BAB III
METODOLOGI PENELITIAN

Bab ini akan menjelaskan secara rinci metodologi dan perancangan implementasi *Hypergraph-Partitioning* dan Algoritma Genetika dalam penentuan lintasan distribusi barang. Penjelasan ini bertujuan untuk memberikan gambaran mengenai sistem yang akan dikembangkan berdasarkan kajian pustaka yang telah dibahas pada bab sebelumnya.

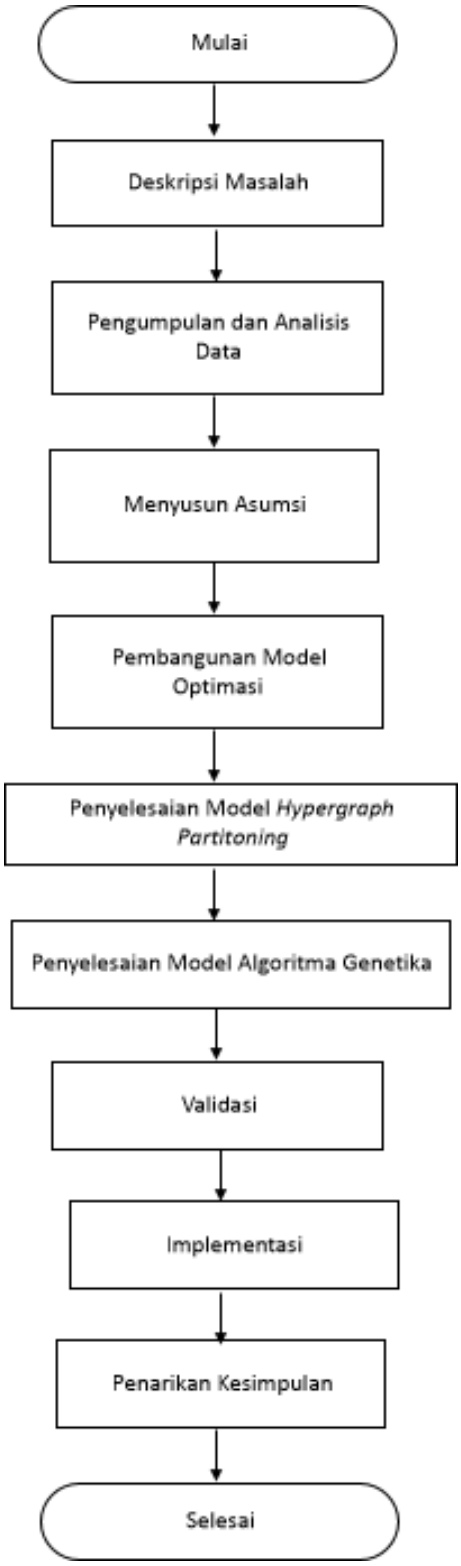
3.1. Deskripsi Masalah

Penelitian ini membahas penentuan jalur distribusi oleh dua kendaraan dengan mengaplikasikan metode *Hypergraph-Partitioning* dan Algoritma Genetika. Kendaraan yang akan mengirim barang ke sejumlah pelanggan memiliki batasan kapasitas. Untuk mengangkut barang, setiap kendaraan memulai perjalanannya dari sebuah tempat, yang disebut depot, lalu melakukan perjalanan untuk memenuhi sejumlah permintaan setiap pelanggan. Pada akhir perjalanan, setiap kendaraan harus kembali lagi ke depot.

Pengiriman barang untuk pelanggan dilakukan dengan mengantarkan permintaan langsung ke setiap pelanggan, sehingga setiap pelanggan akan dikunjungi oleh satu kendaraan pendistribusi tepat satu kali. Pembagian pengiriman untuk setiap kendaraan akan diselesaikan dengan menggunakan *Hypergraph-Partitioning*, selanjutnya metode optimasi untuk penentuan rute yang akan dilalui oleh setiap kendaraan akan diselesaikan dengan menggunakan Algoritma Genetika. Tujuan dari *Hypergraph-Partitioning* untuk memaksimumkan *cut size* dan menjaga keseimbangan kapasitas (*capacity balance*) sehingga diperoleh pembagian pelanggan untuk setiap kendaraan. Sedangkan TSP bertujuan untuk menentukan rute setiap kendaraan dari depot ke pelanggan dengan membawa muatan barang agar permintaan setiap pelanggan terpenuhi dan total jarak perjalanan yang dibutuhkan oleh setiap kendaraan adalah seminimum mungkin.

3.2. Tahapan Penelitian

Tahapan-tahapan dalam penelitian ini ditunjukan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.3. Jenis dan Sumber Data

Data yang akan digunakan dalam penelitian ini diambil dari pabrik roti yang terletak di Kabupaten Cianjur. Jenis data yang dikumpulkan mencakup beberapa aspek penting berikut:

1) Lokasi Pelanggan dan Jarak
    a) Data Lokasi Pelanggan: Informasi mengenai koordinat geografis atau alamat pelanggan yang menjadi titik distribusi.
    b) Jarak Antar Lokasi: Jarak yang harus ditempuh antara lokasi pabrik dan masing-masing pelanggan, serta jarak antar pelanggan. Data ini penting untuk menentukan rute distribusi yang efisien.
2) Jumlah Permintaan dari Masing-Masing Pelanggan
   Informasi tentang jumlah produk roti yang diminta oleh setiap pelanggan. Data ini membantu dalam merencanakan kapasitas kendaraan dan rute distribusi yang optimal.
3) Kendaraan dan Kapasitas
    a) Jumlah Kendaraan: Total jumlah kendaraan yang tersedia untuk proses distribusi.
    b) Kapasitas Kendaraan: Jumlah maksimum produk roti yang dapat diangkut oleh setiap kendaraan. Data ini digunakan untuk memastikan bahwa jumlah permintaan dari pelanggan dapat dipenuhi tanpa melebihi kapasitas kendaraan.

Dengan data tersebut, penelitian ini bertujuan untuk merancang sistem distribusi yang optimal, dengan mempertimbangkan faktor-faktor seperti jarak tempuh, permintaan pelanggan, dan kapasitas kendaraan.

3.4. Menyusun Asumsi

Penelitian ini memiliki beberapa asumsi dan batasan sebagai berikut:

1. Distribusi dilakukan dalam satu hari (*one day service)* dengan mengunjungi pelanggan tepat hanya satu kali oleh satu kendaraan.
2. Jalur yang dilewati oleh satu kendaraan dapat dilewati oleh kendaraan lainnya.
3. Kondisi jalan diasumsikan lancar pada dua arah.

4. Penelitian ini hanya mempertimbangkan jarak antara titik-titik pada graf. Waktu tempuh dan biaya operasional, seperti perawatan kendaraan, gaji kurir, dan biaya bahan bakar tidak dipertimbangkan
5. Simpul pada *hypergraph* mewakili lokasi pelanggan, sedangkan sisi menyatakan jalur yang dapat dilalui antar lokasi. Titik-titik yang berada pada *hyperedge* yang sama menunjukkan lokasi-lokasi yang terletak pada jalur searah tanpa adanya percabangan.
6. Jarak pulang adalah jarak terpendek yang bisa dilalui oleh kendaraan tersebut.
7. Hasil *partitioning* menyatakan pembagian pelanggan yang dikunjungi oleh masing-masing kendaraan yang tersedia.
8. Jumlah kendaraan yang tersedia adalah 2 kendaraan dengan kapasitas maksimal sebesar 1100 buah roti per kendaraan.

3.5. Model Optimasi TSP

Masalah TSP dapat dimodelkan sebagai model optimasi dengan sebuah fungsi tujuan dan sejumlah kendala. Tahapan pertama pemodelan adalah mendefinisikan himpunan, parameter, dan variabel keputusan yang digunakan dalam model ini. Misalkan I adalah himpunan pelanggan dan d_{ij} adalah jarak tempuh dari pelanggan i ke pelanggan j. Variabel keputusan x_{ij} menentukan apakah kendaraan melakukan perjalanan atau tidak dari pelanggan i ke pelanggan j, yang didefinisikan sebagai berikut.

$$Xij = \begin{cases} 1, & \text{jika kendaraan melakukan perjalanan dari } i \text{ ke } j \\ 0, & \text{untuk lainnya} \end{cases}$$

Fungsi tujuan dari model optimasi TSP adalah untuk mencari rute yang meminimumkan total jarak tempuh dalam pendistribusian barang ke pelanggan. Fungsi tersebut didefinisikan sebagai berikut.

Meminimumkan :

$$f = \sum_{i \in I} \sum_{j \in I} d_{ij} x_{ij}$$

Kendala dari model adalah batasan yang harus dipenuhi dalam menentukan rute yang dilalui. Kendala pertama adalah kendala yang menjamin bahwa hanya ada satu jalan yang masuk ke setiap lokasi. Kendala ini diekspresikan sebagai berikut.

$$\sum_{i \in I} x_{ij} = 1, \forall\, i \in I$$

Kendala kedua adalah kendala yang menjamin bahwa hanya ada satu jalan yang keluar dari setiap lokasi. Kendala ini diekspresikan sebagai berikut.

$$\sum_{j \in I} x_{ij} = 1, \forall\, j \in I$$

Kendala ketiga adalah kendala yang menjamin bahwa xij harus membentuk perjalanan, sehingga

$$\sum_{i \in S_t} \sum_{j \in S_t} x_{ij} \geq 1, \forall\, S_t \subset V$$

Dimana $S_t = V - S_t$, denganS_t adalah himpunan lokasi yang telah dikunjungi dan S_t adalah himpunan lokasi yang belum dikunjungi. Untuk selanjutnya, lokasi pelanggan tersebut akan disebut sebagai titik.

3.6. Pembangunan Penyelesaian Model

Langkah pertama dalam memperoleh solusi dari permasalahan jalur distribusi barang adalah memodelkan masalah dalam bentuk graf biasa. Simpul pada graf mewakili pelanggan, sedangkan sisi yang menghubungkan dua simpul mewakili jalur yang dapat dilewati antar titik tersebut. Model graf yang dibangun ini sesuai dengan matriks ajasensi yang ada. Setelah graf biasa terbentuk, graf tersebut direpresentasikan menjadi *hypergraph* dengan menghubungkan simpul-simpul yang saling berajasen oleh satu *hyperedge* yang sama.

Setelah permasalahan dimodelkan dalam bentuk *hypergraph*, selanjutnya dilakukan proses *Hypergraph-Partitioning* untuk membagi pelanggan ke dalam dua partisi, yang masing-masing partisi akan mewakili tujuan pengiriman oleh satu kendaraan. Partisi ini harus membagi pelanggan secara seimbang dan merata untuk masing-masing kendaraan.

3.6.1. Proses *Hypergraph-Partitioning*

3.6.1.1. Coarsening

Proses *Hypergraph-Partitioning* diawali dengan merepresentasikan masalah ke dalam *hypergraph* tanpa melibatkan titik 0 dan sisi yang berinsiden dengan titik 0. Kemudian, titik-titik yang berada pada *hyperedge* yang sama digabungkan menjadi *supernode* sehingga diperoleh *coarsened hypergraph*.

Selanjutnya, sepasang simpul yang bertetangga dikelompokkan setahap demi setahap hingga terbentuk dua partisi awal. Proses pengelompokkan ini dilakukan menggunakan Algoritma *Maxima Matching (Greedy)* dengan langkah-langkah sebagai berikut:

1. Bangun tabel ajasensi dengan bobot jarak antar simpul.
2. Pilih sisi dengan bobot terkecil (> 0) pada setiap baris
3. Kelompokkan simpul yang terhubung dengan sisi yang dipilih pada poin 2.
4. Masukkan simpul yang belum memiliki pasangan ke dalam kelompok yang memiliki simpul terdekat dengan simpul tersebut.
5. Bentuk kelompok menjadi 2 partisi dengan menggabungkan beberapa pasangan yang memiliki jarak terdekat ke dalam partisi yang sama.

Contoh proses *coarsening* disajikan pada bagian 3.7.

3.6.1.2. Initial Partition / Balancing

Tahap selanjutnya adalah tahap *balancing* dengan menggunakan algoritma FM, di mana setiap simpul yang telah berada pada partisi awal dievaluasi nilai gain-nya sehingga nilai gain bernilai non-negatif. Langkah-langkahnya sebagai berikut: (Swanjaya, 2015)

1. Inisiasi semua simpul dalam keadaan *unlocked*.
2. Beri label pada kedua partisi sebagai Partisi 1 dan Partisi 2.
3. Selama ada simpul yang *unlocked*, lakukan langkah-langkah berikut:
   b. Hitung nilai gain untuk setiap simpul yang *unlocked*.
      Nilai gain **(Δg)** dihitung menggunakan rumus berikut:

$$\Delta g = FS - TE$$

      Di mana *FS(v)* merupakan jumlah bobot dari sisi yang menghubungkan simpul *v* dengan simpul-simpul yang ada di luar partisi dimana simpul *v*

berada, dan *TE(v)* merupakan jumlah bobot dari *edge* yang menghubungkan simpul *v* dengan simpul-simpul yang ada di dalam partisi dimana simpul *v* berada.

c. Cari simpul yang memiliki nilai gain terkecil (< 0), misal v_i.

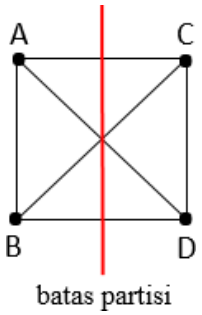
d. Pindahkan v_i ke partisi lainnya.

e. Atur v_i menjadi *locked* (tidak diizinkan untuk bergerak lagi pada iterasi berikutnya)

f. Hitung kembali nilai gain untuk setiap simpul setelah dilakukan perpindahan simpul

4. Lakukan poin 3 hingga semua simpul bernilai non-negatif atau setiap simpul telah terkunci (*locked*)

Sebagai contoh, diberikan *hypergraph* dengan 4 simpul pada Gambar 3.2 berikut.



Gambar 3.2 Contoh *Hypergraph* untuk Perhitungan Gain

Diketahui bobot dari setiap sisi pada *hypergraph* pada Gambar 3.2 disajikan pada Tabel 3.1 berikut.

Tabel 3.1 Contoh *Hypergraph* untuk Perhitungan Gain

	A	**B**	**C**	**D**
A	0	4	3	5
B		0	2	7
C			0	6
D				0

Misalkan *hypergraph* tersebut dibagi menjadi dua partisi yaitu $S_1 = \{A,B\}$ dan $S_2 = \{C,D\}$. Sehingga perhitungan gainnya disajikan pada tabel berikut:

Simpul	FS	TE	Δg
A	AC + AD = 8	AB = 4	4
B	BC + BD = 9	BA = 4	5
C	CA + CB = 5	CD = 6	-1
D	DA + DB = 12	DC = 6	6

Berdasarkan perhitungan gain di atas, maka simpul yang dipindahkan ke partisi lainnya adalah simpul C. Tetapi apabila tidak terdapat nilai gain yang lebih kecil dari nol, maka tidak dilakukan perpindahan, karena perpindahan yang dilakukan tidak akan berdampak, hal ini hanya berlaku pada perhitungan gain yang pertama (Swanjaya, 2015).

3.6.1.3. Uncoarsening

Tahap akhir dari proses *Hypergraph-Partitioning* adalah proses *uncoarsening*. Pada proses *uncoarsening*, *hypergraph* yang telah disederhanakan (*coarsened hypergraph*) dikembalikan ke bentuk aslinya (*original hypergraph*) sambil mengoptimasi kelompok-kelompok hasil proses *coarsening*, sehingga setiap partisi memiliki jumlah permintaan yang kurang dari kapasitas kendaraan yang ada.

3.6.2. Proses Algoritma Genetika

1) Populasi Awal

Tahap awal pada proses Algoritma Genetika adalah membangkitkan populasi awal secara acak sebanyak jumlah populasi yang ditentukan. Representasi kromosom yang digunakan berupa lintasan tujuan pelanggan yang dilayani oleh masing-masing kendaraan. Pada tahapan ini, ukuran populasi perlu ditentukan. Selanjutnya, populasi awal dibangkitkan secara acak sebanyak ukuran populasi yang ditentukan. Kromosom dibangkitkan dalam bentuk pemutasi dengan nilai acak yang berada pada rentang 1 hingga jumlah simpul.

2) Nilai *Fitness*

Setelah populasi awal dibangkitkan, dilakukan perhitungan nilai *fitness* untuk setiap individu dalam populasi sebelum memulai proses seleksi. Nilai *fitness* diperoleh dengan menghitung jarak total dari jalur yang terbentuk, yang dimulai dan diakhiri di titik 0. Jarak tersebut kemudian dimasukkan ke dalam rumus nilai *fitness* yang telah didefinisikan pada subbab 2.4.3.2.

3) Seleksi *Parent*

Sebelum proses *crossover* dan *mutation*, *parent* diseleksi dengan metode seleksi ranking (*rank selection)*. Langkah-langkah *rank selection* adalah sebagai berikut:

1. Berikan peringkat untuk setiap kromosom dalam populasi berdasarkan nilai *fitness*-nya. Kromosom dengan nilai *fitness* terkecil akan mendapatkan peringkat 1, kromosom kedua terkecil mendapatkan peringkat 2, dan seterusnya hingga kromosom terbaik mendapatkan peringkat n, di mana n adalah jumlah total kromosom dalam populasi.
2. Hitung probabilitas pemilihan untuk setiap individu berdasarkan peringkatnya. Probabilitas pemilihan untuk individu ke-i dihitung dengan membagi peringkat individu ke-i dengan jumlah semua peringkat individu dalam populasi.
3. Buat rentang probabilitas kumulatif untuk pemilihan. Rentang ini digunakan untuk memilih individu berdasarkan probabilitas yang telah dihitung. Rentang probabilitas kumulatif untuk individu ke-i diperoleh dengan menjumlahkan probabilitas pemilihan dari individu pertama hingga individu ke-i.
4. Pilih individu menggunakan bilangan acak. Dengan rentang probabilitas kumulatif yang telah ditentukan, gunakan bilangan acak dalam rentang [0, 1] untuk memilih individu. Bandingkan bilangan acak ini dengan rentang probabilitas kumulatif untuk menentukan individu yang terpilih.
5. Ulangi langkah pemilihan ini sesuai dengan jumlah individu yang diperlukan dalam generasi berikutnya

4) *Crossover*

Setelah populasi awal terbentuk, langkah selanjutnya adalah melakukan reproduksi dengan cara *crossover* untuk mendapatkan individu baru. Beberapa individu dalam populasi dipilih dengan metode *rank selection* sebagai *parent* yang akan menghasilkan anak sesuai tingkat *crossover*.

Dalam penelitian ini, digunakan *Partially-Mapped Crossover (PMX)* karena metode ini efektif dalam mencegah munculnya gen ganda pada keturunan, memastikan keberagaman, serta mampu mempertahankan urutan gen. Hal ini

menjadikannya sangat sesuai untuk diterapkan pada masalah TSP. Berikut merupakan langkah-langkah dalam metode PMX menurut Azmi, Jamaran, Arkeman, dan Mangunwidjaja (dalam Putri, 2014):

1. Tentukan dua titik secara acak pada kromosom untuk membagi *parent*. Di antara kedua titik tersebut, akan terdapat area pemetaan yang disebut dengan *mapping* seperti yang ditunjukkan pada gambar berikut.

Parent 1 :	1	2	3	4	5	6	7
Parent 2 :	4	3	7	2	6	1	5

Area pemetaan PMX

2. Tukar *substring* antara *parent1* dan *parent2* pada area *mapping* yang telah ditentukan. Proses ini akan menghasilkan dua ProtoChild seperti yang ditunjukkan pada gambar berikut.

ProtoChild 1 :	1	2	7	2	6	6	7
ProtoChild 2 :	4	3	3	4	5	1	5

Hasil ProtoChild PMX

3. Definisikan hubungan pemetaan dalam area *mapping*. Proses pemetaan ini dilakukan untuk mengganti nilai 7, 2, 6 menjadi 3, 4, 5 sesuai dengan posisi dan nilai pada area *mapping* seperti yang ditunjukkan pada gambar berikut.

7	2	6
↓↑	↓↑	↓↑
3	4	5

Pemetaan PMX

4. Lakukan pemetaan kromosom sesuai dengan ProtoChild1 dan ProtoChild2. Pertahankan *substring* yang berada dalam area pemetaan. Jika terdapat angka yang sama antara nilai di dalam dan di luar area pemetaan, gantilah angka yang berada di luar area pemetaan dengan nilai sesuai mengikuti hasil definisi hubungan pemetaan sebelumnya. Proses ini akan menghasilkan keturunan yang memiliki gen berbeda dengan *parent*, seperti yang ditunjukkan pada gambar berikut.

Child 1 :	1	4	7	2	6	5	3
Child 2 :	2	7	3	4	5	1	6

Hasil Keturunan PMX

5) Mutasi

Proses selanjutnya dalam reproduksi adalah proses mutasi. Dalam proses mutasi, individu dipilih secara acak dari populasi. Pada individu yang terpilih sebagai induk, dilakukan pertukaran gen di dalamnya untuk menghasilkan individu baru hasil mutasi (*offspring*).

Dalam penelitian ini, metode mutasi yang digunakan adalah *Reciprocal Exchange Mutation*. Metode ini melibatkan pemilihan individu sebagai *parent* secara acak, kemudian merubah susunan gen di dalamnya. Menurut Pratama dkk. (2020), perubahan dalam proses mutasi dilakukan dengan memilih dua posisi gen secara acak dalam kromosom. Nilai pada kedua posisi gen tersebut kemudian dipertukarkan, sehingga menghasilkan *offspring* yang memiliki gen yang berbeda dari *parent*-nya.

Individu 1 :	1	2	3	4	5	6	7
Individu hasil mutasi :	1	2	7	4	5	6	3

Contoh Mutasi

6) Elitism

Tahap akhir untuk mendapatkan individu terbaik adalah dengan menggunakan proses elitisme. Elitisme dilakukan pada semua individu, baik *parent* maupun *offspring* yang ada dalam populasi. Proses ini melibatkan pengurutan individu berdasarkan nilai *fitness*-nya, dan hanya individu dengan nilai *fitness* tertinggi yang akan terpilih untuk membentuk populasi generasi berikutnya, sesuai dengan ukuran populasi yang ditentukan (*pop-size*).

3.7. Contoh Kasus

Pada bagian ini, akan dibahas penerapan *Hypergraph-Partitioning* dan Algoritma Genetika pada sebuah contoh kasus untuk memperlihatkan perhitungan manual sekaligus memperjelas langkah-langkah dalam penyelesaian masalah distribusi barang menggunakan *Hypergraph-Partitioning* dan Algoritma Genetika.

Hasil dari kasus ini akan digunakan sebagai validasi terhadap program yang dibuat untuk data yang lebih besar, guna memastikan apakah hasil perhitungan secara manual sesuai dengan hasil perhitungan menggunakan program tersebut. Oleh karena itu, penerapan pada contoh kasus diperlukan untuk memastikan validitas hasil.

Langkah-langkah penyelesaian permasalahan distribusi barang menggunakan *Hypergraph-Partitioning* dan Algoritma Genetika diterapkan pada contoh kasus dengan 10 pelanggan. Data permintaan setiap pelanggan disajikan pada Tabel 3.2, sedangkan data jarak antar pelanggan disajikan pada Tabel 3.3 berikut.

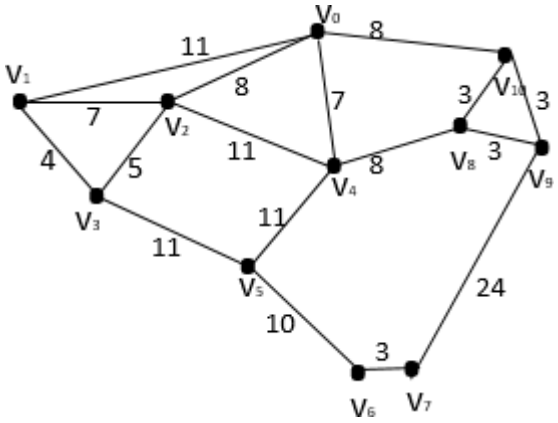
Tabel 3.2 Data Pelanggan Contoh Kasus

Pelanggan	**Permintaan (kg)**
1	15
2	17
3	19
4	20
5	24
6	20
7	21
8	25
9	15
10	20
Total Permintaan	196

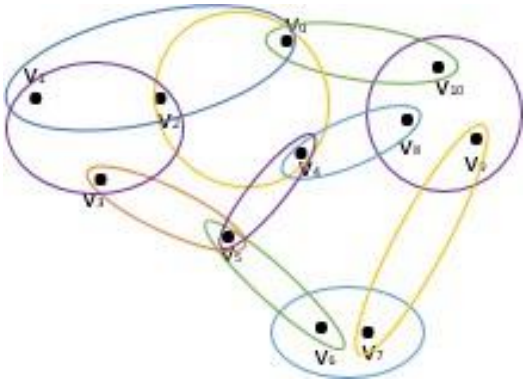
Tabel 3.3 Matriks Jarak Antar Pelanggan

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	11	8	-	7	-	-	-	-	-	8
1	11	0	7	4	-	-	-	-	-	-	-
2	8	7	0	5	11	-	-	-	-	-	-
3	-	4	5	0	-	11	-	-	-	-	-
4	7	-	11	-	0	11	-	-	8	-	-
5	-	-	-	11	11	0	10	-	-	-	-
6	-	-	-	-	-	10	0	3	-	-	-
7	-	-	-	-	-	-	3	0	-	24	-
8	-	-	-	-	8	-	-	-	0	3	3
9	-	-	-	-	-	-	-	24	3	0	3
10	8	-	-	-	-	-	-	-	3	3	0

Model graf dan *hypergraph* dari permasalahan ini ditunjukkan pada Gambar 3.3 dan Gambar 3.4 berikut.



Gambar 3. 3 Graf Representasi dari Kasus



Gambar 3. 4 *Hypergraph* Representasi dari Kasus

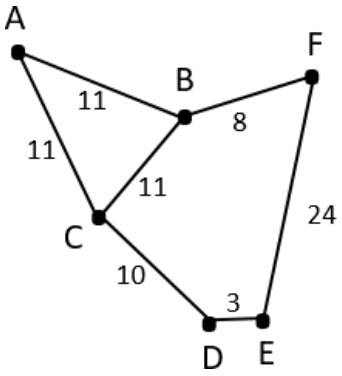
Pada Gambar 3.3 dan Gambar 3.4, titik awal pengiriman direpresentasikan sebagai v_0, dan pelanggan sebagai v_i , dengan i = 1, 2, ..., 10. Bobot pada graf merepresentasikan jarak pengiriman antar titik yang dihubungkan oleh sisi tersebut. *Hyperedge* menyatakan bahwa simpul di dalamnya saling berajasen. Pada model terlihat bahwa (v_1, v_2, $v_{3)}$ saling berajasen, begitu juga untuk (v_8, v_9, v_{10}), (v_0, v_1, v_2), dan (v_0, v_2, v_4).

3.7.1 Proses *Hypergraph-Partitioning*

Langkah awal dari penyelesaian masalah pendistribusian adalah membagi data pelanggan ke dalam dua partisi menggunakan metode *Hypergraph-Partitioning.*

1) Coarsening

Pada tahapan ini, *hypergraph* representasi dari kasus diubah menjadi bentuk *hypergraph* yang lebih sederhana (*coarsened hypergraph*) dengan menggabungkan titik-titik pada *hyperedge* yang sama menjadi satu *supernode* yang mewakili titik-titik tersebut. Hasil penyederhanaan *hypergraph* pada contoh kasus ini ditunjukkan pada Gambar 3.5 berikut.



Gambar 3.5 *Coarsened Hypergraph*

Gambar 3.5 menunjukkan graf yang lebih sederhana untuk merepresentasikan kasus, di mana titik-titik yang berada pada *hyperedge* yang sama diwakili oleh *supernode*. Titik A mewakili E1 = {1, 2, 3}, B = {4}, C = {5}, D = {6}, E = {7}, dan F = {8, 9, 10}.

Selanjutnya, Algoritma *Maxima Matching (Greedy)* dijalankan. Berikut langkah-langkah proses Algoritma *Maxima Matching (Greedy)* pada contoh kasus ini.

1. Bangun tabel ajasensi dengan bobot jarak antar simpul.

Matriks ajasensi untuk *coarsened hypergraph* pada Gambar 3.5 ditunjukkan pada Tabel 3.4 berikut.

Tabel 3.4 Matriks Ajasensi Tahap *Coarsening*

	A	**B**	**C**	**D**	**E**	**F**
A	0	11	11	-	-	-
B	11	0	11	-	-	8
C	11	11	0	10	-	-
D	-	-	10	0	3	-
E	-	-	-	3	0	24
F	-	8	-	-	24	0

2. Pilih sisi dengan bobot terkecil (> 0) pada setiap baris

Hal ini menunjukkan bahwa sisi tersebut merupakan jalur terpendek yang bisa dilalui dari simpul pada tiap baris ke simpul lainnya. Sisi yang terpilih pada tiap baris ditunjukkan pada Table 3.5 berikut.

Tabel 3.5 Sisi yang Terpilih pada Tiap Baris

	A	B	C	D	E	F
A	0	11	11	-	-	-
B	11	0	11	-	-	8
C	11	11	0	10	-	-
D	-	-	10	0	3	-
E	-	-	-	3	0	24
F	-	8	-	-	24	0

3. Kelompokkan simpul yang terhubung dengan sisi yang dipilih pada poin 2. Simpul yang berada pada kelompok yang sama dapat dilihat pada tabel, di mana sisi yang berseberangan terhadap diagonal matriks sama-sama terpilih sebagai sisi terpendek. Pasangan simpul yang terbentuk dari tahap ini ditunjukkan pada Tabel 3.6 berikut.

Tabel 3.6 Pasangan Awal yang Terbentuk dari Proses Algoritma *Maxima Matching*

	A	B	C	D	E	F
A	0	11	11	-	-	-
B	11	0	11	-	-	8
C	11	11	0	10	-	-
D	-	-	10	0	3	-
E	-	-	-	3	0	24
F	-	8	-	-	24	0

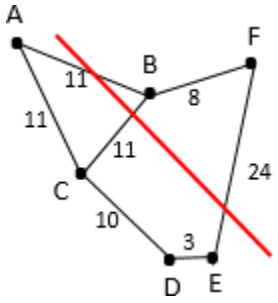
Dari Tabel 3.6 diperoleh kelompok : {B,F}, {D,E}, dengan simpul A dan C belum memiliki kelompok.

4. Masukkan simpul yang belum memiliki pasangan ke dalam kelompok yang memiliki simpul terdekat dengan simpul tersebut.

Dari Tabel 3.6, terlihat bahwa sisi terpendek yang terhubung ke simpul A adalah sisi yang terhubung dengan simpul B dan sisi yang terhubung dengan simpul C, di mana keduanya memiliki bobot yang sama yaitu 11. Sedangkan untuk simpul C memiliki jarak lebih dekat dengan simpul D dengan bobot 10. Kelompokkan simpul tersebut ke kelompok yang memiliki simpul terdekat dengannya. Sehingga diperoleh kelompok : {A,C,D,E}, {B,F}.

5. Membentuk kelompok menjadi 2 partisi

Karena kita telah memperoleh dua kelompok pada poin 4, maka partisi awal telah diperoleh. Partisi awal yang terbentuk ditunjukkan pada Gambar 3.6 berikut.



Gambar 3.6 Partisi Awal

2) Balancing

Pada proses *balancing*, nilai bobot penalti untuk sisi yang menghubungkan titik-titik yang tidak bertetangga diperlukan dalam perhitungan gain. Pada contoh kasus ini, digunakan nilai penalti sebesar 100. Bobot jarak antar pelanggan yang sudah diberikan bobot penalti yang akan digunakan untuk perhitungan nilai gain pada tahap *balancing* ini disajikan pada Tabel 3.7.

Tabel 3.7 Bobot Jarak Antar Pelanggan yang Sudah Diberikan Bobot Penalti

	1	2	3	4	5	6
1	0	11	11	100	100	100
2	11	0	11	100	100	8
3	11	11	0	10	100	100
4	100	100	10	0	3	100
5	100	100	100	3	0	24
6	100	8	100	100	24	0

Hasil perhitungan gain untuk simpul pada partisi awal ditunjukkan pada Tabel 3.8 berikut.

Tabel 3.8 Nilai Gain Setiap Simpul pada Partisi Awal

simpul	FS	TE	Δg
A	111	211	-100
B	222	8	214
C	111	121	-10
D	200	113	87
E	124	203	-79
F	324	8	316

Pada Tabel 3.8, terlihat bahwa nilai gain simpul A bernilai -100 yang merupakan nilai paling negatif. Oleh karena itu, simpul A akan dipindahkan ke partisi lain, sehingga partisi baru adalah {C, D, E} dan {A, B, F}. Selanjutnya, evaluasi nilai gain dilakukan kembali untuk setiap simpul dalam partisi baru. Hasil perhitungan gain untuk simpul pada partisi baru ditunjukkan pada Tabel 3.9 berikut..

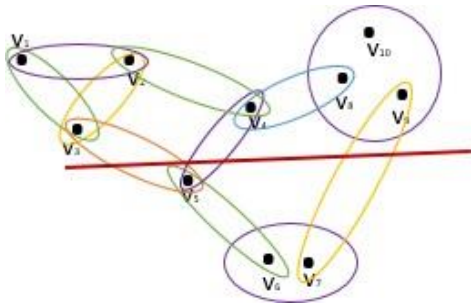
Tabel 3.9 Nilai Gain Setiap Simpul pada Partisi Baru

simpul	FS	TE	Δg
A	211	111	100
B	211	19	192
C	122	110	12
D	300	13	287
E	224	103	121
F	224	108	116

Pada tabel 3.9, terlihat bahwa nilai gain semua simpul telah bernilai non-negatif, sehingga tidak ada simpul yang perlu dipindahkan ke partisi lain dan proses selesai. Dengan demikian, partisi akhir yang diperoleh adalah {C, D, E} dan {A, B, F}.

3) Uncoarsening

Setelah proses balancing selesai, *coarsened hypergraph* dikembalikan ke bentuk awal. Hasil dari mengembalikan *coarsened hypergraph* tersebut kemudian menjadi *hypergraph* yang digunakan pada proses *uncoarsening*, di mana titik-titik yang sebelumnya diwakili oleh suatu *supernode* akan berada pada partisi yang sesuai dengan *supernode* yang sebelumnya mewakilinya. Partisi awal untuk proses *uncoarsening* ditunjukkan pada Gambar 3.7.



Gambar 3.7 Partisi Awal untuk Proses *Uncoarsening*

Selanjutnya, algoritma FM dijalankan sambil memperhatikan kesesuaian jumlah muatan pada tiap partisi. Hasil perhitungan iterasi pertama pada tahap *uncoarsening* disajikan pada Tabel 3.10.

Tabel 3.10 Perhitungan Jumlah Muatan dan Nilai Gain pada Partisi Awal Tahap *Uncoarsening*

P1

Simpul	Permintaan
1	15
2	17
3	19
4	20
8	25
9	15
10	20
Jumlah	131

P2

Simpul	Permintaan
5	24
6	20
7	21
Jumlah	65

simpul	FS	TE	Δg
1	300	411	-111
2	300	323	-23
3	211	409	-198
4	211	419	-208
5	522	110	412
6	700	13	687
7	624	103	521
8	300	314	-14
9	224	406	-182
10	300	406	-106

Pada Tabel 3.10, terlihat bahwa jumlah muatan pada kedua partisi belum baik, di mana diharapkan setiap partisi memiliki jumlah muatan kurang dari 100. Selain itu, masih terdapat nilai gain yang negatif. Oleh karena itu, titik 4 yang memiliki nilai gain terkecil dipindahkan ke partisi lainnya. Setelah itu, keseimbangan muatan dan nilai gain pada setiap titik diperiksa kembali. Hasil perhitungan jumlah muatan dan nilai gain untuk setiap titik setelah titik 4 dipindahkan disajikan pada Tabel 3.11 berikut.

Tabel 3.11 Perhitungan Jumlah Muatan dan Nilai Gain Iterasi Kedua Tahap *Uncoarsening*

P1

Simpul	Permintaan
1	15
2	17
3	19
8	25
9	15
10	20
6	111

P2

Simpul	Permintaan
4	20
5	24
6	20
7	21
4	85

Simpul	FS	TE	Δg
1	400	311	89
2	311	312	-1
3	311	309	2
4	419	211	208
5	511	121	390
6	600	113	487
7	524	203	321
8	308	306	2
9	324	306	18
10	400	306	94

Terlihat pada Tabel 3.11 bahwa nilai gain dari simpul 2 masih negatif, sehingga perlu dipindahkan ke partisi lain. Selain itu, jumlah muatan pada kedua partisi masih belum seimbang. Oleh karena itu, dilakukan perhitungan berikutnya dengan simpul 2 yang sudah dipindahkan partisinya. Hasil perhitungan jumlah muatan dan nilai gain untuk setiap titik setelah pemindahan titik 2 disajikan pada Tabel 3.12 berikut.

Tabel 3.12 Perhitungan Jumlah Muatan dan Nilai Gain Iterasi Ketiga pada Tahap *Uncoarsening*

P1

Simpul	Permintaan
1	15
3	19
8	25
9	15
10	20
5	94

P2

Simpul	Permintaan
2	17
4	20
5	24
6	20
7	21
4	85

Simpul	FS	TE	Δg
1	407	304	103
2	312	311	1
3	316	304	12
4	408	222	186
5	411	221	190
6	500	213	287
7	424	303	121
8	408	206	202
9	424	206	218
10	500	206	294

Karena semua titik telah memiliki nilai gain yang non-negatif dan jumlah muatan pada kedua partisi sudah cukup seimbang (<100), maka solusi akhir telah diperoleh. Partisi yang diperoleh adalah P1 = {1, 3, 8, 9, 10} dan P2 = {2, 4, 5, 6, 7}. Titik-titik pada partisi pertama (P1) akan dilayani oleh kendaraan pertama, sedangkan titik-titik pada partisi kedua (P2) akan dilayani oleh kendaraan kedua.

Setelah diperoleh dua partisi yang sudah seimbang, berikutnya Algoritma Genetika dijalankan pada masing-masing partisi untuk mencari lintasan yang optimal dari tiap kendaraan.

3.7.2 Algoritma Genetika

Pada bagian ini akan disajikan proses Algoritma Genetika pada partisi 2 hasil *Hypergraph-Partitioning* sebelumnya.

1) Populasi Awal

Misalkan ukuran populasi yang ditentukan adalah 5. Contoh populasi awal yang dapat dibangkitkan untuk kendaraan kedua dapat dilihat pada tabel 3.13 berikut:

Tabel 3.13 Contoh Populasi Awal

Parent	**Kromosom**				
P1	2	4	5	6	7
P2	7	6	4	2	5
P3	5	4	2	6	7
P4	2	5	4	6	7
P5	5	7	6	2	4

2) Nilai *Fitness*

Selanjutnya nilai *fitness* digitung dengan menjumlahkan jarak antar titik, di mana jarak dari titik terakhir ke titik 0 menggunakan jarak terpendek yang dapat dilalui. Hasil perhitungan nilai *fitness* ditunjukkan pada tabel 3.14 berikut:

Tabel 3.14 Nilai *Fitness* dari Kromosom pada Populasi Awal

Individu	Lintasan							Fitness
P1	0	2	4	5	6	7	0	0,013514
P2	0	7	6	4	2	5	0	0,005464
P3	0	5	4	2	6	7	0	0,003906
P4	0	2	5	4	6	7	0	0,005917
P5	0	5	7	6	2	4	0	0,003115

3) Seleksi *Parent*

Metode seleksi *rank selection* dilakukan dengan terlebih dahulu memberikan bobot ranking sesuai nilai *fitness* setiap kromosom. Kemudian dihitung probabilitas pemilihannya dengan membagi peringkat individu ke-i dengan jumlah semua peringkat individu dalam populasi. Jumlah semua peringkat individu dalam populasi = 5 + 4 + 3 + 2 + 1 = 15. Setelah itu, dibuat rentang probabilitas kumulatif berdasarkan probabilitas dari masing-masing kromosom. Rentang probabilitas kumulatif untuk individu ke-i diperoleh dengan menjumlahkan probabilitas pemilihan dari individu pertama hingga individu ke-i. Hasil perhitungan bobot rank hingga probabilitas kumulatif dari setiap kromosom pada populasi disajikan pada Tabel 3.15 berikut:

Tabel 3.15 Contoh Perhitungan Probabilitas Kumulatif untuk Seleksi *Parent*

Kromosom	Nilai Fitness	Bobot Rank	Probabilitas	Probabilitas Kumulatif
P1	0,013513514	5	0,333333333	0 - 0,3333
P4	0,00591716	4	0,266666667	0,3333 - 0,6
P2	0,005464481	3	0,2	0,6 - 0,8
P3	0,00390625	2	0,133333333	0,8 - 0,933
P5	0,003115265	1	0,066666667	0,933 - 1

Kemudian, pilih individu menggunakan bilangan acak. Dengan rentang probabilitas kumulatif yang telah ditentukan, gunakan bilangan acak dalam rentang [0, 1] untuk memilih individu. Bandingkan bilangan acak ini dengan rentang probabilitas kumulatif untuk menentukan individu yang terpilih. Misalkan diperoleh bilangan acak 0,3 dan 0,95, maka kromosom yang terpilih sebagai *parent* pada proses reproduksi adalah P1 dan P5.

4) Crossover

Pada contoh ini, digunakan Cr = 0,4 untuk proses *crossover*. Karena jumlah anak yang dihasilkan $offspring = Cr \times popSize = 0{,}4 \times 5 = 2$, maka PMX hanya dilakukan satu kali yang akan menghasilkan 2 *offspring*. Misalkan P1 dan P5 terpilih sebagai *parent*. Kemudian, terpilih titik 1 dan 3 sebagai titik *crossover*. Berikut adalah hasil *crossover* dari P1 dan P5:

P1	2	4	5	6	7
P5	5	7	6	2	4

Child:

C1	5	7	6	2	4
C2	2	4	5	6	7

5) Mutasi

Dalam penelitian ini, digunakan metode *Reciprocal Exchange Mutation* dengan nilai *mutation rate* yang dipilih adalah Mr = 0,4. Karena jumlah anak yang dihasilkan adalah $offspring = Mr \times popSize = 0{,}4 \times 5 = 2$, maka proses mutasi hanya dilakukan pada dua *parent* yang akan menghasilkan 2 *offspring*. Misalkan P4 dan P2 terpilih sebagai *parent*. Kemudian, terpilih titik 1 dan 2 sebagai titik mutasi. Berikut hasil mutasi pada P4 dan P2:

P4	2	5	4	6	7
C3	5	2	4	6	7

P2	7	6	4	2	5
C4	5	6	4	2	7

6) Elitism

Elitism dilakukan dengan mengurutkan individu dengan nilai *fitness* terbesar sebanyak jumlah elit yang diinginkan, kemudian individu-individu terbaik ini akan dipilih sebagai individu terbaik untuk generasi berikutnya. Hasil pengurutan nilai *fitness* dari yang terbesar pada populasi ditunjukkan pada Tabel 3.16 berikut.

Tabel 3.16 Hasil *Elitism*

Individu	Lintasan							Fitness
P1	0	2	4	5	6	7	0	0,013514
C2	0	2	4	5	6	7	0	0,013514
P4	0	2	5	4	6	7	0	0,005917
P2	0	7	6	4	2	5	0	0,005464
P3	0	5	4	2	6	7	0	0,003906
C3	0	5	2	4	6	7	0	0,003831
P5	0	5	7	6	2	4	0	0,003115
C1	0	5	7	6	2	4	0	0,003115
C4	0	5	6	4	2	7	0	0,002841

Misalkan jumlah elit yang ditetapkan adalah 5, individu yang dipilih untuk generasi berikutnya adalah 5 individu dengan nilai *fitness* terbesar, yaitu P1, C2, P4, P2, dan P3. Tahap *crossover* kemudian dilakukan kembali pada generasi baru hingga mendapatkan generasi baru lagi. Algoritma berhenti setelah jumlah generasi yang telah ditentukan terpenuhi. Proses yang sama dilakukan pada partisi lainnya. Hasil yang diperoleh dari proses Algoritma Genetika menunjukkan rute pengiriman optimal yang perlu dilakukan oleh kedua kendaraan.

3.8. Validasi

Pada tahap validasi, dilakukan pemeriksaan untuk memastikan bahwa logika model yang dibuat sudah benar dan dapat diimplementasikan dengan baik. Penyelesaian masalah distribusi barang di pabrik roti melibatkan banyak simpul, sehingga akan dikonstruksi program dengan bahasa pemrograman p*ython* untuk proses algoritmanya. Validasi dilakukan untuk pengecekan apakah model tersebut mampu merepresentasikan sistem nyata dengan baik. Ini dilakukan dengan menguji apakah *output* model komputer sesuai dengan hasil perhitungan manual pada studi kasus kecil. Proses ini memastikan bahwa model bekerja sesuai dengan ekspektasi dan dapat digunakan untuk menganalisis skenario yang lebih besar dan kompleks dengan keyakinan bahwa hasilnya akan tetap valid.