Jurnal Rekayasa Sistem Industri
Volume 14 No 1 - April 2025
http://journal.unpar.ac.id/index.php/jrsi/index
ISSN 2339-1499 (*online*) – ISSN 0216-1036 (*print*)

Disampaikan : 27 Agustus 2024
Direview : 24 September 2024
Diterima : **5 Februari 2025**

Usulan Rute Pendistribusian Gas LPG Menggunakan Algoritma Dijkstra dan Algoritma Genetika Pada Model CGVRP

Putri Oktaviani[1], Intan Permata Dalnis[2], Eri Wirdianto[3*]
[1,2,3)] Departemen Teknik Industri, Fakultas Teknik, Universitas Andalas
Jl. Limau Manis, Kecamatan Pauh, Padang, Kota Padang, Sumatera Barat
Email: putriiioktvn@gmail.com, intanpermatadalnis@gmail.com, e.wirdianto@gmail.com*

Abstract

Determining the route of distribution of goods for companies is an important aspect to consider. The distribution of 3 kg LPG gas at one of the companies in Bandung City does not yet have an optimal route. However, in previous research, a shorter route has been obtained with the Sweep algorithm, which is 118.85 km. The research continued using the Cluster Generalized Vehicle Routing Problem (CGVRP) model with the Dijkstra and Genetics algorithms. CGVRP is the determination of the shortest route using customer clusters. Dijkstra's algorithm is the determination of the route between 2 points from the starting point to the destination point. The result of the distance calculation with the Djkstraa algoritma is 114.779 km. Genetic Algorithm is the determination of the shortest route by reducing the attributes that are less dominant. The result of the distance calculating with Genetic Algorithm is 112 km. This shows a reduction in the total distribution mileage of 6.85 km compared to the completion of the Sweep algorithm in previous studies. The amount of this reduction is large enough to save the company's costs by 5.76%.

Keywords: CGVRP, Dijkstra, Genetika, Route, Sweep Algorithm

Abstrak

Penentuan rute pendistribusian barang bagi perusahaan merupakan aspek penting untuk diperhatikan. Pendistribusian gas LPG 3 kg pada salah satu perusahaan di Kota Bandung belum memiliki rute optimal. Namun pada penelitian terdahulu, telah diperoleh rute yang lebih pendek dengan algoritma *Sweep* yaitu sebesar 118,85 km. Penelitian dilanjutkan menggunakan model *Cluster Generalized Vehicle Routing Problem* (CGVRP) dengan algoritma Dijkstra dan Genetika. CGVRP merupakan penentuan rute terpendek menggunakan kluster pelanggan. Algoritma Dijkstra merupakan penentuan rute antara 2 titik dari titik awal ke titik tujuan. Hasil perhitungan jarak dengan algoritma Djkstraa sebesar 114,779 km. Algoritma Genetika merupakan penentuan rute terpendek dengan mereduksi atribut-atribut yang kurang dominan. Hasil perhitungan jarak dengan algoritma Genetika sebesar 112 km. Hal tersebut memperlihatkan adanya pengurangan total jarak tempuh pendistribusian sebesar 6,85 km dibandingkan penyelesaian algoritma *Sweep* pada penelitian sebelumnya. Jumlah pengurangan ini cukup besar sehingga dapat menghemat pengeluaran biaya perusahaan sebesar 5,76%.

Kata kunci: CGVRP, Dijkstra, Genetika, Rute, Algoritma *Sweep*

Pendahuluan

Distribusi barang merupakan aktivitas penting dalam industri manufaktur yang berupa penyaluran barang kepada konsumen (Mafaza & Muslim, 2023). Rute pengantaran barang sering kali menjadi permasalahan bagi perusahaan atau distributor (Baihaqi & Hermansyah, 2023) seperti permasalahan keterlambatan pengantaran, total biaya yang besar, dan penggunaan jarak (Hidayah et al., 2021). Rute yang digunakan dalam pengantaran akan dirancang sesuai dengan kebutuhan dari masing-masing *node* (Pratiwi, 2022). Masalah perancangan rute dapat diatasi dengan *Vehicle Routing Problem* (VRP) (Chandra & Setiawan, 2018). VRP merupakan suatu metode yang digunakan untuk menyelesaikan permasalahan pendistribusian barang dari depot kepada pelanggan (Moudya

et al., 2023). *Cluster Vehicle Routing Problem* (CVRP) bertujuan untuk mendapatkan rute pendistribusian yang optimal (Hanzani et al., 2023). Algoritma *Sweep* merupakan salah satu metode heuristik dalam menyelesaikan permasalahan CVRP. Kurnia dkk, pada tahun 2020 telah meneliti bahwa algoritma *Sweep* memberikan hasil perhitungan yang lebih rendah dibandingkan hasil dari algoritma Savings.

Cluster Generalized Vehicle Routing Problem (CGVRP) merupakan salah satu metode pengembangan dari model distribusi VRP (Hermanto, Adiasa, et al., 2020). Metode ini berkaitan dengan permasalahan pendistribusian barang menggunakan kelompok - kelompok pelanggan (Zubir et al., 2022). Tujuan dari penggunaan metode ini yaitu untuk mendapatkan identifikasi rute dengan biaya minimum dalam m kendaraan (Gautama & Hermanto, 2020). Metode CGVRP dimulai dari gudang dan berakhir di gudang (Hermanto & Ruskartina, 2018). Hal ini mengisyaratkan tepat satu kali dikunjungi setiap simpul dalam grafik. Jalur yang dibentuk oleh metode CGVRP dalam pendistribusian barang yaitu jalur Hamilton (Trisatya, 2023). Pendistribusian barang pada setiap simpul tidak akan melampaui kapasitas maksimum pada setiap kendaraan (Wang & Lu, 2019).

Algoritma Dijkstra dapat digunakan dalam penyelesaian metode CGVRP (Firmansyah & Hermanto, 2021). Algoritma Dijkstra merupakan algoritma penentuan rute antara 2 titik dari titik awal ke titik tujuan (Sutresno & Suni, 2023). Penyelesaian algoritma Dijkstra menggunakan prinsip Greedy (Awalloedin et al., 2022). Prinsip Greedy melakukan pemilihan sisi dengan bobot minimum pada setiap step langkah penyelesaian (Suryani & Murniyasih, 2022).

Selain menggunakan algoritma Dijkstra, masalah CGVRP juga dapat diselesaikan dengan algoritma Genetika. Algoritma genetika adalah bentuk metode optimasi metaheuristik dengan mekanisme pencarian yang didasarkan pada proses evolusi biologis yang terjadi pada makhluk hidup (Fatnita & Lukmandono, 2020). Algoritma Genetika memiliki kemampuan dalam menangani masalah optimasi yang kompleks dan berskala besar. Algoritma ini dapat mereduksi atribut-atribut yang kurang dominan (Busono, 2020). Proses perhitungan Algoritma Genetika mengalami proses seleksi dengan membangkitkan populasi awal yang kemudian akan dilakukan proses pindah silang (*Cross Over*) dan diakhiri dengan proses mutasi (Hermanto, Ruskartina, et al., 2020).

Salah satu perusahaan di Kota Bandung memiliki permasalahan dalam pendistribusian gas LPG 3 kg. Perusahaan ini memiliki 24 titik pelanggan dengan kapasitas angkut terbatas. Gas LPG 3 kg akan didistribusikan dengan jumlah yang berbeda-beda tiap pelanggan. Perusahaan terkendala menentukan rute terbaik dalam pendistribusian tabung gas LPG 3 kg. Perusahaan menginginkan total jarak pendistribusian yang minimum agar mengurangi biaya transportasi (Simanungkalit et al., 2022). Penelitian sebelumnya telah menyelesaikan permasalahan ini menggunakan metode CVRP dengan algoritma *Sweep.* Hasil yang didapatkan yaitu 7 kelompok rute pendistribusian. Hasil ini memiliki pengurangan jarak sebesar 14,35 km atau 10,77% dari total jarak tempuh sebelumnya (Simanungkalit et al., 2022).

Penggunaan algoritma *Sweep* pada penelitian sebelumnya telah terbukti efektif dalam perhitungan distribusi barang. Namun, hasil penelitian ini belum memiliki kajian yang dapat membandingkan efektivitas algoritma *Sweep* dengan algoritma lainnya. Penelitian sebelumnya diasumsikan memiliki keterbatasan dalam mengakomodasi berbagai atribut seperti prioritas pelanggan dan kondisi dinamis seperti fleksibilitas rute alternatif yang sering terjadi pada proses distribusi.

Algoritma Dijkstra relatif sederhana dalam memperoleh jalur terpendek dan telah banyak diaplikasikan dalam pencarian rute terdekat, sehingga tepat digunakan untuk optimisasi jarak atau waktu tempuh (Suryani & Murniyasih, 2022). Algoritma Genetika dapat menyelesaikan berbagai permasalahan seperti penentuan rute terpendek dengan dua tujuan, penentuan jalur multi-tujuan, rencana evakuasi pada situasi darurat, dan optimisasi distribusi rute (Trisatya, 2023). Algoritma ini tidak memiliki kriteria khusus sehingga dapat menyelesaikan permasalahan dalam waktu yang relatif singkat dengan alternatif solusi yang bernilai objektif (Saputra & Sukmono, 2024). Algoritma Genetika dapat mereduksi atribut-atribut yang kurang dominan sehingga mampu memberikan hasil yang lebih optimal (Saputra & Sukmono, 2024). Oleh karena itu, penelitian ini bertujuan untuk memperbaiki solusi yang dihasilkan dari penelitian

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

sebelumnya sehingga memperoleh solusi rute distribusi yang lebih baik dengan menggunakan algoritma Dijkstraa dan algoritma Genetika. Keterbaruan penelitian ini dibandingkan penelitian terdahulu dapat dilihat pada Tabel 1 di bawah ini.

Tabel 1. Keterbaruan Penelitian

Aspek	Penelitian Terdahulu	Keterbaruan
Objek penelitian	Distribusi gas LPG 3 kg	Distribusi gas LPG 3 kg
Model yang digunakan	CVRP	CGVRP
Algoritma	*Sweep*	*Sweep*, Djikstraa, dan Genetika
Pendekatan optimasi	*Single-stage optimization*	*Multi-stage optimization*
Fleksibilitas rute	Terbatas pada hasil algoritma *Sweep*	Lebih fleksibel dengan kombinasi tiga algoritma
Keunggulan	Implementasi lebih mudah	- Dapat menangani kondisi dinamis - Mengoptimalkan hasil *Sweep* dengan dua algoritma tambahan - Mempertimbang kan lebih banyak atribut

Metodologi

Pemodelan Data

CGVRP merupakan variasi dari permasalahan VRP yang terdiri dari beberapa klaster rute kendaraan (Hermanto, Adiasa, et al., 2020). Setiap verteks pada setiap klaster harus dikunjungi secara berurutan dalam rute kendaraan. CGVRP dapat menentukan biaya minimum dari pendistribusian tanpa melebihi kapasitas angkut kendaraan (Hermanto, Adiasa, et al., 2020). CGVRP dapat diselesaikan menggunakan pendekatan eksak dan pendekatan heuristik. Model eksak berupa notasi penyelesaian terhadap kendala-kendala dari permasalahan. Model matematis dari CGVRP adalah sebagai berikut (Qi et al., 2024).

1. Notasi:

n = Jumlah pelanggan

i,j = Himpunan node asal – tujuan (node 0 adalah depot), $ij \in \{0,1,2,\ldots,n\}$

C = Klaster, $C(1,2,\ldots,m)$

K = Indeks kendaraan, $k\{1,2,\ldots,k)$

V = Himpunan node yang terdiri dari pelanggan dan depot

d_i = Permintaan di node i

d_{ij} = Permintaan yang dimulai dari node i ke node j

q = Kapasitas kendaraan k (ton)

c_{ijk} = Jarak tempuh perjalanan dari node i ke node j dengan kendaraan k

2. Variabel Keputusan

x_{ijk} = 1, jika mengunjungi dari node i ke node j dengan kendaraan k

x_{ijk} = 0, jika tidak mengunjungi node i ke node j dengan kendaraan k

y_{ik} = 1, jika mengunjungi node i dengan kendaraan k

y_{ik} = 0, jika tidak mengunjungi node i dengan kendaraan k

3. Fungsi Tujuan

Minimize $Z = \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk}$ Pers. 1

4. Fungsi Kendala

a. Verteks setiap kluster harus dikunjungi setidaknya satu kali.

$\sum_{i \in C_j}^{n} \sum_{k=1}^{n} y_{ik} \geq 1, \forall c_j \in C$ Pers. 2

b. Jumlah aliran kendaraan yang masuk harus sama dengan jumlah aliran kendaraan yang keluar.

$\sum_{j \in V}^{n} x_{ijk} = y_{ik}$, $\forall i \in V, \forall k \in K$ Pers. 3

$\sum_{i \in V}^{n} x_{ijk} = y_{ik}$, $\forall j \in V, \forall k \in K$ Pers. 4

c. Muatan masing-masing kendaraan tidak melebihi kapasitas.

$\sum_{i \in V}^{n} d_i \sum_{i \in V}^{n} y_{ik} = q, \forall k \in K$ Pers. 5

d. Setiap rute berawal dan berakhir di depot.

$\sum_{j \in V}^{n} x_{0jk} = 1$, $\forall k \in K$ Pers. 6

$\sum_{i \in V}^{n} x_{i0k} = 1$, $\forall k \in K$ Pers. 7

e. Variabel keputusan adalah variabel biner.

$x_{ijk} = \{0,1\}$, $\forall i,j \in V$ Pers. 8

$y_{ijk} = \{0,1\}$, $\forall i,j \in V$ Pers. 9

Pendekatan eksak dapat memberikan solusi yang optimal, akan tetapi pendekatan ini memiliki waktu perhitungan yang lama terhadap persoalan yang besar ataupun kompleks. Untuk itu dikembangkan pendekatan penyelesaian dengan metode heuristik/metahueristik (Ruben & Imran, 2020)

Algoritma *Sweep*

Algoritma *Sweep* merupakan salah satu algoritma heuristik klasik dengan metode *clustering* yang paling sederhana dalam menyelesaikan permasalahan (Simanungkalit

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

et al., 2022). Algoritma *Sweep* menggunakan metode dua fase dalam penyelesaiannya, dengan fase pertama berupa *clustering* pelanggan dan fase kedua berupa pembentukan rute pendistribusian (Ruben & Imran, 2020). Berikut tahapan penyelesaian algoritma *Sweep*.

1. Tahap pengelompokan (*clustering*)

a. Menentukan setiap titik pelanggan dalam koordinat kartesius dan menetapkan pusat koordinat pada lokasi depot.
b. Menentukan semua koordinat polar pada setiap pelanggan yang berhubungan dengan depot.
c. Kelompokkan pelanggan mulai dari sudut polar terkecil dan seterusnya secara berurutan.
d. Pengelompokan dihentikan ketika satu kluster akan melebihi kapasitas kendaraan.
e. Jika pengelompokan awal dihentikan, dilanjutkan dengan pengelompokan pada kluster baru.
f. Lakukan secara berulang-ulang hingga semua pelanggan mendapatkan kluster.

2. Tahap pembentukan rute

Tahap pembentukan rute menggunakan *Nearest Neighbor*.

a. Memilih depot sebagai titik awal
b. Mencari jarak pelanggan yang terdekat dari depot dan menetapkan pelanggan tersebut sebagai titi kedua dari rute.
c. Lakukan secara berulang-ulang hingga semua titik pelanggan terpenuhi.

Algoritma Dijkstra

Algoritma Dijkstra merupakan sebuah algoritma penentuan rute terpendek dengan pencarian graf yang berarah atau tak berarah. Algoritma Dijkstra menentukan jarak terpendek antara 2 titik dari titik awal ke titik tujuan. Penyelesaian algoritma Dijkstra dilakukan dengan memberikan bobot pada semua rute alternatif yang memungkinkan menjadi solusi optimal (Gautama & Hermanto, 2020). Berikut tahapan penyelesaian algoritma Dijkstra.

1. Penentuan titik awal dan matriks jarak dari tiap-tiap kluster pelanggan.
2. Penentuan bobot jarak dari masing-masing *node*. *Node* awal diberikan bobot 0 dan *node* lainnya berbobot tak hingga.
3. Pengaturan pada *node* keberangkatan untuk *node* pertama dan pengaturan setiap *node* untuk yang belum dilewati sekalipun.
4. *Node* keberangkatan dilakukan perhitungan *node* terdekat lainnya dan *node* yang belum dilewati. Apabila jarak lebih kecil dari perhitungan jarak awal, maka simpan data jarak yang lebih pendek.
5. Dilakukan *sign node* atau penandaan pada *node* yang sudah dilalui. Dilanjutkan dengan memilih *node* selanjutnya dengan nilai bobot terkecil.
6. *Node* yang sekalipun belum terlewati diatur dengan memilih bobot jarak terkecil, ulangi langkah hingga semua *node* terlewati.

Algoritma Genetika

Algoritma Genetika adalah salah satu pendekatan metaheuristik yang dirancang untuk menemukan solusi optimal dari berbagai permasalahan yang kompleks. Algoritma Genetika merupakan algoritma yang tahapan solusinya meniru mekanisme evolusi yang terjadi pada makhluk hidup (Nazarius et al., 2024). Adapun mekanisme dari proses pengembangbiakan pada algoritma Genetika sebagai berikut.

1. Tahap Inisialisasi

Tahap inisialisasi populasi dilakukan dengan membangkitkan beberapa individu dalam satu populasi (Nazarius et al., 2024). Penelitian ini menggunakan rute pendistribusian hasil perhitungan algoritma Dijkstra sebagai individu pertama, sedangkan hasil acak individu pertama akan menjadi individu kedua. Hasil acak individu pertama diperoleh dengan membangkitkan bilangan *random* [0,1] yang diurutkan berdasarkan nilai bilangan *random* terkecil hingga terbesar. Jumlah individu yang digunakan pada penelitian ini, yaitu sebanyak 14 individu.

2. Tahap Evaluasi

Setiap individu akan direpresentasikan dalam bentuk kromosom atau genom. Kromosom berisikan informasi terkait lokasi yang akan dikunjungi dalam suatu rute pengiriman. Evaluasi terhadap setiap individu didasarkan pada nilai *fitness*. Kemampuan kromosom untuk tetap muncul pada generasi berikutnya ditentukan oleh nilai *fitness* tersebut (Nazarius et al., 2024). Nilai *fitness* yang tinggi

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

menunjukkan semakin baiknya kualitas individu. Nilai *fitness* dari setiap individu dihitung dengan formula berikut.

$$Fitness = \frac{1}{\text{Total Jarak Distribusi}}$$

3. Tahap Seleksi

Seleksi kromosom bertujuan untuk menentukan kromosom terbaik yang akan menjadi induk dalam generasi selanjutnya. Metode seleksi yang digunakan pada penelitian ini, yaitu metode *roulette wheel* karena metode ini memberikan probabilitas seleksi yang proporsional dengan *fitness* dari setiap individu (Mayyani et al., 2023). Hal Ini mengisyaratkan bahwa individu dengan *fitness* yang lebih tinggi memiliki peluang lebih besar untuk dipilih, ini memungkinkan individu yang lebih kuat untuk berkontribusi lebih banyak pada generasi berikutnya (Nazarius et al., 2024).

Prinsip metode *roulette wheel* yaitu setiap individu dibagi ke beberapa wilayah tertentu. Nilai acak individu yang terpilih mempunyai rentang nilai acak [0,1]. Area nilai *random* masing-masing individu ditentukan dengan rumus $pcum_{n-1} < ri \leq pcum_n$ dengan pcum merupakan nilai kumulatif peluang *fitness* individu. Hasil dari tahap ini akan menghasilkan induk sebagai rute yang terpilih untuk menghasilkan individu baru.

4. Tahap Kawin Silang

Proses kawin silang atau *crossover* yang dilakukan pada penelitian ini menggunakan metode *Order Crossover* (OX). Alasan penggunaan metode ini karena OX dapat menjaga urutan relatif dari kromosom dalam individu, sehingga mampu memberikan solusi yang valid (Wulandari et al., 2019). Nilai probabilitas pertukaran yang digunakan, yaitu sebesar 0,7. Jika nilai acak pasangan lebih kecil dari probabilitas, maka proses *crossover* dapat dilakukan dan sebaliknya. Jika tidak terjadi persilangan, anak yang dihasilkan adalah induk itu sendiri.

5. Tahap Mutasi

Proses mutasi dilakukan dengan cara mengganti satu gen yang terpilih secara acak dengan nilai acak baru (Nazarius et al., 2024). Proses mutasi yang digunakan pada penelitian ini menggunakan metode *swapping mutation* karena metode ini dapat menjaga struktur permutasi tetap valid dengan hanya menukar dua elemen (Wulandari et al., 2019). Probabilitas yang digunakan dalam penelitian ini adalah 0,3. Jika nilai *random* kurang dari nilai probabilitas, maka proses mutasi dapat dilakukan begitu juga sebaliknya. Proses ini menghasilkan dua bilangan acak yang berfungsi untuk menentukan lokasi gen yang bermutasi. Gen yang menggantikan dua bilangan acak akan dipindahkan ke tempat bilangan lainnya. Proses mutasi menghasilkan populasi baru yang disebut dengan generasi 1.

6. Regenerasi

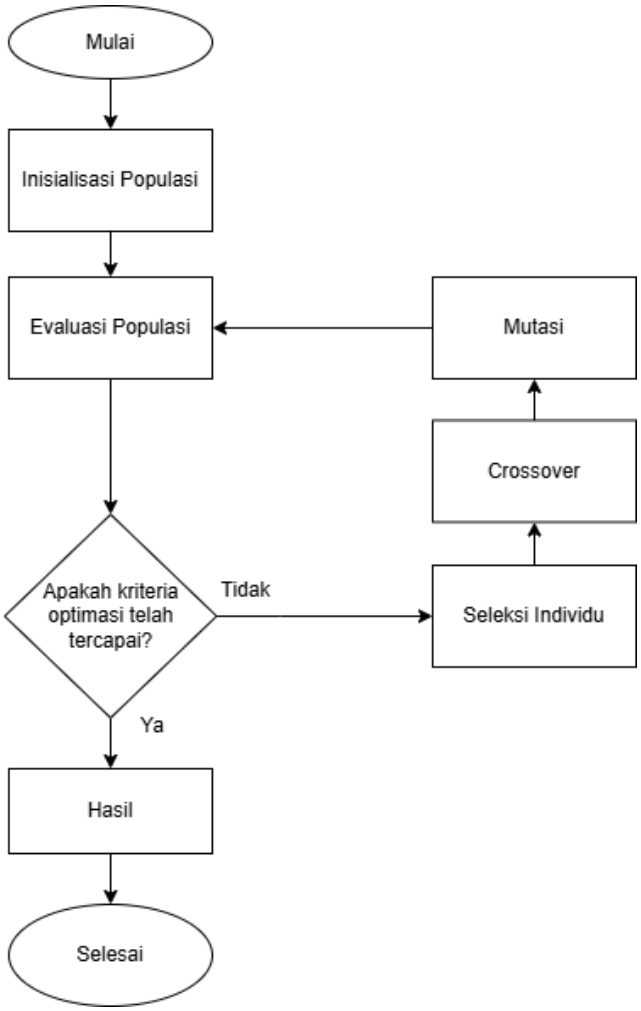
Regenerasi adalah tahap pengulangan proses seleksi individu, *crossover*, mutasi, dan evaluasi dari populasi tahap sebelumnya (Nazarius et al., 2024). Regenerasi akan terus berlanjut hingga kriteria berhenti (*stopping criteria*) terpenuhi. Kriteria berhenti yang umum digunakan dalam algoritma Genetika, yaitu jumlah generasi, nilai *fitness*, dan waktu eksekusi (Misbahul Munir et al., 2023). Populasi baru yang dihasilkan dari pengulangan akan digunakan sebagai populasi awal dalam pengulangan berikutnya.

Penelitian ini menerapkan kriteria berhenti dengan jumlah maksimal 100 generasi agar dapat mempercepat proses pencarian rute tanpa mengurangi kualitas solusi yang dihasilkan (Ding et al., 2024). Selain itu, kriteria berhenti ini juga dapat menghindari terjadinya "*overfitting*" solusi karena proses iterasi terlalu lama. Penggunaan kriteria berhenti dengan batas 100 generasi membuat pengulangan secara otomatis berhenti saat mencapai jumlah tersebut. Gambar 1 merupakan *flowchart* dari algoritma Genetika yang dimulai dari proses inisialisasi populasi hingga menghasilkan solusi terbaik. Pencarian solusi akan melalui proses pengulangan jika kriteria optimasi atau berhenti masih belum tercapai.

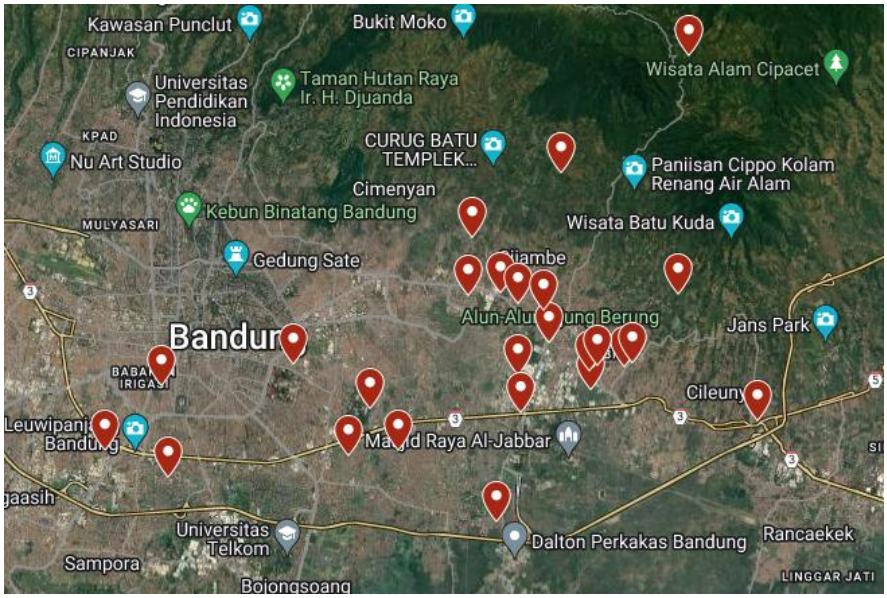
Hasil dan Diskusi

Penyelesaian masalah pendistribusian gas LPG 3 kg pada penelitian ini menggunakan algoritma Dijkstra dan algoritma Genetika. Gas LPG 3 kg didistribusikan ke 24 lokasi pelanggan yang berbeda-beda. Adapun sebaran lokasi pendistribusian gas LPG 3 kg dapat dilihat pada Gambar 2.

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145



Gambar 1. *Flowchart* Algortima Genetika



Gambar 2. Sebaran Lokasi Pendistribusian

Pendistribusian gas LPG 3 kg mengunjungi 24 lokasi pengantaran yang berbeda-beda dengan jumlah permintaan yang beragam. Data jumlah permintaan pelanggan dapat dilihat pada Tabel 2.

Penentuan rute pendistribusian gas LPJ 3 kg diselesaikan menggunakan algoritma Dijkstra dan algoritma Genetika. Penyelesaian rute dengan algoritma Dijkstra membutuhkan inisial kluster sebagai pedoman awal. Rute

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

pendistribusian yang dihasilkan dari algoritma *Sweep* diasumsikan sebagai klaster-klaster yang akan dilanjutkan perhitungannya menggunakan algoritma Dijkstra. Berikut rute pendistribusian berdasarkan algoritma *Sweep* dan algoritma Dijkstraa berturut-turut terdapat pada Tabel 3 dan Tabel 4.

Rute pendistribusian setiap kluster pada Tabel 3 yang dihasilkan dari algoritma Dijkstra, diasumsikan menjadi inisial kluster pada algoritma Genetika untuk menghasilkan rute yang lebih baik menggunakan Python. Penggunaan Python pada penelitian ini bertujuan agar penyelesaian dapat dilakukan dengan efektif dan tidak membutuhkan waktu perhitungan yang lama seperti cara manual. Adapun hasil penyelesaian pendistribusian dengan algoritma Genetika dapat dilihat pada Tabel 5 berikut.

Tabel 2. Permintaan Pelanggan

Node	**Permintaan**	***Node***	**Permintaan**
0	0	13	70
1	50	14	60
2	85	15	100
3	70	16	100
4	65	17	30
5	30	18	30
6	50	19	30
7	110	20	30
8	30	21	50
9	50	22	70
10	85	23	65
11	105	24	60
12	30		

Tabel 3. Rute Pendistribusian Berdasarkan Algoritma *Sweep*

Sweep			
Kelompok	**Rute**	**Rute Pendistribusian**	**Total Jarak Tempuh**
1	1	0, 20, 18, 23, 21, 19, 4, 0	15,15
	2	0, 22, 17, 24, 0	10,4
2	1	0, 12, 14, 13, 15, 0	35,7
	2	0, 11, 9,10, 0	19,7
3	1	0, 5, 1, 8, 7, 0	10,05
	2	0, 3, 2, 6, 0	7,85
	3	0, 16, 0	20
Total			118,85

Sumber: (Simanungkalit et al., 2022)

Tabel 4. Rute Pendistribusian Berdasarkan Algoritma Dijkstra

Dijkstra			
Kelompok	**Rute**	**Rute Pendistribusian**	**Total Jarak Tempuh**
1	1	0, 18, 23, 20, 19, 21, 4, 0	14,946
	2	0, 22, 17, 24, 0	10,4
2	1	0, 12, 13, 14, 15, 0	32,6
	2	0, 11, 10, 9, 0	18,95
3	1	0, 7, 1, 8, 5, 0	10,03
	2	0, 3, 2, 6, 0	7,85
	3	0, 16, 0	20
Total			114,779

Tabel 5. Kluster Pendistribusian Berdasarkan Algoritma Genetika

No	**Kluster**	**Total Permintaan (Ton)**	**Jarak Tempuh (km)**
1	0, 18, 23, 20, 21, 19, 4, 0	270	13,716
2	0, 17, 24, 22, 0	160	10,35
3	0, 12, 13, 14, 15, 0	260	31,6
4	0, 11, 10, 9, 0	240	18,95
5	0, 1, 5, 8, 7, 0	220	9,533
6	0, 3, 2, 6, 0	205	7,85
7	0, 7, 0	100	20
Total Jarak Tempuh			112

Tabel 6 berikut ini menunjukkan perbandingan total jarak yang diperoleh dari perhitungan menggunakan algoritma *Sweep* dan algoritma Genetika.

Kesimpulan

Penggunaan algoritma Dijkstra dan algoritma Genetika pada penelitian ini berfungsi untuk mendapatkan solusi yang lebih baik dari masalah pendistribusian gas LPG 3 kg. Berdasarkan hasil perhitungan yang telah dilakukan pada pendistribusian gas yang terdiri atas 7 kluster pengiriman, didapatkan total jarak tempuh pendistribusian menggunakan algoritma Dijkstra sebesar 114,779 km dan algoritma Genetika sebesar 112 km. Hal tersebut memperlihatkan adanya pengurangan total jarak tempuh pendistribusian sebesar 6,85 km dibandingkan penyelesaian algoritma

Sweep pada penelitian sebelumnya. Jumlah pengurangan ini cukup besar sehingga dapat menghemat pengeluaran biaya perusahaan sebesar 5,76%. Penelitian selanjutnya dapat menggunakan metode metaheuristik lainnya untuk mendapatkan rute yang lebih baik.

Daftar Pustaka

Awalloedin, N., Gata, W., & Qomariyah, N. (2022). Algoritma Dijkstra dalam Penentuan Rute Terpendek pada Jalan Raya Antar Kota Jakarta - Tangerang. *Just IT : Jurnal Sistem Informasi, Teknologi Informasi dan Komputer*, *13*(1), 8–13.

Baihaqi, M. M., & Hermansyah, M. (2023). Optimalisasi Vehicle Routing Problem Pada UD. Kopwan Yasmin Nongkojajar. *Journal of Scientech Research and Development*, *5*(2), 62–71. https://doi.org/10.56670/jsrd.v5i2.159

Busono, S. (2020). Optimasi Naive Bayes Menggunakan Algoritma Genetika Sebagai Seleksi Fitur untuk Memprediksi Performa Siswa. *Jurnal Ilmiah Teknologi Informasi Asia*, *14*(1), 31–40. https://doi.org/10.32815/jitika.v14i1.400

Chandra, A., & Setiawan, B. (2018). Optimasi Jalur Distribusi dengan Metode Vehicle Routing Problem (VRP). *Jurnal Manajemen Transportasi & Logistik (JMTRANSLOG)*, *5*(2), 105–115. https://doi.org/10.54324/j.mtl.v5i2.233

Ding, B., Rao, Z., Yin, W., Liu, Y., Fang, J., Wang, Y., & Jin, P. (2024). The Optimization of Urban Traffic Routes Using an Enhanced Genetic Algorithm: A Case Study of Beijing South Railway Station. *Applied Sciences (Switzerland)*, *14*(14). https://doi.org/10.3390/app14146130

Fatnita, A. V, & Lukmandono. (2020). Optimasi Rute Distribusi Tabung LPG 3 Kg dengan Menggunakan Alogaritma Genetika pada Penyelesaian Capacitated Vehicle Routing Problem (CVRP) (Studi kasus pada PT. Jana Pusaka Migas). *Prosiding Seminar Nasional Sains dan Teknologi Terapan*, *1*(1), 39–46.

Firmansyah, B., & Hermanto, K. (2021). Analisis Perbandingan Algoritma Warshall dan Dijkstraa pada Metode GVRP dalam Penentuan Rute Terpendek (Studi Kasus : PT Yakult Cabang Sumbawa). *Jurnal Tambora*, *5*(1).

Gautama, I. P. W., & Hermanto, K. (2020). Penentuan Rute Terpendek dengan Menggunakan Algoritma Dijkstra pada Jalur Bus Sekolah. *Jurnal Matematika (JMAT)*, *10*(2), 116–123. https://doi.org/10.24843/JMAT.2020.v10.i02.p128

Hanzani, I. W., Aprilia, R., & Panjaitan, D. J. (2023). Penerapan Metode Stepping Stone dan VRP (Vehicle Routing Problem) Untuk Transportasi Pengiriman Hasil Produksi Kelapa Sawit. *Journal of Science and Social Research*, *4307*(3), 828–834.

Hermanto, K., Adiasa, I., Altarisi, S., Rabani, R., & Amirul, M. (2020). Rute Usulan Pendistribusian LPG Menggunakan Model Clustered Generalized Vehicle Routing Problem (CGVRP) dan Algoritma Dijkstra. *Journal of Physics: Conference Series*, *19*(1), 27–36. https://doi.org/10.20961/performa.19.1.41858

Hermanto, K., & Ruskartina, E. (2018). Usulan Rute Optimal Distribusi Sampah Shift I Kota Sumbawa Besar Menggunakan Metode GVRP. *Eigen Mathematis Journal*, *1*(2).

Hermanto, K., Ruskartina, E., & Gunawan. (2020). Application of the CGVRP Model and the 0-1 Linear Programming Determines the RASKIN (Subsidized Rice) Distribution Route. *Performa: Media Ilmiah Teknik Industri*, *19*(1). https://doi.org/10.1088/1742-6596/1778/1/012002

Hidayah, A. A., Samsudin, & Triase. (2021). Penerapan Algoritma Dijkstra Pada Aplikasi Jasa Transportasi Online Di Kota Medan. *Al-Ulum: Jurnal Sains Dan Teknologi*, *7*(1), 9–13. https://doi.org/10.31602/ajst.v7i1.5710

Kurnia, G., Kurniawan, A. C., Nawadir, M., Yasmin, M. S., & Hibatullah, M. (2020). Route Optimization of Oil Country Tubular Goods Distribution Using Sweep and Savings Algorithm. *IPTEK Journal of Proceedings Series*, *0*(5), 28–33. https://doi.org/10.12962/j23546026.y2020i5.7927

Mafaza, G. A., & Muslim, E. (2023). Perancangan Rute Distribusi Air Minum dalam Kemasan dengan Capacitated Vehicle Routing Problem. *Matrik : Jurnal Manajemen dan Teknik Industri Produksi*, *23*(2),121. https://doi.org/10.30587/matrik.v23i2.4366

Mayyani, H., Nurbaiti, M., Supriyo, P. T., Aman, A., & Silalahi, B. P. (2023). Penerapan Algoritma Genetika Dengan Metode Roulette Wheel dan Replacement pada Optimasi Omzet. *MILANG Journal of Mathematics and Its Applications*, *19*(2), 153–172. https://doi.org/10.29244/milang.19.2.153-172

Misbahul Munir, M., Pujianto, A., & Aulia Muhali Lamuru, H. (2023). Optimisasi Algoritma Genetika dengan Particle Swarm Optimization (PSO) untuk Sistem Rekomendasi Diet Gizi bagi Penderita Diabetes. *Jurnal Riset Sistem Dan Teknologi Informasi (RESTIA)*, *1*(2), 38–48. https://doi.org/10.30787/restia.v1i2.1289

Moudya, F., Rarasati, N., & Syafmen, W. (2023). Optimisasi Rute pada CVRP dalam Pendistribusian Gas Oksigen Menggunakan Algoritma Clarke and Wright Savings. *FIBONACCI: Jurnal Pendidikan Matematika dan Matematika*, *9*(1),105. https://doi.org/10.24853/fbc.9.1.105-118

Nazarius, A., Delon Pratama, R., Aryapati Soebagijo, R., Priskila, R., & Handrianus Pranatawijaya, V. (2024). Pengimplementasian Algoritma Genetika dalam Sistem Penjadwalan Praktikum. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *8*(3), 3237–3243. https://doi.org/10.36040/jati.v8i3.9656

Pratiwi, H. (2022). Application of The Dijkstra Algorithm to Determine the Shortest Route from City Center Surabaya to Historical Places. *Jurnal Teknologi dan Sistem Informasi Bisnis*, *4*(1), 213–223. https://doi.org/10.47233/jteksis.v4i1.407

Qi, D., Zhao, Y., Wang, Z., Wang, W., Pi, L., & Li, L. (2024). Joint Approach for Vehicle Routing Problems Based on Genetic Algorithm and Graph Convolutional Network. *Mathematics*, *12*(19), 1–18. https://doi.org/10.3390/math12193144

Ruben, M., & Imran, A. (2020). Usulan Rute Distribusi Menggunakan Algoritma Sweep dan Local Search (Studi Kasus di Perusahaan X). *Jurnal Rekayasa Sistem Industri*, *6*(1), 40–44.

Saputra, N. Q., & Sukmono, T. (2024). Analisa Optimalisasi Rute Distribusi untuk Mengefisiensikan Logistik Menggunakan Algoritma Genetika. *Jurnal Manajemen dan Teknik Industri Produksi*, *XXV*(1), 67–78. https://doi.org/10.350587/Matrik

Simanungkalit, I., Sawaluddin, Gultom, P., & Nasution, P. K. (2022). Analysis of The Use of Sweep Algorithms to Solve Capacitated Vehicle Routing Problems. *Jurnal Matematika dan Pendidikan Matematika*, *5*(2), 161–166.

Suryani, L., & Murniyasih, E. (2022). Pencarian Rute Terpendek pada Aplikasi Ojek Sampah dengan Menggunakan Algoritma Djikstra. *Jurnal TEKINKOM*, *5*(2), 385–392. https://doi.org/10.37600/tekinkom.v5i2.586

Sutresno, S. A., & Suni, E. K. (2023). Rancang Bangun Aplikasi Petani Sawah Sugita Menggunakan Algoritma Dijkstra untuk Mempermudah Penjualan Hasil Pertanian. *Journal of Information System Research (JOSH)*, *5*(1), 110–121. https://doi.org/10.47065/josh.v5i1.4279

Trisatya, D. (2023). Digitalisasi Pengembangan Sistem KKN Universitas Pancasakti Tegal Berbasis GIS Menggunakan Metode Prototyping. *Jurnal Teknik Informatika dan Sistem Informasi*, *10*(4), 272–280.

Wang, L., & Lu, J. (2019). A Memetic Algorithm With Competition for the Capacitated Green Vehicle Routing Problem. *Journal of Automatica Sinica*, *6*(2), 516–526.

Wulandari, S., Helmi, & Yudhi. (2019). Penyelesaian Multiple Travelling Salesman Problem (Multi-TSP) dengan Metode Order Crossover dalam Algoritma Genetika (Studi Kasus: Data Pelanggan Agen Surat Kabar di Kota Singkawang). *Bimaster : Buletin Ilmiah Matematika, Statistika dan Terapannya*, *8*(2),157–166. https://doi.org/10.26418/bbimst.v8i2.31310

Zubir, A. A., Andriansyah, & Derisma, S. (2022). Determining the Distribution Route Using the Clustered Generalised Vehicle Routing Problem Model and Dijkstra's Algorithm. *Journal of Science, Technology, and Virtual Culture*, *2*(2), 232–240

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

Lampiran 1 Matriks Jarak Kluster 1

***Cluster* 1**							
Kode Toko	Depot (0)	Willy (20)	Eden (18)	Hj. Lilis (23)	Sumber (21)	Triyono (19)	Itang (4)
Depot (0)	0	1,51	1	1,5	3	2	4,7
Willy (20)	1,51	0	2,21	1,9	0,806	2,506	1,5
Eden (18)	1	2,21	0	2,1	5,5	1,9	5
Hj. Lilis (23)	1,5	1,9	2,1	0	1,42	1,4	5,8
Sumber (21)	3	0,806	5,5	1,42	0	1,4	1,34
Triyono (19)	2	2,506	1,9	1,4	1,4	0	1,81
Itang (4)	4,7	1,5	5	5,8	1,34	1,81	0

Lampiran 2 Perhitungan Algoritma Dijkstra Kluster 1

***Cluster* 1**								
Iterasi	**D(0)**	**D(20)**	**D(18)**	**D(23)**	**D(21)**	**D(19)**	**D(4)**	**L**
0	0	~	~	~	~	~	~	0
1	-	Min {~; 0+1,51}= 1,51	Min {~; 0+1} = 1	Min {~; 0+1,5} = 1,5	Min {~; 0+3} = 3	Min {~; 0+2} = 2	Min {~; 0+4,7} = 4,7	0, 18
2	-	Min {1,51; 1+2,21}= 1,51	-	Min {1,5; 1+2,1} = 1,5	Min {3; 1+5,5} = 3	Min {2; 1+1,9} = 2	Min {4,7; 1+5} = 4,7	0, 18, 23
3	-	Min {1,51; 1,5+1,9}= 1,51	-	-	Min {3; 1,5+1,42} = 2,92	Min {2; 1,5+1,4} = 2	Min {4,7; 1,5+5,8} = 4,7	0, 18, 23, 20
4	-	-	-	-	Min {2,92; 1,51+0,806} = 2,316	Min {2; 1,51+2,506} = 2	Min {4,7; 1,51+1,5} = 3,01	0, 18, 23, 20, 19
5	-	-	-	-	Min {2,316; 2+1,4} = 2,316	-	Min {3,01; 2+1,81} = 3,01	0, 18, 23, 20, 19, 21
6	-	-	-	-	-	-	Min {3,01; 2,316+1,34} = 3,01	0, 18, 23, 20, 19, 21, 4

Lampiran 3 Matriks Jarak Kluster 2

Cluster 2				
Kode Toko	Depot (0)	Dedi S (22)	Vianti L (17)	Timbul T (24)
Depot (0)	0	2,4	3,5	3,65
Dedi S (22)	2,4	0	2,9	3
Vianti L (17)	3,5	2,9	0	1,45
Timbul T (24)	3,65	3	1,45	0

Lampiran 4 Perhitungan Algoritma Dijkstra Kluster 2

Cluster 2					
Iterasi	**D(0)**	**D(22)**	**D(17)**	**D(24)**	**L**
0	0	~	~	~	0
1	-	Min {~; 0+2,4}= 2,4	Min {~; 0+3,5} = 3,5	Min {~; 0+3,65} =	0, 22
2	-	-	Min {3,5; 2,4+2,9} = 3,5	Min {3,65; 2,4+4,7} = 3,65	0, 22, 17
3	-	-	-	Min {3,65; 3,5+1,45} = 3,65	0, 22, 17, 24

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

Lampiran 5 Matriks Jarak Kluster 3

Cluster 3					
Kode Toko	Depot (0)	Ernawati	H. Eni (14)	H. Isoh (13)	Endri S
Depot (0)	0	8,4	12,2	11,3	12,4
Ernawati (12)	8,4	0	4,3	4,2	5,3
H. Eni (14)	12,2	4,3	0	4,3	3,3
H. Isoh (13)	11,3	4,2	4,3	0	6,3
Endri S (15)	12,4	5,3	3,3	6,3	0

Lampiran 6 Perhitungan Algoritma Dijkstra Kluster 3

Cluster 3						
Iterasi	**D(0)**	**D(12)**	**D(14)**	**D(13)**	**D(15)**	**L**
0	0	~	~	~	~	0
1	-	Min {~; 0+8,4}= 8,4	Min {~; 0+12,2} = 12,2	Min {~; 0+11,3} = 11,3	Min {~; 0+12,4} = 12,4	0, 12
2	-	-	Min {12,2; 8,4+4,3} = 12,2	Min {11,3; 8,4+4,2} = 11,3	Min {12,4; 8,4+5,3}= 12,4	0, 12, 13
3	-	-	Min {12,2; 11,2+3,3} = 12,2	-	Min {12,4; 11,2+3,3}= 12,4	0, 12, 13, 14
4	-	-	-	-	Min {12,4; 12,2+3,2}= 12,4	0, 12, 13, 14, 15

Lampiran 7 Matriks Jarak Kluster 4

Cluster 4				
Kode Toko	Depot (0)	Ella T (11)	Hj Neni (9)	Christian (10)
Depot (0)	0	5,15	6,3	6,05
Ella T (11)	5,15	0	2,4	1,4
Hj Neni (9)	6,3	2,4	0	6,1
Christian (10)	6,05	1,4	6,1	0

Lampiran 8 Perhitungan Algoritma Dijkstra Kluster 4

Cluster 4					
Iterasi	**D(0)**	**D(11)**	**D(9)**	**D(10)**	**L**
0	0	~	~	~	0
1	-	Min {~; 0+5,15}= 5,15	Min {~; 0+6,3} = 6,3	Min {~; 0+6,05} = 6,05	0, 11
2	-	-	Min {6,3; 5,15+2,4} = 6,3	Min {6,05; 5,15+1,4} = 6,05	0. 11, 10
3	-	-	Min {6,3; 6,05+6,1} = 6,3	-	0, 11, 10, 9

Lampiran 9 Matriks Jarak Kluster 5

Cluster 5					
Kode Toko	Depot (0)	Ragil (5)	Ayu (1)	Sandi (8)	Harun (7)
Depot (0)	0	3	2	2,9	1,517
Ragil (5)	3	0	1,1	1,4	2,9
Ayu(1)	2	1,1	0	1,116	3
Sandi (8)	2,9	1,4	1,116	0	3,32
Harun (7)	1,517	2,9	3	3,32	0

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

Lampiran 10 Perhitungan Algoritma Dijkstra Kluster 5

Cluster 5						
Iterasi	**D(0)**	**D(5)**	**D(1)**	**D(8)**	**D(7)**	**L**
0	0	~	~	~	~	0
1	-	Min {~; 0+3}= 3	Min {~; 0+2} = 2	Min {~; 0+2,9} = 2,9	Min {~; 0+1,517} = 1,517	0, 7
2	-	Min {3; 1,517+2,9}= 3	Min {2; 1,517+3} = 2	Min {2,9; 1,517+3,32} = 2,9	-	0, 7, 1
3	-	Min {3; 2+1,1}= 3	-	Min {2,9; 2+1,116} = 2,9	-	0, 7, 1, 8
4	-	Min {3; 2,9+1,4}= 3	-	-	-	0, 7, 1, 8, 5

Lampiran 11 Matriks Jarak Kluster 6

Cluster 6				
Kode Toko	Depot (0)	Polin (3)	Nanang (2)	Yanto (6)
Depot (0)	0	2,6	3,3	3,6
Polin (3)	2,6	0	1,3	2,4
Nanang (2)	3,3	1,3	0	0,35
Yanto (6)	3,6	2,4	0,35	0

Lampiran 12 Perhitungan Algoritma Dijkstra Kluster 6

Cluster 6					
Iterasi	**D(0)**	**D(3)**	**D(2)**	**D(6)**	**L**
0	0	~	~	~	0
1	-	Min {~; 0+2,6}= 2,6	Min {~; 0+3,3} = 3,3	Min {~; 0+3,6} = 3,6	0, 3
2	-	-	Min {3,3; 2,6+1,3} = 3,3	Min {3,6; 3,3+2,4} = 3,6	0, 3, 2
3	-	-	-	Min {3,6; 2,6+0,35} = 3,6	0, 3, 2, 6

Lampiran 13 Matriks Jarak Kluster 7

Cluster 7		
Kode Toko	Depot (0)	Dedi J (16)
Depot (0)	0	10
Dedi J (16)	10	0

Lampiran 14 Perhitungan Algoritma Dijkstra Kluster 7

Cluster 7			
Iterasi	**D(0)**	**D(16)**	**L**
0	0	~	0
1	-	Min {~; 0+9,3}= 9,3	0, 16

Lampiran 15 Coding Python

```
import random
import numpy as np

# Definisikan parameter algoritma genetika
POPULATION_SIZE = 2
NUM_GENERATIONS = 100
CROSSOVER_PROB = 0.7
MUTATION_PROB = 0.3

# Definisikan matriks jarak
distance_matrix = [
    [0, 2, 3.3, 2.6, 4.7, 3, 3.6, 1.517, 3, 7.9, 6.05, 5.15, 8.4, 14.3, 13.8, 13.4, 10, 3.5, 1, 2, 1.51, 3, 2.4, 1.5, 3.65],
```

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

```
    [2, 0, 1.3, 1.3, 4.1, 1.1, 20, 3, 1.116, 18.6, 19.3, 18.5, 21.9, 21.3, 18.8, 19.9, 19.8, 20.3, 20.3, 20.3, 20.3, 20.3,
20.3, 20.3, 20.3],
    [3.3, 1.3, 0, 1.3, 2.8, 20, 0.35, 20, 20, 20.6, 19.3, 20.4, 20.2, 20.3, 19.1, 19.8, 19.7, 18.8, 18.8, 18.8, 18.8,
18.8, 18.8, 18.8, 18.8],
    [2.6, 1.3, 1.3, 0, 3.6, 20, 2.4, 20, 20, 20.2, 19.2, 19.3, 19, 20.4, 18.8, 20.7, 19.8, 21.3, 21.3, 21.3, 21.3, 21.3,
21.3, 21.3, 21.3],
    [4.7, 4.1, 2.8, 3.6, 0, 20, 20, 20, 20, 19.7, 18, 19.5, 21.1, 18.5, 21.5, 20.1, 21.4, 20.3, 5, 1.81, 1.5, 1.34, 21.3,
5.8, 21.3],
    [3, 1.1, 20, 20, 20, 0, 2.7, 2.9, 1.4, 20.6, 18.7, 18.5, 19.8, 19.3, 23.3, 18, 19.2, 20, 20, 20, 20, 20, 20, 20, 20],
    [3.6, 20, 0.35, 2.4, 20, 2.7, 0, 6.78, 4.2, 21.2, 20.1, 19.4, 20.7, 21.4, 20, 20.2, 18.9, 20.9, 20.9, 20.9, 20.9,
20.9, 20.9, 20.9, 20.9],
    [1.517, 3, 20, 20, 20, 2.9, 6.78, 0, 3.32, 19.7, 19, 20.6, 20.5, 20.5, 18.9, 20.3, 19.2, 21.4, 21.4, 21.4, 21.4,
21.4, 21.4, 21.4, 21.4],
    [3, 1.116, 20, 20, 20, 1.4, 4.2, 3.32, 0, 19.8, 20.4, 19.3, 18.9, 21.2, 19.4, 21.1, 21.1, 18.8, 18.8, 18.8, 18.8,
18.8, 18.8, 18.8, 18.8],
    [7.9, 18.6, 20.6, 20.2, 19.7, 20.6, 21.2, 19.7, 19.8, 0, 6.1, 2.4, 4.8, 19, 18, 20, 19, 18, 18, 20, 18, 19, 20, 20,
20],
    [6.05, 19.3, 19.3, 19.2, 18, 18.7, 20.1, 19, 20.4, 6.1, 0, 1.4, 5.6, 18, 19, 18, 18, 19, 19, 18, 20, 20, 20, 20, 18],
    [5.15, 18.5, 20.4, 19.3, 19.5, 18.5, 19.4, 20.6, 19.3, 2.4, 1.4, 0, 7.2, 19, 18, 20, 18, 20, 18, 19, 20, 20, 20, 18,
19],
    [8.4, 21.9, 20.2, 19, 21.1, 19.8, 20.7, 20.5, 18.9, 4.8, 5.6, 7.2, 0, 4.2, 4.3, 5.3, 20, 19, 20, 19, 20, 18, 18, 20,
18],
    [14.3, 21.3, 20.3, 20.4, 18.5, 19.3, 21.4, 20.5, 21.2, 19, 18, 19, 4.2, 0, 4.3, 6.3, 19, 20, 20, 20, 19, 20, 19, 20,
20],
    [13.8, 18.8, 19.1, 18.8, 21.5, 23.3, 20, 18.9, 19.4, 18, 19, 18, 4.3, 4.3, 0, 3.3, 18, 18, 19, 18, 20, 20, 20, 19,
19],
    [13.4, 19.9, 19.8, 20.7, 20.1, 18, 20.2, 20.3, 21.1, 20, 18, 20, 5.3, 6.3, 3.3, 0, 20, 19, 20, 20, 20, 19, 18, 18,
18],
    [10, 19.8, 19.7, 19.8, 21.4, 19.2, 18.9, 19.2, 21.1, 19, 18, 18, 20, 19, 18, 20, 0, 1.106, 7.8, 9.1, 17.9, 12.9, 18,
19, 19],
    [3.5, 20.3, 18.8, 21.3, 20.3, 20, 20.9, 21.4, 18.8, 18, 19, 20, 19, 20, 18, 19, 1.106, 0, 1.181, 3.6, 15.6, 7.4, 2.9,
19, 1.45],
    [1, 20.3, 18.8, 21.3, 5, 20, 20.9, 21.4, 18.8, 18, 19, 18, 20, 20, 19, 20, 7.8, 1.181, 0, 1.9, 2.21, 5.5, 18, 2.1,
18],
    [2, 20.3, 18.8, 21.3, 1.81, 20, 20.9, 21.4, 18.8, 20, 18, 19, 19, 20, 18, 20, 9.1, 3.6, 1.9, 0, 2.506, 1.4, 18.2, 1.4,
19.4],
    [1.51, 20.3, 18.8, 21.3, 1.5, 20, 20.9, 21.4, 18.8, 18, 20, 20, 20, 19, 20, 20, 17.9, 15.6, 2.21, 2.506, 0, 0.806,
19.3, 1.9, 23.3],
    [3, 20.3, 18.8, 21.3, 1.34, 20, 20.9, 21.4, 18.8, 19, 20, 20, 18, 20, 20, 19, 12.9, 7.4, 5.5, 1.4, 0.806, 0, 20.2,
1.42, 22],
    [2.4, 20.3, 18.8, 21.3, 21.3, 20, 20.9, 21.4, 18.8, 20, 20, 20, 18, 19, 20, 18, 18, 2.9, 18, 18.2, 19.3, 20.2, 0,
1.8, 3],
    [1.5, 20.3, 18.8, 21.3, 5.8, 20, 20.9, 21.4, 18.8, 20, 20, 18, 20, 20, 19, 18, 19, 19, 2.1, 1.4, 1.9, 1.42, 1.8, 0,
3.05],
    [3.65, 20.3, 18.8, 21.3, 21.3, 20, 20.9, 21.4, 18.8, 20, 18, 19, 18, 20, 19, 18, 19, 1.45, 18, 19.4, 23.3, 22, 3,
3.05, 0],
]

# Definisikan cluster awal
initial_clusters = [
    [[20, 18, 23, 21, 19, 4], [23, 20, 18, 19, 21, 4]],
    [[22, 17, 24], [22, 14, 17]],
    [[12, 14, 13, 15], [12, 13, 14, 15]],
    [[11, 9, 10], [11, 10, 9]],
    [[5, 1, 8, 7], [1, 7, 8, 5]],
    [[3, 2, 6], [3, 6, 2]],
    [[7], [7]]
]

# Fungsi untuk menghitung jarak total rute
```

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

```
def calculate_distance(route, distance_matrix):
    total_distance = 0
    for i in range(len(route) - 1):
        total_distance += distance_matrix[route[i]][route[i + 1]]
    return total_distance

# Fungsi seleksi roulette wheel
def roulette_wheel_selection(population, fitness):
    max_val = sum(fitness)
    pick = random.uniform(0, max_val)
    current = 0
    for i in range(len(population)):
        current += fitness[i]
        if current > pick:
            return population[i]

# Fungsi crossover Order Crossover (OX)
def order_crossover(parent1, parent2):
    start, end = sorted(random.sample(range(len(parent1)), 2))
    child = [None] * len(parent1)
    child[start:end] = parent1[start:end]
    ptr = end
    for i in range(len(parent2)):
        if parent2[(i + end) % len(parent2)] not in child:
            child[ptr % len(child)] = parent2[(i + end) % len(parent2)]
            ptr += 1
    return child

# Fungsi mutasi swapping mutation
def swapping_mutation(individual):
    idx1, idx2 = random.sample(range(len(individual)), 2)
    individual[idx1], individual[idx2] = individual[idx2], individual[idx1]
    return individual

# Algoritma genetika
def genetic_algorithm(clusters, distance_matrix):
    best_routes = []
    for cluster in clusters:
        population = cluster
        for generation in range(NUM_GENERATIONS):
            fitness = [1 / calculate_distance([0] + individual + [0], distance_matrix) for individual in population]
            new_population = []
            while len(new_population) < POPULATION_SIZE:
                parent1 = roulette_wheel_selection(population, fitness)
                parent2 = roulette_wheel_selection(population, fitness)
                if random.random() < CROSSOVER_PROB:
                    # Ensure parents have enough elements for crossover
                    if len(parent1) >= 2 and len(parent2) >= 2:
                        child = order_crossover(parent1, parent2)
                    else:
                        child = parent1  # Keep one of the parents if too short
                else:
                    child = parent1
                if random.random() < MUTATION_PROB:
                    # Only apply mutation if the individual has enough elements
                    if len(child) >= 2:  # Check if child has at least 2 elements for swapping
                        child = swapping_mutation(child)
                new_population.append(child)
            population = new_population
        best_individual = min(population, key=lambda x: calculate_distance([0] + x + [0], distance_matrix))
```

DOI: https://doi.org/10.26593/jrsi.v14i1.8491.131-145

```
        best_routes.append([0] + best_individual + [0])
    return best_routes

# Jalankan algoritma genetika
optimal_routes = genetic_algorithm(initial_clusters, distance_matrix)

# Print rute optimal untuk setiap cluster
for i, route in enumerate(optimal_routes):
    print(f"Optimal route for cluster {i+1}: {route}")
```