BAB III
METODE PENELITIAN

Bab ini membahas tentang metodologi yang akan digunakan untuk mencari rute pendisribusian dengan mengaplikasikan penyelesaian *Clustering Traveling Salesman Problem* (CTSP). Tahapan penyelesaian dimulai dengan mendeskripsikan masalah, lalu mengkonstruksi model masalah optimisasi dengan mendefinisikan variabel dan parameter yang digunakan, dan diakhiri dengan teknik penyelesaian dengan menggunakan algoritma *Lexiseacrh*.

3.1. Deskripsi Masalah

Penelitian ini akan menyelesaikan *Clustering Traveling Salesman Problem* (CTSP) dengan menggunakan algoritma *Lexisearch*. CTSP termasuk ke dalam masalah optimisasi yang merupakan perluasan dari TSP, di mana pada CTSP lokasi-lokasi wilayah diklasterkan berdasarkan jarak kedekatan antar klaster. Seperti halnya pada TSP, pada CTSP seorang *salesman* berangkat dari lokasi awal, kemudian mengunjungi setiap lokasi hingga berakhir kembali ke lokasi awal, namun terdapat aturan bahwa setiap lokasi pada suatu klaster harus terkunjungi sebelum beralih ke klaser yang lain. Pada penyelesaian CTSP ini, akan dihasilkan rute optimal. Rute optimal yang dimaksud adalah rute yang dapat meminimumkan waktu tempuh.

Pada CTSP, lokasi-lokasi diklasterkan dengan memperhatikan kedekatan antar klaster. Rute dicari dengan memperhatikan aturan bahwa setiap lokasi pada setiap klaster harus dikunjungi secara berdekatan, atau dalam kata lain, setiap lokasi pada suatu klaster harus dikunjungi sebelum beralih mengunjungi klaster selanjutnya. Pada penelitian ini, akan dicari suatu rute pendisribusian sembako yang oprimal dengan mengaplikasikan penyelesaian *Clustering Traveling Salesman Problem* (CTSP). Setiap lokasi pendistribusian diklasterkan dan seorang *salesman* harus mengunjungi setiap lokasi pendistribusian yang dimulai dari lokasi awal kemudian beralih

mengunjungi setiap lokasi pada setiap klaster secara berdekatan hingga berakhir kembali ke lokasi awal pendistribusian.

3.2. Tahapan Penelitian

Tahapan penelitian yang dilakukan adalah sebagai berikut:

1. Studi Pustaka

   Penelitian ini dilakukan dengan metode studi pustaka. Sumber dari penelitian ini diperoleh dengan mempelajari topik yang berkaitan dengan CTSP dan algoritma *Lexisearch* dari buku, jurnal, dan penelitian lainnya yang berkenaan dengan topik yang akan diteliti.

2. Pengumpulan Data

   Pengumpulan data dilakukan dengan metode wawancara. Data yang dikumpulkan pada penelitian ini terdiri dari:

   a. Data lokasi

      Data lokasi terdiri dari dua macam data, yaitu data lokasi awal (depot) dan data lokasi pendistribusian yang harus dikunjungi.

   b. Data jarak antar lokasi

      Data jarak antar lokasi terdiri dari jarak antar lokasi pendistribusian dan data jarak antara lokasi awal dengan setiap lokasi pendistribusian.

   c. Data klaser lokasi

      Data ini mencakup jumlah klaster pada lokasi pendistribusian dan juga data lokasi pada setiap klaster.

3. Pengkonstruksian Model

   Pada tahapan ini, akan dikonstruksi model matematika untuk mengoptimalkan rute pendistribusian dengan menetapkan asumsi-asumsi yang akan digunakan dan juga mendefinisikan variabel dan parameter yang akan digunakan.

4. Penyelesaian Model

   Pada tahapan ini, akan dijelaskan mengenai teknik penyelesaian yang akan digunakan untuk mengoptimalkan rute pendistribusian dengan menggunakan Algoritma *Lexisearch*.

5. Validasi

   Validasi dilakukan untuk menguji apakah model dan teknik penyelesaiannya sudah benar atau tidak. Validasi dilakukan dengan cara membandingkan solusi hasil simulasi program dengan solusi hasil perhitungan kasus kecil yang sudah diketahui. Jika solusi keduanya sama, maka model dan teknik penyelesiannya telah valid dan akan dilanjutkan ke tahapan implementasi. Jika hasil perbadingan tidak sama, maka tahapan akan diulang dari tahapan pemodelan.

6. Implementasi

   Pada tahapan ini, Algoritma *Lexisearch* ini akan diimpementasikan untuk penyelesaian pencarian rute pendisribusian sembako dari suatu perusahaan di Kota Bandung dengan mengaplikasikan penyelesaian *Clustering Traveling Salesman Problem* (CTSP).

7. Analisis kinerja algoritma *Lexisearch*

   Pada tahapan ini kinerja algoritma *Lexisearch* akan dianalisis dengan cara menguji algoritma tersebut dalam menyelesaikan sejumlah kasus CTSP dengan jumlah lokasi yang berbeda-beda. Kinerja algoritma akan diukur berdasarkan solusi optimalnya beserta waktu komputasi.

8. Penarikan Kesimpulan

   Pada tahapan ini akan dilakukan penarikan kesimpulan berdasar hasil implementasi.

3.3. Model Optimisasi

Pada bagian ini akan dijelaskan mengenai bagaimana cara mengkonstruksi model matematika untuk mengoptimalkan rute pendistribusian. Asumsi-asumsi yang digunakan dalam pemodelan adalah sebagai berikut:

1. Setiap lokasi pendistribusian dilalui tepat satu kali.
2. Hanya terdapat satu depot sebagai lokasi Lokasi awal sekaligus lokasi akhir.

3. Setiap jalan yang dilewati ada dalam keadaan normal, di mana tidak terjadinya kemacetan maupun perbaikan jalan.
4. Terdapat hanya satu kendaraan yang digunakan salesman dan dianggap mampu untuk mengangkut barang pada saat pendistribusian.

Tahapan pertama dalam pemodelan adalah mendefinisikan variabel dan parameter yang akan digunakan. Setiap lokasi pendistribusian barang didefinisikan sebagai himpunan verteks V. Parameter yang digunakan adalah c_{ij}, yaitu jarak antara lokasi pendistribusian i ke lokasi pendistribusian j, Misalkan terdapat n lokasi pendistribusian yang harus dikunjungi, dan *m* klaster. Variabel keputusan pada model CTSP didefinisikan untuk menentukan ada atau tidaknya perjalanan seorang *salesman* yang dimulai dari lokasi awal, ke lokasi pendirstribusian i hingga lokasi pendistribusian j. Variabel keputusan ini dinyatakan sebagai dengan x_{ij} dimana x_{ij} akan bernilai 1 jika lokasi pendistribusian ke j dikunjungi tepat setelah lokasi pendistribusian ke i dikunjungi dan akan bernilai 0 jika lokasi pendistribusian ke j tidak dikunjungi tepat setelah lokasi pendistribusian ke 1 dikunjungi. Dalam artian yang sama, x_{ij} akan bernilai i jika terdapat perjalanan seorang *salesman* dari lokasi pendistribusian ke i ke lokasi pendistribusian ke j, dan akan bernilai 0 jika tidak terdapat perjalanan seorang *salesman* dari lokasi pendistribusian ke i ke lokasi pendistribusian ke j. Dengan demikian, diperoleh variabel keputusan:

$$x_{ij} = \begin{cases} 1, \text{jika salesman melakukan perjalanan dari } i \text{ ke } j \\ 0, \quad \text{yang lainnya.} \end{cases}$$

Tujuan dari penyelesaian CTSP adalah untuk menentukan rute pendistribusian dengan total jarak minimum. Berdasarkan penjabaran tersebut, maka fungsi tujuan model dapat dituliskan dengan:

Meminimumkan:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Adapun kendala-kendala dari model adalah sebagai berikut:

1. Setiap lokasi pendistribusian dikunjungi tepat satu kali

   Kendala ini dituliskan dalam persamaan berikut:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall i \in V$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall j \in V$$

2. Setiap lokasi pendistribusian pada setiap klaster dikunjungi secara berdekatan, maka terdapat kendala

$$\sum_{i \in V_k} \sum_{j \in V_k} x_{ij} = |V_k| - 1 \qquad \forall V_k \in V, |V_k| > 1, k = 1,2, \dots, m$$

dengan V_k merupakan banyaknya simpul pada suatu klaster k.

3. Kendala sub-tour

Untuk mengeliminasi sub-tour, dalam artian bahwa rute perjalanan *salesman* terhubung atau tidak ada sub-tour yang tidak terhubung. Dalam kendala ini digunakan formula yang menambahkan variabel tambahan u_i dan u_j yang menyatakan nilai simpul (lokasi pendistribusian) jika salesman melakukan perjalanan dari simpul (lokasi pendistribusian) i ke simpul (lokasi pendistribusian) j. Kendala ini dinyatakan dengan:

$$u_i - u_j + (n-1) \, . \, x_{ij} \leq n - 2 \qquad 2 \leq i \neq j \leq n$$

Batasan dari variabel model adalah adalah bahwa variabel keputusan x_{ij} bernilai biner, yaitu

$$x_{ij} \in \{0,1\}, i \neq j, i, j \in n$$

Selengkapnya model CTSP dapat dinyatakan dalam model optimasi berikut:

Meminimumkan:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

terhadap:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall i \in V$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall j \in V$$

$$u_i - u_j + (n-1)\,.\,x_{ij} \leq n-2 \qquad 2 \leq i \neq j \leq n$$

$$\sum_{i \in V_k} \sum_{j \in V_k} x_{ij} = |V_k| - 1 \qquad \forall V_k \in V, |V_k| > 1, k = 1,2,\dots,m$$

$$x_{ij} \in \{0,1\}, i \neq j, i, j \in n$$

Model optimasi di atas termasuk model optimasi integer linear programming. Bagian selanjutnya akan dibahas mengenai teknik penyelesaian CTSP dengan menggunakan algoritma *Lexisearch*.

3.4. Teknik Penyelesaian

Penelitian ini, akan menggunakan Algoritma *Lexisearch* untuk menyelesaikan CTSP. Algoritma *Lexisearch* merupakan algoritma yang mencari solusi optimal dengan cara yang sistematis (Siti, 2018). Ide dasar dari algoritma *Lexisearch* adalah seperti halnya mencari suatu kata dalam kamus yang memperhatikan urutan alfabet, algoritma ini bekerja dengan mencari solusi penyelesaian dengan memperhatikan tabel alfabet. Tahapan penyelesaian dimulai dengan menghitung matriks jarak yang telah dimodifikasi. Setelah memperoleh tabel matriks jarak yang dimodifikasi, maka selanjutnya adalah dengan menghapuskan bias pada matriks jarak yang dimodifikasi. Setelah itu, konstruksi tabel alfabet yang digunakan untuk pencarian rute optimal. Rute yang dihasilkan pada setiap iterasi tidak boleh melanggar aturan klaster dan tidak membentuk sub-tour. Cara kerja Algoritma *Lexisearch* dapat dilihat pada Gambar 3.1. Seperti halnya dengan permasalahan TSP, pada CTSP juga dibutuhkan matriks jarak. Dalam penyelesaian CTSP dengan menggunakan algoritma *Lexisearch*, terdapat modifikasi pada matriks jarak. Modifikasi pada matriks jarak ini dimaksudkan untuk menjaga aturan klaster, yaitu setiap lokasi pada suatu klaster harus terkunjungi terlebih dahulu sebelum beranjak ke klaster selanjutnya. Matriks jarak dimodifikasi dengan menetapkan jarak yang cukup besar (misalkan M) untuk sisi-sisi tertentu dalam matriks jarak tersebut. Misalkan $C_{ij} = [c_{ij}]$ merupakan matriks jarak berukuran $n\ x\ n$, dan c_{ij}

merupakan jarak dari lokasi pendistribusian i ke lokasi pendistribusian j dengan v_1 merupakan lokasi awal. Misalkan juga setiap lokasi pendistribusian (kecuali v_1) dikelompokkan menjadi m klaster dengan jumlah lokasi pada setiap klaster masing masing $n_1, n_2, \dots, n_m$. Ahmed (2013) menjelaskan bahwa pada modifikasi matriks jarak, matriks awal diperoleh dengan memasukkan nilai jarak sebesar mungkin (misalkan M) dari setiap sisi (i, j) kecuali sisi-sisi berikut ini:

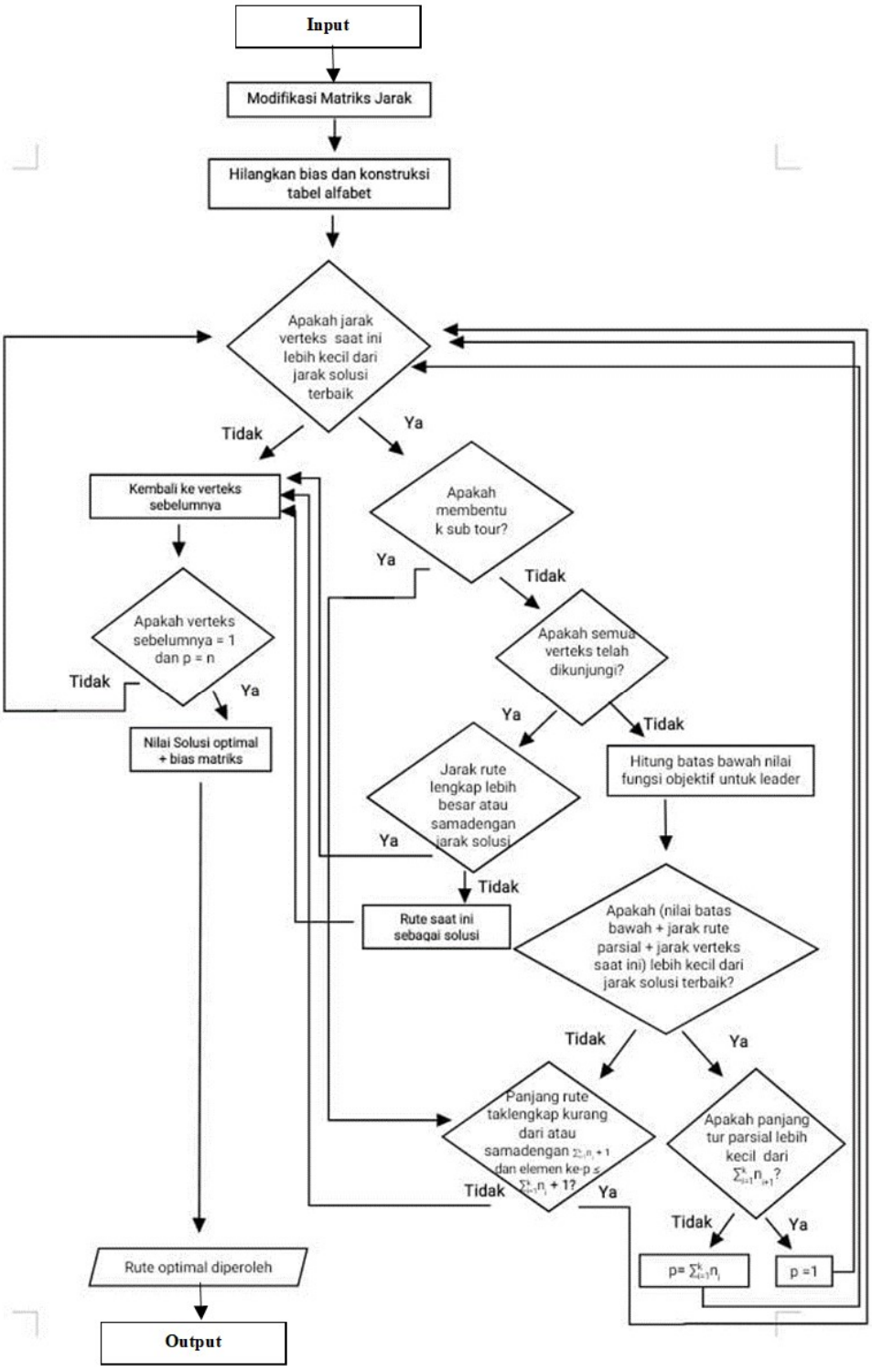
i. Sisi $(1, j)$, untuk $j = 2,3, \dots, n_1 + 1$;
ii. Sisi (i, j), untuk $i = (n_1 + n_2 + \cdots + n_{m-1}) + 2, \dots, n$;
iii. Sisi (i, j), untuk $i \neq j$, dengan:
    e. $i = 2,3, \dots, n_1 + 1;\ j = 2,3, \dots, n_1 + n_2 + 1;$
    f. $i = n_1 + 2, \dots, n_1 + n_2 + 1;\ j = n_1 + 2, \dots, n_1 + n_2 + n_3 + 1;$
    g. $i = n_1 + n_2 + 2, \dots, n_1 + n_2 + n_3 + 1;\ j = n_1 + n_2 + 2, \dots, n_1 + n_2 + n_3 + n_4 + 1;$ sampai dengan
    h. $i = (n_1 + n_2 + \cdots + n_{m-1}) + 2, \dots, n;\ j = (n_1 + n_2 + \cdots + n_{m-1}) + 2, \dots, n.$

Setelah memperoleh matriks jarak yang dimodifikasi, maka langkah selanjutnya adalah mencari bias dari matriks.

Misalkan diberikan himpunan verteks V = {1, 2, … ,5}, dengan klaster $V_1 = \{2,3\}$ dan $V_2 = \{4,5\}$, dengan verteks ke-1 adalah lokasi awal pendistribusian. Ini berarti bahwa banyaknya $n_1 = 2$ dan banyaknya $n_2 = 2$. Pada tahap modifikasi matriks jarak ini, misalkan diberikan matriks $C_{ij} = [c_{ij}]$ merupakan matriks jarak yang dimodifikasi berukuran $n \times n$ dengan c_{ij} merupakan jarak dari simpul i ke simpul j. Diketahui bahwa $n_1 = 2$ dan $n_2 = 2$ Selanjutnya dilakukan modifikasi pada matriks jarak pada setiap sisi (i, j) diberikan nilai sebesar besarnya (misalkan 999) kecuali pada sisi berikut:

1. Sisi $(1, j)$, untuk $j = 2, 3$, yaitu sisi $(1,2)$ dan sisi $(1,3)$
2. Sisi $(i, 1)$, untuk $i = 4, 5$, yaitu sisi $(4,1)$ dan sisi $(5,1)$
3. Sisi (i, j), untuk $i \neq j$ dan
    a. $i = 2, 3;\ j = 2, 3, 4, 5;$
    b. $i = 4, 5;\ j = 4, 5;$

sehingga diperoleh matriks jarak yang dimodifikasi seperti pada Tabel 3.1.



Gambar 3. 1 Langkah-Langkah Algoritma Lexisearch.

Tabel 3. 1 Matriks jarak awal yang dimodifikasi.

Vertex	1	2	3	4	5
1	999	103	106	999	999
2	999	999	21	47	37
3	999	16	999	31	21
4	131	999	999	999	8
5	122	999	999	10	999

Misalkan u_i merupakan elemen terkecil pada baris ke-i dan v_j merupakan eleman terkecil pada kolom ke-j pada Tabel 3.1. Cari elemen terkecil dari setiap baris pada matriks jarak yang telah dimodifikasi. Maka diperoleh bahwa elemen terkecil dari baris pertama adalah 103, elemen terkecil dari baris kedua adalah 21, elemen terkecil dari baris ketiga adalah 16, elemen terkecil dari baris keempat adalah 8, dan elemen terkecil dari baris kelima adalah 10. Sehingga diperoleh $\sum_{i=1}^{n} u_i = 103 + 21 + 16 + 8 + 10 = 158$. Kemudian elemen pada setiap baris dikurangi dengan elemen terkecil pada setiap baris, sehingga diperoleh nilai pada Tabel 3.2.

Tabel 3. 2 Update 1 Matriks Jarak.

Vertex	1	2	3	4	5	Minimal Baris
1	896	0	3	896	896	103
2	978	978	0	26	16	21
3	983	0	983	15	5	16
4	123	991	991	991	0	8
5	112	989	989	0	989	10

Langkah selanjutnya adalah mencari elemen terkecil dari setiap kolom. Diperoleh bahwa elemen terkecil dari kolom pertama adalah 112, elemen terkecil dari kolom kedua adalah 0, elemen terkecil dari kolom ketiga adalah 0, elemen terkecil dari kolom keempat adalah 0, dan elemen terkecil dari kolom kelima adalah 0, sehingga diperoleh $\sum_{j=1}^{n} v_j = 112 + 0 + 0 + 0 + 0 = 112$. Kemudian elemen pada setiap

kolom dikurangi dengan elemen minimal pada setiap kolom, sehingga diperoleh Tabel 3.3.

Tabel 3. 3 Update 2 Matriks Jarak.

Vertex	1	2	3	4	5	Minimal baris
1	884	0	3	896	896	103
2	866	978	0	26	16	21
3	871	0	983	15	5	16
4	11	991	991	991	0	8
5	0	989	989	0	989	10
Minimal kolom	112	0	0	0	0	

bias dari matriks adalah penjumlahan dari setiap elemen terkecil pada setiap baris dengan setiap elemen terkecil pada setiap kolom. Dengan demikian diperoleh Bias Matriks:

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j = 158 + 112 = 270 \ldots\ldots\ldots\ldots \quad (3.1)$$

Setelah diperoleh matriks jarak yang dimodifikasi, maka akan dikonstruksi tabel alfabet dengan menghapus bias pada matriks jarak kemudian membuat urutan secara tak turun berdasarkan nilai jarak pada setiap baris i dan diidentiifkasi verteks j yang bersesuaian dengan jarak c_{ij}. Matriks pada Tabel 3.3, setelah diurutkan maka diperoleh baris ke-1 sebagai berikut:$c_{12} = 0$,$c_{13} = 3$, $c_{11} = 884$, $c_{14} = 896$, $c_{15} = 896$. Dengan cara yang sama untuk baris-baris yang lainnya, sehingga diperoleh Tabel Alfabet pada Tabel 3.4.

Tabel 3. 4 Tabel Alfabet.

	$v - jarak$	$v - jarak$	$v - jarak$	$v - jarak$	$v - jarak$
1	2-0	3-3	1-884	4-896	5-896
2	3-0	5-16	4-26	1-866	2-978
3	2-0	5-5	4-15	1-871	3-983
4	5-0	1-11	2-991	3-991	4-991
5	1-0	4-0	2-989	3-989	5-989

Proses pencarian rute optimal dimulai dengan memperhatikan baris pertama pada tabel alfabet. Selanjutnya rute dicari dengan menelusuri setiap jalur atau rute ke verteks (lokasi pendisribusian lain) lainnya yang berdekatan dengan menelusuri setiap klasternya hingga kembali lagi ke lokasi awal dan seluruh lokasi pendisribusian telah terkunjungi. Rute ini diperoleh dengan menerapkan algoritma *Lexisearch* berikut:

Langkah 0: Hilangkan bias dari matriks jarak dan mengkonstruksi tabel alfabet dengan acuan pada matriks jarak yang telah dimodifikasi. Nilai dari jarak solusi terbaik diberi nilai sebesar besarnya (misalkan M). Perhitungan dimulai dari baris pertama pada tabel alfabet. Masukan nilai awal "rute parsial" = 0 dan $p = 1$.

Langkah 1: Cek elemen ke-p pada baris matriks (misalkan verteks v) dengan jarak verteks sesuai dengan "jarak verteks saat ini". Jika jarak verteks saat ini kurang dari nilai solusi terbaik, maka dilanjutkan ke Langkah 2, jika tidak dilanjutkan ke Langkah 9. Sebagai contoh, untuk iterasi pertama, cek elemen ke-p, yaitu elemen pertama pada baris pertama pada Table Alfabet 3.4. Elemen pertama untuk baris pertama adalah 2, dengan besarnya jarak antara verteks 1 dan 2 adalah 0, sehingga: jarak verteks saat ini = 0. Pada contoh modifikasi matriks jarak di Tabel 3.1, diberikan bahwa nilai M = 999, sehingga jarak verteks saat ini = 0 < M dan langkah berlanjut ke Langkah 2.

Langkah 2: Cek apakah verteks v membentuk suatu sub-*tour* atau melanggar aturan klister atau tidak. Jika ya, maka verteks v tidak akan digunakan dan nilai $p = p + 1$, kemudian beralih ke Langkah 7. Jika tidak, berlanjut ke Langkah 3. Sebagai contoh, verteks 2 tidak membentuk suatu sub-*tour*, dan juga tidak melanggar aturan klaster sehingga berlanjut ke Langkah 3.

Langkah 3: Cek apakah semua verteks telah dikunjungi. Jika ya, tambahkan *edge*(sisi) untuk menghubungkan v ke v_1 dan lanjutkan ke langkah 4, jika tidak, dilanjutkan ke langkah 5. Sebagai contoh, pada iterasi pertama belum semua verteks terkunjungi, sehingga berlanjut ke Langkah 5.

Langkah 4: Jika jarak rute lengkap lebih dari atau sama dengan jarak solusi terbaik, maka dilanjut ke tahap 9, jika tidak, maka jarak rute saat ini diganti sebagai solusi terbaik dan lanjut ke langkah 9. Langkah ini mengecek apakah solusi rute lengkap yang

dihasilkan dari suatu iterasi lebih optimal atau tidak dibandingkan solusi terbaik sebelumnya.

Langkah 5: Hitung nilai dari batas bawah pada *leader* (rute) saat ini.

Misalkan terdapat rute parsial B $= (v_1, a_1, .., a_2)$ dan verteks v terpilih untuk dirangkaikan ke dalam rute B, maka harus dihitung terlebih dahulu batas bawah dari *leader* $(v_1, a_1, \dots, a_2, v)$. Untuk itu, perhitungan dari nilai batas bawah dimulai dari baris ke-2 dari tabel alfabet, sampai dengan baris ke n dengan menjumlahkan nilai jarak untuk verteks yang tidak termasuk dalam rute parsial B dan juga tidak termasuk baris a_1,.., a_2 , dan termasuk verteks awal (v_1) . Nilai jumlah yang telah diperoleh merupakan nilai dari batas bawah untuk *leader* $(v_1, a_1, \dots, a_2, v)$. Lanjutkan ke langkah 6. Misalkan sebelumnya diperoleh bahwa *leader* adalah (1,2), dengan mengacu pada Tabel Alfabet 3.4, maka batas bawah untuk *leader* ini adalah:

Batas bawah= $c_{2,\alpha(2,1)} + c_{3,\alpha(3,1)} + c_{4,\alpha(4,1)} + c_{5,\alpha(5,1)}$

$= c_{2,3} + c_{3,2} + c_{4,5} + c_{5,1}$

$= 0 + 0 + 0 + 0 + 0$

$= 0$

Langkah 6: Jika nilai dari (batas bawah+jarak rute parsial+jarak verteks sekarang) lebih besar atau sama dengan solusi terbaik, maka verteks v diabaikan, nilai p bertambah 1, dan dilanjutkan ke Langkah 7. Jika tidak, maka verteks v dimasukkan ke dalam rute parsial dan dihitung jarak rute parsialnya kemudian berlanjut ke langkah 8.

Langkah 7: Untuk suatu klaster k, jika panjang dari rute taklengkap dan elemen ke-p kurang dari atau sama dengan $\sum_{i=1}^{k} n_i + 1$, maka berlanjut ke langkah 1, jika tidak, berlanjut ke Langkah 9.

Langkah 8: Pindah ke baris ke v di tabel alfabet. Perhatikan panjang rute parsialnya, jika lebih besar atau sama dengan $\sum_{i=1}^{k} n_i + 1$, maka nilai $p = \sum_{i=1}^{k} n_i$. Jika tidak, maka nilai p = 1 dan proses beranjut ke Langkah 1.

Langkah 9: Kembali ke verteks sebelumya (misalkan verteks r) yakni baris ke r pada tabel alfabet, kemudian nilai p ditambah 1. Jika nilai verteks r = 1 dan $p = n$, berlanjut ke langkah 10, jika tidak, beralih ke langkah 1.

Langkah 10: Hitung nilai solusi optimal relatif terhadap matriks tereduksi, yaitu nilai solusi terbaik saat ini. Agar diperoleh nilai solusi optimal terhadap matriks jarak awal, maka bias matriks ditambahkan ke dalam solusi optimal, dan berlanjut ke langkah 11.

Langkah 11: Proses dihentikan. Rute yang diperoleh merupakan rute optimal relatif terhadap matriks jarak awal. Sehingga diperoeh solusi rute optimal dari pendistribusian.

Contoh pengerjaan lengkap bagi setiap langkah disajikan lebih lanjut dalam contoh kasus pada bagian selanjutnya.

3.5. Contoh Kasus

Misalkan terdapat data distribusi barang dengan data jarak disajikan dalam Tabel 3.5. Misalkan pula diberikan himpunan verteks ialah $V = \{1,2,\dots,5\}$, dengan lokasi 1 sebagai lokasi awal sekaligus lokasi akhir. Lokasi yang akan diklasterkan adalah lokasi-lokasi pendistribusian yang akan dikunjungi kecuali lokasi awal, sehingga hanya 4 lokasi yang akan diklasterkan. Data jarak antar 4 lokasi yang akan diklasterkan dimuat pada Tabel Jarak 3.6. Akan ditentukan terlebih dahulu berapa banyak klaster yang akan dibuat.

Tabel 3. 5 Matriks Jarak Awal

Lokasi / Jarak (km)	**1**	**2**	**3**	**4**	**5**
1	0	103	106	131	122
2	97	0	21	47	37
3	107	16	0	31	21
4	131	44	31	0	8
5	122	36	23	10	0

Tabel 3. 6 Data Jarak 4 Lokasi

Lokasi / Jarak (km)	**2**	**3**	**4**	**5**
2	0	21	47	37
3	16	0	31	21
4	44	31	0	8
5	36	23	10	0

Elemen paling kecil pada Tabel 3.6 adalah 8, yang merupakan jarak antara lokasi 4 dan 5, maka klaster pertama adalah (4,5). Selanjutnya, dilakukan perhitungan jarak antara klaster pertama dengan lokasi lainnya.

$$d_{(4,5)2} = \max\{d_{4,2}; d_{5,2}\} = \max\{44; 36\} = 44$$

$$d_{(4,5)3} = \max\{d_{4,3}; d_{5,3}\} = \max\{31; 23\} = 31$$

Sehingga diperoleh data jarak baru pada Tabel 3.7.

Tabel 3. 7 Update 1 Data Jarak 4 Lokasi

Lokasi / Jarak (km)	**(4,5)**	**2**	**3**
(4,5)	0	44	31
2	44	0	21
3	31	16	0

Elemen paling kecil pada Tabel 3.7 adalah 16, yang merupakan jarak antara lokasi 3 dan 2, maka klaster selanjutnya adalah (2,3). Karena setiap lokasi terlah mendapatkan klaster, maka diperoleh bahwa banyaknya klaster adalah 2, dengan klaster $V_1 = \{2,3\}$ dan $V_2 = \{4,5\}$. Ini berarti bahwa banyaknya $n_1 = 2$ dan banyaknya $n_2 = 2$. Pada tahap modifikasi matriks jarak ini, misalkan diberikan matriks $C_{ij} = [c_{ij}]$ merupakan matriks jarak yang dimodifikasi berukuran $n \times n$ dengan c_{ij} merupakan jarak dari simpul i ke simpul j. Selanjutnya dilakukan modifikasi pada matriks jarak sebagai berikut:

Jarak pada setiap sisi (i, j) diberikan nilai sebesar besarnya (misalkan 999) kecuali pada sisi berikut:

1. Sisi $(1, j)$, untuk j = 2, 3, yaitu sisi $(1,2)$ dan sisi $(1,3)$
2. Sisi $(i, 1)$, untuk i = 4, 5, yaitu sisi $(4,1)$ dan sisi $(5,1)$
3. Sisi (i, j), untuk $i \neq j$ dan
   a. $i = 2, 3;\ j = 2, 3, 4, 5$; yaitu sisi (2,3), (2,4), (2,5), (3,4), dan (3,5).
   b. $i = 4, 5;\ j = 4, 5$; yaitu sisi (4,5) dan (5,4).

Sehingga diperoleh modifikasi matriks jarak pada Tabel 3.8.

Tabel 3. 8 Matriks jarak awal yang dimodifikasi

Vertex	1	2	3	4	5
1	999	103	106	999	999
2	999	999	21	47	37
3	999	16	999	31	21
4	131	999	999	999	8
5	122	999	999	10	999

Kemudian cari elemen terkecil pada setiap baris, sehingga diperoleh Tabel 3.9.

Tabel 3. 9 Minimal Baris Matriks Jarak

Vertex	1	2	3	4	5	Minimal baris
1	999	103	106	999	999	103
2	999	999	21	47	37	21
3	999	16	999	31	21	16
4	131	999	999	999	8	8
5	122	999	999	10	999	10

Selanjutnya elemen pada setiap baris dikurangi dengan elemen terkecil (nilai minimal baris) pada setiap baris, sehingga diperoleh Tabel 3.10.

Tabel 3. 10 Update Minimal Baris Matriks Jarak

Vertex	1	2	3	4	5	Minimal baris
1	896	0	3	896	896	103
2	978	978	0	26	16	21
3	983	0	983	15	5	16
4	123	991	991	991	0	8
5	112	989	989	0	989	10

Kemudian cari elemen terkecil pada setiap kolom, sehingga diperoleh Tabel 3.11. Setelah itu, kurangi elemen pada setiap kolom dengan masing-masing elemen terkecil kolomnya, sehingga diperoleh Matriks Jarak Direduksi pada Tabel 3.12.

Bias dari matriks adalah penjumlahan dari setiap elemen terkecil pada setiap baris dengan setiap elemen terkecil pada setiap kolom. Dengan demikian diperoleh Bias Matriks:

$$\sum_{i=1}^{5} u_i + \sum_{j=1}^{5} v_j = 158 + 112 = 270\ km$$

Tabel 3. 11 Minimal Kolom Matriks Jarak

Vertex	**1**	**2**	**3**	**4**	**5**	**Minimal baris**
1	896	0	3	896	896	103
2	978	978	0	26	16	21
3	983	0	983	15	5	16
4	123	991	991	991	0	8
5	112	989	989	0	989	10
Minimal Kolom	112	0	0	0	0	

Tabel 3. 12 Matriks Jarak Direduksi

Vertex	**1**	**2**	**3**	**4**	**5**	**Minimal baris**
1	884	0	3	896	896	103
2	866	978	0	26	16	21
3	871	0	983	15	5	16
4	11	991	991	991	0	8
5	0	989	989	0	989	10
Minimal kolom	112	0	0	0	0	

Setelah diperoleh matriks jarak yang direduksi, maka akan dikonstruksi tabel alfabet dengan menghapus bias pada Matriks Jarak Direduksi kemudian membuat urutan secara tak turun berdasarkan nilai jarak pada setiap baris i dan diidentiifkasi verteks j yang bersesuaian dengan jarak c_{ij}. Matriks pada Tabel 3.10, setelah diurutkan maka diperoleh baris ke-1 sebagai berikut:$c_{12} = 0$,$c_{13} = 3$, $c_{11} = 884$, $c_{14} = 896$, $c_{15} = 896$. Dengan cara yang sama untuk baris-baris yang lainnya, sehingga diperoleh Tabel Alfabet pada Tabel 3.13.

Setelah diperoleh tabel alfabet, maka selanjutnya adalah mencari rute dari lokasi awal (verteks ke-1) dengan menelusuri kedua klaster hingga kembali lagi ke lokasi awal.

Tabel 3. 13 Tabel Alfabet.

	$v-jarak$	$v-jarak$	$v-jarak$	$v-jarak$	$v-jarak$
1	2-0	3-3	1-884	4-896	5-896
2	3-0	5-16	4-26	1-866	2-978
3	2-0	5-5	4-15	1-871	3-983
4	5-0	1-11	2-991	3-991	4-991
5	1-0	4-0	2-989	3-989	5-989

Iterasi 1

Langkah 0: Tetapkan asumsi "solusi terbaik" = 999 dan "jarak rute parsial" = 0, dan $p = 1$.

Langkah 1: Dimulai dari baris pertama pada Tabel Alfabet, dengan $a(1,1) = 2$ dan "jarak verteks sekarang" = jarak = $c_{12} = 0$. Diperoleh: jarak + jarak rute parsial = 0+0 < solusi terbaik = 999. Maka berlanjut ke langkah 2.

Langkah 2: Verteks 2 tidak membentuk sub-*tour*, dilanjutkan ke langkah 3

Langkah 3: Belum semua verteks dikunjungi, dilanjutkan ke langkah 5.

Langkah 5: Leader blok saat ini: (1,2). Batas bawah nilai fungsi objektif ialah:

$$\text{Batas} = c_{2,\alpha(2,1)} + c_{3,\alpha(3,1)} + c_{4,\alpha(4,1)} + c_{5,\alpha(5,1)}$$

$$= c_{2,3} + c_{3,2} + c_{4,5} + c_{5,1}$$

$$= 0 + 0 + 0 + 0 + 0$$

= 0

Langkah 6: Karena Batas + jarak + jarak rute parsial= 0+0+0 < solusi terbaik =999, maka verteks 2 diterima untuk masuk ke rute: {1→2}. Jarak rute parsial = 0 + 0 = 0

Langkah 8: Pergi ke subblok, yaitu baris ke-2 pada Tabel Alfabet.

Karena panjang rute tak lengkap = 2 < n=5, maka $p = 1$, dan berlanjut ke langkah 1 untuk iterasi ke-2.

Iterasi 2:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke dua Tabel Alfabet, dengan $a(2,1) = 3$ dan jarak verteks sekarang = jarak = $c_{23} = 0$. Diperoleh: jarak + jarak rute parsial = 0+0 =0 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 3 tidak membentuk sub-tour.

Langkah 3: Belum semua verteks dikunjungi, dilanjutkan ke langkah 5

Langkah 5: Leader blok saat ini: (1,2,3). Batas bawah nilai fungsi objektif:

Batas = $c_{3,\alpha(3,2)} + c_{4,\alpha(4,1)} + c_{5,\alpha(5,1)}$

= $c_{3,5} + c_{4,5} + c_{5,1}$

= 5 + 0 + 0

= 5

Langkah 6: Karena Batas + jarak + jarak rute parsial = 5+0+0< solusi terbaik =999, maka verteks 3 diterima untuk masuk ke rute: {1→2→3}. Jarak rute parsial = 0 + 0 = 0

Langkah 8: Pergi ke subblok, yaitu baris ke-3 pada Tabel Alfabet.

Karena panjang rute tak lengkap = 3 < n =5, maka $p = 1$, dan berlanjut ke tahap 1 untuk iterasi ke 3.

Iterasi 3:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke-3 Tabel Alfabet, dengan $a(3,1) = 2$ dan jarak verteks sekarang = jarak = $c_{32} = 0$. Diperoleh:

jarak + jarak rute parsial = 0+0 =0 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 2 membentuk sub-*tour*, maka $p = p + 1 = 2$, dan dilanjut ke langkah 7

Langkah 7: Untuk klaster pertama maka: $\sum_{i=1}^{1} n_i + 1 = 3$. Panjang rute tak lengkap = 3 ≤ 3 dan $p = 2 < 3$, maka berlanjut ke langkah 1.

Langkah 1: Berlanjut pada elemen ke-2 pada baris ke-3 Tabel Alfabet, dengan $a(3,2) = 5$ dan jarak verteks sekarang = jarak = $c_{35} = 5$. Diperoleh: jarak + jarak rute parsial = 5+0 =5 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 5 tidak membentuk sub-*tour*, maka $p = 2$, dan dilanjut ke langkah 3

Langkah 3: Belum semua verteks dikunjungi, dilanjutkan ke langkah 5

Langkah 5: Leader blok saat ini: (1,2,3,5). Batas bawah nilai fungsi objektif:

Batas = $c_{4,\alpha(4,1)} + c_{5,\alpha(5,1)}$

= $c_{4,5} + c_{5,1}$

= 0+0

= 0

Langkah 6: Karena Batas + jarak + jarak rute parsial = 0 +5+0< solusi terbaik =999, maka verteks 5 diterima untuk masuk ke rute: {1→2→3→5}. Jarak rute parsial = 0+5 = 5.

Langkah 8: Pergi ke subblok, yaitu baris ke-5 pada Tabel Alfabet.

Karena panjang rute tak lengkap = 4 < n=5, maka $p = 1$, dan berlanjut ke langkah 1 untuk iterasi ke 4.

Iterasi 4:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke-5 Tabel Alfabet, dengan $a(5,1) = 1$ dan jarak verteks sekarang = jarak = $c_{5,1} = 0$. Diperoleh: jarak + jarak rute parsial = 0+5=5 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 1 membentuk sub-*tour*, maka $p = p + 1 = 2$, dan dilanjut ke langkah 7

Langkah 7: Untuk klaster kedua, maka: $\sum_{i=1}^{2} n_i + 1 = 3 + 3 = 6$. Panjang rute tak lengkap = 4<6 dan $p = 2 < 6$, maka berlanjut ke langkah 1.

Langkah 1: Berlanjut pada elemen ke-2 pada baris ke-5 Tabel Alfabet, dengan $a(5,2) = 4$ dan jarak verteks sekarang = jarak = $c_{5,4} = 0$. Diperoleh: jarak + jarak rute parsial = 0+5=5 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 4 tidak membentuk sub-*tour*, maka dilanjut ke langkah 3

Langkah 3: Belum semua verteks dikunjungi, dilanjutkan ke langkah 5

Langkah 5: Leader blok saat ini: (1,2,3,5,4). Batas bawah nilai fungsi objektif:

Batas = $c_{4,\alpha(4,2)}$

= $c_{4,1}$

= 11

Langkah 6: Karena Batas + jarak + jarak rute parsial = 11+0+5 < solusi terbaik =999, maka verteks 4 diterima untuk masuk ke rute: {1→2→3→5→4}. Jarak rute parsial = 5 + 0= 5

Langkah 8: Pergi ke subblok, yaitu baris ke-4 pada tabel alfabet.

Karena panjang rute tak lengkap = 5 ≤ 5, maka $p = 1$, dan berlanjut ke langkah 1 untuk iterasi ke 5.

Iterasi 5:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke-4 Tabel Alfabet, dengan $a(4,1) = 5$ dan jarak verteks sekarang = jarak = $c_{4,5} = 0$. Diperoleh: jarak + jarak rute parsial = 0+11=11 < solusi terbaik = 999. Lanjutkan ke langkah 2.

Langkah 2: Verteks 5 membentuk sub-*tour*, maka $p = p + 1 = 2$, dan dilanjut ke langkah 7.

Langkah 7: Untuk klaster kedua, maka: $\sum_{i=1}^{2} n_i + 1 = 3 + 3 = 6$. Panjang rute tak lengkap = 5 ≤ 5 dan $p = 2 < 5$, maka berlanjut ke langkah 1.

Langkah 1: Berlanjut pada elemen ke-2 pada baris ke-4 tabel alphabet, dengan $a(4,2) = 1$ dan jarak verteks sekarang = jarak = $c_{4,1} = 11$. Diperoleh: jarak + jarak rute parsial = 11+6 = 16 < solusi terbaik (999). Lanjutkan ke langkah 2.

Langkah 2: Verteks 1 tidak membentuk sub-*tour*, berlanjut ke langkah 3

Langkah 3: Semua verteks telah dikunjungi, dengan rute lengkap {1→2→3→5→4→1} dan jarak rute lengkap = 16 < 999. Berlanjut ke langkah 4.

Langkah 4: Karena jarak rute lengkap =16 < solusi terbaik=999, maka solusi terbaik saat ini adalah 16.

Langkah 9: Rute lengkap menjadi {1→2→3→5→4→1} dengan jarak rute terbaik adalah 16 km. Lompati blok ini dengan level lebih tinggi. Misalkan lompat ke blok dengan leader (1, 2, 3), berarti pindah ke baris ke-3 dan p = p +1 = 3, dan berlanjut ke Langkah 1.

Iterasi 6:

Langkah 1: Berlanjut pada elemen ke-3 pada baris ke-3 Tabel Alfabet, dengan $a(3,3) = 4$ dan jarak verteks sekarang = jarak = $c_{3,4} = 15$. Diperoleh: jarak + jarak rute parsial = 15+0=15 < solusi terbaik = 16. Lanjutkan ke langkah 2.

Langkah 2: Verteks 4 tidak membentuk sub-*tour*, berlanjut ke langkah 3.

Langkah 3: Belum semua verteks dikunjungi, dilanjutkan ke langkah 5.

Langkah 5: Leader blok saat ini: (1,2,3,4). Batas bawah nilai fungsi objektif:

Batas = $c_{4,\alpha(4,1)} + c_{5,\alpha(5,1)}$

$= c_{4,5} + c_{5,1}$

$= 0 + 0$

$= 0$

Langkah 6: Karena Batas + jarak + jarak rute parsial = 0+0+15 = 15 < solusi terbaik =16, maka verteks 4 diterima untuk masuk ke rute: {1→2→3→4}. Jarak rute parsial = 15 + 0= 15

Langkah 8: Pergi ke subblok, yaitu baris ke-4 pada tabel alfabet.

Karena panjang rute tak lengkap = 4 ≤ 5, maka $p = 1$, dan berlanjut ke langkah 1 untuk iterasi ke 7.

Iterasi 7:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke-4 Tabel Alfabet, dengan $a(4,1) = 5$ dan jarak verteks sekarang = jarak = $c_{4,5} = 0$. Diperoleh: jarak + jarak rute parsial = 15+0=15 < solusi terbaik = 16. Lanjutkan ke langkah 2.

Langkah 2: Verteks 5 tidak membentuk sub-*tour*, berlanjut ke Langkah 3

Langkah 3: Belum semua verteks dikunjungi.

Langkah 5: Leader blok saat ini: (1,2,3,4,5). Batas bawah nilai fungsi objektif:

Batas = $c_{5,\alpha(5,1)}$

= $c_{5,1}$

= 0

Langkah 6: Karena Batas + jarak + jarak rute parsial = 0+0+15 = 15 < solusi terbaik =16, maka verteks 4 diterima untuk masuk ke rute: {1→2→3→4}. Jarak rute parsial = 15 + 0= 15

Langkah 8: Pergi ke subblok, yaitu baris ke-4 pada tabel alfabet.

Karena panjang rute tak lengkap = 5 ≤ 5, maka $p = 1$, dan berlanjut ke langkah 1 untuk iterasi ke 8.

Iterasi 8:

Langkah 1: Berlanjut pada elemen ke-1 pada baris ke-5 tabel alphabet, dengan $a(5,1) = 1$ dan jarak verteks sekarang = jarak = $c_{5,1} = 0$. Diperoleh: jarak + jarak rute parsial = 0 + 15 = 15 < solusi terbaik = 16. Lanjutkan ke langkah 2.

Langkah 2: Verteks 1 tidak membentuk sub-*tour* dan tidak melanggar aturan klaster.

Langkah 3: Semua verteks telah dikunjungi, dengan rute lengkap {1→2→3→4→5→1} dan jarak rute lengkap = 15 < 16. Berlanjut ke langkah 4.

Langkah 4: Karena jarak rute lengkap =15 < solusi terbaik = 16, maka solusi terbaik saat ini adalah 15.

Langkah 9: Rute lengkap menjadi {1→2→3→4→5→1} dengan jarak rute terbaik adalah 15 km.

Langkah 10: Bias matriks C yaitu: $\sum_{i=1}^{5} u_i + \sum_{j=1}^{5} v_j = 158 + 112 = 270$ km

Langkah 11: Sehingga jarak rute lengkap dengan matriks sebenarnya adalah:

Bias matriks + jarak rute terbaik = 270 + 15 = 285 km.

Sehingga diperoleh rute optimal 1→2→3→4→5→1, dengan jarak keseluruhan rute tersebut adalah 285 km.