OPTIMASI RUTE DISTRIBUSI MULTI-DEPOT VEHICLE ROUTING PROBLEM DENGAN PERBANDINGAN ALGORITMA GENETIKA DAN SIMULATED ANNEALING

Muhammad Zildan Ali Firmansyah, Syahri Romadhona, Rizqon Mukhlasin A. ,
Putra Aditya Purwanto, Samsul Amar , Trisita Novianti
Jurusan Teknik Industri, Fakultas Teknik, Universitas Trunojoyo Madura
Jl. Raya Telang, Perumahan Telang Indah, Telang, Kec. Kamal, Kabupaten Bangkalan, Jawa Timur 69162
220421100034@student.trunojoyo.ac.id

ABSTRAK

Perkembangan pesat perdagangan elektronik dan sistem logistik menuntut efisiensi tinggi dalam perencanaan rute distribusi. Salah satu permasalahan utama dalam sistem logistik adalah penentuan rute kendaraan yang optimal dengan keterbatasan jumlah kendaraan dan kapasitas angkut yang dikenal sebagai *Capacitated Vehicle Routing Problem* (CVRP). Penelitian ini bertujuan untuk membandingkan kinerja Algoritma Genetika (GA) dan *Simulated Annealing* (SA) dalam menyelesaikan permasalahan CVRP. Metode penelitian dilakukan dengan membangun dataset sintetis yang terdiri dari 25 pelanggan, 1 depot, dan 3 kendaraan dengan batasan kapasitas tertentu. Kedua algoritma diimplementasikan menggunakan bahasa pemrograman *Python* dan diuji dengan jumlah iterasi yang sama. Hasil penelitian menunjukkan bahwa Algoritma Genetika menghasilkan total jarak tempuh sebesar 7170,87, sedangkan *Simulated Annealing* menghasilkan total jarak tempuh sebesar 7188,00. Meskipun *Simulated Annealing* mampu mencapai solusi awal yang baik dengan lebih cepat, Algoritma Genetika menunjukkan kualitas solusi akhir yang lebih optimal. Hasil ini menunjukkan bahwa pemilihan algoritma optimasi rute perlu disesuaikan dengan kebutuhan untuk berfokus pada kualitas solusi atau kecepatan pencarian solusi awal.

Kata kunci : *Vehicle Routing Problem (VRP), Multi-Depot VRP (MDVRP), Optimasi Rute, Algoritma Genetika (GA), Simulated Annealing (SA).*

1. PENDAHULUAN

Perkembangan teknologi dan perdagangan elektronik (*e-commerce*) yang semakin pesat mendorong perusahaan untuk meningkatkan efisiensi dalam sistem distribusi dan manajemen rantai pasok. Salah satu faktor kunci dalam sistem logistik adalah penentuan rute distribusi yang efektif dan efisien, karena rute distribusi berpengaruh langsung terhadap biaya operasional, konsumsi bahan bakar, serta ketepatan waktu pengiriman kepada pelanggan. Oleh karena itu, perencanaan rute kendaraan menjadi aspek penting dalam mendukung keberlangsungan operasional perusahaan.

Permasalahan penentuan rute distribusi menjadi semakin kompleks ketika melibatkan banyak pelanggan dengan keterbatasan jumlah kendaraan dan kapasitas angkut. Dalam penelitian operasi, permasalahan ini dikenal sebagai *Capacitated Vehicle Routing Problem* (CVRP). CVRP termasuk ke dalam kategori permasalahan optimasi kombinatorial yang bersifat NP-*hard*, sehingga sulit diselesaikan secara optimal menggunakan metode matematis konvensional, terutama ketika skala permasalahan semakin besar.

Keterbatasan tersebut mendorong penggunaan algoritma metaheuristik sebagai pendekatan alternatif dalam menyelesaikan CVRP. Dua algoritma metaheuristik yang sering digunakan adalah Algoritma Genetika (GA) dan *Simulated Annealing* (SA). Algoritma Genetika bekerja dengan pendekatan berbasis populasi yang meniru mekanisme evolusi biologis, sedangkan *Simulated Annealing* merupakan metode pencarian lokal yang terinspirasi dari proses pendinginan logam dalam fisika.

Permasalahan penelitian dalam studi ini adalah belum diketahuinya algoritma metaheuristik yang paling efektif dalam menghasilkan solusi rute distribusi yang optimal pada permasalahan CVRP, khususnya jika dibandingkan dari sisi kualitas solusi akhir dan kecepatan pencarian solusi.

Tujuan penelitian ini adalah membandingkan kinerja Algoritma Genetika dan *Simulated Annealing* dalam menyelesaikan permasalahan CVRP berdasarkan total jarak tempuh yang dihasilkan dan karakteristik konvergensi solusi.

Rencana penyelesaian masalah dilakukan dengan membangun model CVRP menggunakan dataset sintetis, mengimplementasikan Algoritma Genetika dan *Simulated Annealing* berbasis Python, serta melakukan analisis kuantitatif dan visual terhadap hasil yang diperoleh.

2. TINJAUAN PUSTAKA

2.1. Manajemen Distribusi dan Logistik

Dalam era perdagangan elektronik yang berkembang pesat, manajemen distribusi memegang peranan krusial dalam keberlangsungan operasional perusahaan. Efisiensi dalam penyaluran barang tidak hanya berdampak pada struktur biaya operasional, tetapi juga menjadi penentu utama kepuasan pelanggan. Menurut Wijaya, optimalisasi rute dan biaya distribusi merupakan faktor kunci dalam memastikan ketepatan waktu pengiriman serta efisiensi konsumsi bahan bakar. Kompleksitas distribusi ini semakin meningkat ketika melibatkan

penyaluran bantuan atau produk dari berbagai titik sumber ke banyak titik tujuan, yang menuntut strategi perencanaan yang matang[3].

2.2. *Vehicle Routing Problem* (VRP) dan *Capacitated Vehicle Routing Problem* (CVRP)

Vehicle Routing Problem (VRP) merupakan permasalahan optimasi dalam bidang logistik yang bertujuan untuk menentukan rute kendaraan dari depot ke sejumlah pelanggan dengan biaya atau jarak tempuh minimum. VRP menjadi komponen penting dalam sistem distribusi karena penentuan rute yang tidak efisien dapat meningkatkan biaya operasional, konsumsi bahan bakar, serta menurunkan ketepatan waktu pengiriman yang berdampak pada kepuasan pelanggan akhir[3].

Salah satu pengembangan dari VRP adalah *Capacitated Vehicle Routing Problem* (CVRP), yaitu permasalahan VRP yang mempertimbangkan batasan kapasitas kendaraan. Dalam CVRP, setiap kendaraan memiliki kapasitas maksimum yang tidak boleh dilampaui saat melayani permintaan pelanggan. CVRP termasuk dalam kategori permasalahan optimasi kombinatorial yang bersifat NP-*hard*, sehingga sulit diselesaikan secara optimal menggunakan pendekatan matematis konvensional, terutama pada skala permasalahan yang besar[1].

2.3. Multi-Depot *Vehicle Routing Problem* (MDVRP)

Multi-Depot *Vehicle Routing Problem* (MDVRP) merupakan pengembangan lebih lanjut dari CVRP, di mana distribusi tidak hanya berasal dari satu depot, tetapi dari beberapa depot yang berbeda. Pada MDVRP, selain menentukan urutan rute kendaraan, sistem juga harus menentukan depot yang paling sesuai untuk melayani setiap pelanggan. Kompleksitas permasalahan MDVRP jauh lebih tinggi dibandingkan CVRP karena adanya keputusan tambahan terkait pengelompokan pelanggan ke depot tertentu[4].

MDVRP merepresentasikan kondisi distribusi di dunia nyata secara lebih akurat, khususnya pada perusahaan dengan jaringan gudang yang tersebar. Oleh karena itu, MDVRP sering digunakan sebagai studi lanjutan dari CVRP dalam penelitian optimasi rute distribusi[2].

2.4. Algoritma Genetika

Algoritma Genetika (*Genetic Algorithm*/GA) merupakan algoritma metaheuristik yang terinspirasi dari teori evolusi Darwin mengenai *survival of the fittest*. GA bekerja dengan sekumpulan solusi yang disebut populasi, di mana setiap solusi direpresentasikan sebagai kromosom. Proses pencarian solusi dilakukan melalui mekanisme seleksi, *crossover*, dan mutasi untuk menghasilkan solusi yang semakin baik pada setiap generasi[1].

Dalam konteks permasalahan VRP dan CVRP, Algoritma Genetika banyak digunakan karena kemampuannya dalam melakukan eksplorasi ruang solusi secara luas dan menghindari jebakan solusi optimum lokal. Setiap kromosom umumnya direpresentasikan sebagai urutan kunjungan pelanggan, sehingga GA mampu menghasilkan variasi rute yang beragam dan efisien[7].

2.5. Simulated Annealing

Simulated Annealing (SA) merupakan algoritma optimasi berbasis pencarian lokal yang terinspirasi dari proses *annealing* pada material logam. Algoritma ini bekerja dengan satu solusi pada satu waktu dan memungkinkan penerimaan solusi yang lebih buruk berdasarkan probabilitas tertentu, khususnya pada temperatur tinggi, sehingga mampu keluar dari jebakan solusi optimum lokal[6].

SA dikenal memiliki keunggulan dalam kesederhanaan implementasi dan kecepatan dalam menemukan solusi awal yang cukup baik. Meskipun cakupan eksplorasi ruang solusinya tidak seluas Algoritma Genetika, SA tetap efektif digunakan untuk permasalahan optimasi rute dengan kompleksitas sedang[10].

2.6. Simulated Annealing

Simulated Annealing (SA) merupakan algoritma optimasi berbasis pencarian lokal yang terinspirasi dari proses *annealing* pada material logam. Algoritma ini bekerja dengan satu solusi pada satu waktu dan memungkinkan penerimaan solusi yang lebih buruk berdasarkan probabilitas tertentu, khususnya pada temperatur tinggi, sehingga mampu keluar dari jebakan solusi optimum lokal[6].

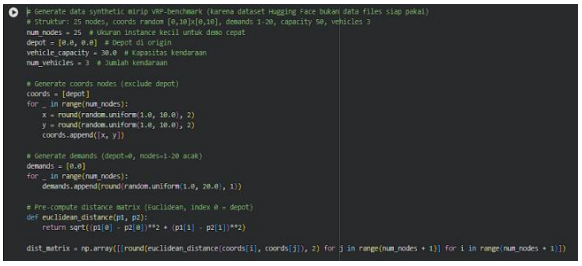
SA dikenal memiliki keunggulan dalam kesederhanaan implementasi dan kecepatan dalam menemukan solusi awal yang cukup baik. Meskipun cakupan eksplorasi ruang solusinya tidak seluas Algoritma Genetika, SA tetap efektif digunakan untuk permasalahan optimasi rute dengan kompleksitas sedang[10].

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimen komputasi untuk membandingkan kinerja Algoritma Genetika dan *Simulated Annealing* dalam menyelesaikan CVRP.

3.1. Dataset dan Parameter

Dataset yang digunakan bersifat sintetis dan terdiri dari 25 pelanggan, 1 depot, dan 3 kendaraan dengan kapasitas tertentu. Koordinat pelanggan dihasilkan secara acak dan jarak antar titik dihitung menggunakan jarak Euclidean.



```
# Generate data synthetic mirip VRP-benchmark (karena dataset Hugging Face bukan data files siap pakai)
# Struktur: 25 nodes, coords random [0,10][0,10], demands 1-20, capacity 50, vehicles 3
num_nodes = 25  # Ukuran instance kecil untuk demo cepat
depot = [0.0, 0.0]  # Depot di origin
vehicle_capacity = 30.0  # Kapasitas kendaraan
num_vehicles = 3  # Jumlah kendaraan

# Generate coords nodes (exclude depot)
coords = [depot]
for _ in range(num_nodes):
    x = round(random.uniform(1.0, 10.0), 2)
    y = round(random.uniform(1.0, 10.0), 2)
    coords.append([x, y])

# Generate demands (depot=0, nodes=1-20 acak)
demands = [0.0]
for _ in range(num_nodes):
    demands.append(round(random.uniform(1.0, 20.0), 1))

# Pre-compute distance matrix (Euclidean, index 0 = depot)
def euclidean_distance(p1, p2):
    return sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

dist_matrix = np.array([[round(euclidean_distance(coords[i], coords[j]), 2) for j in range(num_nodes + 1)] for i in range(num_nodes + 1)])
```

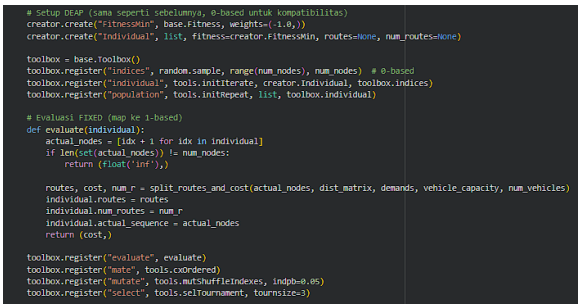
Gambar 1. Program G*enerate* Dataset

Karena dataset VRP tidak diambil dari sumber eksternal, program membuat dataset secara sintetis untuk keperluan pengujian. Data yang dihasilkan meliputi:

a. Jumlah node pelanggan ditetapkan sebanyak 25 titik.
b. Letak depot ditempatkan pada koordinat (0, 0).
c. Kapasitas kendaraan ditetapkan sebesar 30 satuan.
d. Jumlah kendaraan sebanyak 3 unit.
e. Koordinat pelanggan dihasilkan secara acak dalam domain 1–10 agar menyerupai persebaran lokasi geografis nyata.
f. *Demand* pelanggan dihasilkan acak antara 1–20 satuan.
g. *Distance Matrix* dihitung menggunakan jarak *Euclidean* antara seluruh pasangan titik.
h. Pembangkitan dataset ini membuat program dapat dijalankan tanpa ketergantungan terhadap file eksternal dan sesuai dengan skala eksperimen yang diinginkan.

3.2. Implementasi Algoritma Genetika

Algoritma Genetika diimplementasikan menggunakan library DEAP pada Python. Setiap individu direpresentasikan sebagai permutasi pelanggan. Proses GA meliputi inisialisasi populasi, evaluasi fitness berdasarkan total jarak tempuh, seleksi menggunakan tournament selection, crossover menggunakan Ordered Crossover (OX), serta mutasi untuk menjaga keberagaman solusi.



```
# Setup DEAP (sama seperti sebelumnya, 0-based untuk kompatibilitas)
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin, routes=None, num_routes=None)

toolbox = base.Toolbox()
toolbox.register("indices", random.sample, range(num_nodes), num_nodes)  # 0-based
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.indices)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Evaluasi FIXED (map ke 1-based)
def evaluate(individual):
    actual_nodes = [idx + 1 for idx in individual]
    if len(set(actual_nodes)) != num_nodes:
        return (float('inf'),)

    routes, cost, num_r = split_routes_and_cost(actual_nodes, dist_matrix, demands, vehicle_capacity, num_vehicles)
    individual.routes = routes
    individual.num_routes = num_r
    individual.actual_sequence = actual_nodes
    return (cost,)

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxOrdered)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
```

Gambar 2. Program Algoritma Genetika

Algoritma genetika digunakan untuk melakukan pencarian solusi. Program menggunakan library DEAP untuk memudahkan implementasi GA. Setiap individu dalam populasi diwakilkan sebagai tiap mutasi dari nomor-nomor pelanggan.

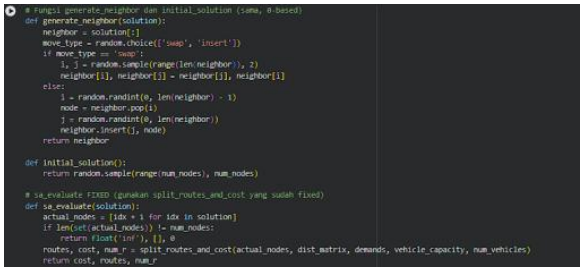
Proses GA terdiri dari beberapa langkah berikut:

a. Inisialisasi Populasi
   Populasi awal dibentuk dari sejumlah permutasi acak node pelanggan.
b. Evaluasi Fitness
   Fungsi evaluate() memanggil split_routes_and_cost() untuk menghitung:
   - Rute yang terbentuk,
   - *Total cost* (total jarak + penalti)
   - Jumlah rute.

   Individu dengan *cost* yang lebih rendah dianggap lebih baik.
c. Seleksi
   Algoritma menggunakan *tournament selection* untuk memilih individu yang kuat untuk dikawinkan.
d. *Crossover*
   Operator Ordered Crossover (OX) digunakan agar struktur tetap jelas dan tidak terjadi duplikasi node.
e. Mutasi
   Operator *mutShuffleIndexes* digunakan untuk mengacak sebagian kecil posisi dalam permutasi, menjaga keberagaman populasi dan mencegah *konvergensi* dini.
f. Pembentukan Generasi Baru
   Setiap generasi menghasilkan populasi baru berdasarkan aturan evolusi. Proses berlangsung hingga mencapai jumlah generasi tertentu atau ketika populasi tidak lagi menunjukkan peningkatan signifikan.
   Program mencatat nilai *fitness* minimum, rata-rata, dan maksimum per generasi sehingga performa GA dapat dievaluasi secara statistik.

3.3. Implementasi Simulated Annealing

Simulated Annealing diimplementasikan dengan memulai dari satu solusi awal acak. Solusi tetangga dihasilkan melalui operasi swap dan insert. Aturan penerimaan solusi didasarkan pada perbedaan biaya dan temperatur yang diturunkan secara bertahap menggunakan cooling schedule.



```
# Fungsi generate_neighbor dan initial_solution (sama, 0-based)
def generate_neighbor(solution):
    neighbor = solution[:]
    move_type = random.choice(['swap', 'insert'])
    if move_type == 'swap':
        i, j = random.sample(range(len(neighbor)), 2)
        neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
    else:
        i = random.randint(0, len(neighbor) - 1)
        node = neighbor.pop(i)
        j = random.randint(0, len(neighbor))
        neighbor.insert(j, node)
    return neighbor

def initial_solution():
    return random.sample(range(num_nodes), num_nodes)

# sa_evaluate FIXED (gunakan split_routes_and_cost yang sudah fixed)
def sa_evaluate(solution):
    actual_nodes = [idx + 1 for idx in solution]
    if len(set(actual_nodes)) != num_nodes:
        return float('inf'), [], 0
    routes, cost, num_r = split_routes_and_cost(actual_nodes, dist_matrix, demands, vehicle_capacity, num_vehicles)
    return cost, routes, num_r
```

Gambar 3. Program *Simulated Annealing*

Simulated Annealing merupakan metaheuristik berbasis prinsip termodinamika—lebih spesifiknya proses pendinginan logam. SA dimulai dengan satu solusi awal acak kemudian melakukan eksplorasi ruang solusi melalui modifikasi kecil pada solusi tersebut.

Tahapan utama implementasi SA adalah sebagai berikut:

a. Solusi Awal
Solusi acak dihasilkan sebagai permutasi dari seluruh node pelanggan.

```
# Algoritma SA FIXED (tambah try-except, ensure best_routes selalu set)
def simulated_annealing(initial_temp=1000.0, cooling_rate=0.995, min_temp=1.0, max_iter=10000):
    try:
        current_solution = initial_solution()
        current_cost, current_routes, current_num_r = sa_evaluate(current_solution)
        best_solution = current_solution[:]
        best_cost = current_cost
        best_routes = current_routes[:] if current_routes else []  # Copy list
        best_num_r = current_num_r
        temp = initial_temp

        costs = [current_cost]
        no_improve = 0

        for iter in range(max_iter):
            try:
                neighbor = generate_neighbor(current_solution)
                neighbor_cost, neighbor_routes, neighbor_num_r = sa_evaluate(neighbor)
                delta = neighbor_cost - current_cost

                if delta < 0 or random.random() < np.exp(-delta / temp):
                    current_solution = neighbor
                    current_cost = neighbor_cost
                    current_routes = neighbor_routes[:]
                    current_num_r = neighbor_num_r
                    no_improve = 0
                else:
                    no_improve += 1
```

Gambar 4. *Generate* Solusi Awal

b. Pembangkit Tetangga (*Neighborhood*)
Fungsi generate_neighbor() menghasilkan solusi tetangga menggunakan dua jenis pergerakan:
Swap: Menukar dua node.
Insert: Memindahkan satu node ke posisi berbeda.
Kedua operator ini menjaga solusi tetap valid dalam bentuk permutasi.

```
                # Update best (salin routes)
                if current_cost < best_cost:
                    best_solution = current_solution[:]
                    best_cost = current_cost
                    best_routes = current_routes[:]
                    best_num_r = current_num_r

                costs.append(current_cost)
                temp *= cooling_rate

                if temp < min_temp or no_improve > 50:
                    break
            except Exception as e:
                print(f"Warning: Error di iterasi {iter}: {e}. Skip iterasi.")
                continue  # Skip, lanjut iterasi

        # Map ke actual sequence
        best_actual_seq_sa = [idx + 1 for idx in best_solution]

        print("Debug SA: Return values - Cost:", best_cost, "Routes len:", len(best_routes), "Num R:", best_num_r)
        return best_actual_seq_sa, best_cost, best_num_r, best_routes, costs

    except Exception as e:
        print(f"Error fatal di SA: {e}. Return fallback.")
        return [], float('inf'), 0, [], []

# Jalankan SA
print("Menjalankan Simulated Annealing (Max 300 Iterasi, Fixed)...")
best_solution_sa, best_cost_sa, best_num_routes_sa, best_routes_sa, cost_history = simulated_annealing()
```

Gambar 5. Mencari Solusi Tetangga

c. Evaluasi *Cost*
Solusi tetangga dihitung biaya rutenya menggunakan sa_evaluate() yang juga memanggil fungsi split_routes_and_cost().

d. Aturan Penerimaan Solusi
Solusi lebih baik (*cost* lebih rendah) selalu diterima.
Solusi lebih buruk dapat diterima dengan probabilitas tertentu:

$$p = e^{-\triangle E/T}$$

Probabilitas ini memungkinkan SA keluar dari local optimum.

e. *Cooling Schedule*
Suhu diperbarui menggunakan formula:

$$T = T \times a$$

dengan α sebagai *cooling rate*.
Saat suhu semakin rendah, peluang menerima solusi buruk semakin kecil.

f. Penyimpanan Solusi Terbaik
Sepanjang proses, SA merekam solusi terbaik yang pernah ditemukan.
Proses berhenti ketika suhu mencapai nilai minimum atau jumlah iterasi maksimum tercapai.

4. HASIL DAN PEMBAHASAN

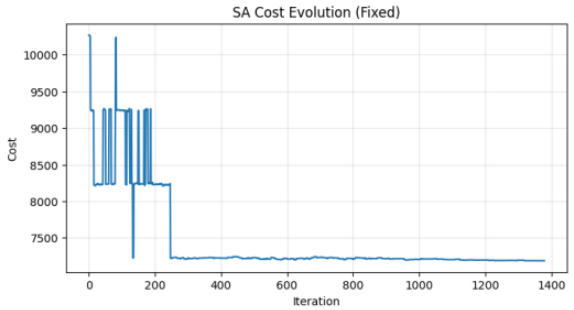
Hasil dan pembahasan meliputi analisis kualitatif, analisis konvergensi, interpretasi kinerja, implikasi praktis, dan MDVRP.

4.1. Hasil

Output percobaan menghasilkan angka yang dapat digunakan sebagai perbandingan kedua metode. Hasil secara jelas menunjukkan keunggulan kinerja Algoritma Genetika (GA) dalam hal mencari kualitas solusi akhir. GA terbukti mampu mencapai total jarak tempuh 7170,87. *Simulated Annealing* (SA) menghasilkan solusi dengan total jarak tempuh 7188,00. Hasil observasi menunjukkan yang dimana dari analisis ini menunjukkan bahwa kedua algoritma dapat mengidentifikasi solusi yang sepenuhnya layak digunakan. Keduanya tercatat menggunakan tepat 3 rute dan sesuai dengan batasan 3 kendaraan yang tersedia. Poin utamanya adalah tidak ada penalti yang diterapkan pada solusi akhir tersebut, sehingga perbandingan biaya yang dihasilkan mencerminkan efisiensi jarak tempuh superior yang dicapai oleh pendekatan GA dengan perbandingan yang sangat sedikit tapi penting.

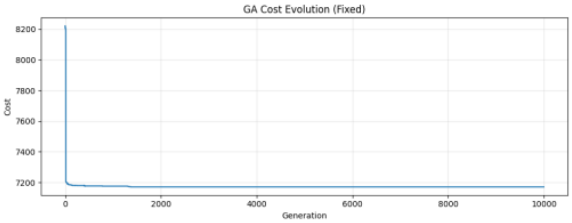
4.2. Analisis Konvergensi

Hasil konvergensi menunjukkan kedua algoritma melakukan pencarian solusi. SA mengalami penurunan biaya yang sangat cepat pada iterasi awal, yang dimana mencerminkan kemampuan pencarian lokal yang kuat dan cepat menemukan solusi yang cukup baik. Namun setelah beberapa pengulangan kinerjanya menjadi lemah dan tidak maksimal, menandakan bahwa SA mulai kesulitan keluar dari jebakan optimum lokal.



Gambar 6. Konvergen SA

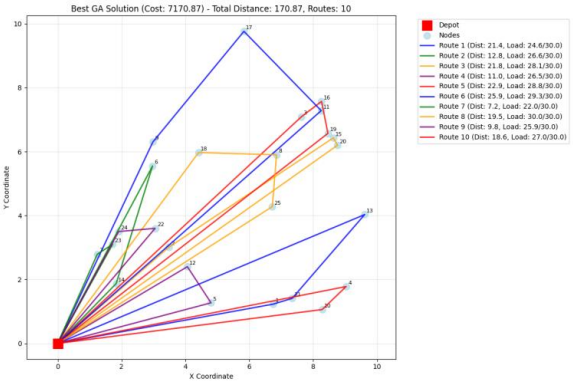
GA menunjukkan penurunan yang lebih bertahap namun konsisten. Kemampuan eksplorasi berbasis populasi memungkinkannya untuk terus menemukan perbaikan kecil, yang pada akhirnya melampaui kualitas solusi SA.



Gambar 7. Konvergen GA

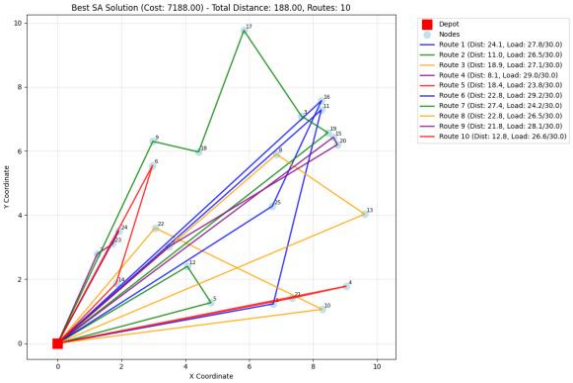
4.3. Analisis Kualitatif (Visual)

Visualisasi rute (GA vs SA) memperlihatkan perbedaan yang jelas. Solusi GA menunjukkan rute sangat teratur dan rapi, dengan minim persilangan antar rute (*route crossing*), yang secara visual menunjukkan tingkat efisiensi tinggi.



Gambar 8. Solusi GA

Solusi SA, meskipun layak, menunjukkan beberapa persilangan rute yang tidak efisien,mengindikasikan adanya jarak tempuh yang terbuang atau lebih panjang.



Gambar 9. Solusi SA

4.4. Intrepertasi Kinerja

Kinerja Algoritma Genetika menawarkan solusi rute yang luas karena penggunaan populasi solusi yang beragam. Penggunaan 30 individu sekaligus algoritma ini mampu menjelajahi berbagai rute solusi tanpa mudah terperangkap di lingkungan sekitar. Ini berbeda dari *Simulated Annealing* yang hanya bekerja dengan satu solusi dalam satu waktu, sehingga cakupan pencarian rute yang dieksplorasi lebih terbatas.

Mekanisme silang di Algoritma Genetika memiliki peran penting dalam menciptakan kombinasi solusi baru dari dua rute terbaik. Proses ini memungkinkan pembentukan rute yang lebih efisien dengan menggabungkan bagian terbaik dari dua solusi induk menjadi satu solusi anak yang lebih optimal. Sebaliknya *Simulated Annealing* bergantung pada modifikasi bertahap terhadap satu solusi dan tidak memiliki kemampuan generatif seperti Algoritma Genetika. Hasil perbedaan ini menjelaskan mengapa Algoritma Genetika biasanya menghasilkan biaya total akhir yang lebih rendah dibanding *Simulated Annealing* dalam jumlah iterasi yang sama.

4.5. Implikasi Praktis

Hasil penelitian ini memiliki keterlibatan yang penting dalam konteks penerapan algoritma optimasi untuk sistem distribusi. Pemilihan algoritma tidak hanya bergantung pada kualitas hasil, tetapi juga pada kecepatan komputasi yang dibutuhkan. Algoritma Genetika (GA) layak dipilih ketika tujuan utama adalah memperoleh rute dengan biaya paling minimal, terutama untuk perencanaan distribusi dengan pola yang stabil dan berulang. Dalam kasus seperti ini, efisiensi biaya jangka panjang lebih bernilai dibandingkan waktu pemrosesan yang sedikit lebih lama.

Sebaliknya, *Simulated Annealing* (SA) menjadi alternatif yang tepat untuk skenario distribusi yang bersifat fleksibel, di mana kecepatan pengambilan keputusan lebih diprioritaskan. Walaupun hasilnya sedikit kurang optimal (sekitar 5-6% lebih tinggi dibanding GA), waktu komputasi yang lebih singkat dapat memberikan keuntungan operasional dalam sistem logistik langsung. Dengan demikian, kedua algoritma ini memiliki peran masing-masing tergantung pada kebutuhan dan karakteristik sistem distribusi yang diterapkan.

4.6. MDVRP

Hasil dari 25 eksperimen ini termasuk dalam kategori model *Capacitated Vehicle Routing Problem* (CVRP) berbasis depot tunggal. Namun, tantangan logistik dunia nyata sering melibatkan skenario yang lebih rumit, yakni Multi-Depot *Vehicle Routing Problem* (MDVRP). Perubahan dari CVRP ke MDVRP bukan hanya peningkatan skala, tetapi ini termasuk perluasan dari permasalahan MDVRP yang membuat hal ini dikategorikan sebagai masalah komputasi yang sulit. Permasalahan tambahan ini muncul dalam bentuk masalah pengelompokan, Jika CVRP dengan depot tunggal hanya bertanya "bagaimana urutan rute terbaik?", MDVRP menimbulkan pertanyaan awal yang penting, yaitu "Depot mana yang harus melayani pelanggan mana?". Kebutuhan untuk mengatur pelanggan secara optimal ke depot-depot yang ada merupakan tantangan ekstra yang harus diatasi selain optimasi rute itu sendiri. Oleh karena itu, penelitian lanjutan yang direkomendasikan untuk menangani MDVRP harus mampu

mengevaluasi algoritma tidak hanya pada kemampuannya merutekan, tetapi juga pada kemampuannya menyelesaikan masalah mengatur depot secara efisien.

5. KESIMPULAN DAN SARAN

Hasil Penelitian ini membandingkan kinerja Algoritma Genetika dan *Simulated Annealing* dalam penyelesaian *Capacitated Vehicle Routing Problem* (CVRP) dengan 25 titik pelanggan. Hasil pengujian menunjukkan bahwa metode Algoritma Genetika (GA) menghasilkan total jarak 7170,87, sementara itu hasil *Simulated Annealing* menghasilkan total jarak 7188,00. Hasil metode Algoritma Genetika menunjukan bahwa mekanisme berbasis populasi pada Algoritma Genetika mampu mencari solusi dengan lebih efektif, sedangkan pada metode *Simulated Annealing* menunjukkan kelebihan pada kecepatan penyatuan di tahap awal pencarian.

Saran penelitian lebih lanjut merekomendasikan untuk memperluas studi pada model *Multi-Depot* VRP (MDVRP) dengan kerumitan rute lebih tinggi serta mengeksplorasi kemungkinan penggabungan kedua pendekatan tersebut menjadi algoritma gabungan. Pendekatan ini diharapkan dapat mengombinasikan. kemampuan eksplorasi global GA dengan kekuatan eksploitasi lokal SA untuk memperoleh hasil optimasi yang lebih seimbang dan efisien.

Berdasarkan temuan penelitian ini, penelitian ke depan disarankan untuk tidak hanya berfokus pada CVRP, tetapi juga mengkaji permasalahan Multi-Depot *Vehicle Routing Problem* (MDVRP) yang memiliki tingkat kompleksitas lebih tinggi dan lebih mencerminkan kondisi distribusi di lapangan. Selain itu, studi lanjutan dapat diarahkan pada pengembangan pendekatan algoritma gabungan yang memanfaatkan kemampuan eksplorasi global dari Algoritma Genetika serta kekuatan eksploitasi lokal dari *Simulated Annealing*, sehingga solusi yang dihasilkan diharapkan lebih optimal dan efisien baik dari segi kualitas rute maupun waktu komputasi.

DAFTAR PUSTAKA

[1] A. S. Kusumawardana, “VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS DENGAN METODE HIBRID SIMULATED ANNEALING – ALGORITMA GENETIKA,” pp. Halaman 1-3, 2013.

[2] Meilinda, “Penentuan Rute Terpendek untuk Multi Depot Vehicle Routing Problem Menggunakan Algoritma Ant Colony,” p. Halaman 1, 2025.

[3] V. P. Wijaya, “Analisis Optimalisasi Rute dan Biaya Distribusi Penyaluran Cadangan Bantuan Pangan Tahun 2024,” Jurnal Manajemen Pendidikan dan Ilmu Sosial, pp. Halaman 1-8, 2025.

[4] P. Stodola, “Multi-Depot Vehicle Routing Problem with Drones: Mathematical formulation, solution algorithm and experiments,” p. Halaman 1, 2024.

[5] H. S. A. Hibatulloh, “PENENTUAN RUTE OPTIMAL DISTRIBUSI AIR MINUM ISI ULANG DI GERAI AFSHEENA DENGAN MENGGUNAKAN METODE SAVING MATRIX DAN NEAREST INSERT,” Jurnal Ilmiah Research Student, pp. Halaman 1-9, 2025.

[6] I. H. A. G. d. S. A. u. O. N. M. A. Endurance, “Shafira Eka Aulia Putri,” Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,” pp. Halaman 1-3, 2021.

[7] N. Q. Saputra, “Analisa Optimalisasi Rute Distribusi Untuk Mengefisiensikan Logistik Menggunakan Algoritma Genetika,” Jurnal Manajemen dan Teknik Industri-Produksi , p. Halaman 3, 2024.

[8] A. K. Ariyani, “Hibridisasi Algoritme Genetika dan Simulated Annealing untuk Optimasi Multi-Trip Vehicle Routing Problem with Time Windows (Studi Kasus: Pariwisata Kabupaten Banyuwangi),” Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, pp. Halaman 1-7, 2017.

[9] N. W. A. F. Lestari, “OPTIMALISASI PENENTUAN JALUR DISTRIBUSI TERPENDEK DALAM PENGIRIMAN PRODUK CHEMICAL PEMBERSIH KOLAM RENANG MENGGUNAKAN METODE NEAREST NEIGHBOR,” Syntax Admiration, pp. Halaman 1-7, 2025.

[10] G. A. P. Agita, “Penerapan Sistem Optimasi Rute Pengiriman Barang dengan Pendekatan Saving Matrix dan Simulated Annealing,” pp. Halaman 1-9, 2025.