PENERAPAN *FLOWER POLLINATION ALGORITHM* (FPA) UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY* (VRPSPD)

SKRIPSI

MOH.RIZKI KURNIAWAN

PROGRAM STUDI S-1 MATEMATIKA

DEPARTEMEN MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

2019

PENERAPAN *FLOWER POLLINATION ALGORITHM* (FPA) UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY* (VRPSPD)

SKRIPSI

MOH.RIZKI KURNIAWAN

PROGRAM STUDI S-1 MATEMATIKA

DEPARTEMEN MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

2019

PENERAPAN *FLOWER POLLINATION ALGORITHM* (FPA) UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY* (VRPSPD)

SKRIPSI

Sebagai salah satu untuk Memperoleh Gelar Sarjana Sains Bidang Matematika pada Fakultas Sains dan Teknologi

Universitas Airlangga

Disetujui oleh :

Pembimbing I,	**Pembimbing II,**
Dr. Herry Suprajitno, M.Si.	Drs. Edi Winarko, M.Cs.
NIP. 196804041994031020	NIP. 196705141993031001

LEMBAR PENGESAHAN NASKAH SKRIPSI

Judul	: Penerapan *Flower Pollination Algorithm* (FPA) untuk Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)
Penyusun	: Moh.Rizki Kurniawan
NIM	: 081511233056
Tanggal Ujian	: 22 Juli 2019

Disetujui Oleh :

Pembimbing I,	Pembimbing II,
Dr. Herry Suprajitno, M.Si.	Drs. Edi Winarko, M.Cs.
NIP. 196804041994031020	NIP. 196705141993031001

Mengetahui,

Ketua Departemen Matematika	Koordinator Program Studi S-1 Matematika
Fakultas Sains dan Teknologi	Fakultas Sains dan Teknologi
Universiras Airlangga	Universitas Airlangga
Drs. Eko Tjahjono, M.Si.	Dr. Mohammad Imam Utoyo, M.Si.
NIP.196007061986011001	NIP. 196401031988101001

PEDOMAN PENGGUNAAN SKRIPSI

SURAT PERNYATAAN TENTANG ORISINALITAS

Yang bertanda tangan dibawah ini, saya:

Nama	: Moh. Rizki Kurniawan
Nim	: 081511233056
Program Studi	: S-1 Matematika
Fakultas	: Sains dan Teknologi
Jenjang	: Sarjana (S1)

Menyatakan bahwa saya tidak melakukan kegiatan plagiat dalam penulisan skripsi saya yang berjudul:

"PENERAPAN *FLOWER POLLINATION ALGORITHM* (FPA) UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY* (VRPSPD)"

Apabila suatu saat nanti terbukti melakukan tindak plagiat, saya akan menerima sanksi yang telah ditetapkan.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, Juni 2019

Moh. Rizki Kurniawan

Nim. 081511233056

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah SWT atas berkat, rahmat dan hidayah yang telah diberikan sehingga penulis dapat menyelesaikan penulisan proposal skripsi ini. Shalawat serta salam semoga senantiasa tercurahkan kepada junjungan kita, Nabi Besar Muhammad SAW, pemimpin sekaligus sebaik-baik suri teladan bagi insan manusia, sehingga penulis dapat menyelesaikan proposal skripsi dengan judul "Penerapan *Flower Pollination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)".

Dalam penulisan skripsi ini penulis mendapatkan banyak bantuan dan dukungan dari berbagai pihak, sehingga dalam kesempatan ini penyusun mengucapkan terima kasih kepada:

1. Prof. Dr. Moh. Nasih, SE., MH., LL.M., Ph.D. selaku rektor Universitas Airlangga yang memberikan kesempatan dan fasilitas kepada penulis untuk menimba ilmu serta pengalaman.
2. Drs. Eko Tjahjono, M.Si. selaku ketua Departemen Matematika Universitas Airlangga yang selalu memberikan motivasi dan arahan.
3. Dr. Moh. Imam Utoyo, M.Si. selaku Koprodi S-1 Matematika Fakultas Sains dan Teknologi Universitas Airlangga yang telah memberikan arahan berupa kritik dan saran yang bermanfaat.
4. Dr. Herry Suprajitno, M.Si. selaku dosen pembimbing I yang telah dengan sabar memberikan bimbingan dan pengarahan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan lancar.
5. Drs. Edi Winarko, M.Cs. selaku dosen wali dan dosen pembimbing II yang telah memberikan saran, motivasi dan bimbingan kepada penulis selama perkuliahan.
6. Bapak dan Ibu dosen khususnya dosen program studi S-1 Matematika Universitas Airlangga atas segala ilmu, nasihat, dan pengalaman yang diberikan kepada penulis.

7. Kedua orang tua dan adik kandung saya, yaitu Bapak Suherman, Ibu Siti Choiriyah dan Saudari Firda Aulia Putri Damayanti serta segenap keluarga besar yang tidak ada henti-hentinya yang selalu mendoakan, mendukung, memotivasi, menghibur dan mengingatkan penulis.
8. Teman-teman grup GPP (Onne, Dhilas, Dhilaf, Alfiah, Faridah, Sofiah, Faizah, Safira, Siwi, Eka, Piping, Nuy, Dini) yang selalu memberikan semangat kepada penulis.
9. Teman-teman cowok (Chris, Miftah, Nanok, Dipta, Safri, Irul, Aldo) serta teman-teman Prodi Matematika 2015 yang selalu mendukung penulis selama masa perkuliahan.
10. Temen-temen Jember Squad (Annisa, Aisyah, Mila) serta Nastiti, Siwi, dan Sidiq yang selalu memberikan dukungan dan saran kepada penulis.
11. Teman-teman grup monopoli (Ais, Patricia, Della) dan Luthfiana yang sudah memberikan motivasi dan warna baru bagi kehidupan penulis.
12. Teman-teman dari HIMATIKA UNAIR khususnya bidang Pengmas HIMATIKA 2017 (Faliq, Annisa, Aisyah, Siska, Alfiah, Devi, Indah, Vyrda) yang telah memberikan kesempatan penulis untuk berkontribusi dan menjadi wadah untuk mengembangkan diri.
13. Semua pihak yang tidak dapat disebutkan, yang telah membantu terselesaikannya proposal skripsi ini.

Penulis berharap semoga skripsi ini bermanfaat bagi bahan pustaka dan penambah informasi khususnya bagi mahasiswa Universitas Airlangga.

Surabaya, Juli 2019

Penulis,

Moh.Rizki Kurniawan

Moh. Rizki Kurniawan, 2019, **Penerapan *Flower Pollination Algorithm* (FPA) untuk Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).** Skripsi ini dibawah bimbingan Dr. Herry Suprajitno, M.Si. dan Drs. Edi Winarko, M.Cs. Departemen Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga, Surabaya.

ABSTRAK

Penulisan skripsi ini bertujuan untuk menyelesaikan *Vehicle Routing Problem with Siimultaneous Pickup and Delivery* dengan menggunakan *Flower Pollination Algorithm. Vehicle Routing Problem with Siimultaneous Pickup and Delivery* (VRPSPD) adalah suatu permasalahan dalam pembentukan rute kendaraan yang digunakan untuk melayani setiap pelanggan baik pengiriman maupun pengambilan barang secara bersamaan dengan meminimumkan total jarak tempuh kendaraan untuk melayani seluruh konsumen, yang masing-masing dilayani sekali tanpa melebihi batasan kapasitas setiap kendaraan yang digunakan. *Flower Pollination Algorithm* (FPA) merupakan salah satu algoritma yang terinspirasi dari alam, yaitu terinspirasi dari proses penyerbukan bunga pada tanaman. FPA mempunyai 2 langkah kunci yaitu penyerbukan global dan penyerbukan lokal yang ditentukan berdasarkan *switch probability* (p_a) yang terletak pada interval [0,1]. Bahasa pemrograman yang digunakan untuk menyelesaikan skripsi ini adalah *Java* yang diimplementasikan pada 3 data yaitu, data 13 pelanggan, data 22 pelanggan dan data 100 pelanggan. Dari hasil running program diperoleh total jarak tempuh minimum data 13 pelanggan yaitu 99, data 22 pelanggan yaitu 124 dan data 100 pelanggan yaitu 2483. Berdasarkan hasil yang diperoleh, dapat disimpulkan bahwa semakin besar jumlah bunga dan jumlah iterasi maka penyelesaian VRPSPD cenderung lebih baik dengan total jarak tempuh yang lebih kecil. Serta semakin besar presentase barang yang diangkut dari depot maka penyelesaian VRPSPD lebih baik dengan total jarak tempuh yang lebih kecil.

Kata Kunci: *Flower Pollination Algorithm* (FPA), *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

Moh. Rizki Kurniawan, 2019, **Penerapan *Flower Pollination Algorithm* (FPA) untuk Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).** This undergraduate thesis is supervised by Dr. Herry Suprajitno, M.Si. dan Drs. Edi Winarko, M.Cs. Department of Mathematics, Faculty of Science and Technology, Airlangga University, Surabaya.

ABSTRAK

The purpose of writing this undergraduate thesis is to solve Vehicle Routing Problem with Simultaneous Pickup and Delivery using Flower Pollination Algorithm. Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) is a problem in the formation of the vehicle routes that used to serve every customer both delivery and pickup simultaneously by minimizing the total mileage of the vehicle to serve all customers, each of which is serviced once without exceeding the limits of the capacity of each vehicle used. Flower Pollination Algorithm (FPA) is one of algorithm inspired by nature, specifically the process of flower pollination. FPA have two main steps namely global pollination and local pollination that are controlled by switch probability (p_a) in the interval [0,1]. The program is created by using Java programming language that will be implemented on 3 cases, data 13 cuustomers, data 22 customers and data 100 customers. Based on the program simulation result, the minimum total distance for data 22 customers is 99, data 22 customers is 124 and data 100 customers is 2483. Based on the result, it can be concluded that higher number of flowers population and iteration, cause the total distances tends to be minimum. Similarly, that higher the percentage of goods transported from the depot, causes the total distance become minimum.

Kata Kunci: *Flower Pollination Algorithm* (FPA), *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada era revolusi industri 4.0 saat ini, dunia industri memiliki peranan yang penting dalam perkembangan suatu negara. Dengan demikian banyak industri baru yang bermunculan dan mengakibatkan daya saing di dunia industri semakin tinggi. Maka dari itu diperlukan strategi perusahaan untuk memperoleh laba yang sebesar-besarnya. Salah satu strategi yang dapat dilakukan perusahaan yaitu dengan mengoptimalkan aspek distribusi barang. Distribusi barang merupakan salah satu kegiatan yang penting guna memperlancar pengiriman produk dari produsen ke konsumen secara cepat dan tepat. Kegiatan pendistribusian produk kepada konsumen dilakukan oleh distributor sesuai dengan pembelian barang yang dibutuhkan konsumen **(Meyers dan Stewart, 2001)**. Karena lokasi konsumen tersebar di beberapa wilayah maka diperlukan strategi sebelum melakukan pendistribusian barang agar setiap konsumen dapat dilayani dengan cepat dan tepat. Selain itu strategi yang digunakan harus meminimumkan biaya distribusi dengan memperhatikan total jarak yang ditempuh dan kapasitas yang dimuat. Penentuan rute dengan mempertimbangkan jarak tempuh dari kendaraan dan kapasitas maksimum kendaraan pengangkut disebut *Vehicle Routing Problem* (VRP) **(Cordeu dkk, 2002)**.

Vehicle Routing Problem (VRP) adalah permasalahan pencarian rute terpendek dengan biaya minimal yang dimulai dari depot untuk melayani beberapa konsumen yang tersebar di wilayah yang berbeda. Tujuan VRP adalah untuk menentukan rencana rute yang meminimumkan total jarak, waktu, atau biaya dengan tidak melebihi kapasitas kendaraan untuk setiap rute yang didapatkan. Selain itu rute yang dibuat harus memenuhi syarat bahwa setiap pelanggan tepat dikunjungi satu kali oleh kendaraan **(Prins, 2009)**.

Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) adalah salah satu jenis VRP yang mempertimbangkan bahwa pelanggan membutuhkan proses pengiriman dan pengambilan barang secara bersamaan. Beberapa contoh kasus di dunia industri yaitu distribusi air minum dalam kemasan, distribusi *Liquefied Petrolieum Gas* (LPG), dan sebagainya **(Catay, 2010)**.

Para peneliti sebelumnya telah menerapkan beberapa algoritma untuk menyelesaikan permasalahan VRPSPD. Algoritma yang pernah digunakan antara lain *Genetic Algorithm* **(Serdar dan Mitsuo, 2012)** dan *Ant Colony System* **(Gajpal dan Abad, 2009)**.

Flower Pollination Algorithm (FPA) adalah algoritma metaheuristik yang terinspirasi dari proses penyerbukan bunga dan diperkenalkan oleh Xin She Yang pada tahun 2012. Di dalam proses *Flower Pollination Algorithm* (FPA) terdapat dua langkah kunci yaitu penyerbukan global dan penyerbukan lokal. Dengan jumlah iterasi yang sama *Flower Pollination Algorithm* (FPA) juga memiliki keunggulan berupa hasil yang lebih baik dibandingkan dengan *Genetic Algorithm* (GA) dan *Particle Swarm Optimizaion* (PSO) karena menggunakan *Lévy Flight* di dalam proses algoritmanya **(Yang, 2012).** Berdasarkan uraian yang telah dijelaskan, sangat menarik untuk menerapkan *Flower Pollination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem With Simultaneous Pickup and Delivery* (VRPSPD).

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka permasalahan yang akan di bahas adalah sebagai berikut:

1. Bagaimana menerapkan *Flower Pollination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)?

2. Bagaimana membuat program untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA)?
3. Bagaimana mengimplementasikan program *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA) pada contoh kasus?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Menerapkan *Flower Pollination Algorithm* (FPA) pada *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
2. Membuat program untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA).
3. Mengimplementasikan program *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA) pada contoh kasus.

1.4 Manfaat

Beberapa manfaat dari penelitian ini adalah:

1. Menambah informasi mahasiswa mengenai *Flower Pollination Algorithm* (FPA) dan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
2. Sebagai bahan referensi dalam perbandingan antara *Flower Pollination Algorithm* (FPA) dan algorithma lainnya dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
3. Mengetahui hasil dari implementasi program pada contoh kasus.

BAB II

TINJAUAN PUSTAKA

Pada bab ini akan diuraikan beberapa definisi yang digunakan pada pembahasan Penerapan *Flower Polination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

2.1. Graf

Pada bagian ini, definisi tentang graf diambil dari **Chartrand dan Oellerman (1993)**.

Definisi 2.1 *Graph G* didefinisikan sebagai himpunan berhingga $V(G)$ yang tak kosong dengan elemen-elemennya disebut titik (*vertice*) dan himpunan $E(G)$ yang mungkin kosong, yang elemennya terdiri dari pasangan dua elemen berbeda dari $V(G)$ dan disebut garis (*edge*).Kemudian $V(G)$ disebut sebagai himpunan titik dan $E(G)$ merupakan himpunan garis. Elemen dari $V(G)$ dinotasikan dengan v_i dan elemen dari E(G) dinotasikan dengan (v_i, v_j) atau dapat ditulis dengan $e = v_i v_j$. Jika terdapat garis e yang menghubungkan titik $v_i$dan v_j maka v_i dikatakan terhubung (*adjacent*) dengan v_j dalam hal ini v_i, v_j dikatakan insiden dengan e.

Definisi 2.2 Perjalanan (*walk)* dari *graph* G adalah urutan secara bergantian titik-titik yang merupakan elemen V(G) dan garis-garis elemen $E(G)$ yang berbentuk :

$$W: v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n, (n \geq 0)$$

yang dimulai dan diakhiri dengan titik, sedemikian sehingga $e_i = (v_{i-1}, v_i)$ untuk $i = 1,2, \dots, n$.

Definisi 2.3 Lintasan (*path)* adalah *walk* dengan titiknya tidak berulang (tidak ada titik yang diulang).

Definisi 2.4 Sikel (*cycle*) adalah *walk* $v_0, v_1, \dots, v_n$ yang mana $n \geq 3$ dengan $v_0 = v_n$ dan semua titiknya yaitu $v_1, v_2, \dots, v_n$ berbeda. Dengan kata lain sikel merupakan *walk* tertutup karena $v_0 = v_n$.

Definisi 2.5 Graph G dikatakan graf terhubung (*connected*) jika setiap dua titiknya dihubungkan oleh suatu *path*. Jika tidak maka graph G dikatakan graph tidak terhubung.

Definisi 2.6 Graph G dikatakan graph lengkap apabila setiap dua titiknya terhubung langsung (*adjacent*).

2.2. *Vehicle Routing Problem* (VRP)

Vehicle routing problem (VRP) adalah salah satu permasalahan untuk menentukan rute kendaraan yang berasal dan berakhir pada sebuah depot. Selain itu dalam permasalahan ini menggunakan beberapa kendaraan dengan muatan yang sama untuk melayani pelanggan dan setiap pelanggan dilayani tepat sebanyak satu kali.

(Solomon, 1998)

Tujuan dari VRP adalah :

1. Meminimumkan biaya pengiriman barang berdasarkan pada total jarak tempuh dengan batasan setiap pelanggan mendapatkan pelayanan tepat satu kali oleh satu kendaraan.
2. Meminimumkan jumlah kendaraan yang digunakan untuk melayani seluruh pelanggan.
3. Menyeimbangkan rute dengan waktu tempuh dan muatan kendaraan.

(Toth dan Vigo, 2002)

2.3. *Vehicle Routing Problem with Pickup and Delivery* (VRPPD)

Vehicle Routing Problem with Pickup and Delivery (VRPPD) merupakan perkembangan dari VRP dengan batasan tambahan yaitu kendaraan tidak hanya mengantarkan barang tetapi juga mengangkut barang di lokasi pelanggan. Tujuan dari VRPPD adalah meminimumkan biaya berdasarkan total jarak dengan batasan tidak melebihi kapasitas dari kendaraan.

VRPPD diklasifikasikan menjadi 3 kategori yaitu:

1. *Delivery First, Pickup second*

   Pada kategori ini pelanggan dibagi menjadi dua bagian yaitu *linehauls* (pelanggan yang menerima barang) dan *backhauls* (pelanggan yang mengirim barang). Kendaraan hanya boleh melakukan pengambilan barang sisa apabila telah melakukan seluruh aktivitas pengiriman barang.
2. *Mixed Pickup and Delivery*

   Pada kategori ini, pelanggan dibagi menjadi *linehauls* (pelanggan yang menerima barang) dan *backhauls* (pelanggan yang mengirim barang) dengan keduanya bisa berada dimana saja pada urutan rute kendaraan.
3. *Simultaneous Pickup and Delivery*

   Pada kategori ini, semua pelanggan secara bersamaan menerima dan memberikan barang sisa. Pada lokasi pelanggan, aktivitas pengiriman harus dilakukan terlebih dahulu.

(Nagy dan Salhi, 2005)

2.4. *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

Vehicle Routing Problem with Simultaneous Pckup and Delivery (VRPSPD) adalah pengembangan dari VRP. Dalam masalah ini yang harus diperhatikan adalah adanya aktivitas pengiriman sekaligus pengambilan barang pada pelanggan melalui sebuah kendaraan dengan kapasitas yang sama. Dengan

demikian, muatan pada kendaraan harus diperhatikan pada setiap pelanggan untuk memastikan kendaraan tidak melebihi muatan (*overload*). Tujuan dari permasalahan ini adalah meminimumkan total biaya berdasarkan total jarak tempuh untuk melayani seluruh pelanggan dengan batasan setiap pelanggan mendapatkan pelayanan tepat satu kali oleh satu kendaraan dan muatan kendaraan tidak melebihi kapasitas maksimalnya.

Dari model VRPSPD dapat diaplikasikan pada dunia industri, seperti distribusi air minum dalam kemasan, distribusi *Liqued Petrolium Gas* (LPG), dan lain sebagainya. Pelanggan dikunjungi untuk medapatkan pelayan ganda, misalnya dalam distribusi LPG. LPG yang terisi dikirim kepada pelanggan dan LPG yang kosong dibawa untuk diisi kembali.

(Catay, 2010)

Berikut ini adalah notasi yang akan digunakan pada model VRPSPD.

C_{ij} : Jarak antara pelanggan i dan j

J : Himpunan dari pelanggan yang harus dilayani

J_0 : Himpunan dari semua pelanggan termasuk depot

D_j : Jumlah permintaan yang harus dikirm ke pelanggan j

P_j : Jumlah pengambilan yang harus diambil dari pelanggan j

C : Kapasitas dari kendaraan

l_j : Jumlah beban suatu kendaraan setelah melayani pelanggan j

l_k : Beban suatu kendaraan saat meninggalkan depot

n : Banyaknya pelanggan

i : Indeks pelanggan awal

j : Indeks pelanggan tujuan

k : Indeks kendaraan

K : Himpunan dari kendaraan yang tersedia

π_j : Posisi dari pelanggan j

M : Angka yang sangat besar (M dapat dihitung sebagai nilai maksimum dari total permintaan pengiriman dan pengambilan atau total jarak dari setiap pelanggan)

x_{ijk} : Variabel biner yang menandakan perjalanan kendaraan k dari pelanggan i ke pelanggan j

Variabel keputusan yang digunakan adalah :

$$x_{ijk} = \begin{cases} 1, & bila\ kendaraan\ k\ melayani\ j\ setelah\ melayani\ i \\ 0, & bila\ tidak\ demikian \end{cases}$$

Fungsi tujuan dari VRPSPD dapat dimodelkan dalam bentuk sebagai berikut :

$$z = min \sum_{i \in J_o} \sum_{j \in J_o} \sum_{k \in K} c_{ij} x_{ijk} \tag{2.1}$$

Dengan kendala sebagai berikut:

1. Setiap pelanggan hanya dilayani tepat satu kendaraan:

$$\sum_{i \in J_o} \sum_{k \in K} x_{ijk} = 1 \qquad j \in J \tag{2.2}$$

2. Setiap kendaraan harus meninggalkan pelanggan yang telah dikunjungi (r = indeks pelanggan):

$$\sum_{i \in J_o} x_{irk} = \sum_{j \in J_o} X_{rjk} \qquad r \in J, k \in K \tag{2.3}$$

3. Total jumlah permintaan yang diambil dari semua lokasi yang dikunjungi kendaraan k tidak boleh melebihi kapasitas maksimum kendaraan k:

$$l_k = \sum_{i \in J_o} \sum_{j \in J} D_j\, x_{ijk} \qquad k \in K \tag{2.4}$$

4. Jumlah beban permintaan pengiriman dan pengambilan pada pelanggan j oleh kendaraan k tidak melebihi sisa kapasitas kendaraan:

$$l_j \geq l_k^i - D_j + P_j - M\left(1 - x_{0jk}\right) \qquad j \in J, k \in K \tag{2.5}$$

5. Jumlah beban permintaan pengiriman dan pengambilan pada pelanggan j oleh kendaraan k tidak melebihi sisa kapasitas kendaraan k setelah melayani pelayan i:

$$l_j \geq l_i - D_j + P_j - M\left(1 - \sum_{k \in K} x_{ijk}\right) \qquad i \in J, j \in J, i \neq j \tag{2.6}$$

6. Jumlah beban dari kendaraan k tidak melebihi kapasitas kendaraan itu sendiri:

$$l_k \leq C \qquad k \in K \tag{2.7}$$

$$l_j \leq C \qquad j \in J \tag{2.8}$$

7. Setiap kendaraan hanya mempunyai satu rute kendaraan:

$$\pi_j \geq \pi_i + 1 - n(1 - \sum_{k \in K} x_{ijk}) \qquad i \in J, j \in J, i \neq j \tag{2.9}$$

8. Ururtan pelayanan pelanggan j tidak boleh bernilai negatif:

$$\pi_j \geq 0 \qquad j \in J \tag{2.10}$$

(Dethloff, 2001)

2.5. *Flower Pollination Algorithm* (FPA)

Algoritma penyerbukan bunga pertama kali diperkenalkan oleh Xin She Yang pada tahun 2012. Algoritma penyerbukan bunga adalah sebuah metode yang dihasilkan melalui pengamatan terhadap penyerbukan tanaman berbunga. Algoritma ini disesuaikan dengan penyerbukan bunga yang sebenarnya yaitu dengan konsep penyerbukan yang terdiri atas dua jenis yaitu penyerbukan biotik dan penyerbukan abiotik.

Ada empat pendekatan yang digunakan dalam *Flower Pollination Algorithm* (FPA) dirumuskan sebagai berikut:

1. Penyerbukan biotik dan penyerbukan silang dipandang sebagai penyerbukan global dan *pollinator* bergerak menurut *Lévy Flight*.
2. Penyerbukan abiotik dan penyerbukan sendiri dipandang sebagai penyerbukan lokal.

3. *Pollinator* seperti serangga dapat membantu penyerbukan secara komstan, setara dengan probabilitas reproduksi yang proporsional antara dua bunga.
4. Pertukaran antara penyerbukan lokal dan penyerbukan global dapat di kontrol dengan probabilitas pertukaran $p_a \in [0, 1]$.

Terdapat dua langkah kunci dalam algoritma ini, yaitu penyerbukan global dan penyerbukan lokal. Berikut dua langka kunci pada *Flower Pollination Algorithm* (FPA):

i. Penyerbukan Lokal

Pada langkah penyerbukan lokal menggunakan fenomena *flower constancy* dapat direpresentasikan secara matematis sebagai berikut:

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{2.11}$$

dengan keterangan:

x_i^t = bunga ke$-i$ pada iterasi ke t

x_i^{t+1} = bunga ke$-i$ pada iterasi ke $t + 1$

ϵ = bilangan real acak yang berdistribusi uniform antara 0 dan 1

j, k = indeks dua bunga yang berbeda

.

ii. Penyerbukan Global

Pada langkah penyerbukan global, serbuk sari dari bunga dibawa oleh hewan penyerbuk seperti serangga, dan serbuk sari dapat melakukan perjalanan jarak jauh karena serangga dapat terbang dan bergerak di daerah yang luas. Proses ini bisa menghasilkan solusi yang paling optimal yang direpresentasikan dengan aturan pertama, ditambah dengan fenomena *flower constancy* dapat direpresentasikan secara matematis sebagai berikut:

$$x_i^{t+1} = x_i^t + \alpha L(\lambda)(x_{best} - x_i^t) \tag{2.12}$$

dengan keterangan:

x_i^t = bunga ke$-i$ pada iterasi ke t

x_i^{t+1} = bunga ke$-i$pada iterasi ke $t + 1$

α = parameter pengontrol *stepsize*

$L(\lambda)$ = *stepsize* atau jarak terbang yang berbasis distribusi *Lévy*.

$xbest$ = bunga solusi terbaik pada iterasi yang berlangsung

(Yang, 2012)

karena serangga dapat berpindah jarak jauh dengan berbagai langkah tempuh, maka dapat menggunakan *Lévy Flight* (L) untuk meniru karakteristik serangga tersebut dengan distribusi *Lévy* sebagai berikut:

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi.\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}} \tag{2.13}$$

dengan $\Gamma(\lambda)$ adalah fungsi gamma standar, dan distribusi ini berlaku untuk *steplengths* > 0. Menurut **Mantegna (1994)**, *step length* (S) dapat dihitung dengan menggunakan:

$$S = \frac{U}{|V|^{\frac{1}{\lambda}}} \tag{2.14}$$

dengan λ merupakan parameter pada interval $[1,2]$, U adalah bilangan real acak yang berdistribusi normal dengan mean 0 dan varians σ^2. Menurut **Vasant, Weber dan Dieu (2016)** nilai U dapat didekati dengan bilangan real acak yang berdistribusi normal $N(0,1)$ dikali dengan σ. V adalah bilangan real acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1. Persamaan (2.14) dapat ditulis menjadi:

$$S = \frac{u\sigma}{|v|^{\frac{1}{\lambda}}} \tag{2.15}$$

nilai σ ditentukan berdasarkan:

$$\sigma^2 = \left(\frac{\Gamma(1+\lambda).\sin(\pi.\lambda/2)}{\Gamma\left(\frac{1+\lambda}{2}\right).\lambda.2^{\left(\frac{\lambda-1}{2}\right)}}\right)^{\frac{1}{\lambda}} \tag{2.16}$$

dengan Γ merupakan fungsi gamma. Menurut Diethelm (2004), fungsi $\Gamma: (0,\infty) \rightarrow \mathbb{R}$, didefinisikan sebagai:

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx \tag{2.17}$$

(Yang, 2012)

2.6. Langkah-langkah pada *Flower Pollination Algorithm* (FPA)

Langkah-langkah standar untuk menerapkan *Flower Pollination Algorithm* (FPA) adalah sebagai berikut:

a. Mendefinisikan fungsi tujuan $f(x_i), x_i = (x_1, x_2, \dots, x_n)^T$.
b. Membangkitkan populasi sebanyak n bunga $x_{i,j}$ (i $= 1,2, \dots, n$).
c. Menghitung fungsi tujuan $f(x_i)$.
d. Menemukan solusi terbaik $(xbest)$ sementara pada populasi awal.
e. Membangkitkan bilangan real acak sejumlah bunga (r_i), kemudian mendefinisikan switch probability $p_a \in [0,1]$.
f. Jika r_i lebih kecil dari p_a, maka akan melakukan penyerbukan global dengan Lévy Flights.
g. Jika r_i lebih besar atau sama dengan p_a, maka akan melakukan penyerbukan lokal dengan ϵ berdistribusi uniform.
h. Membangkitkan bunga baru (x_j) dari penyerbukan lokal dan penyerbukan global
i. Mengevaluasi/menghitung fungsi tujuan hasil penyerbukan $f(x_j)$.
j. Melakukan seleksi dengan memilih i diantara n bunga secara acak. Jika $f(x_j) < f(x_i)$ maka mengganti baris ke-i dengan solusi baru.
k. Urutkan bunga berdasarkan nilai fungsi tujuannya dan tentukan bunga terbaiknya (xbest)
l. Tentukan dan simpan bunga terbaiknya.

(Yang, 2012)

2.7. Pengkodean

Berdasarkan **Obitko (1998)**, pengkodean (*encoding*) merupakan suatu cara menyajikan suatu solusi. Ada beberapa jenis pengkodean, diantaranya:

1. Pengkodean *Biner*

   Dalam pengkodean *biner*, setiap individu disimbolkan dengan dua bilangan yaitu 0 dan 1.
2. Pengkodean Nilai

   Dalam pengkodean nilai, setiap individu disimbolkan dengan rangkaian beberapa nilai.
3. Pengkodean Permutasi

   Dalam pengkodean permutasi, setiap individu disimbolkan dengan untaian angka yang menggambarkan angka dalam barisan.

BAB III

METODE PENELITIAN

Dalam menyelesaikan permasalahan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA) diperlukan langkah-langkah sebagai berikut:

1. Mengkaji pustaka tentang *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dan *Flower Pollination Algorithm* (FPA).
2. Menentukan parameter *Vehicle Rouitng Problem with Simultaneous Pickup and Delivery* (VRPSPD) yang akan digunakan dalam menyelesaikan permasalahan *Vehicle Routing Problem with Simultaneous Pickup and Deliver* (VRPSPD).
3. Menginputkan banyaknya pelanggan, posisi depot, posisi pelanggan, jumlah permintaan pengiriman barang, jumlah barang yang diambil pada masing-masing pelanggan, kapasitas maksimum kendaraan dan kapasitas yang diangkut dari depot.
4. Prosedur untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA) adalah sebagai berikut:
    a. Menentukan parameter *Flower Pollination Algorithm* (FPA) yaitu banyak bunga (n), iterasi maksimal, *switch probability* (p_a), pengontrol *stepsize* (α) dan nilai λ.
    b. Membangkitkan posisi awal bunga
        i. Membangkitkan bilangan real secara acak pada masing-masing bunga sebanyak jumlah pelanggan.
        ii. Mengurutkan bilangan realacak sehingga diperoleh urutan bilangan acak.
        iii. Menampilkan urutan sesuai dengan bilangan real acak awal.
        iv. Menentukan rute yang dilalui kendaraan dengan kendala antara lain : setiap perjalanan harus berawal dan berakhir di

depot dan jumlah permintaan sekaligus pengambilan barang dalam satu rute tidak boleh melebihi kapasitas maksimum kendaraan.

c. Mengevaluasi fungsi tujuan dengan menghitung total jarak yang dilalui oleh kendaraan (F_i).
d. Menentukan solusi terbaik ($xbest$) sementara dari semua bungadengan mengurutkan nilai fungsi tujuan dari terkecil sampai terbesar.
e. Membangkitkan bilangan real pada interval (0,1) secara acak untuk setiap bunga (r_i), kemudian membandingkan r_i dengan *switch probability* (p_a) $\in$ [0,1]. Jika r_i lebih kecil dari p_a, maka akan melakukan penyerbukan global dengan *Lévy Flights*:
    i. Mendapatkan nilai u dan V dengan membangkitkan bilangan real secara acak berdistribusi normal N (0,1) sejumlah pelanggan.
    ii. Menghitung nilai σ.
    iii. Menghitung *step length* (S) dengan membagi setiap elemen $u$dengan setiap elemen v dipositifkan dan dipangkatkan $\frac{1}{\lambda}$ kemudian mengalikannya dengan nilai σ.
    iv. Menghitung nilai L
    v. Menghitung bunga x_i baru dengan mengalikan L dan α dengan hasil pengurangan nilai solusi terbaik ($xbest$) sementara dengan nilai bunga, kemudian dijumlahkan dengan nilai bunga sekarang.
    vi. Mengevaluasi fungsi tujuan bunga x_i baru dengan menghitung total jarak dari hasil proses penyerbukan.
    vii. Update solusi dengan membandingkan hasil nilai fungsi tujuan baru dengan nilai fungsi tujuan yang lama. Jika nilai fungsi tujuan baru lebih baik, maka nilai fungsi tujuan yang

lama digantikan dengan nilai fungsi tujuan yang baru. Jika tidak, gunakan nilai fungsi tujuan lama.

Jika r_i lebih besar atau sama dengan p_a, maka akan melakukan penyerbukan lokal dengan ϵ berdistribusi uniform:

i. Membangkitkan ϵ, bilangan real secara acak berdistribusi uniform $U(0,1)$ sejumlah pelanggan.
ii. Menghitung x_i baru dengan mengalikan nilai ϵ dengan hasil pengurangan x_j dengan x_k kemudian dijumlahkan dengan nilai bunga sekarang x_i.
iii. Mengevaluasi fungsi tujuan bunga x_i baru dengan menghitung total jarak dari hasil proses penyerbukan.
iv. Update solusi dengan membandingkan hasil nilai fungsi tujuan baru dengan nilai fungsi tujuan yang lama. Jika nilai fungsi tujuan baru lebih baik, maka nilai fungsi tujuan yang lama digantikan dengan nilai fungsi tujuan yang baru. Jika tidak, gunakan nilai fungsi tujuan lama.

f. Menentukan solusi terbaik $(xbest)$ dengan mengurutkan nilai fungsi tujuan dari terkecil sampai terbesar.
g. Mengulangi proses e sampai h hingga iterasi maksimal.
h. Mencetak hasil solusi terbaik $(xbest)$.

5. Membuat program dari langkah-langkah metode penelitian diatas.
6. Mengimplementasikan program pada contoh kasus.

BAB IV

PEMBAHASAN

Pada bagian ini akan dijelaskan mengenai prosedur *Flower Pollination Algorithm* (FPA) dalam mencari rute optimal pada *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD). *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) merupakan penentuan rute optimal dalam melayani seluruh pelanggan dengan jumlah permintaan dan pengambilan barang yang berbeda-beda serta melibatkan lebih dari satu kendaraan. Rute yang dibentuk dimulai dan diakhiri pada satu tempat yang sama yaitu depot. Pelanggan dikunjungi hanya satu kali dan total permintaan dari semua pelanggan dalam satu rute tidak melebihi kapasitas maksimum kendaraan. Pelayanan pertama yang dilakukan kendaraan yaitu menurunkan sejumlah barang sesuai permintaan pelanggan kemudian mengangkut sejumlah barang dari pelanggan.

4.1 Prosedur *Flower Pollination Algorithm* (FPA) untuk Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery (*VRPSPD)

Proses Penyelesaian *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Flower Pollination Algorithm* (FPA) dimulai dengan menginputkan data yang terdiri dari data jarak masing-masing pelanggan, data permintaan dan pemngambilan setiap pelanggan serta menginisialisasi parameter *Flower Pollination Algorithm*. Setelah itu dilakukan prosedur *Flower Pollination Algorithm* sehingga diperoleh bunga terbaik yang akan menjadi solusi dari permasalahan VRPSPD. *Flower Pollination Algorithm* akan berhenti jika mencapai iterasi maksimum.

Untuk mempermudah ilustrasi penyelesaian *Vehicle Routing Problem with Simultaneous Pickup and Delivery* menggunakan *Flower Pollination Algorithm*, maka prosedur diatas pada **Gambar 4.1**

Prosedur *Flower Pollination Algorithm* untuk menyelesaikan VRPSPD
Begin
input data ();
inisialisasi parameter ();
membangkitkan posisi awal bunga pada posisi $x_{i,j}(i = 1,2, \dots, n\ dan\ j = 1,2,..,m)$ ();
mengevaluasi fungsi tujuan awal $f(x_i)$();
menentukan x_{best} sementara ();
while (iterasi ≤ MaxIterasi)
for i = 1 to n
mendapatkan bilangan real secara acak sebanyak bunga (r_i)
if $(r_i < P_a)$
proses penyerbukan global bunga ke-*i* ( );
else
proses penyerbukan lokal bunga ke-i ( );
end if
menentukan x_{best} ;
end for
menyimpan solusi terbaik;
end while
end

Gambar 4.1 Prosedur *Flower Pollination Algorithm* untuk Menyelesaikan VRPSPD

$x_{i,j}$ adalah posisi bunga ke-i pelanggan ke-j sehingga n menandakan banyak bunga dan m menandakan banyaknya pelanggan. Prosedur selengkapnya dijelaskan pada sub bab berikut ini :

4.1.1 Input Data

Prosedur untuk input data yang digunakan dalam menyelesaikan VRPSPD menggunakan *Flower Pollination Algorithm* dapat dilihat pada **Gambar 4.2**

```
Prosedur Input data (m)
Begin
    for i = 0 to m
        input permintaan (D_i);
        input pengambilan (P_i);
    end for i
    for i to m
        for j to m
            input jarak (J_i,j);
        end for j
    end i
end
```

Gambar 4.2 Prosedur Input Data

Permintaan (D_i) adalah permintaan jumlah barang dari pelanggan ke-*i*, sedangkan pengambilan (P_i) adalah pengambilan jumlah barang dari pelanggan ke-*i*. Selain itu terdapat $(J_{i,j})$ yang merupakan jarak dari pelanggan *i* ke pelanggan *j*. Data yang diinputkan adalah data jarak masing-masing pelanggan dan depot, data permintaan (d*emand*) dan data pengambilan (*pickup*) dari setiap pelanggan yang bertipe .txt.

4.1.2 Inisialisasi Parameter

Parameter yang digunakan dalam *Flower Pollination Algorithm* adalah banyak bunga (n), *MaxIterasi* atau iterasi maksimum yang merupakan kriteria pemberhentian algoritma, *stepsize* (α), nilai λ, *swich probability* (p_a) dan presentase barang yang diangkut dari depot (*presentase*). Prosedur untuk inisialisasi parameter disajikan dalam **Gambar 4.3.**

```
Prosedur Inisialisasi Parameter
Begin
    n ← input ("Masukan banyak bunga > 2");
    MaxIterasi ← input ("Masukan maksimum iterasi > 0");
    α ← input ("Masukan pengontrol stepsize [0,1]");
    λ ← input ("Masukan nilai lamda [1,2]");
    P_a ← input ("Masukan switch probability [0,1]");
    prsentase ← input ("Masukan presentase barang yang diangkut [0,100]");
end
```

Gambar 4.3 Prosedur Inisialisasi Parameter

Setelah input dan inisialisasi parameter langkah selanjutnya adalah membangkitkan posisi awal bunga.

4.1.3 Membangkitkan Posisi Awal Bunga

Pada proses ini, langkah yang dilakukan adalah membangkitkan bilangan *real* secara acak sejumlah pelanggan. Pembangkitan dilakukan berulang sebanyak jumlah bunga sehingga akan membentuk calon solusi. Nilai yang dibangkitkan dari setiap solusi merupakan bilangan real pada interval (a,b). Di dalam pembahasan ini batas bilangan *real* yang digunakan adalah $a = 0$ dan $b = 1$. Prosedur membangkitkan posisi awal bunga disajikan dalam **Gambar 4.4**

```
Prosedur Membangkitkan Posisi Awal Bunga (n, banyak pelanggan)
Begin
    for i = 1 to n
        for j = 1 to banyak pelanggan
            x_{i,j} ← random (a, b)
        end for j
    end for i
end
```

Gambar 4.4 Prosedur Membangkitkan Posisi Awal Bunga

$x_{i,j}$ adalah bilangan *real* acak pada interval (0,1) dari bunga ke-*i* pelanggan ke-*j* yang akan membentuk calon solusi. Setelah mendapatkan solusi awal bunga, langkah berikutnya adalah mengevaluasi fungsi tujuan dari masing-masing bunga.

4.1.4 Mengevaluasi Nilai Fungsi Tujuan

Evaluasi pada masing-masing bunga dilakukan dengan menghitung fungsi tujuan. Fungsi tujuan masing-masing bunga didapatkan dari nilai total jarak tempuh kendaraan pada masing-masing rute yang berawal dan berakhir pada depot tanpa melanggar batasan kendala kapasitas maksimum kendaraan. Untuk dapat menghitung fungsi tujuan tersebut, perlu adanya konversi posisi awal setiap bunga dari bilangan *real* ke dalam bentuk kode permutasi. Setiap elemen dari posisi bunga ditransformasikan menjadi urutan pelanggan dengan cara menggurutkan elemen-elemen tersebut berdasarkan nilai terkecil hingga yang paling besar. Cara tersebut hanya digunakan untuk memberikan nomor urutan pada elemen-elemen tersebut tanpa memindahkan kolom dari elemen posisi. Sehingga diperoleh urutan pelanggan yang merupakan calon solusi. Prosedur menentukan kode permutasi selengkapnya disajikan pada **Gambar 4.5.**

```
Prosedur menentukan kode permutasi (n, m, x )
Begin
    for i = 1 to n
        for j = 1 to m
            k ← 1;
            for l = 1 to m
                    if (x_{i,j} > x_{i,j})
                        k ← k+1;
                    end if
                        px_{i,j} ← k;
            end for l
        end for j
    end for i
end
```

Gambar 4.5 Prosedur Menentukan Kode Permutasi

Dalam representasi permutasi sebelumnya didapatkan posisi awal bunga merupakan urutan pelanggan atau disebut calon rute. Calon rute inilah yang akan dibentuk rute tanpa melanggar kapasitas maksimum kendaraan, sehingga dapat meminimumkan total jarak tempuh dalam melayani seluruh permintaan dan pengambilan barang dari pelanggan. Prosedur untuk menghitung nilai fungsi tujuan VRPSPD disajikan dalam **Gambar 4.6.**

Prosedur menghitung fungsi tujuan VRPSPD (n, banyak pelanggan)
Begin
for i = 1 to n
$jumlah\ permintaan\ (JP_0) \leftarrow kapasitas\ awal$;
$beban\ barang\ saat\ ini\ (B_0) \leftarrow kapasitas\ awal$;
$jarak \leftarrow 0$;
$total \leftarrow 0$
for j = 0 to m-1
$JP_{j+1} = JP_j - D_{px_{i,J+1}}$;
$B_{j+1} = B_j - D_{px_{i,j+1}} + P_{px_{i,j+1}}$;
$jarak = jarak + J_{i,px_{j+1}}$;
$if\ \Big(JP_{j+1} < D_{px_{i,j+1}} || B_{j+1} - D_{px_{i,j+2}} + P_{px_{i,j+2}} > maxkapasitas || j = pelanggan - 1\Big)$
$JP_{j+1} = kapasitas\ awal$;
$B_{j+1} = kapasitas\ awal$;
$tambahan = J_{j+1,0}$;
$jarak = jarak + tambahan$;
$total = total + jarak$;
end if
end for j
$f(x_i) = total$;
end for i
end

Gambar 4.6 Prosedur Meghitung Nilai Fungsi Tujuan

Pada perhitungan nilai fungsi tujuan, JP_i merupakan sisa jumlah permintaan setelah melayani pelanggan ke-i, B_i merupakan beban yang dimuat kendaraan setelah melayani pelanggan ke-i. sedangkan $f(x_i)$ adalah nilai fuungsi tujuan dari bunga ke-i. Nilai fungsi tujuan masing-masing bunga ke-i inilah yang akan diminimalkan, dengan harapan dapat diperoleh nilai fungsi tujuan yang minimal.

4.1.5 Menentukan bunga terbaik sementara

Pada proses ini, langkah yang dilakukan adalah menentukan bunga terbaik sementara dengan mencari nilai fungsi terkecil dari setiap bunga. Prosedur menentukan bunga terbaik disajikan pada **Gambar 4.7.**

Prosedur menentukan buunga terbaik sementara (n)
Begin
  $xbest = 5000$;
  for i = 1 to n
    $if(f(x_i) < gbest)$
      $xbest = f(x_i)$;
      $indeks = i$;
    end if
  end for i
end

Gambar 4.7 Menentukan Bunga Terbaik Sementara

Posisi terbaik yang ditentukan berdasarkan prosedur dalam **Gambar 4.7**, dinyatakan sebagai $xbest$ sementara. Setelah nilai $xbest$ sementara didapatkan, langkah selanjutnya adalah memperbarui bunga dengan penyerbukan bunga yaitu penyerbukan global dan penyerbukan lokal.

4.1.6 **Memperbarui bunga dengan penyerbukan global**

Bunga yang mengalami penyerbukan global adalah bunga yang memiliki nilai R_i kurang dari *switch probability* (P_a), dengan R_i adalah bilangan real yang dibangkitkan secara acak. Pada proses memperbarui bunga dengan penyerbukan global diperlukan tahap awal yaitu menentukan nilai u, V, S dan L. Prosedur memperbarui bunga penyerbukan global disajikan pada **Gambar 4.8.**

Prosedur penyerbukan global (n, banyak pelanggan, x)
Begin
for *i* = 1 to n
$for\ j$ = 1 to banyak pelanggan
$u_{i,j} \leftarrow random(0,1)$
$V_{i,j} \leftarrow random(0,1)$
$\sigma \leftarrow \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left(\frac{1+\lambda}{2}\right)}\frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{\frac{\lambda-1}{2}}}\right]^{1/\lambda}$;
$s_{i,j} \leftarrow \frac{\sigma.u_j}{|V_j|^{1/\lambda}}$;
$L_{i,j} \leftarrow \frac{\lambda\Gamma(\lambda)\sin(\frac{\lambda\pi}{2})}{\pi}\frac{1}{s_{i,j}^{1/\lambda}}$;
$x_{i,j}\ baru \leftarrow x_{i,j} + \alpha L(x_{gbest,j} - x_{i,j})$;
end for *j*
if $(f(x_i) > f(x_i)\ baru)$
$f(x_i) \leftarrow f(x_i)\ baru$
for *j* = 1 to banyak pelanggan
$x_{i,j} \leftarrow x_{i,j}\ baru$;
end for *j*
end if
end for *i*
end

Gambar 4.8 Proses Memperbarui Bunga dengan Penyerbukan Global

Langkah di atas menghasilkan fungsi tujuan baru dari proses penyerbukan lokal.

4.1.7 Memperbarui bunga dengan penyerbukan lokal

Bunga yang mengalami penyerbukan lokal adalah bunga yang memiliki nilai r_i lebih dari atau sama dengan *switch probability*. Pada proses memperbarui bunga dengan penyerbukan lokal diperlukan tahap awal yaitu menentukan dua bunga secara acak dan menentukan nilai ε. Prosedur memperbarui bunga penyerbukan lokal disajikan pada **Gambar 4.9.**

Prosedur penyerbukan lokal (n, banyak pelanggan, x)
Begin
  for i = 1 to n
    $for\ j$ = 1 to banyak pelanggan
      $b_1 \leftarrow random(n)$;
      $b_2 \leftarrow random(n)$;
      $\varepsilon \leftarrow random(0,1)$;
      $x_{i,j}\ baru \leftarrow x_{i,j} + \varepsilon(x_{b_1,j} - x_{b_2,j})$;
    end for j
    if $(f(x_i) > f(x_i)\ baru)$
      $f(x_i) \leftarrow f(x_i)\ baru$
      for j = 1 to banyak pelanggan
        $x_{i,j} \leftarrow x_{i,j}\ baru$;
      end for j
    end if
  end for i
end

Gambar 4.9 Proses Memperbarui Bunga dengan Penyerbukan Lokal

Langkah di atas menghasilkan fungsi tujuan baru dari proses penyerbukan lokal. Setelah memperbarui semua bunga dengan penyerbukan, langkah selanjutnya adalah menentukan bunga terbaik di setiap iterasi.

4.1.8 Menentukan bunga terbaik setiap iterasi

Penentuan bunga terbaik ini menggunakan informasi bunga baru sehingga diperoleh fungsi tujuan bunga baru. Bunga terbaik yang dinotasikan *best* diperoleh dari fungsi tujuan terkecil untuk setiap bunga. Proses ini dilakukan setiap iterasi, sehingga akan diperoleh bunga terbaik setiap iterasinya. Prosedur penentuan bunga terbaik setiap iterasi dapat dilihat pada **Gambar 4.7**

Pembahasan selanjutnya adalah data yang akan digunakan dalam perhitungan manual dan implementasi program.

4.2 Data

Data yang digunakan data sekunder dengan tiga tipe data, yaitu :

1. Data dengan 13 Pelanggan

   Data kecil ini diambil dari **Mohandas dkk (2008)** yang berisi 13 pelanggan dengan banyak kendaraan 5 serta kapasitas maksimum kendaraan 180 barang. Data lengkap dapat dilihat pada **Lampiran 4.**
2. Data dengan 22 Pelanggan

   Data berukuran sedang ini diambil dari **Min (1989)** yang berisi 22 pelanggan dengan banyak kendaraan 5 serta kapasitas maksimal kendaraan 10500 barang. Data lengkap dapat dilihat pada **Lampiran 5.**
3. Data dengan 100 Pelanggan

   Data berukuran besar ini diambil dari **Solomon dan Desrosiers (1988)** yang berisi 100 pelanggan dengan banyak kendaraan 5 serta kapasitas maksimal kendaraan 700 barang. Data lengkap dapat dilihat pada **Lampiran 6.**

4.3 Penyelesaian Secara Manual untuk Contoh Kasus VRPSPD Menggunakan Data dengan 13 Pelanggan

Langkah-langkah yang digunakan dalam menyelesaikan VRPSPD secara manual dengan data pada **Lampiran 4** menggunakan *Flower Pollination Algorithm* adalah sebagai berikut :

Langkah I : Input Data dan Inisialisasi Parameter

Parameter yang digunakan dalam penyelesaian VRPSPD secara manual ini adalah sebagai berikut : banyak bunga (n) adalah 5, banyak pelanggan adalah 13, *MaxCapacity* adalah 180, presentase jumlah barang yang diangkut dari depot 50% adalah 90, *MaxVehicle* adalah 4, *MaxIteration* adalah 1, *stepsize* (α) adalah 0.01, probabilitas penyerbukan bunga (P_a) adalah 0.25 dan nilai λ =adalah 1.5 yang berada pada interval [1,2].

Langkah II : Membangkitkan Posisi Awal Bunga

Membangkitkan posisi awal sarang dimulai dengan melakukan pembangkitan elemen-elemen bunga sebanyak pelanggan yang terdiri atas bilangan real yang dibangitkan secara acak pada interval (a,b). Pada kasus batas bilangan real yang digunakan adalah a = 0 dan b = 1. Proses tersebut dilakukan sebanyak bunga. Hasil pembangkitan posisi awal bunga selengkapnya disajikan dalam **Tabel 4.1.**

Tabel 4.1 Posisi Awal Bunga

Bunga	Pelanggan						
	1	2	3	4	5	…	13
$Bunga_1$	0.2277	0.9234	0.9049	0.1111	0.5949	…	0.4889
$Bunga_2$	0.4357	0.4302	0.9797	0.2581	0.2622	…	0.6241
$Bunga_3$	0.3111	0.1848	0.4389	0.4087	0.6028	…	0.6791
$Bunga_4$	0.8147	0.1270	0.6324	0.2785	0.9575	…	0.6787
$Bunga_5$	0.9058	0.9134	0.0975	0.5469	0.9649	...	0.7577

Pada **Tabel 4.1** menampilkan hasil pembangkitan bilangan real secara acak yang mewakili posisi awal setiap bunga. Setiap bunga memiliki posisi awal dengan elemen sejumlah pelanggan, sehingga posisi bunga terdiri dari 13 elemen berisi

bilangan real pada interval (0,1) yang diperoleh secara acak. Posisi awal bunga selengkapnya dapat dilihat pada **Lampiran 7.**

Langkah III : Menghitung fungsi tujuan

Pada langkah ini dilakukan evaluasi pada setiap bunga dengan menghitung fungsi masing-masing bunga ($f(x_i^t)$). Sebelum mendapatkan nilai fungsi tujuan, maka harus dilakukan pengurutan bilangan real acak. Proses ini dilakukan dengan cara memberikan nomor urut pada setiap elemen bunga yang mewakili posisi dimulai dari yang terkecil hingga terbesar. Hasil pengurutan posisi setiap bunga terdapat pada **Tabel 4.2.**

Tabel 4.2 Hasil Pengurutan Posisi Setiap Bunga

Bunga	Calon Rute												
$Bunga_1$	2	13	12	1	8	9	4	6	11	10	3	7	5
$Bunga_2$	8	7	13	4	5	3	6	2	1	10	9	11	12
$Bunga_3$	5	2	8	6	10	1	7	4	12	9	13	3	11
$Bunga_4$	10	1	5	3	13	2	12	9	4	8	6	11	7
$Bunga_5$	7	8	2	5	12	13	4	3	9	11	1	10	6

Hasil pengurutan bilangan real secara acak pada **Tabel 4.2** mempresentasikan pelanggan serta sebagai langkah sebelum membentuk rute. Oleh karena itu, urutan bilangan real secara acak disebut juga sebagai calon rute.

Setelah memperoleh calon rute, akan dibentuk rute. Sesuai prinsip VRPSPD bahwa setiap rute kendaraan harus berawal dan berakhir di depot (dalam pembahasan ini depot disimbolkan dengan 0) serta muatan setiap kendaraan (dalam satu rute) tidak boleh melebihi kapasitas maksimum kendaraan. Sebagai contoh calon rute $bunga_1$ adalah :

0 – 2 – 13 – 12 – 1 – 8 – 9 – 4 – 6 - 11 – 10 – 3 – 7 – 5 – 0

Pembentukan rute pada bunga 1 dimulai dari depot menuju pelanggan 2, pelanggan 13, pelanggan 12 dan seterusnya. Lalu menjumlahkan permintaan dan

pengambilan setiap pelanggan tersebut dan rute terbentuk setelah kapasitas maksimum kendaraan terpenuhi atau jumlah total permintaan dalam rute tersebut kurang dari permintaan pelanggan selanjutnya.

0-2

Pada pelanggan 12, kendaraan harus menurunkan 30 unit barang dan mengangkut 35 unit barang kosong, sehingga sisa permintaan barang pada kendaraan tersisa 90 - 30 = 60 unit barang dan total beban 90 - 30 + 35 = 95 unit barang. Jumlah tersebut belum melebihi kapasitas maksimum kendaraan, maka kendaraan melanjutkan ke rute selanjutnya, yaitu menuju ke pelanggan 13.

0-2-13

Pada pelanggan 13, kendaraan harus menurunkan 25 unit barang dan mengangkut 40 unit barang kosong, sehingga sisa permintaan barang pada kendaraan tersisa 60 – 25 = 35 unit barang dan total beban 95 – 25 + 40 = 110 unit barang. Jumlah tersebut belum melebihi kapasitas maksimum kendaraan, maka kendaraan melanjutkan ke rute selanjutnya, yaitu menuju ke pelanggan 12.

0-2-13-12

Pada pelanggan 12, kendaraan harus menurunkan 25 unit barang dan mengangkut 35 unit barang kosong, sehingga sisa permintaan barang pada kendaraan tersisa 35 – 25 = 10 unit barang dan total beban 110 – 25 + 35 = 120 unit barang. Jumlah tersebut belum melebihi kapasitas maksimum kendaraan, maka kendaraan melanjutkan ke rute selanjutnya, yaitu menuju ke pelanggan 1. Karena permintaan di pelanggan 1 25 unit barang, sedangkan sisa barang hanya 10 unit maka kendaraan kembali ke depot dan pelanggan 1 dilayani oleh kendaraan berikutnya. Sehingga rutenya menjadi:

0 – 2 – 13 – 12 – 0

Ulangi langkah diatas sehingga setiap kota dilayani tepat satu kali. Hasil pembentukan rute keseluruhan bunga disajikan dalam **Tabel 4.3.**

Tabel 4.3 Pembentukan Rute posisi awal bunga

Bunga	Rute yang terbentuk	Total Jarak
$Bunga_1$	0-2-13-12-0	29
	0-1-8-9-0	33
	0-4-6-11-0	41
	0-10-3-7-0	42
	0-5-0	30
	Fungsi tujuan $bunga_1$	175
$Bunga_2$	0-8-7-0	32
	0-13-4-0	37
	0-5-3-6-2-0	48
	0-1-10-9-0	32
	0-11-12-0	16
	Fungsi tujuan $bunga_2$	165
$Bunga_3$	0-5-2-8-0	41
	0-6-10-1-0	38
	0-7-4-0	48
	0-12-9-13-0	23
	0-3-11-0	24
	Fungsi tujuan $bunga_3$	174
$Bunga_4$	0-10-1-5-0	53
	0-3-13-2-0	43

	0-12-9-0	21
	0-4-8-0	41
	0-6-11-7-0	53
	Fungsi tujuan $bunga_4$	211
$Bunga_5$	0-7-8-0	32
	0-2-5-12-0	38
	0-13-4-3-0	38
	0-9-11-1-0	31
	0-10-6-0	30
	Fungsi tujuan $bunga_5$	169

Berdasarkan **Tabel 4.3** nilai fungsi tujuan setiap bunga diperoleh dari penjumlahan jarak tempuh rute masing-masing bunga. Sedangkan fungsi terbaik sementara diperoleh dari proses pengurutan fungsi tujuan dari yang terkecil hingga terbesar yang dimiliki dari setiap bunga, kemudian menjadikan fungsi tujuan dengan nilai terkecil sebagai solusi terbaik sementara. Dari hasil di atas diperoleh fungsi tujuan terbaik sementara pada $bunga_2$.

Langkah IV : Memperbarui Posisi Bunga dan Update Solusi Baru

Pada langkah ini dilakukan pembaruan bunga dengan penyerbukan. Sebelum memperbarui bunga bangkitan bilangan real ri pada interval (0,1) secara acak untuk setiap bunga. Jika $r_i < p_a$ maka bunga melakukan penyerbukan global jika $r_i \geq p_a$ maka bunga diperbarui dengan penyerbukan lokal.

Hasil perbandingan r dengan $p_a = 0.25$ disajikan dalam **Tabel 4.4.**

Tabel 4.4 Hasil penentuan penyerbukan

bunga	r_i	Penyerbukan
$bunga_1$	0.1	Global
$bunga_2$	0.63	Lokal
$bunga_3$	0.73	Lokal
$bunga_4$	0.22	Global
$bunga_5$	0.59	Lokal

Berdasarkan **Tabel 4.4** diperoleh $bunga_1$ dan $bunga_4$ mengalami penyerbukan global sedangkan $bunga_2, bunga_3$ dan $bunga_5$ mengalami penyerbukan lokal.

a) Penyerbukan Global

Proses memperbarui elemen $bunga_1$ dengan penyerbukan global ditunjukan pada langkah berikut :

1. Menghitung nilai s
   - Membangkitkan nilai u yang merupakan bilangan real secara acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1 sebanyak pelanggan. Misalkan diperoleh nilai u untuk $bunga_1$ sebagai berikut :
     0.538 1.834 -2.259 0.862 0.319 -1.308 -0.434 … 0.725
   - Membangkitkan nilai v yang merupakan bilangan real secara acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1 sebanyak pelanggan. Misalkan diperoleh nilai u untuk $bunga_1$ sebagai berikut :
     0.727 -0.303 0.294 -0.787 0.888 -1.147 -1.069 … 1.370
   - Menghitung nilai σ
     Nilai σ didalamnya terdapat fungsi gamma yang diselesaikan dengan Matlab.

$$\sigma^2 = \left(\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left(\frac{1+\lambda}{2}\right)} \cdot \frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{\frac{\lambda-1}{2}}} \right)^{1/\lambda}$$

$$\sigma^2 = \left(\frac{\Gamma(1+1.5)}{(1.5)\Gamma\left(\frac{1+1.5}{2}\right)} \cdot \frac{\sin\left(\frac{\pi(1.5)}{2}\right)}{2^{\frac{1.5-1}{2}}} \right)^{1/1.5}$$

$$\sigma^2 = 0.696579$$

- Menghitung nilai s setiap elemen bunga

$$s = \frac{ux\sigma}{|V|^{1/\lambda}}$$

$$s = \frac{0.538X0.696579}{|0.727|^{1/\lambda}}$$

$$s = 0.55509669$$

Cara serupa juga dilakukan pada elemen-elemen pada $bunga_1$ lainnya. Sehingga diperoleh nilai s dari $bunga_1$ sebagai berikut :

0.555 3.390 -4.265 0.844 0.288 -0.996 -0.346 … 0.491

2. Menghitung nilai (L)

$$L = \frac{\lambda\Gamma(\lambda)sin(\frac{\pi\lambda}{2})}{\pi} \frac{1}{|s|^{1+\lambda}}$$

$$L = \frac{1.5\Gamma(1.5)sin(\frac{\pi 1.5}{2})}{\pi} \frac{1}{|0.555|^{2.5}}$$

$$L = 1.304$$

Cara serupa juga dilakukan pada elemen-elemen pada $bunga_1$ lainnya. Sehingga diperoleh nilai L dari $bunga_1$ sebagai berikut :

1.304 0.014 -0.008 0.457 6.730 -0.302 -4.246 … 1.774

3. Menentukan posisi baru $bunga_1$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \alpha L(\lambda)(xbest - x_{i,j}^{t})$$

$$x_{1,1}^{2} = x_{1,1}^{1} + \alpha L(\lambda)(xbest - x_{1,1}^{1})$$

$$x_{1,1}^{2} = 0.2277 + 1.304(0.4357 - 0.2277)$$

$$x_{1,1}^{2} = 0.255$$

Cara serupa juga dilakukan pada elemen-elemen pada $bunga_1$ lainnya. Sehingga diperoleh posisi baru dari $bunga_1$ sebagai berikut :

0.255 0.923 -0.905 0.118 0.371 0.696 0.306 … 0.513

4. Evaluasi nilai fungsi tujuan hasil penyerbukan

Setelah mendapatkan posisi baru maka langkah selanjutnya yang dilakukan adalah menentukan rute bunga hasil penyerbukan sehingga diperoleh nilai fungsi tujuan berdasarkan total jarak tempuh. Hasil penentuan rute kendaraan pada $bunga_1$ disajikan dalam **Tabel 4.5.**

Tabel 4.5. Rute kendaraan Hasil Penyerbukan $bunga_1$

$Bunga_1$ baru	0-3-13-12-0	35
	0-1-6-9-0	38
	0-5-4-11-0	36
	0-10-2-8-0	43
	0-7-0	32
	Fungsi tujuan $bunga_1$ baru	184

Untuk hasil selengkapnya dari posisi baru bunga hasil penyerbukan global dapat dilihat pada **Lampiran 8.** Langkah selanjutnya yaitu update nilai fungsi tujuan.

5. Mengupdate nilai fungsi tujuan

Setelah menghitung nilai fungsi tujuan baru dari hasil proses penyerbukan langkah selanjutnya adalah melakukan proses update nilai fungsi tujuan dengan membandingkan hasil nilai fungsi tujuan $bunga_1$ lama dengan nilai fungsi tujuan $bunga_1$baru. Jika nilai fungsi tujuan baru lebih baik, maka nilai fungsi tujuan yang lama akan digantikan dengan nilai fungsi tujuan yang baru. Jika nilai fungsi tujuan

baru tidak lebih baik daripada fungsi tujuan lama, maka fungsi tujuan yang lama sebagai solusi. Proses update nilai $bunga_1$ disajikan dalam **Tabel 4.6.**

Tabel 4.6. Update nilai fungsi tujuan $bunga_1$

Bunga	Fungsi tujuan lama	Fungsi tujuan baru	Keterangan
$bunga_1$	175	184	Karena $f(x_1)$ lama < $f(x_1)$ baru, maka yang dipilih adalah $f(x_1)$ = 175

Proses memperbarui posisi dan update nilai fungsi tujuan yang telah ditunjukkan menggunakan $bunga_1$ juga diterapkan pada $bunga_4$ yang mengalami penyerbukan global. Posisi baru bunga hasil penyerbukan global disajikan dalam **Tabel 4.7** dan **Tabel 4.8.**

Tabel 4.7. Posisi baru bunga hasil penyerbukan global

Bunga	Pelanggan						
	1	2	3	4	5	…	13
$Bunga_1$	0.255	0.923	-0.905	0.118	0.371	…	0.513
$Bunga_4$	0.683	0.127	0.658	-152.9	0.945	…	0.675

Posisi bunga baru untuk setiap bunga yang mengalami penyerbukan global didapatkan dari proses penyerbukan global pada langkah pertama sampai langkah ketiga. Setelah mendapatkan posisi bunga terbaru, kemudian menentukan nilai fungsi tujuan dari posisi bunga baru yang didapatkan dari proses penyerbukan global pada langkah keempat. Langkah terakhir dalam proses ini yaitu membandingkan nilai fungsi tujuan lama dengan nilai fungsi tujuan yang baru. Hasil perbandingan disajikan pada **Tabel 4.8**.

Tabel 4.8. Update nilai fungsi tujuan bunga hasil penyerbukan global

Bunga	Fungsi tujuan lama	Fungsi tujuan baru	Keterangan
$bunga_1$	175	184	Karena $f(x_1)$ lama < $f(x_1)$ baru, maka yang dipilih adalah $f(x_1)$ = 175
$bunga_4$	211	178	Karena $f(x_4)$ lama > $f(x_4)$ baru, maka yang dipilih adalah $f(x_4)$ = 178

b) Penyebukan Lokal

Proses memperbarui elemen $bunga_2$ dengan penyerbukan lokal ditunjukan pada langkah berikut :

1. Menentukan 2 bunga secara acak untuk proses penyerbukan lokal.
   - Misalkan diperoleh $bunga_1$ dan $bunga_3$. Ambil masing-masing elemen dari $bunga_1$ dan $bunga\_3$.

     $bunga_1$ 0.228 0.923 0.905 0.111 0.595 0.711 … 0.489

     $bunga_3$ 0.311 0.185 0.439 0.409 0.603 0.117 … 0.679

2. Menentukan elemen $bunga_2$ baru
   - Membangkitkan secara acak ε yang merupakan bilangan real berdisstribusi uniform. Misalkan diperoleh nilai ε untuk $bunga_2$ sebagai berikut :

     0.117 0.536 0.164 0.940 0.193 0.766 0.061 … 0.624

   - Setelah membangkitkan ε, langkah selanjutnya adalah menentukan $bunga_2$ baru.

     $x_{i,j}^{t+1} = x_{i.j}^{t} + \varepsilon(x_j^t - x_k^t)$

$$x_{2,1}^2 = x_{2,1}^1 + \varepsilon(x_{1,1}^1 - x_{3,1}^1)$$

$$x_{2,1}^2 = 0.4357 + 0.117228525(0.2277 - 0.3111)$$

$$x_{2,1}^2 = 0.426$$

Cara serupa juga dilakukan pada elemen-elemen pada $bunga_2$ lainnya. Sehingga diperoleh posisi baru dari $bunga_2$ sebagai berikut :

0.426 0.826 1.056 -0.021 0.261 0.676 0.311 … 0.580

3. Evaluasi nilai fungsi tujuan hasil penyerbukan

Setelah mendapatkan posisi bunga baru maka langkah selanjutnya yang dilakukan adalah menentukan rute bunga hasil penyerbukan sehingga diperoleh nilai fungsi tujuan berdasarkan total jarak tempuh. Hasil penentuan rute kendaraan pada $bunga_1$ disajikan dalam **Tabel 4.9.**

Tabel 4.9. Rute kendaraan Hasil Penyerbukan $bunga_2$

$Bunga_2$ baru	0-7-12-13-0	33
	0-1-5-10-6-0	48
	0-4-2-0	38
	0-9-3-11-0	30
	0-8-0	20
	Fungsi tujuan $bunga_2$ baru	169

Untuk hasil selengkapnya dari posisi baru bunga hasil penyerbukan lokal dapat dilihat pada **Lampiran 8.** Langkah selanjutnya yaitu update nilai fungsi tujuan.

4. Mengupdate nilai fungsi tujuan

Setelah menghitung nilai fungsi tujuan baru dari hasil proses penyerbukan langkah selanjutnya adalah melakukan proses update nilai fungsi tujuan dengan

membandingkan hasil nilai fungsi tujuan $bunga_1$ lama dengan nilai fungsi tujuan $bunga_1$baru. Jika nilai fungsi tujuan baru lebih baik, maka nilai fungsi tujuan yang lama akan digantikan dengan nilai fungsi tujuan yang baru. Jika nilai fungsi tujuan baru tidak baik, maka fungsi tujuan yang lama sebagai solusi. Proses update nilai $bunga_1$ disajikan dalam **Tabel 4.10.**

Tabel 4.10. Update nilai fungsi tujuan $bunga_2$

Bunga	Fungsi tujuan lama	Fungsi tujuan baru	Keterangan
$bunga_2$	165	169	Karena $f(x_2)$ lama < $f(x_2)$ baru, maka yang dipilih adalah $f(x_2)$ = 169

Proses memperbarui posisi dan update nilai fungsi tujuan yang telah ditunjukkan menggunakan $bunga_2$ juga diterapkan pada $bunga_3$ dan $bunga_5$ yang mengalami penyerbukan lokal. Posisi baru bunga hasil penyerbukan global disajikan dalam **Tabel 4.11** dan **Tabel 4.12.**

Tabel 4.11. Posisi baru bunga hasil penyerbukan lokal

Bunga	Pelanggan						
	1	2	3	4	5	…	13
$Bunga_2$	0.426	0.826	1.056	-0.021	0.261	…	0.580
$Bunga_3$	0.367	0.857	1.024	0.101	0.192	…	0.580
$Bunga_5$	0.896	1.309	0.174	0.267	0.963	…	0.714

Posisi bunga baru untuk setiap bunga yang mengalami penyerbukan lokal didapatkan dari proses penyerbukan lokal pada langkah pertama sampai langkah kedua. Setelah mendapatkan posisi bunga terbaru, kemudian menentukan nilai fungsi tujuan dari posisi bunga baru yang didapatkan dari proses penyerbukan lokal pada langkah ketiga. Langkah terakhir dalam proses ini yaitu membandingkan nilai

fungsi tujuan lama dengan nilai fungsi tujuan yang baru. Hasil perbandingan disajikan pada **Tabel 4.12.**

Tabel 4.12. Update nilai fungsi tujuan bunga hasil penyerbukan lokal

Bunga	Fungsi tujuan lama	Fungsi tujuan baru	Keterangan
$bunga_2$	165	169	Karena $f(x_2)$ lama < $f(x_2)$ baru, maka yang dipilih adalah $f(x_2)$ = 169
$bunga_3$	174	190	Karena $f(x_3)$ lama < $f(x_3)$ baru, maka yang dipilih adalah $f(x_3)$ = 174
$bunga_5$	169	178	Karena $f(x_5)$ lama < $f(x_5)$ baru, maka yang dipilih adalah $f(x_5)$ = 169

Setelah setiap bunga mengalami penyerbukan dan update fungsi tujuan, maka didapat nilai fungsi tujuan terbaru dari $bunga_1$ sebesar 175, $bunga_2$ sebesar 165, $bunga_3$ sebesar 174, $bunga_4$ sebesar 178 dan $bunga_5$ sebesar 169.

Langkah V : Menentukan Solusi Terbaik

Tahapan ini dilakukan jika seluruh bunga telah melalui keseluruhan satu proses iterasi. Penentuan solusi terbaik ini menggunakan informasi dari fungsi tuujuan yang telah diupdate. Solusi terbaik ini ditemukan diseluruh bunga dengan nilai fungsi tujuan terkecil atau dapat disebut sebagai $xbest$ yang nantinya akan digunakan sebagai $xbest$ sementara untuk iterasi berikutnya. Proses ini dilakukan pada tiap iterasi sehingga akan diperoleh solusi terbaik setiap iterasinya (t iterasi) maka untuk keseluruhan akan didapat sejumlah t solusi terbaik.

Tabel 4.13. Rute kendaraan terbaik

Bunga	Rute yang terbentuk	Total Jarak
$Bunga_1$	0-2-13-12-0	29
	0-1-8-9-0	33
	0-4-6-11-0	41
	0-10-3-7-0	42
	0-5-0	30
	Fungsi tujuan $bunga_1$	175
$Bunga_2$	0-8-7-0	32
	0-13-4-0	37
	0-5-3-6-2-0	48
	0-1-10-9-0	32
	0-11-12-0	16
	Fungsi tujuan $bunga_2$	165
$Bunga_3$	0-5-2-8-0	41
	0-6-10-1-0	38
	0-7-4-0	48
	0-12-9-13-0	23
	0-3-11-0	24
	Fungsi tujuan $bunga_3$	174
$Bunga_4$	0-9-4-0	39
	0-7-1-13-0	46
	0-5-12-3-6-0	59

	0-11-2-10-0	30
	0-8-0	20
	Fungsi tujuan $bunga_4$	194
$Bunga_5$	0-7-8-0	32
	0-2-5-12-0	38
	0-13-4-3-0	38
	0-9-11-1-0	31
	0-10-6-0	30
	Fungsi tujuan $bunga_5$	169

Berdasarkan **Tabel 4.13** dapat disimpulkan bahwa solusi terbaik pada iterasi pertama adalah pada $bunga_2$ dengan nilai fungsi tujuan sebesar, $f(x_2) = 165$.

Langkah VI : Mengecek Maksimum Iterasi

Pada langkah ini, kriteria yang harus dipenuhi adalah pengulangan proses algoritma sebanyak maksimum iterasi (maxiterasi). Saat proses inisialisasi parameter diinputkan maxiterasi = 1, maka pengerjaan manual ini telah selesai karena mencapai maksimum iterasi sehingga solusi terbaik terbentuk dari $bunga_2$ dengan nilai fungsi tujuan (jarak tempuh) sebesar 165 satuan jarak. Rute terbaik sebagai berikut :

$rute\ 1 \leftarrow 0 - 8 - 7 - 0$

$rute\ 2 \leftarrow 0 - 13 - 4 - 0$

$rute\ 3 \leftarrow 0 - 5 - 3 - 6 - 2 - 0$

$rute\ 4 \leftarrow 0 - 1 - 10 - 9 - 0$

$rute\ 1 \leftarrow 0 - 11 - 12 - 0$

4.4 Program

Berdasarkan prosedur yang telah dijelaskan pada metode penelitian, agar mempermudah penyelesaian *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) menggunakan *Flower Pollination Algorithm* menggunakan Bahasa pemrograman java dengan software Netbeans IDE 8.2 dan Matlab R2017A. *Source code* program dapat dilihat pada **Lampiran 9.** Pemilihan penggunaan bahasa pemrograman java didasarkan pada kemudahan struktur penulisan program dan pemahaman penerapan *Flower Pollination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

4.5 Implementasi Program pada Contoh Kasus *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

4.5.1 Implementasi pada Data dengan 13 pelanggan

Solusi terbaik dari hasil yang diperoleh pada data 13 pelanggan dengan menggunakan nilai $\alpha = 0.01$ **(Civicioglu dan Besdok, 2013),** $\lambda = 1.5$ **(Kaveh dan Bakhshpoori, 2011),** banyak pelanggan adalah 13, banyak kendaraan adalah 5, kapasitas maksimum kendaraan sebesar 180, presentase jumlah barang yang diangkut kendaraan dari depot sebesar 70% dari total kapasitas kendaraan (penentuan presentase jumlah barang yang diangkut dari depot, berdasarkan hasil *running* program yang cenderung lebih baik dengan data kecil pada **Lampiran 14)** dan banyak bunga, maksimum iterasi, serta nilai *switch probability* dijalankan bervariasi. Hasil selengkapnya disajikan pada **Tabel 4.14**.

Tabel 4.14. Hasil *Running* Program Pada Data dengan 13 Pelanggan

Jumlah bunga	Maksimum Iterasi	*Switch Probability*		
		0.25	0.5	0.8
10	10	124	121	120
	100	113	110	104
	1000	103	100	99
50	`10	113	112	112
	100	105	104	104
	1000	102	102	102
100	10	115	115	107
	100	107	107	101
	1000	99	99	99

Berdasarkan **Tabel 4.14** diperoleh solusi terbaik dengan jarak total sebesar 99 satuan jarak dengan jumlah bunga 10 maksimum iterasi 1000 *switch probability* 0.8, bunga 100 maksimum iterasi 1000 *switch probability* 0.25, bunga 100 maksimum iterasi 1000 *switch probability* 0.5 dan bunga 100 maksimum iterasi 1000 *switch probability* 0.8. Berdasarkan pola hasil *running* dapat disimpulkan bahwa semakin banyak jumlah bunga dan jumlah iterasi, maka hasil yang diperoleh cenderung lebih baik dengan total jarak tempuh yang kecil. Sedangkan untuk nilai *switch probability*, semakin besar *switch probability* maka solusi yang dihasilkan semakin baik dengan total jarak tempuh yang kecil. Rute yang terbentuk untuk menyelesaikan contoh kasus pada data dengan 13 pelanggan berdasarkan hasil terbaik ini dapat dilihat pada **Lampiran 10.**

4.5.2 Implementasi pada Data dengan 22 pelanggan

Solusi terbaik dari hasil yang diperoleh pada data 22 pelanggan dengan menggunakan nilai α = 0.01 **(Civicioglu dan Besdok, 2013),** λ = 1.5 **(Kaveh dan Bakhshpoori, 2011),** banyak pelanggan adalah 22, banyak kendaraan adalah 5, kapasitas maksimum kendaraan sebesar 10500, presentase jumlah barang yang diangkut kendaraan dari depot sebesar 70% dari total kapasitas kendaraan (penentuan presentase jumlah barang yang diangkut dari depot, berdasarkan hasil *running* program yang cenderung lebih baik dengan data kecil pada **Lampiran 15)** dan banyak bunga, maksimum iterasi, serta nilai *switch probability* dijalankan bervariasi. Hasil selengkapnya disajikan pada **Tabel 4.15**.

Tabel 4.15. Hasil *Running* Program Pada Data dengan 22 Pelanggan

Jumlah bunga	Maksimum Iterasi	*Switch Probability*		
		0.25	0.5	0.8
10	10	159	144	152
	100	156	155	152
	1000	140	140	130
50	`10	164	166	164
	100	154	154	154
	1000	132	132	126
100	10	163	157	156
	100	149	144	144
	1000	131	128	124

Berdasarkan **Tabel 4.15** diperoleh solusi terbaik dengan jarak total sebesar 124 satuan jarak dengan jumlah bunga 100, maksimum iterasi 1000 dan *switch probability* 0.8. Berdasarkan pola hasil *running* dapat disimpulkan bahwa semakin banyak jumlah bunga dan jumlah iterasi, maka hasil yang diperoleh cenderung lebih

baik dengan total jarak tempuh yang kecil. Sedangkan untuk nilai *switch probability*, semakin besar *switch probability* maka solusi yang dihasilkan juga cenderung semakin baik dengan total jarak tempuh yang kecil. Rute yang terbentuk untuk menyelesaikan contoh kasus pada data dengan 22 pelanggan berdasarkan hasil terbaik ini dapat dilihat pada **Lampiran 11.**

4.5.3 Implementasi pada Data dengan 100 pelanggan

Solusi terbaik dari hasil yang diperoleh pada data 100 pelanggan dengan menggunakan nilai $\alpha = 0.01$ **(Civicioglu dan Besdok, 2013),** $\lambda = 1.5$ **(Kaveh dan Bakhshpoori, 2011),** banyak pelanggan adalah 100, banyak kendaraan adalah 5, kapasitas maksimum kendaraan sebesar 700, presentase jumlah barang yang diangkut kendaraan dari depot sebesar 70% dari total kapasitas kendaraan (penentuan presentase jumlah barang yang diangkut dari depot, berdasarkan hasil *running* program yang cenderung lebih baik dengan data kecil pada **Lampiran 16)** dan banyak bunga, maksimum iterasi, serta nilai *switch probability* dijalankan bervariasi. Hasil selengkapnya disajikan pada **Tabel 4.16**.

Tabel 4.16. Hasil *Running* Program dengan 100 Pelanggan

Jumlah bunga	Maksimum Iterasi	*Switch Probability*		
		0.25	0.5	0.8
10	10	3066	3179	3111
	100	2840	2809	2803
	1000	2641	2630	2531
50	`10	3057	3040	2975
	100	2723	2622	2622
	1000	2605	2574	2545
100	10	2950	2803	2922
	100	2681	2681	2591
	1000	2483	2466	2432

Berdasarkan **Tabel 4.16** diperoleh solusi terbaik dengan jarak total sebesar 2432 satuan jarak dengan jumlah bunga 100, maksimum iterasi 1000 dan *switch probability* 0.8. Berdasarkan pola hasil *running* dapat disimpulkan bahwa semakin banyak jumlah bunga dan jumlah iterasi, maka hasil yang diperoleh cenderung lebih baik dengan total jarak tempuh yang kecil. Sedangkan untuk nilai *switch probability*, semakin besar *switch probability* maka solusi yang dihasilkan juga cenderung semakin baik dengan total jarak tempuh yang kecil. Rute yang terbentuk untuk menyelesaikan contoh kasus pada data dengan 100 pelanggan berdasarkan hasil terbaik ini dapat dilihat pada **Lampiran 12.**

Dari ketiga contoh kasus diperoleh solusi terbaik data 13 pelanggan dengan nilai fungsi tujuan sebesar 99 satuan jarak, data 22 pelanggan sebesar 124 satuan jarak dan data 100 pelanggan sebesar 2432 satuan jarak. Berdasarkan pola hasil *running* dari ketiga contoh kasus dapat disimpulkan bahwa semakin banyak jumlah

bunga, jumlah iterasi, jumlah presentase barang yang diangkut dari depot dan nilai *switch probability*, maka hasil yang diperoleh cenderung lebih baik dengan total jarak tempuh yang kecil. Kesimpulan ini diperkuat dengan *running* program data besar dengan banyak bunga = 120, maximum iterasi = 1500, *switch probability* = 0.8, nilai α = 0.01 **(Civicioglu dan Besdok, 2013),** λ = 1.5 **(Kaveh dan Bakhshpoori, 2011),** banyak pelanggan adalah 100, banyak kendaraan adalah 5, kapasitas maksimum kendaraan sebesar 700, dan presentase jumlah barang yang diangkut kendaraan dari depot sebesar 70% dari total kapasitas kendaraan didapatkan hasil terbaik sebesar 2274 satuan jarak.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pembahasan pada BAB IV, diperoleh kesimpulan sebagai berikut :

1. *Flower Pollination Algorithm* (FPA) dapat digunakan untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD). Proses yang dilakukan untuk menerapkan *Flower Pollination Algorithm* untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* antara lain : input data dan inisialisasi parameter, membangkitkan elemen bunga awal, menghitung nilai fungsi tujuan dari setiap bunga, menentukan $xbest$ sementara, membandingkan nilai setiap bunuga dengan nilai *switch probability*, memperbarui bunga dengan penyerbukan global dan local, mengupdate bunga tiap penyerbukan, menentukan bunga terbaik untuk setiap iterasi, mengulangi proses membandingkan nilai fungsi tujuan sampai menentukan bunga terbaik untuk setiap iterasi hingga iterasi maksimum terpenuhi.
2. Program untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* dengan menggunakan *Flower Pollination Algorithm* dibuat dengan Bahasa pemrograman *Java* dengan *software* NetBeans 8.2. Program ditulis secara urut sesuai langkah dalam *Flower Pollination Algorithm*, dimulai dengan input data dan inisialisasi parameter, membangkitkan elemen awal bunga, mengurutkan elemen bunga, menghitung fungsi tujuan, menentukan $xbest$ sementara, membandingan nilai r_i setiap bunga dengan *switch probability*, melakukan proses penyerbukan global atau local, menghitung fungsi tujuan hasil penyerbukan, mengupdate bunga setiap penyerbukan, lalu menetapkan bunga terbaik sampai maksimum iterasi terpenuhi.

3. Implementasi program untuk contoh kasus VRPSPD menggunakan data 13 pelanggan diperoleh solusi terbaik dengan total jarak tempuh yaitu 99, dengan data 22 pelanggan diperoleh solusi terbaik dengan total jarak tempuh 124 dan dengan data 100 pelanggan diperoleh solusi terbaik dengan jarak tempuh 2432. Pola hasil *running* program menunjukan bahwa semakin banyak jumlah bunga dan jumlah iterasi maka hasil yang diperoleh cenderung lebih baik. Serta semakin besar nilai *switch probability* diperoleh hasil yang baik yaitu dengan total jarak tempuh yang lebih kecil.

5.2 Saran

Untuk penelitian selanjutnya, *Flower Pollination Algorithm* dapat dilakukan pengembangan algoritma seperti *Multi Objective Flower Pollination Algorithm* maupun *hybrid* dengan algorithma lainnya seperti *Cuckoo Search Algorithm, Ant Colony, Bat Algorithm* atau algoritma lainnya yang memungkinkan mendapatkan hasil yang lebih baik.

DAFTAR PUSTAKA

Chartrand, G. dan Oellermann, O. R., 1993, *Applied and Algorithm GraphTheory*, McGraw-Hill, NewYork.

Catay, B., 2010, A New Saving-based Ant Algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, *Expert System with Application*, 37: 6809-6817.

Civicioglu, P. dan Besdok, E.,2013., Comparative Analysis of the Cuckoo Search Algorithm, In: Yang XS.(eds) Cuckoo Search and Firefly Algorithm. *Studies in Computational Intelligence*, Vol. 516,85-113.

Cordeu, M., Hertz,G. dan Sormany., 2002, *A Guide for Vehicle Routing Problem, Journal of the Operational Research Society* 55: 542-546.

Dethloff, J., 2001, Vehicle Routing Problem and Reserve Logistics : The Vehicle Routing Problem with Simultaneous Delivery and Pickup, *Operational Research Spektrum,* 23: 79-96.

Diethelm, K., 2004, *The Analysis of Fractional Differential Equation,* Springer, New York.

Gajpal, Y., dan Abad, P., 2009,*An Ant Colony System*(ACS) *for Vehicle Routing Problem with Simultaneous Delivery and Pickup, Computers and Operations Research*, Elsevier, 33: 3215-3223.

Kaveh, A. dan Bakhshpoori, T., 2011., Optimum design of Steel Frames Using Cuckoo Search Algorithm with Levi Flights, *The Structural Design of Tall and Special Buildings*, 22 (13), 1023-1036.

Lanczos, C. 1964, *A Precision Approximation of The Gamma Function, SIAM J. Numer. Anal I,* 86-96.

Mantegna, R.N. 1994. *Fast, Accurate Algorithm for Numerical Simulation of Levy Stable Stochastic Processes Phys. Rev E 49,* 4677-4683.

Meyers, F.E, dan Stewart, J.R. 2001. *Motion and time study for lean manufacturing*, edisi ke-3, New Jersey : Prentice Hall.

Min, H., 1989, The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points, *Transportation Research A,* **23**(5), 377-386.

Nagy, G., dan Salhi, S., 2005, Heuristic Algorithm for Single and Multiple Depot Vehicle Routing Problem with Pickup and Deliveries, *European Journal of Operational Research,* 162: 126-141.

Obitko, M., 1998, *Genetic Algorithm,* Czech Technical University.

Prins, C., 2009, A GRASP x Evolutionary Local Search Hybrid for the Vehicle Routing Problem, *Bio-Inspired Algorithm for the Vehicle Routing Problem*,Springer,161 : 35-53.

Serdar, A. Tasan, dan Gen, Mitsuo, 2012, A Genetic Algorithm Based Approach to Vehicle Routing Problem with Simultaneous Pickup and Deliveries, *Computers and Industrial Engineering*, Elsevier, 62 : 755-761.

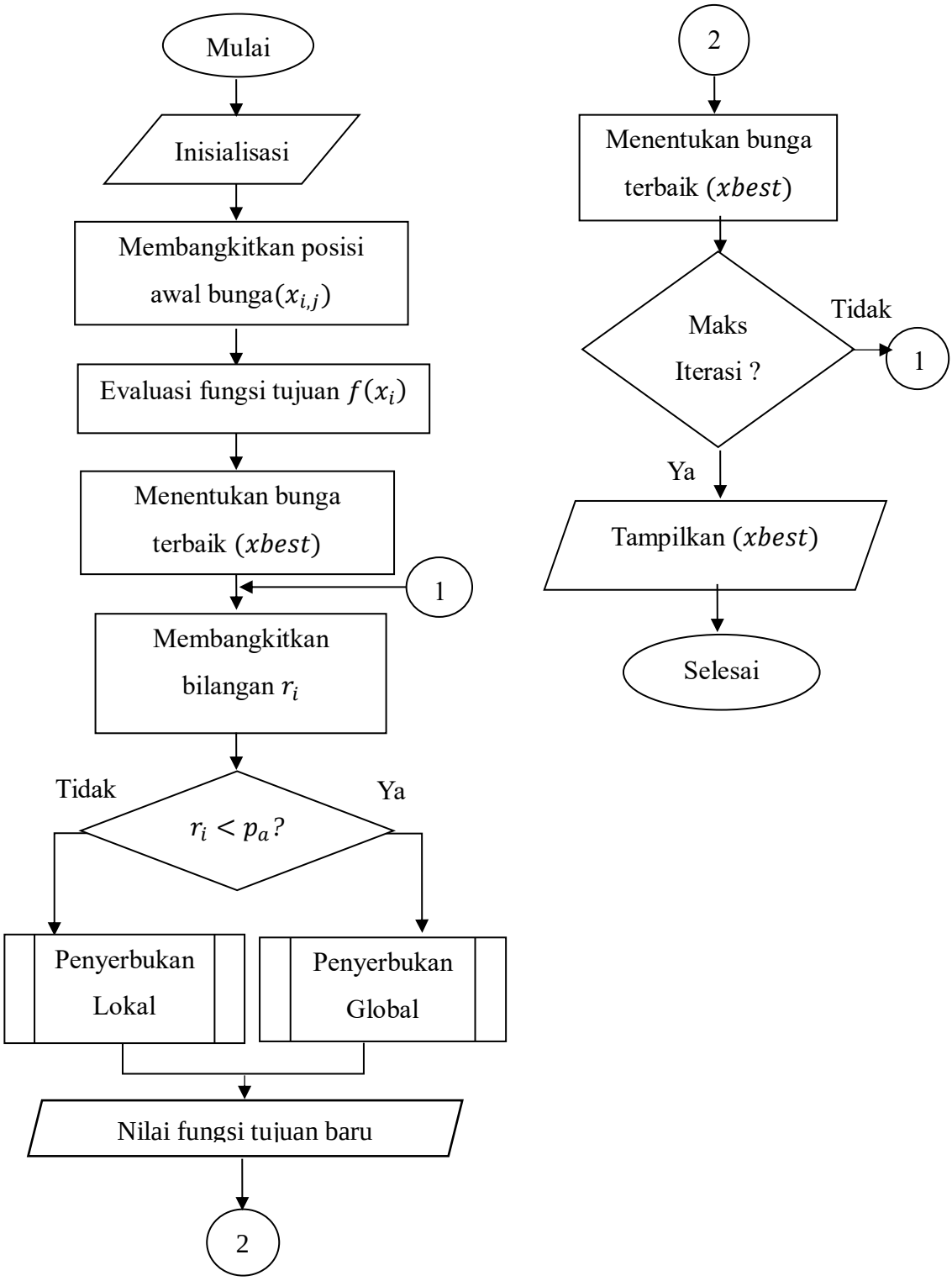
Solomon, M. dan Desrosiers, J, 1998, Time Windows Constrained Routing and Scheduling Problem, *Operation Research Society*, 22 : 1-13.

Toth P dan Vigo D, 2002, *The Vehicle Routing Problem, society for industrial and Applied Matematics*, Philadelphia, USA.

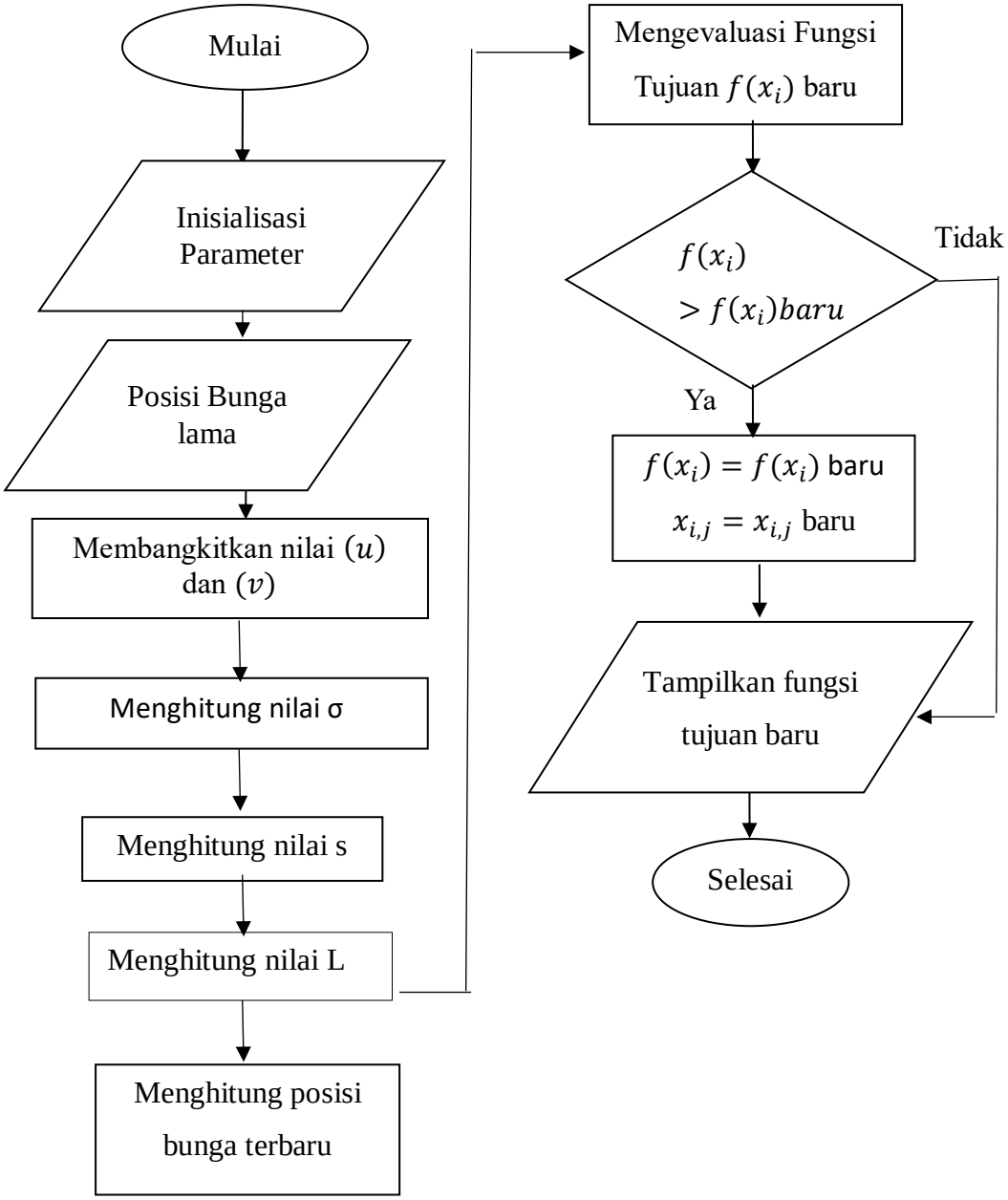
Vasant, P., Weber, G-W., Dieu V.N., 2016., Handbook of Research on Modern Optimization Algorithms and Applications in engineering and economics, IGI Global, USA.

Yang, X.-S, 2012, *Nature-Inspired Optimization Algorithms Chapter 11 Flower Pollination Algorithms,* Elsevier, 155-173.

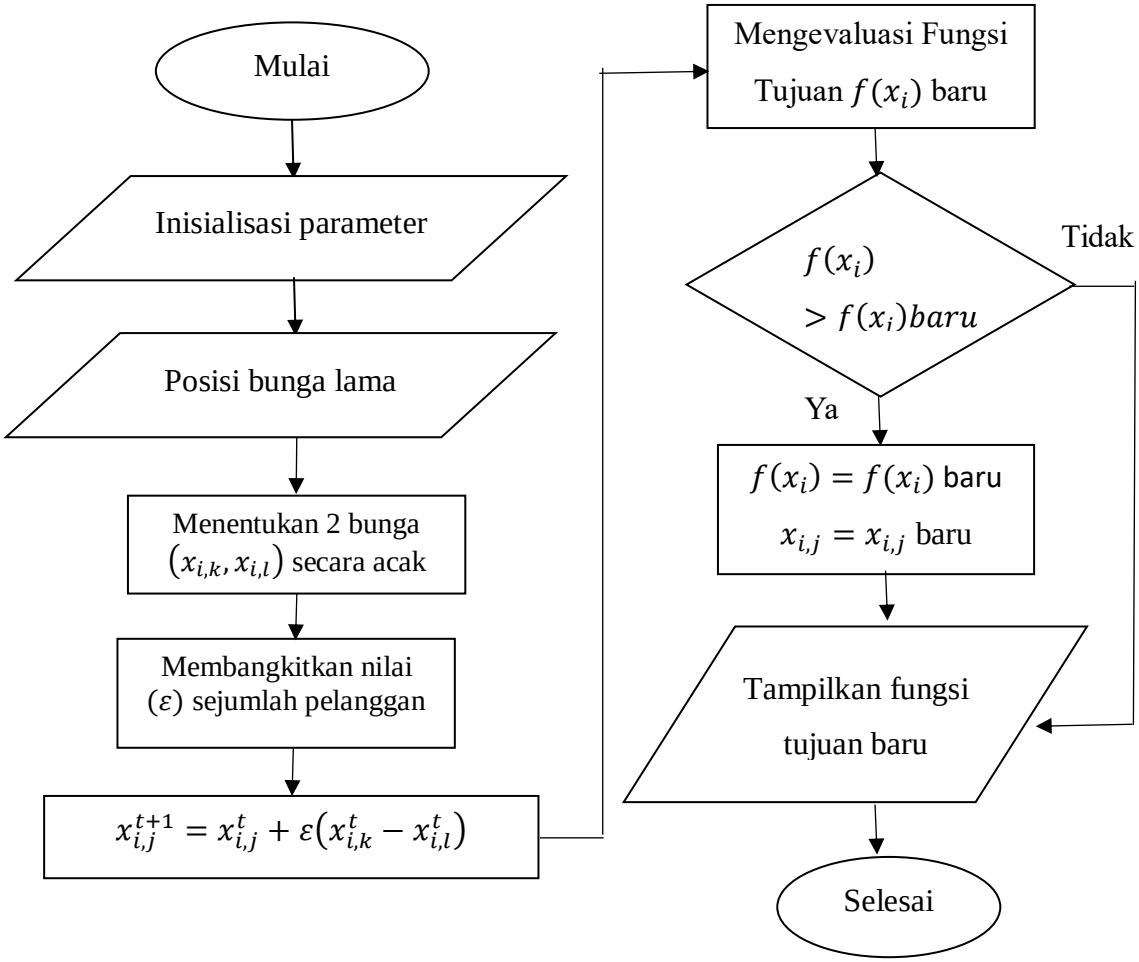
Lampiran 1 : Flowchart Penerapan *Flower Pollination Algorithm* (FPA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)



Lampiran 2 : Flowchart Penyerbukan Global



Lampiran 3 : Flowchart Penyerbukan Lokal



Lampiran 4 : Data *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan 13 Pelanggan

Kendaraan : 5
Pelanggan : 13
Kapasitas max : 180

Data jarak antar pelanggan

$C_{i,j}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	0	6	8	12	17	15	15	16	10	10	10	5	6	2
1	6	0	4	12	17	16	17	20	14	14	12	8	11	8
2	8	4	0	8	13	12	14	18	14	12	9	6	11	10
3	12	12	8	0	6	4	7	12	10	8	4	7	10	12
4	17	17	13	6	0	4	9	15	15	12	9	12	15	18
5	15	16	12	4	4	0	5	11	11	8	6	10	12	15
6	15	17	14	7	9	5	0	6	8	5	5	10	10	15
7	16	20	18	12	15	11	6	0	6	6	9	12	10	15
8	10	14	14	10	15	11	8	6	0	3	6	7	4	9
9	10	14	12	8	12	8	5	6	3	0	4	7	5	10
10	10	12	9	4	9	6	5	9	6	4	0	5	6	10
11	5	8	6	7	12	10	10	12	7	7	5	0	5	6
12	6	11	11	10	15	12	10	10	4	5	6	5	0	5
13	2	8	10	12	18	15	15	15	9	10	10	6	5	0

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**Total**
D	0	25	30	20	45	25	15	35	35	30	25	20	25	25	355
P	0	35	35	20	35	20	20	25	40	15	20	20	35	40	360

Keterangan :

i = j = depot dan pelanggam
D = *Delivery*
P = *Pickup*

Lampiran 5 : Data *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan 22 Pelanggan

Vehicle : 5 ; Customer : 22 ; Kapasitas Max :10500 ;

Data jarak antar pelanggan

$C_{i,j}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
0	0	7	14	23	3	24	17	3	13	10	4	13	1	7	9	16	10	6	9	9	5	13	15
1	7	0	7	20	9	15	6	5	10	6	12	6	6	2	2	5	5	2	3	3	4	16	14
2	14	7	0	9	11	10	9	14	3	4	17	14	10	9	7	9	5	6	4	12	6	6	4
3	23	20	9	0	13	6	12	19	9	12	25	19	18	16	14	14	12	14	12	17	15	11	7
4	3	9	11	13	0	20	15	6	8	7	6	12	4	8	10	15	6	8	9	10	6	7	10
5	24	15	10	6	20	0	8	18	12	12	22	15	18	15	12	10	13	13	12	15	15	15	12
6	17	6	9	12	15	8	0	12	11	10	17	7	12	9	5	1	10	8	7	7	10	15	13
7	3	5	14	19	6	18	12	0	10	8	5	6	2	3	6	10	6	4	6	5	3	12	14
8	13	10	3	9	8	12	11	10	0	2	12	12	9	9	8	11	3	6	5	9	6	3	3
9	10	6	4	12	7	12	10	8	2	0	11	10	6	7	6	10	1	4	3	8	4	4	5
10	4	12	17	25	6	22	17	5	12	11	0	8	4	8	11	15	9	9	10	9	7	13	16
11	13	6	14	19	12	15	7	6	12	10	8	0	8	4	4	6	9	5	6	2	6	15	15
12	1	6	10	18	4	18	12	2	9	6	4	8	0	4	7	12	5	5	6	6	3	10	12
13	7	2	9	16	8	15	9	3	9	7	8	4	4	0	3	8	5	2	4	2	3	11	12
14	9	2	7	14	10	12	5	6	8	6	11	4	7	3	0	5	5	3	3	3	5	11	11
15	16	5	9	14	15	10	1	10	11	10	15	6	12	8	5	0	10	7	7	6	9	15	13

Lampiran 5-2

16	10	5	5	12	6	13	10	6	3	1	9	9	5	5	5	10	0	3	2	6	3	6	7
17	6	2	6	14	8	13	8	4	6	4	9	5	5	2	3	7	3	0	2	3	2	9	10
18	9	3	4	12	9	12	7	6	5	3	10	6	6	4	3	7	2	2	0	5	3	8	8
19	9	3	12	17	10	15	7	5	9	8	9	2	6	2	3	6	6	3	5	0	5	13	13
20	5	4	6	15	6	15	10	3	6	4	7	6	3	3	5	9	3	2	3	5	0	8	10
21	13	16	6	11	7	15	15	12	3	4	13	15	10	11	11	15	6	9	8	13	8	0	4
22	15	14	4	7	10	12	13	14	3	5	16	15	12	12	11	13	7	10	8	13	10	4	0

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
D	0	2700	1500	800	1000	900	1600	800	1250	900	750	950	600	850	450	500	400	600	800	650	550	900	850
P	0	2500	1600	900	1100	850	1500	700	1000	800	800	750	700	900	400	400	350	500	900	850	450	1150	800

Keterangan :

i = j = indeks pelanggan
D = *Delivery*
P = *Pickup*

Lampiran 6 : Data *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan 100 Pelanggan

Vehicle : 4
Customer : 100
Kapasitas Max: 700

Data Jarak Antar Pelanggan

i	x	y	D	P
0	40	50	0	0
1	52	75	5	15
2	45	70	15	25
3	62	69	5	15
4	60	66	5	15
5	42	65	5	15
6	16	42	10	20
7	58	70	10	20
8	34	60	10	20
9	28	70	5	15
10	35	66	5	15
11	35	69	5	15
12	25	85	10	20
13	22	75	15	25
14	22	85	5	15
15	20	80	20	30
16	20	85	20	30
17	18	75	10	20
18	15	75	10	20
19	15	80	5	15
20	30	50	5	15
21	30	56	10	20
22	28	52	10	20
23	14	66	5	15
24	25	50	5	15
25	22	66	20	30
26	8	62	5	15

i	x	y	D	P
27	23	52	5	15
28	4	55	10	20
29	20	50	5	15
30	20	55	5	15
31	10	55	10	20
32	10	40	15	15
33	8	40	20	30
34	8	45	10	20
35	5	35	5	15
36	5	45	5	15
37	2	40	10	20
38	0	40	15	25
39	0	45	10	20
40	36	18	5	15
41	35	32	5	15
42	33	32	10	20
43	33	35	5	15
44	32	20	5	15
45	30	30	5	15
46	34	25	15	25
47	30	35	5	15
48	36	40	5	15
49	48	20	5	15
50	26	32	5	15
51	25	30	5	15
52	25	35	5	15
53	44	5	10	20

i	x	y	D	P
54	42	10	20	30
55	42	15	5	15
56	40	5	15	25
57	38	15	20	30
58	38	5	15	25
59	38	10	5	15
60	35	5	10	20
61	50	30	5	15
62	50	35	10	20
63	50	40	25	35
64	48	30	5	15
65	44	25	5	15
66	47	35	5	15
67	47	40	5	15
68	42	30	5	15
69	45	35	5	15
70	95	30	15	25
71	95	35	10	20
72	53	30	5	15
73	92	30	5	15
74	53	35	25	35
75	45	65	10	20
76	90	35	5	15
77	72	45	5	5
78	78	40	10	5
79	87	30	5	3
80	85	25	5	4

81	85	35	15	7
82	75	55	10	5
83	72	55	5	3
84	70	58	10	5
85	86	46	15	8
86	66	55	5	4
87	64	56	10	5
88	65	60	15	7
89	56	64	5	4
90	60	55	5	4
91	60	60	5	3
92	67	85	10	5
93	42	58	20	10
94	65	82	5	2
95	62	80	15	10
96	62	40	5	5
97	60	85	15	7
99	58	75	10	5
100	55	80	5	5

Keterangan :

i = indeks pelanggan
x = koordinat x
y = koordinat y
D = *Demand*
P = *Pickup*

Lampiran 7 : Posisi Awal Bunga

Bunga	Bilangan Real Acak												
	1	2	3	4	5	6	7	8	9	10	11	12	13
$bunga_1$	0.228	0.923	0.905	0.111	0.595	0.711	0.2967	0.508	0.801	0.730	0.237	0.547	0.489
$bunga_2$	0,436	0,430	0,980	0,258	0,262	0,222	0,319	0,085	0,029	0,489	0,459	0,521	0,624
$bunga_3$	0,311	0,185	0,439	0,409	0,603	0,117	0,424	0,262	0,929	0,578	0,963	0,232	0,679
$bunga_4$	0.815	0.127	0.632	0.278	0.957	0.158	0.957	0.800	0.422	0.792	0.656	0.849	0.679
$bunga_5$	0.906	0.913	0.097	0.547	0.965	0.971	0.485	0.142	0.916	0.959	0.036	0.934	0.758

Lampiran 8 : Posisi Baru Bunga Hasil Penyerbukan Global dan Lokal

Posisi Baru Bunga Hasil Penyerbukan Global

Bunga	Bilangan Real Acak												
	1	2	3	4	5	6	7	8	9	10	11	12	13
$bunga_1$	0.255	0.923	0.905	0.118	0.371	0.696	0.306	0.304	0.791	0.729	0.238	0.547	0.513
	3	13	12	1	6	9	5	4	11	10	2	8	7
$bunga_4$	0.683	0.127	0.657	-152,9	0.945	0.182	0.799	-0.057	0.420	0.783	-30,22	0.730	0.675
	9	4	7	1	13	5	12	3	6	11	2	10	8

Posisi Baru Bunga Hasil Penyerbukan Lokal

Bunga	Bilangan Real Acak												
	1	2	3	4	5	6	7	8	9	10	11	12	13
$bunga_2$	0.426	0.826	1.056	-0.021	0.261	0.676	0.311	0.151	-0.007	0.596	0.048	0.702	0.580
	7	12	13	1	5	10	6	4	2	9	3	11	8
$bunga_3$	0.367	0.857	1.024	0.101	0.192	0.646	0.278	0.008	0.138	0.445	0.222	0.348	0.580
	8	12	13	2	4	11	6	1	3	9	5	7	10
$bunga_5$	0.896	1.310	0.174	0.268	0.963	1.425	0.478	0.207	0.879	1.067	-0.375	1.146	0.714
	8	12	2	4	9	13	5	3	7	10	1	11	6

Lampiran 9 : *Source Code*

```
import java.io.*;
import java.util.*;
import java.util.logging.*;
import java.util.logging.Logger;
import java.util.Random;
import matlabcontrol.MatlabConnectionException;
import matlabcontrol.MatlabInvocationException;
import matlabcontrol.MatlabProxy;
import matlabcontrol.MatlabProxyFactory;
import matlabcontrol.MatlabProxyFactoryOptions;
import sun.security.tools.keytool.Main;
public class SKRIPSI {
    int x=1002,y=500,z=500;
    Scanner input = new Scanner (System.in);
    static int random, pilihan, pelanggan, i, j, k, bunga, maxiterasi, iterasi=1, stop, kendaraan, urut,
    indeks, cb;
    static double rad, sigma, ab, ba, kapasitas, alpha, sp, lamda, temp, maxkapasitas, presentase,
    jumawal, jrk, ttl, bil1, bil2, tambahan, gbest, totalpermintaan, totalpengambilan;
    double [ ][ ]jarak=new double[x][y], permintaan=new double[x][y], pengambilan=new double
    [x][y], jumper=new double[x][y], bsi=new double[x][y], total=new double[x][y];
    double[ ]prob=new double[x],koordinatx=new double[x], koordiinaty=new double[x];
    int[ ][ ][ ]per=new int[1002][500][100], ururtrandom=new int[1002][500][102], duplikat=new
    double[1002][500][100];
    double[ ][ ][ ]xsol=new double[1002][102][102];
    int[ ]bungalokal=new int[x];

    //input jenis data
      void Pilihdata(){
        int check;
        System.out.println("----------------------------------------------------------------------------------
    ----------------------------------------------------------------------");
        System.out.println("|                    Penerapan Flower Pollination Algorithm untuk
    Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and Delivery          |");
        System.out.println("|                                                                    Moh.Rizki Kurniawan
    |");
        System.out.println("|                                                                     Universitas Airlangga
    |");
        System.out.println("----------------------------------------------------------------------------------
    ----------------------------------------------------------------------");
        System.out.println("");
        System.out.println("\tPilihan data : ");
        System.out.println("\t1. Data kecil");
        System.out.println("\t2. Data sedang");
        System.out.println("\t3. Data besar");
        System.out.println("");
        do{
          System.out.print("\tPilihan data :\t");
          if(input.hasNextInt()){
            pilihan=input.nextInt();
            if(pilihan==1||pilihan==2||pilihan==3)
              check=1;
            else
              check=0;
```

```
            }
            else
            {String bad=input.next();check=0;}
            if(check==0)
               System.out.println("\tInput salah,mohon ulangi");
         } while(check==0);
      }

   //memanggil data dari notepad
      void Inisialisasidata (int pilihan){
         Scanner datajarak = new Scanner(System.in);
         Scanner datakoordinatx = new Scanner(System.in);
         Scanner datakoordinaty = new Scanner(System.in);
         Scanner datapermintaan = new Scanner (System.in);
         Scanner datapengambilan = new Scanner (System.in);
         switch(pilihan){
         case 1 :
         pelanggan=13;
         try{
         datajarak = new Scanner (new
         File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\J1.txt"));
         datapermintaan = new Scanner (new
         File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\D1.txt"));
         datapengambilan = new Scanner (new
         File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\P1.txt"));
         i=0;j=0;
         while(datajarak.hasNext()){
         String a=datajarak.next();
         double b=Double.parseDouble(a.trim());
          jarak[i][j++]=b;
          if(j==pelanggan+1)
          {i++;j=0;}
                }
          }
          catch(FileNotFoundException e)
          {System.out.println("\tFile tidak ditemukan");}
           break;
           case 2 :
           pelanggan=22;
           try{
           datajarak = new Scanner (new
           File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\J2.txt"));
           datapermintaan = new Scanner (new
           File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\D2.txt"));
           datapengambilan = new Scanner (new
           File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\P2.txt"));
           i=0;j=0;
           while(datajarak.hasNext()){
           String a=datajarak.next();
           double b=Double.parseDouble(a.trim());
            jarak[i][j++]=b;
            if(j==pelanggan+1)
            {i++;j=0;}
                }
             }
```

```
            catch(FileNotFoundException e)
            {System.out.println("\tFile tidak ditemukan");}
            break;
            case 3 :
            pelanggan=100;
            try{
            datakoordinatx = new Scanner (new
            File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\X3.txt"));
            datakoordinaty = new Scanner (new
            File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\X3.txt"));
            datapermintaan = new Scanner (new
            File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\D3.txt"));
            datapengambilan = new Scanner (new
            File("C:\\Users\\ACER\\Desktop\\rizki\\skripsi\\java\\P3.txt"));
            //(x,y)
             i=0;
             while(datakoordinatx.hasNext()){
             String a=datakoordinatx.next();
             double b=Double.parseDouble(a.trim());
              koordinatx[i++]=b;
              }
              i=0;
              while(datakoordinaty.hasNext()){
              String a=datakoordinaty.next();
              double b=Double.parseDouble(a.trim());
              koordinaty[i++]=b;
              }
              for(i=0;i<=pelanggan;i++){
                 for(j=0;j<=pelanggan;j++){
                    jarak[i][j]=Math.sqrt(Math.pow((koordinatx[i]-
                    koordinatx[j]),2)+Math.pow((koordinaty[i]-koordinaty[j]),2));
                     }
                 }
              }
              catch(FileNotFoundException e)
              {System.out.println("\tFile tidak ditemukan");}
              break;
          }
//menampilkan data di notepad

          //datajarak
        System.out.println("");
        System.out.println("\t------------------------------");
        System.out.println("\t          Data jarak          ");
        System.out.println("\t------------------------------");
        System.out.println("");
        for(i=0;i<=pelanggan;i++){
           for(j=0;j<=pelanggan;j++)
              System.out.print("\t"+jarak[i][j]);
           System.out.println();
        }
           //datapermintaan
        i=0;j=0;
        while(datapermintaan.hasNext()){
```

```
            String a=datapermintaan.next();
            double b=Double.parseDouble(a.trim());
            permintaan[i][j++]=b;

        }

        System.out.println("");
        System.out.println("\t-------------------------------------");
        System.out.println("\t          Data permintaan          ");
        System.out.println("\t-------------------------------------");
        System.out.println("");
        totalpermintaan=0;
        for(i=0;i<1;i++){
            for(j=0;j<=pelanggan;j++)
                System.out.print("\t"+permintaan[i][j]);
            System.out.println();
        }
        for(i=0;i<1;i++){
            for(j=0;j<=pelanggan;j++){
                totalpermintaan=totalpermintaan+permintaan[i][j];
            }
        }
        System.out.println("");
        System.out.println("\ttotal permintaan :\t"+totalpermintaan);
        System.out.println("");
            //datapengambilan
        i=0;j=0;
        while(datapengambilan.hasNext()){
            String a=datapengambilan.next();
            double b=Double.parseDouble(a.trim());
            pengambilan[i][j++]=b;
        }

        System.out.println("");
        System.out.println("\t-------------------------------------");
        System.out.println("\t          Data pengambilan          ");
        System.out.println("\t-------------------------------------");
        System.out.println("");
        for(i=0;i<1;i++){
            for(j=0;j<=pelanggan;j++)
                System.out.print("\t"+pengambilan[i][j]);
            System.out.println();
        }
        for(i=0;i<1;i++){
            for(j=0;j<=pelanggan;j++){
                totalpengambilan=totalpengambilan+pengambilan[i][j];
            }
        }
        System.out.println("");
        System.out.println("\ttotal pengambilan :\t"+totalpengambilan);
        System.out.println("");
    }

//input parameter
```

Lampiran 9-5

```
void Inputparameter(){
   int check=1;
   //kapasitas maksimum kendaraan
   System.out.println("");
   System.out.println("\t-------------------------------------");
   System.out.println("\t          INPUT PARAMETER          ");
   System.out.println("\t-------------------------------------");
   System.out.println("");
   do{
      System.out.print("\tMasukkan kapasitas maksimum kendaraan :\t\t\t");
      if(input.hasNextInt()){
         maxkapasitas=input.nextInt();
         if(maxkapasitas>0)
            check=1;
         else
            check=0;
      }
      else
         {String bad=input.next(); check=0;}
      if(check==0)
         System.out.println("\tSalah input, ulangi");
   }while(check==0);

   //presentase jumlah barang yang diangkut dari depot
   do{
      System.out.print("\tMasukkan presentase barang yang diangkut dari depot :\t");
      if(input.hasNextInt()){
         presentase=input.nextInt();
         if(presentase>0 && presentase<=100 )
            check=1;
         else
            check=0;
      }
      else
         {String bad=input.next(); check=0;}
      if(check==0)
         System.out.println("\tSalah input, ulangi");
   }while (check==0);
   jumawal=(presentase*maxkapasitas)/100;

   System.out.println("\tJumlah barang yang dikirim :\t\t\t\t"+jumawal);

   //jumlah kendaraan
   do{
      System.out.print("\tMasukkan jumlah kendaraan yang tersedia :\t\t");
      if(input.hasNextInt()){
         kendaraan=input.nextInt();
         if(kendaraan>0)
            check=1;
         else
            check=0;
      }
      else
         {String bad=input.next(); check=0;}
```

```
   if(check==0)
      System.out.println("\tSalah input, ulangi");
}while(check==0);

//banyak bunga
do{
   System.out.print("\tMasukkan banyak bunga [1,1000] :\t\t\t");
   if(input.hasNextInt()){
      bunga=input.nextInt();
      if(bunga>0)
         check=1;
      else
         check=0;
   }
   else
      {String bad=input.next(); check=0;}
   if(check==0)
      System.out.println("\tSalah input, ulangi");
}while(check==0);
//stepsize
do{
   System.out.print("\tMasukkan nilai stepsize [0,1] :\t\t\t\t");
   if(input.hasNextDouble()){
      alpha=input.nextDouble();
      if(alpha>=0 && alpha<=1)
         check=1;
      else
         check=0;
   }
   else
      {String bad=input.next();check=0;}
   if(check==0)
      System.out.println("\tSalah input, ulangi");
}while(check==0);
//switch probability
do{
   System.out.print("\tMasukkan nilai switch probability :\t\t\t");
   if(input.hasNextDouble()){
      sp=input.nextDouble();
      if(sp>=0 && sp<=1)
         check=1;
      else
         check=0;
   }
   else
      {String bas=input.next(); check=0;}
   if(check==0)
      System.out.println("\tSalah input, ulangi");
}while (check==0);
//beta
do{
   System.out.print("\tMasukkan nilai beta :\t\t\t\t\t");
   if(input.hasNextDouble()){
      lamda=input.nextDouble();
```

```
            if(lamda>=1 && lamda<=2)
               check=1;
            else
               check=0;
         }
         else
            {String bad=input.next(); check=0;}
         if(check==0)
            System.out.println("\tSalah input, ulangi");
      }while (check==0);
      //iterasi
      do{
         System.out.print("\tMasukkan banyaknya iterasi :\t\t\t\t");
         if(input.hasNextInt()){
            maxiterasi=input.nextInt();
            if(maxiterasi>0)
               check=1;
            else
               check=0;
         }
         else
            {String bad=input.next(); check=0;}
         if(check==0)
            System.out.println("\tSalah input, ulangi");
      }while (check==0);
   }

   public double Gamma(double x){
      double[] p = {0.99999999999980993, 676.5203681218851, -1259.1392167224028,
                    771.32342877765313, -176.61502916214059, 12.507343278686905,
                    -0.13857109526572012, 9.9843695780195716e-6, 1.5056327351493116e-7};
      int g = 7;
      if(x < 0.5) return Math.PI / (Math.sin(Math.PI*x)*Gamma(1-x));
      x -= 1;
      double a = p[0];
      double t = x+g+0.5;
      for(int b=1;b < p.length;b++)
         a +=p[b]/(x+b);
      return Math.sqrt(2*Math.PI)*Math.pow(t,x+0.5)*Math.exp(-t)*a;

//membangkitkan populasi
   void Bangkitpopulasi() {
      Random r=new Random();
      //System.out.println("");
      //System.out.println("\t-----------------------------------------------");
      //System.out.println("\t|        HASIL DARI PROSES ALGORITMA        |");
      //System.out.println("\t-----------------------------------------------");
      //System.out.println("");
      //System.out.println("----------------------");
      //System.out.println("   Iterasi ke-1  ");
      //System.out.println("----------------------");
      //System.out.println("");
      for(i=1;i<=bunga;i++){
         //System.out.print("\tBunga ke-"+i+" :");
```

```
            for(j=1;j<=pelanggan;j++){
                random=r.nextInt(1000);
                ba=random;
                xsol[1][i][j]=ba/1000;
                //System.out.print("\t"+xsol[1][i][j]);
            }//System.out.println("");
        }
        //System.out.println("");
        stop=1;
        do{
            //System.out.print("\tBunga ke-"+stop+" :");
            for(j=1;j<=pelanggan;j++){
                urut=1;
                for(k=1;k<=pelanggan;k++){
                    if(xsol[1][stop][j]>xsol[1][stop][k]){
                        urut=urut+1;
                    }urutrandom[1][stop][j]=urut;
                    duplikat[1][stop][j]=urut;
                }//System.out.print("\t"+urutrandom[1][stop][j]);
            }
            //System.out.println("");
            stop++;
        }while(stop<=bunga);

    }

void Fungsitujuan(){
        stop=1;
        do{
            duplikat[1][stop][0]=0;
            jumper[stop][0]=jumawal;
            bsi[stop][0]=jumawal;
            jrk=0;
            x=0;
            y=0;
            ttl=0;
            //System.out.println("");
            //System.out.println("\t-------------------------------------------------------------");
            //System.out.println("\t    Rute dan fitness pada iterasi ke-1 bunga ke-"+stop     );
            //System.out.println("\t-------------------------------------------------------------");
            //System.out.println("");
            for(j=0;j<pelanggan;j++){
                x=x+1;
                if(x==1)
                {
                    //System.out.print("\tRute kendaraan ke-"+(y+1)+" :\t0-");
                }
                jumper[stop][j+1]=jumper[stop][j]-permintaan[0][duplikat[1][stop][j+1]];
                bsi[stop][j+1]=bsi[stop][j]-
permintaan[0][duplikat[1][stop][j+1]]+pengambilan[0][duplikat[1][stop][j+1]];
                //System.out.print(duplikat[1][stop][j+1]+"-");
                jrk=jrk+jarak[duplikat[1][stop][j]][duplikat[1][stop][j+1]];
```

```
            if(jumper[stop][j+1]<permintaan[0][duplikat[1][stop][j+1]]  ||  bsi[stop][j+1]-
permintaan[0][duplikat[1][stop][j+2]]+pengambilan[0][duplikat[1][stop][j+2]]>maxkapasitas
|| j==pelanggan-1)
            {
               //System.out.println("0");
               jumper[stop][j+1]=jumawal;
               bsi[stop][j+1]=jumawal;
               tambahan=jarak[duplikat[1][stop][j+1]][0];
               jrk=jrk+tambahan;
               //System.out.println("\tJarak :\t"+jrk);
               //System.out.println("");
               ttl=ttl+jrk;
               jrk=0;
               x=0;
               y=y+1;
               duplikat[1][stop][j+1]=0;
               }
         }
         total[1][stop]=ttl;
         //System.out.println("\tTotal jarak :\t"+total[1][stop]);
         stop++;
      }while(stop<=bunga);

      gbest=5000;

      for(i=1;i<=bunga;i++){
         if(total[1][i]<gbest){
            gbest=total[1][i];
            indeks=i;
         }
      }
      //System.out.println("\tSolusi terbaik pada iterasi ke-1 : "+gbest+" pada bunga ke-
"+indeks);
      //System.out.println("");
   }

void Iterasi() {
      double [] u = new double[1001], V = new double[1001], E = new double[1001], S = new
double[1001],levy = new double [1001];

      iterasi=2;
      System.out.println("\t----------------------------------------");
      System.out.println("\t        PROSES PENYERBUKAN          ");
      System.out.println("\t----------------------------------------");
      do{
         stop=1;
         Random r=new Random();
         System.out.println("---------------------");
         System.out.println("   Iterasi ke-"+(iterasi-1));
         System.out.println("---------------------");
         //System.out.println("\t  Bunga ke-\t|  Nilai Random\t|  Jenis Penyerbukan\t");
         //System.out.println("\t  ------------------------------------------------");
         do{
            random=r.nextInt(1000);
```

```
ba=random;
prob[stop]=ba/1000;
Random normal = new Random();
//System.out.println("");
//penyerbukan global
int check=1;
if(prob[stop]==sp){
   do{
      random=r.nextInt(1000);
      ba=random;
      prob[stop]=ba/1000;
      if(prob[stop]==sp){
         check=0;
         }
      else{
         check=1;
      }
   } while(check==0);
}
if(prob[stop]<sp){
   //posisi global
   //System.out.println("\t\t"+stop+"\t|  "+prob[stop]+"\t|  Global");
   rad=Math.toRadians((lamda*Math.PI)/2);

sigma=(Gamma(1+lamda)*Math.sin(rad))/(Gamma((1+lamda)/2)*lamda*Math.pow(2,(lamd
a-1)/2));
      sigma=Math.pow(sigma,(1/lamda));
      for(j=1;j<=(pelanggan);j++){
         u[j]=Math.random();
         V[j]=normal.nextGaussian();
         S[j]=(u[j]*sigma)/Math.pow(Math.abs(V[j]),1/lamda);

levy[j]=lamda*Gamma(lamda)*Math.sin(rad)/(Math.PI*Math.pow(S[j],(1+lamda)));
         xsol[iterasi][stop][j]=  xsol[iterasi-1][stop][j]+(alpha*levy[j]*(xsol[iterasi-
1][indeks][j]-xsol[iterasi-1][stop][j]));
         //System.out.print("\t"+xsol[iterasi][stop][j]);

      }
      //permutasi global
      //System.out.print("\tBunga ke-"+stop+" :");
      for(j=1;j<=pelanggan;j++){
      urut=1;
      for(k=1;k<=pelanggan;k++){
      if(xsol[iterasi][stop][j]>xsol[iterasi][stop][k]){
         urut=urut+1;
            }urutrandom[iterasi][stop][j]=urut;
            duplikat[iterasi][stop][j]=urut;
      }//System.out.print("\t"+urutrandom[iterasi][stop][j]);
      }
      //System.out.println("");
      //rute dan fitness global
      duplikat[iterasi][stop][0]=0;
      jumper[stop][0]=jumawal;
      bsi[stop][0]=jumawal;
```

```
                jrk=0;
                x=0;
                y=0;
                ttl=0;
                //System.out.println("");
                //System.out.println("\t-------------------------------------------------------------");
                //System.out.println("\t     Rute dan fitness pada iterasi ke-"+ (iterasi-1)+" bunga
ke-"+stop     );
                //System.out.println("\t-------------------------------------------------------------");
                //System.out.println("");
                for(j=0;j<pelanggan;j++){
                x=x+1;
                if(x==1)
                {
                   //System.out.print("\tRute kendaraan ke-"+(y+1)+" :\t0-");
                   }
                jumper[stop][j+1]=jumper[stop][j]-permintaan[0][duplikat[iterasi][stop][j+1]];
                bsi[stop][j+1]=bsi[stop][j]-
permintaan[0][duplikat[iterasi][stop][j+1]]+pengambilan[0][duplikat[iterasi][stop][j+1]];
                //System.out.print(duplikat[iterasi][stop][j+1]+"-");
                jrk=jrk+jarak[duplikat[iterasi][stop][j]][duplikat[iterasi][stop][j+1]];
                if(jumper[stop][j+1]<permintaan[0][duplikat[iterasi][stop][j+1]] || bsi[stop][j+1]-
permintaan[0][duplikat[iterasi][stop][j+2]]+pengambilan[0][duplikat[iterasi][stop][j+2]]>ma
xkapasitas || j==pelanggan-1)
                {
                   //System.out.println("0");
                   jumper[stop][j+1]=jumawal;
                   bsi[stop][j+1]=jumawal;
                   tambahan=jarak[duplikat[iterasi][stop][j+1]][0];
                   jrk=jrk+tambahan;
                   //System.out.println("\tJarak :\t"+jrk);
                   //System.out.println("");
                   ttl=ttl+jrk;
                   jrk=0;
                   x=0;
                   y=y+1;
                   duplikat[iterasi][stop][j+1]=0;
                   }
                }
                total[iterasi][stop]=ttl;
                //System.out.println("\tTotal jarak :\t"+total[iterasi][stop]);
                //membandingkan dengan solusi lama
                if(total[iterasi][stop]>total[iterasi-1][stop]){
                   for(j=1;j<=pelanggan;j++){
                      xsol[iterasi][stop][j]=xsol[iterasi-1][stop][j];
                      urutrandom[iterasi][stop][j]=urutrandom[iterasi-1][stop][j];
                      //System.out.print("\t"+xsol[iterasi][stop][j]);
                   }
                   total[iterasi][stop]=total[iterasi-1][stop];
                   //System.out.println("");
                   //System.out.println("\tSolusi pada iterasi ke-"+(iterasi-1)+" bunga ke-"+stop+"
terletak pada solusi lama");
                }
                else{
```

```
                for(j=1;j<=pelanggan;j++){
                    xsol[iterasi][stop][j]=xsol[iterasi][stop][j];
                    urutrandom[iterasi][stop][j]=urutrandom[iterasi][stop][j];
                    //System.out.print("\t"+xsol[iterasi][stop][j]);
                }
                total[iterasi][stop]=total[iterasi][stop];
                //System.out.println("");
                //System.out.println("\tSolusi pada iterasi ke-"+(iterasi-1)+" bunga ke-"+stop+"
terletak pada solusi baru");
            }


        }
        //penyerbukan lokal
        if(prob[stop]>sp){
            //System.out.println("\t\t"+stop+"\t|  "+prob[stop]+"\t|  Lokal");
            Random lokal=new Random();
            do{
                for(i=1;i<=2;i++){
                bungalokal[i]=lokal.nextInt(bunga+1);
                }
                if(bungalokal[1]==bungalokal[2] || bungalokal[1]==0 || bungalokal[2]==0)
                {
                    check=0;
                    }
                else
                {
                    check=1;
                }
            }while(check==0);

            for(j=1;j<=pelanggan;j++){
            E[j]=normal.nextGaussian();
            xsol[iterasi][stop][j]=xsol[iterasi-1][stop][j]+(E[j]*(xsol[iterasi-
1][bungalokal[1]][j]-xsol[iterasi-1][bungalokal[2]][j]));
            //System.out.print("\t"+xsol[iterasi][stop][j]);
                }
            //System.out.println("");
            //System.out.print("\tBunga ke-"+stop+" :");
            for(j=1;j<=pelanggan;j++){
            urut=1;
            for(k=1;k<=pelanggan;k++){
            if(xsol[iterasi][stop][j]>xsol[iterasi][stop][k]){
                urut=urut+1;
                    }urutrandom[iterasi][stop][j]=urut;
                    duplikat[iterasi][stop][j]=urut;
                }//System.out.print("\t"+urutrandom[iterasi][stop][j]);
            }
            //System.out.println("");

            //rute dan fitness lokal
            duplikat[iterasi][stop][0]=0;
            jumper[stop][0]=jumawal;
            bsi[stop][0]=jumawal;
```

```
                jrk=0;
                x=0;
                y=0;
                ttl=0;
                //System.out.println("");
                //System.out.println("\t-------------------------------------------------------------");
                //System.out.println("\t     Rute dan fitness pada iterasi ke-"+ (iterasi-1)+" bunga
ke-"+stop     );
                //System.out.println("\t-------------------------------------------------------------");
                //System.out.println("");
                for(j=0;j<pelanggan;j++){
                x=x+1;
                if(x==1)
                {
                   //System.out.print("\tRute kendaraan ke-"+(y+1)+" :\t0-");
                   }
                jumper[stop][j+1]=jumper[stop][j]-permintaan[0][duplikat[iterasi][stop][j+1]];
                bsi[stop][j+1]=bsi[stop][j]-
permintaan[0][duplikat[iterasi][stop][j+1]]+pengambilan[0][duplikat[iterasi][stop][j+1]];
                //System.out.print(duplikat[iterasi][stop][j+1]+"-");
                jrk=jrk+jarak[duplikat[iterasi][stop][j]][duplikat[iterasi][stop][j+1]];
                if(jumper[stop][j+1]<permintaan[0][duplikat[iterasi][stop][j+1]] || bsi[stop][j+1]-
permintaan[0][duplikat[iterasi][stop][j+2]]+pengambilan[0][duplikat[iterasi][stop][j+2]]>ma
xkapasitas || j==pelanggan-1)
                {
                   //System.out.println("0");
                   jumper[stop][j+1]=jumawal;
                   bsi[stop][j+1]=jumawal;
                   tambahan=jarak[duplikat[iterasi][stop][j+1]][0];
                   jrk=jrk+tambahan;
                   //System.out.println("\tJarak :\t"+jrk);
                   //System.out.println("");
                   ttl=ttl+jrk;
                   jrk=0;
                   x=0;
                   y=y+1;
                   duplikat[iterasi][stop][j+1]=0;
                   }
                }
                total[iterasi][stop]=ttl;
                //System.out.println("\tTotal jarak :\t"+total[iterasi][stop]);
                //membandingkan dengan solusi lama
                if(total[iterasi][stop]>total[iterasi-1][stop]){
                   for(j=1;j<=pelanggan;j++){
                      xsol[iterasi][stop][j]=xsol[iterasi-1][stop][j];
                      urutrandom[iterasi][stop][j]=urutrandom[iterasi-1][stop][j];
                      //System.out.print("\t"+xsol[iterasi][stop][j]);
                   }
                   total[iterasi][stop]=total[iterasi-1][stop];
                   //System.out.println("");
                   //System.out.println("\tSolusi pada iterasi ke-"+(iterasi-1)+" bunga ke-"+stop+"
terletak pada solusi lama");
                }
                else{
```

```
                for(j=1;j<=pelanggan;j++){
                   xsol[iterasi][stop][j]=xsol[iterasi][stop][j];
                   urutrandom[iterasi][stop][j]=urutrandom[iterasi][stop][j];
                   //System.out.print("\t"+xsol[iterasi][stop][j]);
                }
                total[iterasi][stop]=total[iterasi][stop];
                //System.out.println("");
                //System.out.println("\tSolusi pada iterasi ke-"+(iterasi-1)+" bunga ke-"+stop+"
terletak pada solusi baru");
             }

          }

          stop++;
       }while(stop<=bunga);

       for(i=1;i<=bunga;i++){
          for(j=1;j<=pelanggan;j++){
             //System.out.print("\t"+xsol[iterasi][i][j]);
          }//System.out.println("");
       }

       for(i=1;i<=bunga;i++){
          //System.out.println("\t"+total[iterasi][i]);
       }

       gbest=5000;

       for(i=1;i<=bunga;i++){
       if(total[iterasi][i]<gbest){
          gbest=total[iterasi][i];
          indeks=i;
          }
       }
       if(iterasi==maxiterasi+1){
       System.out.println("\tSolusi terbaik pada iterasi ke-"+(iterasi-1)+" : "+gbest+" pada
bunga ke-"+indeks);
             urutrandom[iterasi][indeks][0]=0;
             jumper[indeks][0]=jumawal;
             bsi[indeks][0]=jumawal;
             jrk=0;
             x=0;
             y=0;
             ttl=0;
             System.out.println("");
             //System.out.println("\t-------------------------------------------------------------");
             System.out.println("dengan rute sebagai berikut :");
             //System.out.println("\t-------------------------------------------------------------");
             System.out.println("");
             for(j=0;j<pelanggan;j++){
             x=x+1;
             if(x==1)
             {
                System.out.print("\tRute kendaraan ke-"+(y+1)+" :\t0-");
```

```
                }
                jumper[indeks][j+1]=jumper[indeks][j]-
permintaan[0][urutrandom[iterasi][indeks][j+1]];
                bsi[indeks][j+1]=bsi[indeks][j]-
permintaan[0][urutrandom[iterasi][indeks][j+1]]+pengambilan[0][urutrandom[iterasi][indeks
][j+1]];
                System.out.print(urutrandom[iterasi][indeks][j+1]+"-");
                jrk=jrk+jarak[urutrandom[iterasi][indeks][j]][urutrandom[iterasi][indeks][j+1]];
                if(jumper[indeks][j+1]<permintaan[0][urutrandom[iterasi][indeks][j+1]]          ||
bsi[indeks][j+1]-
permintaan[0][urutrandom[iterasi][indeks][j+2]]+pengambilan[0][urutrandom[iterasi][indeks
][j+2]]>maxkapasitas || j==pelanggan-1)
                {
                   System.out.println("0");
                   jumper[indeks][j+1]=jumawal;
                   bsi[indeks][j+1]=jumawal;
                   tambahan=jarak[urutrandom[iterasi][indeks][j+1]][0];
                   jrk=jrk+tambahan;
                   //System.out.println("\tJarak :\t"+jrk);
                   //System.out.println("");
                   ttl=ttl+jrk;
                   jrk=0;
                   x=0;
                   y=y+1;
                   urutrandom[iterasi][indeks][j+1]=0;
                   }
                }
                total[iterasi][indeks]=ttl;
                //System.out.println("\tTotal jarak :\t"+total[iterasi][indeks]);
          }
          else{
             System.out.println("\tSolusi terbaik sementara pada iterasi ke-"+(iterasi-1)+" :
"+gbest+" pada bunga ke-"+indeks);
             System.out.println("");
          }


          iterasi++;
       }while(iterasi<=maxiterasi+1);

    }

public static void main(String[] args) throws IOException {
       SKRIPSI kode = new SKRIPSI();
       kode.Pilihdata();
       kode.Inisialisasidata(pilihan);
       kode.Inputparameter();
       kode.Bangkitpopulasi();
       kode.Fungsitujuan();
       if(maxiterasi>1){
          kode.Iterasi();
       }
    }
```

Lampiran 10 : Hasil *Running* Data dengan 13 Pelanggan

1. Bunga 10, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 124. Rute :
   - Rute kendaraan ke-1 : 0-7-6-5-2-0
   - Rute kendaraan ke-2 : 0-12-10-9-8-0
   - Rute kendaraan ke-3 : 0-3-4-11-1-0
   - Rute kendaraan ke-4 : 0-13-0
2. Bunga 10, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 121. Rute :
   - Rute kendaraan ke-1 : 0-4-3-2-1-0
   - Rute kendaraan ke-2 : 0-8-12-10-11-13-0
   - Rute kendaraan ke-3 : 0-5-6-9-7-0
3. Bunga 10, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 120. Rute :
   - Rute kendaraan ke-1 : 0-1-8-10-9-0
   - Rute kendaraan ke-2 : 0-13-12-11-2-0
   - Rute kendaraan ke-3 : 0-4-3-5-6-7-0
4. Bunga 10, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 113. Rute :
   - Rute kendaraan ke-1 : 0-11-8-7-12-0
   - Rute kendaraan ke-2 : 0-10-5-6-9-13-0
   - Rute kendaraan ke-3 : 0-4-3-2-1-0
5. Bunga 10, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 110. Rute :
   - Rute kendaraan ke-1 : 0-7-6-5-3-4-0
   - Rute kendaraan ke-2 : 0-1-2-11-10-9-0
   - Rute kendaraan ke-3 : 0-13-12-8-0
6. Bunga 10, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 104. Rute :
   - Rute kendaraan ke-1 : 0-4-5-3-10-0
   - Rute kendaraan ke-2 : 0-12-8-9-6-7-0
   - Rute kendaraan ke-3 : 0-11-2-1-13-0
7. Bunga 10, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 103. Rute :
   - Rute kendaraan ke-1 : 0-10-6-7-9-0
   - Rute kendaraan ke-2 : 0-1-2-3-5-4-0
   - Rute kendaraan ke-3 : 0-13-12-8-11-0
8. Bunga 10, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 100. Rute :
   - Rute kendaraan ke-1 : 0-12-9-7-8-0
   - Rute kendaraan ke-2 : 0-3-4-5-6-10-0
   - Rute kendaraan ke-3 : 0-11-2-1-13-0
9. Bunga 10, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 99. Rute :
   - Rute kendaraan ke-1 : 0-12-9-7-8-0
   - Rute kendaraan ke-2 : 0-3-4-5-6-10-0

Rute kendaraan ke-3 : 0-13-11-2-1-0

10. Bunga 50, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 113. Rute :
    Rute kendaraan ke-1 : 0-7-6-5-10-12-0
    Rute kendaraan ke-2 : 0-11-8-9-3-4-0
    Rute kendaraan ke-3 : 0-2-1-13-0
11. Bunga 50, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 112. Rute :
    Rute kendaraan ke-1 : 0-3-4-2-1-0
    Rute kendaraan ke-2 : 0-13-12-10-5-11-0
    Rute kendaraan ke-3 : 0-9-6-7-8-0
12. Bunga 50, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 112. Rute :
    Rute kendaraan ke-1 : 0-1-10-11-2-0
    Rute kendaraan ke-2 : 0-3-4-5-6-9-0
    Rute kendaraan ke-3 : 0-8-7-12-13-0Bunga 50, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 105. Rute :
    Rute kendaraan ke-1 : 0-3-4-5-6-10-0
    Rute kendaraan ke-2 : 0-12-9-7-13-0
    Rute kendaraan ke-3 : 0-8-11-2-1-0
13. Bunga 50, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 104. Rute :
    Rute kendaraan ke-1 : 0-12-9-7-8-0
    Rute kendaraan ke-2 : 0-10-3-6-5-4-0
    Rute kendaraan ke-3 : 0-1-2-11-13-0
14. Bunga 50, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 104. Rute :
    Rute kendaraan ke-1 : 0-7-6-5-4-0
    Rute kendaraan ke-2 : 0-1-2-3-10-11-0
    Rute kendaraan ke-3 : 0-9-8-12-13-0
15. Bunga 50, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 102. Rute :
    Rute kendaraan ke-1 : 0-1-2-11-12-13-0
    Rute kendaraan ke-2 : 0-9-7-8-0
    Rute kendaraan ke-3 : 0-3-4-5-6-10-0
16. Bunga 50, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 102. Rute :
    Rute kendaraan ke-1 : 0-1-2-11-12-13-0
    Rute kendaraan ke-2 : 0-8-7-9-0
    Rute kendaraan ke-3 : 0-3-4-5-6-10-0
17. Bunga 50, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 102. Rute :
    Rute kendaraan ke-1 : 0-1-2-11-12-13-0
    Rute kendaraan ke-2 : 0-9-7-8-0
    Rute kendaraan ke-3 : 0-3-4-5-6-10-0

18. Bunga 100, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 115. Rute :
    Rute kendaraan ke-2 : 0-9-5-3-10-8-0
    Rute kendaraan ke-3 : 0-11-2-1-13-0
19. Bunga 100, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 115. Rute :
    Rute kendaraan ke-1 : 0-9-8-12-13-0
    Rute kendaraan ke-2 : 0-5-4-3-6-7-0
    Rute kendaraan ke-3 : 0-10-1-2-11-0
20. Bunga 100, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 107. Rute :
    Rute kendaraan ke-1 : 0-9-7-6-8-0
    Rute kendaraan ke-2 : 0-5-4-3-10-0
    Rute kendaraan ke-3 : 0-1-2-11-12-13-0
21. Bunga 100, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 107. Rute :
    Rute kendaraan ke-1 : 0-1-2-11-12-13-0
    Rute kendaraan ke-2 : 0-8-7-9-0
    Rute kendaraan ke-3 : 0-4-5-6-3-10-0
22. Bunga 100, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 107. Rute :
    Rute kendaraan ke-1 : 0-12-10-2-1-0
    Rute kendaraan ke-2 : 0-3-4-5-6-9-0
    Rute kendaraan ke-3 : 0-11-7-8-13-0
23. Bunga 100, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 101. Rute :
    Rute kendaraan ke-1 : 0-3-4-5-6-7-0
    Rute kendaraan ke-2 : 0-10-9-8-12-0
    Rute kendaraan ke-3 : 0-11-2-1-13-0
24. Bunga 100, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 99. Rute :
    Rute kendaraan ke-1 : 0-12-9-7-8-0
    Rute kendaraan ke-2 : 0-3-4-5-6-10-0
    Rute kendaraan ke-3 : 0-13-11-2-1-0
25. Bunga 100, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 99. Rute :
    Rute kendaraan ke-1 : 0-12-9-7-8-0
    Rute kendaraan ke-2 : 0-3-4-5-6-10-0
    Rute kendaraan ke-3 : 0-13-11-2-1-0
26. Bunga 100, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 99. Rute :
    Rute kendaraan ke-1 : 0-12-9-7-8-0
    Rute kendaraan ke-2 : 0-3-4-5-6-10-0
    Rute kendaraan ke-3 : 0-1-2-11-13-0

Lampiran 11 : Hasil *Running* Data dengan 22 Pelanggan

1. Bunga 10, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 159. Rute :
   Rute kendaraan ke-1 : 0-4-2-22-21-8-17-14-20-0
   Rute kendaraan ke-2 : 0-15-11-6-5-13-7-10-12-0
   Rute kendaraan ke-3 : 0-1-19-16-9-3-18-0
2. Bunga 10, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 144. Rute :
   Rute kendaraan ke-1 : 0-1-6-14-8-10-0
   Rute kendaraan ke-2 : 0-13-20-18-5-3-22-21-4-0
   Rute kendaraan ke-3 : 0-12-7-11-15-17-19-16-9-2-0
3. Bunga 10, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 152. Rute :
   Rute kendaraan ke-1 : 0-1-17-2-21-4-0
   Rute kendaraan ke-2 : 0-7-13-10-16-5-3-8-22-0
   Rute kendaraan ke-3 : 0-20-9-6-14-11-15-19-18-12-0
4. Bunga 10, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 156. Rute :
   Rute kendaraan ke-1 : 0-8-16-10-2-13-7-4-0
   Rute kendaraan ke-2 : 0-14-5-3-22-21-9-19-18-20-12-0
   Rute kendaraan ke-3 : 0-17-11-15-6-1-0
5. Bunga 10, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 155. Rute :
   Rute kendaraan ke-1 : 0-20-1-14-6-11-7-0
   Rute kendaraan ke-2 : 0-10-12-9-21-13-17-18-16-4-0
   Rute kendaraan ke-3 : 0-22-8-3-2-15-5-19-0
6. Bunga 10, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 152. Rute :
   Rute kendaraan ke-1 : 0-13-17-9-2-3-5-7-10-0
   Rute kendaraan ke-2 : 0-1-18-6-14-4-0
   Rute kendaraan ke-3 : 0-15-19-11-22-8-21-16-20-12-0
7. Bunga 10, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 140. Rute :
   Rute kendaraan ke-1 : 0-18-9-8-11-13-20-17-14-7-0
   Rute kendaraan ke-2 : 0-4-16-2-6-1-0
   Rute kendaraan ke-3 : 0-12-19-15-5-3-22-21-10-0
8. Bunga 10, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 140. Rute :
   Rute kendaraan ke-1 : 0-10-9-5-3-6-15-11-7-0
   Rute kendaraan ke-2 : 0-13-18-14-19-1-0
   Rute kendaraan ke-3 : 0-12-16-17-20-4-2-22-21-8-0
9. Bunga 10, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 130. Rute :
   Rute kendaraan ke-1 : 0-19-14-6-15-8-16-18-1-0
   Rute kendaraan ke-2 : 0-4-21-22-2-3-5-9-0
   Rute kendaraan ke-3 : 0-12-17-20-13-11-10-7-0

10. Bunga 50, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 164. Rute :
    Rute kendaraan ke-1 : 0-7-19-18-6-15-3-5-16-10-0
    Rute kendaraan ke-2 : 0-1-14-22-11-13-4-0
    Rute kendaraan ke-3 : 0-12-2-17-9-20-8-21-0
11. Bunga 50, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 166. Rute :
    Rute kendaraan ke-1 : 0-10-12-1-5-3-18-16-4-0
    Rute kendaraan ke-2 : 0-15-17-14-11-19-21-22-9-20-7-0
    Rute kendaraan ke-3 : 0-13-6-8-2-0
12. Bunga 50, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 164. Rute :
    Rute kendaraan ke-1 : 0-6-19-10-12-4-9-17-16-7-0
    Rute kendaraan ke-2 : 0-13-11-14-2-5-3-15-1-0
    Rute kendaraan ke-3 : 0-20-21-8-22-18-0
13. Bunga 50, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 154. Rute :
    Rute kendaraan ke-1 : 0-10-4-2-21-5-3-13-0
    Rute kendaraan ke-2 : 0-17-18-22-8-15-6-14-19-7-0
    Rute kendaraan ke-3 : 0-20-16-9-12-11-1-0
14. Bunga 50, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 154. Rute :
    Rute kendaraan ke-1 : 0-12-16-6-15-19-11-1-0
    Rute kendaraan ke-2 : 0-8-3-5-9-13-18-17-20-14-0
    Rute kendaraan ke-3 : 0-7-2-22-21-10-4-0
15. Bunga 50, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 154. Rute :
    Rute kendaraan ke-1 : 0-4-20-17-16-2-8-22-21-0
    Rute kendaraan ke-2 : 0-14-15-11-9-7-12-19-1-0
    Rute kendaraan ke-3 : 0-18-3-5-6-13-10-0
16. Bunga 50, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 132. Rute :
    Rute kendaraan ke-1 : 0-13-8-14-5-3-22-21-4-0
    Rute kendaraan ke-2 : 0-2-6-15-11-19-7-10-0
    Rute kendaraan ke-3 : 0-20-1-17-18-16-9-12-0
17. Bunga 50, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 132. Rute :
    Rute kendaraan ke-1 : 0-4-8-22-16-2-18-17-14-13-0
    Rute kendaraan ke-2 : 0-20-21-3-5-6-15-1-0
    Rute kendaraan ke-3 : 0-12-19-11-9-7-10-0
18. Bunga 50, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 126. Rute :
    Rute kendaraan ke-1 : 0-12-5-3-22-18-9-16-13-20-7-0
    Rute kendaraan ke-2 : 0-10-11-14-2-8-21-4-0
    Rute kendaraan ke-3 : 0-1-15-6-19-17-0
19. Bunga 100, Iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 163. Rute :

Rute kendaraan ke-1 : 0-10-20-9-17-4-16-2-18-8-0

Rute kendaraan ke-2 : 0-12-19-11-13-22-3-5-1-0

Rute kendaraan ke-3 : 0-7-15-6-14-21-0

20. Bunga 100, Iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 157. Rute :

    Rute kendaraan ke-1 : 0-10-7-11-14-2-8-20-18-0

    Rute kendaraan ke-2 : 0-4-6-19-1-0

    Rute kendaraan ke-3 : 0-17-22-3-5-15-9-21-13-16-12-0

21. Bunga 100, Iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 156. Rute :

    Rute kendaraan ke-1 : 0-10-17-20-14-13-9-6-15-5-0

    Rute kendaraan ke-2 : 0-1-16-2-3-22-4-0

    Rute kendaraan ke-3 : 0-12-21-8-18-19-11-7-0

22. Bunga 100, Iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 149. Rute :

    Rute kendaraan ke-1 : 0-20-13-15-6-5-3-22-17-18-0

    Rute kendaraan ke-2 : 0-12-4-16-19-7-11-10-14-1-0

    Rute kendaraan ke-3 : 0-9-21-8-2-0

23. Bunga 100, Iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 144. Rute :

    Rute kendaraan ke-1 : 0-17-6-18-13-19-1-0

    Rute kendaraan ke-2 : 0-4-10-20-11-15-5-3-12-7-0

    Rute kendaraan ke-3 : 0-16-9-8-21-22-2-14-0

24. Bunga 100, Iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 144. Rute :

    Rute kendaraan ke-1 : 0-17-9-3-5-21-22-2-0

    Rute kendaraan ke-2 : 0-1-14-6-15-19-10-0

    Rute kendaraan ke-3 : 0-12-16-20-11-13-18-8-4-0

    Rute kendaraan ke-4 : 0-7-0

25. Bunga 100, Iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 131. Rute :

    Rute kendaraan ke-1 : 0-7-12-16-9-8-21-18-20-4-0

    Rute kendaraan ke-2 : 0-17-2-22-3-5-6-0

    Rute kendaraan ke-3 : 0-10-19-11-14-15-13-1-0

26. Bunga 100, Iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 128. Rute :

    Rute kendaraan ke-1 : 0-20-18-8-3-5-17-13-1-0

    Rute kendaraan ke-2 : 0-10-19-16-2-22-21-9-4-0

    Rute kendaraan ke-3 : 0-12-7-11-15-6-14-0

27. Bunga 100, Iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 124. Rute :

    Rute kendaraan ke-1 : 0-10-17-15-6-5-3-22-4-0

    Rute kendaraan ke-2 : 0-12-14-13-1-11-19-7-0

    Rute kendaraan ke-3 : 0-20-9-18-16-21-8-2-0

Lampiran 12 : Hasil *Running* Data dengan 100 Pelanggan

1. Bunga 10, iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 3066. Rute :
   Rute kendaraan ke-1 : 0-80-67-63-36-33-47-35-39-13-61-46-24-3-79-62-18-69-83-73-76-68-88-53-10-57-32-31-0
   Rute kendaraan ke-2 : 0-23-16-45-65-89-86-40-100-95-74-87-84-81-51-17-55-49-54-34-29-77-71-90-85-97-38-43-22-50-60-64-91-9-96-27-58-72-82-75-20-0
   Rute kendaraan ke-3 : 0-7-8-93-1-98-56-92-15-25-42-48-14-59-70-94-99-37-28-26-11-21-6-30-5-41-19-78-12-66-0
   Rute kendaraan ke-4 : 0-52-4-44-2-0
2. Bunga 10, iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 3179. Rute :
   Rute kendaraan ke-1 : 0-71-4-35-56-89-44-39-25-100-8-57-54-6-31-38-10-60-23-74-64-12-99-22-7-90-0
   Rute kendaraan ke-2 : 0-21-24-13-29-26-32-42-59-81-73-78-48-14-63-40-41-52-94-80-72-82-96-50-55-43-30-49-84-70-53-47-95-88-46-19-0
   Rute kendaraan ke-3 : 0-34-65-67-76-16-18-87-66-15-36-93-77-91-92-85-83-68-75-98-27-51-3-9-2-97-5-45-28-69-33-37-58-11-1-0
   Rute kendaraan ke-4 : 0-61-62-79-86-17-20-0
3. Bunga 10, iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 3111. Rute :
   Rute kendaraan ke-1 : 0-50-45-98-36-33-54-21-31-6-85-12-4-92-55-28-35-48-60-37-52-51-91-42-29-19-67-96-95-58-0
   Rute kendaraan ke-2 : 0-74-3-14-63-97-26-65-83-59-22-24-32-41-75-93-88-82-61-62-72-49-18-23-73-87-77-11-10-38-40-5-56-0
   Rute kendaraan ke-3 : 0-71-17-20-100-70-76-84-47-94-78-44-43-25-27-34-46-53-68-64-1-80-79-81-57-2-89-16-99-8-30-7-66-13-90-86-0
   Rute kendaraan ke-4 : 0-69-15-9-39-0
4. Bunga 10, iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 2840. Rute :
   Rute kendaraan ke-1 : 0-48-41-28-32-93-30-17-8-50-70-80-81-62-54-45-9-73-66-51-22-6-12-38-55-3-40-0
   Rute kendaraan ke-2 : 0-18-33-69-56-31-92-67-35-14-16-39-15-58-84-94-7-4-91-76-85-34-23-27-47-96-88-64-43-19-89-98-42-83-79-78-77-37-36-0
   Rute kendaraan ke-3 : 0-29-75-99-100-72-74-61-46-13-90-5-1-57-20-2-44-10-21-25-63-71-97-49-59-65-60-82-0
   Rute kendaraan ke-4 : 0-52-86-68-53-26-87-95-11-24-0
5. Bunga 10, iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 2809. Rute :
   Rute kendaraan ke-1 : 0-73-76-98-22-64-59-63-71-79-53-10-96-42-48-54-20-94-78-70-61-9-29-35-56-24-40-21-0

Rute kendaraan ke-2 : 0-27-2-26-86-31-16-36-18-1-74-3-67-95-72-80-83-55-93-84-8-65-100-41-7-5-38-25-91-62-57-92-4-89-43-0

Rute kendaraan ke-3 : 0-46-58-47-6-66-97-85-88-81-82-52-68-69-87-33-30-34-99-90-77-17-60-15-12-19-14-23-37-32-13-28-39-11-50-44-0

Rute kendaraan ke-4 : 0-45-49-51-75-0

6. Bunga 10, iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 2803. Rute :

   Rute kendaraan ke-1 : 0-60-1-77-18-27-45-69-61-68-59-62-86-67-85-21-16-32-14-42-39-13-56-19-38-97-96-80-87-84-92-3-2-12-10-0

   Rute kendaraan ke-2 : 0-7-66-58-33-64-48-98-9-50-49-28-52-20-30-23-17-51-44-73-94-34-95-54-5-99-53-0

   Rute kendaraan ke-3 : 0-25-46-72-78-81-82-43-11-83-88-70-55-57-63-76-71-75-90-89-8-100-36-37-26-22-29-41-79-74-91-4-40-35-24-0

   Rute kendaraan ke-4 : 0-47-93-6-15-31-65-0

7. Bunga 10, iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 2642. Rute :

   Rute kendaraan ke-1 : 0-35-76-71-52-32-30-45-17-27-26-39-36-19-51-24-43-98-91-28-37-10-40-11-12-0

   Rute kendaraan ke-2 : 0-3-99-61-53-42-15-16-62-4-89-59-23-79-66-68-94-1-22-31-50-25-46-47-58-34-0

   Rute kendaraan ke-3 : 0-20-18-95-93-69-75-63-13-2-81-70-49-7-83-97-82-80-86-85-88-90-92-84-100-60-33-14-56-54-74-67-73-72-87-64-38-9-55-6-77-57-5-96-78-48-8-0

   Rute kendaraan ke-4 : 0-41-29-21-65-44-0

8. Bunga 10, iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 2630. Rute :

   Rute kendaraan ke-1 : 0-19-31-75-5-33-36-18-60-55-68-42-27-6-26-37-66-64-67-94-100-16-40-62-0

   Rute kendaraan ke-2 : 0-10-29-82-50-7-99-81-70-85-96-76-4-47-84-95-83-77-22-23-56-24-39-32-30-20-46-28-61-8-88-52-2-59-71-78-53-93-51-45-0

   Rute kendaraan ke-3 : 0-54-11-43-25-12-35-38-57-21-86-72-63-97-3-92-65-14-89-58-9-34-87-79-73-69-49-98-90-80-74-1-17-0

   Rute kendaraan ke-4 : 0-15-48-13-44-41-91-0

9. Bunga 10, iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 2531. Rute :

   Rute kendaraan ke-1 : 0-93-57-80-76-91-83-23-24-11-46-20-48-17-41-15-27-51-13-16-8-63-42-96-55-73-70-21-0

   Rute kendaraan ke-2 : 0-14-33-35-34-39-25-5-43-64-85-7-84-37-40-88-79-38-59-99-3-50-9-52-18-60-1-81-78-19-26-32-6-0

   Rute kendaraan ke-3 : 0-75-87-71-77-94-92-98-68-45-67-62-86-97-61-22-82-95-4-90-69-56-31-28-2-100-65-54-66-29-30-53-12-47-36-10-58-0

   Rute kendaraan ke-4 : 0-74-44-89-72-49-0

10. Bunga 50, iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 3057. Rute :
    Rute kendaraan ke-1 : 0-61-74-37-40-62-50-57-88-84-82-4-39-93-80-2-89-77-11-76-54-60-17-38-31-9-90-91-44-7-12-27-48-51-78-0
    Rute kendaraan ke-2 : 0-56-35-16-58-100-71-65-49-24-34-73-87-99-95-67-3-98-66-30-20-21-8-19-45-94-83-53-26-36-46-0
    Rute kendaraan ke-3 : 0-43-42-23-32-69-81-79-63-86-85-97-1-22-75-15-41-55-14-25-52-33-28-92-72-70-96-13-18-47-64-59-29-0
    Rute kendaraan ke-4 : 0-6-68-10-5-0
11. Bunga 50, iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 3040. Rute :
    Rute kendaraan ke-1 : 0-23-26-66-45-59-50-38-18-82-74-53-44-10-72-22-4-9-21-56-33-91-67-83-87-29-46-0
    Rute kendaraan ke-2 : 0-54-5-15-14-3-78-76-98-17-63-43-84-25-93-92-62-55-52-88-71-99-8-24-7-49-60-85-79-2-20-97-81-70-1-64-61-57-11-0
    Rute kendaraan ke-3 : 0-42-32-96-77-80-19-27-75-100-90-16-34-47-35-28-12-13-89-95-68-65-51-41-69-73-40-58-48-36-39-0
    Rute kendaraan ke-4 : 0-6-31-37-30-86-94-0
12. Bunga 50, iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 2975. Rute :
    Rute kendaraan ke-1 : 0-27-28-50-72-54-20-42-63-12-85-78-71-97-96-68-22-57-30-91-83-77-59-90-13-37-38-36-44-61-8-7-0
    Rute kendaraan ke-2 : 0-14-10-53-25-92-76-79-49-93-34-48-29-46-19-94-26-73-84-70-58-86-99-64-5-66-98-2-89-1-80-24-17-62-52-0
    Rute kendaraan ke-3 : 0-23-35-41-21-88-75-40-9-31-39-4-60-6-33-65-51-16-43-32-47-69-55-95-67-3-0
    Rute kendaraan ke-4 : 0-56-74-100-87-82-81-15-18-11-45-0
13. Bunga 50, iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 2724. Rute :
    Rute kendaraan ke-1 : 0-99-30-37-18-27-6-40-48-72-44-53-41-10-83-97-88-5-23-19-45-3-54-17-39-85-7-98-15-16-36-0
    Rute kendaraan ke-2 : 0-1-63-75-51-20-49-64-68-58-76-100-81-96-89-69-14-31-25-32-86-28-11-57-9-50-74-71-73-78-77-91-13-0
    Rute kendaraan ke-3 : 0-56-8-29-42-46-52-38-35-24-92-43-59-62-47-93-87-70-2-82-4-12-61-66-33-65-55-95-94-79-80-84-90-67-60-21-0
    Rute kendaraan ke-4 : 0-34-22-26-0
14. Bunga 50, iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 2622. Rute :
    Rute kendaraan ke-1 : 0-92-87-76-21-15-58-50-35-57-75-2-84-5-94-27-22-26-19-32-38-18-45-29-39-7-43-98-11-67-0
    Rute kendaraan ke-2 : 0-65-77-85-4-59-100-53-14-51-48-79-73-63-95-1-74-62-49-81-71-83-80-56-54-69-42-52-20-10-9-86-78-90-97-82-61-64-23-17-0

Rute kendaraan ke-3 : 0-70-44-99-88-46-13-31-24-47-33-36-96-72-60-3-89-66-40-68-55-93-8-6-28-30-34-41-0

Rute kendaraan ke-4 : 0-16-37-25-91-12-0

15. Bunga 50, iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 2622. Rute :

Rute kendaraan ke-1 : 0-88-69-72-93-80-73-57-94-53-21-67-5-68-25-30-50-97-37-6-60-40-59-55-47-52-44-16-2-75-85-77-96-0

Rute kendaraan ke-2 : 0-49-64-19-23-39-32-65-14-83-63-41-42-26-34-9-98-99-81-24-18-28-15-27-46-43-51-74-82-4-0

Rute kendaraan ke-3 : 0-36-38-48-100-10-17-29-22-58-61-20-56-91-66-35-71-84-92-90-3-76-86-87-62-45-54-33-31-13-12-0

Rute kendaraan ke-4 : 0-1-7-11-8-79-95-70-78-89-0

16. Bunga 50, iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 2605. Rute :

Rute kendaraan ke-1 : 0-99-29-46-34-2-11-30-57-61-21-100-67-77-87-3-13-63-20-8-9-55-26-22-40-4-93-88-82-94-90-84-1-53-92-65-91-98-81-85-78-70-76-83-97-79-28-64-74-89-0

Rute kendaraan ke-2 : 0-66-62-75-60-37-43-5-44-54-59-50-49-58-48-16-52-27-42-6-35-56-0

Rute kendaraan ke-3 : 0-51-39-32-10-19-33-36-18-24-14-96-73-95-15-68-69-38-25-72-12-7-71-80-41-47-0

Rute kendaraan ke-4 : 0-45-17-31-23-86-0

17. Bunga 50, iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 2574. Rute :

Rute kendaraan ke-1 : 0-63-34-25-20-22-32-30-39-37-38-24-26-55-65-7-92-33-52-41-40-15-28-72-0

Rute kendaraan ke-2 : 0-17-49-51-35-3-67-31-36-44-47-46-50-16-86-76-61-83-12-9-23-6-10-84-89-1-0

Rute kendaraan ke-3 : 0-45-27-62-90-99-21-77-96-14-18-69-74-58-60-48-87-79-71-85-29-19-43-13-2-57-91-93-82-8-59-64-68-42-11-0

Rute kendaraan ke-4 : 0-54-53-56-75-97-94-88-80-78-95-70-81-98-73-5-4-66-100-0

18. Bunga 50, iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 2545. Rute :

Rute kendaraan ke-1 : 0-7-98-66-21-97-90-43-92-86-84-94-100-93-23-45-27-30-75-74-33-36-28-6-39-37-48-50-32-9-44-2-47-69-89-10-20-78-87-3-0

Rute kendaraan ke-2 : 0-26-59-61-40-5-72-1-8-4-29-22-73-68-76-85-62-25-31-51-46-58-53-0

Rute kendaraan ke-3 : 0-49-70-55-52-12-60-42-11-83-82-35-38-54-63-91-80-79-77-81-88-71-67-65-64-96-24-13-41-16-14-99-95-57-17-56-0

Rute kendaraan ke-4 : 0-34-18-15-19-0

19. Bunga 100, iterasi 10, *switch probability* = 0.25, proses FPA dengan jarak 2950. Rute :

Rute kendaraan ke-1 : 0-96-83-26-11-89-1-37-36-28-35-9-33-58-4-6-40-52-18-75-22-21-44-55-69-0

Rute kendaraan ke-2 : 0-14-12-7-31-90-72-51-42-32-59-19-25-53-81-93-74-87-68-29-41-43-8-48-65-88-3-78-99-46-39-38-70-0

Rute kendaraan ke-3 : 0-24-67-61-84-76-73-82-30-71-86-98-95-45-13-10-63-94-77-80-15-5-66-79-92-49-54-17-57-62-100-34-60-64-56-91-2-0

Rute kendaraan ke-4 : 0-47-50-23-16-27-20-97-85-0

20. Bunga 100, iterasi 10, *switch probability* = 0.5, proses FPA dengan jarak 2803. Rute :

Rute kendaraan ke-1 : 0-22-13-32-9-47-60-8-67-99-90-98-61-79-97-5-93-23-75-1-19-6-37-38-11-36-77-50-78-31-25-85-94-76-65-49-52-0

Rute kendaraan ke-2 : 0-15-34-24-10-39-33-46-17-18-57-56-62-100-30-16-58-51-20-91-64-55-81-72-88-3-28-40-0

Rute kendaraan ke-3 : 0-7-48-74-63-53-29-26-35-84-86-80-4-66-68-45-21-89-95-87-59-43-14-70-73-92-2-54-44-27-83-69-0

Rute kendaraan ke-4 : 0-12-42-41-71-82-96-0

21. Bunga 100, iterasi 10, *switch probability* = 0.8, proses FPA dengan jarak 2922. Rute :

Rute kendaraan ke-1 : 0-1-93-71-76-87-84-91-85-42-16-27-97-96-78-53-9-4-72-64-70-82-83-81-67-99-50-68-10-75-47-12-51-77-94-55-38-19-31-100-86-62-39-3-0Rute kendaraan ke-2 : 0-46-65-61-92-33-34-26-18-25-6-15-56-44-43-98-41-5-57-40-36-2-28-80-63-95-69-0

Rute kendaraan ke-3 : 0-54-11-30-13-59-49-60-17-37-14-20-73-35-90-7-22-29-52-66-21-45-48-0

Rute kendaraan ke-4 : 0-58-89-23-32-88-79-8-24-74-0

22. Bunga 100, iterasi 100, *switch probability* = 0.25, proses FPA dengan jarak 2681. Rute :

Rute kendaraan ke-1 : 0-39-98-88-14-48-2-86-57-97-46-87-95-51-32-15-16-6-25-84-94-83-53-9-37-42-44-28-31-74-12-7-93-5-49-61-29-58-0

Rute kendaraan ke-2 : 0-34-19-26-38-33-52-85-79-47-56-54-3-72-23-36-50-17-18-30-24-65-64-75-0

Rute kendaraan ke-3 : 0-70-80-71-68-91-66-77-20-92-96-89-90-78-67-63-81-82-22-60-10-55-99-13-62-11-43-21-27-35-8-45-1-4-69-0

Rute kendaraan ke-4 : 0-76-73-41-100-40-59-0

23. Bunga 100, iterasi 100, *switch probability* = 0.5, proses FPA dengan jarak 2681. Rute :

Rute kendaraan ke-1 : 0-22-32-17-75-10-54-76-74-24-23-50-30-43-39-14-20-31-33-26-37-35-34-0

Rute kendaraan ke-2 : 0-67-73-92-47-15-28-16-18-94-1-81-3-72-66-87-52-78-82-79-45-13-65-27-11-53-90-88-41-93-36-69-25-5-49-55-9-0

Rute kendaraan ke-3 : 0-59-84-83-77-86-51-95-40-85-97-63-6-61-46-29-91-64-62-7-99-89-96-58-68-4-48-19-44-42-21-2-98-70-71-80-12-38-0

Rute kendaraan ke-4 : 0-60-100-8-57-56-0

24. Bunga 100, iterasi 100, *switch probability* = 0.8, proses FPA dengan jarak 2591. Rute :

Rute kendaraan ke-1 : 0-100-34-23-30-33-16-32-67-97-73-64-22-60-7-92-70-56-24-98-89-69-99-61-14-28-36-1-93-53-2-95-76-12-79-83-82-80-85-91-90-96-74-88-86-84-59-21-25-0

Rute kendaraan ke-2 : 0-49-62-71-63-11-72-58-94-87-75-65-66-51-31-4-9-42-78-46-18-26-39-45-17-57-0

Rute kendaraan ke-3 : 0-44-40-47-55-48-52-37-41-5-6-81-77-3-54-8-10-50-13-29-38-43-27-19-0

Rute kendaraan ke-4 : 0-68-20-35-15-0

25. Bunga 100, iterasi 1000, *switch probability* = 0.25, proses FPA dengan jarak 2483. Rute :

Rute kendaraan ke-1 : 0-19-15-50-52-63-48-91-92-74-79-96-100-76-58-99-5-7-85-34-31-40-37-23-88-80-43-93-59-46-28-38-36-33-18-42-0

Rute kendaraan ke-2 : 0-51-65-54-10-72-61-6-32-8-68-29-26-22-2-75-57-98-14-13-66-44-47-12-0

Rute kendaraan ke-3 : 0-4-17-35-9-25-21-16-30-67-69-87-86-3-39-11-97-55-89-53-1-62-95-71-94-84-77-70-78-81-73-90-60-41-27-56-24-0

Rute kendaraan ke-4 : 0-49-45-20-64-83-82-0

26. Bunga 100, iterasi 1000, *switch probability* = 0.5, proses FPA dengan jarak 2466. Rute :

Rute kendaraan ke-1 : 0-17-41-9-20-22-98-78-81-80-64-65-48-5-72-70-76-66-57-82-92-3-38-94-6-32-63-74-29-15-53-40-31-89-0

Rute kendaraan ke-2 : 0-56-30-43-16-34-99-100-44-10-42-62-7-88-18-77-61-68-50-47-67-28-59-85-73-86-24-54-58-75-0

Rute kendaraan ke-3 : 0-45-52-35-36-27-12-8-83-84-96-60-25-26-39-37-33-19-14-13-93-46-21-51-11-95-2-49-23-0

Rute kendaraan ke-4 : 0-55-4-90-97-91-79-71-87-69-1-0

27. Bunga 100, iterasi 1000, *switch probability* = 0.8, proses FPA dengan jarak 2432. Rute :

Rute kendaraan ke-1 : 0-66-64-62-7-97-9-43-37-39-17-50-31-32-18-84-13-10-12-53-38-25-44-47-21-60-0

Rute kendaraan ke-2 : 0-3-78-33-26-77-4-100-94-88-93-69-70-71-55-40-72-15-24-59-56-57-85-99-80-73-28-35-41-5-82-79-76-75-87-90-65-42-68-22-0

Rute kendaraan ke-3 : 0-27-23-36-46-89-74-61-96-98-52-20-6-67-45-30-29-51-49-83-81-86-95-34-19-48-16-14-54-63-58-0

Rute kendaraan ke-4 : 0-1-91-92-8-11-2-0

Lampiran 13 : Tampilan Antar Muka Program

```
run:
-------------------------------------------------------------------------------------------------------------------------------------------------------
|            Penerapan Flower Pollination Algorithm untuk Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and Delivery                 |
|                                                        Moh.Rizki Kurniawan                                                                          |
|                                                        Universitas Airlangga                                                                        |
-------------------------------------------------------------------------------------------------------------------------------------------------------

      Pilihan data :
      1. Data kecil
      2. Data sedang
      3. Data besar

      Pilihan data :  1

      ------------------------------
               Data jarak
      ------------------------------

      0.0     6.0     8.0     12.0    17.0    15.0    15.0    16.0    10.0    10.0    10.0    5.0     6.0     2.0
      6.0     0.0     4.0     12.0    17.0    16.0    17.0    20.0    14.0    14.0    12.0    8.0     11.0    8.0
      8.0     4.0     0.0     8.0     13.0    12.0    14.0    18.0    14.0    12.0    9.0     6.0     11.0    10.0
      12.0    12.0    8.0     0.0     6.0     4.0     7.0     12.0    10.0    8.0     4.0     7.0     10.0    12.0
      17.0    17.0    13.0    6.0     0.0     4.0     9.0     15.0    15.0    12.0    9.0     12.0    15.0    18.0
      15.0    16.0    12.0    4.0     4.0     0.0     5.0     11.0    11.0    8.0     6.0     10.0    12.0    15.0
      15.0    17.0    14.0    7.0     9.0     5.0     0.0     6.0     8.0     5.0     5.0     10.0    10.0    15.0
      16.0    20.0    18.0    12.0    15.0    11.0    6.0     0.0     6.0     6.0     9.0     12.0    10.0    15.0
      10.0    14.0    14.0    10.0    15.0    11.0    8.0     6.0     0.0     3.0     6.0     7.0     4.0     9.0
      10.0    14.0    12.0    8.0     12.0    8.0     5.0     6.0     3.0     0.0     4.0     7.0     5.0     10.0
      10.0    12.0    9.0     4.0     9.0     6.0     5.0     9.0     6.0     4.0     0.0     5.0     6.0     10.0
      5.0     8.0     6.0     7.0     12.0    10.0    10.0    12.0    7.0     7.0     5.0     0.0     5.0     6.0
      6.0     11.0    11.0    10.0    15.0    12.0    10.0    10.0    4.0     5.0     6.0     5.0     0.0     5.0
      2.0     8.0     10.0    12.0    18.0    15.0    15.0    15.0    9.0     10.0    10.0    6.0     5.0     0.0
```

```
   Iterasi ke-6
----------------------
        Solusi terbaik sementara pada iterasi ke-6 : 145.0 pada bunga ke-84

----------------------
   Iterasi ke-7
----------------------
        Solusi terbaik sementara pada iterasi ke-7 : 143.0 pada bunga ke-91

----------------------
   Iterasi ke-8
----------------------
        Solusi terbaik sementara pada iterasi ke-8 : 143.0 pada bunga ke-91

----------------------
   Iterasi ke-9
----------------------
        Solusi terbaik sementara pada iterasi ke-9 : 143.0 pada bunga ke-4

----------------------
   Iterasi ke-10
----------------------
        Solusi terbaik pada iterasi ke-10 : 143.0 pada bunga ke-4

dengan rute sebagai berikut :

        Rute kendaraan ke-1 :   0-13-3-4-0
        Rute kendaraan ke-2 :   0-5-2-1-0
        Rute kendaraan ke-3 :   0-10-9-6-7-0
        Rute kendaraan ke-4 :   0-8-11-12-0
BUILD SUCCESSFUL (total time: 22 seconds)
```

Lampiran 14 : Hasil *Running* Program pada Data Dengan 13 Pelanggan

Jumlah bunga	Maksimum Iterasi	Presentase Kapasitas (%)		
		50	60	70
10	10	139	127	124
	100	128	122	113
	1000	117	110	103
50	`10	127	123	113
	100	126	113	105
	1000	120	110	102
100	10	132	123	115
	100	121	110	107
	1000	117	107	99

Lampiran 15 : Hasil *Running* Program pada Data Dengan 22 Pelanggan

Jumlah bunga	Maksimum Iterasi	Presentase Kapasitas (%)		
		50	60	70
10	10	181	177	159
	100	175	168	156
	1000	163	149	140
50	`10	184	173	164
	100	172	167	154
	1000	154	142	132
100	10	171	163	163
	100	162	160	149
	1000	152	138	131

Lampiran 16 : Hasil *Running* Program pada Data Dengan 100 Pelanggan

Jumlah bunga	Maksimum Iterasi	Presentase Kapasitas (%)		
		50	60	70
10	10	3137	3077	3066
	100	2953	2848	2840
	1000	2695	2695	2641
50	`10	3108	3103	3057
	100	2596	2772	2723
	1000	2621	2607	2605
100	10	3057	2956	2950
	100	2811	2777	2681
	1000	2543	2540	2483