

**IMPLEMENTASI *HONEY BADGER ALGORITHM* (HBA)
DALAM MENYELESAIKAN *VEHICLE ROUTING PROBLEM*
WITH SIMULTANEOUS PICKUP AND DELIVERY (VRPSPD)**

SKRIPSI



REZA ABID ABISTHA

PROGRAM STUDI S-1 MATEMATIKA

DEPARTEMEN MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

2023

**IMPLEMENTASI *HONEY BADGER ALGORITHM* (HBA)
DALAM MENYELESAIKAN *VEHICLE ROUTING PROBLEM*
WITH SIMULTANEOUS PICKUP AND DELIVERY (VRPSPD)**

SKRIPSI



REZA ABID ABISTHA

PROGRAM STUDI S-1 MATEMATIKA

DEPARTEMEN MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

2023

**IMPLEMENTASI *HONEY BADGER ALGORITHM* (HBA) DALAM
MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH
SIMULTANEOUS PICKUP AND DELIVERY* (VRPSD)**

SKRIPSI

**Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Sains Bidang
Matematika pada Fakultas Sains dan Teknologi
Universitas Airlangga**

**Reza Abid Abistha
NIM 081911233079**

Disetujui oleh:

Pembimbing I,



**Dr. Henry Suprajitno, S.Si., M.Si.
NIP. 196804041994031020**

Pembimbing II,



**Asri Bakti Pratiwi, S.Si., M.Si.
NIP. 198312222012122001**

LEMBAR PENGESAHAN NASKAH SKRIPSI

Judul : Implementasi *Honey Badger Algorithm* (HBA) dalam
Menyelesaikan *Vehicle Routing Problem with*
Simultaneous Pickup and Delivery (VRPSPD)
Penyusun : Reza Abid Abistha
NIM : 081911233079
Tanggal Ujian : 17 April 2023

Disetujui oleh:

Pembimbing I,



Dr. Herry Suprajitno, S.Si., M.Si.
NIP. 196804041994031020

Pembimbing II,



Asri Bakti Pratiwi, S.Si., M.Si.
NIP. 198312222012122001

Mengetahui,

Ketua Departemen Matematika
Fakultas Sains dan Teknologi
Universitas Airlangga



Dr. Herry Suprajitno, S.Si., M.Si.
NIP. 196804041994031020

PEDOMAN PENGGUNAAN SKRIPSI

Skripsi ini tidak dipublikasikan, namun tersedia di perpustakaan dalam lingkungan Universitas Airlangga, diperkenankan untuk dipakai sebagai referensi kepustakaan, tetapi pengutipan harus seizin penyusun dan harus menyebutkan sumbernya sesuai kebiasaan ilmiah. Dokumen skripsi ini merupakan hak milik Universitas Airlangga.

SURAT PERNYATAAN TENTANG ORISINALITAS

Yang bertanda tangan di bawah ini, saya:

Nama : Reza Abid Abistha

NIM : 081911233079

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Jenjang : Sarjana (S1)

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

“Implementasi *Honey Badger Algorithm* (HBA) dalam Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)”

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

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, 26 Maret 2023



Reza Abid Abistha

NIM. 081911233079

KATA PENGANTAR



Puji syukur kehadiran Allah SWT berkat rahmat, hidayah, dan karunia-Nya kepada penulis sehingga dapat menyelesaikan skripsi dengan judul “Implementasi *Honey Badger Algorithm* (HBA) dalam Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)”.

Penulis menyadari dalam penyusunan skripsi ini tidak akan selesai tanpa bantuan dari berbagai pihak. Karena itu pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Prof. Dr. Moh. Nasih, S.E., M.T., Ak. selaku rektor Universitas Airlangga yang memberikan kesempatan dan fasilitas kepada penulis untuk menimba ilmu serta pengalaman.
2. Prof. Dr. Moh. Yasin, M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Airlangga yang memberikan kesempatan dan fasilitas kepada penulis untuk menimba ilmu serta pengalaman.
3. Dr. Herry Suprajitno, S.Si., M.Si. selaku Ketua Departemen Matematika Universitas Airlangga, Koordinator Prodi S-1 Matematika Fakultas Sains dan Teknologi Universitas Airlangga, dan dosen pembimbing I yang selalu memberikan bimbingan, arahan, serta motivasi kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan lancar.
4. Asri Bkti Pratiwi, S.Si., M.Si. selaku dosen pembimbing II yang telah memberikan arahan, masukan, bimbingan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan lancar.
5. Muhammadun, S.Si., M.Si. selaku dosen wali yang telah memberikan saran, motivasi kepada penulis selama masa 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. Yang tercinta kedua orang tua penulis, yaitu Bapak Sailan dan Ibu Heni Sovieti yang tiada henti-hentinya selalu mendoakan, mendukung, memberikan motivasi kepada penulis.
8. Teman-teman penulis Geometri 2019 yang selalu memberikan bantuan dan semangat kepada penulis selama masa perkuliahan.
9. Teman-teman dari HIMATIKA UNAIR yang telah memberikan kesempatan kepada penulis untuk berkontribusi dan menjadi wadah untuk mengembangkan diri.
10. Semua pihak yang tidak dapat disebutkan, yang telah membantu terselesaikannya skripsi ini.

Penulis juga menyadari skripsi ini tidak luput dari berbagai kekurangan. Penulis mengharapkan saran dan kritik demi kesempurnaan dan perbaikannya sehingga akhirnya skripsi ini dapat memberikan manfaat bagi bahan pustaka dan penambah informasi khususnya bagi mahasiswa Universitas Airlangga.

Surabaya, 26 Maret 2023

Penyusun,

Reza Abid Abistha

Reza Abid Abistha, 2023. **Implementasi *Honey Badger Algorithm* dalam Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery***. Skripsi ini dibawah bimbingan Dr. Herry Suprajitno, S.Si., M.Si. dan Asri Bakti Pratiwi, S.Si., M.Si. Departemen Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga, Surabaya.

ABSTRAK

Vehicle Routing Problem (VRP) didefinisikan sebagai permasalahan optimasi dalam menentukan rute yang optimal bagi armada kendaraan dengan biaya transportasi yang minimal. Salah satu varian dari VRP adalah *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) yang menambahkan batasan berupa permintaan, yaitu pengambilan dan pengiriman barang, untuk setiap pelanggan dalam satu kunjungan harus dipenuhi. Penyelesaian VRPSPD pada skripsi ini menggunakan *Honey Badger Algorithm* (HBA) yang merupakan algoritma metaheuristik dengan menerapkan perilaku musang madu dalam mencari makanannya, yaitu pada fase menggali dan fase madu. Proses algoritma ini diawali dengan membangkitkan posisi awal musang madu, mengevaluasi fungsi tujuan, menentukan mangsa, menentukan faktor kepadatan, menentukan intensitas, menentukan posisi baru musang madu menggunakan fase menggali atau fase madu, mengevaluasi fungsi tujuan baru, memperbarui posisi dan fungsi tujuan musang madu, memperbarui posisi dan fungsi tujuan mangsa, kemudian mengulangi proses pembaruan posisi musang madu untuk semua musang madu dalam populasi hingga diperoleh solusi terbaik dan iterasi maksimum terpenuhi. Program penyelesaian dibuat menggunakan bahasa pemrograman C++ dengan bantuan *software Visual Studio Code 1.76* yang diimplementasikan pada tiga macam ukuran data, yaitu data kecil dengan 13 pelanggan, data sedang dengan 75 pelanggan, dan data besar dengan 100 pelanggan. Berdasarkan hasil yang diperoleh, dapat disimpulkan bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi, maka hasil yang diperoleh semakin baik. Sementara semakin kecil parameter C , salah satu parameter konstan pada HBA, maka hasil yang diperoleh cenderung semakin baik dengan total jarak tempuh lebih kecil.

Kata Kunci: *Honey Badger Algorithm, Vehicle Routing Problem, Simultaneous Pickup and Delivery*

Reza Abid Abistha, 2023. **Implementation of Honey Badger Algorithm in Solving Vehicle Routing Problem with Simultaneous Pickup and Delivery**. This undergraduate thesis was supervised by Dr. Herry Suprajitno, S.Si., M.Si. and Asri Bakti Pratiwi, S.Si., M.Si. Department of Mathematics, Faculty of Science and Technology, Airlangga University, Surabaya.

ABSTRACT

Vehicle Routing Problem (VRP) is defined as an optimization problem in determining the optimal route for a fleet of vehicles with minimal transportation costs. One of the variants of VRP is the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) which adds a limit in the form of requests, namely the collection and delivery of goods, for each customer in one visit must be fulfilled. Completion of VRPSPD in this final project uses the Honey Badger Algorithm (HBA) which is a metaheuristic algorithm by applying the behavior of honey badgers in foraging, namely in the digging phase and the honey phase. The process of this algorithm begins by generating the initial position of the honey badger, evaluating the objective function, determining the prey, determining the density factor, determining the intensity, determining the new position of the honey badger using the digging or honey phase, evaluating the new objective function, updating the position and objective function of the honey badger, update the position and objective function of the prey, then repeat the process of updating the position for the entire honey badger population until the best solution and maximum iterations are met. The settlement program was made using the C++ programming language with the help of Visual Studio Code 1.76 software which was implemented on three different data sizes, namely small data with 13 customers, medium data with 75 customers, and large data with 100 customers. Based on the results obtained, it can be concluded that the greater the maximum iteration limit and the greater the honeycomb population, the better the results obtained. Meanwhile, the smaller the parameter C , one of the constant parameters on the HBA, the better the results obtained with a smaller total mileage.

Keywords: *Honey Badger Algorithm, Vehicle Routing Problem, Simultaneous Pickup and Delivery*

DAFTAR ISI

LEMBAR JUDUL	i
LEMBAR PERNYATAAN	ii
LEMBAR PENGESAHAN NASKAH SKRIPSI	iii
PEDOMAN PENGGUNAAN SKRIPSI	iv
SURAT PERNYATAAN TENTANG ORISINALITAS	v
KATA PENGANTAR	vi
ABSTRAK	viii
ABSTRACT	ix
DAFTAR ISI	x
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xiv
DAFTAR LAMPIRAN.....	xv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan.....	3
1.4 Manfaat.....	4
BAB II TINJAUAN PUSTAKA	5
2.1 <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i>	5
2.2 <i>Honey Badger Algorithm (HBA)</i>	8
BAB III METODE PENELITIAN	11
BAB IV PEMBAHASAN	15
4.1 Prosedur <i>Honey Badger Algorithm (HBA)</i> dalam Menyelesaikan <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i> ..	15
4.1.1 Input Data <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i>	16

4.1.2	Inisialisasi Parameter <i>Honey Badger Algorithm</i> (HBA) dan <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery</i> (VRPSPD).....	17
4.1.3	Membangkitkan Posisi Awal Musang Madu.....	18
4.1.4	Menentukan Musang Madu Terbaik/Mangsa (<i>xprey</i>) Sementara..	19
4.1.5	Menentukan Faktor Kepadatan	20
4.1.6	Menentukan Intensitas	21
4.1.7	Menentukan Posisi Baru Musang Madu Menggunakan Fase Menggali	22
4.1.8	Menentukan Posisi Baru Musang Madu Menggunakan Fase Madu.....	23
4.1.9	Memperbarui Posisi dan Fungsi Tujuan Musang Madu.....	24
4.1.10	Memperbarui Posisi dan Fungsi Tujuan Mangsa.....	25
4.2	Data	26
4.3	Penyelesaian Secara Manual Implementasi <i>Honey Badger Algorithm</i> (HBA) pada Contoh Kasus <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery</i> (VRPSPD) Menggunakan Data Kecil	27
4.3.1	Input Data dan Inisialisasi Parameter	27
4.3.2	Membangkitkan Posisi Awal Musang Madu.....	30
4.3.3	Mengevaluasi Posisi Awal Musang Madu.....	Error! Bookmark not defined.
4.3.4	Menentukan Musang Madu Terbaik Sementara/Mangsa (<i>xprey</i>)..	34
4.3.5	Mengecek Jumlah Maksimum Iterasi.....	34
4.3.6	Menentukan Faktor Kepadatan	34
4.3.7	Menentukan Intensitas	35
4.3.8	Menentukan Posisi Baru Musang Madu.....	37

4.3.9	Mengevaluasi Posisi Musang Madu Baru (<i>xnew</i>).....	39
4.3.10	Memperbarui Posisi dan Fungsi Tujuan Musang Madu.....	40
4.3.11	Memperbarui Posisi dan Fungsi Tujuan Mangsa.....	41
4.3.12	Mengecek Jumlah Maksimum Iterasi.....	42
4.3.13	Menampilkan Solusi Terbaik.....	42
4.4	Implementasi Program pada <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery</i> (VRPSPD).....	43
4.4.1	Implementasi pada Data Berukuran Kecil.....	43
4.4.2	Implementasi pada Data Berukuran Sedang.....	44
4.4.3	Implementasi pada Data Berukuran Besar	46
BAB V PENUTUP		48
5.1	Kesimpulan	48
5.2	Saran.....	49
DAFTAR PUSTAKA		51

DAFTAR TABEL

Nomor	Judul Tabel	Halaman
4.1	Data Jarak untuk Depot dan 13 Pelanggan	28
4.2	Data <i>Delivery</i> dan <i>Pickup</i> untuk 13 Pelanggan	29
4.3	Hasil Bangkitan Bilangan <i>Real</i> Acak pada Interval (0, 1)	30
4.4	Calon Rute untuk Setiap Musang Madu	31
4.5	Pembentukan Rute Awal untuk setiap Musang Madu	33
4.6	Intensitas untuk Setiap Musang Madu	37
4.7	Posisi Baru untuk Setiap Musang Madu	38
4.8	Calon Rute untuk Setiap Musang Madu yang Baru	39
4.9	Pembentukan Rute Baru untuk setiap Musang Madu	39
4.10	Pembaruan Posisi dan Fungsi Tujuan untuk Setiap Musang Madu	40
4.11	Nilai Fungsi Tujuan Saat Ini untuk Setiap Musang Madu	41
4.12	Pembaruan Posisi dan Fungsi Tujuan Mangsa	42
4.13	Rute yang Terbentuk pada Musang Madu Terbaik/Mangsa	42
4.14	Hasil <i>Running</i> Program pada Data Kecil	44
4.15	Rute Terbaik yang Terbentuk pada Implementasi Data Kecil	44
4.16	Hasil <i>Running</i> Program pada Data Sedang	45
4.17	Rute Terbaik yang Terbentuk pada Implementasi Data Sedang	45
4.18	Hasil <i>Running</i> Program pada Data Besar	46
4.19	Rute Terbaik yang Terbentuk pada Implementasi Data Besar	47

DAFTAR GAMBAR

Nomor	Judul Gambar	Halaman
4.1	Prosedur HBA untuk Menyelesaikan VRPSPD	16
4.2	Prosedur Input Data	17
4.3	Prosedur Inisialisasi Parameter	18
4.4	Prosedur Membangkitkan Posisi Awal Musang Madu	19
4.5	Prosedur menentukan musang madu terbaik (<i>xprey</i>) sementara	20
4.6	Prosedur Menentukan Faktor Kepadatan	20
4.7	Prosedur Menentukan Intensitas	21
4.8	Prosedur Penentuan Posisi Baru Musang Madu pada Fase Menggali	22
4.9	Prosedur Penentuan Posisi Baru Musang Madu pada Fase Madu	24
4.10	Prosedur Pembaruan Posisi dan Fungsi Tujuan Musang Madu	25
4.11	Prosedur Pembaruan Posisi dan Fungsi Tujuan Mangsa	25

DAFTAR LAMPIRAN

Nomor	Judul Lampiran
1	<i>Flowchart Implementasi Honey Badger Algorithm (HBA) dalam Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i>
2	Prosedur Mengevaluasi Posisi Awal Musang Madu
3	Prosedur Mengevaluasi Posisi Musang Madu Baru
4	Data Kecil <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i> dengan 13 Pelanggan
5	Data Sedang <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i> dengan 75 Pelanggan
6	Data Besar <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i> dengan 100 Pelanggan
7	Data Jarak untuk Depot dan 13 Pelanggan
8	Hasil Bangkitan Bilangan <i>Real</i> Acak pada Interval (0, 1)
9	Hasil Perhitungan Nilai Intensitas Musang Madu dan Hasil Perhitungan Posisi Baru Musang Madu
10	<i>Source Code</i> Program Implementasi <i>Honey Badger Algorithm (HBA)</i> dalam Menyelesaikan <i>Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)</i>
11	Hasil <i>Running</i> Program Berdasarkan Persentase Muatan Awal pada Kendaraan

BAB I PENDAHULUAN

1.1 Latar Belakang

Transportasi merupakan bagian yang melekat serta memainkan peran yang sangat penting bagi sektor industri. Ketika sistem transportasi efisien, maka akan memberikan peluang dan manfaat yang berdampak positif seperti akses yang lebih baik dalam pendistribusian maupun investasi tambahan bagi industri tersebut **(Stanković, 2021)**. Adanya perencanaan yang baik terhadap penentuan rute distribusi barang atau proses logistik kepada pelanggan, diharapkan dapat meminimalkan jarak tempuh dan juga biaya perjalanan **(Cetin dan Gencer, 2015)**.

Permasalahan distribusi diperkenalkan oleh **Dantzig dan Ramser (1959)** dengan nama *Truck Dispatching Problem*. Konsep permasalahan tersebut adalah untuk menentukan rute truk yang optimal dalam pengiriman bensin dari depot ke berbagai stasiun pelayanan yang dipasok oleh depot tersebut, sehingga permintaan dari setiap stasiun pelayanan dapat terpenuhi dan jarak tempuh yang dilalui oleh truk tersebut minimum. Adanya perkembangan ilmu pengetahuan dan teknologi, menimbulkan perubahan yang lumayan signifikan dalam konsep permasalahan *Truck Dispatching Problem* guna memperluas jangkauan kompleksitas permasalahan nyata seperti pada ranah logistik. Permasalahan untuk melayani sekumpulan pelanggan, yang tersebar secara geografis di sekitar depot pusat, menggunakan armada kendaraan dengan kapasitas yang bervariasi dikenal sebagai *Vehicle Routing Problem (VRP)* **(Jaegere dkk, 2014)**.

Vehicle Routing Problem (VRP) memiliki tujuan yaitu untuk menentukan rute distribusi yang terbaik sedemikian sehingga total jarak tempuh yang dilalui oleh kendaraan maupun biaya pendistribusian dapat dilakukan secara optimal. Pada rute distribusi yang ditentukan juga memerhatikan pelanggan yang dikunjungi tidak lebih dari sekali oleh satu kendaraan, dan permintaan total dari seluruh pelanggan dalam satu rute tidak melebihi kapasitas kendaraan pada rute tersebut **(Belfiore dkk, 2008)**.

Dalam penerapannya, kasus *Vehicle Routing Problem* (VRP) banyak ditemui dalam kehidupan masyarakat luas. Salah satu contoh penerapan yang terjadi dalam dunia industri adalah pada budaya masyarakat Amerika Serikat yang menaruh botol susu kosong di depan pintu, yang kemudian akan diambil oleh seorang *milkman* yang akan berkeliling sesuai dengan rute yang telah ditentukan untuk mengganti botol kosong tersebut dengan botol yang berisi susu. Contoh penerapan VRP lainnya di Indonesia antara lain dalam pendistribusian *Liquefied Petroleum Gas* (LPG) maupun galon air mineral. Contoh-contoh tersebut merupakan salah satu bentuk varian dari VRP yang dinamakan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD). VRPSPD merupakan salah satu jenis dari VRP yang tidak hanya melakukan proses pengiriman barang dalam memenuhi permintaan pelanggan, akan tetapi juga memerhatikan proses pengambilan barang dalam waktu yang bersamaan oleh kendaraan yang sama, sehingga muatan barang dalam kendaraan harus diperhatikan agar tidak melebihi kapasitas kendaraan (Catay, 2010).

Berbagai penelitian terhadap VRPSPD telah banyak dilakukan dengan menggunakan beberapa metode penyelesaian atau algoritma yang ada. Beberapa diantaranya adalah menggunakan *Ant Colony System* (Gajpal dan Abad, 2009), *Genetic Algorithm* (Tasan dan Gen, 2012), dan *Particle Swarm Optimization* (Ai dan Kachitvichyanukul, 2009).

Honey Badger Algorithm (HBA) merupakan algoritma yang terinspirasi dari perilaku pencarian makanan dari seekor musang madu. Terdapat dua macam cara musang madu dalam mencari makanan, yaitu dengan menggunakan penciuman serta kemampuan menggali yang disebut sebagai fase penggalian, dan mengikuti *honeyguide bird* atau burung pemandu madu yang disebut sebagai fase madu (Hashim dkk, 2022). Berdasarkan penelitian yang dilakukan oleh Hashim dkk (2022), HBA telah diterapkan ke beberapa permasalahan nyata di bidang teknik seperti *Tension/Compression Spring Design Problem* (Dhiman dan Kumar, 2017) dan *Speed Reducer Design Problem* (Mezura-Montes dan Coello, 2005). HBA juga dapat bersaing dengan beberapa algoritma seperti *Simulated Annealing* (SA), *Particle Swarm Optimization* (PSO), dan *Whale Optimization*

Algorithm (WOA) dalam kecepatannya mencapai tingkat kekonvergensi maupun mencegah solusi terjebak di area lokal, yaitu dengan menyuguhkan penggunaan penanda F yang mengarahkan pencarian solusi menuju area baru, serta fase penggalian dan fase madu yang mempertahankan dampak konstruktif pada proses pembaruan solusi dengan menyeimbangkan sifat eksplorasi dan eksploitasi pada mekanisme pencarian (Hashim dkk, 2022). Berdasarkan uraian tersebut, diharapkan dengan mengimplementasikan HBA ke permasalahan perutean seperti VRPSPD, mampu menentukan hasil yang optimal dengan waktu yang relatif singkat seperti pada studi yang telah ada.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, didapatkan rumusan permasalahan sebagai berikut:

1. Bagaimana penerapan *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)?
2. Bagaimana membuat program implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)?
3. Bagaimana implementasi program *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) pada contoh kasus?

1.3 Tujuan

Berdasarkan rumusan masalah sebelumnya, didapatkan tujuan sebagai berikut:

1. Menerapkan *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
2. Membuat program implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

3. Mengimplementasikan program *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) pada contoh kasus.

1.4 Manfaat

Manfaat yang diharapkan dalam penelitian ini adalah sebagai berikut:

1. Menambah wawasan mahasiswa Menambah wawasan mahasiswa di bidang matematika khususnya pada pengimplementasian *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
2. Sebagai referensi alternatif metode penyelesaian yang dapat digunakan untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).
3. Sebagai pembanding dan bahan masukan pada pengimplementasian algoritma lainnya dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

BAB II

TINJAUAN PUSTAKA

Pada bab ini dijelaskan definisi serta informasi yang digunakan pada pembahasan implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD).

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

Vehicle Routing Problem (VRP) didefinisikan sebagai armada kendaraan yang berawal di depot dan harus mengirimkan sejumlah barang kepada sekumpulan pelanggan. Menentukan rute optimal yang digunakan oleh kelompok kendaraan saat melayani sekumpulan pelanggan merepresentasikan permasalahan VRP yang bertujuan untuk meminimalkan biaya transportasi secara keseluruhan. Solusi dari masalah VRP adalah himpunan rute yang semuanya dimulai dan diakhiri di depot, dan yang memenuhi batasan bahwa semua pelanggan hanya dilayani tepat satu kali. Biaya transportasi dapat diminimalkan dengan mengurangi total jarak tempuh dan dengan mengurangi jumlah kendaraan yang dibutuhkan (**Caric dan Gold, 2008**).

Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) merupakan salah satu varian dari VRP. VRPSPD pertama kali diperkenalkan oleh **Min (1989)** yang secara umum digambarkan sebagai permasalahan pendistribusian barang dengan melakukan pengiriman dan pengambilan barang secara bersamaan pada sekumpulan pelanggan yang ada. Situasi yang demikian sering terjadi dalam menentukan atau merancang rute yang optimal untuk sistem angkutan umum, maskapai penerbangan, bus sekolah, distributor produk bahan makanan, dan lain sebagainya (**Min, 1989**). VRPSPD bertujuan untuk membangun satu set rute kendaraan sedemikian sehingga (i) setiap pelanggan dikunjungi tepat satu kendaraan; (ii) setiap kendaraan melakukan pengiriman paling banyak pada satu rute; (iii) setiap rute kendaraan dimulai dan berakhir di depot; (iv) permintaan pengambilan dan pengiriman setiap pelanggan dalam satu kunjungan harus dipenuhi; (v) kapasitas kendaraan tidak melebihi total

permintaan pada rute kendaraan; dan (vi) total biaya perjalanan diminimalkan (**Koç dkk, 2020**).

Model matematika dari VRPSPD diberikan sebagai berikut:

$$\min Z = \sum_{i \in J_0} \sum_{j \in J_0} \sum_{k \in K} c_{ij} x_{ijk} \quad (2.1)$$

dengan variabel keputusan:

$$x_{ijk} = \begin{cases} 1, & \text{Jika kendaraan } k \text{ melayani pelanggan } j \text{ setelah melayani} \\ & \text{pelanggan } i \\ 0, & \text{Jika sebaliknya} \end{cases} \quad (2.2)$$

dan batasan-batasan:

1. Setiap pelanggan dikunjungi tepat satu kali oleh satu kendaraan

$$\sum_{i \in J_0} \sum_{k \in K} x_{ijk} = 1; j \in J \quad (2.3)$$

2. Setiap kendaraan harus meninggalkan pelanggan yang telah dikunjungi

$$\sum_{i \in J_0} x_{irk} = \sum_{j \in J_0} x_{rjk}; r \in J, k \in K \quad (2.4)$$

3. Total permintaan pengambilan dari semua pelanggan yang dikunjungi oleh kendaraan k tidak melebihi beban awal kendaraan saat meninggalkan depot

$$l \geq \sum_{i \in J_0} \sum_{j \in J} D_j x_{ijk}; k \in K \quad (2.5)$$

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

$$l_j \leq l - D_j + P_j - M(1 - x_{0jk}); j \in J, k \in K \quad (2.6)$$

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

$$l_j \leq l_i - D_j + P_j - M \left(1 - \sum_{k \in K} x_{ijk} \right); i \in J, j \in J, i \neq j \quad (2.7)$$

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

$$l \leq C \quad (2.8)$$

$$l_j \leq C; j \in J \quad (2.9)$$

7. Setiap kendaraan hanya mempunyai satu rute kendaraan

$$\pi_j \leq \pi_i + 1 - n \left(1 - \sum_{k \in K} x_{ijk} \right); i \in J, j \in J, i \neq j \quad (2.10)$$

8. Urutan pelayanan pelanggan j tidak boleh bernilai negatif

$$\pi_j \geq 0; j \in J \quad (2.11)$$

Keterangan:

- i : Indeks pelanggan awal
- j : Indeks pelanggan tujuan
- r : Indeks pelanggan saat itu
- k : Indeks kendaraan
- c_{ij} : Jarak antara pelanggan i ke pelanggan j
- J : Himpunan semua pelanggan yang harus dilayani
- J_0 : Himpunan semua pelanggan termasuk depot, $J_0 = J \cup \{0\}$
- n : Banyak pelanggan, $n = |J_0|$
- D_j : Jumlah permintaan barang yang harus dikirim ke pelanggan j
- P_j : Jumlah permintaan barang yang harus diambil dari pelanggan j
- C : Kapasitas dari kendaraan
- l : Jumlah beban kendaraan saat meninggalkan depot
- l_j : Jumlah beban kendaraan setelah melayani pelanggan j
- K : Himpunan kendaraan yang tersedia
- π_j : Posisi dari pelanggan j dalam sebuah rute
- M : Nilai maksimum antara total permintaan pengiriman dan pengambilan barang dengan total jarak tempuh untuk setiap pelanggan,

$$M = \max\{\sum_{j \in J} (D_j + P_j), \sum_{i \in J_0} \sum_{j \in J_0, j \neq i} c_{ij}\}$$
- x_{ijk} : Variabel biner yang mewakili perjalanan dari pelanggan i ke pelanggan j menggunakan kendaraan k

(Dethloff, 2001)

2.2 *Honey Badger Algorithm (HBA)*

Algoritma *Honey Badger Algorithm* (HBA) merupakan algoritma metaheuristik yang diperkenalkan oleh **Hashim dkk (2022)**, dengan menerapkan prinsip dari perilaku musang madu dalam mencari sumber makanannya. Karakteristik dari perilaku pencarian makanan musang madu tersebut menjadi acuan utama dari setiap fase, yaitu perilaku musang madu yang mengandalkan kemampuan penciuman yang kuat serta penggalian yang handal untuk menemukan mangsanya di dalam tanah pada fase menggali; perilaku musang madu dalam menemukan sarang lebah dengan adanya interaksi simbiosis mutualisme terhadap burung pemandu madu pada fase madu. Setiap musang madu dalam suatu populasi secara dinamis menentukan cara untuk memperoleh sumber makanannya melalui semua fase yang dilakukan secara acak. Hal tersebut terus berulang sampai dengan kriteria untuk berhenti telah terpenuhi (**Hashim dkk, 2022**).

Secara matematis, langkah-langkah HBA adalah sebagai berikut:

Langkah 1: **Fase inisialisasi**

Inisialisai musang madu didefinisikan berdasarkan persamaan

$$x_i = lb + r_1 \times (ub - lb) \quad (2.12)$$

dengan r_1 adalah bilangan *real* acak pada interval $(0, 1)$, x_i adalah posisi musang madu ke- i yang mengacu pada kandidat solusi dalam populasi N , sedangkan lb dan ub secara berurutan merupakan batas bawah dan batas atas dari domain pencarian.

Langkah 2: **Menentukan intensitas (I)**

Intensitas berkaitan dengan kekuatan konsentrasi mangsa dan jarak antara mangsanya dengan musang madu ke- i . I_i merupakan intensitas bau mangsa dari musang madu ke- i . Jika intensitas baunya tinggi, maka pergerakan musang madu akan cepat dan berlaku sebaliknya. Penentuan intensitas yang dimaksud diberikan pada persamaan

$$I_i = r_2 \times \frac{S}{4\pi d_i^2} \quad (2.13)$$

dengan

$$S = (x_i - x_{i+1})^2 \quad (2.14)$$

adalah sumber kekuatan atau konsentrasi kekuatan musang madu,

$$d_i = x_{prey} - x_i \quad (2.15)$$

adalah jarak antara mangsa dan musang madu ke- i , dan r_2 adalah bilangan *real* acak pada interval $(0, 1)$.

Langkah 3: Memperbarui faktor kepadatan (α)

Faktor kepadatan (α) mengontrol kemampuan pengacakan untuk memastikan transisi yang mulus dari eksplorasi ke eksploitasi. Pembaruan faktor kepadatan secara menurun berdasarkan tingkatan iterasi untuk mengurangi pengacakan dilakukan menggunakan persamaan

$$\alpha = C \times \exp\left(\frac{-t}{t_{max}}\right) \quad (2.16)$$

dengan t adalah indeks iterasi saat ini, t_{max} adalah batas iterasi maksimum, dan C adalah konstanta yang bernilai ≥ 1 .

Langkah 4: Keluar dari optimal lokal

Pada langkah ini dan dua langkah berikutnya digunakan untuk keluar dari solusi yang bersifat optimum lokal. Dalam konteks ini digunakan penanda F yang mengubah arah pencarian musang madu untuk memberikan peluang yang tinggi dalam melakukan eksplorasi pada ruang pencarian yang diberikan pada persamaan berikut

$$F = \begin{cases} 1, & \text{jika } r_6 \leq 0.5 \\ -1, & \text{jika sebaliknya} \end{cases} \quad (2.17)$$

dengan r_6 adalah bilangan *real* acak pada interval $(0, 1)$.

Langkah 5: Memperbarui posisi musang madu

Proses pembaruan posisi musang madu dibagi menjadi dua fase, yaitu fase menggali dan fase madu.

Langkah 5-1: Fase menggali

Pada fase menggali, musang madu melakukan tindakan yang mirip dengan bentuk *Cardioid* yang dapat disimulasikan pada persamaan berikut

$$x_{i\ new} = x_{prey} + F \times \beta \times I_i \times x_{prey} + F \times r_3 \times \alpha \times d_i \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]| \quad (2.18)$$

dengan $x_{i\ new}$ adalah posisi musang madu ke- i yang telah dimodifikasi/diperbarui, x_{prey} adalah posisi mangsa, α adalah faktor kepadatan, β adalah kemampuan

musang madu dalam mendapatkan makanan yang bernilai ≥ 1 , intensitas bau mangsa dari musang madu ke- i yang diberikan pada persamaan (2.13), d_i adalah jarak antara mangsa dan musang madu ke- i yang diberikan pada persamaan (2.15). r_3, r_4 , dan r_5 adalah tiga bilangan *real* acak pada interval $(0, 1)$ yang berbeda antara satu dengan lainnya. Sedangkan F sebagai penanda dalam menentukan arah pencarian musang madu yang diberikan pada persamaan (2.17).

Langkah 5-2: **Fase madu**

Kasus ketika musang madu mengikuti burung pemandu madu untuk mencapai sarang lebah dapat disimulasikan pada persamaan berikut

$$x_{i\ new} = x_{prey} + F \times r_7 \times \alpha \times d_i \quad (2.19)$$

dengan $x_{i\ new}$ adalah posisi musang madu ke- i yang telah dimodifikasi/diperbarui, x_{prey} adalah posisi mangsa, α adalah faktor kepadatan, d_i adalah jarak antara mangsa dan musang madu ke- i yang diberikan pada persamaan (2.15), r_7 adalah bilangan *real* acak pada interval $(0, 1)$, dan F sebagai penanda dalam menentukan arah pencarian musang madu yang diberikan pada persamaan (2.17).

(Hashim dkk, 2022)

BAB III

METODE PENELITIAN

Tahapan metode penelitian dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan menggunakan *Honey Badger Algorithm* (HBA) adalah sebagai berikut:

1. Melakukan kajian pustaka tentang *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dan *Honey Badger Algorithm* (HBA).
2. Mengolah data koordinat posisi depot dan pelanggan serta data permintaan pengiriman barang untuk setiap pelanggan yang diambil dari <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/> pada tanggal 28 Maret 2023. Data koordinat posisi depot dan pelanggan diubah menjadi data jarak $n \times n$ dengan n adalah banyak pelanggan ditambah depot, sedangkan data permintaan pengiriman diolah menjadi data permintaan pengiriman serta pengambilan barang menggunakan teknik rasio (Nagy dan Salhi, 2005).
3. Mengimplementasikan *Honey Badger Algorithm* (HBA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan tahapan sebagai berikut:
 - a. Menginputkan nilai parameter *Honey Badger Algorithm*, antara lain banyaknya musang madu (N), konstanta (C), kemampuan musang madu mencari makanan (β), muatan awal kendaraan (%), dan batas maksimum iterasi (t_{max}).
 - b. Membangkitkan posisi awal musang madu.
 - i. Membangkitkan bilangan *real* acak pada interval $(0,1)$ sebanyak n pelanggan untuk setiap musang madu dalam populasi.
 - ii. Mengurutkan bilangan *real* acak dari nilai terkecil hingga terbesar sehingga diperoleh urutan pelanggan sebagai calon rute.

- c. Mengevaluasi fungsi tujuan untuk posisi awal musang madu.
 - i. Menentukan rute dari calon rute berdasarkan perhitungan permintaan dalam pengiriman maupun pengambilan barang untuk setiap pelanggan dan kapasitas maksimum kendaraan.
 - ii. Menentukan total jarak tempuh yang dilalui kendaraan berdasarkan rute yang telah terbentuk.
- d. Menentukan posisi dan nilai fungsi tujuan musang madu terbaik/mangsa, x_{prey} dan f_{prey} , sementara.
- e. Inisialisasi indeks iterasi saat ini, yaitu $t = 1$.
- f. Jika $t \leq t_{max}$, maka lanjut ke langkah 3 – g . Jika sebaliknya, maka langsung ke langkah 3 – l .
- g. Menentukan faktor kepadatan (α) berdasarkan nilai konstanta (C), indeks iterasi saat ini (t), dan batas iterasi maksimum (t_{max}).
- h. Inisialisasi indeks musang madu saat ini, yaitu $i = 1$.
- i. Jika $i \neq i_{prey}$, maka lanjut ke langkah 3 – j . Jika sebaliknya, maka langsung ke langkah 3 – k .
- j. Menentukan posisi baru musang madu.
 - i. Menentukan intensitas musang madu ke- i (I_i) berdasarkan nilai bilangan *real* acak pada interval $(0, 1)$, konsentrasi kekuatan musang madu (S), dan jarak antara mangsa dengan musang madu ke- i (d_i).
 - ii. Membangkitkan bilangan *real* acak (r) pada interval $(0, 1)$.
 - iii. Jika nilai $r < 0.5$, maka posisi musang madu diperbarui menggunakan fase menggali atau “*digging phase*”.
 - a) Menentukan nilai arah pencarian (F) yang bernilai 1 apabila bilangan *real* acak pada interval $(0, 1)$ yang dibangkitkan bernilai kurang dari 0.5, atau -1 jika sebaliknya.
 - b) Menentukan posisi musang madu yang baru berdasarkan posisi musang madu terbaik (x_{prey}),

arah pencarian (F), kemampuan musang madu mencari makanan (β), intensitas musang madu ke- i (I_i), faktor kepadatan (α), jarak antara mangsa dengan musang madu ke- i (d_i), dan bilangan *real* acak pada interval $(0, 1)$.

- iv. Jika sebaliknya atau nilai $r \geq 0.5$, maka posisi musang madu diperbarui menggunakan fase madu atau “*honey phase*”.
 - a) Menentukan nilai arah pencarian (F) yang bernilai 1 apabila bilangan *real* acak pada interval $(0, 1)$ yang dibangkitkan bernilai kurang dari 0.5, atau -1 jika sebaliknya.
 - b) Menentukan posisi musang madu yang baru berdasarkan posisi musang madu terbaik (x_{prey}), arah pencarian (F), faktor kepadatan (α), jarak antara mangsa dengan musang madu ke- i (d_i), dan bilangan *real* acak pada interval $(0, 1)$.
- v. Mengevaluasi fungsi tujuan untuk posisi baru musang madu (x_{new}) sedemikian sehingga diperoleh nilai fungsi tujuan musang madu yang baru (f_{new}).
- vi. Jika nilai $f_{new} \leq f_i$, maka perbarui posisi dan fungsi tujuan musang madu ke- i dengan musang madu yang baru. Jika sebaliknya, maka posisi dan fungsi tujuan musang madu ke- i tetap.
- vii. Jika nilai $f_{new} \leq f_{prey}$, maka perbarui posisi dan fungsi tujuan mangsa dengan musang madu yang baru. Jika sebaliknya, maka posisi dan fungsi tujuan mangsa tetap.
- k. Jika $i \leq N$, maka perbarui indeks musang madu saat ini (i) dengan menambahkan sebesar satu satuan dan kembali ke langkah 3 – j . Jika sebaliknya, maka perbarui indeks iterasi saat ini (t) dengan menambahkan sebesar satu satuan dan kembali ke langkah 3 – f .

1. Menampilkan solusi terbaik berupa posisi mangsa (x_{prey}) dan nilai fungsi tujuannya (f_{prey}).
4. Membuat program berdasarkan langkah-langkah metode penelitian di atas.
5. Mengimplementasikan program pada contoh kasus.

Langkah-langkah implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup And Delivery* (VRPSPD) disajikan dalam bentuk *flowchart* pada **Lampiran 1**.

BAB IV

PEMBAHASAN

Pada bagian ini akan diberikan penjelasan terkait implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan tujuan untuk membangun satu set rute kendaraan sedemikian sehingga (i) setiap pelanggan dikunjungi tepat satu kendaraan; (ii) setiap kendaraan melakukan pengiriman paling banyak pada satu rute; (iii) setiap rute kendaraan dimulai dan berakhir di depot; (iv) permintaan pengambilan dan pengiriman setiap pelanggan dalam satu kunjungan harus dipenuhi; (v) kapasitas kendaraan tidak melebihi total permintaan pada rute kendaraan; dan (vi) total biaya perjalanan diminimalkan.

4.1 Prosedur *Honey Badger Algorithm* (HBA) dalam Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

Prosedur *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dimulai dengan menginputkan data yang terdiri dari data koordinat posisi untuk depot dan masing-masing pelanggan, data permintaan pengambilan dan pengiriman barang untuk setiap pelanggan, serta menginisialisasikan parameter yang dibutuhkan dalam HBA maupun VRPSPD. Kemudian dilanjutkan dengan menjalankan prosedur HBA dengan membangkitkan sejumlah posisi musang madu yang akan diperbarui menggunakan fase menggali maupun fase madu, sedemikian sehingga diperoleh musang madu yang terbaik atau mangsa yang akan menjadi solusi dari VRPSPD. Prosedur dari HBA akan berhenti saat mencapai batas iterasi maksimum yang telah ditentukan. Ilustrasi selengkapnya terkait prosedur implementasi *Honey Badger Algorithm* (HBA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) disajikan pada **Gambar 4.1**.

Prosedur HBA untuk menyelesaikan VRPSPD**Input** : Data VRPSPD dan parameter HBA**Output** : Rute dengan fungsi tujuan optimal**begin**

input data ();

inisialisasi parameter ();

membangkitkan posisi awal musang madu ();

mengevaluasi fungsi tujuan awal ();

menentukan x_{prey} sementara (); $t \leftarrow 1$;**while** ($t \leq tmax$)

menentukan faktor kepadatan ();

 $n \leftarrow 1$;**do****if** ($n \neq n_{prey}$)

menentukan intensitas ();

 $r \leftarrow \text{random}(0, 1)$;**if** ($r < 0.5$)

menentukan posisi baru musang madu menggunakan fase menggali ();

else

menentukan posisi baru musang madu menggunakan fase madu ();

end if

mengevaluasi fungsi tujuan baru ();

memperbarui posisi dan fungsi tujuan musang madu ();

memperbarui posisi dan fungsi tujuan mangsa ();

end if $n++$;**while** ($n \leq N$) $t++$;**end while**menampilkan x_{prey} ;**end**

Gambar 4.1 Prosedur HBA untuk Menyelesaikan VRPSPD

Selengkapnya akan dijelaskan prosedur *Honey Badger Algorithm* (HBA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery*.

4.1.1 Input Data *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

Prosedur yang pertama adalah menginputkan data berupa data koordinat posisi untuk depot dan masing-masing pelanggan, serta data permintaan pengiriman

untuk setiap pelanggan yang disimpan dalam bentuk *file* berformat *txt*. Prosedur untuk input data disajikan pada **Gambar 4.2**.

Prosedur input data

Input : *data_jarak.txt*, *data_delivery_pickup.txt*, dan *banyak_pelanggan*

Output : $jarak_{i,j}$ dan $delivery_pickup_{i,j}$

begin

$tp_jarak_{i,j} \leftarrow \text{openfile}("data_jarak.txt");$

$tp_delivery_pickup_i \leftarrow \text{openfile}("data_delivery_pickup.txt");$

for $i = 1$ **to** *banyak_pelanggan*

for $j = 1$ **to** *banyak_pelanggan*

$$jarak_{i,j} \leftarrow \sqrt{(tp_jarak_{i,0} - tp_jarak_{j,0})^2 + (tp_jarak_{i,1} - tp_jarak_{j,1})^2};$$

end for

$$rasio \leftarrow \min \left\{ \frac{tp_jarak_{i,0}}{tp_jarak_{i,1}}, \frac{tp_jarak_{i,1}}{tp_jarak_{i,0}} \right\};$$

$delivery_pickup_{0,i} \leftarrow \text{round}(rasio \times tp_delivery_pickup_i);$

$delivery_pickup_{1,i} \leftarrow \text{round}((1 - rasio) \times tp_delivery_pickup_i);$

end for

end

Gambar 4.2 Prosedur Input Data

Dalam prosedur penginputan data tersebut, dilakukan pentrasformasian data koordinat posisi depot dan pelanggan menjadi data jarak dalam bentuk matriks $n \times n$ dengan n adalah banyak pelanggan ditambah depot. Selain itu, dilakukan juga pembagian data permintaan pengiriman berdasarkan tingkat rasio yang diperoleh sehingga didapatkan data permintaan untuk pengiriman dan pengambilan barang untuk setiap pelanggan. Setelah melakukan proses penginputan dan pengonversian data, kemudian dilanjutkan dengan proses inisialisasi parameter.

4.1.2 Inisialisasi Parameter *Honey Badger Algorithm* (HBA) dan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

Prosedur inisialisasi parameter dilakukan dengan menginputkan nilai parameter yang dibutuhkan dalam *Honey Badger Algorithm* (HBA) dan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) seperti banyak musang madu dalam suatu populasi (N), konstanta (C), kemampuan musang madu dalam mencari makanan (β), muatan awal barang pada kendaraan

yang akan dikirimkan ke beberapa pelanggan (%), dan batas maksimum iterasi (t_{max}). Prosedur untuk inisialisasi parameter disajikan pada **Gambar 4.3**.

Prosedur inisialisasi parameter

Input : -

Output : N , C , β , ρ , dan t_{max}

begin

$N \leftarrow \text{input}(\text{"Banyak Musang Madu (N) : "});$

$C \leftarrow \text{input}(\text{"Konstanta (C) : "});$

$\beta \leftarrow \text{input}(\text{"Kemampuan Musang Madu Mencari Makanan (\beta) : "});$

$\rho \leftarrow \text{input}(\text{"Persen Muatan Awal Pada Kendaraan (%) : "});$

$t_{max} \leftarrow \text{input}(\text{"Batas Iterasi Maksimum (tmax) : "});$

end

Gambar 4.3 Prosedur Inisialisasi Parameter

Setelah proses penginputan nilai parameter yang akan digunakan, maka langkah selanjutnya adalah dimulai dengan membangkitkan posisi awal musang madu.

4.1.3 Membangkitkan Posisi Awal Musang Madu

Proses pembangkitan posisi awal musang madu dilakukan dengan menggunakan bangkitan bilangan *real* acak pada interval $(0, 1)$ yang dilakukan sejumlah banyak pelanggan dan diulang sebanyak N musang. Dalam proses tersebut juga dibangkitkan urutan pelanggan yang dimulai dari pelanggan 1 hingga pelanggan terakhir. Dari hasil bangkitan bilangan *real* acak tersebut akan diurutkan berdasarkan nilai yang terkecil hingga yang terbesar, yang kemudian juga berpengaruh kepada urutan pelanggan yang telah dibangkitkan tersebut mengikuti urutan dari bilangan *real* acak yang diurutkan. Dengan demikian didapatkan solusi untuk setiap musang madu dalam populasi dalam bentuk bilangan *real* acak dan urutan pelanggan sebagai calon rute yang akan dievaluasi. Prosedur membangkitkan populasi awal disajikan pada **Gambar 4.4**.

Prosedur membangkitkan posisi awal musang madu**Input** : N dan $banyak_pelanggan$ **Output** : $random_{i,j}$ dan $solusi_{i,j}$

```

begin
  for  $i = 1$  to  $N$ 
     $konter\_pelanggan \leftarrow 1$ ;
    for  $j = 1$  to  $banyak\_pelanggan$ 
       $random_{i,j} \leftarrow \mathbf{random}(0,1)$ ;
       $solusi_{i,j} \leftarrow konter\_pelanggan$ ;
       $konter\_pelanggan ++$ ;
    end for
    for  $j = 0$  to  $banyak\_pelanggan$ 
      for  $k = j + 1$  to  $banyak\_pelanggan$ 
        if ( $random_{i,k} < random_{i,j}$ )
           $temp\_solusi \leftarrow solusi_{i,j}$ ;
           $solusi_{i,j} \leftarrow solusi_{i,k}$ ;
           $solusi_{i,k} \leftarrow temp\_solusi$ ;
        end if
      end for
    end for
  end for
end

```

Gambar 4.4 Prosedur Membangkitkan Posisi Awal Musang Madu

Setelah mendapatkan bilangan *real* acak dan calon rute untuk masing-masing musang madu dalam populasi, nilai tersebut akan diteruskan kepada evaluasi fungsi tujuan posisi awal musang madu untuk menentukan rute yang terbentuk dan nilai fungsi tujuan berupa total jarak tempuh kendaraan. Prosedur untuk proses tersebut dapat dilihat pada **Lampiran 2**.

4.1.4 Menentukan Musang Madu Terbaik/Mangsa (x_{prey}) Sementara

Proses selanjutnya adalah menentukan musang madu terbaik sementara dari posisi awal musang madu berdasarkan nilai fungsi tujuan yang terkecil. Proses tersebut dilakukan dengan membandingkan setiap nilai fungsi tujuan yang ada dan menangkap indeks musang madu yang memiliki nilai fungsi tujuan terkecil. Perolehan indeks musang madu yang terkecil tersebut akan digunakan untuk menyimpan nilai bilangan *real* acak, calon rute, dan fungsi tujuan untuk musang

madu terbaik atau mangsa dalam variabel baru. Prosedur untuk menentukan musang madu terbaik/mangsa (x_{prey}) sementara disajikan pada **Gambar 4.5**.

Prosedur menentukan musang madu terbaik/mangsa (x_{prey}) sementara
Input : N , $banyak_pelanggan$, $fungsi_tujuan_i$, $random_{i,j}$, dan $solusi_{i,j}$
Output : x_{prey} , $fungsi_tujuan_prey$, $random_prey_i$, dan $solusi_prey_{0,i}$

begin
 $x_{prey} \leftarrow 0$;
for $i = 1$ **to** N
 if ($fungsi_tujuan_i < fungsi_tujuan_{x_{prey}}$)
 $x_{prey} \leftarrow i$;
 end if
end for
for $i = 1$ **to** $banyak_pelanggan$
 $random_prey_i \leftarrow random_{x_{prey},i}$;
 $solusi_prey_{0,i} \leftarrow solusi_{x_{prey},i}$;
end for
 $fungsi_tujuan_prey \leftarrow fungsi_tujuan_{prey}$;
end

Gambar 4.5 Prosedur menentukan musang madu terbaik (x_{prey}) sementara

Setelah memperoleh bilangan *real* acak, calon rute, dan fungsi tujuan musang madu terbaik atau mangsa, maka proses dilanjutkan untuk menentukan posisi baru untuk setiap musang madu dalam populasi selain musang madu terbaik.

4.1.5 Menentukan Faktor Kepadatan

Sebelum menentukan posisi baru untuk setiap musang madu, dilakukan penentuan nilai faktor kepadatan menggunakan persamaan (2.16) yang menggunakan parameter seperti konstanta (C) dan batas iterasi maksimum (t_{max}). Nilai faktor kepadatan akan terus menurun seiring dengan bertambahnya iterasi. Prosedur menentukan faktor kepadatan disajikan pada **Gambar 4.6**.

Prosedur faktor kepadatan
Input : C , t , dan t_{max}
Output : $alpha$

begin
 $alpha \leftarrow C \times e^{-\frac{t}{t_{max}}}$;
end

Gambar 4.6 Prosedur Menentukan Faktor Kepadatan

Setelah diperoleh nilai dari faktor kepadatan, langkah berikutnya adalah menentukan nilai intensitas bau mangsa untuk setiap musang madu.

4.1.6 Menentukan Intensitas

Penentuan nilai intensitas bau mangsa untuk setiap musang madu dilakukan dengan menghitung terlebih dahulu nilai jarak antara musang madu dengan mangsa menggunakan persamaan (2.15), sumber kekuatan atau konsentrasi kekuatan musang madu menggunakan persamaan (2.14), dan bangkitan bilangan *real* acak pada interval $(0,1)$ yang dilakukan sejumlah banyak pelanggan. Kemudian dilakukan perhitungan untuk menentukan nilai intensitas bau mangsa untuk setiap musang madu berdasarkan perolehan nilai jarak antara musang madu dengan mangsa, sumber kekuatan atau konsentrasi kekuatan musang madu, dan bangkitan bilangan *real* acak pada interval $(0,1)$, menggunakan persamaan (2.13). Prosedur menentukan nilai intensitas musang madu disajikan pada **Gambar 4.7**.

Prosedur menentukan intensitas

Input : $n, N, banyak_pelanggan, random_{i,j}$, dan $random_prey_i$

Output : I_i dan d_i

begin

for $k = 1$ **to** $banyak_pelanggan$

$S_k \leftarrow 0;$

$d_k \leftarrow 0;$

$I_k \leftarrow 0;$

end for

$musang_i \leftarrow n;$

if $(n = N - 1)$

$musang_j \leftarrow 0;$

else

$musang_j \leftarrow n + 1;$

end if

for $k = 1$ **to** $banyak_pelanggan$

$r2_k \leftarrow \text{random}(0,1);$

$S_k \leftarrow (random_{musang_i,k} - random_{musang_j,k})^2;$

$d_k \leftarrow random_prey_k - random_{musang_i,k};$

$I_k \leftarrow r2_k \times \left(\frac{S_k}{4 \times \pi \times (d_k)^2} \right);$

end for

end

Gambar 4.7 Prosedur Menentukan Intensitas

Setelah diperoleh nilai intensitas musang madu, maka dapat dilanjutkan dengan melakukan proses penentuan posisi baru musang madu yang ada dengan menggunakan fase menggali atau fase madu.

4.1.7 Menentukan Posisi Baru Musang Madu Menggunakan Fase Menggali

Salah satu proses pembaruan untuk memperbarui posisi musang madu adalah pada fase menggali yang menggunakan persamaan (2.18). Hasil pembaruan posisi tersebut menghasilkan bilangan *real* acak yang baru dan urutan pelanggan yang dimulai dari pelanggan 1 sampai dengan pelanggan terakhir. Prosedur pembaruan posisi musang madu dengan fase menggali disajikan pada **Gambar 4.8**.

Prosedur menentukan posisi baru musang madu menggunakan fase menggali

Input : n , $banyak_pelanggan$, β , α , I_i , d_i , $random_prey_i$, dan $random_{i,j}$

Output : $random_baru_{i,j}$ dan $solusi_{i,j}$

begin

$r6 \leftarrow random(0, 1);$

if ($r6 \leq 0.5$)

$F \leftarrow 1;$

else

$F \leftarrow -1;$

end if

$konter_pelanggan \leftarrow 1;$

for $k = 1$ **to** $banyak_pelanggan$

$solusi_{n,k} \leftarrow konter_pelanggan;$

$konter_pelanggan ++;$

$r3_k \leftarrow random(0, 1);$

$r4_k \leftarrow random(0, 1);$

$r5_k \leftarrow random(0, 1);$

$random_baru_{n,k} \leftarrow random_prey_k + (F \times \beta \times I_k \times random_{n,k}) + (F \times r3_k \times \alpha \times d_k \times |\cos(2 \times \pi \times r4_k) \times (1 - \cos(2 \times \pi \times r5_k))|);$

end for

for $i = 1$ **to** $banyak_pelanggan$

for $j = i + 1$ **to** $banyak_pelanggan$

if ($random_baru_{n,j} < random_baru_{n,i}$)

$urut_solusi \leftarrow solusi_{n,i};$

$solusi_{n,i} \leftarrow solusi_{n,j};$

$solusi_{n,j} \leftarrow urut_solusi;$

end if

end for

end for

end

Gambar 4.8 Prosedur Penentuan Posisi Baru Musang Madu pada Fase Menggali

Dari bilangan *real* acak yang baru tersebut kemudian akan dilakukan proses pengurutan dari nilai yang terkecil hingga nilai yang terbesar. Pengurutan tersebut juga akan berpengaruh kepada urutan pelanggan untuk musang madu tersebut mengikuti bilangan *real* acak yang telah terurut sebelumnya. Dengan demikian didapatkan solusi untuk setiap musang madu pada populasi dalam bentuk bilangan *real* acak dan urutan pelanggan sebagai calon rute yang akan dievaluasi. Setelah memperoleh bilangan *real* acak dan calon rute yang baru dari musang madu yang diperbarui, nilai tersebut akan diteruskan kepada proses evaluasi fungsi tujuan musang madu untuk menentukan rute yang terbentuk dan nilai fungsi tujuan berupa total jarak tempuh kendaraan. Prosedur untuk proses tersebut dapat dilihat pada **Lampiran 3**.

4.1.8 Menentukan Posisi Baru Musang Madu Menggunakan Fase Madu

Proses pembaruan posisi musang madu yang lainnya adalah pada fase madu yang menggunakan persamaan (2.19). Penentuan posisi baru pada fase madu menggunakan lebih sedikit parameter dibandingkan dengan fase menggali yang menggunakan lebih banyak parameter. Beberapa parameter pembeda dalam fase ini adalah bahwa dalam perhitungannya tidak menggunakan nilai kemampuan musang madu dalam mencari makanan (β) dan intensitas bau mangsa untuk setiap musang madu (I). Hasil pembaruan tersebut akan memberikan bilangan *real* acak yang baru dan urutan pelanggan yang dimulai dari pelanggan 1 sampai dengan pelanggan terakhir, sehingga dilakukan juga proses pengurutan bilangan *real* acak dari nilai yang terkecil hingga nilai yang terbesar. Proses pengurutan tersebut juga akan berpengaruh kepada urutan pelanggan untuk musang madu tersebut mengikuti bilangan *real* acak yang telah terurut sebelumnya. Dengan demikian didapatkan solusi untuk setiap musang madu dalam populasi dalam bentuk bilangan *real* acak dan urutan pelanggan sebagai calon rute yang akan dievaluasi. Prosedur pembaruan posisi musang madu dengan fase madu disajikan pada **Gambar 4.9**.

Prosedur menentukan posisi baru musang madu menggunakan fase madu**Input** : n , banyak_pelanggan , α , d_i , dan random_prey_i **Output** : $\text{random_baru}_{i,j}$ dan $\text{solusi}_{i,j}$ **begin** $r6 \leftarrow \text{random}(0, 1);$ **if** ($r6 \leq 0.5$) $F \leftarrow 1;$ **else** $F \leftarrow -1;$ **end if** $\text{konter_pelanggan} \leftarrow 1;$ **for** $k = 1$ **to** banyak_pelanggan $\text{solusi}_{n,k} \leftarrow \text{konter_pelanggan};$ $\text{konter_pelanggan} ++;$ $r7_k \leftarrow \text{random}(0, 1);$ $\text{random_baru}_{n,k} \leftarrow \text{random_prey}_k + (F \times r7_k \times \alpha \times d_k);$ **end for****for** $i = 1$ **to** banyak_pelanggan **for** $j = i + 1$ **to** banyak_pelanggan **if** ($\text{random_baru}_{n,j} < \text{random_baru}_{n,i}$) $\text{urut_solusi} \leftarrow \text{solusi}_{n,i};$ $\text{solusi}_{n,i} \leftarrow \text{solusi}_{n,j};$ $\text{solusi}_{n,j} \leftarrow \text{urut_solusi};$ **end if****end for****end for****end**

Gambar 4.9 Prosedur Penentuan Posisi Baru Musang Madu pada Fase Madu

Setelah memperoleh bilangan *real* acak dan calon rute yang baru dari musang madu yang diperbarui, nilai tersebut akan diteruskan kepada evaluasi fungsi tujuan yang dapat dilihat pada **Lampiran 3**.

4.1.9 Memperbarui Posisi dan Fungsi Tujuan Musang Madu

Proses selanjutnya adalah melakukan pengecekan terhadap nilai fungsi tujuan yang baru dengan nilai fungsi tujuan yang lama. Proses pengecekan tersebut dilakukan untuk menentukan apakah posisi musang madu yang baru akan menggantikan posisi musang madu yang lama, atau posisi musang madu yang lama tetap. Prosedur memperbarui posisi dan fungsi tujuan musang madu disajikan pada **Gambar 4.10**.

Prosedur memperbarui posisi dan fungsi tujuan musang madu**Input** : n , banyak_pelanggan , $\text{random_baru}_{i,j}$, fungsi_tujuan_i ,
dan $\text{fungsi_tujuan_baru}$ **Output** : $\text{random}_{i,j}$ dan fungsi_tujuan_i

```

begin
  if ( $\text{fungsi\_tujuan\_baru} < \text{fungsi\_tujuan}_n$ )
     $\text{fungsi\_tujuan}_n \leftarrow \text{fungsi\_tujuan\_baru}$ ;
    for  $l = 1$  to  $\text{banyak\_pelanggan}$ 
       $\text{random}_{n,l} \leftarrow \text{random\_baru}_{n,l}$ ;
    end for
  end if
end

```

Gambar 4.10 Prosedur Pembaruan Posisi dan Fungsi Tujuan Musang Madu

Setelah dilakukan proses pengecekan terhadap nilai fungsi tujuan yang baru dengan fungsi tujuan yang lama, proses dilanjutkan dengan melakukan pengecekan juga untuk nilai fungsi tujuan yang baru terhadap nilai fungsi tujuan dari mangsa.

4.1.10 Memperbarui Posisi dan Fungsi Tujuan Mangsa

Proses pengecekan nilai fungsi tujuan yang baru terhadap nilai fungsi tujuan dari mangsa dilakukan untuk menentukan apakah posisi musang madu yang baru akan menggantikan posisi mangsa, atau posisi mangsa tetap. Prosedur memperbarui posisi dan fungsi tujuan mangsa disajikan pada **Gambar 4.11**.

Prosedur memperbarui posisi dan fungsi tujuan mangsa**Input** : n , banyak_pelanggan , $\text{random_baru}_{i,j}$, $\text{solusi}_{i,j}$, $\text{fungsi_tujuan_prey}$, dan
 $\text{fungsi_tujuan_baru}$ **Output** : x_{prey} , $\text{fungsi_tujuan_prey}$, $\text{random_prey}_{i,j}$, dan $\text{solusi_prey}_{i,j}$

```

begin
  if ( $\text{fungsi\_tujuan\_baru} < \text{fungsi\_tujuan\_prey}$ )
     $x_{\text{prey}} \leftarrow n$ ;
     $\text{fungsi\_tujuan\_prey} \leftarrow \text{fungsi\_tujuan\_baru}$ ;
    for  $l = 1$  to  $\text{banyak\_pelanggan}$ 
       $\text{random\_prey}_l \leftarrow \text{random\_baru}_{n,l}$ ;
       $\text{solusi\_prey}_{0,l} \leftarrow \text{solusi}_{n,l}$ ;
    end for
  end if
end

```

Gambar 4.11 Prosedur Pembaruan Posisi dan Fungsi Tujuan Mangsa

Prosedur memperbarui posisi musang madu dilakukan untuk seluruh musang madu dalam populasi hingga menemukan solusi terbaik, yang proses tersebut diulang hingga maksimum iterasi terpenuhi.

4.2 Data

Pada skripsi ini data yang digunakan berupa data koordinat posisi untuk depot dan masing-masing pelanggan, serta data permintaan pengiriman barang untuk setiap pelanggan yang diambil dari <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/> yang diakses pada tanggal 28 Maret 2023. **Nagy dan Salhi (2005)** menggunakan teknik rasio untuk mengolah data permintaan pengiriman barang menjadi data VRPSPD dalam bentuk data permintaan pengiriman dan pengambilan barang. Terdapat tiga macam data yang diambil dari <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/> yaitu:

1. Data kecil

Data berukuran kecil dengan 13 pelanggan dan kapasitas maksimum kendaraan sebesar 100 unit barang. Data selengkapnya dapat dilihat pada **Lampiran 4**. Data berukuran kecil juga digunakan untuk pengerjaan VRPSPD secara manual. Secara rinci, data ini diakses dari http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT1.vrp pada 28 Maret 2023.

2. Data sedang

Data berukuran sedang dengan 75 pelanggan dan kapasitas maksimum kendaraan sebesar 140 unit barang. Data selengkapnya dapat dilihat pada **Lampiran 5**. Secara rinci, data ini diakses dari http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT2.vrp pada 28 Maret 2023.

3. Data besar

Data berukuran besar dengan 100 pelanggan dan kapasitas maksimum kendaraan sebesar 200 unit barang. Data selengkapnya dapat dilihat pada **Lampiran 6**. Secara rinci, data ini diakses dari http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT3.vrp pada 28 Maret 2023.

4.3 Penyelesaian Secara Manual Implementasi *Honey Badger Algorithm* (HBA) pada Contoh Kasus *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) Menggunakan Data Kecil

Langkah-langkah penyelesaian secara manual *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) untuk ukuran data kecil menggunakan *Honey Badger Algorithm* (HBA) adalah sebagai berikut:

4.3.1 Input Data dan Inisialisasi Parameter

Data yang digunakan berupa data koordinat posisi untuk depot dan masing-masing pelanggan, serta data permintaan pengiriman barang untuk 13 pelanggan dengan kapasitas maksimum kendaraan sebesar 100 unit barang seperti yang disajikan pada **Lampiran 4**. Data tersebut kemudian ditransformasikan untuk mempermudah perhitungan dalam penyelesaian VRPSPD. Untuk data koordinat posisi depot dan pelanggan, dilakukan transformasi dalam bentuk matriks jarak sebesar $n \times n$, dengan n adalah banyak pelanggan ditambah depot, menggunakan rumus penentuan jarak yang diberikan oleh persamaan berikut:

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Sebagai contoh, penentuan jarak dari depot menuju pelanggan ke-1 diberikan pada perhitungan berikut:

$$\begin{aligned} c_{0,1} &= \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \\ c_{0,1} &= \sqrt{(30 - 37)^2 + (40 - 52)^2} \\ c_{0,1} &= \sqrt{49 + 144} \\ c_{0,1} &= \sqrt{193} \\ c_{0,1} &= 13.8924 \end{aligned}$$

Dengan menggunakan cara yang sama untuk menentukan jarak dari depot ke setiap pelanggan dan antar pelanggan, diperoleh data jarak yang disajikan pada **Tabel 4.1**. Hasil perhitungan data jarak secara lengkap dapat dilihat pada **Lampiran 7**.

Tabel 4.1 Data Jarak untuk Depot dan 13 Pelanggan

$c_{i,j}$	0	1	2	3	4	5	...	13
0	0.0000	13.8924	21.0238	32.5576	17.2047	14.1421	...	29.1548
1	13.8924	0.0000	12.3693	19.2094	31.0644	22.2036	...	41.8688
2	21.0238	12.3693	0.0000	15.2971	37.0135	21.0238	...	50.1199
3	32.5576	19.2094	15.2971	0.0000	49.6790	36.0555	...	61.0737
4	17.2047	31.0644	37.0135	49.6790	0.0000	20.3961	...	15.0333
5	14.1421	22.2036	21.0238	36.0555	20.3961	0.0000	...	35.3553
6	11.4018	16.7631	28.0713	35.3553	21.0238	25.4951	...	27.2029
7	26.4197	22.8254	34.9285	35.0143	37.1214	40.2244	...	39.8497
8	22.0227	11.6619	22.2036	21.0950	37.6431	33.2415	...	45.2217
9	23.0868	24.2074	16.2788	31.0000	32.7567	12.3693	...	47.6760
10	28.3196	34.0147	28.0713	43.0116	31.4006	14.2127	...	46.1736
11	12.0416	12.0830	10.6301	25.0799	26.6271	11.1803	...	40.3113
12	8.0623	20.8806	24.7588	38.2753	12.5300	9.2195	...	26.9258
13	29.1548	41.8688	50.1199	61.0737	15.0333	35.3553	...	0.0000

Sedangkan untuk data permintaan pengiriman barang, dilakukan transformasi yang membagi data tersebut menjadi data permintaan pengiriman barang dan data permintaan pengambilan barang. Pembagian data tersebut menggunakan teknik rasio yang diperkenalkan oleh **Nagy dan Salhi (2005)** dengan perumusan yang diberikan pada persamaan berikut:

$$rasio = \min \left\{ \frac{x_i}{y_i}, \frac{y_i}{x_i} \right\}$$

sehingga nilai permintaan pengiriman barang untuk setiap pelanggan dapat ditentukan menggunakan persamaan berikut:

$$D_i = round(rasio \times q_i)$$

dan nilai permintaan pengambilan barang untuk setiap pelanggan dapat ditentukan menggunakan persamaan berikut:

$$P_i = round((1 - rasio) \times q_i)$$

dengan D_i adalah nilai permintaan pengiriman barang untuk pelanggan ke- i , P_i adalah nilai permintaan pengambilan barang untuk pelanggan ke- i , dan q_i adalah nilai permintaan pengiriman barang untuk pelanggan ke- i pada data asli.

Sebagai contoh, penentuan rasio, nilai permintaan pengiriman, dan pengambilan barang untuk pelanggan ke-1 secara berturut-turut diberikan pada perhitungan berikut:

$$\begin{aligned} rasio &= \min \left\{ \frac{x_i}{y_i}, \frac{y_i}{x_i} \right\} \\ &= \min \left\{ \frac{37}{52}, \frac{52}{37} \right\} \\ &= \min \{0.7115, 1.4054\} \\ &= 0.7115 \end{aligned}$$

$$\begin{aligned} D_1 &= \text{round}(rasio \times q_1) \\ &= \text{round}(0.7115 \times 7) \\ &= \text{round}(4.9805) \\ &= 5 \end{aligned}$$

$$\begin{aligned} P_1 &= \text{round}((1 - rasio) \times q_1) \\ &= \text{round}((0.2885) \times 7) \\ &= \text{round}(2.0195) \\ &= 2 \end{aligned}$$

Dengan menggunakan cara yang sama untuk menentukan nilai permintaan pengiriman dan pengambilan barang untuk setiap pelanggan, diperoleh data permintaan pengiriman dan pengambilan barang untuk setiap pelanggan disajikan pada **Tabel 4.2**.

Tabel 4.2 Data *Delivery* dan *Pickup* untuk 13 Pelanggan

	Pelanggan												
Permintaan	1	2	3	4	5	6	7	8	9	10	11	12	13
Delivery	5	30	13	7	16	7	5	12	7	2	19	28	5
Pickup	2	0	3	2	5	8	14	12	4	3	0	1	18

Parameter yang digunakan untuk penyelesaiannya antara lain banyak musang madu (N) sebanyak 5 musang, konstanta (C) sebesar 2 satuan, kemampuan musang madu mencari makanan (β) sebesar 6 satuan, muatan awal kendaraan (%) sebesar 60% dari kapasitas maksimum kendaraan yaitu sebanyak 60 unit, dan batas maksimum iterasi (t_{max}) sebanyak 1 iterasi.

4.3.2 Membangkitkan Posisi Awal Musang Madu

Pembangkitan posisi awal musang madu dilakukan dengan cara membangkitkan bilangan *real* acak pada interval (0,1) sebanyak jumlah pelanggan. Proses tersebut dilakukan sebanyak jumlah musang madu yang disajikan pada **Tabel 4.3**. Hasil bangkitan bilangan *real* acak untuk setiap musang madu secara lengkap dapat dilihat pada **Lampiran 8**.

Tabel 4.3 Hasil Bangkitan Bilangan *Real* Acak pada Interval (0, 1)

x_i	Pelanggan						
	1	2	3	4	5	...	13
x_1	0.7961	0.4823	0.3591	0.1010	0.0680	...	0.5133
x_2	0.4400	0.9321	0.5785	0.7138	0.4791	...	0.7641
x_3	0.2820	0.8502	0.6831	0.8342	0.4027	...	0.3630
x_4	0.3621	0.3478	0.4665	0.5383	0.8002	...	0.5032
x_5	0.0734	0.9511	0.9702	0.3111	0.8185	...	0.2090

Kemudian dari hasil pembangkitan bilangan *real* acak tersebut, dilakukan proses pengurutan dengan cara mengurutkan bilangan *real* acak pada setiap musang madu dari nilai yang terkecil hingga terbesar untuk memperoleh calon rute. Sebagai contoh untuk musang madu ke-1, bilangan *real* acak beserta urutan pelanggan awal secara berturut-turut adalah sebagai berikut:

0.7961	0.4823	0.3591	0.1010	0.0680	...	0.5133
1	2	3	4	5	...	13

Pengurutan terhadap bilangan *real* acak yang dilakukan secara *ascending* menghasilkan urutan pelanggan yang baru yang diberikan sebagai berikut:

0.0367	0.0680	0.1010	0.3161	0.3591	...	0.8807
10	5	4	12	3	...	9

Dengan cara yang sama untuk musang madu yang lainnya, hasil urutan pelanggan sebagai calon rute disajikan pada **Tabel 4.4**.

Tabel 4.4 Calon Rute untuk Setiap Musang Madu

x_i	Calon Rute												
x_1	10	5	4	12	3	2	13	8	7	1	11	6	9
x_2	6	11	1	5	8	9	3	10	4	13	12	7	2
x_3	12	9	1	13	5	6	10	11	3	7	4	2	8
x_4	6	11	2	1	9	3	7	13	4	12	10	5	8
x_5	1	10	13	11	12	4	7	8	9	6	5	2	3

4.3.3 Mengevaluasi Posisi Awal Musang Madu

Pada *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD), rute yang dilalui oleh kendaraan selalu berawal dan berakhir di depot, serta muatan total untuk setiap kendaraan dalam setiap rute tidak melebihi kapasitas maksimum kendaraan, atau muatan pengiriman pada setiap kendaraan tidak kurang dari total permintaan pengiriman untuk setiap pelanggan dalam setiap rute yang dilalui oleh kendaraan tersebut. Ketika batasan-batasan tersebut dilanggar atau tidak terpenuhi, maka kendaraan harus kembali ke depot, yang dalam pembahasan ini disimbolkan dengan angka 0.

Sebagai contoh, calon rute pada musang madu ke-1 adalah sebagai berikut:

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

Pembentukan rute pada musang madu ke-1 dimulai dari depot menuju ke pelanggan 10, kemudian dilanjutkan menuju ke pelanggan 5 sampai dengan ke pelanggan terakhir, yaitu pelanggan 9, lalu kembali lagi ke depot. Kemudian untuk setiap pelanggan pada rute tersebut dilakukan perhitungan untuk masing-masing permintaan pengiriman dan pengambilan barang beserta jarak tempuh yang dilalui kendaraan tersebut, sedemikian sehingga pada perjalanan dari depot menuju ke pelanggan 5 diperoleh perhitungan sebagai berikut:

$$c_{0,10} = 28.3196 \text{ satuan jarak}$$

$$l > D_{10}$$

$$60 > 2 \text{ unit barang}$$

$$\begin{aligned} l_{10} &= l_0 - D_{10} + P_{10} \\ &= 60 - 2 + 3 \\ &= 61 \text{ unit barang} < C = 100 \end{aligned}$$

Karena batasan yang ada tidak dilanggar, maka kendaraan dapat melanjutkan perjalanannya menuju ke pelanggan selanjutnya, yaitu pelanggan 5 sedemikian sehingga diperoleh perhitungan sebagai berikut:

$$\begin{aligned} c_{0,10} + c_{10,5} &= 28.3196 + 14.2127 \\ &= 42.5323 \text{ satuan jarak} \end{aligned}$$

$$l > D_{10} + D_5$$

$$60 > 2 + 16$$

$$60 > 18 \text{ unit barang}$$

$$\begin{aligned} l_5 &= l_{10} - D_5 + P_5 \\ &= 61 - 16 + 5 \\ &= 50 \text{ unit barang} < C = 100 \end{aligned}$$

Karena batasan yang ada tidak dilanggar, maka kendaraan dapat melanjutkan perjalanannya menuju ke pelanggan selanjutnya, yaitu pelanggan 4.

Dengan melakukan cara yang sama dalam melakukan perhitungan tersebut hingga terdapat batasan yang dilanggar, maka diperoleh rute kendaraan yang pertama pada musang madu ke-1 adalah sebagai berikut:

$$0 - 10 - 5 - 4 - 12 - 0$$

dengan total jarak yang ditempuh kendaraan adalah $c_{0,10} + c_{10,5} + c_{5,4} + c_{4,12} + c_{12,0} = 83.5207$ satuan jarak.

Dengan menggunakan cara yang sama untuk menentukan rute selanjutnya pada musang madu ke-1 maupun rute-rute yang terbentuk pada musang madu yang lain, diperoleh hasil pembentukan rute dan nilai fungsi tujuan untuk keseluruhan musang madu disajikan pada **Tabel 4.5**.

Tabel 4.5 Pembentukan Rute Awal untuk setiap Musang Madu

x_i	Rute	Rute yang Terbentuk	Jarak
x_1	1	0 - 10 - 5 - 4 - 12 - 0	83.5207
	2	0 - 3 - 2 - 13 - 8 - 0	165.2190
	3	0 - 7 - 1 - 11 - 6 - 9 - 0	140.2699
	Total Jarak		389.0096
x_2	1	0 - 6 - 11 - 1 - 5 - 8 - 0	122.7929
	2	0 - 9 - 3 - 10 - 4 - 13 - 0	172.6871
	3	0 - 12 - 7 - 0	68.4967
	4	0 - 2 - 0	42.0476
	Total Jarak		406.0243
x_3	1	0 - 12 - 9 - 1 - 13 - 0	124.3171
	2	0 - 5 - 6 - 10 - 11 - 3 - 0	158.9053
	3	0 - 7 - 4 - 2 - 8 - 0	144.7809
	Total Jarak		428.0033
x_4	1	0 - 6 - 11 - 2 - 0	64.8960
	2	0 - 1 - 9 - 3 - 7 - 13 - 4 - 0	176.2018
	3	0 - 12 - 10 - 5 - 8 - 0	100.3646
	Total Jarak		341.4624
x_5	1	0 - 1 - 10 - 13 - 11 - 12 - 0	156.6670
	2	0 - 4 - 7 - 8 - 9 - 6 - 5 - 0	177.8187
	3	0 - 2 - 3 - 0	68.8785
	Total Jarak		403.3642

Berdasarkan **Tabel 4.5** diperoleh nilai fungsi tujuan untuk musang madu ke-1 sebesar 389.0096 satuan jarak, musang madu ke-2 sebesar 406.0243 satuan jarak, musang madu ke-3 sebesar 428.0033 satuan jarak, musang madu ke-4 sebesar 341.4624 satuan jarak, dan musang madu ke-5 sebesar 403.3642 satuan jarak.

4.3.4 Menentukan Musang Madu Terbaik Sementara/Mangsa (x_{prey})

Musang madu terbaik atau mangsa (x_{prey}) diperoleh berdasarkan hasil perbandingan nilai fungsi tujuan satu dengan nilai fungsi tujuan lainnya dari keseluruhan musang madu yang ada pada populasi. Berdasarkan perolehan nilai fungsi tujuan pada **Tabel 4.5**, musang madu ke-4 memiliki nilai fungsi tujuan yang lebih kecil dibandingkan dengan musang madu lainnya, yaitu sebesar 341.4624 satuan jarak. Dengan demikian musang madu ke-4 menjadi musang madu terbaik atau mangsa (x_{prey}) sementara.

4.3.5 Mengecek Jumlah Maksimum Iterasi

Sebelum menentukan posisi yang baru untuk musang madu, dilakukan pengecekan indeks iterasi saat ini terhadap jumlah maksimum iterasi. Apabila indeks iterasi saat ini (t) tidak lebih jumlah maksimum iterasi (t_{max}), maka proses diteruskan pada pembaruan posisi musang madu. Sebaliknya, apabila indeks iterasi saat ini (t) melebihi jumlah maksimum iterasi (t_{max}), maka proses algoritma berhenti dan menampilkan musang madu terbaik atau mangsa. Karena indeks iterasi saat ini adalah $t = 1$, sedangkan batas iterasi maksimum adalah $t_{max} = 1$, maka proses diteruskan pada pembaruan posisi musang madu untuk setiap musang madu pada populasi kecuali musang madu ke-4.

4.3.6 Menentukan Faktor Kepadatan

Penentuan nilai faktor kepadatan (α) dihitung menggunakan persamaan (2.16), yaitu:

$$\alpha = C \times \exp\left(\frac{-t}{t_{max}}\right)$$

Dengan nilai konstanta $C = 2$, indeks iterasi saat ini $t = 1$, dan batas iterasi maksimum $t_{max} = 1$, maka nilai faktor kepadatan (α) adalah sebagai berikut:

$$\begin{aligned}\alpha &= C \times \exp\left(\frac{-t}{t_{max}}\right) \\ &= 2 \times \exp\left(-\frac{1}{1}\right) \\ &= 0.7358\end{aligned}$$

4.3.7 Menentukan Intensitas

Proses penentuan nilai intensitas bau mangsa untuk setiap musang madu didahului dengan menentukan sumber kekuatan atau konsentrasi kekuatan musang madu (S) yang diberikan pada persamaan (2.14), yaitu:

$$S = (x_i - x_{i+1})^2$$

dan menentukan jarak antara mangsa dengan musang madu ke- i (d_i) yang diberikan pada persamaan (2.15), yaitu:

$$d_i = x_{prey} - x_i$$

Sebagai contoh untuk menentukan nilai intensitas musang madu ke-1, maka akan ditentukan terlebih dahulu nilai sumber kekuatan musang madu ke-1 dan jarak mangsa dengan musang madu ke-1. Dengan solusi untuk musang madu ke-1, musang madu ke-2, dan mangsa yang merupakan musang madu ke-4 berdasarkan **Tabel 4.3** secara berturut-turut adalah sebagai berikut:

x_1	0.7961	0.4823	0.3591	0.1010	0.0680	...	0.5133
x_2	0.4400	0.9321	0.5785	0.7138	0.4791	...	0.7641
x_{prey}	0.3621	0.3478	0.4665	0.5383	0.8002	...	0.5032

Perhitungan untuk sumber kekuatan musang madu ke-1 dan jarak antara mangsa dengan musang madu ke-1 secara berturut-turut adalah sebagai berikut:

$$\begin{aligned} S &= (x_1 - x_2)^2 \\ &= (0.7961 - 0.4400)^2 \\ &= 0.1269 \end{aligned}$$

$$\begin{aligned} d_1 &= x_{prey} - x_1 \\ &= 0.3621 - 0.7961 \\ &= -0.4340 \end{aligned}$$

Cara serupa dilakukan juga untuk elemen-elemen lainnya pada solusi musang madu ke-1, musang madu ke-2, dan mangsa sehingga diperoleh sumber kekuatan musang madu ke-1 dan jarak antara mangsa dengan musang madu ke-1 adalah sebagai berikut:

S	0.1269	0.2024	0.0481	0.3755	0.1690	...	0.0629
d_1	-0.4340	-0.1345	0.1073	0.4373	0.7322	...	-0.0102

Selanjutnya akan dibangkitkan bilangan *real* acak r_2 pada interval $(0, 1)$ yang dilakukan sebanyak jumlah pelanggan. Hasil bangkitan bilangan *real* acak r_2 adalah sebagai berikut:

r_2	0.9808	0.7676	0.1433	0.1629	0.9815	...	0.3193
-------	--------	--------	--------	--------	--------	-----	--------

Berdasarkan perolehan nilai sumber kekuatan musang madu ke-1, jarak antara mangsa dengan musang madu ke-1, dan bangkitan bilangan *real* acak r_2 , maka akan ditentukan nilai intensitas musang madu ke-1 menggunakan persamaan (2.13), yaitu:

$$I_i = r_2 \times \frac{S}{4\pi d_i^2}$$

Perhitungan untuk menentukan intensitas musang madu ke-1 adalah sebagai berikut:

$$\begin{aligned}
 I_1 &= r_2 \times \frac{S}{4\pi d_1^2} \\
 &= 0.9808 \times \frac{0.1269}{4(3.14)(-0.4340)^2} \\
 &= 0.0526
 \end{aligned}$$

Cara serupa dilakukan juga untuk elemen-elemen lainnya pada sumber kekuatan musang madu ke-1, jarak antara mangsa dengan musang madu ke-1, dan bangkitan bilangan *real* acak r_2 sehingga diperoleh intensitas untuk musang madu ke-1 adalah sebagai berikut:

I_1	0.0526	0.6834	0.0477	0.0255	0.0246	...	15.4815
-------	--------	--------	--------	--------	--------	-----	---------

Dengan menggunakan cara yang sama dalam menentukan intensitas musang madu lainnya, diperoleh intensitas untuk setiap musang madu disajikan pada **Tabel 4.6**. Hasil perolehan nilai intensitas untuk setiap musang madu secara lengkap dapat dilihat pada **Lampiran 9**.

Tabel 4.6 Intensitas untuk Setiap Musang Madu

I_1	0.0526	0.6834	0.0477	0.0255	0.0246	...	15.4815
I_2	0.1619	0.0012	0.0348	0.0306	0.0031	...	0.1664
I_3	0.0735	0.0164	0.0635	0.0554	0.0534	...	0.0442
I_5	0.1083	0.0056	0.1170	0.0228	89.2918	...	0.0650

4.3.8 Menentukan Posisi Baru Musang Madu

Posisi baru musang madu ditentukan berdasarkan perilaku musang madu dalam mencari makanannya, yaitu pada fase menggali atau fase madu. Penentuan ini dilakukan dengan membangkitkan bilangan *real* acak (r) pada interval $(0, 1)$. Posisi baru musang madu akan ditentukan menggunakan fase menggali apabila nilai r memenuhi $0 < r < 0.5$, dan akan ditentukan menggunakan fase madu apabila nilai r memenuhi $0.5 \leq r < 1$. Proses didahului dengan membangkitkan bilangan *real* acak (r) pada interval $(0, 1)$. Sebagai contoh untuk musang madu ke-1, diperoleh bangkitan nilai r adalah sebesar 0.7024, sehingga posisi musang madu ke-1 ditentukan menggunakan fase madu.

Sebelum menentukan posisi baru untuk musang madu ke-1, terlebih dahulu akan ditentukan nilai dari penanda dalam menentukan arah pencarian solusi (F) menggunakan persamaan (2.17), yaitu:

$$F = \begin{cases} 1, & \text{Jika } r_6 \leq 0.5 \\ -1, & \text{Jika sebaliknya} \end{cases}$$

dengan r_6 adalah bilangan *real* acak pada interval $(0, 1)$. Dalam kasus ini didapatkan bangkitan nilai r_6 untuk musang madu ke-1 adalah sebesar 0.1348, sehingga nilai dari F adalah sebesar 1.

Selanjutnya akan dibangkitkan juga bilangan *real* acak r_7 pada interval $(0, 1)$ yang dilakukan sebanyak jumlah pelanggan. Hasil bangkitan bilangan *real* acak r_7 adalah sebagai berikut:

r_7	0.3982	0.5390	0.8171	0.3604	0.7693	...	0.1873
-------	--------	--------	--------	--------	--------	-----	--------

Berdasarkan perolehan nilai F dan bangkitan bilangan *real* acak r_7 maka akan ditentukan posisi baru untuk musang madu ke-1 dengan menggunakan fase madu yang diberikan pada persamaan (2.19), yaitu:

$$x_{new} = x_{prey} + F \times r_7 \times \alpha \times d_i$$

Dengan nilai faktor kepadatan sebesar 0.7358, serta solusi untuk mangsa dan jarak antara mangsa dengan musang madu ke-1 berdasarkan **Tabel 4.3** secara berturut-turut adalah sebagai berikut:

x_{prey}	0.3621	0.3478	0.4665	0.5383	0.8002	...	0.5032
d_1	-0.4340	-0.1345	0.1073	0.4373	0.7322	...	-0.0102

Perhitungan untuk menentukan posisi baru untuk musang madu ke-1 adalah sebagai berikut:

$$\begin{aligned}
 x_{1\ new} &= x_{prey} + F \times r_7 \times \alpha \times d_1 \\
 &= 0.3621 + 1 \times 0.3982 \times 0.7358 \times (-0.4340) \\
 &= 0.2350
 \end{aligned}$$

Cara serupa dilakukan juga untuk elemen-elemen lainnya pada posisi untuk mangsa, jarak antara mangsa dengan musang madu ke-1, dan bangkitan bilangan *real* acak r_7 sehingga diperoleh posisi baru untuk musang madu ke-1 adalah sebagai berikut:

x_1	0.2350	0.2945	0.5310	0.6542	1.2147	...	0.5018
-------	--------	--------	--------	--------	--------	-----	--------

Dengan menggunakan cara yang sama dalam menentukan posisi baru untuk musang madu lainnya, diperoleh posisi baru untuk setiap musang madu disajikan pada **Tabel 4.7**. Hasil perolehan posisi baru untuk setiap musang madu secara lengkap dapat dilihat pada **Lampiran 9**.

Tabel 4.7 Posisi Baru untuk Setiap Musang Madu

x_i	Posisi Baru						
x_1	0.2350	0.2945	0.5310	0.6542	1.2147	...	0.5018
x_2	0.7070	0.1671	0.5497	0.5695	0.8153	...	0.9994
x_3	0.5368	0.3643	0.4891	0.7096	1.1470	...	0.6382

x_i	Posisi Baru						
x_5	0.2297	0.3671	0.4944	0.4838	0.8093	...	0.3418

Kemudian dilakukan proses pengurutan yang sama seperti pada langkah sebelumnya untuk memperoleh calon rute yang disajikan pada **Tabel 4.8**.

Tabel 4.8 Calon Rute untuk Setiap Musang Madu yang Baru

x_i	Calon Rute												
x_1	6	11	1	2	9	7	13	3	4	12	10	8	5
x_2	6	2	11	7	3	4	1	5	10	13	12	8	9
x_3	6	11	7	2	3	1	9	13	4	12	10	5	8
x_5	11	1	13	2	9	7	6	12	4	3	10	8	5

4.3.9 Mengevaluasi Posisi Musang Madu Baru (x_{new})

Evaluasi posisi baru untuk setiap musang madu dihitung berdasarkan calon rute baru yang telah diperoleh. Dengan menggunakan cara yang sama pada proses evaluasi sebelumnya, diperoleh hasil pembentukan rute dan nilai fungsi tujuan untuk keseluruhan musang madu disajikan pada **Tabel 4.9**.

Tabel 4.9 Pembentukan Rute Baru untuk setiap Musang Madu

x_i	Rute	Rute yang Terbentuk	Jarak
x_1	1	0 - 6 - 11 - 1 - 0	59.2175
	2	0 - 2 - 9 - 7 - 13 - 3 - 0	216.8813
	3	0 - 4 - 12 - 10 - 8 - 0	120.2008
	4	0 - 5 - 0	28.2842
	Total Jarak		424.5838
x_2	1	0 - 6 - 2 - 11 - 0	62.1448
	2	0 - 7 - 3 - 4 - 1 - 5 - 10 - 13 - 0	253.9221
	3	0 - 12 - 8 - 9 - 0	96.9541
	Total Jarak		413.0210

x_i	Rute	Rute yang Terbentuk	Jarak
x_3	1	0 - 6 - 11 - 7 - 0	92.9635
	2	0 - 2 - 3 - 1 - 9 - 13 - 0	156.5685
	3	0 - 4 - 12 - 10 - 5 - 0	80.9149
	4	0 - 8 - 0	44.0454
	Total Jarak		374.4923
x_5	1	0 - 11 - 1 - 13 - 2 - 0	137.1371
	2	0 - 9 - 7 - 6 - 12 - 4 - 0	133.4394
	3	0 - 3 - 10 - 8 - 5 - 0	168.5708
	Total Jarak		439.1473

Berdasarkan **Tabel 4.9** diperoleh nilai fungsi tujuan baru untuk musang madu ke-1 sebesar 424.5838 satuan jarak, musang madu ke-2 sebesar 413.0210 satuan jarak, musang madu ke-3 sebesar 374.4923 satuan jarak, dan musang madu ke-5 sebesar 439.1473 satuan jarak.

4.3.10 Memperbarui Posisi dan Fungsi Tujuan Musang Madu

Pembaruan posisi dan fungsi tujuan musang madu dilakukan dengan membandingkan nilai fungsi tujuan musang madu sebelum diperbarui dengan nilai fungsi tujuan musang madu sesudah diperbarui. Apabila nilai fungsi tujuan baru lebih baik atau sama dengan nilai fungsi tujuan lama, maka posisi dan fungsi tujuan musang madu yang baru akan menggantikan posisi dan fungsi tujuan musang madu yang lama. Apabila sebaliknya, maka posisi dan fungsi tujuan musang madu yang lama tetap atau tidak terjadi perubahan. Proses pembaruan nilai fungsi tujuan disajikan pada **Tabel 4.10**.

Tabel 4.10 Pembaruan Posisi dan Fungsi Tujuan untuk Setiap Musang Madu

x_i	Nilai fungsi tujuan lama	Nilai fungsi tujuan baru	Nilai fungsi tujuan yang dipilih	Keterangan
x_1	389.0096	424.5838	389.0096	Nilai yang lama
x_2	406.0243	413.0210	406.0243	Nilai yang lama
x_3	428.0033	374.4923	374.4923	Nilai yang baru

x_i	Nilai fungsi tujuan lama	Nilai fungsi tujuan baru	Nilai fungsi tujuan yang dipilih	Keterangan
x_5	403.3642	439.1473	403.3642	Nilai yang lama

Berdasarkan **Tabel 4.10** diperoleh bahwa pada musang madu ke-3 memiliki nilai fungsi tujuan baru yang lebih kecil daripada nilai fungsi tujuan lama sehingga posisi dan fungsi tujuan musang madu yang baru menggantikan posisi dan fungsi tujuan musang madu yang lama. Sedangkan pada musang madu ke-1, ke-2, dan ke-5 diperoleh bahwa nilai fungsi tujuan baru tidak lebih baik daripada nilai fungsi tujuan lama, sehingga posisi dan fungsi tujuan musang madu yang baru tidak menggantikan posisi dan fungsi tujuan musang madu yang lama.

4.3.11 Memperbarui Posisi dan Fungsi Tujuan Mangsa

Pembaruan posisi dan fungsi tujuan mangsa dilakukan dengan membandingkan nilai fungsi tujuan mangsa dengan nilai fungsi tujuan musang madu sesudah diperbarui. Apabila nilai fungsi tujuan baru lebih baik atau sama dengan nilai fungsi tujuan mangsa, maka posisi dan fungsi tujuan musang madu yang baru akan menggantikan posisi dan fungsi tujuan mangsa. Apabila sebaliknya, maka posisi dan fungsi tujuan mangsa tetap atau tidak terjadi perubahan.

Berdasarkan perolehan nilai fungsi tujuan yang telah diperbarui pada **Tabel 4.10** maka keseluruhan nilai fungsi tujuan saat ini untuk setiap musang madu selain mangsa disajikan pada **Tabel 4.11**.

Tabel 4.11 Nilai Fungsi Tujuan Saat Ini untuk Setiap Musang Madu

x_i	Nilai fungsi tujuan
x_1	389.0096
x_2	406.0243
x_3	374.4923
x_5	403.3642

Diketahui nilai fungsi tujuan mangsa adalah sebesar 341.4624 satuan jarak, maka proses pembaruan nilai fungsi tujuan mangsa disajikan pada **Tabel 4.12**.

Tabel 4.12 Pembaruan Posisi dan Fungsi Tujuan Mangsa

x_i	Nilai fungsi tujuan saat ini	Nilai fungsi tujuan mangsa saat ini	Nilai fungsi tujuan yang dipilih	Keterangan
x_1	389.0096	341.4624	341.4624	Mangsa tetap
x_2	406.0243	341.4624	341.4624	Mangsa tetap
x_3	374.4923	341.4624	341.4624	Mangsa tetap
x_5	403.3642	341.4624	341.4624	Mangsa tetap

Berdasarkan **Tabel 4.12** diperoleh bahwa posisi dan fungsi tujuan mangsa tetap yaitu pada musang madu ke-4.

4.3.12 Mengecek Jumlah Maksimum Iterasi

Setelah melakukan proses pembaruan posisi musang madu secara keseluruhan, langkah selanjutnya adalah dengan memperbarui nilai indeks iterasi saat ini atau t . Diketahui bahwa nilai indeks iterasi saat ini adalah $t = 1$, maka dengan menambahkan sebesar satu satuan pada t didapatkan nilai indeks iterasi yang baru adalah $t = 2$. Karena indeks iterasi saat ini adalah $t = 2$, sedangkan jumlah iterasi maksimum adalah $t_{max} = 1$, maka proses algoritma berhenti dan menampilkan solusi musang madu yang terbaik.

4.3.13 Menampilkan Solusi Terbaik

Karena batas maksimum iterasi telah terlampaui, maka berdasarkan perhitungan manual menggunakan *Honey Badger Algorithm* yang dilakukan sebanyak satu iterasi, diperoleh musang madu ke-4 sebagai solusi musang madu terbaik atau mangsa dengan total jarak tempuh yang dilalui kendaraan sebesar 341.4624 satuan jarak. Rute yang terbentuk pada musang madu ke-4 disajikan pada **Tabel 4.13**.

Tabel 4.13 Rute yang Terbentuk pada Musang Madu Terbaik/Mangsa

Rute	Rute yang Terbentuk	Jarak
1	0 - 6 - 11 - 2 - 0	64.8960
2	0 - 1 - 9 - 3 - 7 - 13 - 4 - 0	176.2018

Rute	Rute yang Terbentuk	Jarak
3	0 - 12 - 10 - 5 - 8 - 0	100.3646
Total Jarak		341.4624

4.4 Implementasi Program pada *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

Program implementasi *Honey Badger Algorithm* (HBA) pada *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dibuat menggunakan bahasa pemrograman C++ dengan bantuan *software Visual Studio Code* 1.76. *Source Code* program secara lengkap dapat dilihat pada **Lampiran 10**. Program yang telah dibuat, diimplementasikan pada tiga ukuran data VRPSPD, yaitu untuk data kecil (13 pelanggan), data sedang (75 pelanggan), dan data besar (100 pelanggan) yang masing-masing dapat dilihat pada **Lampiran 4**, **Lampiran 5**, dan **Lampiran 6**. Dalam implementasi tersebut, juga dilakukan penentuan persentase muatan awal barang yang akan diangkut dari depot berdasarkan kapasitas maksimum kendaraan menggunakan tiga nilai, diantaranya adalah 50%, 60%, dan 70%. Hal tersebut dilakukan atas dasar pertimbangan agar rute pelanggan yang terbentuk dapat lebih efisien dikarenakan dalam VRPSPD melibatkan pengambilan maupun pengiriman barang dari pelanggan yang dilakukan secara bersamaan. Untuk masing-masing data, akan digunakan nilai parameter yang berbeda pada konstanta (C), banyak musang madu dalam populasi (N), dan batas maksimum iterasi (t_{max}), serta dilakukan proses *running* sebanyak 10 kali untuk setiap nilai parameternya sehingga mendapatkan hasil yang lebih baik.

4.4.1 Implementasi pada Data Berukuran Kecil

Nilai fungsi tujuan terbaik dari hasil *running* program pada data kecil menggunakan $\beta = 6$ (Hashim dkk, 2022), persentase muatan awal barang pada kendaraan sebesar 70% dari kapasitas maksimum kendaraan, yaitu sebanyak 70 unit barang (pemilihan persentase muatan awal barang yang diangkut dari depot ditentukan berdasarkan hasil *running* program dengan hasil yang cenderung lebih baik menggunakan data kecil pada **Lampiran 11**), serta jumlah musang madu

dalam populasi, batas maksimum iterasi, dan konstanta (C) yang bervariasi, yang disajikan pada **Tabel 4.14**.

Tabel 4.14 Hasil *Running* Program pada Data Kecil

Maksimal Iterasi	Jumlah Musang Madu	Konstanta (C)		
		1	2	10
10	10	285.737	296.191	295.49
	50	295.052	298.563	276.612
	100	265.759	297.078	290.776
100	10	269.566	274.621	262.511
	50	271.598	250.845	252.483
	100	261.171	252.835	255.415
1000	10	242.562	250.586	250.235
	50	237.028	238.348	235.482
	100	236.308	238.348	235.482

Berdasarkan pola hasil *running* pada **Tabel 4.14**, dapat disimpulkan bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi maka hasil yang diperoleh cenderung semakin baik. Sementara besar kecil nilai parameter C tidak berpengaruh terhadap hasil yang diperoleh. Selain itu, pada **Tabel 4.14** diperoleh juga nilai fungsi tujuan terbaik sebesar 235.482 satuan jarak dengan rute pelanggan yang terbentuk disajikan pada **Tabel 4.15**.

Tabel 4.15 Rute Terbaik yang Terbentuk pada Implementasi Data Kecil

Rute	Rute yang Terbentuk	Jarak
1	0 - 12 - 4 - 13 - 6 - 7 - 8 - 1 - 0	118.911
2	0 - 3 - 2 - 9 - 10 - 5 - 0	104.53
3	0 - 11 - 0	12.0416
Total Jarak		235.482

4.4.2 Implementasi pada Data Berukuran Sedang

Nilai fungsi tujuan terbaik dari hasil *running* program pada data sedang menggunakan $\beta = 6$ (Hashim dkk, 2022), persentase muatan awal barang pada kendaraan sebesar 70% dari kapasitas maksimum kendaraan, yaitu sebanyak 98 unit barang (pemilihan persentase muatan awal barang yang diangkut dari depot ditentukan berdasarkan hasil *running* program dengan hasil yang cenderung lebih

baik menggunakan data sedang pada **Lampiran 11**), serta jumlah musang madu dalam populasi, batas maksimum iterasi, dan konstanta (C) yang bervariasi, yang disajikan pada **Tabel 4.16**.

Tabel 4.16 Hasil *Running* Program pada Data Sedang

Maksimal Iterasi	Jumlah Musang Madu	Konstanta (C)		
		1	2	10
10	10	2261.1	2257.48	2184.75
	50	2247.59	2219.85	2169.19
	100	2222.95	2210	2228.63
100	10	2113.42	2169.36	2199.33
	50	2116.42	2088.54	2119.84
	100	1999.83	2000.74	2109.65
1000	10	1965.62	2022.16	2052.01
	50	1901.08	1885.45	1938.87
	100	1859.56	1860.24	1915.67

Berdasarkan pola hasil *running* pada **Tabel 4.16**, dapat disimpulkan bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi, maka hasil yang diperoleh semakin baik. Sementara semakin kecil nilai parameter C , maka hasil yang diperoleh cenderung semakin baik. Selain itu, pada **Tabel 4.16** diperoleh juga nilai fungsi tujuan terbaik sebesar 1859.56 satuan jarak dengan rute pelanggan yang terbentuk disajikan pada **Tabel 4.17**.

Tabel 4.17 Rute Terbaik yang Terbentuk pada Implementasi Data Sedang

Rute	Rute yang Terbentuk	Jarak
1	0 - 13 - 11 - 66 - 58 - 29 - 61 - 0	177.358
2	0 - 38 - 8 - 37 - 60 - 47 - 36 - 2 - 43 - 6 - 0	192.888
3	0 - 57 - 46 - 34 - 32 - 9 - 24 - 68 - 62 - 0	184.052
4	0 - 39 - 25 - 56 - 41 - 16 - 17 - 31 - 14 - 72 - 0	238.133
5	0 - 64 - 22 - 49 - 63 - 28 - 1 - 20 - 30 - 0	216.397
6	0 - 67 - 40 - 53 - 59 - 0	103.318
7	0 - 19 - 70 - 71 - 73 - 4 - 74 - 69 - 42 - 3 - 33 - 55 - 50 - 23 - 0	367.5
8	0 - 26 - 52 - 75 - 65 - 18 - 12 - 44 - 51 - 0	177.07

Rute	Rute yang Terbentuk	Jarak
9	0 - 7 - 10 - 45 - 15 - 5 - 21 - 48 - 35 - 27 - 0	175.823
10	0 - 54 - 0	27.018
Total Jarak		1859.56

4.4.3 Implementasi pada Data Berukuran Besar

Nilai fungsi tujuan terbaik dari hasil *running* program pada data besar menggunakan $\beta = 6$ (Hashim dkk, 2022), persentase muatan awal barang pada kendaraan sebesar 70% dari kapasitas maksimum kendaraan, yaitu sebanyak 140 unit barang (pemilihan persentase muatan awal barang yang diangkut dari depot ditentukan berdasarkan hasil *running* program dengan hasil yang cenderung lebih baik menggunakan data besar pada **Lampiran 11**), serta jumlah musang madu dalam populasi, batas maksimum iterasi, dan konstanta (C) yang bervariasi, yang disajikan pada **Tabel 4.18**.

Tabel 4.18 Hasil *Running* Program pada Data Besar

Maksimal Iterasi	Jumlah Musang Madu	Konstanta (C)		
		1	2	10
10	10	3095.07	3088.87	3050.82
	50	3002.57	3012.2	3041.39
	100	3041.16	2993.88	2959.23
100	10	2872.35	2928.74	2978.62
	50	2829.37	2864.66	2826.06
	100	2824.06	2667.75	2889.84
1000	10	2697.37	2682.1	2792.71
	50	2661.05	2637.78	2775.38
	100	2474.58	2632.72	2658.63

Berdasarkan pola hasil *running* pada **Tabel 4.18**, dapat disimpulkan bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi, maka hasil yang diperoleh semakin baik. Sementara semakin kecil nilai parameter C , maka hasil yang diperoleh cenderung semakin baik dengan total jarak tempuh lebih kecil. Selain itu, pada **Tabel 4.18** diperoleh juga nilai fungsi tujuan terbaik sebesar 2474.58 satuan jarak dengan rute pelanggan yang terbentuk disajikan pada **Tabel 4.19**.

Tabel 4.19 Rute Terbaik yang Terbentuk pada Implementasi Data Besar

Rute	Rute yang Terbentuk	Jarak
1	0 - 44 - 38 - 58 - 82 - 79 - 34 - 92 - 46 - 17 - 68 - 76 - 26 - 37 - 0	387.937
2	0 - 10 - 32 - 7 - 49 - 63 - 20 - 19 - 36 - 30 - 33 - 9 - 3 - 25 - 70 - 71 - 90 - 77 - 73 - 0	444.261
3	0 - 100 - 42 - 41 - 75 - 4 - 62 - 51 - 29 - 66 - 78 - 80 - 69 - 67 - 24 - 55 - 22 - 86 - 52 - 50 - 65 - 0	557.628
4	0 - 12 - 13 - 21 - 23 - 85 - 87 - 60 - 8 - 31 - 27 - 0	197.708
5	0 - 59 - 88 - 48 - 16 - 61 - 6 - 94 - 93 - 45 - 14 - 28 - 0	213.542
6	0 - 81 - 84 - 18 - 5 - 95 - 98 - 47 - 56 - 40 - 97 - 91 - 53 - 39 - 72 - 15 - 57 - 89 - 0	392.037
7	0 - 2 - 54 - 74 - 43 - 99 - 96 - 1 - 35 - 64 - 11 - 83 - 0	281.467
Total Jarak		2474.58

Berdasarkan hasil *running* untuk tiga ukuran data yang diberikan pada **Tabel 4.14**, **Tabel 4.16**, dan **Tabel 4.18**, diperoleh bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi, maka hasil yang diperoleh semakin baik. Sementara semakin kecil parameter konstanta (C), maka hasil yang diperoleh cenderung semakin baik dengan total jarak tempuh lebih kecil.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pembahasan yang telah dijabarkan pada BAB IV diperoleh kesimpulan sebagai berikut:

1. *Honey Badger Algorithm* (HBA) dapat diterapkan untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dengan langkah-langkah antara lain; menginputkan data dan inisialisasi parameter, membangkitkan posisi awal musang madu, mengevaluasi fungsi tujuan awal, menentukan posisi musang madu terbaik/mangsa (x_{prey}) sementara, menentukan faktor kepadatan (α), menentukan intensitas bau mangsa terhadap musang madu (I), menentukan posisi baru musang madu menggunakan fase menggali atau fase madu, mengevaluasi fungsi tujuan baru, memperbarui posisi dan fungsi tujuan musang madu, memperbarui posisi dan fungsi tujuan mangsa, mengulangi proses pembaruan posisi musang madu untuk semua musang madu dalam populasi sampai dengan diperoleh musang madu terbaik/mangsa (x_{prey}) hingga iterasi maksimum terpenuhi.
2. Program *Honey Badger Algorithm* (HBA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) dibuat menggunakan bahasa pemrograman C++ dengan bantuan *software Visual Studio Code 1.76*. program ditulis secara urut sesuai dengan langkah-langkah dalam HBA, dimulai dengan prosedur input data dalam fungsi “input data ()”, menginisialisasikan parameter dalam fungsi “inisialisasi parameter ()”, membangkitkan posisi awal musang madu dalam fungsi “membangkitkan posisi awal musang madu ()”, mengevaluasi fungsi tujuan awal dalam fungsi “mengevaluasi fungsi tujuan awal ()”, menentukan musang madu terbaik (x_{prey}) sementara dalam fungsi “menentukan x_{prey} sementara ()”, menentukan nilai faktor kepadatan dalam fungsi “menentukan faktor kepadatan ()”, menentukan intensitas bau mangsa

terhadap musang madu dalam fungsi “menentukan intensitas ()”, menentukan posisi baru musang madu menggunakan fase menggali dalam fungsi “menentukan posisi baru musang madu menggunakan fase menggali ()” atau fase madu dalam fungsi “menentukan posisi baru musang madu menggunakan fase madu ()”, mengevaluasi fungsi tujuan baru dalam fungsi “mengevaluasi fungsi tujuan baru ()”, memperbaiki posisi dan fungsi tujuan musang madu dalam fungsi “memperbarui posisi dan fungsi tujuan musang madu ()”, memperbaiki posisi dan fungsi tujuan mangsa dalam fungsi “memperbarui posisi dan fungsi tujuan mangsa ()”, kemudian mengulangi proses pembaruan posisi musang madu untuk semua musang madu dalam populasi sampai dengan diperoleh musang madu terbaik/mangsa (x_{prey}) hingga iterasi maksimum terpenuhi.

3. Implementasi program untuk ukuran data kecil diperoleh solusi terbaik dengan total jarak tempuh yaitu sebesar 235.482 satuan jarak, untuk ukuran data sedang diperoleh solusi terbaik dengan total jarak tempuh yaitu sebesar 1859.56 satuan jarak, dan untuk ukuran data besar diperoleh solusi terbaik dengan total jarak tempuh yaitu sebesar 2474.58 satuan jarak. Pola yang diperoleh dalam hasil *running* program untuk tiga ukuran data, menunjukkan bahwa semakin besar batas maksimum iterasi dan semakin banyak musang madu dalam populasi, maka hasil yang diperoleh semakin baik. Sementara semakin kecil nilai parameter C , maka hasil yang diperoleh cenderung semakin baik dengan total jarak tempuh lebih kecil.

5.2 Saran

Pada penelitian ini, implementasi program *Honey Badger Algorithm* (HBA) untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) ditinjau berdasarkan nilai parameter C yang bervariasi. Untuk penelitian selanjutnya, diharapkan melakukan peninjauan juga terhadap parameter HBA yang lain, yaitu parameter β , untuk mengetahui apakah parameter tersebut berpengaruh dalam menentukan solusi yang lebih baik pada HBA. Selain itu, diharapkan juga dilakukan pengembangan maupun hybrid terhadap HBA untuk

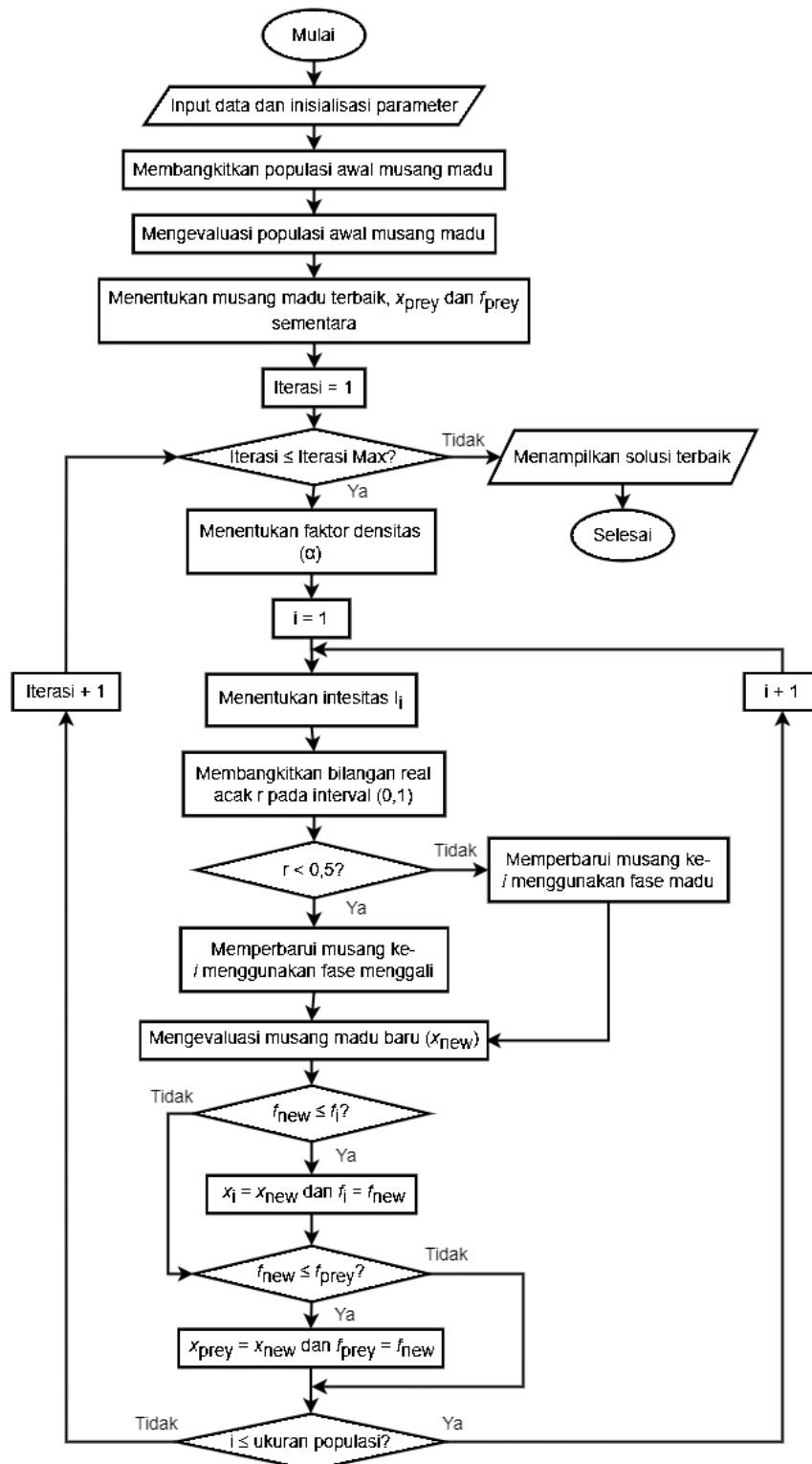
dapat meningkatkan performa dan efektivitas dalam menyelesaikan permasalahan optimasi lainnya atau permasalahan nyata.

DAFTAR PUSTAKA

- Ai, T. J. dan Kachitvichyanukul, V., 2009, A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*, **36**(5), 1693-1702
- Belfiore, P., Tsugunobu, H., dan Yoshizaki, Y., 2008, *Vehicle Routing Problem*, Intech, Croatia
- Capacitated Vehicle Routing Problem Library, <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/>, 28 Maret 2023
- Caric, T. dan Gold, H., 2008, *Vehicle Routing Problem*. Vienna: InTech.
- Catay, B., 2010, A New Saving-Based Ant Algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, *Expert systems with applications*, **37**(10), 6809-6817
- Cetin, S. and Gencer, C., 2015, A Heuristic Algorithm for Vehicle Routing Problems with Simultaneous Pick-Up and Delivery and Hard Time Windows. *Open Journal of Social Sciences*, **3**(3), 35-41
- Dantzig, G. B. dan Ramser, J. H., 1959, The Truck Dispatching Problem, *Management science*, **6**(1), 80-91
- Dethloff, J., 2001, Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-Up, *OR-Spektrum*, **23**(1), 79-96
- Dhiman, G. dan Kumar, V., 2017, Spotted Hyena Optimizer: A Novel Bio-Inspired Based Metaheuristic Technique for Engineering Applications, *Advances in Engineering Software*, **114**, 48-70
- Gajpal, Y., dan Abad, P., 2009, An Ant Colony System (ACS) for Vehicle Routing Problem with Simultaneous Delivery and Pickup, *Computers & Operations Research*, **36**(12), 3215-3223

- Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S., dan Al-Atabany, W., 2022, Honey Badger Algorithm: New Metaheuristic Algorithm for Solving Optimization Problems, *Mathematics and Computers in Simulation*, **192**, 84-110
- Jaegere, N. D., Defraeye, M., dan Nieuwenhuyse, I. V., 2014, *The Vehicle Routing Problem: State of the Art Classification and Review*, KU Leuven – Faculty of Economics and Business, Leuven
- Koç, Ç., Laporte, G., dan Tükenmez, İ., 2020, A Review of Vehicle Routing with Simultaneous Pickup and Delivery, *Computers & Operations Research*, **122**, 104987
- Mezura-Montes, E., dan Coello, C. A. C., 2005, Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms, *Mexican international conference on artificial intelligence*, 652-662
- Min, H., 1989, The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points, *Transportation Research Part A: General*, **23**(5), 377-386
- Nagy, G. dan Salhi, S., 2005, Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, *European journal of operational research*, **162**(1), 126-141
- Stanković, M., 2021, The Economic Importance of Transportation Sector, *Knowledge-International Journal*, **47**(1), 143-146
- Tasan, A. S. dan Gen, M., 2012, A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries, *Computers & Industrial Engineering*, **62**(3), 755-761

Lampiran 1 Flowchart Implementasi Honey Badger Algorithm (HBA) dalam Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)



Lampiran 2 Prosedur Mengevaluasi Posisi Awal Musang Madu

Prosedur mengevaluasi posisi awal musang madu**Input** : $n, N, banyak_pelanggan, solusi_{i,j}$, dan $jarak_{i,j}$ **Output** : $fungsi_tujuan_i$ **begin****for** $i = 0$ **to** N $m \leftarrow 0$; $permintaan_pengiriman \leftarrow kapasitas \times persen/100$; $konter_rute \leftarrow 0$; $penanda \leftarrow 0$; $total_jarak \leftarrow 0$;**while** ($solusi_{i,m} \neq solusi_{i,banyak_pelanggan}$) $permintaan_pengiriman -= delivery_pickup_{0,solusi_{i,m}}$; $muatan_tersedia \leftarrow permintaan_pengiriman +$
 $delivery_pickup_{1,solusi_{i,m}}$; $sisa_kapasitas \leftarrow kapasitas - muatan_tersedia$; $berhenti \leftarrow m$;**if** ($permintaan_pengiriman \leq 0 \parallel muatan_tersedia >$
 $kapasitas \parallel sisa_kapasitas \leq 0$)**for** $k = penanda$ **to** $berhenti$ **if** ($k = penanda$) $total_jarak += jarak_{0,solusi_{i,k}}$;**else if** ($k = berhenti - 1$) $total_jarak += jarak_{solusi_{i,k},0}$; $total_jarak += jarak_{solusi_{i,k-1},solusi_{i,k}}$;**else** $total_jarak += jarak_{solusi_{i,k-1},solusi_{i,k}}$;**end if****end for** $konter_rute ++$; $permintaan_pengiriman \leftarrow kapasitas \times persen/100$; $penanda \leftarrow m$; $m \leftarrow berhenti - 1$;**end if** $m ++$;**end while****if** ($permintaan_pengiriman > 0 \parallel muatan_tersedia <$
 $kapasitas \parallel sisa_kapasitas > 0 \parallel solusi_{i,m} =$ $solusi_{i,banyak_pelanggan}$)**for** $k = penanda$ **to** $berhenti$ **if** ($k = penanda$)

```
         $total\_jarak += jarak_{0,solusi_{i,k}};$   
    else if ( $k = berhenti$ )  
         $total\_jarak += jarak_{solusi_{i,k},0};$   
         $total\_jarak += jarak_{solusi_{i,k-1},solusi_{i,k}};$   
    else  
         $total\_jarak += jarak_{solusi_{i,k-1},solusi_{i,k}};$   
    end if  
    end for  
    end if  
     $fungsi\_tujuan_i \leftarrow total\_jarak;$   
    end for  
end
```

Lampiran 3 Prosedur Mengevaluasi Posisi Musang Madu Baru

Prosedur mengevaluasi posisi musang madu baru**Input** : $n, N, banyak_pelanggan, solusi_{i,j}$, dan $jarak_{i,j}$ **Output** : $fungsi_tujuan_baru$ **begin** $m \leftarrow 0;$ $permintaan_pengiriman \leftarrow kapasitas \times persen/100;$ $konter_rute \leftarrow 0;$ $penanda \leftarrow 0;$ $total_jarak \leftarrow 0;$ **while** ($solusi_{n,m} \neq solusi_{n,banyak_pelanggan}$) $permintaan_pengiriman -= delivery_pickup_{0,solusi_{n,m}};$ $muatan_tersedia \leftarrow permintaan_pengiriman +$ $delivery_pickup_{1,solusi_{n,m}};$ $sisa_kapasitas \leftarrow kapasitas - muatan_tersedia;$ $berhenti \leftarrow m;$ **if** ($permintaan_pengiriman \leq 0 \parallel muatan_tersedia > kapasitas \parallel sisa_kapasitas \leq 0$)**for** $k = penanda$ **to** $berhenti$ **if** ($k = penanda$) $total_jarak += jarak_{0,solusi_{n,k}};$ **else if** ($k = berhenti - 1$) $total_jarak += jarak_{solusi_{n,k},0};$ $total_jarak += jarak_{solusi_{n,k-1},solusi_{n,k}};$ **else** $total_jarak += jarak_{solusi_{n,k-1},solusi_{n,k}};$ **end if****end for** $konter_rute ++;$ $permintaan_pengiriman \leftarrow kapasitas \times persen/100;$ $penanda \leftarrow m;$ $m \leftarrow berhenti - 1;$ **end if** $m ++;$ **end while****if** ($permintaan_pengiriman > 0 \parallel muatan_tersedia < kapasitas \parallel sisa_kapasitas > 0 \parallel solusi_{n,m} =$ $solusi_{n,banyak_pelanggan}$)**for** $k = penanda$ **to** $berhenti$ **if** ($k = penanda$)

```
         $total\_jarak += jarak_{0,solusi_{n,k}};$   
    else if ( $k = berhenti$ )  
         $total\_jarak += jarak_{solusi_{n,k},0};$   
         $total\_jarak += jarak_{solusi_{n,k-1},solusi_{n,k}};$   
    else  
         $total\_jarak += jarak_{solusi_{n,k-1},solusi_{n,k}};$   
    end if  
    end for  
    end if  
     $fungsi\_tujuan\_baru \leftarrow total\_jarak;$   
end
```

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

		Koordinat		Permintaan
		X	Y	
Depot	0	30	40	0
Pelanggan	1	37	52	7
	2	49	49	30
	3	52	64	16
	4	20	26	9
	5	40	30	21
	6	21	47	15
	7	17	63	19
	8	31	62	23
	9	52	33	11
	10	51	21	5
	11	42	41	19
	12	31	32	29
	13	5	25	23

Kapasitas Maksimum Kendaraan
100

Sumber: http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT1.vrp

Lampiran 5 Data Sedang Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) dengan 75 Pelanggan

		Koordinat		Permintaan
		X	Y	
Depot	0	40	40	0
Pelanggan	1	22	22	18
	2	36	26	26
	3	21	45	11
	4	45	35	30
	5	55	20	21
	6	33	34	19
	7	50	50	15
	8	55	45	16
	9	26	59	29
	10	40	66	26
	11	55	65	37
	12	35	51	16
	13	62	35	12
	14	62	57	31
	15	62	24	8
	16	21	36	19
	17	33	44	20
	18	9	56	13
	19	62	48	15
	20	66	14	22
	21	44	13	28
	22	26	13	12
	23	11	28	6
	24	7	43	27
	25	17	64	14
	26	41	46	18
	27	55	34	17
	28	35	16	29
	29	52	26	13
	30	43	26	22
	31	31	76	25
	32	22	53	28
	33	26	29	27

		Koordinat		Permintaan
		X	Y	
Pelanggan	34	50	40	19
	35	55	50	10
	36	54	10	12
	37	60	15	14
	38	47	66	24
	39	30	60	16
	40	30	50	33
	41	12	17	15
	42	15	14	11
	43	16	19	18
	44	21	48	17
	45	50	30	21
	46	51	42	27
	47	50	15	19
	48	48	21	20
	49	12	38	5
	50	15	56	22
	51	29	39	12
	52	54	38	19
	53	55	57	22
	54	67	41	16
	55	10	70	7
	56	6	25	26
	57	65	27	14
	58	40	60	21
	59	70	64	24
	60	64	4	13
	61	36	6	15
	62	30	20	18
	63	20	30	11
	64	15	5	28
	65	50	70	9
	66	57	72	37
	67	45	42	30
	68	38	33	10
	69	50	4	8
	70	66	8	11

		Koordinat		Permintaan
		X	Y	
Pelanggan	71	59	5	3
	72	35	60	1
	73	27	24	6
	74	40	20	10
	75	40	37	20

Kapasitas Maksimum Kendaraan
140

Sumber: http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT2.vrp

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

		Koordinat		Permintaan
		X	Y	
Depot	0	35	35	0
Pelanggan	1	41	49	10
	2	35	17	7
	3	55	45	13
	4	55	20	19
	5	15	30	26
	6	25	30	3
	7	20	50	5
	8	10	43	9
	9	55	60	16
	10	30	60	16
	11	20	65	12
	12	50	35	19
	13	30	25	23
	14	15	10	20
	15	30	5	8
	16	10	20	19
	17	5	30	2
	18	20	40	12
	19	15	60	17
	20	45	65	9
	21	45	20	11
	22	45	10	18
	23	55	5	29
	24	65	35	3
	25	65	20	6
	26	45	30	17
	27	35	40	16
	28	41	37	16
	29	64	42	9
	30	40	60	21
	31	31	52	27
	32	35	69	23
	33	53	52	11

		Koordinat		Permintaan
		X	Y	
Pelanggan	34	65	55	14
	35	63	65	8
	36	2	60	5
	37	20	20	8
	38	5	5	16
	39	60	12	31
	40	40	25	9
	41	42	7	5
	42	24	12	5
	43	23	3	7
	44	11	14	18
	45	6	38	16
	46	2	48	1
	47	8	56	27
	48	13	52	36
	49	6	68	30
	50	47	47	13
	51	49	58	10
	52	27	43	9
	53	37	31	14
	54	57	29	18
	55	63	23	2
	56	53	12	6
	57	32	12	7
	58	36	26	18
	59	21	24	28
	60	17	34	3
	61	12	24	13
	62	24	58	19
	63	27	69	10
	64	15	77	9
	65	62	77	20
	66	49	73	25
	67	67	5	25
	68	56	39	36
	69	37	47	6
	70	37	56	5

		Koordinat		Permintaan
		X	Y	
Pelanggan	71	57	68	15
	72	47	16	25
	73	44	17	9
	74	46	13	8
	75	49	11	18
	76	49	42	13
	77	53	43	14
	78	61	52	3
	79	57	48	23
	80	56	37	6
	81	55	54	26
	82	15	47	16
	83	14	37	11
	84	11	31	7
	85	16	22	41
	86	4	18	35
	87	28	18	26
	88	26	52	9
	89	26	35	15
	90	31	67	3
	91	15	19	1
	92	22	22	2
	93	18	24	22
	94	26	27	27
	95	25	24	20
	96	22	27	11
	97	25	21	12
	98	19	21	10
	99	20	26	9
	100	18	18	17

Kapasitas Maksimum Kendaraan
200

Sumber: http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances//CMT/CMT3.vrp

Lampiran 7 Data Jarak untuk Depot dan 13 Pelanggan

$c_{i,j}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	13.8924	21.0238	32.5576	17.2047	14.1421	11.4018	26.4197	22.0227	23.0868	28.3196	12.0416	8.0623	29.1548
1	13.8924	0	12.3693	19.2094	31.0644	22.2036	16.7631	22.8254	11.6619	24.2074	34.0147	12.0830	20.8806	41.8688
2	21.0238	12.3693	0	15.2971	37.0135	21.0238	28.0713	34.9285	22.2036	16.2788	28.0713	10.6301	24.7588	50.1199
3	32.5576	19.2094	15.2971	0	49.6790	36.0555	35.3553	35.0143	21.0950	31.0000	43.0116	25.0799	38.2753	61.0737
4	17.2047	31.0644	37.0135	49.6790	0	20.3961	21.0238	37.1214	37.6431	32.7567	31.4006	26.6271	12.5300	15.0333
5	14.1421	22.2036	21.0238	36.0555	20.3961	0	25.4951	40.2244	33.2415	12.3693	14.2127	11.1803	9.2195	35.3553
6	11.4018	16.7631	28.0713	35.3553	21.0238	25.4951	0	16.4924	18.0278	34.0147	39.6989	21.8403	18.0278	27.2029
7	26.4197	22.8254	34.9285	35.0143	37.1214	40.2244	16.4924	0	14.0357	46.0977	54.0370	33.3017	34.0147	39.8497
8	22.0227	11.6619	22.2036	21.0950	37.6431	33.2415	18.0278	14.0357	0	35.8050	45.6180	23.7065	30.0000	45.2217
9	23.0868	24.2074	16.2788	31.0000	32.7567	12.3693	34.0147	46.0977	35.8050	0	12.0416	12.8062	21.0238	47.6760
10	28.3196	34.0147	28.0713	43.0116	31.4006	14.2127	39.6989	54.0370	45.6180	12.0416	0	21.9317	22.8254	46.1736
11	12.0416	12.0830	10.6301	25.0799	26.6271	11.1803	21.8403	33.3017	23.7065	12.8062	21.9317	0	14.2127	40.3113

$c_{i,j}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
12	8.0623	20.8806	24.7588	38.2753	12.5300	9.2195	18.0278	34.0147	30.0000	21.0238	22.8254	14.2127	0	26.9258
13	29.1548	41.8688	50.1199	61.0737	15.0333	35.3553	27.2029	39.8497	45.2217	47.6760	46.1736	40.3113	26.9258	0

Lampiran 8 Hasil Bangkitan Bilangan *Real* Acak pada Interval (0, 1)

x_i	Pelanggan												
	1	2	3	4	5	6	7	8	9	10	11	12	13
x_1	0.7961	0.4823	0.3591	0.1010	0.0680	0.8538	0.6649	0.5707	0.8807	0.0367	0.8517	0.3161	0.5133
x_2	0.4400	0.9321	0.5785	0.7138	0.4791	0.0809	0.8642	0.5305	0.5327	0.6855	0.0831	0.7724	0.7641
x_3	0.2820	0.8502	0.6831	0.8342	0.4027	0.4200	0.7068	0.8957	0.1059	0.6650	0.6747	0.0303	0.3630
x_4	0.3621	0.3478	0.4665	0.5383	0.8002	0.0137	0.4980	0.9107	0.3875	0.7538	0.0151	0.5601	0.5032
x_5	0.0734	0.9511	0.9702	0.3111	0.8185	0.7543	0.3808	0.6335	0.6898	0.1993	0.2684	0.2699	0.2090

Lampiran 9 Hasil Perhitungan Nilai Intensitas Musang Madu dan Hasil Perhitungan Posisi Baru Musang Madu

I_1	0.0526	0.6834	0.0477	0.0255	0.0246	0.0456	0.1101	0.0005	0.0228	0.0220	0.0081	0.2346	15.4815
I_2	0.1619	0.0012	0.0348	0.0306	0.0031	0.2819	0.0005	0.0499	0.4242	0.0063	3.7476	0.1479	0.1664
I_3	0.0735	0.0164	0.0635	0.0554	0.0534	0.0706	0.0010	0.0622	0.0543	0.0542	0.0345	0.0386	0.0442
I_5	0.1083	0.0056	0.1170	0.0228	89.2918	0.0012	0.4656	0.0018	0.0121	0.0045	0.2832	0.0015	0.0650

x_i	Solusi Baru												
x_1	0.2350	0.2945	0.5310	0.6542	1.2147	-0.3739	0.3843	1.0955	0.3303	1.0545	-0.0055	0.7360	0.5018
x_2	0.7070	0.1671	0.5497	0.5695	0.8153	0.0268	0.4014	1.2245	1.3680	0.8182	0.3345	1.0250	0.9994
x_3	0.5368	0.3643	0.4891	0.7096	1.1470	-0.0047	0.3053	1.2505	0.6034	1.0430	-0.0034	0.8342	0.6382
x_5	0.2297	0.3671	0.4944	0.4838	0.8093	0.4329	0.4180	0.8031	0.3899	0.7239	0.1955	0.4626	0.3418

Lampiran 10 Source Code Program Implementasi *Honey Badger Algorithm* (HBA) dalam Menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD)

```
#include <iostream>
#include <conio.h>
#include <fstream>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <cmath>
#include <cstring>
#include <algorithm>

using namespace std;

int t1, cek, banyak_pelanggan, konter_pelanggan, kapasitas, n,
delivery_pickup[110][110], temp_delivery_pickup[110], solusi[110][110],
solusi_baru[110][110], solusi_prey[110][110], temp_solusi, urutan_solusi;
int akhir, konter_rute, penanda, berhenti, permintaan_pengiriman,
muatan_tersedia, sisa_kapasitas;
int m, F, musang_i, musang_j, xprey, pilih;
double jarak_tempuh, rasio, persen, random[110][110],
random_baru[110][110], random_prey[110], urutan_random[110][110], I[110],
d[110], S[110], r, r2[110], r3[110], r4[110], r5[110], r6, r7[110],
temp_random, alpha, C, temp_C, beta, temp_beta, t, tmax, temp_tmax, N,
temp_N, temp_persen;
double jarak[110][110], temp_jarak[110][110], fungsi_tujuan[110],
fungsi_tujuan_prey, fungsi_tujuan_baru, total_jarak, temp_pilih;
char file[110];
ofstream tulis;
ifstream input_jarak, input_delivery_pickup;

void input_data(int banyak_pelanggan) {
    for(int i=0; i<banyak_pelanggan+1; i++) {
        for(int j=0; j<2; j++) {
            input_jarak >> temp_jarak[i][j];
        }
        input_delivery_pickup >> temp_delivery_pickup[i];
    }
    for(int i=0; i<banyak_pelanggan+1; i++) {
        for(int j=0; j<banyak_pelanggan+1; j++) {
```

```

        jarak[i][j] = sqrt(pow(temp_jarak[i][0]-temp_jarak[j][0], 2)
+ pow(temp_jarak[i][1]-temp_jarak[j][1], 2));
    }
    rasio = min(temp_jarak[i][0]/temp_jarak[i][1],
temp_jarak[i][1]/temp_jarak[i][0]);
    delivery_pickup[0][i] = round(rasio*temp_delivery_pickup[i]);
    delivery_pickup[1][i] = round((1-
rasio)*temp_delivery_pickup[i]);
    }
}

void inisialisasi_parameter() {
    do {
        cek = 0;
        cout << "Masukkan Jumlah Musang Madu (N >= 2) : "; cin >>
temp_N;
        N = temp_N;
        if(cin.fail()) {
            cin.clear(); cin.ignore(999, '\n'); cek = 1;
        }
        if(temp_N < 2 || cek == 1 || temp_N != round(temp_N)) {
            cout << "Nilai yang Anda masukkan salah, jumlah musang madu
harus bilangan bulat positif lebih dari satu!" << endl;
        }
    } while(temp_N < 2 || cek == 1 || temp_N != round(temp_N));
    do {
        cek = 0;
        cout << "Masukkan Nilai Konstanta (C >= 1) : "; cin >> temp_C;
        C = temp_C;
        if(cin.fail()) {
            cin.clear(); cin.ignore(999, '\n'); cek = 1;
        }
        if(temp_C < 1 || cek == 1 || temp_C != round(temp_C)) {
            cout << "Nilai yang Anda masukkan salah, konstanta harus
bilangan bulat positif!" << endl;
        }
    } while(temp_C < 1 || cek == 1 || temp_C != round(temp_C));
    do {
        cek = 0;
        cout << "Kemampuan Musang Madu Mencari Makanan (beta >= 1) : ";
cin >> temp_beta;
        beta = temp_beta;
        if(cin.fail()) {
            cin.clear(); cin.ignore(999, '\n'); cek = 1;
        }
    }
}

```

```

        if(temp_beta < 1 || cek == 1 || temp_beta != round(temp_beta)) {
            cout << "Nilai yang Anda masukkan salah, kemampuan musang
            madu mencari makanan harus bilangan bulat positif!" << endl;
        }
    } while(temp_beta < 1 || cek == 1 || temp_beta != round(temp_beta));
    do {
        cek = 0;
        cout << "Persen Muatan Awal Pada Kendaraan [50, 70] : "; cin >>
temp_persen;
        persen = temp_persen;
        if(cin.fail()) {
            cin.clear(); cin.ignore(999, '\n'); cek = 1;
        }
        if(temp_persen < 50 || temp_persen > 70 || cek == 1 ||
temp_persen != round(temp_persen)) {
            cout << "Nilai yang Anda masukkan salah, persentase muatan
            awal kendaraan harus berada di interval [50, 70]!" << endl;
        }
    } while(temp_persen < 50 || temp_persen > 70 || cek == 1 ||
temp_persen != round(temp_persen));
    do {
        cek = 0;
        cout << "Batas Iterasi Maksimum (tmax >= 1) : "; cin >>
temp_tmax;
        tmax = temp_tmax;
        if(cin.fail()) {
            cin.clear(); cin.ignore(999, '\n'); cek = 1;
        }
        if(temp_tmax < 1 || cek == 1 || temp_tmax != round(temp_tmax)) {
            cout << "Nilai yang Anda masukkan salah, batas iterasi
            maksimum harus bilangan bulat positif!" << endl;
        }
    } while(temp_tmax < 1 || cek == 1 || temp_tmax != round(temp_tmax));
}

void membangkitkan_populasi_awal(int N, int banyak_pelanggan) {
    for(int i=0; i<N; i++) {
        konter_pelanggan = 1;
        for(int j=0; j<banyak_pelanggan; j++) {
            random[i][j] = (double) rand()/RAND_MAX;
            urut_random[i][j] = 0;
            urut_random[i][j] = random[i][j];
            solusi[i][j] = 0;
            solusi[i][j] = konter_pelanggan;
            konter_pelanggan++;
        }
    }
}

```

```

    }

    temp_random = 0; temp_solusi = 0;
    for(int j=0; j<banyak_pelanggan; j++) {
        for(int k=j+1; k<banyak_pelanggan; k++) {
            if(urut_random[i][k] < urut_random[i][j]) {
                temp_random = urut_random[i][j];
                urut_random[i][j] = urut_random[i][k];
                urut_random[i][k] = temp_random;

                temp_solusi = solusi[i][j];
                solusi[i][j] = solusi[i][k];
                solusi[i][k] = temp_solusi;
            }
        }
    }

    tulis << "Musang madu ke-" << i+1 << " : " << endl;
    tulis << "\tRandom      : \t";
    for(int j=0; j<banyak_pelanggan; j++) {
        tulis << random[i][j] << " ";
    }
    tulis << endl << "\tCalon Rute : \t";
    for(int j=0; j<banyak_pelanggan; j++) {
        tulis << solusi[i][j] << "\t";
    }
    tulis << endl << endl;
}

}

void evaluasi_fungsi_tujuan_awal(int kapasitas, double persen, int
banyak_pelanggan, double jarak[110][110], int delivery_pickup[110][110],
int solusi[110][110]) {
    for(int i=0; i<N; i++) {
        tulis << "Musang madu ke-" << i+1 << " : " << endl;
        m = 0;
        permintaan_pengiriman = kapasitas*persen/100, konter_rute = 0,
        penanda = 0, total_jarak = 0, jarak_tempuh = 0;
        while(solusi[i][m] != solusi[i][banyak_pelanggan]) {
            permintaan_pengiriman -= delivery_pickup[0][solusi[i][m]];
            muatan_tersedia = permintaan_pengiriman +
            delivery_pickup[1][solusi[i][m]];
            sisa_kapasitas = kapasitas - muatan_tersedia;
            berhenti = m;

```

```

        if(permintaan_pengiriman <= 0 || muatan_tersedia > kapasitas
|| sisa_kapasitas <= 0) {
            tulis << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
            for(int k=penanda; k<berhenti; k++) {
                tulis << solusi[i][k] << " - ";
                if(k == penanda) {
                    total_jarak += jarak[0][solusi[i][k]];
                    jarak_tempuh += jarak[0][solusi[i][k]];
                } else if(k == berhenti-1) {
                    total_jarak += jarak[solusi[i][k]][0];
                    jarak_tempuh += jarak[solusi[i][k]][0];
                    total_jarak += jarak[solusi[i][k-
1]][solusi[i][k]];
                    jarak_tempuh += jarak[solusi[i][k-
1]][solusi[i][k]];
                } else {
                    total_jarak += jarak[solusi[i][k-
1]][solusi[i][k]];
                    jarak_tempuh += jarak[solusi[i][k-
1]][solusi[i][k]];
                }
            }
            tulis << "\t| Jarak : " << jarak_tempuh << endl;
            konter_rute++; permintaan_pengiriman =
kapasitas*persen/100; penanda = m; m = berhenti-1; jarak_tempuh = 0;
        }
        m++;
    }

    if(permintaan_pengiriman > 0 || muatan_tersedia < kapasitas ||
sisa_kapasitas > 0 || solusi[i][m] == solusi[i][banyak_pelanggan]) {
        tulis << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
        for(int k=penanda; k<=berhenti; k++) {
            tulis << solusi[i][k] << " - ";
            if(k == penanda) {
                total_jarak += jarak[0][solusi[i][k]];
                jarak_tempuh += jarak[0][solusi[i][k]];
            } else if(k == berhenti) {
                total_jarak += jarak[solusi[i][k]][0];
                jarak_tempuh += jarak[solusi[i][k]][0];
                total_jarak += jarak[solusi[i][k-1]][solusi[i][k]];
                jarak_tempuh += jarak[solusi[i][k-1]][solusi[i][k]];
            } else {
                total_jarak += jarak[solusi[i][k-1]][solusi[i][k]];
                jarak_tempuh += jarak[solusi[i][k-1]][solusi[i][k]];
            }
        }
    }

```

```

        }
    }
    tulis << "0\t| Jarak : " << jarak_tempuh << endl;
}
tulis << "Dengan total jarak : " << total_jarak << endl << endl;
fungsi_tujuan[i] = total_jarak;
}
}

void solusi_terbaik(int N, double fungsi_tujuan[110], int
banyak_pelanggan, double random[110][110], int solusi[110][110]) {
    xprey = 0;
    for(int i=0; i<N; i++) {
        if(fungsi_tujuan[i] < fungsi_tujuan[xprey]) {
            xprey = i;
        }
    }
    for(int i=0; i<banyak_pelanggan; i++) {
        random_prey[i] = random[xprey][i];
        solusi_prey[0][i] = solusi[xprey][i];
    }
    fungsi_tujuan_prey = fungsi_tujuan[xprey];
}

double faktor_kepadatan(double C, double t, double tmax) {

    alpha = C*exp(-t/tmax);

    return alpha;
}

void intensitas(int n, int N, int banyak_pelanggan, double
random[110][110], double random_prey[110]) {
    for(int k = 0; k < banyak_pelanggan; k++) {
        S[k] = 0; d[k] = 0; I[k] = 0; r2[k] = 0;
    }
    musang_i = n;
    if(n == N-1) {
        musang_j = 0;
    } else {
        musang_j = n+1;
    }
    for(int k = 0; k < banyak_pelanggan; k++) {
        r2[k] = (double) rand()/RAND_MAX;
        S[k] = pow(random[musang_i][k]-random[musang_j][k], 2);
    }
}

```

```

        d[k] = random_prey[k]-random[musang_i][k];
        I[k] = r2[k]*(S[k]/(4*M_PI*pow(d[k], 2)));
    }
}

void fase_menggali(int n, int banyak_pelanggan, double beta, double
alpha, double I[110], double d[110], double random_prey[110], double
random[110][110]) {
    for(int k = 0; k < banyak_pelanggan; k++) {
        solusi[n][k] = 0; r3[k] = 0; r4[k] = 0; r5[k] = 0;
        random_baru[n][k] = 0; urut_random[n][k] = 0;
    }
    r6 = (double) rand()/RAND_MAX;
    if(r6 <= 0.5) {
        F = 1;
    } else {
        F = -1;
    }
    konter_pelanggan = 1;
    for(int k=0; k<banyak_pelanggan; k++) {
        solusi[n][k] = konter_pelanggan;
        konter_pelanggan++;
        r3[k] = (double) rand()/RAND_MAX;
        r4[k] = (double) rand()/RAND_MAX;
        r5[k] = (double) rand()/RAND_MAX;
        random_baru[n][k] =
        random_prey[k]+(F*beta*I[k]*random[n][k])+(F*r3[k]*alpha*d[k]*abs(cos(2*
        M_PI*r4[k]))*(1-cos(2*M_PI*r5[k]))));
        urut_random[n][k] = random_baru[n][k];
    }
    temp_random = 0; temp_solusi = 0;
    for(int i = 0; i < banyak_pelanggan; i++) {
        for(int j = i+1; j < banyak_pelanggan; j++) {
            if(urut_random[n][j] < urut_random[n][i]) {
                temp_random = urut_random[n][i];
                urut_random[n][i] = urut_random[n][j];
                urut_random[n][j] = temp_random;

                temp_solusi = solusi[n][i];
                solusi[n][i] = solusi[n][j];
                solusi[n][j] = temp_solusi;
            }
        }
    }
}

```

```

void fase_madu(int n, int banyak_pelanggan, double alpha, double d[110],
double random_prey[110]) {
    for(int k = 0; k < banyak_pelanggan; k++) {
        solusi[n][k] = 0; r7[k] = 0; random_baru[n][k] = 0;
    }
    urut_random[n][k] = 0;
    r6 = (double) rand()/RAND_MAX;
    if(r6 <= 0.5) {
        F = 1;
    } else {
        F = -1;
    }
    konter_pelanggan = 1;
    for(int k = 0; k < banyak_pelanggan; k++) {
        solusi[n][k] = konter_pelanggan;
        konter_pelanggan++;
        r7[k] = (double) rand()/RAND_MAX;
        random_baru[n][k] = random_prey[k] + (F*r7[k]*alpha*d[k]);
        urut_random[n][k] = random_baru[n][k];
    }
    temp_random = 0; temp_solusi = 0;
    for(int i = 0; i < banyak_pelanggan; i++) {
        for(int j = i+1; j < banyak_pelanggan; j++) {
            if(urut_random[n][j] < urut_random[n][i]) {
                temp_random = urut_random[n][i];
                urut_random[n][i] = urut_random[n][j];
                urut_random[n][j] = temp_random;

                urut_solusi = solusi[n][i];
                solusi[n][i] = solusi[n][j];
                solusi[n][j] = urut_solusi;
            }
        }
    }
}

```

```

int evaluasi_fungsi_tujuan(int akhir, int n, int kapasitas, double
persen, int banyak_pelanggan, double jarak[110][110], int
delivery_pickup[110][110], int solusi[110][110]) {
    m = 0;
    permintaan_pengiriman = kapasitas*persen/100, konter_rute = 0,
penanda = 0, total_jarak = 0, jarak_tempuh = 0;
    while(solusi[n][m] != solusi[n][banyak_pelanggan]) {
        permintaan_pengiriman -= delivery_pickup[0][solusi[n][m]];
    }
}

```

```

        muatan_tersedia = permintaan_pengiriman +
delivery_pickup[1][solusi[n][m]];
        sisa_kapasitas = kapasitas - muatan_tersedia;
        berhenti = m;
        if(permintaan_pengiriman <= 0 || muatan_tersedia > kapasitas ||
sisa_kapasitas <= 0) {
            if(akhir == 1) {
                cout << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
            }
            tulis << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
            for(int k=penanda; k<berhenti; k++) {
                if(akhir == 1) {
                    cout << solusi[n][k] << " - ";
                }
                tulis << solusi[n][k] << " - ";
                if(k == penanda) {
                    total_jarak += jarak[0][solusi[n][k]];
                    jarak_tempuh += jarak[0][solusi[n][k]];
                } else if(k == berhenti-1) {
                    total_jarak += jarak[solusi[n][k]][0];
                    jarak_tempuh += jarak[solusi[n][k]][0];
                    total_jarak += jarak[solusi[n][k-1]][solusi[n][k]];
                    jarak_tempuh += jarak[solusi[n][k-1]][solusi[n][k]];
                } else {
                    total_jarak += jarak[solusi[n][k-1]][solusi[n][k]];
                    jarak_tempuh += jarak[solusi[n][k-1]][solusi[n][k]];
                }
            }
            if(akhir == 1) {
                cout << "0\t| Jarak : " << jarak_tempuh << endl;
            }
            tulis << "0\t| Jarak : " << jarak_tempuh << endl;
            konter_rute++; permintaan_pengiriman = kapasitas*persen/100;
penanda = m; m = berhenti-1; jarak_tempuh = 0;
        }
        m++;
    }
    if(permintaan_pengiriman > 0 || muatan_tersedia < kapasitas ||
sisa_kapasitas > 0 || solusi[n][m] == solusi[n][banyak_pelanggan]) {
        if(akhir == 1) {
            cout << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
        }
        tulis << "\tRute ke-" << konter_rute+1 << " :\t0 - ";
        for(int k=penanda; k<=berhenti; k++) {
            if(akhir == 1) {

```

```

        cout << solusi[n][k] << " - ";
    }
    tulis << solusi[n][k] << " - ";
    if(k == penanda) {
        total_jarak += jarak[0][solusi[n][k]];
        jarak_tempuh += jarak[0][solusi[n][k]];
    } else if(k == berhenti) {
        total_jarak += jarak[solusi[n][k]][0];
        jarak_tempuh += jarak[solusi[n][k]][0];
        total_jarak += jarak[solusi[n][k-1]][solusi[n][k]];
        jarak_tempuh += jarak[solusi[n][k-1]][solusi[n][k]];
    } else {
        total_jarak += jarak[solusi[n][k-1]][solusi[n][k]];
        jarak_tempuh += jarak[solusi[n][k-1]][solusi[n][k]];
    }
}
if(akhir == 1) {
    cout << "0\t| Jarak : " << jarak_tempuh << endl;
}
tulis << "0\t| Jarak : " << jarak_tempuh << endl;
}
if(akhir == 1) {
    cout << "Dengan total jarak : " << total_jarak << endl << endl;
}
tulis << "Dengan total jarak : " << total_jarak << endl << endl;
fungsi_tujuan_baru = total_jarak;

return fungsi_tujuan_baru;
}

void update_fungsi_tujuan(int n, double fungsi_tujuan_baru, double
fungsi_tujuan[110], int banyak_pelanggan, double random_baru[110][110])
{
    if(fungsi_tujuan_baru <= fungsi_tujuan[n]) {
        tulis << "Karena fungsi tujuan baru bernilai " <<
fungsi_tujuan_baru << " sedangkan fungsi tujuan lama bernilai " <<
fungsi_tujuan[n] << ", maka solusi lama diperbarui" << endl << endl;
        fungsi_tujuan[n] = fungsi_tujuan_baru;
        for(int l=0; l<banyak_pelanggan; l++) {
            random[n][l] = 0;
            random[n][l] = random_baru[n][l];
        }
    } else {

```

```

        tulis << "Karena fungsi tujuan baru bernilai " <<
fungsi_tujuan_baru << " sedangkan fungsi tujuan lama bernilai " <<
fungsi_tujuan[n] << ", maka solusi lama tetap" << endl << endl;
    }
}

void update_fungsi_tujuan_terbaik(int n, double fungsi_tujuan_baru,
double fungsi_tujuan_prey, int banyak_pelanggan, double
random_baru[110][110], int solusi[110][110]) {
    if(fungsi_tujuan_baru <= fungsi_tujuan_prey) {
        tulis << "Karena fungsi tujuan baru bernilai " <<
fungsi_tujuan_baru << " sedangkan fungsi tujuan mangsa bernilai " <<
fungsi_tujuan_prey << ", maka solusi terbaik/mangsa diperbarui" << endl
<< endl;
        xprey = n; fungsi_tujuan_prey = fungsi_tujuan_baru;
        for(int l=0; l<banyak_pelanggan; l++) {
            random_prey[l] = 0;
            random_prey[l] = random_baru[n][l];
            solusi_prey[0][l] = 0;
            solusi_prey[0][l] = solusi[n][l];
        }
    } else {
        tulis << "Karena fungsi tujuan baru bernilai " <<
fungsi_tujuan_baru << " sedangkan fungsi tujuan mangsa bernilai " <<
fungsi_tujuan_prey << ", maka solusi terbaik/mangsa tetap" << endl <<
endl;
    }
}

int main() {

    system("cls");
    srand(time(NULL));

    cout << "-----
-----" << endl;
    cout << "| Implementasi Honey Badger Algorithm (HBA) dalam
Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and
Delivery (VRPSPD) |" << endl;
    cout <<
"|
Abistha
endl;
Reza Abid
|" <<

```

```

        cout <<
        "|                                08191123
3079                                |" <<
endl;
        cout <<
        "|                                Universitas
Airlangga                                |" <<
endl;
        cout << "-----
-----" << endl << endl;

        cout << "Pilihan Data:" << endl << endl;
        cout << "1. Data Kecil  (13 Pelanggan)" << endl;
        cout << "2. Data Sedang (75 Pelanggan)" << endl;
        cout << "3. Data Besar  (100 Pelanggan)" << endl << endl;

        do {
            cek = 0;
            cout << "Pilihan Data Anda (1/2/3): "; cin >> temp_pilih;
            pilih = temp_pilih;
            if(cin.fail()) {
                cin.clear(); cin.ignore(999, '\n'); cek = 1;
            }
            if(temp_pilih < 1 || temp_pilih > 3 || cek == 1 || temp_pilih !=
round(temp_pilih)) {
                cout << "Pilihan yang Anda masukkan salah, silakan pilih
1/2/3!" << endl;
            }
        } while(temp_pilih < 1 || temp_pilih > 3 || cek == 1 || temp_pilih
!= round(temp_pilih));

        switch(pilih) {
            case 1:
                input_jarak.open("jarak_kecil.txt");
input_delivery_pickup.open("dp_kecil.txt");
                banyak_pelanggan = 13; kapasitas = 100;
                break;
            case 2:
                input_jarak.open("jarak_sedang.txt");
input_delivery_pickup.open("dp_sedang.txt");
                banyak_pelanggan = 75; kapasitas = 140;
                break;
            case 3:

```

```

        input_jarak.open("jarak_besar.txt");
input_delivery_pickup.open("dp_besar.txt");
        banyak_pelanggan = 100; kapasitas = 200;
        break;
    default:
        break;
}

input_data(banyak_pelanggan);

    cout << endl << "-----"
<< endl;
    cout << "|  Masukkan Parameter Honey Badger Algorithm !  |" << endl;
    cout << "-----" << endl
<< endl;

    inisialisasi_parameter();

    cout << endl << "-----"
-----" << endl;
    cout << "|  Masukkan Nama File Perhitungan yang Akan Disimpan !  |"
<< endl;
    cout << "-----"
<< endl << endl;

    cout << "Nama file : "; cin >> file; strcat(file, ".txt");
    tulis.open(file);

    tulis << "-----"
-----" << endl;
    tulis << "|  Implementasi Honey Badger Algorithm (HBA) dalam
Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and
Delivery (VRPSPD)  |" << endl;
    tulis <<
"|
Abistha
endl;
    tulis <<
"|
3079
endl;
    tulis <<
"|
Reza Abid
|" <<
08191123
|" <<
Universitas

```

```

Airlangga |" <<
endl;
    tulis << "-----"
-----
-----" << endl << endl;

    tulis << "-----" << endl;
    tulis << "    Data Jarak    " << endl;
    tulis << "-----" << endl << endl;

    for(int i = 0; i < banyak_pelanggan+1; i++) {
        for(int j = 0; j < banyak_pelanggan+1; j++) {
            tulis << jarak[i][j] << "\t";
        }
        tulis << endl;
    }

    tulis << endl << "-----" << endl;
    tulis << "    Data Delivery dan Pickup    " << endl;
    tulis << "-----" << endl << endl;

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < banyak_pelanggan+1; j++) {
            tulis << delivery_pickup[i][j] << "\t";
        }
        tulis << endl;
    }

    tulis << endl << "-----" << endl;
    tulis << "    Populasi Awal / Solusi Awal    " << endl;
    tulis << "-----" << endl << endl;

    membangkitkan_populasi_awal(N, banyak_pelanggan);

    tulis << "-----" << endl;
    tulis << "    Evaluasi Solusi Awal    " << endl;
    tulis << "-----" << endl << endl;

    evaluasi_fungsi_tujuan_awal(kapasitas, persen, banyak_pelanggan,
jarak, delivery_pickup, solusi);

    cout << endl << "-----" << endl;
    cout << "    Solusi Terbaik Sementara    " << endl;
    cout << "-----" << endl << endl;

```

```

    tulis << "-----" << endl;
    tulis << "    Solusi Terbaik Sementara    " << endl;
    tulis << "-----" << endl << endl;

    solusi_terbaik(N, fungsi_tujuan, banyak_pelanggan, random, solusi);
    akhir = 1;
    cout << "Solusi terbaik sementara (xprey) adalah musang madu ke-" <<
xprey+1 << endl << endl;
    cout << "Rute distribusi :" << endl;

    tulis << "Solusi terbaik sementara (xprey) adalah musang madu ke-"
<< xprey+1 << endl << endl;
    tulis << "Rute distribusi :" << endl;
    fungsi_tujuan_pre = evaluasi_fungsi_tujuan(akhir, 0, kapasitas,
persen, banyak_pelanggan, jarak, delivery_pickup, solusi_pre);

    t = 1;
    t1 = 1;
    while(t <= tmax) {
        akhir = 0;
        cout << "-----
-----" << endl;
        cout <<
"|                                                    ITERAS
I " << t <<
"                                                    |" <<
endl;
        cout << "-----
-----" << endl << endl;

        tulis << "-----
-----" << endl;
        tulis <<
"|                                                    ITERAS
I " << t <<
"                                                    |" <<
endl;
        tulis << "-----
-----" << endl << endl;

        alpha = faktor_kepadatan(C, t, tmax);

```

```

n = 0;
do {
    if(n != xprey) {
        intensitas(n, N, banyak_pelanggan, random, random_prey);

        tulis << "-----" <<
endl;
        tulis << "    Penentuan Metode Pembaruan Solusi    " <<
endl;
        tulis << "-----" <<
endl << endl;

        r = (double) rand()/RAND_MAX;

        if(r < 0.5) {
            tulis << "Karena nilai r adalah sebesar " << r << ",
maka Musang madu/solusi diperbarui menggunakan fase menggali" << endl <<
endl;
            fase_menggali(n, banyak_pelanggan, beta, alpha, I,
d, random_prey, random);
        } else {
            tulis << "Karena nilai r adalah sebesar " << r << ",
maka Musang madu/solusi diperbarui menggunakan fase madu" << endl <<
endl;
            fase_madu(n, banyak_pelanggan, alpha, d,
random_prey);
        }

        tulis << "-----" << endl;
        tulis << "    Solusi yang Telah Diperbarui    " << endl;
        tulis << "-----" << endl <<
endl;

        tulis << "Musang madu ke-" << n+1 << " :" << endl;
        tulis << "\tRandom      : \t";
        for(int i = 0; i < banyak_pelanggan; i++) {
            tulis << random_baru[n][i] << " ";
        }
        tulis << endl << "\tCalon Rute : \t";
        for(int i = 0; i < banyak_pelanggan; i++) {
            tulis << solusi[n][i] << "\t";
        }
        tulis << endl << endl;
    }
}

```



```

        akhir = 1;
        fungsi_tujuan_prej = evaluasi_fungsi_tujuan(akhir, 0, kapasitas,
persen, banyak_pelanggan, jarak, delivery_pickup, solusi_prej);

        for(int i=0; i<N; i++) {
            for(int j=0; j<banyak_pelanggan; j++) {
                random[i][j] = random[i][j] * 1e-300 * 1e-300 * 1e-300;
            }
        }

        t++;
    }

    cout << "-----" << endl;
    cout << "    Solusi Terbaik    " << endl;
    cout << "-----" << endl << endl;

    tulis << "-----" << endl;
    tulis << "    Solusi Terbaik    " << endl;
    tulis << "-----" << endl << endl;

    akhir = 1;
    cout << "Solusi terbaik (xprej) adalah musang madu ke-" << xprej+1
<< endl << endl;
    cout << "Rute distribusi :" << endl;

    tulis << "Solusi terbaik (xprej) adalah musang madu ke-" << xprej+1
<< endl << endl;
    tulis << "Rute distribusi :" << endl;

    fungsi_tujuan_prej = evaluasi_fungsi_tujuan(akhir, 0, kapasitas,
persen, banyak_pelanggan, jarak, delivery_pickup, solusi_prej);

    input_jarak.close(); input_delivery_pickup.close();
    return 0;
}

```

Lampiran 11 Hasil *Running* Program Berdasarkan Persentase Muatan Awal pada Kendaraan

Hasil *Running* Program untuk Data Kecil

Maksimal Iterasi	Jumlah Musang Madu	Persentase Kapasitas (%)		
		50	60	70
10	10	316.345	293.639	296.191
	50	285.611	289.972	298.563
	100	268.432	290.496	297.078
100	10	287.722	268.04	274.621
	50	266.172	267.306	250.845
	100	262.866	266.659	252.835
1000	10	255.15	265.051	250.586
	50	247.175	258.907	238.348
	100	247.175	252.753	238.348

Hasil *Running* Program untuk Data Sedang

Maksimal Iterasi	Jumlah Musang Madu	Persentase Kapasitas (%)		
		50	60	70
10	10	2353.7	2268.63	2257.48
	50	2308.83	2198.91	2219.85
	100	2321.07	2233.18	2210
100	10	2206.94	2154.33	2169.36
	50	2148.96	2107.58	2088.54
	100	2106.49	2087.75	2000.74
1000	10	2060.49	2016.95	2022.16
	50	2007.89	1950.56	1885.45
	100	1964.65	2005.15	1860.24

Hasil *Running* Program untuk Data Besar

Maksimal Iterasi	Jumlah Musang Madu	Persentase Kapasitas (%)		
		50	60	70
10	10	3121.99	3089.43	3088.87
	50	3057.29	3005.73	3012.2
	100	3047.22	2997.28	2993.88
100	10	2885.56	2824.85	2928.74
	50	2873.32	2814.3	2864.66
	100	2866.9	2856.43	2667.75
1000	10	2759.73	2779.24	2682.1
	50	2744.91	2805.42	2637.78
	100	2750.09	2622.98	2632.72