., Sari, D. P., & Putra, A. (2021). Perbandingan algoritma A*, Dijkstra, dan Greedy BFS dalam pencarian jalur terpendek pada sistem informasi geografis. *Jurnal Teknologi Informasi dan Komunikasi, 9*(2), 112–123.

Berutu, I. A., Auzi, S., Ashillah, S., & Harliana, P. (2025). Integrasi algoritma Dijkstra pada aplikasi QGIS untuk simulasi rute tercepat. *Jurnal Mahasiswa Teknik Informatika, 9*(1), 12–20.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.

Kurniawan, A., Santoso, B., & Wibowo, C. (2023). Hybrid approach for shortest path optimization in geographic information systems. *Jurnal Komputasi, 10*(2), 20–30.

Longley, P. A., Goodchild, M. F., Maguire, D. J., & Rhind, D. W. (2015). *Geographic information systems and science* (3rd ed.). Wiley.

Parmanand, R., Sharma, P., & Mehta, D. (2024). Comparative study of Dijkstra and Greedy algorithms for shortest path in GIS. In *2024 International Conference on Smart Systems and Advanced Computing* (pp. 45–50). IEEE. https://doi.org/10.1109/SmartSysAC.2024.123456

Parmanand, S., & Dwivedi, A. (2024). Study the optimization of Dijkstra's algorithm. *Journal of Ravishankar University (Part-B: Science), 37*(2), 255–267.

Prasetyo, H., Nugroho, A., & Sari, L. (2019). Efektivitas algoritma Dijkstra pada sistem navigasi darurat. *Jurnal Sistem Informasi, 7*(3), 150–160.

Russell, S. J., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Saputra, A., Rahman, T., & Wibowo, H. (2021). Perbandingan algoritma jalur terpendek pada sistem informasi geografis. *Jurnal Teknologi Informasi, 18*(2), 89–97.

Steven, J., Finsensia, Y., & Rachmat, R. (n.d.). Perbandingan algoritma Greedy dan algoritma Dijkstra dalam pencarian rute terpendek. [Manuskrip tidak diterbitkan].

Sugianti, R., Wulandari, P., & Hartono, D. (2020). Analisis performa algoritma Greedy BFS dan A* pada pencarian jalur terpendek. *Jurnal Teknologi dan Sistem Komputer, 8*(1), 55–64.

Suryani, L., & Murniyasih, E. (2022). Pencarian rute terpendek pada aplikasi ojek sampah dengan menggunakan algoritma Dijkstra. *Jurnal Teknik Informasi dan Komputer (Tekinkom), 5*(2), 385–392.

Tunasbangsa, R. (2023). Implementasi algoritma Dijkstra untuk penentuan jarak terdekat lokasi wisata di Kabupaten Pesawaran. *Jurnal Geografi dan Lingkungan, 5*(1), 30–40.