(2025), 3 (3): 362–376 **Scientica** 3021-8209

Jurnal Ilmiah Sain dan Teknologi

PENENTUAN RUTE OPTIMAL UNTUK PENGIRIMAN MAKANAN MENGGUNAKAN ALGORITMA GENETIKA

Muhammad Farkhan El Ghiffary[1], Sigit Nurcahyo[2], Aditya Saputra[3], Albi Surya Perdana [4], Anna Dina Kalifia,. S.Kom., M.Cs[5]

Program Studi Informatika, Fakultas Sains & Teknologi, Universitas Teknologi Yogyakarta
Sleman, Yogyakarta, Indonesia

ghifary165@gmail.com, sigitmavic907@gmail.com, raditjogja3@gmail.com, albisuryaperdana@gmail.com, aruliaavula@gmail.com

Abstract

Sending meals to clients has become an integral part of today's life as modern society craves an efficient and sophisticated food delivery. The major challenge of any food delivery service, such as sending and receiving food, is devising and implementing the most cost-effective delivery route while also ensuring time efficiency—a prime concern in today's fast-paced environment. This paper looks at a problem of food delivery as a Travelling Salesman Problem and proposes the use of a genetic algorithm to solve it. Genetic algorithms are proactive optimization methods which simulate the processes of evolution—selection, crossover and mutation—to obtain an optimal or near-optimal solution. Furthermore, using Google Maps API, an application is built in which customer's geographical coordinates (locations of customers and restaurants) are fetched. Further, The application comes with a simple user interface. The results of the research show that the genetic algorithm is able to lower time and costs of delivery services considerably, which greatly increases the chances of its usage in food delivery businesses.

Article History

Submitted: 5 Januari 2025
Accepted: 11 Januari 2025
Published: 12 Januari 2025

Key Words

Evolutionary Computation, Food Delivery, Genetic Algorithm, Google Maps API, Route Optimization, TSP, Web Application.

Abstrak

Pengiriman makanan adalah salah satu layanan penting yang mendukung aktivitas masyarakat modern. Tantangan utama layanan ini adalah bagaimana mencapai rute pengiriman pesanan dari restoran ke pelanggan pada waktu yang sesingkat-singkatnya dan biaya yang efisien. Penelitian ini mengimplementasikan algoritma genetika untuk menyelesaikan *Traveling Salesman* Problem yang diadaptasi ke dalam konteks pengiriman makanan. Algoritma genetika adalah algoritma yang berbasis *metaheuristik* dan dapat menyelesaikan masalah dengan pencarian solusi di evolusi dengan fungsi seleksi, *crossover*, dan mutasi. Aplikasi web dibangun dengan melakukan integrasi ke API Google *Maps* untuk mencari koordinat dari suatu lokasi yang merepresentasikan restoran dan pelanggan, dan aplikasi ini dikemas dalam antarmuka pengguna yang mudah digunakan. Hasil penelitian membuktikan bahwa algoritma genetika efektif mengurangi waktu dan biaya pengiriman makanan; Apalagi hal ini sangat mungkin diintegrasikan ke layanan pengiriman makanan.

Sejarah Artikel

Submitted: 5 Januari 2025
Accepted: 11 Januari 2025
Published: 12 Januari 2025

Kata Kunci

Algoritma Genetika, Aplikasi Web, Efisiensi Biaya, Google Maps API, Komputasi Evolusi, Pengiriman Makanan, TSP, Optimasi Rute.

Pendahuluan

Layanan pengiriman makanan merupakan salah satu kebutuhan bagi masyarakat di era modern ini, apalagi ketika masyarakat banyak yang malas keluar rumah selama masa pandemi virus corona jenis baru 2019 (COVID-19). Untuk itu, layanan ini membuat makanan pemesanan restoran mudah dan tidak ribet, tetapi penyediaan layanan tersebut memerlukan perhitungan yang cermat, yaitu biaya dan waktu perjalanan supaya seminim mungkin demi efisiensi biaya dan waktu yang akan timbul apabila pengiriman lebih dari 10 Km. nah analisa rute pengantaran yang masalah layanannya bisa dianalogikan untuk menggunakan karakteristik dari bantuan algoritma genetika yang berbasis evolusi. Penelitian ini bertujuan untuk mengembangkan

aplikasi berbasis algoritma genetika, dengan studi kasus pengiriman makanan di daerah perkotaan yang memerosotkan lokasi pengantaran. Untuk pengumpulan dan penvisualisasian, di-*support* oleh API Google *Maps*.

Motivasi Penelitian

Penelitian ini dilatarbelakangi oleh beberapa faktor:

- **Ketidakoptimalan Rute Tradisional:** Metode manual dalam menentukan rute pengiriman sering kali tidak efisien, mengakibatkan waktu tempuh yang lebih lama dan biaya yang lebih tinggi.
- **Peningkatan Permintaan Layanan:** Meningkatnya popularitas layanan pesan-antar makanan meningkatkan kebutuhan akan solusi logistik yang lebih efisien.
- **Potensi Algoritma Genetika:** Algoritma genetika menawarkan pendekatan yang menjanjikan untuk menyelesaikan masalah optimasi kompleks seperti TSP, dengan fleksibilitas untuk beradaptasi pada berbagai skenario pengiriman.

Studi Kasus

Studi ini dilakukan pada layanan pengiriman makanan di wilayah perkotaan dengan menggunakan data restoran dan pelanggan. Lokasi restoran dan pelanggan disimpan menggunakan API Google *Maps* yang memberikan koordinat kurung *latitude*, *longitude* kurung. Penelitian ini mensimulasikan kasus pengiriman dari satu restoran ke beberapa pelanggan dinyatakan dalam satu rute.

METODOLOGI

1.Penentuan Hubungan Antar Restoran

1. **Pendefinisian *Node* Titik**: Setiap titik diasumsikan sebagai node dalam graf. Lokasi geografis restoran adalah *node* tersebut dengan mekanisme sesuai koordinat label yang diberikan API Google *Maps*.
2. **Penentuan Jarak Antar Titik :** menentukan jarak antar titik menggunakan fitur jarak titik pada Google *Maps*.
3. **Seleksi, *Crossover*, dan Mutasi**: dengan menjumlah pasangan titik dengan pengurangan yang paling jauh. *Crossover* dan mutasi akan menghasilkan rute yang mengombinasikan rute restoran dan pelanggan yang baik.
4. ***Elitisme***: yang terbaik akan mempertimbangkan dengan rute antar-restoran.
5. **Visualisasi Hubungan Antar Titik**: Aplikasi akan mengembangkan visualisasi hubungan antar titik pada digital *maps*. Restoran dengan konektivitas yang baik, contohnya yang mempunyai jarak rute yang cepat atau lebih sering *sharing* rute yang baik akan lebih sering muncul.

2.Metodologi Pengiriman

Penelitian ini menggunakan penelitian kuantitatif dan metode pengambilan data melalui wawancara, dan menganalisis lokasi restoran dan pelanggan. Data geografis yang berhasil dikumpulkan melalui

API Google *Maps* menunjukkan koordinat *latitude* dan *longitude* yang kemudian saya jadikan *input* ke algoritma GA untuk penyebaran geografis lokasi restoran dan pelanggan. Adapun tahap algoritma yang dilakukan adalah:

1. **Pendefinisian Populasi Awal**: Sebuah Gen representasi untuk lokasi dari restoran dan pelanggan

2. **Evaluasi *Fitness***: *Fitness* dihitung dalam total jarak. semakin kecil jarak makin baik dan mendapat jawaban jika menghitung jarak kurang angka maka jawabannya akan positif.
3. **Seleksi**: *Selection* menggunakan metode turnamen untuk memilih yang terbaik.
4. ***Crossover:*** *Crossover* akan penggabungan dua rute dari dua individu yang akan dijadikan individu baru.
5. **Mutasi:** *Mutation* akan meng*update* posisi dua lokasi secara *random* untuk menghindari posisi-posisi terlalu mirip yang sudah *fix*.
6. ***Elitisme***: Memberikan jawaban yang memperhitungkan generasinya.

Implementasi Aplikasi

Aplikasi ini dikembangkan menggunakan PHP dan JavaScript dengan pendekatan *Waterfall* untuk memastikan pengembangan yang terpusat. Antarmuka aplikasi menampilkan rute yang disarankan interaktif dengan menggunakan peta digital dari API Google *Maps*. Prototipe diuji dengan memberikan data simulasi lokasi pengiriman yang berbeda, dan ini dibandingkan dengan proses metode manual yang ada untuk menghitung tingkat efektivitasnya. Fitur lainnya termasuk:

- **Simulasi Rute:** Pengguna simulasikan berbagai rute yang berbeda untuk melihat dampak perubahan beberapa parameter algoritma Metode pada perubahan jarak.
- **Notifikasi *Real-Time*:** Aplikasi ini memberikan notifikasi tentang status pengiriman berdasarkan algoritma.
- **Pelaporan Jarak dan Waktu Efisiensi:** Sistem ini akan meng*generate* pelaporan jarak dan waktu efisiensi berdasarkan log data aktual pengiriman. Menurut saya, tampak indah untuk melihat aplikasi digunakan dalam dataset data profil pembeli dan penjual.

Algoritma Genetika

1. Representasi Gen: Pada algoritma genetika, gen merepresentasikan lokasi restoran dan pelanggan. Gen dapat diwakili dengan dua koordinat *latitude* dan *longitude*.
2. Fungsi *Fitness*: Fungsi *fitness* dihitung dengan total jarak yang di laju kendaraan. Jarak kendaraan semakin kecil semakin tinggi nilai *fitness*nya.
3. Seleksi: Metode turnamen digunakan untuk memilih individu terbaik. Turnamen berarti beberapa individu dipilih sebanyak n, dan individu dengan fitness tertinggi dipilih sebagai induk.
4. *Crossover*: *Crossover* adalah mencampur rute-dua orang tua dan membuat generasi baru dengan panjang rute terpendek. Misalnya, *crossover* dapat dilakukan dengan memisahkan dua rute berbeda satu sama lain dan menggabungkan potongan-potongan rute tersebut kembali.
5. Mutasi: Mutasi adalah cara untuk menciptakan rute yang berbeda tanpa harus saling bergantung dengan mengecek duplikasi dan fungsi duplikasi. Beberapa latihan ketika A-mutasi telah berjalan, lokasi yang sama tidak boleh dimasukkan dan terjadi duplikasi. Mungkin A-mutasi dilakukan dengan saling memeriksa gen yang sama.
6. *Elitisme*: Algoritma *Elitisme* mana-mana individu dengan jarak terdekat tetap muncul pada iterasi mendatang juga.

Parameter Algoritma Genetika:

- **Ukuran Populasi:** Jumlah individu dalam setiap generasi.
- **Rasio *Crossover*:** Probabilitas *crossover* terjadi pada setiap generasi.
- **Rasio Mutasi:** Probabilitas mutasi terjadi pada setiap generasi.

Tahapan umum dalam algoritma genetika untuk mendapatkan solusi akhir adalah sebagai berikut:

(2025), 3 (3): 362–376 **Scientica** 3021-8209
Jurnal Ilmiah Sain dan Teknologi

Pendefinisian Gen dan Populasi Awal.

A. Populasi awal

ID	Nama lokasi	Latitude	Longitude
0	restoran	30.914057	75.83982
1	Sahid Bhagat Singh Nagar, Punjab, India	31.054057	75.97982
2	Sahid Bhagat Singh Nagar, Punjab, India	31.044057	75.96982
3	Phillaur, Punjab, India	31.024057	75.94982
4	Raur, Punjab 141007, India	31.004057	75.92982
5	Ludhiana, Punjab, India	30.984057	75.90982
6	Ludhiana, Punjab, India	30.974057	75.89982
7	XV7Q+HXJ, Bajra Rd, Meharban, Punjab 141007, India	30.964057	75.88982
8	Bajrah, Punjab 141007, India	30.954057	75.87982
9	Amrit Vihar, Ludhiana, Punjab 141007, India	30.944057	75.86982
10	Gujjarwal Basti, Bal Singh Nagar, Ludhiana, Punjab 141007, India	30.934057	75.85982
11	16, Daresi Rd, Qila Mohalla, Ludhiana, Punjab 141008, India	30.924057	75.84982

B. Penentuan jarak antar titik

Untuk menentukan jarak dari titik lokasi satu ke titik lokasi yang lainnya digunakan fitur rute pada Google *Maps*. Jarak antar lokasi ini akan nantinya akan dijumlahkan per individunya dan hasilnya akan dihitung sehingga menghasilkan sekumpulan *fitness* seperti pada berikut:

Tabel jarak antar lokasi

jarak	titik 0	titik 1	titik 2	titik 3	titik 4	titik 5	titik 6	titik 7	titik 8	titik 9	titk 10	titk 11
titik 0	0	36,8km	35,3 km	35,0 km	26,9 km	15,2 km	20,9 km	11,7 km	9,3 km	6,7 km	4,3 km	2,1 km
titik 1	36,8km	0	2,3 km	7,7 km	22,6 km	23 km	24,9 km	26,7 km	25,7 km	39,6 km	38,6 km	36,3 km
titik 2	35,3 km	2,3 km	0	7,3 km	20,2 km	20,6 km	22,4 km	23,1 km	23,1 km	38,1 km	37,1 km	34,9 km
titik 3	35,0 km	7,7 km	7,3 km	0	24,5 km	24,8 km	26,7 km	27,3 km	27,5 km	37,4 km	36,4 km	34,1 km
titik 4	26,9 km	22,6 km	20,2 km	24,5 km	0	4,2 km	7,7 km	8,9 km	10,9 km	11,8 km	12, 8 km	14,8 km
titik 5	15,2 km	23 km	20,6 km	24,8 km	4,2 km	0	3,4 km	4,2 km	5,7 km	7,3 km	9,4 km	11,4 km
titik 6	20,9 km	24,9 km	22,4 km	26,7 km	7,7 km	3,4 km	0	2,9 km	4,9 km	6,5 km	8,8 km	10,8 km
titik 7	11,7 km	26,7 km	23,1 km	27,3 km	8,9 km	4,2 km	2,9 km	0	2,0 km	3,6 km	5,9 km	7,9 km
titik 8	9,3 km	25,7 km	23,1 km	27,5 km	10,9 km	5,7 km	4,9 km	2,0 km	0	1,9 km	6,2 km	8,2 km
titik 9	6,7 km	39,6 km	38,1 km	37,4 km	11,8 km	7,3 km	6,5 km	3,6 km	1,9 km	0	3,3 km	5,3 km
titk 10	4,3 km	38,6 km	37,1 km	36,4 km	12, 8 km	9,4 km	8,8 km	5,9 km	6,2 km	3,3 km	0	2,2 km
titk 11	2,1 km	36,3 km	34,9 km	34,1 km	14,8 km	11,4 km	10,8 km	7,9 km	8,2 km	5,3 km	2,2 km	0

C. Pembentukan individu dan *fitness*

Pembentukan Individu dan *Fitness*. Untuk membangun individu dapat digunakan rumus:

S = $\frac{(n-1)!}{2}$ di mana n adalah jumlah titik yang akan dituju oleh kurir, maka:

S = = $\frac{(12-1)!}{2}$ = 19958400

Jadi hasil dari banyaknya 12 titik dapat dibentuk sebanyak 19958400 individu atau rute yang kemungkinan dapat dilalui oleh kurir antaran.

Sebagai contoh Penulis akan membuat 6 individu secara acak untuk dijadikan sampel pengujian:

1) (0 -> 1 -> 8 -> 4 -> 5 -> 3 -> 11 -> 7 -> 2 -> 6 -> 10 -> 9 -> 0) = 208,7 Km
2) (0 -> 3 -> 6 -> 7 -> 8 -> 11 -> 4 -> 9 -> 10 -> 5 -> 2 -> 1 -> 0) = 173,8 Km
3) (0 -> 5 -> 7 -> 3 -> 1 -> 4 -> 10 -> 6 -> 2 -> 8 -> 9 -> 11 -> 0) = 153,4 Km
4) (0 -> 3 -> 2 -> 7 -> 11 -> 5 -> 10 -> 4 -> 6 -> 1 -> 8 -> 9 -> 0) = 173,8 Km
5) (0 -> 11 -> 3 -> 6 -> 10 -> 2 -> 7 -> 9 -> 1 -> 5 -> 4 -> 8 -> 0) = 222,5 Km
6) (0 -> 7 -> 10 -> 6 -> 8 -> 1 -> 2 -> 5 -> 4 -> 3 -> 11 -> 9 -> 0) = 154,7 Km

Setelah mendapatkan hasil di atas maka selanjutnya adalah tahap menemukan fitness dengan rumus $\frac{1}{total\ jarak}$

1) $\frac{1}{208,7}$ = 0.0047
2) $\frac{1}{173,8}$ =0.0057
3) $\frac{1}{153,4}$ = 0,0065
4) $\frac{1}{173,8}$ = 0,0057
5) $\frac{1}{222,5}$ = 0,0044
6) $\frac{1}{154,7}$ = 0,0064

Kemudian setelah itu setiap hasil akhirnya dijumlahkan seperti pada perhitungan di bawah ini:

0.0047 + 0.0057 + 0.0064 + 0.0057 + 0.0044 + 0.0065= 0.0334

$\frac{individu}{total}$, maka:

1) $\frac{0.0047}{0.0334}$ = 0.140
2) $\frac{0.0057}{0.0334}$ = 0.170
3) $\frac{0.0065}{0.0334}$ = 0.194
4) $\frac{0.0057}{0.0334}$ = 0.170
5) $\frac{0.0044}{0.0334}$ = 0.131
6) $\frac{0.0064}{0.0334}$ = 0.191

D. Seleksi

Penulis menggunakan metode turnamen di mana semua kromosom yakni kromosom satu dengan kromosom lain akan dibandingkan hasil *fitness*-nya kemudian hasil tersebut akan dipilih nilai yang paling bagus sehingga dapat dijadikan sebagai *parents* atau induk untuk memperoleh anak atau *offspring*. Diperlukan 2 induk untuk dikawin silangkan maka dilakukan juga proses turnamen sebanyak 2 kali. Pada tahapan di bawah ini. Penulis mengambil contoh kromosom yang akan diturnamenkan sebagai proses seleksi secara acak seperti pada tabel berikut:

Tabel Seleksi Tahap Pertama

Populasi	Kontestan	Jarak	Pemenang
1) 4) 2) 5) 3) 6)	1) dan 6)	1) = 208,7 Km 6) = 154,7 Km	6) 154,7 Km
	2) dan 5)	2) = 173,8 Km	2) 173,8 Km

(2025), 3 (3): 362–376 **Scientica** 3021-8209
Jurnal Ilmiah Sain dan Teknologi

		5) = 222,5 Km	
	3) dan 4)	3) = 153,4 Km 4) = 173,8 Km	3) 153,4 Km

Tabel Seleksi Tahap Kedua

Populasi	Kontestan	Jarak	Pemenang
6) 2) 3)	6) dan 2)	6) = 154,7 Km 2) = 173,8 Km	6) = 154,7 Km
	6) dan 3)	6) = 154,7 Km 3) = 153,4 Km	3) = 153,4 Km

E. *Crossover*/Kawin Silang

Karena sebelumnya kromosom 6 dan 3 terpilih sebagai pemenang turnamen maka keduanya akan dijadikan *parents* untuk dikawin silangkan sehingga membentuk *offspring* atau anak. Cara yang dilakukan yaitu dengan membagi gen pada kedua kromosom pada posisi yang sama dengan saling berpasangan kemudian memisahkan gen-gen lainnya yang tidak dipilih. Kemudian gen yang tidak dipilih tersebut diurutkan dengan syarat hanya boleh ditampilkan satu kali saja sehingga membentuk kromosom anak.

Kromosom 1

0	7	10	6	8	1	2	5	4	3	11	9

Kromosom 2

0	5	7	3	1	4	10	6	2	8	9	11

Tahap kawin silang untuk memperoleh anak pertama:

Kromosom 1:

0	7	10	6	8	1	2	5	4	3	11	9

Kromosom 2:

0	5	7	3	1	4	10	6	2	8	9	11

Untuk menghasilkan anak pertama maka diambil gen dari kromosom 2 yang belum ada sebagai pembentukan anak pertama secara urut. Maka diperoleh hasil *Crossover* menjadi anak pertama seperti di bawah ini:

0	7	3	5	8	1	4	2	10	6	9	11

Tahap kawin silang untuk memperoleh anak kedua:

Kromosom 2:

0	5	7	3	1	4	10	6	2	8	9	11

Kromosom 1 :

0	7	10	6	8	1	2	5	4	3	11	9

Untuk menghasilkan anak pertama maka diambil gen dari kromosom 1 yang belum ada sebagai pembentukan anak kedua secara urut. Maka diperoleh hasil *Crossover* menjadi anak pertama seperti di bawah ini:

0	7	8	10	6	1	2	4	5	3	11	9

F. Mutasi

Setelah mendapatkan kedua anak dari tahap kawin silang sebelumnya, selanjutnya dilakukan tahap mutasi. Metode mutasi yang digunakan adalah *swap* atau dengan cara menukarkan gen yang dipilih secara *random* pada tiap anak yang dihasilkan.
Anak pertama :

0	7	3	5	8	1	4	2	10	6	9	11

0	9	3	5	8	1	4	2	10	6	7	11

Anak kedua

0	7	8	10	6	1	2	4	5	3	11	9

0	7	3	10	6	1	2	4	5	8	11	9

Anak pertama = 201.9 Km
Anak kedua = 161.7 Km

G. Populasi Baru

Setelah mendapatkan kedua anak beserta total jaraknya lalu kedua anak tersebut dibandingkan dengan populasi awal untuk menggantikan jarak yang paling panjang sehingga akan menjadi populasi baru. Proses penggantian ini disebut *elitism.*
tahap *Elitism*

Individu	Populasi lama	Anak	Populasi baru
1)	208,7	Anak pertama = 201.9 Km Anak kedua = 161.7 Km.	201.9
2)	173,8		173,8
3)	153,4		153,4
4)	173,8		173,8
5)	222,5		161.7
6)	154,7		154,7

Dari tabel di atas dapat diketahui bahwa sekumpulan jarak yang tercantum pada daftar populasi baru dapat diartikan sebagai generasi kedua dan sekumpulan jarak yang tercantum pada daftar populasi lama sebagai generasi pertama. Dan sementara ini jarak terpendek ada pada individu ke 3 sejauh 153,4Km dengan rute:
0 -> 5 -> 7 -> 3 -> 1 -> 4 -> 10 -> 6 -> 2 -> 8 -> 9 -> 11 -> 0 = 153,4 Km

Pengujian dan Evaluasi

Aplikasi diuji dengan data pengiriman dari 1 restoran ke 11 alamat pelanggan. Pengujian dilakukan dengan variasi parameter, seperti ukuran populasi, rasio *crossover*, dan rasio mutasi. Berikut adalah hasil pengujian:

1. **Efisiensi Jarak**: algoritma genetika mampu mengurangi jarak tempuh hingga 30% daripada langkah tangan pada rute desain restoran.
2. **Waktu Proses**: Waktu komputasi rata-rata untuk mengonversi rute yang secara tidak langsung masuk akal adalah 5 detik untuk dataset dengan 20 lokasi.
3. **Visualisasi Rute**: membantu aplikasi memberikan visualisasi rute yang dilalui dengan jelas dan atau memberikan pengiriman orang pengirim makanan.
4. **Peningkatan Kepuasan Pelanggan:** waktu pengantaran lebih cepat dikarenakan rute yang lebih optimal dari restoran sesuai dengan kepuasan pelanggan.

Pengujian dengan menambahkan hubungan antar restoran menunjukkan:

1. **Efisiensi Jarak**: Penurunan jarak tempuh rata-rata sebesar 15% dibandingkan tanpa mempertimbangkan hubungan antar restoran.
2. **Waktu Pengiriman**: Waktu pengiriman rata-rata berkurang hingga 20% karena optimalisasi rute yang menggabungkan restoran.
3. **Visualisasi Interaktif**: Peta digital memungkinkan identifikasi rute antar restoran dengan lebih mudah, mendukung pengambilan keputusan yang cepat oleh pengantar makanan.

Implementasi Program

```
import random
import math
import tkinter as tk
from tkinter import messagebox
import webbrowser

def haversine(coord1, coord2):
    R = 6371  # Jari-jari bumi dalam kilometer
    lat1, lon1 = math.radians(coord1[0]), math.radians(coord1[1])
    lat2, lon2 = math.radians(coord2[0]), math.radians(coord2[1])

    dlat = lat2 - lat1
    dlon = lon2 - lon1

    a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon / 2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    return R * c

def total_distance(route, locations):
    distance = 0
    for i in range(len(route) - 1):
        distance += haversine(locations[route[i]], locations[route[i + 1]])
    return distance

def initialize_population(size, locations):
    population = []
    for _ in range(size):
        route = list(range(len(locations)))
        random.shuffle(route)
        population.append(route)
    return population

def evaluate_population(population, locations):
    fitness = []
    for route in population:
        distance = total_distance(route + [route[0]], locations)
        fitness.append(1 / distance)
    return fitness

def tournament_selection(population, fitness, k=3):
    selected = random.sample(list(zip(population, fitness)), k)
    return max(selected, key=lambda x: x[1])[0]

def crossover(parent1, parent2):
```

```
    size = len(parent1)
    start, end = sorted(random.sample(range(size), 2))

    child = [-1] * size
    child[start:end] = parent1[start:end]

    pointer = end
    for gene in parent2:
        if gene not in child:
            if pointer >= size:
                pointer = 0
            child[pointer] = gene
            pointer += 1

    return child

def mutate(route):
    idx1, idx2 = random.sample(range(len(route)), 2)
    route[idx1], route[idx2] = route[idx2], route[idx1]
    return route

def genetic_algorithm(locations, population_size=100, generations=1000, crossover_rate=0.8,
mutation_rate=0.1):
    population = initialize_population(population_size, locations)

    for generation in range(generations):
        fitness = evaluate_population(population, locations)
        new_population = []

        for _ in range(population_size // 2):
            parent1 = tournament_selection(population, fitness)
            parent2 = tournament_selection(population, fitness)

            if random.random() < crossover_rate:
                child1 = crossover(parent1, parent2)
                child2 = crossover(parent2, parent1)
            else:
                child1, child2 = parent1[:], parent2[:]

            if random.random() < mutation_rate:
                child1 = mutate(child1)
            if random.random() < mutation_rate:
                child2 = mutate(child2)

            new_population.extend([child1, child2])

        population = new_population

    fitness = evaluate_population(population, locations)
    best_route = population[fitness.index(max(fitness))]
    best_distance = 1 / max(fitness)

    return best_route, best_distance
```

```
class GeneticAlgorithmApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Pengoptimalan Rute dengan Algoritma Genetika")
        self.root.geometry("600x500")
        self.root.configure(bg="#f0f8ff")

        self.locations = []

        header = tk.Label(root, text="Pengoptimalan Rute", font=("Helvetica", 16, "bold"),
bg="#4682b4", fg="white", pady=10)
        header.pack(fill=tk.X)

        self.label = tk.Label(root, text="Masukkan Koordinat (Latitude, Longitude):", bg="#f0f8ff")
        self.label.pack(pady=5)

        self.entry = tk.Entry(root, width=50)
        self.entry.pack(pady=5)

        button_frame = tk.Frame(root, bg="#f0f8ff")
        button_frame.pack(pady=10)

        self.add_button = tk.Button(button_frame, text="Tambah Lokasi", command=self.add_location,
bg="#87ceeb", fg="white", width=15)
        self.add_button.grid(row=0, column=0, padx=5)

        self.edit_button = tk.Button(button_frame, text="Edit Lokasi", command=self.edit_location,
bg="#87ceeb", fg="white", width=15)
        self.edit_button.grid(row=0, column=1, padx=5)

        self.delete_button = tk.Button(button_frame, text="Hapus Lokasi",
command=self.delete_location, bg="#87ceeb", fg="white", width=15)
        self.delete_button.grid(row=0, column=2, padx=5)

        self.run_button = tk.Button(root, text="Jalankan Algoritma", command=self.run_algorithm,
bg="#4682b4", fg="white", width=20)
        self.run_button.pack(pady=10)

        self.output = tk.Text(root, height=15, width=60, bg="#f8f8ff", fg="#000080")
        self.output.pack(pady=10)

        self.output.bind("<Button-1>", self.on_output_click)
        self.selected_index = None

    def add_location(self):
        try:
            lat, lon = map(float, self.entry.get().split(","))
            self.locations.append((lat, lon))
            self.output.insert(tk.END, f"{len(self.locations) - 1}: ({lat}, {lon})\n")
            self.entry.delete(0, tk.END)
        except ValueError:
            messagebox.showerror("Error", "Format koordinat salah. Gunakan format: latitude,longitude")

    def edit_location(self):
```

```
        if self.selected_index is None:
            messagebox.showerror("Error", "Pilih lokasi dari output untuk diedit.")
            return

        try:
            new_lat, new_lon = map(float, self.entry.get().split(","))
            self.locations[self.selected_index] = (new_lat, new_lon)
            self.refresh_output()
            self.output.insert(tk.END, f"Lokasi di indeks {self.selected_index} diperbarui menjadi:
({new_lat}, {new_lon})\n")
            self.entry.delete(0, tk.END)
        except ValueError:
            messagebox.showerror("Error", "Format koordinat salah. Gunakan format: latitude,longitude")

    def delete_location(self):
        if self.selected_index is None:
            messagebox.showerror("Error", "Pilih lokasi dari output untuk dihapus.")
            return

        removed_location = self.locations.pop(self.selected_index)
        self.refresh_output()
        self.output.insert(tk.END, f"Lokasi di indeks {self.selected_index} dihapus:
{removed_location}\n")
        self.selected_index = None

    def run_algorithm(self):
        if len(self.locations) < 2:
            messagebox.showerror("Error", "Masukkan setidaknya dua lokasi.")
            return

        best_route, best_distance = genetic_algorithm(self.locations, generations=1500)
        self.output.insert(tk.END, "\nRute terbaik:\n")
        route_coords = []
        for idx in best_route:
            coord = self.locations[idx]
            route_coords.append(coord)
            self.output.insert(tk.END, f"{coord}\n")
        self.output.insert(tk.END, f"\nJarak Perkiraan: {best_distance:.6f} km\n")

        # Membuat link Google Maps
        google_maps_url = "https://www.google.com/maps/dir/" + "/".join(f"{lat},{lon}" for lat, lon in
route_coords)
        self.output.insert(tk.END, f"\nLihat rute di Google Maps: {google_maps_url}\n")
        webbrowser.open(google_maps_url)

    def refresh_output(self):
        self.output.delete(1.0, tk.END)
        for i, loc in enumerate(self.locations):
            self.output.insert(tk.END, f"{i}: {loc}\n")

    def on_output_click(self, event):
        try:
            index = int(self.output.get(f"@{event.x},{event.y} linestart", f"@{event.x},{event.y}
lineend").split(":")[0])
```

(2025), 3 (3): 362–376 **Scientica** 3021-8209
Jurnal Ilmiah Sain dan Teknologi

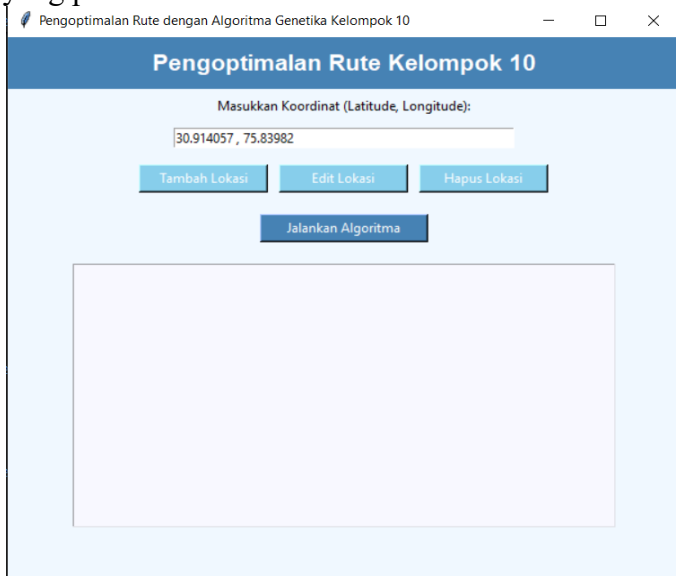
```
            self.selected_index = index
            self.entry.delete(0, tk.END)
            self.entry.insert(0, f"{self.locations[index][0]},{self.locations[index][1]}")
        except (ValueError, IndexError):
            pass

if __name__ == "__main__":
    root = tk.Tk()
    app = GeneticAlgorithmApp(root)
    root.mainloop()
```

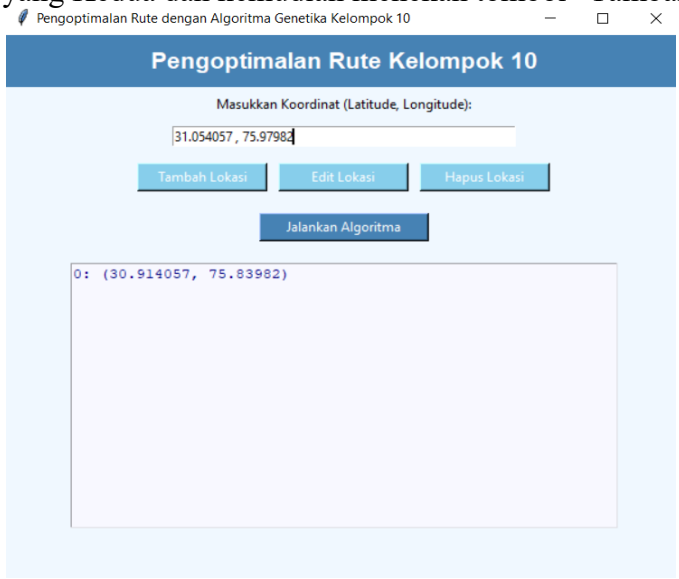
Link Aplikasi:
https://drive.google.com/drive/folders/17q6YgwfEguh7kCTt1QO8jT79ThdidxEt?usp=drive_link

***Output* Program:**

Masukan Koordinat yang pertama dan kemudian menekan tombol “Tambah Lokasi”

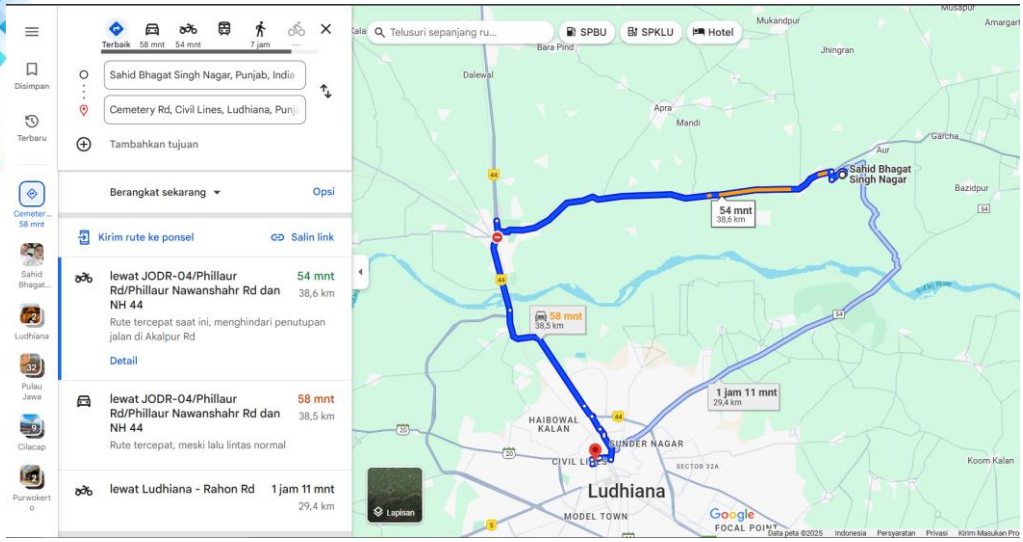


Masukan Koordinat yang Kedua dan kemudian menekan tombol “Tambah Lokasi”

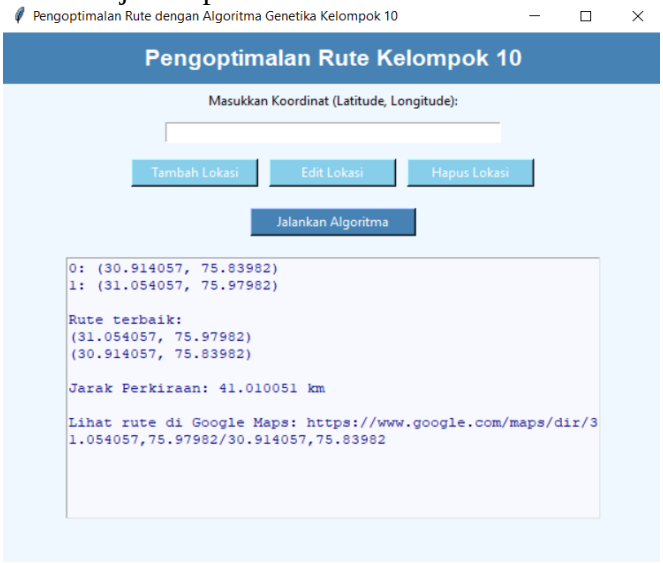


Lalu tekan “Jalankan Algoritma”, maka akan langsung diarahkan ke google maps

(2025), 3 (3): 362–376 **Scientica** 3021-8209
Jurnal Ilmiah Sain dan Teknologi

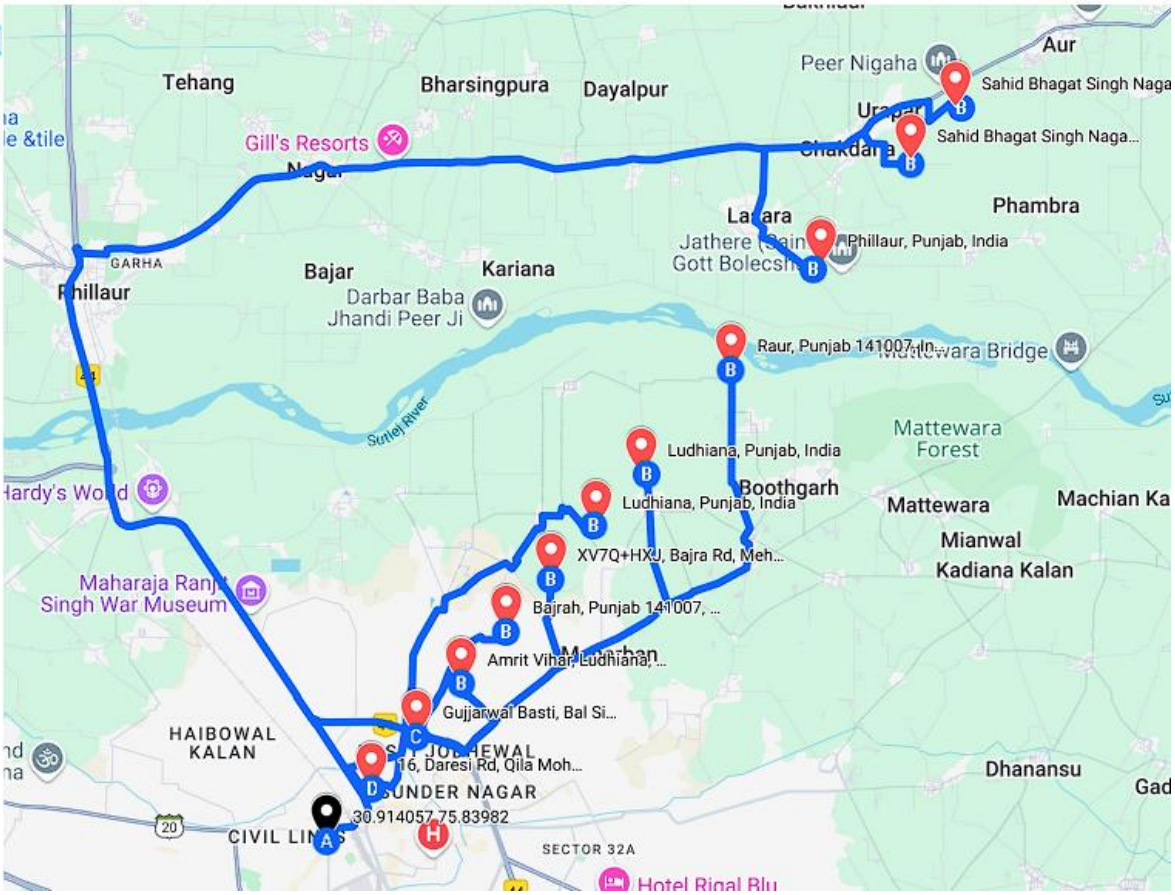


Dan *output* di dalam GUI menjadi seperti ini:



(2025), 3 (3): 362–376 **Scientica** 3021-8209
Jurnal Ilmiah Sain dan Teknologi

Rute Peta Pengiriman Dari Datasheet Di Terapkan Dalam Maps



Studi Perbandingan

Dalam studi ini, *output* yang dihasilkan oleh algoritma genetika dibandingkan dengan dua metode lain, yaitu algoritma *greedy* dan algoritma *ant colony*. Berdasarkan hasil studi yang baru disebutkan di atas, algoritma genetika bahkan meningkatkan *output*nya, terutama saat masalah yang harus dipecah adalah jika setiap kota dapat jatuh pada posisi yang sangat berjauhan.

Analisis Sensitivitas

Analisis sensitivitas Untuk melihat seberapa sensitif algoritma genetik terhadap perubahan parameter, maka dilakukan analisis sensitivitas. Hasil dari analisis sensitivitas menunjukkan bahwa perubahan parameter akan mempengaruhi performa algoritma genetika, namun metode ini masih dapat menghasilkan solusi optimum.

Perbandingan dengan Metode Lain

Kinerja algoritma genetika harus dibandingkan dengan metode optimasi lain seperti algoritma *greedy* dan algoritma *ant colony*. Dari perbandingan, dapat disimpulkan bahwa algoritma genetika dapat menemukan solusi optimal untuk TSP dalam konteks pengiriman makanan.

Kesimpulan

Berbagai penelitian telah membuktikan bahwa algoritma genetika dengan antarmuka dapat membantu layanan makanan dalam menyelesaikan masalah paling optimal. Sistem ini berkaitan dengan penurunan waktu dan biaya pengiriman makanan. Daripada hasil literatur saya, dengan algoritma ini, layanan pengiriman makanan akan berjalan lebih efisien daripada

sebelumnya. Maka, berdasarkan dari hasil penelitian juga membuktikan bahwa karena algoritma genetika dapat diimplementasikan dengan mudah untuk semua jenis dan jarak pengiriman.

Saran untuk Penelitian Selanjutnya

- **Pengembangan Algoritma Genetika:** pada pengembangan algoritma genetika, algoritma genetika dapat dikembangkan lebih lanjut dengan mempertimbangkan waktu pengiriman, kapasitas kendaraan, dan preferensi pelanggan.
- **Integrasi dengan Sistem Pemesanan Makanan:** menambah fitur aplikasi, integrasi pula dengan sistem pembayaran, pelacakan pengiriman makanan, dan sistem *rating*.
- **Pengembangan Fitur Aplikasi:** Fitur yang bisa dikembangkan misalnya integrasi dengan sistem pembayaran, pelacakan pengiriman, sistem *rating* dan sebagainya.
- **Pengembangan Model Simulasi:** pengembangan model simulasi yang lebih realistis untuk memilih performa algoritma genetika dengan berbagai skenario
- Integrasi algoritma genetika dengan sistem pemesanan lagi makanan secara langsung.
- Pengembangan model simulasi yang lebih realistis dengan sudah menambahkan faktor waktu pengiriman dan kapasitas kendaraan.
- Fitur pelacakan dan notifikasi waktu pengiriman
- Menerapkan algoritma hibrida yang menggabungkan algoritma genetika dengan metode optimasi lainnya seperti algoritma *tabu search* dan *simulated annealing* untuk meningkatkan kinerja lebih lanjut.

Referensi

Morib, B., Nugroho, D., & Susyanto, T. (n.d.). *Penentuan Rute Berbasis Algoritma Genetika (Studi Kasus: Angkutan Wisata Surakarta)*. https://p3m.sinus.ac.id/jurnal/index.php/e-jurnal_SINUS/article/viewFile/116/pdf_19

Cahya, G., Bagus, P. W., & Rosita, Y. D. (n.d.). *Penentuan Rute Optimal untuk Jasa Pengiriman Barang Menggunakan Algoritma Genetika*. https://journal.sekawan-org.id/index.php/jtim/article/download/322/213

Fatikawati, I., Syaripuddin, & Huda, M. N. (n.d.). *Implementasi Algoritma Genetika dalam Menentukan Rute Terpendek Pendistribusian Barang PT. J&T Samarinda*. https://jurnal.fmipa.unmul.ac.id/index.php/Basis/article/view/1071

Mubarok, A. Y., & Chotijah, U. (n.d.). *Penerapan Algoritma Genetika untuk Mencari Optimasi Kombinasi Jalur Terpendek dalam Kasus Travelling Salesman Problem*. https://journal.nurulfikri.ac.id/index.php/jtt/article/view/424

Ihsani, I., Pramuntadi, A., Gutama, D. H., & Wijaya, D. P. (n.d.). *Implementasi Algoritma Genetika dalam Penentuan Rute Optimal untuk Kurir Kantor Pos Berbasis Web (Studi Kasus: Kantor Pos)*. https://ejournal.almaata.ac.id/index.php/IJUBI/article/view/2662/1740

Tohari, A., & Astuti, Y. P. (n.d.). *Penerapan Algoritma Genetika dalam Menentukan Rute Terpendek PT. Pos Cabang Lamongan*. Retrieved from https://ejournal.unesa.ac.id/index.php/mathunesa/article/view/56485

Gaurav Malik. *Food Delivery Dataset (train.csv)* https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset/data?select=train.csv