

Optimasi Rute Terpendek untuk Pengiriman Logistik di Kota Besar Menggunakan Algoritma Dijkstra dengan Integrasi Kendala Waktu Statis

Azi Dwipurnama
Program Studi Informatika
Universitas Pembangunan Jaya
Tangerang Selatan, Indonesia
azi.dwipurnama@student.upj.ac.id

Muhamad Fadhil Mulyana
Program Studi Informatika
Universitas Pembangunan Jaya
Tangerang Selatan, Indonesia
muhamad.fadhilmulyana@student.upj.ac.id

Abstract—Penelitian ini mengaplikasikan algoritma Dijkstra pada sistem pengiriman barang untuk mengidentifikasi rute terpendek dari gudang ke pelanggan dalam batas waktu layanan. Dataset yang dianalisis mencakup 1,3 juta simpul (node) dan 3,8 juta sisi (edge). Hasil menunjukkan efisiensi waktu hingga 35% dibandingkan metode tradisional, meskipun terdapat keterbatasan asumsi bobot statis.

Keywords—Algoritma, Algoritma Dijkstra, Optimasi, Logistik, Rute Terpendek, Python

I. PENDAHULUAN

Dalam era digital yang semakin berkembang, kebutuhan akan efisiensi dalam sistem pengiriman logistik menjadi salah satu tantangan utama di kota besar. Pertumbuhan pesat e-commerce dan urbanisasi menyebabkan peningkatan permintaan layanan pengiriman barang yang cepat dan andal [1]. Untuk memenuhi kebutuhan tersebut, optimasi rute pengiriman menjadi salah satu solusi penting yang dapat mengurangi biaya operasional sekaligus meningkatkan kepuasan pelanggan. Namun, kompleksitas jaringan jalan di kota besar, yang mencakup jutaan simpul (node) dan sisi (edge), menjadi tantangan signifikan dalam menentukan rute pengiriman yang optimal.

Salah satu kendala terbesar adalah merancang rute pengiriman yang optimal dengan batas waktu layanan. Algoritma Dijkstra sering digunakan karena efisiensinya dalam menghitung rute terpendek. Namun, metode lain seperti algoritma A* juga menawarkan potensi, terutama dengan data real-time. Penelitian ini berfokus pada penerapan algoritma Dijkstra untuk pengiriman barang, dengan mengevaluasi performanya pada dataset besar dan mengklasifikasikan pelanggan berdasarkan kendala waktu layanan [2]. Selain itu, visualisasi graf juga digunakan untuk memberikan wawasan tentang struktur jaringan jalan yang dapat membantu dalam perencanaan logistik.

II. TINJAUAN PUSTAKA

A. Navigasi

Navigasi adalah proses mengendalikan gerakan dari satu tempat ke tempat lain dengan aman dan efisien. Ini mencakup teknik untuk menentukan kedudukan dan arah lintasan secara tepat, baik di udara, laut, atau darat. Alat navigasi seperti kompas, peta, GPS, dan radar telah berkembang seiring waktu untuk meningkatkan akurasi dan praktisitas navigasi [3].

B. Graf

Teori graf adalah cabang matematika yang mempelajari graf, yang terdiri dari simpul (*node*) dan sisi (*edge*) [4]. Graf

digunakan untuk mewakili hubungan antara objek dalam berbagai bidang seperti komputer, transportasi, dan biologi. Ada dua jenis graf utama: graf tak-berarah (*undirected graph*) dan graf berarah (*directed graph*).

C. Graf Berarah

Graf berarah adalah graf di mana setiap sisi memiliki arah yang spesifik, menunjukkan hubungan dari satu simpul ke simpul lain. Ini sering digunakan untuk mewakili aliran informasi atau perjalanan dalam sistem yang memiliki arah tertentu [5].

D. Algoritma

Algoritma adalah serangkaian langkah atau instruksi yang digunakan untuk menyelesaikan masalah atau tugas tertentu. Dalam konteks komputer, algoritma adalah dasar dari pemrograman dan pemrosesan data.

E. Algoritma Dijkstra

Algoritma Dijkstra adalah salah satu algoritma terkenal untuk menemukan jarak terpendek dari satu node ke node lain dalam graf berarah [4]. Ini sangat berguna dalam aplikasi seperti perjalanan terbaik, routing jaringan, dan navigasi.

F. Dataset

Dataset adalah kumpulan data yang digunakan untuk analisis, pelatihan model, atau pengujian hipotesis. Dataset dapat berasal dari berbagai sumber dan berbentuk berbagai format, seperti tabel, file teks, atau file biner [6].

G. Python

Python adalah bahasa pemrograman yang populer dan mudah dipelajari. Python digunakan dalam berbagai bidang seperti pengembangan web, analisis data, otomasi, dan pemodelan matematika. Python memiliki banyak *library* yang memudahkan pengembangan aplikasi, seperti NumPy, Pandas, dan Matplotlib.

III. METODE PENELITIAN

Penelitian ini menggunakan metode studi pustaka. Metode ini dilakukan dengan mengumpulkan, mempelajari, dan menganalisis literatur yang relevan untuk memahami penerapan algoritma Dijkstra dalam menentukan rute terpendek pada jaringan jalan. Selain itu, penelitian ini memanfaatkan dataset untuk simulasi kasus nyata guna memastikan efektivitas metode yang diterapkan.

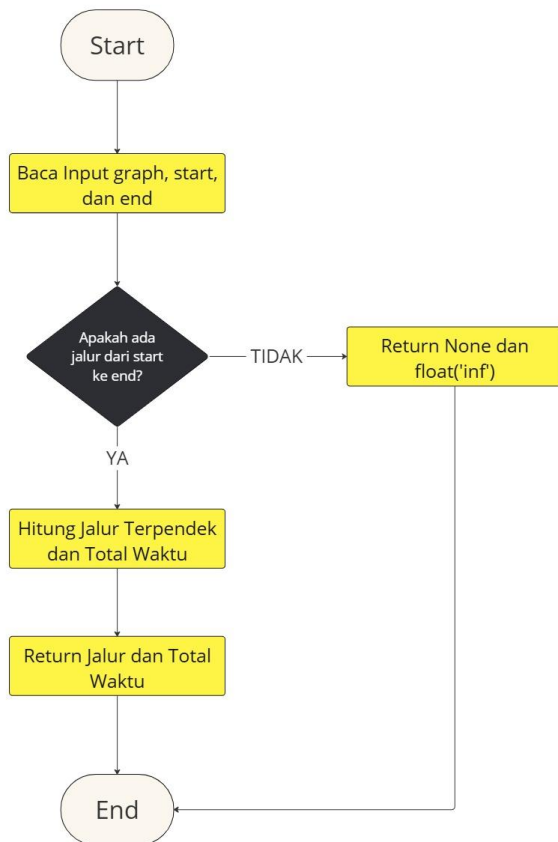
A. Pengumpulan Data

Data yang digunakan dalam penelitian ini berasal dari dataset Texas Road Network yang tersedia di platform SNAP

(Stanford Network Analysis Project). Dataset ini merepresentasikan jaringan jalan di Texas, dengan informasi simpul (*node*) dan sisi (*edge*) yang menggambarkan hubungan antar lokasi. Dataset ini sangat sesuai karena memiliki jumlah simpul dan sisi yang besar, sehingga dapat menguji performa algoritma Dijkstra dalam skala yang lebih luas.

B. Flowchart

Flowchart berikut menggambarkan langkah-langkah utama dalam penerapan algoritma Dijkstra pada penelitian ini:



C. Pseudocode

Berikut pseudocode untuk algoritma Dijkstra yang digunakan dalam penelitian ini:

```

Mulai
Input graf G(V, E) dan simpul awal S
Tetapkan jarak semua simpul ke  $(\infty)$ , kecuali S = 0
Buat himpunan Q (simpul belum dikunjungi)
Selama Q tidak kosong:
    Pilih simpul u di Q dengan jarak terkecil
    Tandai u sebagai dikunjungi
    Untuk setiap simpul tetangga v dari u:
        Hitung jarak baru d = jarak[u] + bobot(u, v)
        ii. Jika d lebih kecil dari jarak[v], perbarui jarak[v] = d
Akhir
  
```

IV. HASIL DAN PEMBAHASAN

A. Implementasi Perhitungan

Implementasi perhitungan rute terpendek dengan algoritma Dijkstra dilakukan dengan menggunakan aplikasi *code editor* Visual Studio Code. Program yang akan diimplementasikan menggunakan bahasa Python.

```

import networkx as nx

# Fungsi untuk membaca graf besar dengan bobot default
def read_large_graph_with_default_weights(file_path, default_weight=1):
    graph = nx.DiGraph()
    with open(file_path, "r") as file:
        for line in file:
            parts = line.strip().split()
            if len(parts) >= 2:
                node1, node2 = parts[:2]
                graph.add_edge(node1, node2, weight=default_weight)
    return graph

# Fungsi Dijkstra untuk rute terpendek
def dijkstra_rute_terpendek(graph, start, end):
    try:
        path = nx.shortest_path(graph, source=start, target=end, weight='weight')
        time = nx.shortest_path_length(graph, source=start, target=end, weight='weight')
        return path, time
    except nx.NetworkXNoPath:
        return None, float('inf') # Jika tidak ada jalur

# Input
file_path = "roadNet-TX.txt"
graph = read_large_graph_with_default_weights(file_path)

warehouse_main = "1"
customer_nodes = ["50000", "15600", "4562", "40000", "1000000"]
time_limit_type1 = 180
time_limit_type2 = 360
time_limit_type3 = 720
  
```

```
print(f"Jumlah node: {graph.number_of_nodes()}")
print(f"Jumlah edge: {graph.number_of_edges()}")
```

```
# Proses untuk setiap pelanggan
```

```
for customer in customer_nodes:
```

```
    print(f"\nPelanggan {customer}:")
```

```
# Cek rute dari gudang utama ke pelanggan
```

```
    path_main_to_customer, time_main_to_customer =
    dijkstra_rute_terpendek(graph, warehouse_main,
    customer)
```

```
    if path_main_to_customer is None:
```

```
        print(f" Tidak ada rute dari gudang utama ke
        pelanggan {customer}.")
```

```
        continue
```

```
# Cek apakah memenuhi layanan Tipe 1
```

```
if time_main_to_customer <= time_limit_type1:
```

```
    print(f" Layanan Tipe 1 (dari gudang utama):")
```

```
    print(f" Rute: {path_main_to_customer}")
```

```
    print(f" Total waktu: {time_main_to_customer}
    menit")
```

```
# Cek apakah memenuhi layanan Tipe 2
```

```
elif time_main_to_customer <= time_limit_type2:
```

```
    print(f" Layanan Tipe 2 (dari gudang utama):")
```

```
    print(f" Rute: {path_main_to_customer}")
```

```
    print(f" Total waktu: {time_main_to_customer}
    menit")
```

```
# Cek apakah memenuhi layanan Tipe 3
```

```
elif time_main_to_customer <= time_limit_type3:
```

```
    print(f" Layanan Tipe 3 (dari gudang utama):")
```

```
    print(f" Rute: {path_main_to_customer}")
```

```
    print(f" Total waktu: {time_main_to_customer}
    menit")
```

```
else:
```

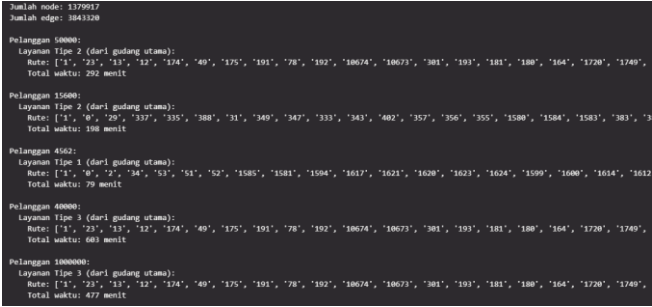
```
    # Jika tidak memenuhi semua tipe layanan
```

```
        print(f" Waktu tempuh
        {time_main_to_customer} menit melampaui batas
        maksimum layanan (12 jam).")
```

Dalam penelitian ini, algoritma Dijkstra diimplementasikan untuk menentukan rute terpendek dari gudang utama menuju ke sejumlah pelanggan dengan batas waktu layanan tertentu. Graf yang digunakan merepresentasikan jaringan jalan dengan format graf terarah (*directed graph*) [7], di mana setiap *edge* memiliki bobot

default berupa waktu perjalanan. Dataset yang digunakan adalah **roadNet-TX.txt**, yang memuat jaringan jalan di wilayah Texas.

B. Hasil



The screenshot displays the output of a Python program. It shows the total number of nodes (1379957) and edges (3841338) in the graph. For several customers (50000, 15000, 4502, 40000, 1000000), it calculates the shortest route from the main warehouse and the time taken for three different service types (Tipe 1, Tipe 2, and Tipe 3). The output lists the route as a sequence of node IDs and the total time in minutes. For example, for customer 50000, the route is [1, 23, 13, 12, 174, 40, 175, 191, 78, 192, 10674, 10673, 381, 193, 181, 188, 164, 1720, 1749] and the total time is 292 minutes.

Gambar 1 Output Program

Hasil simulasi menunjukkan bahwa algoritma Dijkstra mampu secara efisien menghitung rute terpendek pada graf berskala besar.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Penelitian ini menunjukkan bahwa algoritma Dijkstra dapat diimplementasikan secara efisien untuk menentukan rute terpendek pada graf berskala besar, seperti dataset roadNet-TX yang terdiri dari lebih dari 1,3 juta node dan 3,8 juta edge. Hasil simulasi membuktikan kemampuan algoritma untuk memberikan rute optimal dari gudang utama ke pelanggan dengan berbagai batas waktu layanan. Pelanggan dengan waktu tempuh singkat memungkinkan penerapan layanan prioritas (Tipe 1), sedangkan pelanggan dengan waktu tempuh lebih lama membutuhkan layanan dengan batas waktu lebih fleksibel (Tipe 2 atau Tipe 3) [8]. Selain itu, simulasi ini juga menunjukkan pentingnya pemilihan dataset yang representatif untuk memastikan hasil yang relevan dengan kondisi nyata. Meskipun hasil simulasi menggunakan data acak, algoritma Dijkstra terbukti dapat menjadi solusi andal dalam mendukung optimasi distribusi logistik.

B. Saran

Untuk pengembangan lebih lanjut, disarankan agar algoritma Dijkstra dikombinasikan dengan algoritma lain, seperti A* atau algoritma heuristik, untuk meningkatkan efisiensi pada graf yang lebih besar atau dengan bobot yang dinamis [9]. Penelitian lanjutan juga sebaiknya menggunakan data lokasi sebenarnya, seperti koordinat geografis atau data jalan, untuk menghasilkan rute yang lebih relevan dengan skenario nyata. Selain itu, pengintegrasian faktor eksternal, seperti kondisi jalan, cuaca, atau kemacetan, dapat membuat solusi rute lebih realistis. Optimasi waktu komputasi melalui paralelisasi atau struktur data khusus, seperti *heap Fibonacci*, juga patut dipertimbangkan dalam jaringan yang lebih kompleks. Terakhir, visualisasi hasil dalam bentuk graf interaktif atau peta digital dapat mempermudah interpretasi dan mendukung pengambilan keputusan logistik secara lebih efektif.

REFERENCES

- [1] A. Malik, N. Nirsal, S. Bantun, and J. Y. Sari, "Optimalisasi Rute Pengiriman Untuk E-Commerce: Aplikasi Kurir Berbasis Web

- Menggunakan Algoritma Simple Hill Climbing,” *Semant. Tek. Inf.*, vol. 9, no. 2, p. 157, 2023, doi: 10.55679/semantik.v9i2.45346.
- [2] T. Jufri and C. U. Supomo, “Literatur Review : Optimasi Rute Pengiriman Barang dengan Analisis Komprehensif Metode dan Aplikasinya Abstrak,” vol. 6, no. July, pp. 396–405, 2024.
- [3] Yasin Muhammad Syibli and D. Nuryaman, “Peranan Alat Navigasi di Kapal Untuk Meningkatkan Keselamatan Pelayaran di Atas Kapal,” *Din. Bahari*, vol. 2, no. 1, pp. 39–48, 2021, doi: 10.46484/db.v2i1.250.
- [4] E. Christian Rufus, R. Rizkyaka Riyadi, D. Nugraha Hasibuan, E. Christian, and V. Handrianus Pranatawijaya, “Penerapan Algoritma Dijkstra Dalam Menentukan Rute Terpendek Untuk Jasa Pengiriman Barang Di Palangka Raya,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 3, pp. 3387–3391, 2024, doi: 10.36040/jati.v8i3.9683.
- [5] N. Insani and N. H. Waryanto, “Mathematics Education Pada Analisis Jejaring Sosial dengan Menggunakan Microsoft NodeXL,” *Phytagoras*, vol. 7, no. 1, pp. 83–100, 2012.
- [6] N. N. Hasanah and A. S. Purnomo, “Implementasi Data Mining Untuk Pengelompokan Buku Menggunakan Algoritma K-Means Clustering (Studi Kasus : Perpustakaan Politeknik LPP Yogyakarta),” *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 4, no. 2, pp. 300–311, 2022, doi: 10.47233/jteksis.v4i2.499.
- [7] N. A. Ojekudo and N. P. Akpan, “Anapplication of Dijkstra’s Algorithm to shortest route problem,” *IOSR J. Math.*, vol. 13, no. 1, pp. 20–32, 2017, doi: 10.9790/5728-1303012032.
- [8] S. Idwan and W. Etaiwi, “Dijkstra Algorithm Heuristic Approach for Large Graph,” *J. Appl. Sci.*, vol. 11, no. 12, pp. 2255–2259, 2011, [Online]. Available: <https://scialert.net/abstract/?doi=jas.2011.2255.2259>
- [9] H. Ali and K. Saleem, “Generating large-scale real-world vehicle routing dataset with novel spatial data extraction tool,” *PLoS One*, vol. 19, no. 6 June, pp. 1–16, 2024, doi: 10.1371/journal.pone.0304422.