Jurnal Indonesia : Manajemen Informatika dan Komunikasi
www.journal.amikindonesia.ac.id/jimik/ Vol 5 No 1, Januari (2024)
E-ISSN: 2723-7079, P-ISSN: 2776-8074

Pencarian Rute Terpendek menggunakan Algoritma Genetika (Studi Kasus : Pengoptimalan Mobilitas Kota Salatiga Terhadap Kota-Kota Tetangga)

Agustho Isai [1*], Adi Nugroho [2]

[1*,2] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

Email: 672017226@student.uksw.edu [1*], adi.nugroho@uksw.edu [2]

Histori Artikel:
Dikirim 13 November 2023; *Diterima dalam bentuk revisi* 26 November 2023; *Diterima* 10 Desember 2023; *Diterbitkan* 10 Januari 2024.

Abstrak

Meningkatnya populasi penduduk pada sebuah daerah memberikan dampak yang signifikan pada kepadatan kendaraan di daerah tersebut. Masalah yang sering muncul adalah kemacetan yang terjadi di beberapa rute sekaligus. Faktor ini menjadi sebuah permasalahan umum, yang cukup mengganggu masyarakat dan pengguna jalan. Untuk mengatasi masalah tersebut digunakan metode algoritma genetika sebagai metode penyelesaian. Studi kasus ini ditetapkan pada lingkup daerah atau kota yang berada di sekitar Kota Salatiga, diantaranya Kota Ambarawa, Kota Semarang, Kota Boyolali, Kota Solo dan Kota Magelang. Dengan menggunakan metode algoritma genetika, rute tercepat dengan jarak terpendek dapat ditemukan. Algoritma Genetika yang digunakan dalam penelitian ini memungkinkan hasil pencarian rute terpendek dengan optimal. Selain itu, penelitian ini juga dapat meminimalisir jarak dan waktu tempuh dari beberapa rute alternatif yang didapat, sehingga memberikan manfaat yang signifikan dalam pengembangan sistem transportasi di wilayah tersebut.

Kata Kunci: Algoritma genetika; Pencarian Rute Terpendek; Sistem Transportasi Wilayah.

Abstract

The increase in population in an area has a significant impact on vehicle density in that area. The problem that often arises is traffic jams that occur on several routes at once. This factor is a common problem, which is quite disturbing to the public and road users. To overcome this problem, the genetic algorithm method is used as a solution method. This case study was set at the regional or city level around Salatiga City, including Ambarawa City, Semarang City, Boyolali City, Solo City and Magelang City. By using the genetic algorithm method, the fastest route with the shortest distance can be found. The Genetic Algorithm used in this research allows optimal shortest route search results. Apart from that, this research can also minimize the distance and travel time of several alternative routes obtained, thus providing significant benefits in developing the transportation system in the region.

Keyword: Genetic Algorithm; Shortest Route Search; Regional Transportation System.

1. Pendahuluan

Perkembangan Teknologi Informasi telah memberikan dampak positif yang signifikan pada kehidupan manusia, membawa inovasi dan kemudahan dalam berbagai aspek, termasuk penyelesaian permasalahan sehari-hari melalui metode yang sederhana dan proses yang lebih efektif serta efisien. Meskipun demikian, pertumbuhan pesat penduduk seiring dengan kemajuan zaman dan teknologi informasi menyebabkan peningkatan populasi, yang pada gilirannya menciptakan kepadatan kendaraan sebagai salah satu tantangan yang dihadapi di beberapa wilayah. Oleh karena itu, dibutuhkan sistem yang mampu memberikan bantuan dalam menentukan rute terpendek antara berbagai daerah. Penelitian ini mengeksplorasi pencarian rute terpendek menggunakan algoritma genetika, dengan fokus pada pengoptimalan mobilitas dan konektivitas di Kota Salatiga terhadap kota-kota tetangga. Manfaat dari penelitian ini dapat dirasakan dalam perkembangan teknologi, di mana efisiensi transportasi dapat ditingkatkan, biaya perjalanan dapat diminimalkan, dan konektivitas antar daerah dapat diperkuat. Penerapan algoritma genetika dalam konteks ini juga memberikan kontribusi terhadap perkembangan teknologi terkini dalam bidang transportasi dan optimasi. Selain itu, penelitian ini mendorong penggunaan teknologi informasi dalam konteks transportasi dan konektivitas.

Sejumlah penelitian sebelumnya juga telah membahas penerapan algoritma genetika dalam pencarian rute terpendek. Sebagai contoh, penelitian oleh Irianti, Cokrowibowo, dan Aswandi (2021) mengenai optimasi Multiple Traveling Salesman Problem dengan Algoritma Genetika pada kasus model rute terpendek penjemputan sampah di Kabupaten Majene [1]. Melladia (2020) juga menyelidiki penggunaan Algoritma Genetika untuk menentukan jalur jalan dengan lintasan terpendek [2]. Mubarok dan Chotijah (2021) melakukan penerapan Algoritma Genetika untuk mencari optimasi kombinasi jalur terpendek dalam kasus Travelling Salesman Problem [3]. Penelitian Muhandhis et al. (2023) membahas pencarian rute terpendek tim promosi kampus dengan menggunakan Algoritma Genetik [4]. Oktaviandi et al. (2019) melakukan perbandingan Algoritma Genetika dengan Algoritma Greedy untuk pencarian rute terpendek [5]. Selain itu, beberapa penelitian lain juga relevan dengan topik ini, seperti penelitian Ramadhania dan Rani (2021) yang mengimplementasikan kombinasi algoritma genetika dan tabu search untuk penyelesaian Travelling Salesman Problem [6], penelitian Saputra dan Ahmad (2020) yang menggunakan Algoritma Genetika untuk menentukan jalur terpendek wisata Kota Bukittinggi [7], serta penelitian Tohari dan Astuti (2023) yang menerapkan Algoritma Genetika dalam menentukan rute terpendek PT. Pos Cabang Lamongan [8].

Dengan demikian, penelitian mengenai pencarian rute terpendek menggunakan algoritma genetika memiliki potensi untuk memberikan solusi efektif dalam mengatasi masalah kepadatan lalu lintas di wilayah tertentu. Algoritma genetika dapat memberikan perhitungan yang tepat dengan waktu yang relatif singkat dan biaya yang sedikit. Selain itu, penggunaan teknologi informasi yang canggih dalam penelitian ini akan membantu mengoptimalkan rute perjalanan dan mengurangi waktu tempuh bagi para pengguna jalan.

2. Metode Penelitian

Algoritma genetika adalah metode yang digunakan untuk menyelesaikan masalah optimasi, termasuk mencari rute terpendek. Algoritma genetika meniru cara kerja proses genetika dalam evolusi organisme hidup untuk mencari solusi yang optimal. Proses algoritma genetika melibatkan pengkodean solusi dalam bentuk kromosom dan menggunakan operasi genetika seperti seleksi, penjelajahan, dan mutasi untuk menghasilkan populasi baru yang semakin optimal. Algoritma genetika merupakan teknik pencarian yang diadopsi dari proses evolusi alam. Proses komputasi yang terjadi dalam algoritma ini analog dengan proses seleksi makhluk hidup dalam sebuah populasi. Oleh karena itu, proses pencarian dalam algoritma genetika dilakukan sekaligus atas sejumlah penyelesaian masalah yang mungkin [9]. Menurut Haupt dan Haupt dalam (Zukhri 2014) struktur dasar algoritma genetika terdiri atas beberapa langkah [9]:

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

1) Inisialisasi Populasi. Populasi awal dengan ukuran N diinisialisasi secara acak. Populasi awal direpresentasikan sebagai kumpulan kromosom, [K1, K2, ..., KN].
2) Evaluasi Populasi. Setiap individu dalam populasi dievaluasi menggunakan fungsi fitness, f(Ki). Hasil evaluasi fitness digunakan untuk menentukan seberapa baik kromosom tersebut dalam mencapai solusi yang diinginkan.
3) Seleksi. Populasi Sebagian kromosom terbaik dipilih dari populasi saat ini untuk melahirkan generasi berikutnya. Seleksi dapat dilakukan berdasarkan nilai fitness individu atau probabilitas seleksi.
4) Proses Penyilangan. Pada langkah ini, pasangan kromosom tertentu dipilih untuk melakukan proses penyilangan (crossover). Proses ini melibatkan pertukaran materi genetik antara pasangan kromosom untuk menghasilkan kromosom anak.
5) Proses Mutasi. Pada langkah ini, kromosom tertentu dipilih untuk mengalami proses mutasi. Proses mutasi menghasilkan perubahan acak pada gen dalam kromosom.
6) Evaluasi Populasi Baru. Setelah proses penyilangan dan mutasi selesai, populasi baru terbentuk. Langkah ini melibatkan evaluasi fitness terhadap populasi baru yang terbentuk.
7) Ulangi dari Langkah 3. Proses langkah 3 hingga langkah 6 diulang sampai syarat berhenti terpenuhi. Syarat berhenti dapat berupa mencapai solusi optimal, mencapai batas iterasi tertentu, atau kondisi berhenti lainnya.

Dalam pencarian rute terpendek, algoritma genetika dapat digunakan untuk mencari rute terpendek antara beberapa titik atau lokasi. Misalnya, jika ingin mencari rute terpendek antara kota-kota di sekitar Salatiga diantaranya Kota Ambarawa, Kota Semarang, Kota Boyolali, Kota Solo dan Kota Yogyakarta, dapat menerapkan algoritma genetika dengan langkah-langkah berikut :

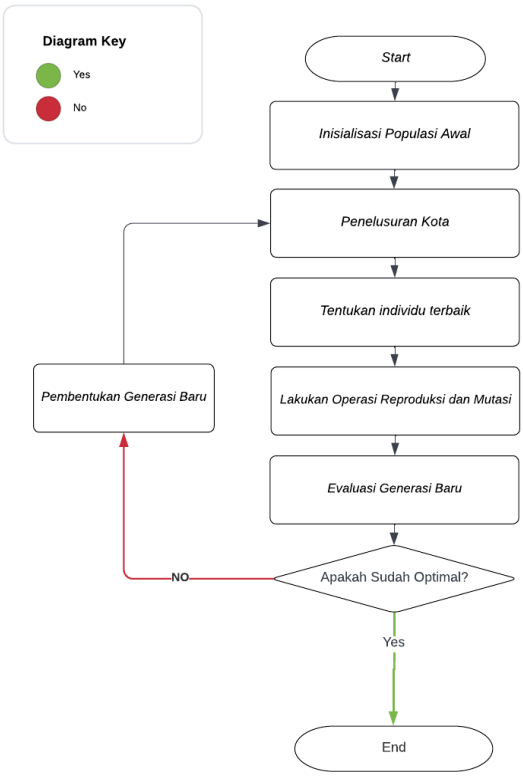
1) Representasikan setiap kota dalam populasi sebagai gen dalam kromosom. Kromosom ini akan mengkodekan urutan kota yang membentuk rute.
2) Inisialisasi populasi awal dengan kromosom-kromosom acak.
3) Evaluasi setiap kromosom dalam populasi berdasarkan jarak rute yang dihasilkan.
4) Pilih individu atau kromosom terbaik (berdasarkan jarak terpendek) untuk menjadi induk reproduksi.
5) Lakukan operasi reproduksi seperti penjelajahan (*crossover*) dan mutasi untuk menghasilkan populasi baru.
6) Lakukan evaluasi dan seleksi generasi baru berdasarkan jarak terpendek.
7) Ulangi langkah 5-6 hingga mencapai kriteria berhenti, misalnya jumlah generasi tertentu atau rute terpendek sudah ditemukan.
8) Pilih individu terbaik sebagai solusi untuk rute terpendek.

Dalam praktiknya, penggunaan algoritma genetika untuk mencari rute terpendek dapat disesuaikan dengan masalah spesifik dan preferensi pengguna. Implementasi algoritma genetika umumnya menggunakan bahasa pemrograman seperti Python, dan untuk mengukur jarak terpendek antara dua titik koordinat, dapat digunakan library seperti Geopy atau Haversine.

3. Hasil dan Pembahasan

Pada bagian ini menjelaskan hasil penelitian yang diperoleh secara detail, dapat dinyatakan dalam bentuk tabel, kode program atau grafik agar mudah dipahami. Dapat dilihat pada tabel berikut:

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538



Gambar 1. Tahap Penerapan Pencarian Rute Terpendek

1) *Import Library*

Pada bagian ini, Kode dimulai dengan mengimpor beberapa library yang akan digunakan, yaitu random, math, folium, dan Nominatim dari geopy.geocoders.

```
import random
import math
import folium
from geopy.geocoders import Nominatim
```

2) Membuat Objek Geocoder

Selanjutnya, objek geocoder dengan nama geolocator dibuat menggunakan Nominatim dari geopy.geocoders. Objek ini akan digunakan untuk mendapatkan koordinat dari nama kota.

```
geolocator = Nominatim(user_agent="my_geocoder")
```

3) Daftar Kota di Jawa Tengah

Terdapat sebuah dictionary bernama cities yang berisi daftar kota di Jawa Tengah beserta koordinat mereka. Awalnya, semua koordinat diatur menjadi None.

```
cities = {
    'Semarang': None,
    'Boyolali': None,
    'Solo': None,
    'Ambarawa': None,
    'Magelang': None,
    'Salatiga': None
}
```


Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

4) Mengambil Koordinat untuk Setiap Kota

Dalam loop for, koordinat setiap kota di Jawa Tengah diambil menggunakan objek geocoder yang telah dibuat sebelumnya. Nama kota dan provinsi ditambahkan ke dalam string untuk melakukan pencarian koordinat. Jika koordinat ditemukan, koordinat tersebut disimpan dalam dictionary cities.

```
for city in cities:
    location = geolocator.geocode(city + ', Jawa Tengah, Indonesia')
    if location:
cities[city] = (location.latitude, location.longitude)
```

5) Fungsi untuk Menghitung Jarak

Terdapat sebuah fungsi bernama distance yang digunakan untuk menghitung jarak antara dua kota. Fungsi ini menggunakan rumus jarak Euclidean antara dua titik dalam koordinat.

```
def distance(city1, city2):
    x1, y1 = cities[city1]
    x2, y2 = cities[city2]
return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

6) Fungsi untuk Menghitung Total Jarak Rute

Fungsi routeDistance digunakan untuk menghitung total jarak dari suatu rute yang diberikan. Fungsi ini menjumlahkan jarak antara setiap pasangan kota dalam rute dan juga jarak antara kota terakhir dan kota pertama.

```
def routeDistance(route):
    distanceSum = 0
    for i in range(len(route)-1):
        distanceSum += distance(route[i], route[i+1])
return distanceSum + distance(route[-1], route[0])
```

7) Menentukan Rute Terpendek Dengan Algoritma Genetika

Fungsi ini merupakan inti dari algoritma genetika, dimana menerima tiga parameter: population (populasi awal), fitness_func (fungsi untuk menghitung fitness individu), dan n_generations (jumlah generasi yang akan dievaluasi). Pada setiap generasi, fungsi ini melakukan evaluasi fitness untuk setiap individu dalam populasi, seleksi individu untuk reproduksi, reproduksi dan mutasi individu terpilih, serta mengganti populasi lama dengan populasi baru. Setelah sejumlah generasi, fungsi ini mengembalikan individu terbaik (rute terpendek).

```
def geneticAlgorithm(population, fitness_func, n_generations):
    for _ in range(n_generations):
        # Evaluasi fitness untuk setiap individu dalam populasi
        fitness_scores = [fitness_func(individual) for individual in population]

        # Seleksi individu untuk reproduksi
        selected_individuals = selection(population, fitness_scores)

        # Reproduksi dan mutasi individu terpilih
        offspring_population = reproduction(selected_individuals)

        # Evaluasi fitness untuk populasi baru
        offspring_fitness_scores = [fitness_func(individual) for individual in
offspring_population]

        # Elitisme: mempertahankan individu terbaik dari populasi sebelumnya
        best_individual = population[fitness_scores.index(max(fitness_scores))]
        offspring_population.append(best_individual)
```


Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

```
        # Mengganti populasi lama dengan populasi baru
        population = offspring_population

    # Mengembalikan individu terbaik setelah sejumlah generasi
    best_individual                                                    =
population[offspring_fitness_scores.index(max(offspring_fitness_scores))]
    return best_individual
```

8) Fungsi Seleksi menggunakan Turnamen

Fungsi ini digunakan untuk seleksi individu untuk reproduksi, menerima dua parameter: population (populasi individu) dan fitness_scores (skor fitness individu dalam populasi). Pada setiap iterasi, fungsi ini memilih dua individu secara acak dan memilih individu dengan fitness terbaik untuk ditambahkan ke daftar individu terpilih. Daftar individu terpilih kemudian dikembalikan.

```
def selection(population, fitness_scores):
    selected_individuals = []
    for _ in range(len(population)):
           # Memilih dua individu secara acak
individual1 = random.choice(population)
        individual2 = random.choice(population)

        # Memilih individu dengan fitness terbaik
        if           fitness_scores[population.index(individual1)]          >
fitness_scores[population.index(individual2)]:
            selected_individuals.append(individual1)
        else:
            selected_individuals.append(individual2)
    return selected_individuals
```

9) Fungsi Reproduksi menggunakan Crossover dan Mutasi

Fungsi ini digunakan untuk melakukan reproduksi dan mutasi individu terpilih, menerima satu parameter: selected_individuals (daftar individu terpilih). Pada setiap iterasi, fungsi ini mengambil dua individu terpilih sebagai orang tua, melakukan crossover menggunakan titik potong acak, dan menghasilkan dua anak. Selain itu, fungsi ini juga melakukan mutasi pada anak-anak dengan probabilitas rendah. Daftar anak-anak yang dihasilkan kemudian dikembalikan.

```
def reproduction(selected_individuals):
    offspring_population = []
    for i in range(0, len(selected_individuals)-1, 2):
        parent1 = selected_individuals[i]
        parent2 = selected_individuals[i+1]

        # Crossover menggunakan titik potong acak
        crossover_point = random.randint(1, len(parent1)-1)
        offspring1 = parent1[:crossover_point] + parent2[crossover_point:]
        offspring2 = parent2[:crossover_point] + parent1[crossover_point:]

        # Mutasi dengan probabilitas rendah
        if random.random() < 0.1:
            mutation_point = random.randint(1, len(offspring1)-1)
            offspring1[mutation_point],      offspring2[mutation_point]      =
offspring2[mutation_point], offspring1[mutation_point]

        offspring_population.append(offspring1)
        offspring_population.append(offspring2)
    # Jika jumlah individu terpilih ganjil, tambahkan individu terakhir ke
populasi baru
    if len(selected_individuals) % 2 != 0:
        offspring_population.append(selected_individuals[-1])
     return offspring_population
```


Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

10) Menentukan dan Menggambar Rute menggunakan Algoritma Genetika

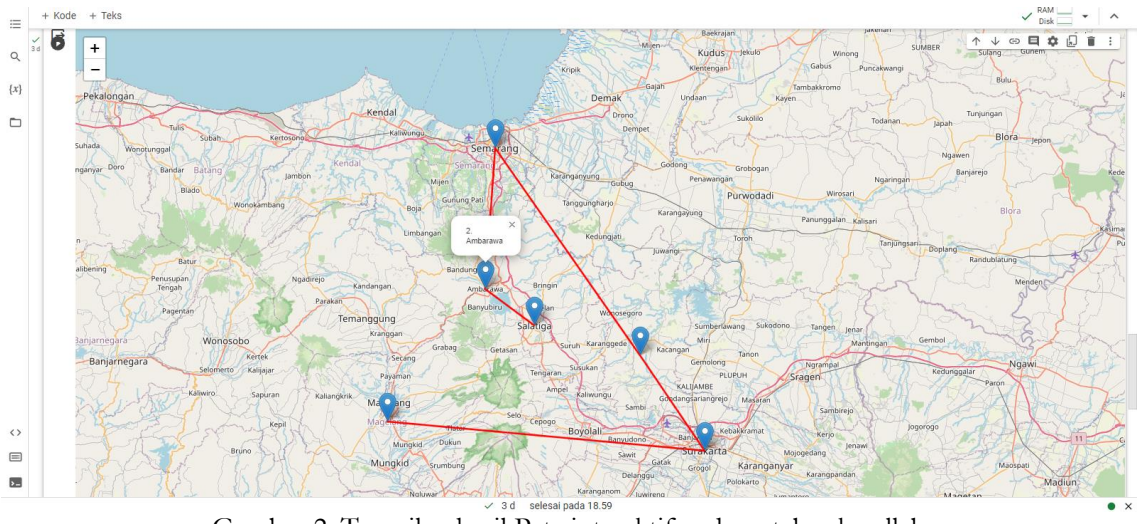
Pada baris pertama, dibuat populasi awal dengan menggunakan daftar kunci kota yang ada dalam variabel cities. Populasi awal ini terdiri dari 10 individu yang memiliki daftar kota yang sama. Pada baris kedua, menjalankan algoritma genetika dengan memanggil fungsi geneticAlgorithm dengan parameter populasi awal, fungsi evaluasi routeDistance, dan jumlah generasi sebanyak 100. Fungsi geneticAlgorithm akan menghasilkan rute terpendek dari kota terdekat ke Salatiga berdasarkan populasi dan fungsi evaluasi yang diberikan. Selanjutnya, menggunakan pustaka Folium untuk menggambar rute terpendek pada peta. membuat objek peta dengan lokasi awal di kota Salatiga dan level zoom 9. Kemudian, menggunakan perulangan for, menggambar garis merah yang menghubungkan setiap kota dalam rute terpendek.

```
population = [list(cities.keys())] * 10  # Populasi awal
best_route = geneticAlgorithm(population, routeDistance, 100)  # Menjalankan
algoritma genetika selama 100 generasi

m = folium.Map(location=cities['Salatiga'], zoom_start=9)

for i in range(len(best_route)-1):
    folium.PolyLine([cities[best_route[i]], cities[best_route[i+1]]],
color='red', weight=2.5).add_to(m)
    folium.Marker(location=cities[best_route[i]], popup=str(i+1) + '. ' +
best_route[i]).add_to(m)
folium.Marker(location=cities[best_route[-1]], popup=str(len(best_route)) + '.
' + best_route[-1]).add_to(m)
```

11) Menampilkan Peta

Ditambahkan penanda pada setiap kota dengan nomor urut dan nama kota. Setelah itu, membuat variabel result yang berisi string "Urutan kota: ". Selanjutnya, menggunakan perulangan for, menambahkan nomor urut dan nama kota ke dalam variabel result. Terakhir, menghapus koma terakhir pada variabel result menggunakan slicing. Pada baris terakhir, menampilkan peta yang telah digambar menggunakan pustaka Folium di dalam notebook.

```
result = "Urutan kota: "
for i, city in enumerate(best_route):
    result += str(i+1) + ". " + city + ", "
result = result[:-2]  # Menghapus koma terakhir
result
m
```



Gambar 2. Tampilan hasil Peta interaktif pada notebook collab

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

Setelah didapat hasil dari rute kota-kota terdekat, hasil tersebut diolah agar dapat menampilkan peta serta rute dan nama jalan dengan menjalankan program pencarian rute terpendek. untuk menampilkan rute antara dua lokasi menggunakan pustaka OSMnx, NetworkX, dan Matplotlib. Program ini meminta pengguna untuk memasukkan latitude dan longitude lokasi awal serta lokasi tujuan, dan kemudian menampilkan rute antara kedua lokasi tersebut dengan penjelasan sebagai berikut:

1) Import Library

Kode pertama adalah mengimpor tiga pustaka yang diperlukan, yaitu OSMnx, NetworkX, dan Matplotlib. Pustaka OSMnx digunakan untuk mengambil data jaringan jalan dari OpenStreetMap, NetworkX digunakan untuk melakukan perhitungan rute terpendek, dan Matplotlib digunakan untuk menampilkan grafik rute.

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

2) Input Lokasi

Fungsi input_lokasi() digunakan untuk meminta pengguna memasukkan latitude dan longitude lokasi awal serta lokasi tujuan. Fungsi ini mengembalikan nilai latitude awal, longitude awal, latitude tujuan, dan longitude tujuan yang dimasukkan oleh pengguna.

```
def input_lokasi():
    lat_awal = float(input("Masukkan Latitude lokasi awal: "))
    lon_awal = float(input("Masukkan Longitude lokasi awal: "))
    lat_tujuan = float(input("Masukkan Latitude lokasi tujuan: "))
    lon_tujuan = float(input("Masukkan Longitude lokasi tujuan: "))
    return lat_awal, lon_awal, lat_tujuan, lon_tujuan
```

3) Input Lokasi

Fungsi input_lokasi() digunakan untuk meminta pengguna memasukkan latitude dan longitude lokasi awal serta lokasi tujuan. Fungsi ini mengembalikan nilai latitude awal, longitude awal, latitude tujuan, dan longitude tujuan yang dimasukkan oleh pengguna.

```
def input_lokasi():
    lat_awal = float(input("Masukkan Latitude lokasi awal: "))
    lon_awal = float(input("Masukkan Longitude lokasi awal: "))
    lat_tujuan = float(input("Masukkan Latitude lokasi tujuan: "))
    lon_tujuan = float(input("Masukkan Longitude lokasi tujuan: "))
    return lat_awal, lon_awal, lat_tujuan, lon_tujuan
```

4) Tampilkan Rute

Fungsi tampilkan_rute() digunakan untuk menampilkan rute antara dua lokasi yang diberikan. Pada awal fungsi ini, menggunakan pustaka OSMnx untuk mengambil data jaringan jalan dari OpenStreetMap berdasarkan latitude dan longitude yang diberikan. Selanjutnya, menggunakan pustaka NetworkX untuk mencari rute terpendek antara dua node yang terdekat dengan lokasi awal dan tujuan yang diberikan. Jika rute ditemukan, menggunakan pustaka Matplotlib untuk menampilkan grafik jaringan jalan beserta rute yang dihitung. Rute ditandai dengan garis biru, sedangkan jalan-jalan lain ditandai dengan warna abu-abu. Fungsi ini juga mencetak nama-nama jalan yang dilewati oleh rute dengan keterangan, seperti dimulai dari jalan tertentu, berbelok ke jalan tertentu, dan berhenti di jalan tertentu.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

```
def tampilkan_rute(lat_awal, lon_awal, lat_tujuan, lon_tujuan):
    G = ox.graph_from_bbox(lat_awal, lat_tujuan, lon_awal, lon_tujuan,
network_type='drive')

    start_node = ox.distance.nearest_nodes(G, lon_awal, lat_awal)
    end_node = ox.distance.nearest_nodes(G, lon_tujuan, lat_tujuan)

    try:
        route = nx.shortest_path(G, start_node, end_node, weight='length')
        edge_colors = ['grey' if (u, v) not in zip(route, route[1:]) else
'blue' for u, v, k in G.edges(keys=True)]
        edge_widths = [1 if color == 'grey' else 3 for color in edge_colors]
        fig, ax = ox.plot_graph(G, edge_color=edge_colors,
edge_linewidth=edge_widths, node_size=3, bgcolor='w', figsize=(30, 30),
dpi=1200)

        # Mendapatkan nama jalan yang dilewati oleh rute dengan garis biru
blue_route_names = []
for u, v, k in G.edges(keys=True):
            if (u, v) in zip(route[:-1], route[1:]):
                data = G.edges[u, v, k]
                if 'name' in data:
                    street_name = data['name']
                    if street_name not in blue_route_names:
                        blue_route_names.append(street_name)

        # Menampilkan nama jalan yang dilewati dengan keterangan
        print("Nama jalan yang dilewati :")
        for i, name in enumerate(blue_route_names):
            if i == 0:
                print(f"Dimulai dari {name} lurus terus")
            elif i == len(blue_route_names) - 1:
                print(f"Berhenti di {name}")
            else:
                print(f"Kemudian berbelok ke {name} lalu lurus terus")

        plt.show()
    except nx.NetworkXNoPath:
        print("Tidak ada jalur yang tersedia antara dua tempat tersebut.")
```

5) Main Menu

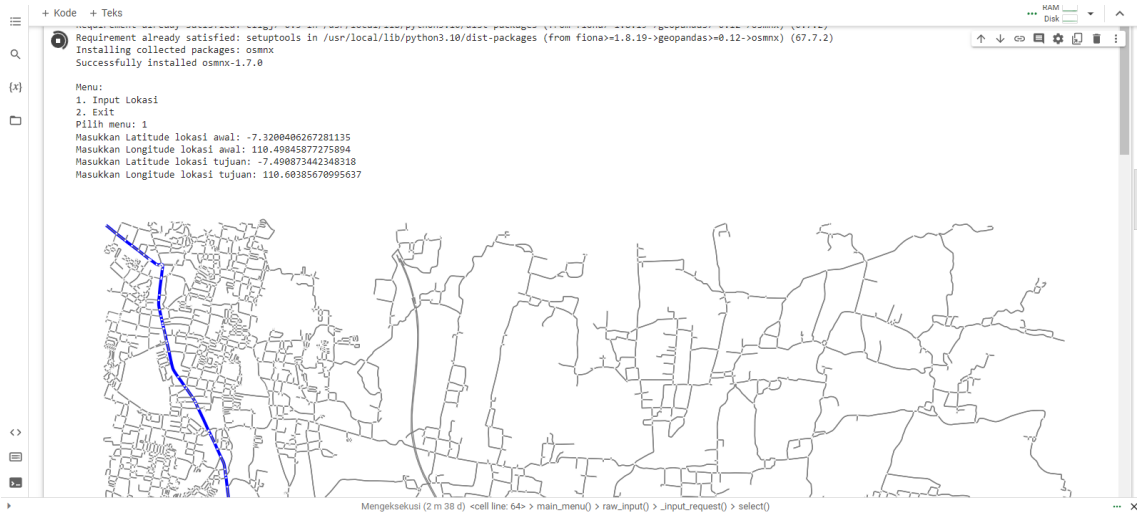
Fungsi main_menu() digunakan untuk menampilkan menu utama dan mengatur alur program. Program akan terus menampilkan menu hingga pengguna memilih untuk keluar. Jika pengguna memilih menu "1", program akan meminta pengguna memasukkan lokasi awal dan tujuan, kemudian menampilkan rute antara kedua lokasi tersebut menggunakan fungsi tampilkan_rute(). Jika pengguna memilih menu "2", program akan berhenti. Jika pengguna memilih menu lainnya, program akan mencetak pesan bahwa pilihan tidak valid.

```
def main_menu():
    while True:
        print("\nMenu:")
        print("1. Input Lokasi")
        print("2. Exit")
        choice = input("Pilih menu: ")
        if choice == "1":
            lat_awal, lon_awal, lat_tujuan, lon_tujuan = input_lokasi()
            tampilkan_rute(lat_awal, lon_awal, lat_tujuan, lon_tujuan)
        elif choice == "2":
            break
        else:
       print("Pilihan tidak valid. Silakan pilih menu yang tersedia.")
       main_menu()
```

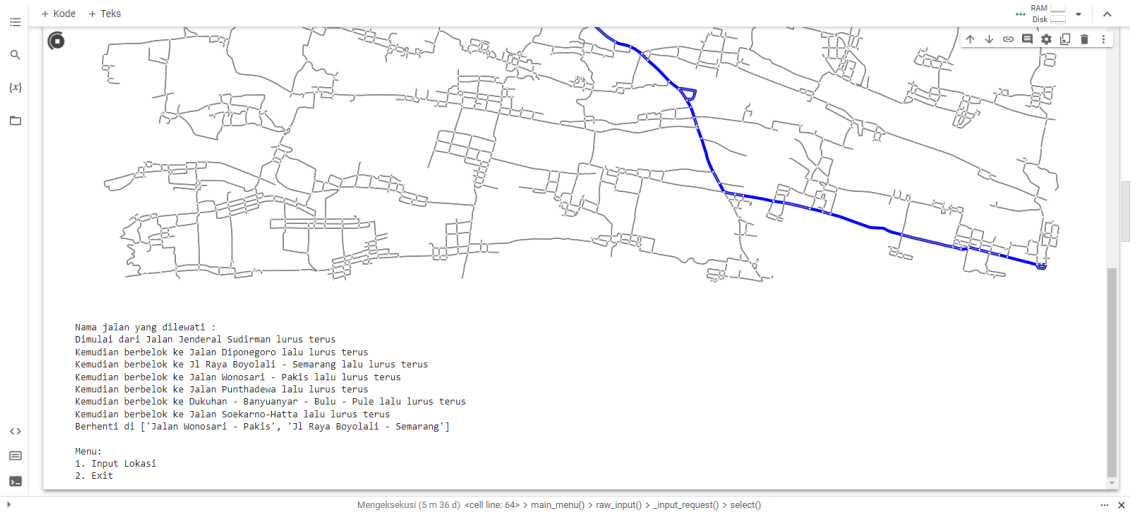

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

6) Menampilkan Peta Dan Nama Jalan

Setelah program dijalankan, akan diminta untuk memasukan latitude dan longitude lokasi awal dan lokasi tujuan kemudian peta yang telah digambar ditampilkan dalam notebook, beserta dengan rute terpendek dan keterangan nama jalan yang urut dari lokasi awal ke lokasi tujuan. Dapat dilihat pada gambar 3 dan 4.



Gambar 3. Tampilan rute peta berdasarkan inputan latitude & longitude pada notebook collab



Gambar 4. Tampilan hasil nama jalan pada notebook collab

4. Kesimpulan

Melalui pengujian dan penyelesaian masalah yang telah dijelaskan, hasil akhir menunjukkan bahwa kode Python ini berhasil mengimplementasikan solusi algoritma genetika untuk mencari rute terpendek menggunakan peta interaktif di Google Collab. Langkah-langkah yang dilakukan meliputi pemilihan kota awal dan tujuan, pembentukan populasi awal, operasi seleksi genetika, rekombinasi, mutasi, evaluasi kecocokan, dan pengulangan proses hingga mencapai solusi optimal. Solusi terbaik yang ditemukan adalah rute dengan jarak terpendek dari kota-kota di sekitar Salatiga menuju Salatiga. Dengan menggunakan penentuan rute terpendek yang dioptimalkan, algoritma genetika dapat menghemat waktu, energi, dan sumber daya yang diperlukan selama perjalanan. Penerapan algoritma

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 681-692
DOI : https://doi.org/10.35870/jimik.v5i1.538

genetika dalam perencanaan rute perjalanan diharapkan memberikan efisiensi dan efektivitas bagi individu dan organisasi yang menggunakannya. Selain itu, dengan pemahaman yang mendalam dan penggunaan yang tepat, algoritma genetika memiliki potensi besar untuk mengoptimalkan solusi dalam berbagai bidang, termasuk transportasi dan logistik. Hal ini memfasilitasi pertumbuhan ekonomi, pariwisata, dan pertukaran informasi antara Salatiga dan kota-kota tetangganya.

5. Ucapan Terima Kasih

Saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada seluruh tim penelitian yang terlibat dalam penulisan jurnal ini. Terima kasih kepada Prof. Ir. Daniel H.F. Manongga, M.Sc., Ph.D. selaku Dekan Fakultas Teknologi Informasi UKSW dan Budhi Kristianto, S.Kom., M.Sc., Ph.D. selaku Kepala Program Studi S1 Teknik Informatika yang telah memberikan arahan dan panduan yang berharga. Juga, terima kasih kepada Bapak Isai Katu dan Ibu Maria Pehenbising sebagai Orang Tua yang telah memberikan kontribusi serta berbagi pengetahuan yang sangat berharga dalam penulisan jurnal ini. Terima kasih juga kepada tim teknis yang telah membantu dalam pengumpulan dan analisis data. Saya juga ingin mengucapkan terima kasih kepada departemen riset universitas yang telah memberikan dukungan dan sumber daya yang dibutuhkan. Terakhir, tetapi tidak kalah pentingnya, terima kasih kepada keluarga dan teman-teman yang selalu memberikan dukungan dan motivasi selama proses penelitian ini. Tanpa kontribusi dan dukungan kalian semua, jurnal ini tidak akan terwujud. Terima kasih sekali lagi kepada semua pihak yang telah turut serta dalam keberhasilan penulisan jurnal ini.

6. Daftar Pustaka

[1] Irianti, A., Cokrowibowo, S., & Aswandi, A. (2021). Optimasi Multiple Traveling Salesman Problem dengan Algoritma Genetika pada Kasus Model Rute Terpendek Penjemputan Sampah di Kabupaten Majene. *Proceeding KONIK (Konferensi Nasional Ilmu Komputer)*, *5*, 86-89.

[2] Melladia, M. (2020). Algoritma Genetika Menentukan Jalur Jalan dengan Lintasan Terpendek (Shortest Path). *Prosiding SISFOTEK*, *4*(1), 112-117.

[3] Mubarok, A. Y., & Chotijah, U. (2021). Penerapan Algoritma Genetika Untuk Mencari Optimasi Kombinasi Jalur Terpendek Dalam Kasus Travelling Salesman Problem. *Jurnal Teknologi Terpadu*, *7*(2), 77-82. DOI: https://doi.org/10.54914/jtt.v7i2.424.

[4] Muhandhis, I., Shubhan, M., Dani, H. I., Rakasyah, A., Ritonga, A. S., & Sari, M. U. (2023). Pencarian Rute Terpendek Tim Promosi Kampus dengan Menggunakan Algoritma Genetik. *Jurnal Teknologi dan Manajemen*, *4*(1), 6-12. DOI: https://doi.org/10.31284/j.jtm.2023.v4i1.4106.

[5] Oktaviandi, R. B., Hadi, M. S. T., Santoso, A. G., & El Maidah, N. (2019). Perbandingan Algoritma Genetika dengan Algoritma Greedy Untuk Pencarian Rute Terpendek. *INFORMAL: Informatics Journal*, *3*(1), 6-11.

[6] Ramadhania, S. E., & Rani, S. (2021). Implementasi kombinasi algoritma genetika dan tabu search untuk penyelesaian travelling salesman problem. *AUTOMATA*, *2*(1).

[7] Saputra, I., & Ahmad, D. (2020). Algoritma Genetika Untuk Menentukan Jalur Terpendek Wisata Kota Bukittinggi. *Journal of Mathematics UNP*, *5*(1). DOI: http://dx.doi.org/10.24036/unpjomath.v5i1.8905.

[8] Tohari, A., & Astuti, Y. P. (2023). PENERAPAN ALGORITMA GENETIKA DALAM MENENTUKAN RUTE TERPENDEK PT. POS CABANG LAMONGAN. *MATHunesa: Jurnal Ilmiah Matematika*, *11*(03), 458-467.

[9] Zukhri, Z. (2014). Algoritma Genetika Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi. *Yogyakarta: Andi Offset*.