# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1 Formulasi Model

Formulasi model VRP yang secara umum telah dijelaskan secara umum pada BAB II akan digunakan dalam merumuskan fungsi tujuan dan fungsi kendala pada pendistribusian roti. Sebelum merumuskan kedua fungsi tersebut, akan dijelaskan terlebih dahulu indeks, variabel, dan parameter yang akan digunakan dalam formulasi model. Adapun indeks yang digunakan adalah sebagai berikut:

$i$ : indeks untuk konsumen awal, $i = 1,2,..,N$
$j$ : indeks untuk konsumen tujuan, $j = 1,2,..,N$
$k$ : indeks untuk kendaraan, $k = 1,2,..K$

Adapun variabel dan parameter yang digunakan dalam formulasi model adalah:

$N$ : jumlah konsumen
$K$ : jumlah kendaraan
$c_{ij}$ : jarak antar konsumen
$q_{ik}$ : total *demand* kendaraan $k$ sampai konsumen $i$
$v_k$ : kapasitas maksimum kendaraan $k$

### 4.1.1 Variabel keputusan

Variabel keputusan yang digunakan dalam pendistribusian roti adalah sebagai berikut:

$$y_{ik} = \begin{cases} 1 & \text{Jika konsumen } i \text{ dilayani oleh kendaraan } k \\ 0 & \text{Jika tidak demikian} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{Jika kendaraan } k \text{ dari konsumen } i \text{ langsung ke } j \\ 0 & \text{Jika tidak demikian} \end{cases}$$

dengan $i, j = 1,2,..,N; k = 1,2,..,K$.

### 4.1.2 Formulasi tujuan

a. Tujuan ini memastikan bahwa untuk meminimalkan jarak perjalanan berlaku persamaan (4.1). Variabel $x_{ijk}$ adalah variabel keputusan dimana kendaraan $k$ dari konsumen $i$ langsung ke konsumen $j$.

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} c_{ij}\, x_{ijk} \tag{4.1}$$

b. Tujuan ini memastikan bahwa total *demand* yang dibawa oleh kendaraan $k$ tidak boleh melebihi kapasitas kendaraan tersebut. Untuk setiap $k = 1,2,..,K$ berlaku persamaan (4.2). Variabel $y_{ik}$ adalah variabel keputusan untuk pelayanan konsumen $i$ oleh kendaraan $k$.

$$\sum_{i=0}^{N} q_{ik} y_{ik} \leq v_k \tag{4.2}$$

c. Tujuan ini memastikan bahwa tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Untuk setiap $i = 1,2,..,N$ berlaku persamaan (4.3). Variabel $y_{ik}$ adalah variabel keputusan untuk pelayanan konsumen $i$ oleh kendaraan $k$.

$$\sum_{k=1}^{K} y_{ik} = 1 \tag{4.3}$$

d. Tujuan ini memastikan bahwa semua kendaraan berangkat dan berakhir di depot. Untuk $i = 0$ berlaku persamaan (4.4). Variabel $y_{ik}$ adalah variabel keputusan untuk pelayanan konsumen $i$ oleh kendaraan $k$.

$$\sum_{k=1}^{K} y_{ik} = K \tag{4.4}$$

e. Tujuan ini memastikan bahwa tiap konsumen dikunjungi oleh kendaraan yang sama dengan yang sudah dijadwalkan untuk konsumen tersebut. Untuk setiap $i = 1,2,..,N$, $j = 0,1,,..,N$, dan $k = 1,2,..,K$ berlaku persamaan (4.5) dan (4.6). Variabel $x_{ijk}$ adalah variabel keputusan untuk menyatakan bahwa kendaraan $k$ melayani konsumen $j$ setelah konsumen $i$,

sedangkan $y_{ik}$ adalah variabel keputusan untuk pelayanan konsumen $i$ oleh kendaraan $k$.

$$\sum_{i=0}^{N} x_{ijk} = y_{jk} \quad (4.5)$$

$$\sum_{j=0}^{N} x_{ijk} = y_{ik} \quad (4.6)$$

### 4.1.3 Perumusan fungsi kendala

Kendala tujuan dalam pendistribusian roti adalah sebagai berikut:

a. Total jumlah *demand* atau permintaan yang dibawa oleh kendaraan $k$ tidak boleh melebihi kapasitas kendaraan tersebut. Untuk setiap $k = 1,2,..,K$ berlaku persamaan (4.2).
b. Tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Untuk setiap $i = 1,2,..,N$ berlaku persamaan (4.3).
c. Semua kendaraan berangkat dan berakhir di depot. Untuk $i = 0$ berlaku persamaan (4.4).
d. Tiap konsumen dikunjungi oleh kendaraan yang sama dengan yang sudah dijadwalkan untuk konsumen tersebut. Untuk setiap $i = 1,2,..,N$ , $j = 0,1,,..,N$ , dan $k = 1,2,..,K$ berlaku persamaan (4.5) dan (4.6).

## 4.2 Rute Distribusi

Rute distribusi diperoleh dengan menggunakan algoritma *nearest neighbor* dan algoritma *tabu search.* Algoritma *nearest neighbor* digunakan untuk menyelesaikan solusi awal, sedangkan algoritma *tabu search* digunakan untuk menyelesaikan solusi optimal. Dari pengerjaan solusi awal, akan didapatkan banyaknya rute perjalanan, jumlah kendaraan yang diperlukan, serta banyaknya permintaan yang harus dibawa oleh tiap kendaraan, sedangkan dari pengerjaan solusi optimum, akan didapatkan rute perjalanan yang optimum yaitu rute yang memiliki jarak minimum perjalanan tiap rute.

### 4.2.1 Pengerjaan solusi awal

#### 4.2.1.1 *Input*

*Input* data yang diperlukan untuk pengolahan data awal dengan menggunakan algoritma *nearest neighbor* ini adalah matriks jarak (yaitu jarak dari depot ke konsumen dan dari konsumen ke konsumen lainnya), kapasitas kendaraan, dan jumlah permintaan tiap konsumen.

#### 4.2.1.2 Langkah Pengerjaan

Langkah pengerjaan solusi awal ini bertujuan untuk memperoleh rute awal, jumlah kendaraan yang diperlukan serta banyaknya permintaan yang harus dibawa oleh tiap kendaraan. Langkah awal dalam algoritma *nearest neighbor* adalah mencari konsumen yang terdekat dari depot kemudian melanjutkannya sampai rute dikatakan *feasible.* Rute dikatakan *feasible* apabila kendala terpenuhi, yaitu jumlah permintaan (*demand*) kurang dari sama dengan kapasitas kendaraan. Jika jumlah permintaan (*demand*) melebihi kapasitas kendaraan, maka kendaraan kembali ke depot.

Untuk perhitungan solusi awal dengan algoritma *nearest neighbor*, disediakan 5 buah kendaraan dengan kapasitas yang sama yaitu 200 *create*. Matriks jarak dan permintaan tiap konsumen tercantum pada Lampiran 1.1 dan Lampiran 1.2. Berikut ini adalah perhitungannya.

1. Kendaraan I

- Langkah 1

Rute terbentuk dari konsumen yang jaraknya terdekat dari posisi sebelumnya. Pada Tabel 4.1 rute yang dilayani oleh kendaraan I berawal dari depot menuju ke konsumen ke-1 karena konsumen ke-1 jaraknya paling dekat dengan depot dibandingkan konsumen-konsumen lainnya. Setelah dari konsumen ke-1 kemudian kendaraan I menuju konsumen ke-4, konsumen ke-5, dan selanjutnya.

Untuk konsumen ke-24 tidak *feasible* karena setelah melewati konsumen ke-23 sisa muatan yang dibawa oleh kendaraan I kurang dari permintaan konsumen ke-24. Permintaan konsumen ke-24 yaitu

28*crate*, sedangkan sisa muatan yang dibawa oleh kendaraan I hanya 16 *crate.*

Tabel 4.1 Langkah 1 kendaraan ke-1

| Konsumen ke- | *Demand* | Yang isi (awalnya=200) | *Feasible* |
|---|---|---|---|
| 1 | 17 | 183 | Ya |
| 4 | 17 | 166 | Ya |
| 5 | 8 | 158 | Ya |
| 3 | 7 | 151 | Ya |
| 9 | 2 | 149 | Ya |
| 8 | 17 | 132 | Ya |
| 7 | 19 | 113 | Ya |
| 10 | 19 | 94 | Ya |
| 14 | 22 | 72 | Ya |
| 16 | 9 | 63 | Ya |
| 17 | 4 | 59 | Ya |
| 22 | 18 | 41 | Ya |
| 23 | 25 | 16 | Ya |
| 24 | 28 | - | Tidak |

- Langkah 2

Karena muatan kendaraan I setelah melewati konsumen ke-23 masih ada 16 *crate*, maka pada Tabel 4.2 muatan kendaraan dikurangi menjadi 184 agar setelah melewati konsumen ke-23 tidak ada sisa muatan. Setelah tidak ada muatan yang tersisa, kendaraan kembali ke depot dan didapatkan rute sebagai berikut: 0-1-4-5-3-9-8-7-10-14-16-17-22-23-0.

Tabel 4.2 Langkah 2 kendaraan ke-1

| Konsumen ke- | *Demand* | Yang isi (awalnya200-16 = 184) | *Feasible* |
|---|---|---|---|
| 1 | 17 | 167 | Ya |
| 4 | 17 | 150 | Ya |
| 5 | 8 | 142 | Ya |
| 3 | 7 | 135 | Ya |
| 9 | 2 | 133 | Ya |
| 8 | 17 | 116 | Ya |
| 7 | 19 | 97 | Ya |
| 10 | 19 | 78 | Ya |
| 14 | 22 | 56 | Ya |
| 16 | 9 | 47 | Ya |
| 17 | 4 | 43 | Ya |
| 22 | 18 | 25 | Ya |
| 23 | 25 | 0 | Ya |

Karena muatan kendaraan I setelah melewati konsumen ke-23 masih ada 16 *crate*, maka pada Tabel 4.2 muatan kendaraan dikurangi menjadi 184 agar setelah melewati konsumen ke-23 tidak ada sisa muatan. Setelah tidak ada muatan yang tersisa, kendaraan kembali ke depot dan didapatkan rute sebagai berikut: 0-1-4-5-3-9-8-7-10-14-16-17-22-23-0.

2. Kendaraan II

- Langkah 1

Pada Tabel 4.3 rute yang dilayani oleh kendaraan II berawal dari depot menuju ke konsumen ke-6 karena konsumen ke-6 jaraknya paling dekat dengan depot dibandingkan konsumen-konsumen lainnya. Setelah dari konsumen ke-6 kemudian kendaraan II menuju konsumen ke-18, konsumen ke-19, dan selanjutnya.

Tabel 4.3 Langkah 1 kendaraan ke-2

| Konsumen | *Demand* | Yang isi (awalnya=200) | *Feasible* |
|---|---|---|---|
| 6 | 12 | 188 | Ya |
| 18 | 11 | 177 | Ya |
| 19 | 18 | 159 | Ya |
| 20 | 18 | 141 | Ya |
| 24 | 18 | 123 | Ya |
| 21 | 37 | 86 | Ya |
| 15 | 15 | 71 | Ya |
| 12 | 16 | 55 | Ya |
| 13 | 12 | 43 | Ya |
| 11 | 37 | 6 | Ya |
| 2 | 3 | 3 | Ya |

- Langkah 2

Tabel 4.4 Langkah 2 kendaraan ke-2

| Konsumen | *Demand* | Yang isi (awalnya 200-3 = 197) | *Feasible* |
|---|---|---|---|
| 6 | 12 | 185 | Ya |
| 18 | 11 | 174 | Ya |
| 19 | 18 | 156 | Ya |
| 20 | 18 | 138 | Ya |
| 24 | 18 | 120 | Ya |
| 21 | 37 | 83 | Ya |
| 15 | 15 | 68 | Ya |
| 12 | 16 | 52 | Ya |
| 13 | 12 | 40 | Ya |
| 11 | 37 | 3 | Ya |
| 2 | 3 | 0 | Ya |

Karena muatan kendaraan II setelah melewati konsumen ke-2 masih ada 3 *crate*, maka pada Tabel 4.4 muatan kendaraan dikurangi menjadi 197 agar setelah melewati konsumen ke-2 tidak ada sisa muatan. Setelah tidak ada muatan yang tersisa, kendaraan kembali ke

depot dan didapatkan rute sebagai berikut: 0-6-18-19-20-24-21-15-12-13-11-2-0.

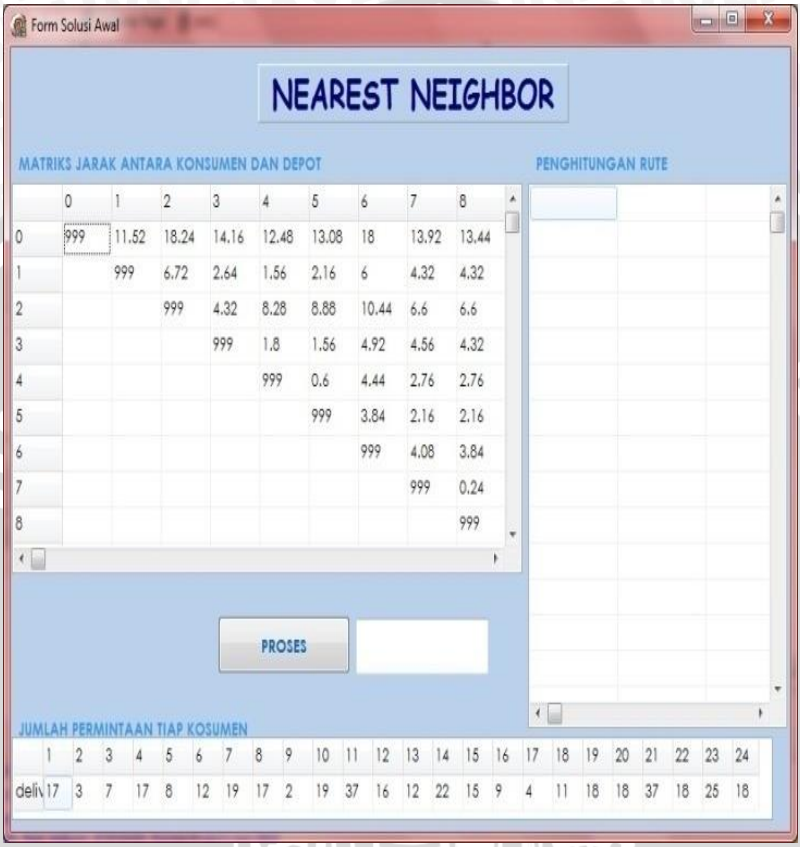
#### 4.2.1.3 *Output*

Hasil dari tahap pengerjaan awal ini berupa rute distribusi awal yang menjadi solusi awal bagi tahap pengerjaan selanjutnya yaitu menggunakan algoritma *tabu search.* Terdapat 2 rute yang dilayani oleh 2 kendaraan. Kendaraan I membawa 184 *crate* yang dikirim ke 13 konsumen dengan rute 0-1-4-5-3-9-8-7-10-14-16-17-22-23-0. Kendaraan II membawa 197 *crate* yang dikirim ke 11 konsumen dengan rute 0-6-18-19-20-24-21-15-12-13-11-2-0.

#### 4.2.1.4 Pengerjaan solusi awal dengan menggunakan software Delphi

Untuk pengolahan data yang pertama yaitu mencari solusi awal, dibuat program dengan menggunakan *software* Delphi. Data yang diperlukan untuk membuat program ini antara lain adalah data permintaan tiap konsumen, kapasitas kendaraan, dan letak konsumen yang disajikan dengan matriks jarak. Pada setiap proses pengerjaan atau *run* program, data yang perlu dimasukkan adalah matriks jarak dan permintaan tiap konsumen. Selanjutnya data yang dimasukkan akan diolah sesuai dengan algoritma *nearest neighbor*. Adapun *output* dari pengolahan data menggunakan program ini adalah urutan rute serta jumlah kendaraan yang diperlukan.*Source code* program ini ada pada Lampiran 2.1.

Form Solusi Awal seperti Gambar 4.1 dan Gambar 4.2 merupakan form yang digunakan untuk memasukkan letak konsumen dan depot yang disajikan dengan matriks jarak serta jumlah permintaan tiap konsumen ke dalam *stringgrid* yang tersedia. Matriks yang dimasukkan berupa matriks segitiga atas. Setelah data dimasukkan, program dijalankan dengan menekan *speedbutton*, kemudian hasilnya akan dikeluarkan pada *edit* dan *stringgrid* penghitungan rute. Nilai pada *edit* adalah jumlah permintaan semua konsumen yang harus dilayani, sedangkan nilai pada *stringgrid* penghitungan rute adalah solusi berupa rute yang dihasilkan serta muatan truk yang harus dibawa. Gambar 4.3 adalah hasil perhitungan solusi awal dengan menggunakan algoritma *nearest neighbor*.




Gambar 4.1 *Input* data matriks jarak dan jumlah permintaan


Gambar 4.2 Hasil perhitungan solusi awal

Gambar 4.3 merupakan rute ke-1 yang dihasilkan dengan muatan yang harus dibawa oleh kendaraan I yaitu 184 *crate*. Gambar 4.4 merupakan rute ke-2 yang dihasilkan dengan muatan yang harus dibawa oleh truk yaitu 167 *crate*.

-

PENGHITUNGAN RUTE

| | | |
|---|---|---|
| 0 | 1 | 17 |
| 1 | 4 | 17 |
| 4 | 5 | 8 |
| 5 | 3 | 7 |
| 3 | 9 | 2 |
| 9 | 8 | 17 |
| 8 | 7 | 19 |
| 7 | 10 | 19 |
| 10 | 14 | 22 |
| 14 | 16 | 9 |
| 16 | 17 | 4 |
| 17 | 22 | 18 |
| 22 | 23 | 25 |
| | | |
| Sisa demand | | 197 |
| Sisa isi truk | | 16 |
| Isi truk skg | | 184 |

Gambar 4.3 Rute ke-1

PENGHITUNGAN RUTE

| | | |
|---|---|---|
| 0 | 6 | 12 |
| 6 | 18 | 11 |
| 18 | 19 | 18 |
| 19 | 20 | 18 |
| 20 | 24 | 18 |
| 24 | 21 | 37 |
| 21 | 15 | 15 |
| 15 | 12 | 16 |
| 12 | 13 | 12 |
| 13 | 11 | 37 |
| 11 | 2 | 3 |
| | | |
| Sisa demand | | 0 |
| Sisa isi truk | | 3 |
| Isi truk skg | | 197 |

Gambar 4.4 Rute ke-2

### 4.2.2 Pengerjaan solusi optimal

#### 4.2.2.1 *Input*

*Input* data yang diperlukan untuk pengolahan solusi awal menjadi solusi optimal dengan menggunakan algoritma *tabu search* ini adalah matriks jarak (yaitu jarak dari depot ke konsumen dan dari konsumen ke konsumen lainnya) dan rute distribusi awal.

#### 4.2.2.2 Langkah pengerjaan

Langkah pengerjaan solusi optimal ini bertujuan untuk memperoleh rute optimal yaitu rute terpendek dari kemungkinan kemungkinan rute yang dibentuk oleh rute awal. Langkah awal dalam algoritma *tabu search* adalah meng-*input*-kan rute awal pada *tabu list*. Selanjutnya dilakukan pengacakan rute awal untuk menentukan kandidat-kandidat solusi baru dengan melakukan *move* (penukaran) *node* (konsumen) yang terdekat, sehingga seluruh *node* (konsumen) telah melalui penukaran satu sama lain. Langkah terakhir yaitu melakukan pemilihan solusi terbaik dari kandidat-kandidat solusi baru yang ada dan menetapkannya sebagai solusi optimal yang dilihat dari nilai terkecil jarak yang ditempuh oleh rute tersebut.

#### 4.2.2.3 *Output*

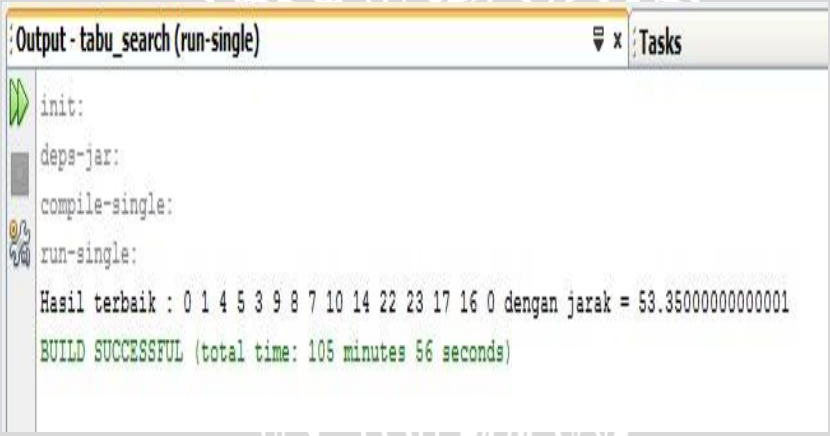
Hasil dari tahap pengerjaan solusi optimal ini berupa rute baru dan jarak total rute. Terdapat 2 rute optimal yang dilayani oleh 2 kendaraan. Rute yang ditempuh oleh kendaraan ke-1 adalah 0-1-4-5-3-9-8-7-10-14-22-23-17-16-0 dengan jarak 53,35 km. Rute yang ditempuh oleh kendaraan ke-2 adalah 0-6-18-19-20-21-24-15-13-12-11-2-0 dengan jarak 78,19 km.

#### 4.2.2.4 Pengerjaan solusi optimal dengan menggunakan *software* Netbeans

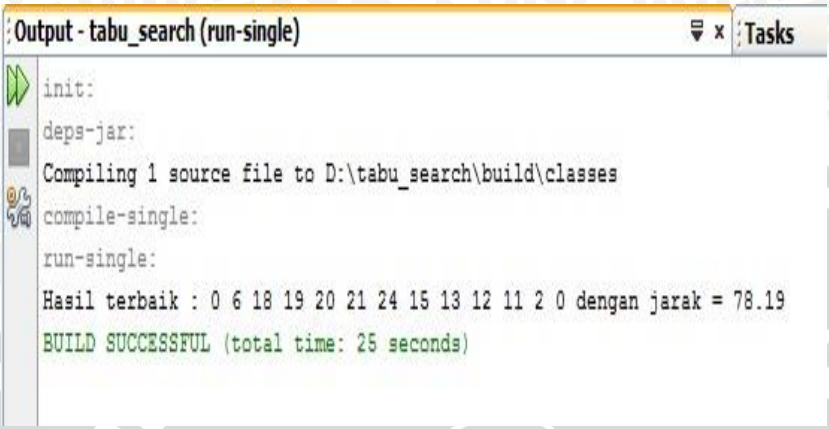
Untuk pengolahan data yang ke-2 yaitu mencari solusi optimal, dibuat program dengan menggunakan *software* Netbeans. Solusi optimal akan diperoleh setelah memasukkan solusi awal yang

dihasilkan dari pengerjaan solusi awal dengan menggunakan program Delphi. Solusi awal tersebut berupa susunan konsumen yang membentuk rute suatu pendistribusian barang. Selain solusi awal, data matriks jarak juga dimasukkan. Pada setiap proses pengerjaan atau *run* program, posisi konsumen ditukar sehingga membentuk *n*! kemungkinan rute dan masing-masing jarak tiap rute tersebut. Rute yang mempunyai jarak paling minimum diantara *n*! kemungkinan disebut sebagai solusi optimum. Adapun *output* dari pengolahan data menggunakan program ini adalah dan rute optimal beserta jaraknya.

Gambar 4.5 merupakan *consule* program yang berisi rute optimal dari 13! kemungkinan rute yang didapat serta jarak rute tersebut. Gambar 4.6 merupakan *consule* program yang berisi rute optimal dari 11! kemungkinan rute yang didapat serta jarak rute tersebut. Untuk *source code*-nya ada pada Lampiran 2.2 dan untuk gambaran rutenya ada pada Lampiran 3.




Gambar 4.5 Rute optimal kendaraan ke-1




Gambar 4.6 Rute optimal kendaraan ke-2

### 4.3 Perbandingan Solusi Optimal VRP Menggunakan Algoritma *Tabu Search* dengan Metode *2-Opt, Or-Opt, Relocate, Exchange, dan Cross* Oleh Aji Raditya (2009)

Penyelesaian VRP menggunakan metode *2-Opt, Or-Opt, Relocate, Exchange,* dan *Cross* yang dilakukan oleh Aji Raditya (2009) dan penyelesaian menggunakan algoritma *tabu search* dengan data yang sama yang ada pada Lampiran 1.1 dan Lampiran 1.2, didapatkan perbedaan-perbedaan yang disajikan pada Tabel 4.5.

Dari Tabel 4.5 aspek perbandingannya adalah jumlah kendaraan yang digunakan atau jumlah rute, rute-rute yang dilalui, dan jarak total semua rute. Kesimpulannya adalah dibandingkan metode *2-Opt, Or-Opt, Relocate, Exchange,* dan *Cross,* algoritma *tabu search* lebih optimal dalam biaya pendistribusian. Dengan lebih sedikitnya kendaraan yang digunakan dan lebih pendek jarak yang ditempuh, maka uang sewa kendaraan, gaji untuk sopir, dan biaya bahan bakar kendaraan juga lebih sedikit dikeluarkan.

Tabel 4.5 Tabel perbandingan

| Perbandingan | *Tabu search* | *2-Opt, Or-Opt, Relocate, Exchange,* dan *Cross* |
|---|---|---|
| Jumlah kendaraan yang digunakan atau jumlah rute | 2 | 3 |
| Rute-rute yang dilalui | Kendaraan I: 0-1-4-5-3-9-8-7-10-14-22-23-17-16-0<br>Kendaraan II: 0-6-18-19-20-21-24-15-13-12-11-2-0 | Kendaraan I: 0-1-4-0<br>Kendaraan II: 0-3-5-2-11-12-13-15-22-23-6-0<br>Kendaraan III: 0-9-8-7-10-14-16-17-24-21-20-19-18-0 |
| Jarak total semua rute (kilometer) | 131.54 | 136.3887 |


UNIVERSITAS BRAWIJAYA