OPTIMASI RUTE *MULTIPLE TRAVELLING SALESMAN PROBLEM* PADA DISTRIBUSI ES BATU DENGAN ALGORITME *ARTIFICIAL BEE COLONY* (ABC)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Aghni Nur Lazuardy
NIM: 145150200111054

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

Optimasi Rute *Multiple Travelling Salesman Problem* Pada Distribusi Es Batu Dengan Algoritme *Artificial Bee Colony* (ABC)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Aghni Nur Lazuardy
NIM: 145150200111054

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Muhammad Tauzil Furqon, S.Kom, M.CompSc
NIK: 2012018507191001	NIP: 198209302008011004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 02 Agustus 2018

Muhammad Aghni Nur Lazuardy

NIM: 145150200111054

KATA PENGANTAR

Bismillahirrahmanirrahim. Segala Puji bagi Allah SWT. Tuhan semesta alam, atas limpahan rahmat dan kasih sayang-Nya sehingga penulis dapat menyelesaikan penulisan skripsi ini. Shalawat serta salam tidak lupa saya panjatkan kepada Rasulullah Muhammad SAW. beserta sahabat, keluarga, dan umat beliau semoga senantiasa diberikan keselamatan hingga akhir zaman.

Skripsi ini disusun sebagai salah satu syarat memperoleh gelar Sarjana Komputer pada Fakultas Ilmu Komputer Universitas Brawijaya dengan judul **"OPTIMASI RUTE *MULTIPLE TRAVELLING SALESMAN PROBLEM* PADA DISTRIBUSI ES BATU DENGAN ALGORITME *ARTIFICIAL BEE COLONY* (ABC)".**

Penulis menyadari bahwa tugas akhir ini dapat terselesaikan berkat bantuan, petunjuk, bimbingan, dan dukungan dari berbagai pihak yang telah banak membantu proses penyelesaian tugas akhir ini. Oleh karena itu penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Imam Cholissodin, S.Si, M.Kom selaku Pembimbing I yang telah bersedia meluangkan waktu, membimbing, dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini dengan baik.
2. Muhammad Tanzil Furqon, S.Kom, M.CompSc selaku Pembimbing II yang telah meluangkan waktu dan mengarahkan penulis pada proses penyusunan skripsi sehingga dapat menyelesaikan skripsi ini dengan baik.
3. PT. Jatim Es Tube-Malang selaku sumber penyedia data pada penelitian optimasi rute distribusi es batu.
4. Wayan Firdaus Mahmudy, S.Si, M.T, P.hD. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
5. Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
6. Agus Wahyu Widodo S.T, M.Cs selaku Ketua Program Studi Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
7. Orang tua penulis, Bapak Nunung Nurmudin dan Ibu Yuniarti Iriani tercinta, terima kasih atas segala doa, kasih sayang, semangat, pengorbanan, dan ketulusannya dalam memberikan dukungan terbaik kepada penulis. Semoga Allah SWT. senantiasa memberikan rahmat dan ridha-Nya kepada keduanya.
8. Saudara penulis, Amelia Sartikarani beserta keluarga yang telah memberikan motivasi dan semangat serta menjadikan tempat beristirahat dan melepas penat pada saat pengerjaan skripsi hingga selesai.
9. Seluruh dosen Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.

10. Anggi Maulina Saputri, partner penulis terima kasih atas kesabaran dan kesediaan dalam memberikan semangat kepada penulis sehingga bisa terselesaikan skripsi ini.
11. Teman-teman pengurus BIOS EXALT angkatan 2014 yang sama-sama sedang mengerjakan skripsi dan telah setia menemai serta menghibur dikala penat dalam pengerjaan skripsi.
12. Falih Gozi Febrinanto, Nirmala Fa'izah Saraswati, dan Muhammad Isradi Azhar teman-teman yang telah meluangkan waktu untuk bertukar fikiran dalam pengerjaan skripsi.
13. Teman-teman kelas Informatika D yang telah memberi motivasi dan semangat kepada penulis dalam proses pengerjaan skripsi baik secara langsung maupun tidak.
14. Seluruh *civitas academica* Fakultas Ilmu Komputer Universitas Brawijaya
15. Semua pihak yang tidak bisa disebutkan satu per satu. Terima kasih atas segala bantuan baik secara secara langsung atau tidak.

Penulis sadar bahwa skripsi ini masih banyak kekurangan, oleh karena itu kritik dan saran yang membangun sangat diharapkan penulis. Penulis berharap skripsi ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, 15 Juli 2018

Penulis

muhammadaghni@gmail.com

ABSTRAK

Muhammad Aghni Nur Lazuardy, Optimasi Rute Multiple Travelling Salesman Problem Pada Distribusi Es Batu Dengan Algoritme Artificial Bee Colony (ABC)

Dosen Pembimbing: Imam Cholissodin, S.Si, M.Kom dan Muhammad Tanzil Furqon, S.Kom, M.CompSc

Proses distribusi dilakukan untuk meningkatkan produktivitas perusahaan. Sebuah strategi dalam sebuah proses distribusi diperlukan terutama dalam penentuan rute distribusi. Sebuah rute yang optimal sangat penting dalam distribusi produk terutama es batu. Sebuah perusahaan perlu mengirimkan produknya ke banyak alamat, oleh karena banyaknya alamat pengiriman dan dengan jarak yang bervariasi memunculkan sebuah masalah seperti waktu yang lama untuk sampai ke tempat tujuan. Dalam memecahkan masalah tersebut perlu sebuah sistem yang memiliki tujuan untuk membantu proses distribusi dengan jumlah *sales* lebih dari satu, permasalahan tersebut disebut dengan *Multiple Travelling Salesman Problem* (M-TSP). Metode yang dapat menyelesaikan masalah M-TSP salah satunya adalah dengan algoritme *Artificial Bee Colony* (ABC) dibandingkan dengan algoritme lain berdasarkan pada kecerdasan berkelompok. Proses awal algoritme ABC mencari rute pengiriman es batu secara *random* berdasarkan data pelanggan yang sudah memesan. Selanjutnya dilakukan *swapping* dan *insertion* rute tersebut kemudian diambil rute dengan *fitness* optimal. Terakhir adalah dilakukan perbandingan dengan rute awal apakah hasilnya lebih baik atau tidak. Hasil pengujian menunjukkan parameter optimal yaitu jumlah *size problem* 23, jumlah *pop size* 80, jumlah *limit* 10, dan banyak iterasi 600. Dari parameter tersebut didapatkan rata-rata nilai *fitness* berdasarkan optimasi sistem sebesar 0,078163 dan untuk pemilihan jalur secara manual yang dilalui *sales* mendapatkan rata-rata nilai *fitness* sebesar 0,043472, sehingga pemilihan jalur dapat dioptimasi sistem.

Kata kunci: artificial bee colony (ABC), optimasi, distribusi, multiple travelling salesman problem (M-TSP)

ABSTRACT

Muhammad Aghni Nur Lazuardy, Multiple Travelling Salesman Problem Route Optimization of Ice Cube Distribution Using Artificial Bee Colony (ABC) Algorithm

Supervisors: Imam Cholissodin, S.Si, M.Kom dan Muhammad Tanzil Furqon, S.Kom, M.CompSc

The distribution is done to improving the productivity of the company. A strategy in the process of distribution is required primarily in determining the distribution route. An optimal route is essential in product distribution especially ice cubes. A company needs to send its products to multiple addresses, because the numbers of shipping addresses and varying distances creates a problem such as needing a long time to reach the destination. In solving these problem need a system that has a purpose to help the distribution process with the number of sales more than one, the problem is named Multiple Travelling Salesman Problem (M-TSP). One of the methods that can solve the problem of M-TSP is Artificial Bee Colony (ABC) algorithm which compared to other algorithm based on swarm intelligence. The initial process of ABC algorithm looks for random ice cubes distribution routes based on customer's ordering data. Furthermore swapping and insertion route is done then taken the route with optimal fitness. The last is comparison with the initial route whether the result is better or not. The test result show the numbers of optimal parameters are 23 size problems, 80 pop sizes, 10 limits, and 600 iterations. From these parameters obtained average fitness value based on system optimization of 0,078163 and manual selection of routes the sales goes through obtain average fitness value of 0,043472, with the result that path selection can be optimized by system.

Keywords: artificial bee colony (ABC), optimization, distribution, multiple travelling salesman problem (M-TSP)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Es batu adalah sebuah komoditas yang sering digunakan sebagai suatu cara untuk mengawetkan bahan makanan seperti ikan, daging, serta digunakan juga sebagai pendingin minuman. Es batu sendiri merupakan air yang didinginkan terus menurus dengan suhu di bawah 0 °C sehingga air tersebut menjadi beku (Irawan, 2015). Letak geografis Indonesia yang memiliki iklim tropis menyebabkan banyak masyarakat yang membutuhkan es batu.

Dalam memenuhi kebutuhan masyarakat terhadap es batu, proses distribusi sangat penting karena merupakan proses untuk menyampaikan barang dari perusahaan kepada konsumen. Masalah-masalah yang dihadapi pada proses pendistribusian es batu tersebut antara lain adalah sifat dari es batu yang mudah mencair karena suhu Indonesia yang beriklim tropis sehingga menyebabkan suhu tinggi. Sehingga dibutuhkan rute atau jalur tercepat untuk pengirimannya. Selain sifat dari es batu yang mudah mencair, masalah-masalah lain yang juga sering dihadapi dalam pendistribusian adalah jarak tujuan pengiriman es batu yang jauh. Masalah selanjutnya adalah sopir atau *salesman* yang bertugas tidak mengetahui kondisi jalan yang terkadang harus mengambil rute yang lebih jauh sehingga biaya untuk bahan bakar kendaraan bertambah. Terakhir adalah agar pelanggan tidak menunggu terlalu lama saat menunggu barang yang dikirimkan sampai sehingga perlu rute tercepat untuk melakukan pengiriman agar pelanggan senang dan perusahaan tidak mengalami kerugian.

Oleh karena itu diperlukan strategi distribusi yang tepat untuk menyalurkan barang agar barang yang didistribusikan dapat berjalan lancar dan baik agar perusahaan tersebut tidak mengalami kerugian. Saluran distribusi barang adalah bagian terpenting, karena perusahaan perlu menyalurkan barang hasil produksi agar dapat sampai kepada pelanggan dengan tepat dan cepat. Dalam perjalanannya rute yang dilewati untuk mendistribusikan barang terkadang kurang optimal. Dari permasalahan tersebut diperlukan suatu penyelesaian masalah dengan salah satunya adalah *Travelling Salesman Problem* (TSP) dalam penentuan rute yang optimal.

Travelling Salesman Problem (TSP) adalah sebuah masalah yang berkaitan dengan proses melakukan distribusi barang untuk menentukan rute perjalanan dari pusat distribusi yang dilakukan oleh seorang *sales*. Jika *sales* yang ada di pusat distribusi lebih dari satu orang, maka dibutuhkan pengembangan dari TSP yaitu *Multiple Travelling Salesman Problem* (MTSP). MTSP digunakan sebagai representasi dari masalah tersebut karena fungsinya yaitu untuk memilih rute tercepat yang memiliki *salesman* lebih dari satu orang.

Agar proses distribusi barang seperti es batu dapat teroptimasi perlu algoritme yang tepat dalam menentukan rute terpendek dengan solusi terbaik. Salah satu algoritme yang dapat menyelesaikan masalah penentuan rute terpendek adalah *Artificial Bee Colony* (ABC). Algoritme *Artificial Bee Colony*

dipilih karena memiliki kemampuan untuk keluar dari *local minimum* dan dapat secara efisien digunakan untuk multimodal dan *multivariable* optimasi fungsi (Karaboga & Basturk, 2007).

Penelitian sebelumnya mengenai algoritme *Artificial Bee Colony* (ABC) sudah pernah dimanfaatkan oleh Wong, Malcolm, dan Chong (2008) untuk *Travelling Salesman Problem* dengan mendapatkan hasil bahwa hasil algoritme ABC dapat ditingkatkan hasil optimasinya dengan menggunakan teknik *neighbourhood operator* serta penelitian oleh Hermawan, Hidayat, dan Setiawan (2017) tentang optimasi terhadap rute tempat wisata kuliner di Malang menggunakan Algoritme *Bee Colony* dan juga penelitian oleh Roudhotus (2015) yang membahas optimasi rute tempat wisata di Malang menggunakan algoritme *Bee Colony* yang memiliki kesimpulan bahwa Algoritme *Artificial Bee Colony* dapat bekerja secara efektif dengan proses yang tidak banyak memakan waktu untuk mendapat nilai optimal. Pada penelitian lain yang dilakukan oleh Xue, Wang, dan Mao (2016) mendapatkan hasil bahwa algoritme ABC lebih baik digunakan untuk penggunaan MTSP dibandingkan dengan algoritme lain seperti *Ant Colony Optimization* (ACO) dan *Genetic Algorithm.*

Berdasarkan latar belakang tersebut, penulis melakukan penelitian dengan judul "Optimasi Rute *Multiple Travelling Salesman Problem* Pada Distribusi Es Batu Dengan Algortim *Artificial Bee Colony* (ABC)". Penulis berharap hasil dari penelitian ini dapat digunakan sebagai solusi dan bahan pertimbangan bagi perusahaan dalam melakukan proses distribusi yang lebih efektif dan optimal sehingga mendapatkan keuntungan yang maksimal.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas, maka didapatkan hasil rumusan masalah berikut:

1. Bagaimana hasil perbandingan optimai sistem dengan kondisi aktual di lapangan?
2. Bagaimana nilai *fitness* solusi algoritme *Artificial Bee Colony* (ABC) terhadap hasil optimasi pendistribusian es batu dengan menggunakan parameter optimal?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini berdasarkan rumusan masalah di atas adalah:

1. Mendapatkan hasil perbandingan optimai sistem dengan kondisi aktual di lapangan.
2. Mendapatkan nilai *fitness* solusi algoritme *Artificial Bee Colony* (ABC) terhadap hasil optimasi pendistribusian es batu dengan menggunakan parameter optimal.

1.4 Manfaat

Hasil penelitian ini diharapkan dapat memberikan manfaat bagi berbagai pihak. Manfaat yang diharapkan didapat dari penelitian ini adalah dapat dijadikan sebagai bahan pertimbangan dalam membuat kebijakan penjualan, masukan, evaluasi dalam mengelola sumber daya yang dimiliki oleh perusahaan sehingga perusahaan akan mendapat keuntungan yang maksimal. Manfaat yang lainnya adalah dapat dijadikan landasan bagi penelitian selanjutnya yang memiliki masalah serupa.

1.5 Batasan masalah

Batasan-batasan berikut dibutuhkan agar pembahasan pada penelitian ini tidak menyimpang dari yang telah dirumuskan:

1. Data yang digunakan adalah data alamat pelanggan dan data jarak berdasarkan survei langsung dan Google Maps.
2. Data pelanggan yang digunakan adalah pelanggan tetap.
3. Proses distribusi yang dilakukan hanya distribusi yang berada di dalam kota Malang.
4. Tempat keberangkatan *sales* dari pusat distribusi dan berakhir ke pusat distribusi.
5. Perusahaan diasumsikan memiliki kendaraan yang bermuatan lebih besar sama dengan pesanan pelanggan.
6. Pengiriman barang diasumsikan hanya dilakukan sekali pada tiap alamat pada waktu yang sama sehingga tidak ada pengulangan pengiriman.
7. Jarak *sales* dari lokasi terakhir pengiriman kembali ke pusat distribusi diabaikan, karena hanya membahas pengiriman barang ke pelanggan.
8. Kondisi jalan dan waktu yang dibutuhkan tidak dimasukkan sebagai parameter.

1.6 Sistematika pembahasan

Sistematika pembahasan pada penelitian ini adalah sebagai berikut;

BAB 1 : PENDAHULUAN

Bab ini dijelaskan mengenai uraian dari latar belakang masalah, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan.

BAB 2 : LANDASAN KEPUSTAKAAN

Bab ini dijelaskan mengenai kajian pustaka dan dasar teori yang dijadikan acuan dalam penulisan skripsi yang berasal dari beberapa pustaka seperti jurnal dan halaman *website* yang berkaitan dengan masalah optimasi rute distribusi es batu dengan *Artificial Bee Colony* (ABC).

BAB 3 : METODOLOGI

Bab ini berisi langkah-langkah yang dilakukan dalam menulis laporan penelitian yang diawali dengan studi pustaka, analisis kebutuhan, pengambilan data, perancangan, implementasi, pengujian dan analisis, serta pengambilan keputusan.

BAB 4 : PERANCANGAN

Bab ini berisi perancangan yang dilakukan dalam membuat sistem untuk permasalahan optimasi rute distribusi es batu dengan *Artificial Bee Colony* (ABC).

BAB 5 : IMPLEMENTASI

Bab ini berisi penjelasan mengenai teknis implementasi, batasan-batasan implementasi, dan metode operasi yang digunakan untuk mengembangkan sistem.

BAB 6 : PENGUJIAN DAN ANALISIS

Bab ini dijelaskan proses dan hasil pengujian metode pada sistem yang sudah dibuat dan sesuai dengan perancangan sistem yang dilengkapi dengan analisis terhadap metode yang digunakan.

BAB 7 : PENUTUP

Bab ini berisi kesimpulan dari hasil penelitian dan saran atas kekurangan pada penelitian sehingga ke depannya dapat dilakukan pengembangan menjadi penelitian yang lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini dijelaskan mengenai kajian pustaka dan dasar teori yang berkaitan dengan optimasi rute distribusi es batu menggunakan algoritme *Artifical Bee Colony* (ABC). Dasar teori yang diuraikan pada bab ini adalah optimasi, algoritme *Artifical Bee Colony* (ABC), dan *Multiple Travelling Salesman Probelam* (MTSP).

2.1 Kajian Pustaka

Dalam melakukan kajian pustaka, sumber pustaka harus relevan terhadap masalah yang akan diteliti. Informasi yang didapatkan dari sumber pustaka tersebut disarankan berasal dari sumber-sumber pustaka ilmiah, seperti jurnal, prosiding konferensi atau seminar, tesis, disertasi, skripsi, atau buku teks, dan dihindari sumber-sumber yang tidak jelas penulisnya atau kapasitas penulisnya.

Pada penelitian ini metode yang digunakan adalah dengan algoritme *Arificial Bee Colony* (ABC) untuk mendapatkan solusi dari optimasi rute distribusi es batu. Algoritme ABC dipilih karena memiliki limit yang telah ditetapkan sehingga dapat mengendalikan banyaknya solusi tertentu yang tidak mengalami peningkatan *fitness* (Wiyanti, 2013). Adapun perbandingan objek yang relevan berdasarkan penelitian sebelumnya yang dijadikan sebagai referensi ditunjukkan pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No.	Judul	Objek	Metode	Hasil
1.	Sistem Optimasi Rute Tempat Wisata Kuliner Di Malang Menggunakan Algoritme Bee Colony (Hermawan, et al., 2017)	Penentuan rute optimal terhadap banyaknya wisata kuliner	Algoritme *Bee Colony*	Nilai rata-rata hasil *fitness* maksimum didapatkan pada jumlah *colony* 40 dengan nilai *fitness* 0,04403 dan pada iterasi 50 dengan nilai *fitness* 0,04410.
2.	Optimasi Menu Makanan Bagi Pasien Gagal Ginjal Menggunakan Algoritme Lebah (Herawati & Mahmudy,	Menu makanan	Algoritme *Bee Colony*	Hasil optimal didapatkan pada jumlah *colony* sebanyak 100 dengan nilai

	2017)			*fitness* 0,107823 dan jumlah iterasi 100 dengan nilai *fitness* 0,103334.
3.	A Bee Colony Optimization Algorithm for Traveling Salesman Problem (Wong, et al., 2008)	Jarak dan rute yang dilalui	*Artificial Bee Colony*	Algoritme *Artificial Bee Colony* (ABC) mendpatkan hasil terbaik dari pengujian tiga benchmark dan dapat ditingkatkan lagi hasilnya dengan menggunakan *neighbourhood operator.*

Pada penelitian sebelumnya yang dilakukan oleh Herawati dan Mahmudy (2017) memberikan kesimpulan bahwa jumlah individu yang kecil akan menghasilkan nilai *fitness* yang kecil dan hasil *fitness* menunjukkan konvergensi pada jumlah individu 70. Hal ini dipengaruhi oleh hasil *random* dari berat makanan yang dijadikan parameter pada tiap individu. Banyak iterasi yang dilakukan juga memengaruhi didapatkannya kemungkinan solusi terbaik. Oleh karena itu jika jumlah individu dan banyak iterasi yang digunakan semakin banyak, maka hasil *random* berat makanan yang terbentuk semakin banyak dan jika dilakukan iterasi secara terus menerus maka kemungkinan akan didapatkannya solusi terbaik juga semakin tinggi. Penelitian lainnya yang dilakukan oleh Hermawan, Hidayat, dan Setiawan (2017) memiliki kesimpulan yang sama dengan penelitian oleh Herwati dan Mahmudy (2017) yaitu jumlah *bee* atau individu dan jumlah iterasi semakin banyak maka kemungkinan didapatkannya solusi terbaik juga semakin tinggi. Tetapi, pada penelitian Hermawan, Hidayat, dan Setiawan (2017) ada satu pengujian lagi yaitu pengujian terhadap banyaknya *size problem.* Dari hasil pengujian tersebut didapatkan hasil jika jumlah *size problem* sedikit, maka kemungkinan didapatkannya solusi terbaik tinggi. Pada penelitian oleh Wong, Malcolm, dan Chong (2008) juga mendapatkan hasil apabila jumlah *size problem* semakin sedikit maka kemungkinaan didapatkan hasil optimasi akan semakin besar. Kesimpulan lain yang dihasilkan penelitian oleh Wong, Malcolm, dan Chong (2008) adalah hasil

optimasi dapat ditingkatkan dengan menambahkan *nighbourhood operator* pada setiap iterasi.

Berdasarkan penelitian yang sudah dilakukan sebelumnya, penulis mengusulkan judul "Optimasi Rute *Multiple Travelling Salesman Problem* Pada Distribusi Es Batu Dengan Algoritme *Artificial Bee Colony* (ABC)". Algortime *Artificial Bee Colony* (ABC) dipilih karena algoritme ini sangat sederhana dan efisien dibandingkan dengan algoritme swarm lainnya seperti *Ant Colony Optimization* (ACO), memiliki *robustness* yang tinggi, dan juga memiliki kemampuan untuk keluar dari *local minimum* (Karaboga & Basturk, 2007).

2.2 Distribusi

Distribusi secara umum merupakan segala aktivitas yang di dalamnya terdapat proses pemindahan material dan/atau kemampuan ekonomi melalui barang berwujud dan/atau barang tidak berwujud dari pelaku ekonomi satu ke yang lainnya. Secara singkatnya adalah distribusi mencakup sebuah sistem dari semua kegiatan yang berkaitan dengan proses pemindahan barang ekonomis antara pabrik atau produsen dengan konsumen (Segetlija, et al., 2010).

Dalam istilah lain distribusi dapat disebut dengan logistik. Logistik didefinisikan sebagai ilmu dan seni perolehan, produksi dan dsitribusi material dan produk dalam kuantitas dan tempat yang tepat atau definisi lainnya adalah proses perencanaan, implementasi, dan pengendalian secara efisien, aliran biaya yang efektif dan penyimpanan barang mentah, inventori barang dalam proses, barang jadi dan informasi terkait dari titik asal ke titik konsumsi untuk tujuan memenuhi kebutuhan konsumen (Sembiring, 2014).

2.3 *Artifical Bee Colony* (ABC)

Artificial Bee Colony (ABC) adalah algoritme yang terinspirasi dari perilaku kawanan lebah. Pada algoritme ABC terdapat 2 jenis lebah penjelajah (*foragers*), yaitu *employed foragers* dan *unemployed foragers* (Cholissodin & Riyandani, 2016). *Employed foragers* memiliki tugas untuk mengeksploitasi sumber-sumber makanan (berhubungan dengan solusi dari masalah optimasi) dan membawa informasi tentang posisi sumber makanan ke sarang lebah. Untuk *unemployed foragers* terdapat 2 jenis yaitu *onlooker* dan *scout bee*. *Onlooker bee* bertugas pergi ke sumber makanan untuk dieksploitasi dengan informasi dari *employed forages* sebagai bahan pertimbangan. Tugas dari *scout bee* adalah mencari sumber makanan di sekitar sarang lebah.

Untuk mendapatkan hasil optimum menggunakan algoritme ABC, berikut adalah langkah-langkah yang dilakukan (Cholissodin & Riyandani, 2016):

1. Inisialisasi parameter.
2. Fase *initial*
3. *Improvement Solution*
4. Kondisi Berhenti

2.3.1 Inisialisasi Parameter

Nilai parameter dibutuhkan untuk membantu dalam perhitungan algoritme *Artificial Bee Colony* (ABC). Adapun nilai parameter yang dibutuhkan yaitu *colony size, route size, limit,* dan nilai maksimum iterasi. *Colony Size* merupakan jumlah lebah keseluruhan yang akan digunakan yang disebut juga dengan *Population Size* atau *PopSize*. *Route size* adalah banyak rute dalam satu *PopSize.* Limit merupakan batasan dari *PopSize trial* yang tidak terjadi peningkatan dalam sejumlah iterasi.

2.3.2 Fase *Initial*

Fase *initial* adalah prose yang dilakukan untuk mendapat nilai *initial solution* secara random (Otri, 2011) untuk tiap koloni dan hitung nilai *fitness*-nya. Untuk mendapatkan nilai *fitness* dapat menggunakan Persamaan 2.1.

$$fitness_i = \begin{cases} \frac{1}{1+f_i}, if\ (f_i \geq 0) \\ 1 + abs(f_i), if\ (f_i < 0) \end{cases} \quad (2.1)$$

Keterangan:

$fitness_i$ = Nilai *fitness* lebah ke-*i*

f_i = Jumlah total jarak tiap tujuan

2.3.3 *Improvement Solution*

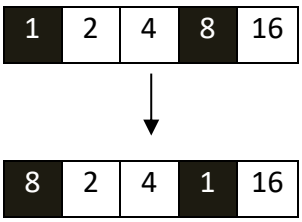
Improvement Solution berada pada fase *Employed Bee* dan *Onlooker bee* yang digunakan untuk memperbarui solusi dengan metode *neighborhood operator. Neighborhood operator* pada fase *Employed Bee* menggunakan *swap operator* dan *swap sequence.* Sedangkan pada fase *Onlooker bee* menggunakan *insertion operator* dan *insertion sequence.*

1. Fase *Employed Bee*

Swap operator(SO). Misal bilangan acaknya SO(1,4), maka solusi baru yang dihasilkan dari *initial solution* dengan SO pada Persamaan 2.2.

$x_i = x_i + SO$

x_i = (1-2-4-8-16)+SO(1,4) (2.2)



Setelah semua *employed bee* sudah melakukan proses *Swap operator* selanjutnya akan dilakukan *Swap sequence* (SS) pada Persamaan 2.3.

$x_i = x_i + SS$

$= x_i + (SO_1, SO_2, \ldots, SO_n)$

$= ((((x_i + SO_1) + SO_2) +) \ldots + SO_n)$ (2.3)

n = banyaknya *swap sequence,* kemudian hitung nilai probabilitas dengan Persamaan 2.4

$$Prob_i = \frac{fitness(x_i)}{\sum_{k=1}^{S} fitness(x_i)} \quad (2.4)$$

Dimana

$Prob_i$ = Peluang memilih *Employed Bee* ke-*i*

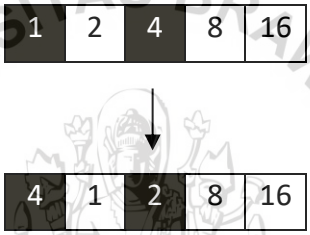
S = Jumlah *Employed Bee*

2. Fase *Onlooker bee*

Insert operator (IO). Misal bilangan acaknya IO(1,3), maka solusi baru yang dihasilkan dari *initial solution* dengan IO pada Persamaan 2.5 berikut:

$x_i = x_i + IO$

$x_i = (1\text{-}2\text{-}4\text{-}8\text{-}16) + IO(1,3)$ (2.5)



Setelah semua *employed bee* sudah melakukan proses *Insert operator* selanjutnya akan dilakukan *Insert Sequence* (IS).

$x_i = x_i + IS$

$= x_i + (IO_1, IO_2, \ldots, IO_n)$

$= ((((x_i + IO_1) + IO_2) +) \ldots + IO_n)$ (2.6)

n = banyaknya *Insert Sequence*

3. Fase *Scout bee*

Setelah melakukan *Improvement Solution,* selanjutnya akan dilakukan perhitungan kualitas masing-masing *Employed Bee*. Jumlah *Scout bee* tergantung pada jumlah *Employed Bee* yang telah melebihi nilai *trial*. Apabila nilai *trial* sudah melebihi nilai *trial* yang sudah ditentukan, maka solusi dari bee tersebut akan diganti dengan solusi baru dengan mengacak rute baru yang berbeda, *update* jarak, dan menyetel ulang nilai *trial* menjadi 0.

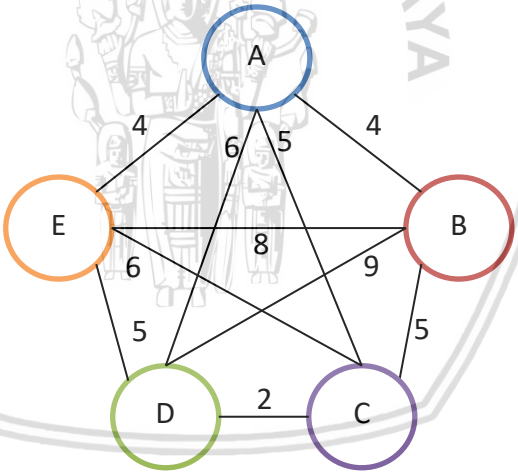
2.3.4 Kondisi Berhenti

Perhitungan akan berhenti dalam algoritme *Artificial Bee Colony* (ABC) dilakukan jika mencapai kondisi berhenti yaitu sudah mencapai nilai maksimum iterasi. Apabila belum terpenuhi proses akan mengulang dari langkah *Improvement Solution*.

2.4 *Multiple Travelling Salesman Problem* (MTSP)

Multiple Travelling Salesman Problem (MTSP) adalah pengembangan dari *Travelling Salesman Problem* (TSP) yaitu sebuah masalah yang menyatakan seseorang yang ingin mengunjungi sebuah kota dengan pilihan banyak jalur dengan melewati beberapa yang mana pilihan kota yang dikunjungi akan membuat rangkaian rute yang sedemikian rupa yang hanya boleh dilewati sebanyak satu kali dan kembali lagi ke kota awal dengan tujuan mencari rute terpendek (Amri, et al., 2012). Perbedaan antara TSP dengan MTSP adalah terletak pada jumlah sales yaitu pada MTSP jumlah sales yang ada adalah lebih dari satu orang (Singh & Lodhi, 2014) oleh karena itu dengan jumlah *sales* yang lebih dari satu, maka proses distribusi akan semakin cepat dan dapat memperkecil biaya transportasi (Karimah, et al., 2017).

Sebagai ilustrasi konsep MTSP, pada Gambar 2.1 terdapat graf yang merepresentasikan kota-kota yang akan dikunjungi. Diasumsikan node atau simpul A sebagai titik awal dan titik akhir rute. Dalam mendapatkan rute awal terdapat dua proses (Xue, et al., 2016) pertama adalah membangkitkan permutasi jumlah *n* titik dan kedua adalah membagi secara acak rangkaian rute ke dalam *m* jalur. *M* adalah jumlah *sales* dan setidaknya pada satu jalur terdapat satu titik.



Gambar 2.1 Ilustrasi Graf *Travelling Salesman Problem*

Misal rute awal yang dihasilkan adalah sebagai berikut:

X1 : A → C → D → B → E → A

Kemudian dibagi menjadi dua tergantung jumlah *sales,* maka rute untuk MTSP-nya adalah sebagai berikut:

X1 : *Salesman* 1 : A → C → D → A | *Salesman* 2 : A → B → E → A

Setelah didaptkan rute awal selanjutnya hitung nilai *fitness* dengan Persamaan 2.1.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini berisi penjelasan tentang alur dari metodologi penelitian yang dilakukan.

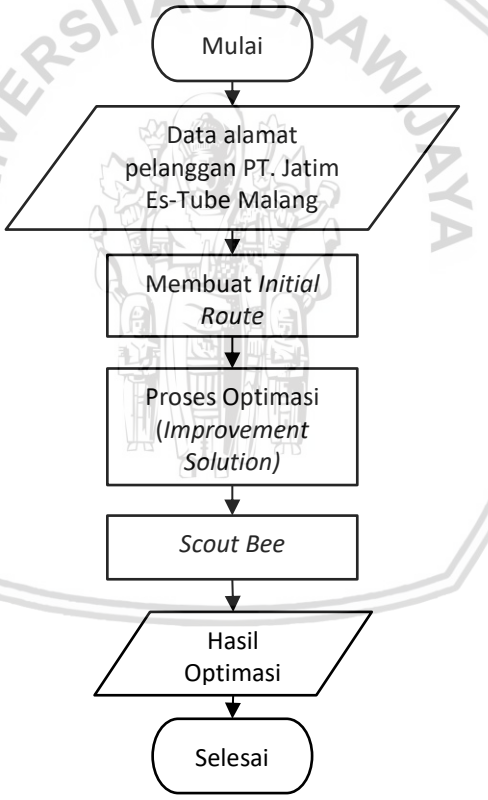
3.1 Tipe Penelitian

Tipe penelitian penelitian ini adalah analitik non-implementatif yaitu berupa kegiatan menganalisis jarak tercepat dengan nilai *fitness* tertinggi dalam objek penelitian yang diteliti.

3.2 Strategi dan Rancangan Penelitian

Strategi dan rancangan penelitian yang dilakukan dijelaskan pada sub-bab berikut.

3.2.1 Metode Secara Umum



Gambar 3.1 Diagram Alir Gambaran Umum Sistem

Pada penelitian ini akan dibuat sebuah program yang akan mengimplementasikan algoritme *Artificial Bee Colony* (ABC) untuk optimasi rute. Pada Gambar 3.1 data alamat pelanggan yang berupa jarak digunakan sebagai masukan sistem. Selanjutnya dilakuakn pembentukan rute *initial.* Setelah didapatkan rute *initial* sleanjutnya dilakukan proses optimasi yang dibagi menjadi 2 bagian yaitu *employeed bee* dan *onlooker bee. Employeed bee* memiliki 2 proses optimasi yaitu *Swap operator* yang berfungsi untuk menukar titik

pengiriman sesuai dengan nilai *operator* dan *swap sequences* proses penukaran titik pengiriman secara berurutan. *Onlooker bee* memiliki 2 proses optimasi yaitu *insert operator* yang berfungsi melakukan penyisipan sesuai dengan nilai optimasi dan *insert sequences* yang berfungsi untuk melakukan penyisipan secara berurutan. Setelah dilakukan optimasi selanjutnya dilakukan pengecekan apakah rute tersebut mengalami peningkatan atau tidak. Jika mengalami peningkatan maka yang terakhir adalah membandingkan hasil *scout bee* dengan rute *initial*. Hasil perbandingan tersebut akan menentukan individu yang menghasilkan nilai optimal. Keluaran yang dihasilkan berupa Rute yang memiliki hasil *fitness* terbaik.

3.2.2 Objek Penelitian

Objek yang disertakan dalam penelitian ini adalah PT. Jatim Es-Tube Malang. Data alamat pelanggan pada satu hari tersebut akan dihitung jarak antar alamat. Optimasi dilakukan dengan menggunakan parameter *size problem* atau jumlah titik pengiriman, *colony size* yaitu ukuran koloni yang akan menentukan banyaknya pilihan rute, *limit* yaitu batasan untuk nilai trial yang tidak mengalami perbaikan, dan iterasi untuk mengeetahui sebanyak apa iterasi yang dibutuhkan untuk mencapai nilai optimal.

3.2.3 Lokasi Penelitian

Penelitian ini dilakukan di laboratorium riset komputasi cerdas Fakultas Ilmu Komputer Universitas Brawijaya.

3.2.4 Metode Pengumpulan Data

Data berasal dari PT. Jatim Es-Tube Malang berupa daftar alamat pelanggan yang sudah melakukan pemesanan sebelumnya untuk satu hari. Data tersebut merupakan data primer karena diperoleh langsung saat melakukan observasi lapangan.

3.2.5 Peralatan Pendukung

Penelitian ini dilakukan pada perangkat komputer dengan spesifikasi sebagai berikut:

- Prosesor : Intel® Core™ i5-6200U CPU @ 2.30GHz
- RAM : 4 Gigabyte
- Sistem Operasi : Windows 10 Home 64-bit

Selain itu, Bahasa pemrograman yang akan digunakan untuk mengimplementasikan algoritme pada penelitian ini adalah Java.

3.3 Jadwal Kegiatan

Penelitian ini dilakukan dalam penjadwalan selama kurun waktu empat bulam. Rincian dari rencana jadwal kegiatan dapat dilihat pada Tabel 3.1.

Tabel 3.1 Rencana Jadwal Kegiatan

No.	Kegiatan	Bulan			
		1	2	3	4
1	Studi Pustaka				
2	Pengumpulan Data				
3	Perancangan				
4	Implementasi				
5	Pengujian dan Analisis				
6	Kesimpulan dan Saran				

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

4.1.1 Deskripsi Umum

Dalam menentukan rute jarak terpendek dari distributor ke tempat tujuan diperlukan adanya jarak antar distributor ke alamat-alamat pelanggan dan jarak antar pelanggan yang satu dengan yang lainnya untuk mendapatkan total jarak yang akan ditempuh oleh *sales*. Sehingga akan memperoleh keuntungan maksimal dari rute terpendek terebut dengan meminimalkan biaya transportasi dalam melakukan distribusi.

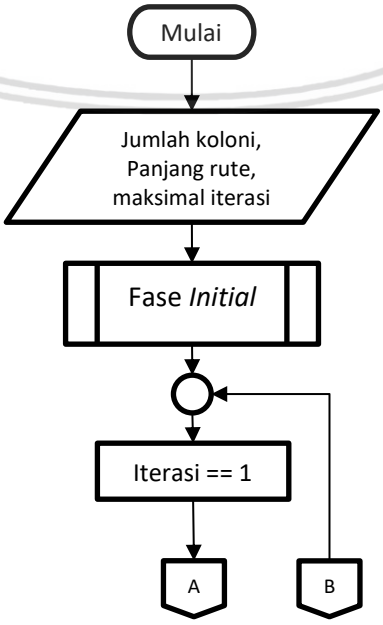
Pada penelitian ini, jumlah toko yang akan dikunjungi oleh *sales* dan jarak yang ditempuh berdasarkan perhitungan dari Google Maps akan dijadikan parameter untuk proses manualisasi. Setelah semua parameter dimasukkan, selanjutnya program akan melakukan optimasi dengan algoritme *Artificial Bee Colony* dengan menghasilkan optimasi jarak terpendek pada saat melakukan distribusi es batu dari distributor.

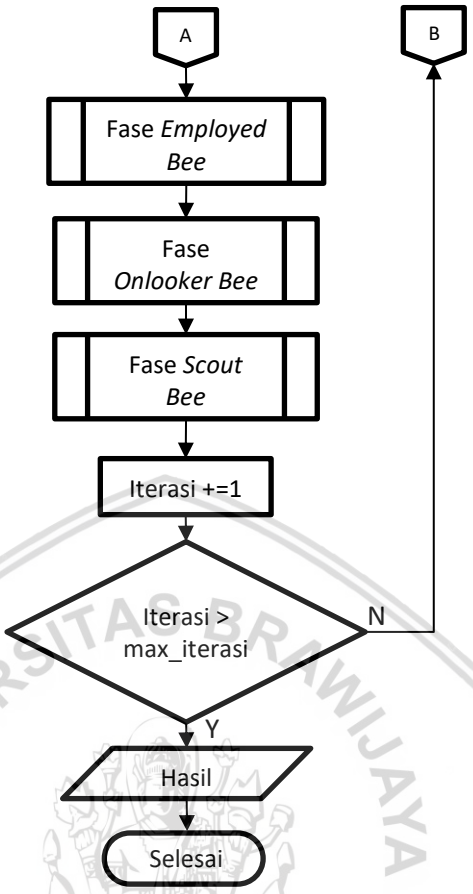
4.2 Perancangan Implementasi *Artificial Bee Colony*

Perancangan implementasi algoritme *artificial bee colony* digunakan sebagai penentu rute terpendek pada setiap alamat pelanggan yang dilalui oleh *sales* dari pusat distribusi ke alamat pelanggan dan kembali lagi ke pusat distribusi.

4.2.1 Implementasi *Artificial Bee Colony*

Pada tahap ini proses implementasi algoritme *Artificial Bee Colony* untuk optimasi rute distribusi es batu digambarkan pada diagram alir seperti pada Gambar 4.1.



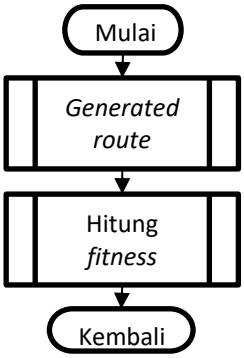


Gambar 4.1 Implementasi *Artificial Bee Colony*

Tahap pertama adalah dengan menginisialisasi parameter berupa maksimum iterasi, *size problem*, *limit*, dan *colony size* yang selanjutnya masuk ke dalam fase *initial* yang secara *random* untuk mendapat *initial solution* tiap *Employeed Bee*. Setelah didapatkan *initial solution*-nya, masuk pada iterasi pertama dilanjutkan dengan *improvement solution* yang terdiri dari fase *employed bee* dan *onlooker bee.* Dari *improvement solution* dilanjutkan ke fase *scout bee* untuk mendapatkan solusi terbaik lalu disimpan. Data akan terus melakukan iterasi sampai jumlah iterasi sama dengan jumlah max_iterasi. Jika kondisi berhenti belum terpenuhi, maka proses akan mengulang dari *improvement solution*.

4.2.2 Fase *Initial*

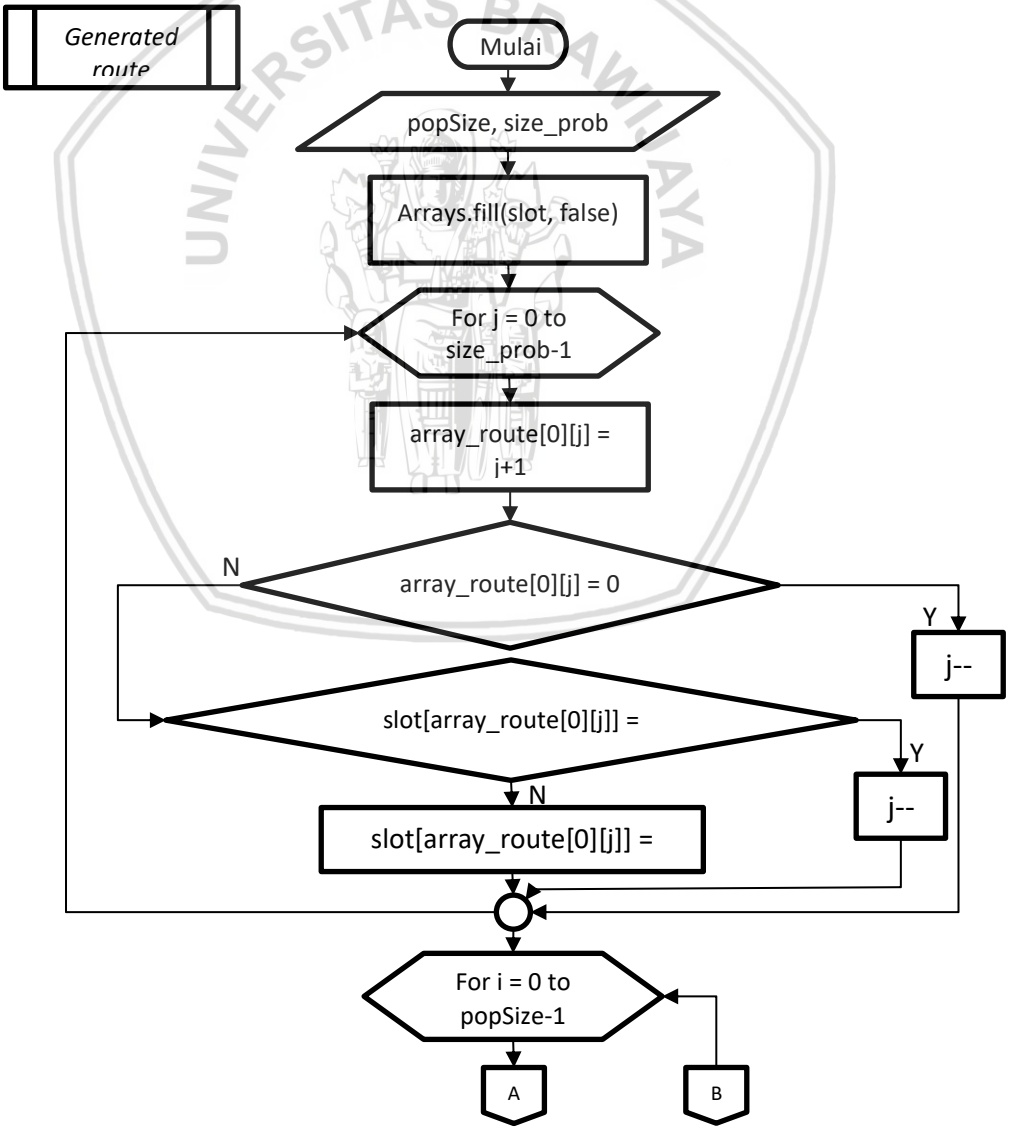
Fase *initial* merupakan fase untuk mendapat nilai solusi awal atau *initial solution* pada tiap *employed bee*. Pada fase ini akan dilakukan penentuan rute secara *random* atau *generated route* dan setelah itu dihitung nilai *fitness*-nya masing-masing dengan menggunakan rumus pada Persamaan 2.1. Fase *initial* dapat dilihat pada Gambar 4.2.

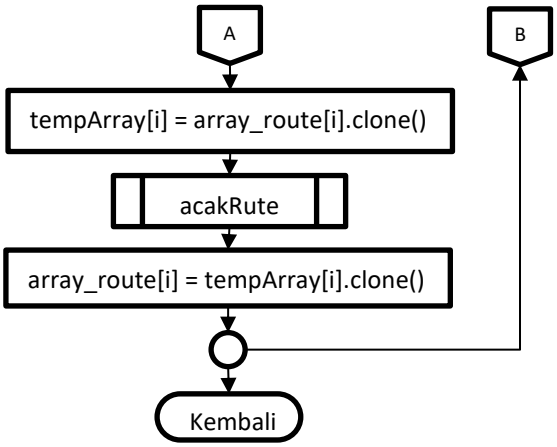


Gambar 4.2 Fase *Initial*

4.2.2.1 Fungsi *Generated Route*

Fungsi *generated route* adalah fungsi yang digunakan untuk menentukan rute awal pada fase *initial solution* yang tergantung jumlah *size problem*. Diagram alir fungsi dapat dilihat pada Gambar 4.3.

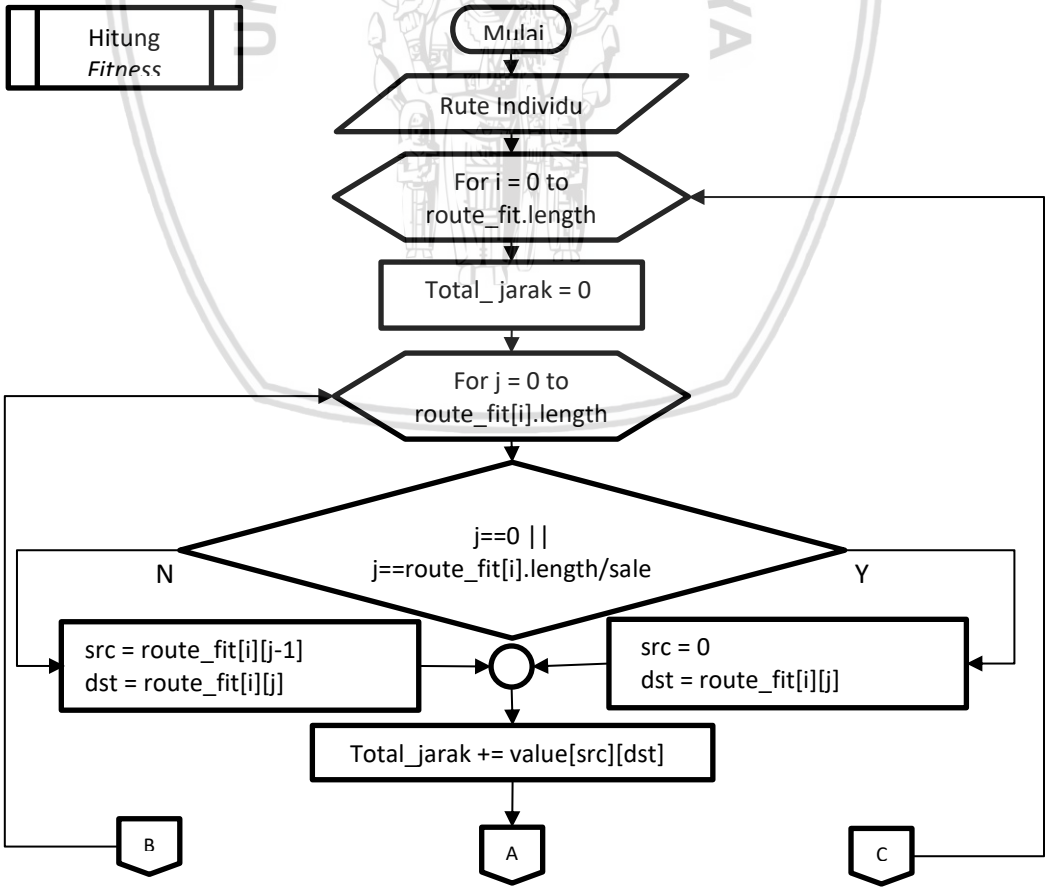


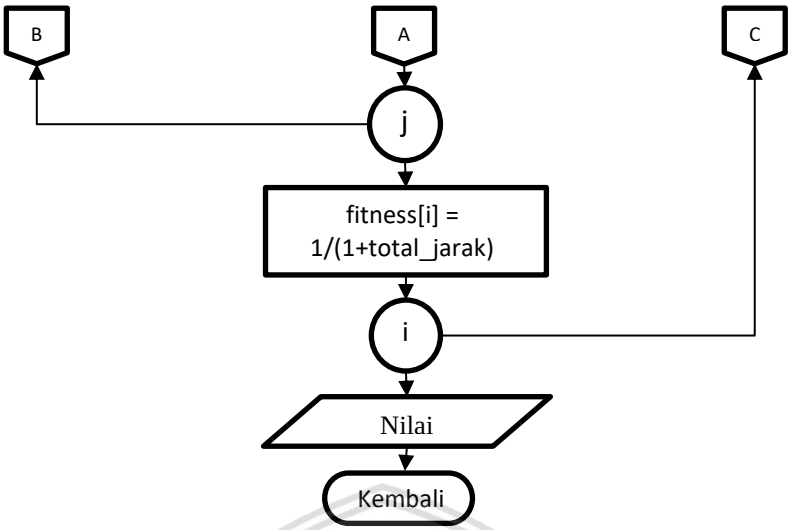


Gambar 4.3 *Generated Route*

4.2.2.2 Fungsi Hitung *Fitness*

Fungsi ini berfungsi untuk menghitung nilai *fitness* tiap individu. Pertama lakukan inisialisasi yang selanjutnya dilakukan *looping* penghitungan total jarak sebanyak *size problem* sesuai dengan banyaknya *colony* yang dideklarasikan sebelumnya, sehingga ditemukan total jarak masing-masing *bee colony* dan dihitung nilai *fitness*-nya sesuai dengan Persamaan 2.1. Diagram alir proses perhitungan nilai *fitness* dapat dilihat pada Gambar 4.4.

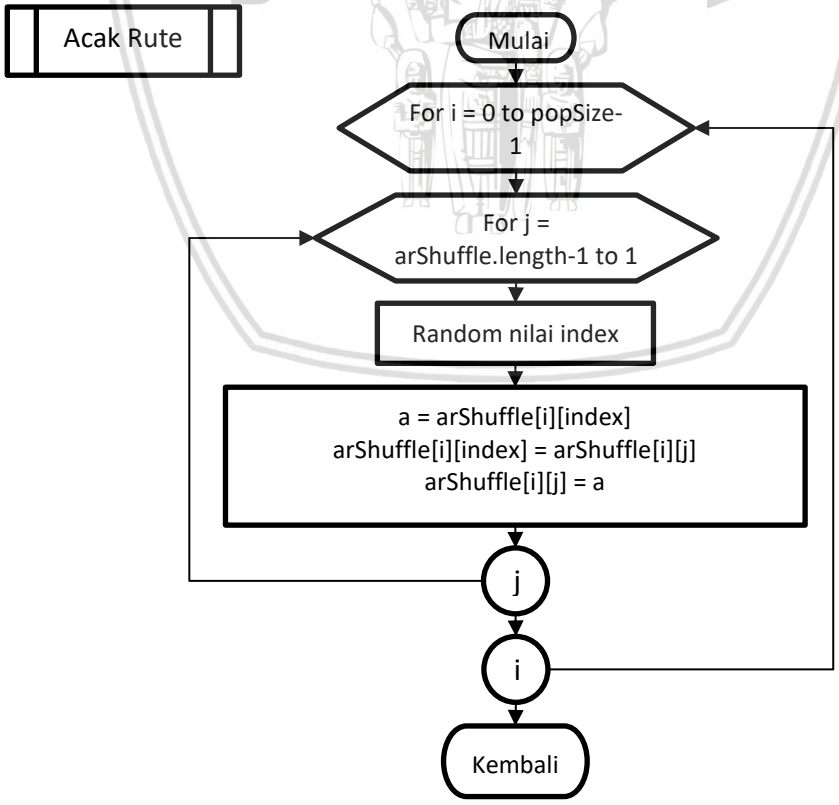




Gambar 4.4 Hitung Nilai *Fitness*

4.2.2.3 Fungsi Acak Rute

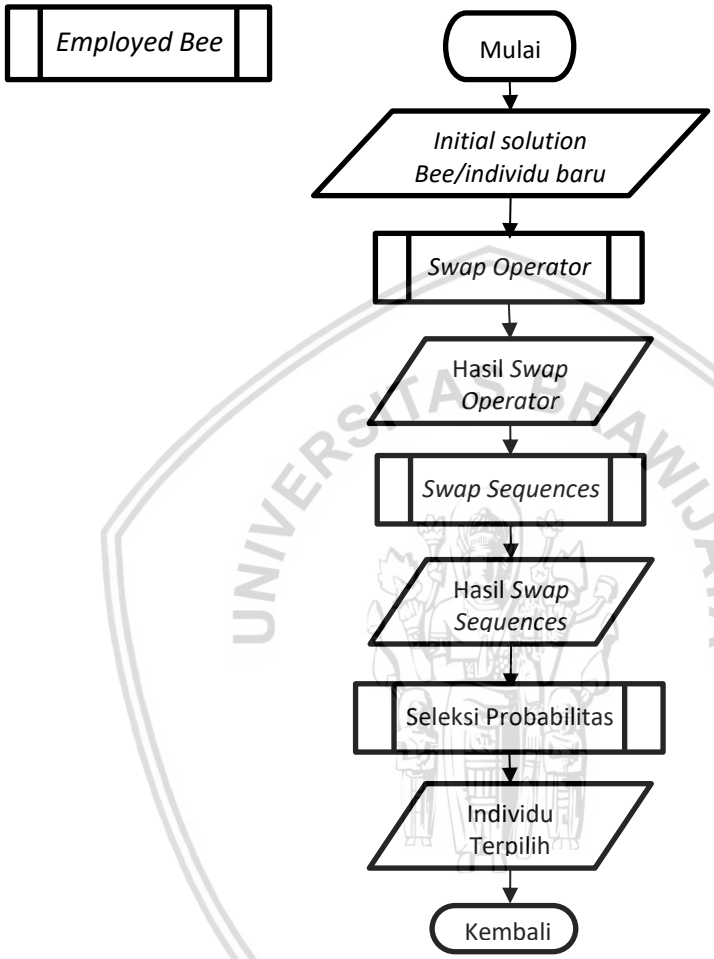
Fungsi acak rute berfungsi untuk mengacak rute pada masing-masing *colony* dengan melakukan *looping* sebanyak jumlah *colony* dan jumlah *size problem.* Di dalam *loop* terdapat fungsi *random* yang kemudian dijadikan *index array* lalu ditukar dengan *array* sesuai posisi *index* pertama. Diagram alir proses acak rute dapat dilihat pada Gambar 4.5



Gambar 4.5 Acak rute

4.2.3 Fase *Employeed Bee*

Fase *Employed Bee* digunakan untuk memperbarui solusi dengan metode *neighborhood operator. Neighborhood operator* pada fase *Employed Bee* menggunakan *swap operator* dan *swap sequence.* Diagram alir untuk fase *employeed bee* dapat dilihat pada Gambar 4.6.

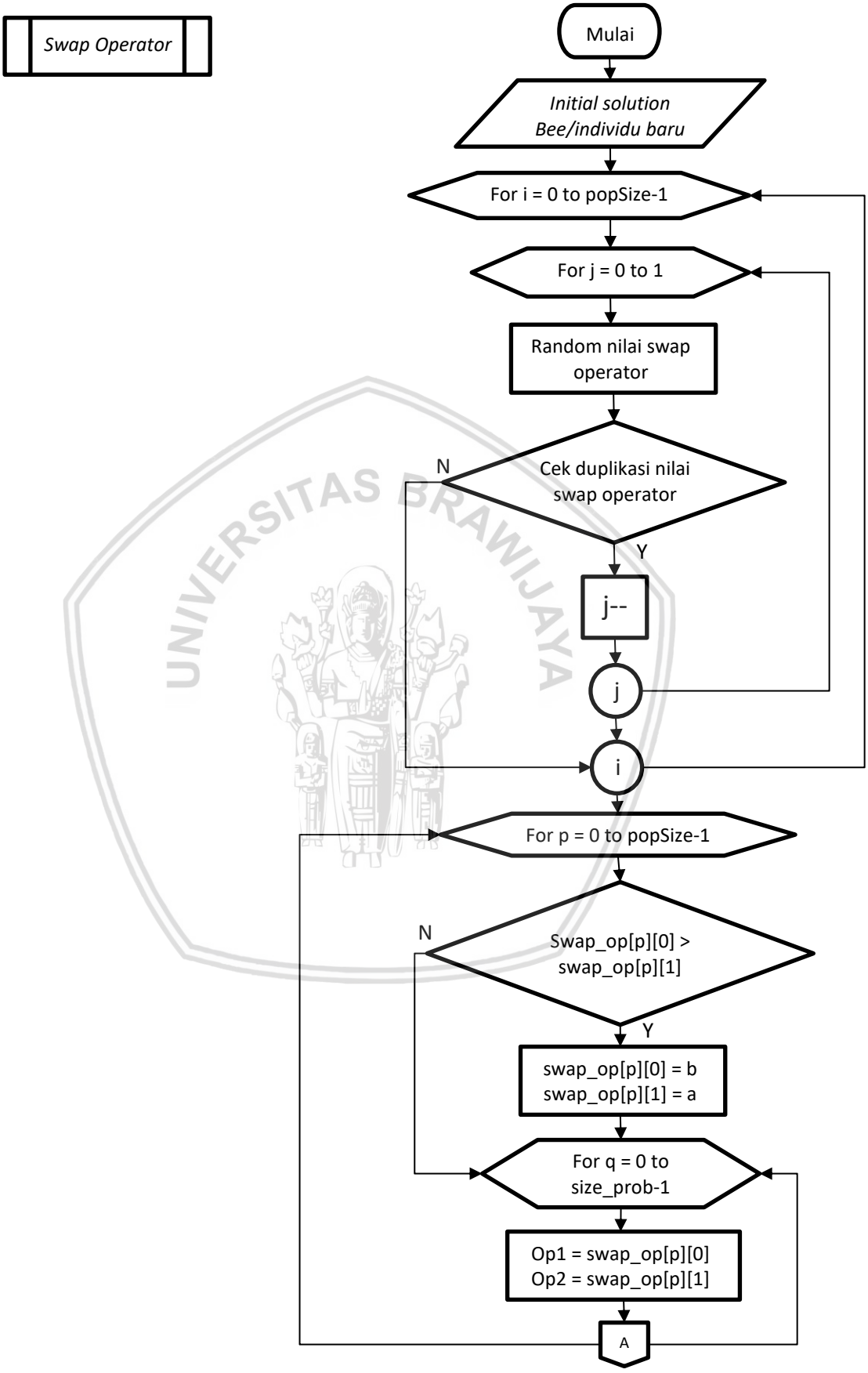


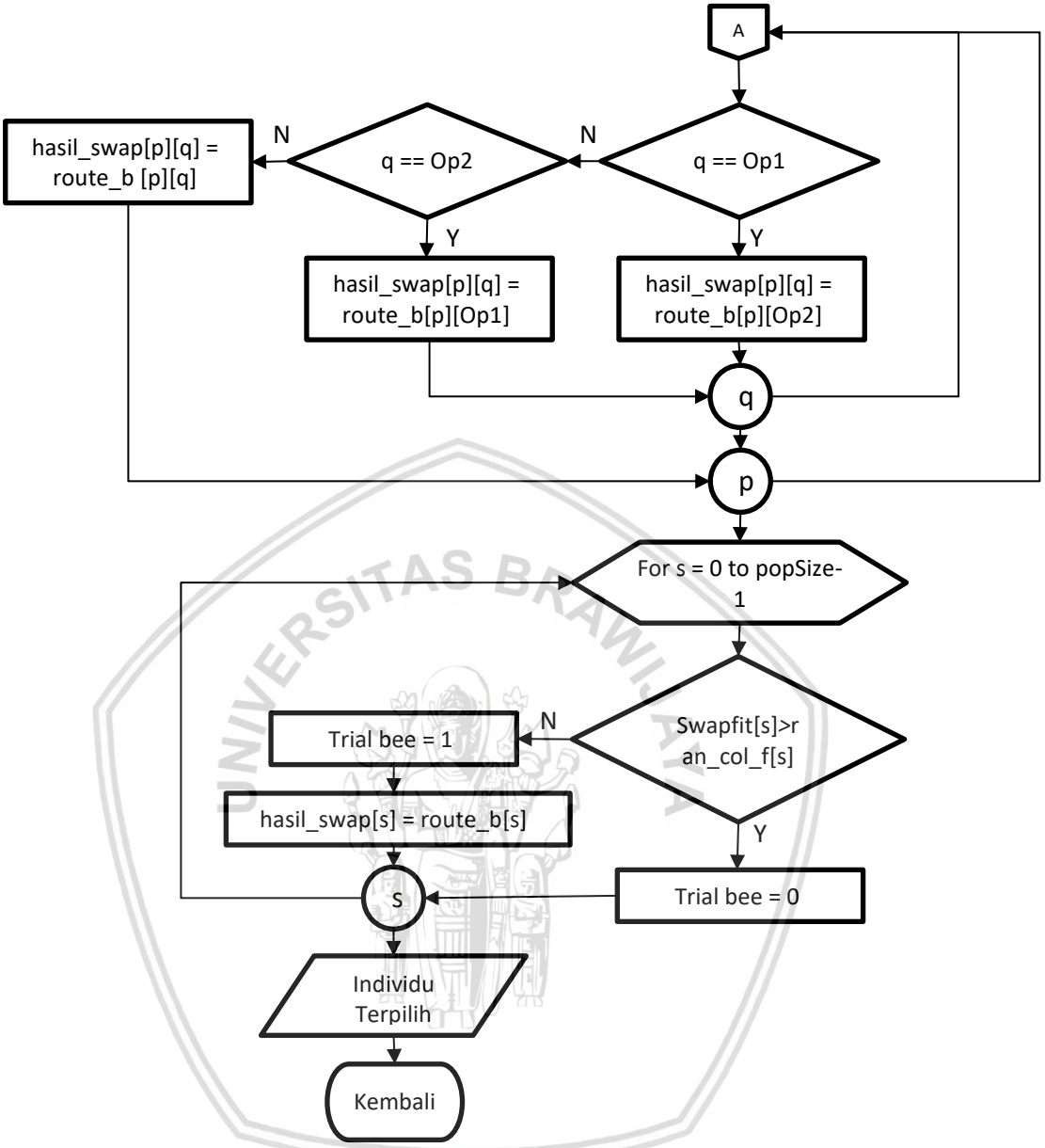
Gambar 4.6 *Employeed Bee*

4.2.3.1 *Swap Operator*

Pada Gambar 4.6 ditampilkan alur penghitungan *swap operator* yang dijabarkan sebagai berikut:

1. Inisialisasi
2. Nilai *swap operator* didapatkan secara *random* sesuai dengan jumlah *popSize* dan jumlah titik pengirimannya.
3. Jika sudah dihasilkan hasil *swap operator,* selanjutnya dihitung nilai *fitness* dari hasil *swap operator.* Apabila nilai *fitness* hasil *swap operator* lebih besar dari nilai *fitness* individu sebelumnya maka nilai trial akan di-*reset* menjadi 0 dan jika tidak maka nilai trial akan ditambah 1.

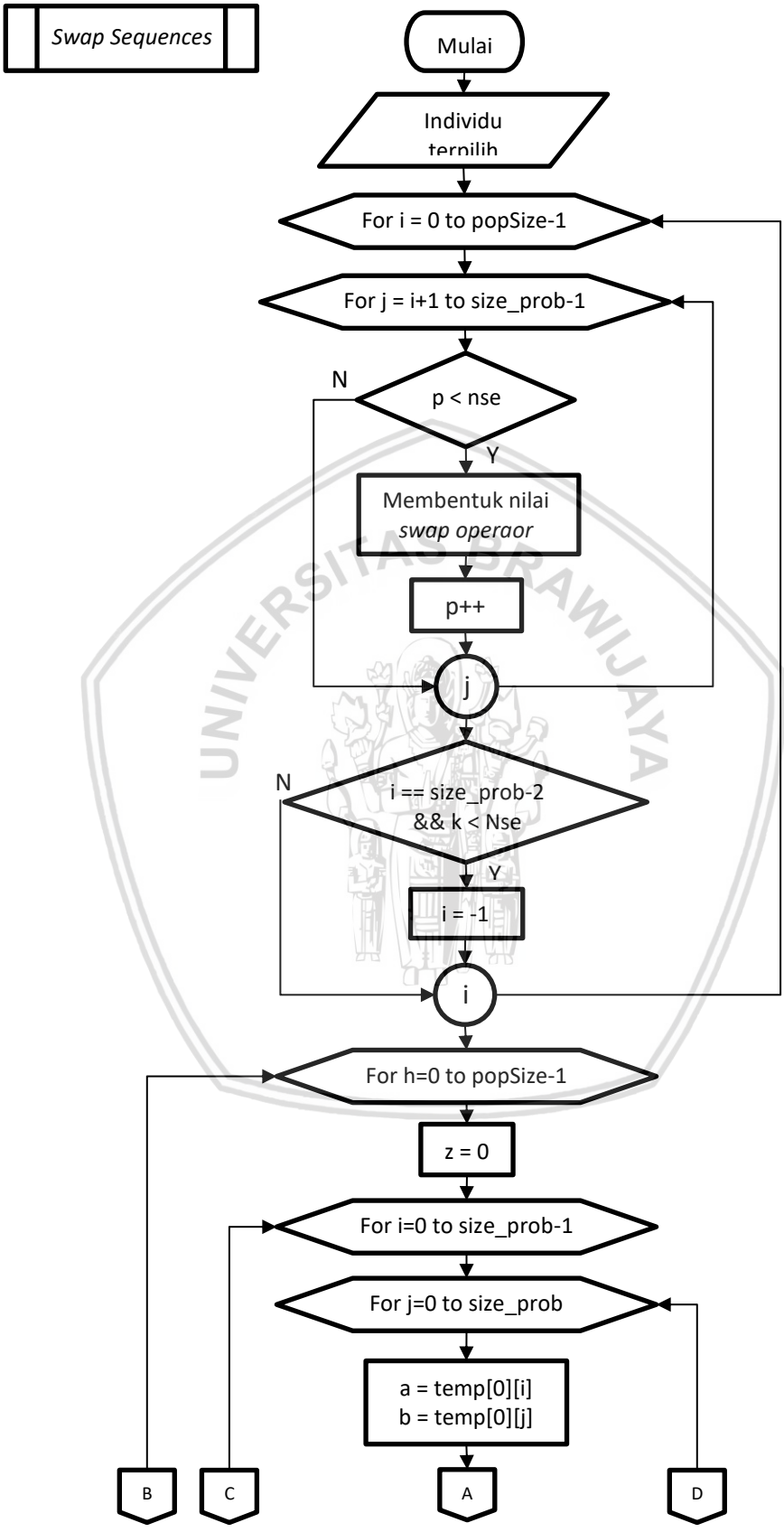

Swap Operator
Mulai
Initial solution
Bee/individu baru
For i = 0 to popSize-1
For j = 0 to 1
Random nilai swap operator
N
Cek duplikasi nilai swap operator
Y
j--
j
i
For p = 0 to popSize-1
N
Swap_op[p][0] > swap_op[p][1]
Y
swap_op[p][0] = b
swap_op[p][1] = a
For q = 0 to size_prob-1
Op1 = swap_op[p][0]
Op2 = swap_op[p][1]
A

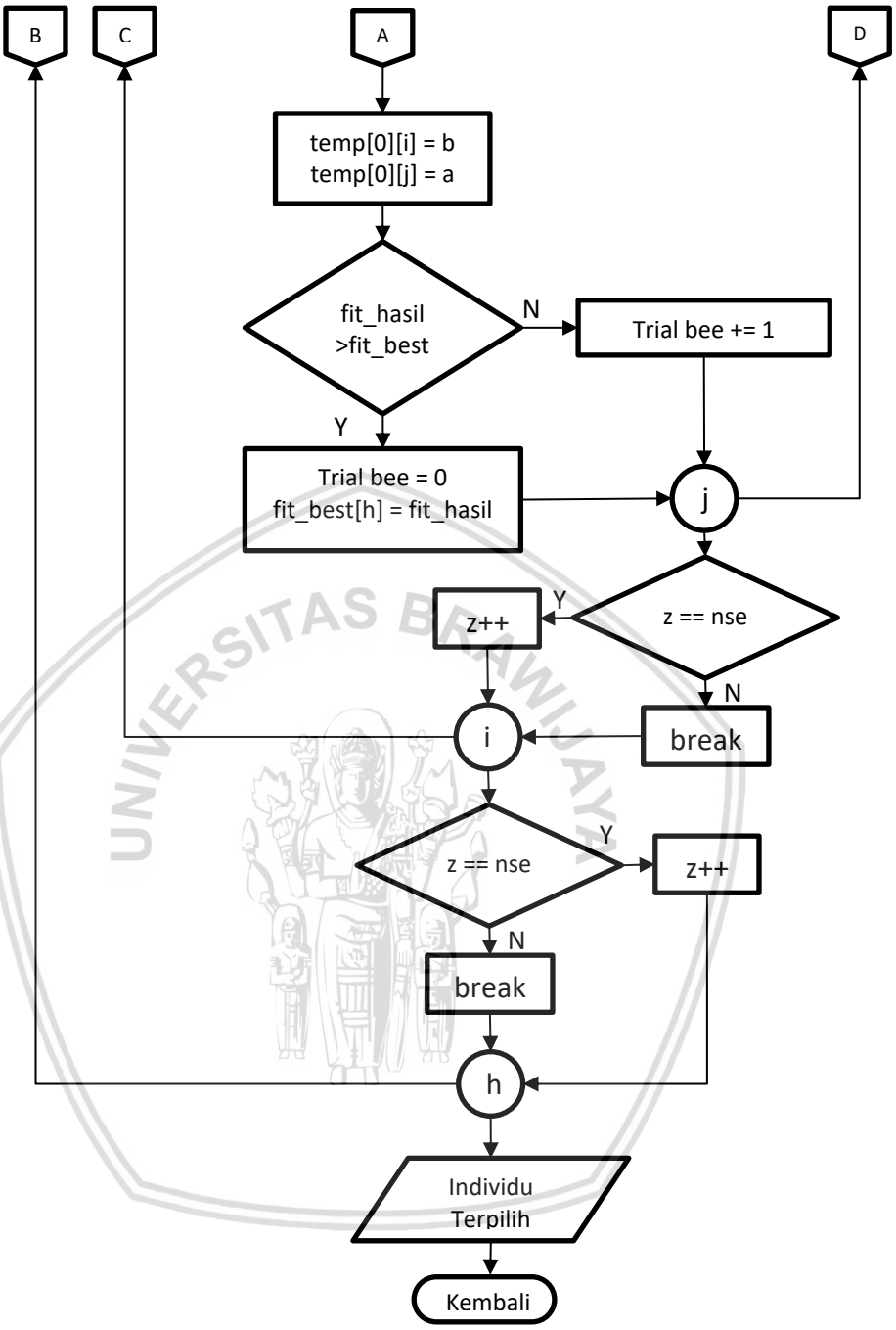


Gambar 4.7 *Swap Operator*

4.2.3.2 *Swap Sequences*

Individu untuk *swap sequences* didapatkan dari hasil *swap operator.* Pada fase *swap sequences,* rute tiap individu akan diacak lagi sebanyak *number of sequences*. Setiap selesai satu kali proses *swap*¸ akan dihitung nilai *fitness*-nya. Jika hasil nilai *fitness* baru lebih besar dari nilai *fitness* lama, maka nilai *trial* akan di-*reset* menjadi 0. Tetapi, jika tidak lebih besar nilai *trial* sebelumnya akan ditambah 1. Gambar 4.7 menunjukan alur perhitungan *swap sequences.*


Swap Sequences
Mulai
Individu terpilih
For i = 0 to popSize-1
For j = i+1 to size_prob-1
N
p < nse
Y
Membentuk nilai swap operaor
p++
j
N
i == size_prob-2 && k < Nse
Y
i = -1
i
For h=0 to popSize-1
z = 0
For i=0 to size_prob-1
For j=0 to size_prob
a = temp[0][i]
b = temp[0][j]
B
C
A
D

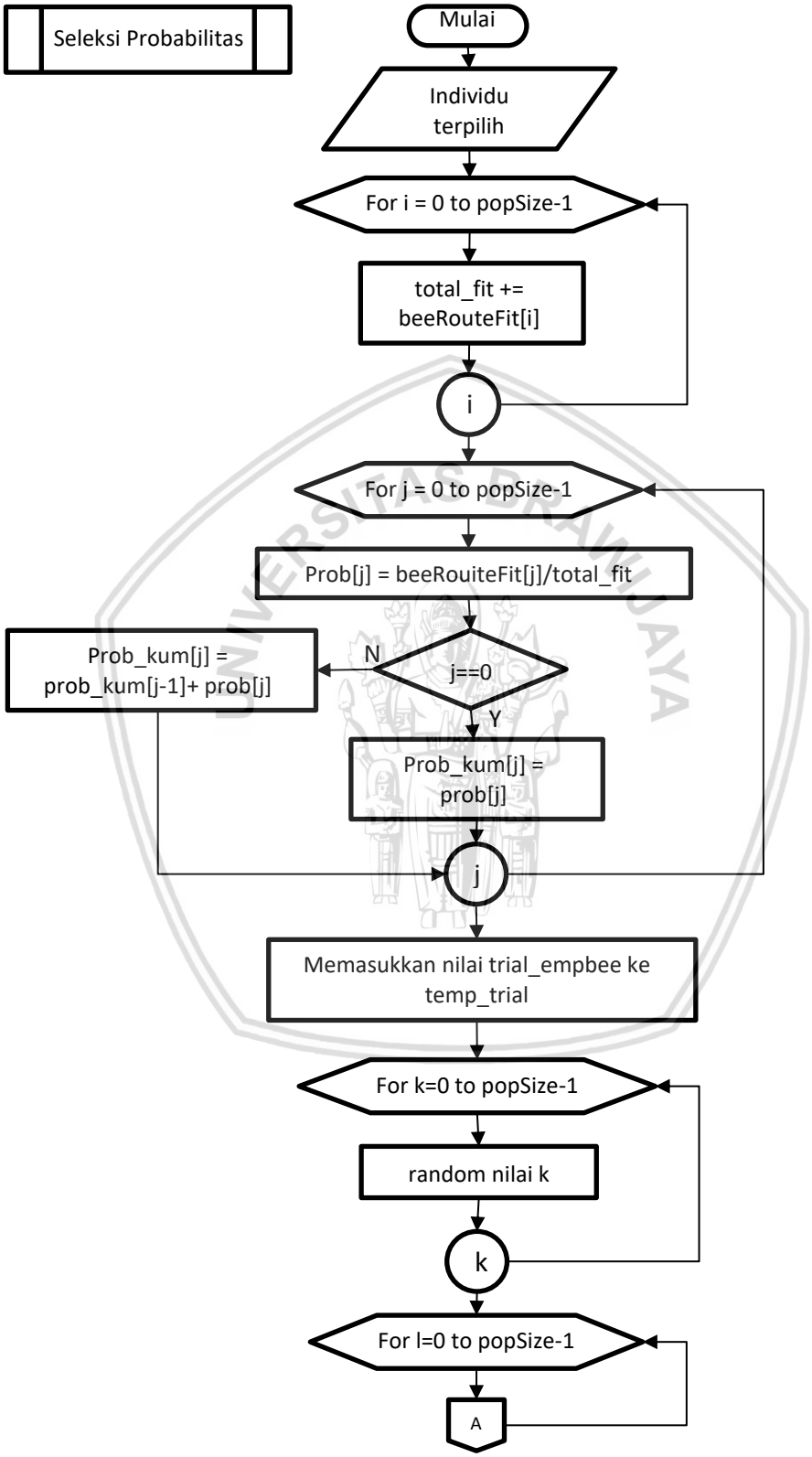


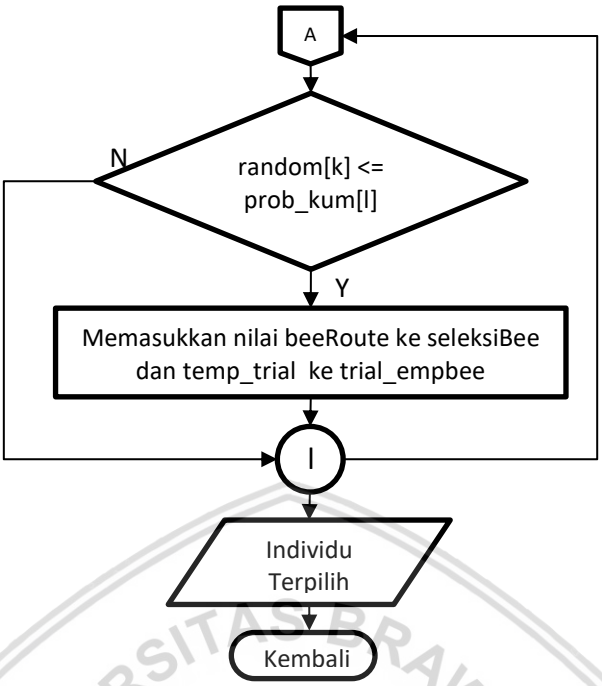
Gambar 4.8 *Swap Sequences*

4.2.3.3 Fungsi Seleksi Probabilitas

Sebelum masuk fase *onlooker bee,* individu yang dipilih diseleksi dengan menghitung probabilitasnya. Kemudian dari probabilitas tersebut didapatkan probabilitas kumulatif yang selanjutnya dijadikan *range.* Setelah didapatkan *range* masing-masing individu, selanjutnya membangkitkan nilai *random* untuk menentukan individu yang akan dipilih. Nilai *random* yang terpilih akan

disesuaikan dengan *range* masing-masing individu. Gambar 4.8 merupakan alur perhitungan fungsi seleksi probabilitas.

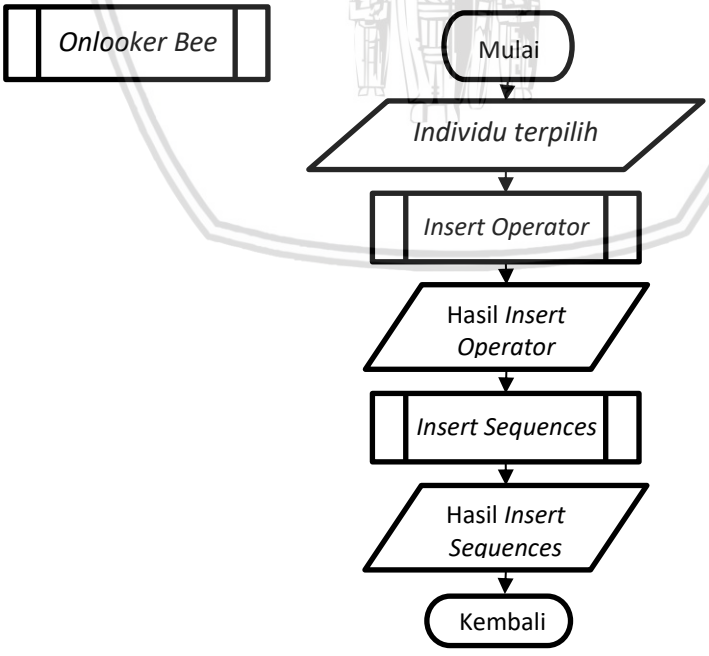




Gambar 4.9 Seleksi Probabilitas

4.2.4 Fase *Onlooker Bee*

Pada fase *Onlooker Bee* digunakan untuk memperbarui solusi dengan metode yang sama dengan *Employeed Bee* yaitu *neighborhood operator. Tetapi neighborhood operator* pada fase *Employed Bee* menggunakan *insert operator* dan *insert sequence.* Alur fase *onlooker bee* dapat dilihat pada Gambar 4.9.

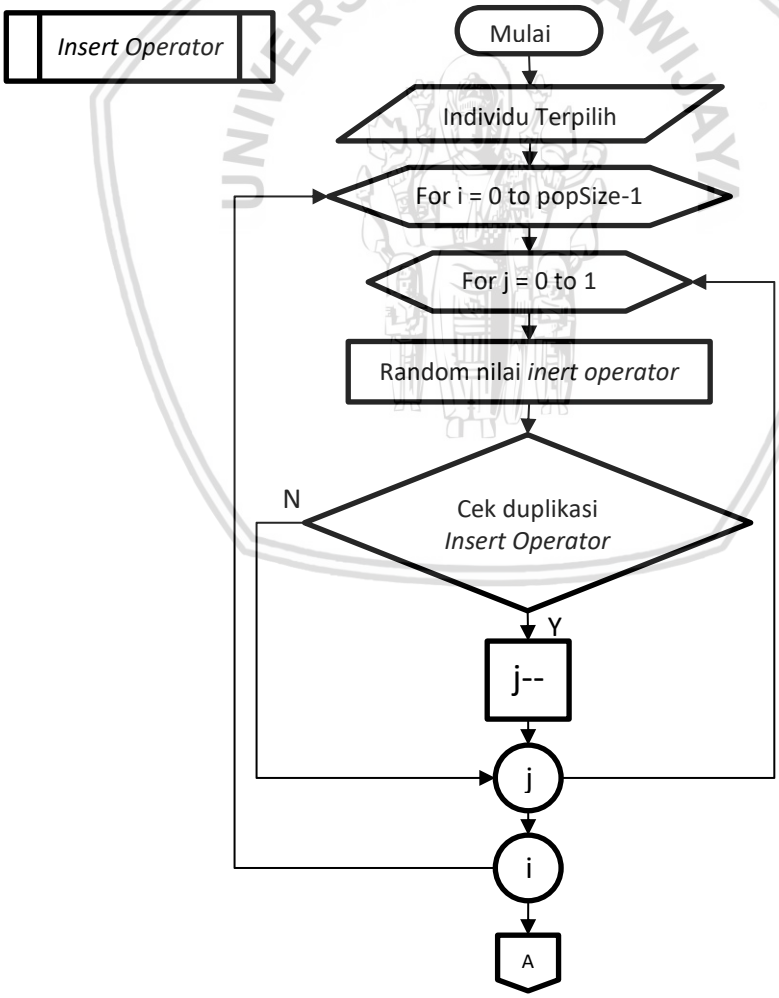


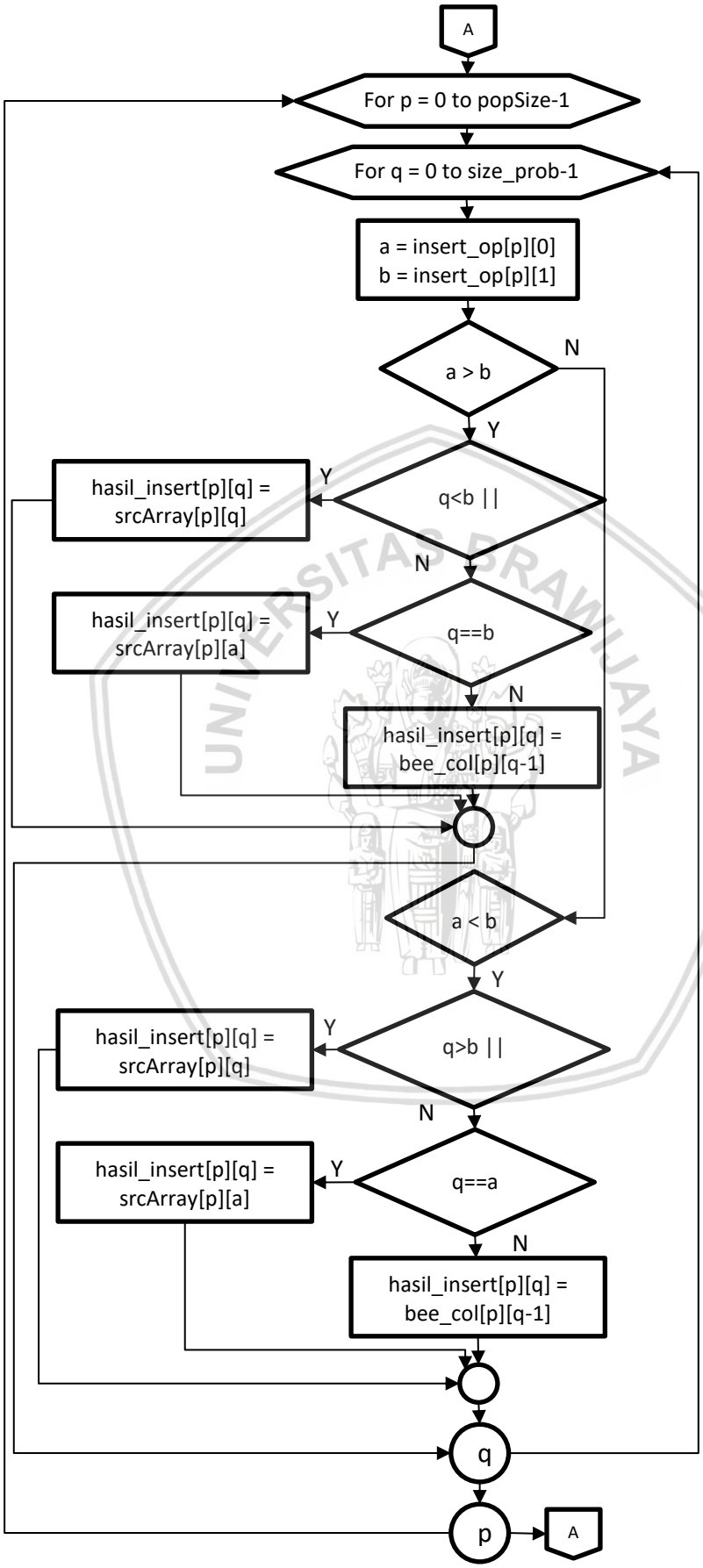
Gambar 4.10 *Onlooker Bee*

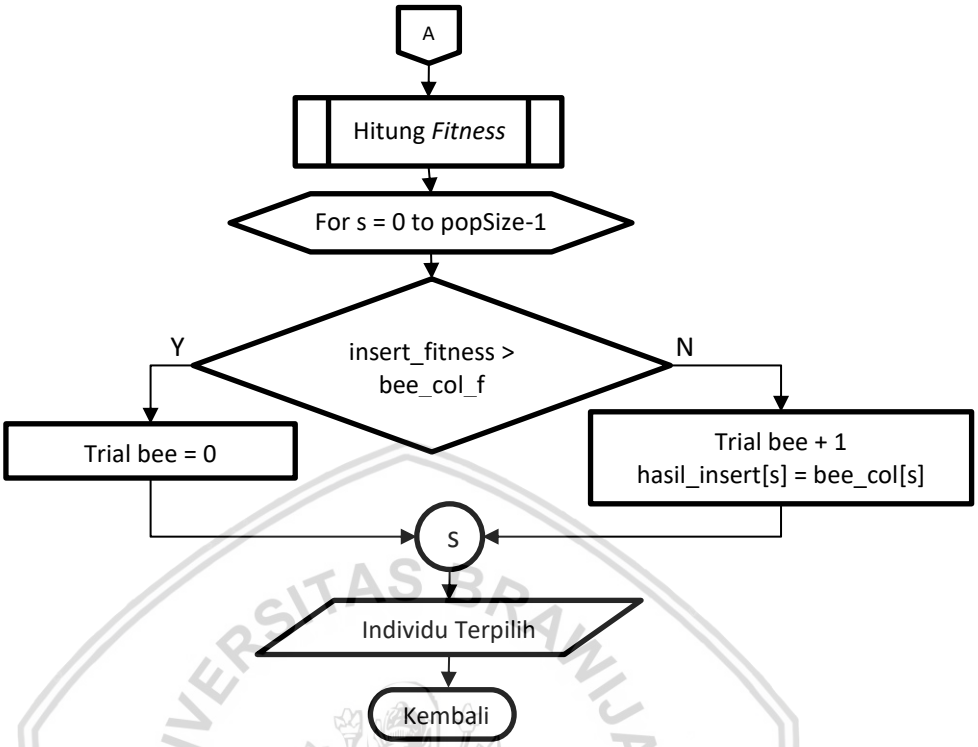
4.2.4.1 Fungsi *Insert Operator*

Pada Gambar 4.10 ditampilkan alur penghitungan *insert operator* yang dijabarkan sebagai berikut:

1. Inisialisasi
2. Nilai *insert operator* didapatkan secara random sesuai dengan jumlah *popSize* dan jumlah rutenya.
3. Setelah itu lakukan perulangan sebanyak jumlah *bee* dan disesuaikan dengan rutenya.
4. Jika sudah dihasilkan hasil *swap operator,* selanjutnya dihitung nilai *fitness* dari hasil *swap operator.*
5. Jika nilai *fitness* hasil *swap operator* lebih besar dari nilai *fitness* individu sebelumnya maka nilai trial kan di-*reset* menjadi 0. Tetapi jika nilai *fitness* dari hasil *swap operator* tidak lebih besar dari nilai *fitness* individu sebelumnya maka nilai trial akan ditambahkan 1.



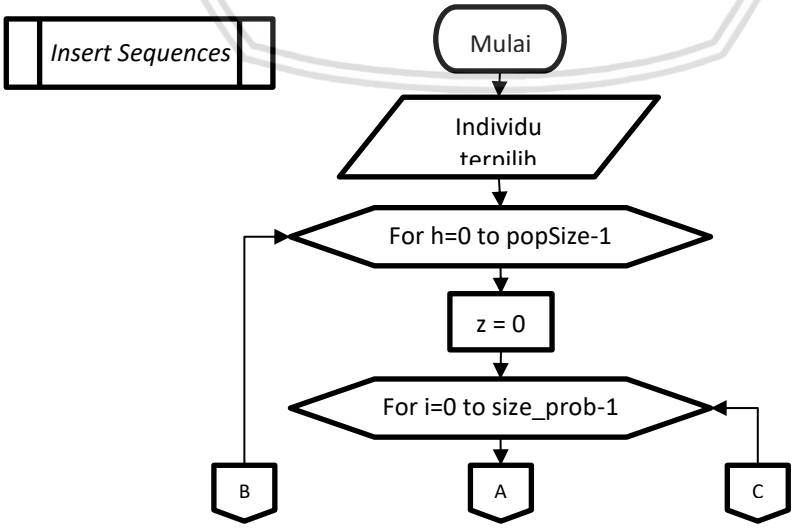

A
For p = 0 to popSize-1
For q = 0 to size_prob-1
a = insert_op[p][0]
b = insert_op[p][1]
a > b
N
Y
q<b ||
Y
hasil_insert[p][q] =
srcArray[p][q]
N
q==b
Y
hasil_insert[p][q] =
srcArray[p][a]
N
hasil_insert[p][q] =
bee_col[p][q-1]
a < b
Y
q>b ||
Y
hasil_insert[p][q] =
srcArray[p][q]
N
q==a
Y
hasil_insert[p][q] =
srcArray[p][a]
N
hasil_insert[p][q] =
bee_col[p][q-1]
q
p
A

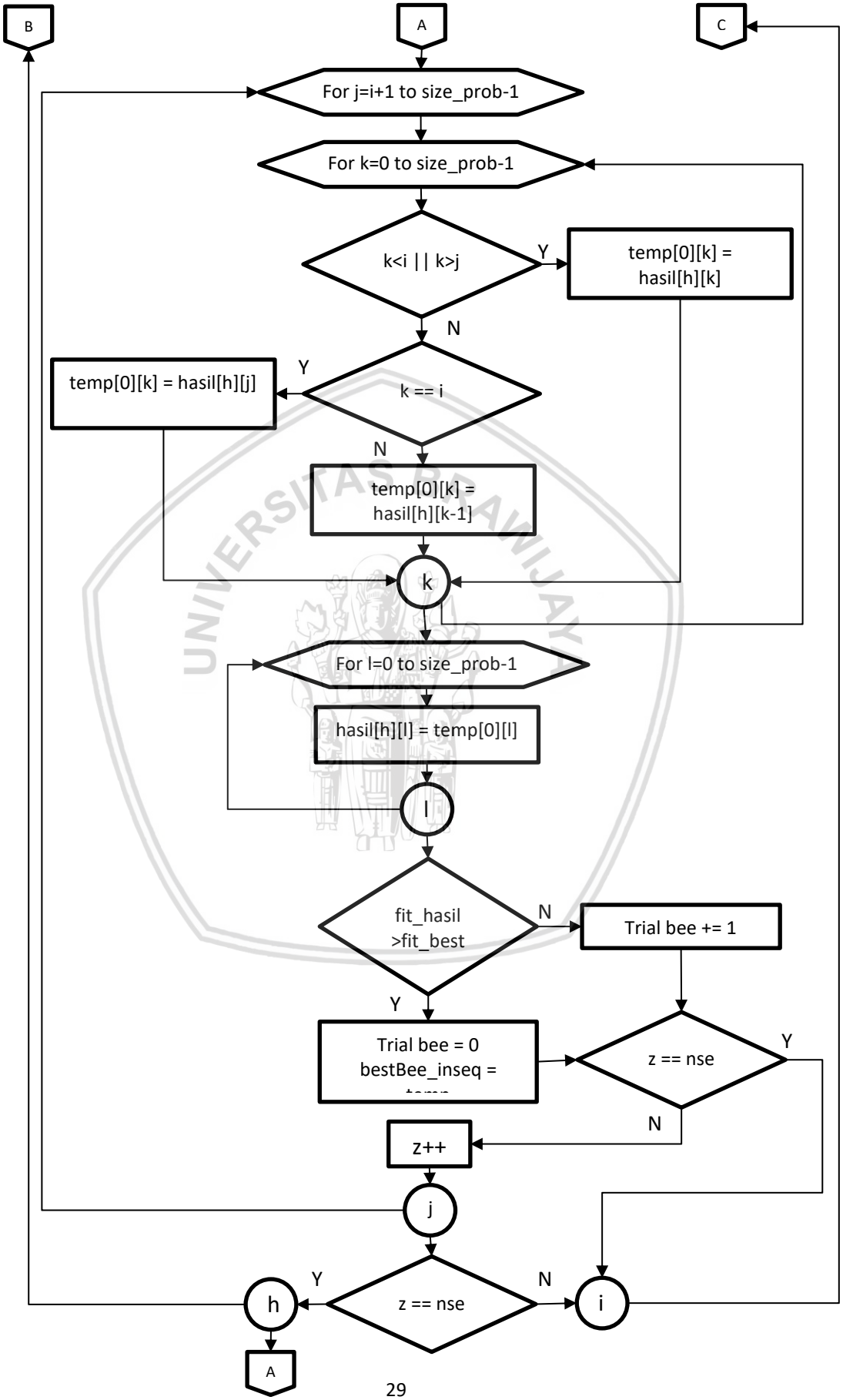


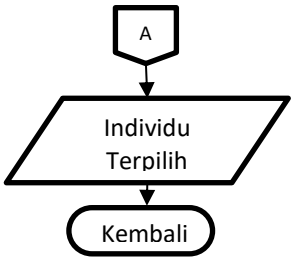
Gambar 4.11 *Insert Operator*

4.2.4.2 *Insert Sequences*

Individu untuk *insert sequences* didapatkan dari hasil *insert operator.* Pada fase insert *sequences,* rute tiap individu akan diacak lagi sebanyak *number of sequences*. Setiap selesai satu kali proses *insert*, akan dihitung nilai *fitness*-nya. Jika hasil nilai *fitness* baru lebih besar dari nilai *fitness* lama, maka nilai trial akan di-*reset* menjadi 0. Tetapi, jika tidak lebih besar nilai trial sebelumnya akan ditambah 1. Gambar 4.11 menunjukan alur perhitungan *insert sequences.*



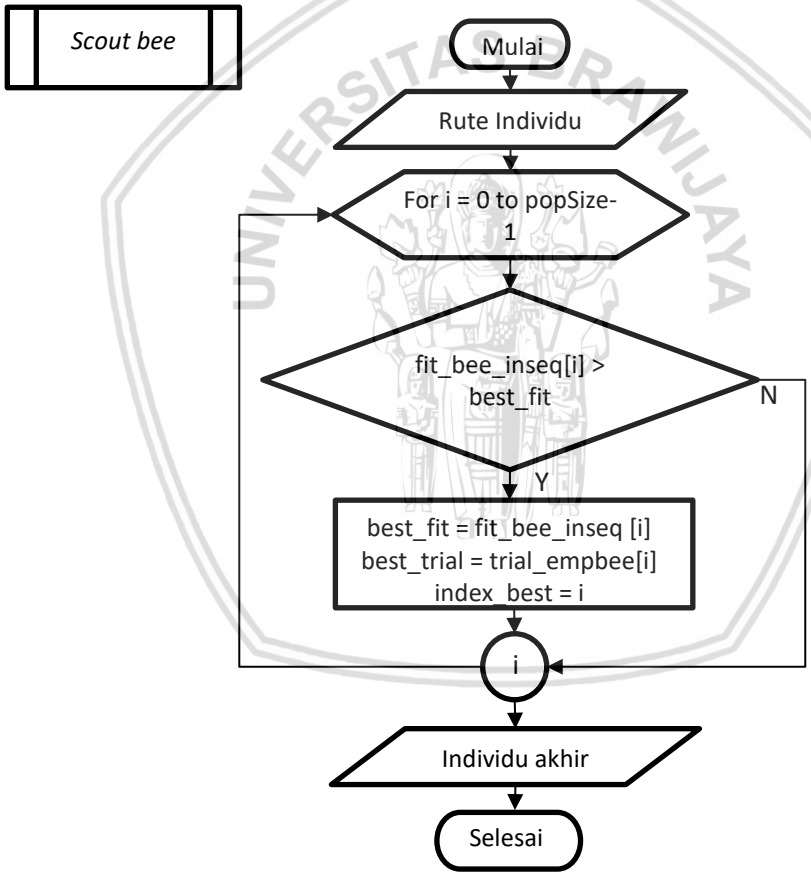




Gambar 4.12 *Insert Sequences*

4.2.5 Fase *Scout bee*

Pada fase *scout bee* individu awal dan akhir akan ditampilkan dan dibandingkan kemudian dari perbandingan keduanya akan dipilih yang menjadi solusi terbaik.



Gambar 4.13 *Scout Bee*

4.3 Manualisasi

Multiple Travelling Salesman Problem akan diselesaikan menggunakan Algoritme *Artificial Bee Colony.* Pertama yakni melakukan inisialisasi semua parameter yang dibutuhkan seperti *colony size, limit,* dan maksimum iterasi.

Tabel 4.1 adalah data yang digunakan berupa jarak antar lokasi pengiriman dimana titik 0 ada pusat distribusi.

Tabel 4.1 Jarak Antar Lokasi Pengiriman

Jarak	0	1	2	3	4	5	6	7	8	9	10
0	0,0	4,0	4,0	4,0	4,3	4,5	4,1	5,2	5,4	5,2	6,2
1	4,0	0,0	0,1	0,1	0,3	0,5	0,1	1,2	1,4	1,2	2,2
2	4,0	0,1	0,0	0,1	0,3	0,5	0,1	1,2	1,4	1,2	2,2
3	4,0	0,1	0,1	0,0	0,3	0,5	0,1	1,2	1,4	1,2	2,2
4	4,3	0,3	0,3	0,3	0,0	0,2	0,2	0,9	1,1	0,9	1,9
5	4,5	0,5	0,5	0,5	0,2	0,0	0,4	0,7	0,9	0,7	1,7
6	4,1	0,1	0,1	0,1	0,2	0,4	0,0	1,1	1,3	1,1	2,1
7	5,2	1,2	1,2	1,2	0,9	0,7	1,1	0,0	0,2	0,1	1,0
8	5,4	1,4	1,4	1,4	1,1	0,9	1,3	0,2	0,0	0,2	0,8
9	5,2	1,2	1,2	1,2	0,9	0,7	1,1	0,1	0,2	0,0	1,0
10	6,2	2,2	2,2	2,2	1,9	1,7	2,1	1,0	0,8	1,0	0,0

Misal pada sebuah kasus terdapat 2 orang *sales* yang akan melakukan pengiriman dengan masing-masing *sales* mengunjungi 3 tempat.

1. **Insialisasi Parameter**

   Total tempat tujuan pada satu koloni adalah 2*3 = 6 = *Size Problem*, *Colony Size* = *PopSize* = 4, maksimum iterasi = 2, *limit* = 5

2. ***Initial Solution***

   Dalam mendapatkan rute awal terdapat dua proses (Xue, et al., 2016) pertama adalah membangkitkan permutasi jumlah *n* titik dan kedua adalah membagi secara acak rangkaian rute ke dalam *m* jalur. *M* adalah jumlah *sales* dan setidaknya pada satu jalur terdapat satu titik. Hasil *initial solution* ditunjukkan pada Tabel 4.2.

Tabel 4.2 *Initial Solution*

Bee ke-*i*	Rute						Fitness
	Sales 1			Sales 2			
x1	8	6	1	9	2	3	0,0699301
x2	10	7	4	6	3	5	0,0724638
x3	4	1	10	7	9	8	0,075188
x4	9	7	8	4	1	5	0,0862069

3. **Iterasi = 1**

4. **Fase *Employeed Bee***

   Pada Tabel 4.3 ditunjukkan contoh perhitungan *swap operator.*

Tabel 4.3 *Swap Operator*

SO	Rute Awal						Rute Setelah Swap					
(1,3)	8	6	1	9	2	3	1	6	8	9	2	3
(2,3)	10	7	4	6	3	5	10	4	7	6	3	5
(1,4)	4	1	10	7	9	8	7	1	10	4	9	8
(3,6)	9	7	8	4	1	5	9	7	5	4	1	8

a. **Nilai *fitness* hasil *swap operator***

Tabel 4.4 merupakan tabel nilai *fitness* dari *swap operator* pada setiap individu dengan menggunakan Persamaan 2.1.

Tabel 4.4 Nilai *Fitness Swap Operator*

Bee ke-*i*	Rute						Fitness
	Sales 1			Sales 2			
x1	1	6	8	9	2	3	0,077519
x2	10	7	4	6	3	5	0,068027
x3	4	1	10	7	9	8	0,066667
x4	9	7	8	4	1	5	0,076923

b. **Perbandingan nilai *fitness***

Setelah didapatkan nilai *fitness* dari hasil *swap operator,* maka nilai *fitness* awal dibandingkan dengan nilai *fitness* hasil *swap operator.* Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka nilai *trial*-nya diberi nilai 0. Jika tidak maka nilai *trial* ditambah dengan 1. Perbandingan nilai *fitness* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Perbandingan Nilai *Fitness*

Bee ke-*i*	Awal		Hasil SO		Trial
	Total Jarak	Fitness	Total Jarak	Fitness	
x1	13,3	0,06993007	11,9	0,07751938	0
x2	12,8	0,072463768	13,7	0,068027211	1
x3	12,3	0,07518797	14	0,066666667	1
x4	10,6	0,086206897	12	0,076923077	1

Selanjutnya Individu yang memiliki nilai *fitness* terbaik dijadikan individu terpilih. Hasil individu terpilih diperlihatkan pada Tabel 4.6.

Tabel 4.6 Individu Terpilih

Bee ke-*i*	Rute					
x1	1	6	8	9	2	3
x2	10	7	4	6	3	5
x3	4	1	10	7	9	8
x4	9	7	8	4	1	5

c. Perhitungan *Swap Sequences*

Setelah terpilihnya individu terbaik hasil perbandingan nilai *fitness* awal dengan hasil *swap operator,* dilakukan perhitungan *swap sequences.* Cara melakukan *swap sequences* sama dengan *swap operator,* yang membedakan adalah tiap individu dilakukan *swap* sebanyak *number of sequences (Nse).* Nse didapatkan dari jumlah *size problem* dikali 2, sehingga jumlah Nse berdasarkan parameter adalah 6*2 = 12. Pada *swap sequences* terdapat trial yang akan diubah menjadi 0 jika nilai *fitness* baru lebih baik, jika tidak maka nilai trial akan ditambah 1. Tabel 4.7 menunjukkan perhitungan *swap sequences* pada individu pertama.

Tabel 4.7 *Swap Sequences* Individu Pertama

x1	1	6	8	9	2	3	fitness	0,077519	
(SO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	6	1	8	9	2	3	12,1	0,076336	0
1,3	8	1	6	9	2	3	13,4	0,069444	1
1,4	9	1	6	8	2	3	13,4	0,069444	2
1,5	2	1	6	8	9	3	11	0,083333	0
1,6	3	1	6	8	9	2	11	0,083333	1
2,3	3	6	1	8	9	2	11	0,083333	2
2,4	3	8	1	6	9	2	13,2	0,070423	3
2,5	3	9	1	6	8	2	13,2	0,070423	4
2,6	3	2	1	6	8	9	9,8	0,092593	0
3,4	3	2	6	1	8	9	9,8	0,092593	1
3,5	3	2	8	1	6	9	10,7	0,08547	2
3,6	3	2	9	1	6	8	10,7	0,08547	3
Terbaik	3	2	1	6	8	9	fitness	0,092593	

Tabel 4.8 menunjukkan perhitungan *swap sequences* pada individu kedua.

Tabel 4.8 *Swap Sequences* Individu Kedua

x2	10	7	4	6	3	5	fitness	0,072464	
(SO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	7	10	4	6	3	5	12,8	0,072464	1
1,3	4	10	7	6	3	5	11,9	0,077519	0
1,4	6	10	7	4	3	5	12,3	0,075188	1
1,5	3	10	7	4	6	5	12,1	0,076336	2
1,6	5	10	7	4	6	3	11,8	0,078125	0
2,3	5	7	10	4	6	3	10,8	0,084746	0
2,4	5	4	10	7	6	3	13	0,071429	1
2,5	5	6	10	7	4	3	13,4	0,069444	2
2,6	5	3	10	7	4	6	13,5	0,068966	3
3,4	5	3	7	10	4	6	14,5	0,064516	4
3,5	5	3	4	10	7	6	13,6	0,068493	5
3,6	5	3	6	10	7	4	13,2	0,070423	6
Terbaik	5	7	10	4	6	3	fitness	0,084746	

Tabel 4.9 menunjukkan perhitungan *swap sequences* pada individu ketiga.

Tabel 4.9 *Swap Sequences* Individu Ketiga

x3	4	1	10	7	9	8	fitness	0,075188	
(SO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	1	4	10	7	9	8	11,7	0,07874	0
1,3	10	4	1	7	9	8	13,9	0,067114	1
1,4	7	4	1	10	9	8	13,8	0,067568	2
1,5	9	4	1	10	7	8	13,8	0,067568	3
1,6	8	4	1	10	7	9	14,1	0,066225	4
2,3	8	1	4	10	7	9	14,4	0,064935	5
2,4	8	10	4	1	7	9	13,4	0,069444	6
2,5	8	7	4	1	10	9	13,7	0,068027	7
2,6	8	9	4	1	10	7	13,7	0,068027	8
3,4	8	9	1	4	10	7	14	0,066667	9
3,5	8	9	10	4	1	7	12,4	0,074627	10
3,6	8	9	7	4	1	10	12,5	0,074074	11
Terbaik	1	4	10	7	9	8	fitness	0,07874	

Tabel 4.10 menunjukkan perhitungan *swap sequences* pada individu keempat.

Tabel 4.10 *Swap Sequences* Individu Keempat

x4	9	7	8	4	1	5	fitness	0,086207	
(SO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	7	9	8	4	1	5	10,6	0,086207	2
1,3	8	9	7	4	1	5	10,8	0,084746	3
1,4	4	9	7	8	1	5	12,6	0,073529	4
1,5	1	9	7	8	4	5	12	0,076923	5
1,6	5	9	7	8	4	1	12,1	0,076336	6
2,3	5	7	9	8	4	1	12,1	0,076336	7
2,4	5	8	9	7	4	1	12	0,076923	8
2,5	5	4	9	7	8	1	12,4	0,074627	9
2,6	5	1	9	7	8	4	12,7	0,072993	10
3,4	5	1	7	9	8	4	12,7	0,072993	11
3,5	5	1	8	9	7	4	12,6	0,073529	12
3,6	5	1	4	9	7	8	10,8	0,084746	13
Terbaik	9	7	8	4	1	5	fitness	0,086207	

Selanjutnya dipilih individu terbaik dengan nilai fitness terbesar hasil *swap sequences.* Hasil individu terpilih diperlihatkan pada Tabel 4.11.

Tabel 4.11 Individu Terpilih

Bee ke-*i*	Rute						fitness	Trial
x1	3	2	1	6	8	9	0,0925926	3
x2	5	7	10	4	6	3	0,0847458	6
x3	1	4	10	7	9	8	0,0787402	11
x4	9	7	8	4	1	5	0,0862069	13

d. Seleksi Probabilitas

Sebelum memasuki fase *onlooker bee,* individu yang terpilih akan dihitung ptobabilitasnya berdasarkan nilai *fitness.* Tabel 4.12 menunjukkan pergitungan probabilitas.

Tabel 4.12 Probabilitas Individu

Bee ke-*i*	Rute						fitness	Prob
x1	3	2	1	6	8	9	0,0925926	0,27
x2	5	7	10	4	6	3	0,0847458	0,25
x3	1	4	10	7	9	8	0,0787402	0,23
x4	9	7	8	4	1	5	0,0862069	0,25
							0,34229	

5. *Onlooker Bee*

Setelah didapatkan probabilitasnya, selanjutnya hitung probabilitas kumulatif pada Tabel 4.13 untuk mendapatkan *range* pada setiap individu.

Tabel 4.13 Probabilitas Kumulatif

Bee ke-*i*	Prob	ProbCum	Range
x1	0,27	0,27	1-27
x2	0,25	0,52	28-52
x3	0,23	0,75	53-75
x4	0,25	1	76-100

Selanjutnya dilakukan seleksi pemilihan individu baru dengan me-*random* nilai *range*-nya pada tiap individu. Nilai *random* yang muncul akan dipilih sesuai dengan *range*-nya. Tabel 4.14 menunjukkan hasil seleksi.

Tabel 4.14 Seleksi Individu Baru

Bee ke-*i*	Random	Terpilih	Rute						Fitness
x1	83	4	9	7	8	4	1	5	0,086206897
x2	55	2	1	4	10	7	9	8	0,078740157
x3	33	2	5	7	10	4	6	3	0,084745763
x4	96	4	9	7	8	4	1	5	0,086206897

Setelah itu dilakukan perhitungan *insert operator* pada individu yang terpilih tersebut. Pada Tabel 4.15 ditunjukkan contoh perhitungan *insert operator.*

Tabel 4.15 *Insert Operator*

IO	Rute Awal						Rute Setelah Insert					
(1,3)	9	7	8	4	1	5	8	9	7	4	1	5
(2,3)	1	4	10	7	9	8	1	10	4	7	9	8
(1,4)	5	7	10	4	6	3	4	5	7	10	6	3
(3,6)	9	7	8	4	1	5	9	7	5	8	4	1

a. **Nilai *fitness* hasil *insert operator***

Tabel 4.16 merupakan tabel nilai *fitness* dari *insert operator* pada setiap individu dengan menggunakan Persamaan 2.1.

Tabel 4.16 Nilai *fitness insert operator*

Bee ke-*i*	Rute						Fitness
	Sales 1			Sales 2			
x1	8	9	7	4	1	5	0,084746
x2	1	10	4	7	9	8	0,068493
x3	4	5	7	10	6	3	0,068493
x4	9	7	5	8	4	1	0,072464

b. **Perbandingan Nilai *Fitness***

Setelah didapatkan nilai *fitness* dari hasil *insert operator,* maka nilai *fitness* awal dibandingkan dengan nilai *fitness* hasil *insert operator.* Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka nilai *trial*-nya diberi nilai 0. Jika tidak maka nilai *trial* ditambah dengan 1. Perbandingan nilai *fitness* dapat dilihat pada Tabel 4.17.

Tabel 4.17 Perbandingan Nilai *fitness*

Bee ke-*i*	Hasil Seleksi		Hasil IO		Trial
	Total Jarak	Fitness	Total Jarak	Fitness	
x1	10,6	0,086207	10,8	0,084746	14
x2	11,7	0,07874	13,9	0,068493	12
x3	10,8	0,084746	13,6	0,068493	7
x4	10,6	0,086207	12,8	0,072464	14

Selanjutnya Individu yang memiliki nilai *fitness* terbaik dijadikan individu terpilih. Hasil individu terpilih diperlihatkan pada Tabel 4.18.

Tabel 4.18 Individu Terpilih

Bee ke-*i*	Rute						Fitness
x1	9	7	8	4	1	5	0,086207
x2	1	4	10	7	9	8	0,07874
x3	5	7	10	4	6	3	0,084746
x4	9	7	8	4	1	5	0,086207

c. Perhitungan *Insert Sequences*

Setelah terpilihnya individu terbaik hasil perbandingan nilai *fitness* awal dengan hasil *insert operator, dilakukan perhitungan swap sequences.* Cara melakukan *insert sequences* sama dengan *insert operator,* yang membedakan adalah tiap individu dilakukan *insert* sebanyak *number of sequences (Nse).* Pada *insert sequences* terdapat *trial* yang akan diubah menjadi 0 jika nilai *fitness* baru lebih baik, jika tidak maka nilai *trial* akan ditambah 1. Tabel 4.19 menunjukkan perhitungan *insert sequences* pada individu pertama.

Tabel 4.19 *Insert Sequences* Individu Pertama

x1	9	7	8	4	1	5	fitness	0,086207	
(IO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	7	9	8	4	1	5	10,6	0,086207	15
1,3	8	7	9	4	1	5	10,8	0,084746	16
1,4	4	8	7	9	1	5	12,5	0,074074	17
1,5	1	4	8	7	9	5	11,4	0,080645	18
1,6	5	1	4	8	7	9	11	0,083333	19
2,3	5	4	1	8	7	9	10,7	0,08547	20
2,4	5	8	4	1	7	9	11,8	0,078125	21
2,5	5	7	8	4	1	9	11,2	0,081967	22
2,6	5	9	7	8	4	1	12,1	0,076336	23
3,4	5	9	8	7	4	1	11,8	0,078125	24
3,5	5	9	4	8	7	1	12,9	0,071942	25
3,6	5	9	1	4	8	7	12	0,076923	26
Terbaik	9	7	8	4	1	5	fitness	0,086207	

Tabel 4.20 menunjukkan perhitungan *insert sequences* pada individu kedua.

Tabel 4.20 *Insert Sequences* Individu Kedua

x2	1	4	10	7	9	8	fitness	0,07874	
(IO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	4	1	10	7	9	8	12,3	0,075188	13
1,3	10	4	1	7	9	8	13,9	0,067114	14
1,4	7	10	4	1	9	8	13,5	0,068966	15
1,5	9	7	10	4	1	8	12,3	0,075188	16
1,6	8	9	7	10	4	1	14,1	0,066225	17
2,3	8	7	9	10	4	1	14,1	0,066225	18
2,4	8	10	7	9	4	1	13,6	0,068493	19
2,5	8	4	10	7	9	1	14,9	0,062893	20
2,6	8	1	4	10	7	9	14,4	0,064935	21
3,4	8	1	10	4	7	9	14,3	0,065359	22
3,5	8	1	7	10	4	9	17	0,055556	23
3,6	8	1	9	7	10	4	16,1	0,05848	24
Terbaik	4	1	10	7	9	8	fitness	0,07874	

Tabel 4.21 menunjukkan perhitungan *insert sequences* pada individu ketiga.

Tabel 4.21 *Insert Sequences* Individu Ketiga

x3	5	7	10	4	6	3	fitness	0,084746	
(IO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	7	5	10	4	6	3	12,2	0,075758	8
1,3	10	7	5	4	6	3	12,5	0,074074	9
1,4	4	10	7	5	6	3	12,2	0,075758	10
1,5	6	4	10	7	5	3	12,6	0,073529	11
1,6	3	6	4	10	7	5	12,2	0,075758	12
2,3	3	4	6	10	7	5	12,4	0,074627	13
2,4	3	10	4	6	7	5	14	0,066667	14
2,5	3	7	10	4	6	5	11,1	0,082645	15
2,6	3	5	7	10	4	6	13,5	0,068966	16
3,4	3	5	10	7	4	6	12,5	0,074074	17
3,5	3	5	4	10	7	6	13	0,071429	18
3,6	3	5	6	4	10	7	12,1	0,076336	19
Terbaik	5	7	10	4	6	3	fitness	0,084746	

Tabel 4.22 menunjukkan perhitungan *insert sequences* pada individu keempat.

Tabel 4.22 *Insert Sequences* Individu Keempat

x4	9	7	8	4	1	5	fitness	0,086207	
(IO)	Sales 1			Sales 2			Total Jarak	Fitness	Trial
	Rute								
1,2	7	9	8	4	1	5	10,6	0,086207	15
1,3	8	7	9	4	1	5	10,8	0,084746	16
1,4	4	8	7	9	1	5	12,5	0,074074	17
1,5	1	4	8	7	9	5	11,4	0,080645	18
1,6	5	1	4	8	7	9	11	0,083333	19
2,3	5	4	1	8	7	9	10,7	0,08547	20
2,4	5	8	4	1	7	9	11,8	0,078125	21
2,5	5	7	8	4	1	9	11,2	0,081967	22
2,6	5	9	7	8	4	1	12,1	0,076336	23
3,4	5	9	8	7	4	1	11,8	0,078125	24
3,5	5	9	4	8	7	1	12,9	0,071942	25
3,6	5	9	1	4	8	7	12	0,076923	26
Terbaik	9	7	8	4	1	5	fitness	0,086207	

Selanjutnya dipilih individu terbaik dengan nilai fitness terbesar hasil *insert sequences.* Hasil individu terpilih diperlihatkan pada Tabel 4.23.

Tabel 4.23 Individu Terpilih *Insert Sequences*

Bee ke-*i*	Rute						fitness	Trial
x1	9	7	8	4	1	5	0,086207	26
x2	1	4	10	7	9	8	0,07874	24
x3	5	7	10	4	6	3	0,084746	19
x4	9	7	8	4	1	5	0,086207	26

d. Perbandingan Nilai *Fitness* Awal dengan Nilai *Fitness Insert Sequences*

Perbandingan dilakukan untuk mendapatkan solusi terbaik pada iterasi pertama ini. Tabel 4.24 menunjukkan nilai perbandingan.

Tabel 4.24 Perbandingan Nilai *Fitness*

Bee ke-*i*	Awal							Hasil IS						
	Rute						Fitness	Rute						Fitness
x1	8	6	1	9	2	3	0,06993007	9	7	8	4	1	5	0,086207
x2	10	7	4	6	3	5	0,07246377	1	4	10	7	9	8	0,07874
x3	4	1	10	7	9	8	0,07518797	5	7	10	4	6	3	0,084746
x4	9	7	8	4	1	5	0,0862069	9	7	8	4	1	5	0,086207

Dari perbandingan tersebut didapatkan solusi terbaik untuk iterasi pertama yaitu

x1	9	7	8	4	1	5	0,0862069

6. ***Scout Bee***

Sebelum masuk ke iterasi kedua, *scout bee* akan memeriksa nilai *fitness* rute awal dengan rute nilai *fitness* hasil *insert sequences.* Individu yang memiliki M(Max(*trial*)) > Limit, maka ganti individu yang tidak ada perbaikan nilai *fitness* dengan individu baru secara *random*, kemudian reset *trial* menjadi 0, jika M(Max(*trial*)) > Limit dan terdapat individu yang mengalami perbaikan nilai *fitness*, maka tidak perlu digantikan dengan individu baru dan reset *trial* menjadi 0. Jika M(Max(*trial*)) < Limit, maka tidak perlu digantikan dengan individu baru dan nilai *trial* tidak perlu di-*reset*. Sebelumnya sudah diinisialisasi nilai limitnya adalah 5, selanjutnya nilai M(Max(Iterasi)) ditentukan dengan melihat nilai *trial* tertinggi, dalam kasus ini adalah 8. Tabel 4.25 contoh perhitungannya.

Tabel 4.25 Penentuan Individu Baru

Bee ke-*i*	Awal							Hasil IS							Trial	Perbaikan
	Rute						Fitness	Rute						Fitness		
x1	8	6	1	9	2	3	0,06993007	9	7	8	4	1	5	0,086207	26	Ada
x2	10	7	4	6	3	5	0,07246377	1	4	10	7	9	8	0,07874	24	Ada
x3	4	1	10	7	9	8	0,07518797	5	7	10	4	6	3	0,084746	19	Ada
x4	9	7	8	4	1	5	0,0862069	9	7	8	4	1	5	0,086207	26	Tidak

Selanjutnya didapatkan individu baru untuk iterasi kedua pada Tabel 4.26.

Tabel 4.26 Individu Baru

Bee ke-*i*	Rute						Trial
x1	9	7	8	4	1	5	0
x2	1	4	10	7	9	8	0
x3	5	7	10	4	6	3	0
x4	6	4	5	7	9	2	0

4.4 Perancangan Pengujian

Tujuan dari dilakukannya pengujian adalah untuk mengetahui nilai yang optimal untuk mencapai keberhasilan metode terhadap sistem optimasi rute pendistribusian es batu. Adapun pengujian yang dilakukan adalah jumlah iterasi yang mencapai titik konvergensi, jumlah *bee colony,* jumlah *limit,* dan jumlah *size problem.*

4.4.1 Perancangan Pengujian Jumlah *Size Problem*

Pengujian jumlah *size problem* terhadap hasil solusi terbaik dilakukan untuk mengetahui tingkat konvergensi sistem. Perancangannya ditunjukkan pada Tabel 4.27

Tabel 4.27 Rancangan Pengaruh Jumlah *Size Problem*

Jumlah *Size Problem*	Hasil Percobaan Ke-*i*					Rata-rata
	1	2	3	4	5	
23						
24						
25						
…						
32						

4.4.2 Perancangan Pengujian Jumlah *Pop Size*

Pengujian jumlah *pop size* dilakukan untuk mengetahui jumlah *pop size* untuk mendapat rata-rata nila *fitness* optimal. Perancangannya ditunjukkan pada Tabel 4.28

Tabel 4.28 Rancangan Pengaruh Jumlah *Pop Size*

Jumlah *Pop Size*	Hasil Percobaan Ke-*i*					Rata-rata
	1	2	3	4	5	
10						
20						
30						
…						
100						

4.4.3 Perancangan Pengujian Jumlah *Limit*

Pengujian banyak *limit* dilakukan untuk mengetahui jumlah *limit* yang optimal. Perancangannya ditunjukkan pada Tabel 4.29

Tabel 4.29 Rancangan Pengaruh Jumlah *Limit*

Jumlah *Limit*	Hasil Percobaan Ke-*i*					Rata-rata
	1	2	3	4	5	
10						
20						
30						
…						
100						

4.4.4 Perancangan Pengujian Konvergensi

Perancangan pengujian konvergensi dilakukan untuk mengetahui banyak iterasi yang diperlukan untuk melihat tingkat konvergensi dari sistem. Iterasi yang digunakan adalah dari 100 hingga 1000 dengan kelipatan 100. Perancangannya ditunjukkan pada Tabel 4.30

Tabel 4.30 Rancangan Pengaruh Jumlah Iterasi

Iterasi	Hasil Percobaan Ke-*i*					Rata-rata
	1	2	3	4	5	
100						
200						
300						
…						
1000						

BAB 5 IMPLEMENTASI

Pada bab implementasi ini akan membahas tentang algoritme *Artificial Bee Colony* (ABC) untuk optimasi rute *multiple traveling salesman problem* pada distribusi es batu berdasarkan perancangan yang sudah dilakukan pada bab sebelumnya.

5.1 Lingkungan Implementasi

Lingkungan yang dimaksud adalah lingkungan pada saat melakukan implementasi seperti lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.1 Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan saat melakukan implementasi adalah:

1. Laptop ASUS A456UR
2. Prosesor Intel Core i5-6200 2.30Ghz
3. RAM 4 GB SDDR 4
4. VGA Nvidia Geforce 930MX 2 GB
5. *Harddisk* 1 TB

5.1.2 Lingkungan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan saat melakukan implementasi adalah:

1. Sistem Operasi *Windows* 10 Home 64-bit
2. Netbeans IDE 8.01
3. Bahasa Pemrograman Java

5.2 Implementasi Algoritme

Pada sub-bab ini akan memiliki sub-bab – sub-bab yang mewakili proses utama dan dibahas berdasarkan perancangan sistem pada bab sebelumnya dalam melakukan implementasi algoritma *Artificial Bee Colony* (ABC) untuk optimasi rute *multiple traveling salesman problem* pada distribusi es batu. Implementasi program dibuat dengan Bahasa pemrograman java. Program yang diimplementasi memiliki 4 proses utama yaitu proses fase *initial*, proses fase *employeed bee*, proses fase *onlooker bee,* dan proses fase *scout bee*.

5.2.1 Proses Fase Initial

Fase *initial* merupakan fase untuk mendapat nilai solusi awal atau *initial solution* pada tiap *employed bee*. Pada fase ini akan dilakukan penentuan rute

secara random atau *generated route* dan setelah itu dihitung jaraknya untuk mendapatkan nilai *fitness* masing-masing rute.

5.2.1.1 Fungsi *Generated Route*

Proses *generated route* memiliki fungsi untuk mendapatkan rute secara random berdasarkan hasil inputan berupa jumlah rute dan jumlah *pop size.*

```
1     public static void generatedRoute(int popSize, int
      size_prob) {
2             int[][] tempArray = new int[popSize][size_prob];
3             Arrays.fill(slot, false);
4             for (int i = 0; i < size_prob; i++) {
5                 array_route[0][i] =
      rand.nextInt(sheet.getPhysicalNumberOfRows());
6                 if (array_route[0][i] == 0) {
7                     i--;
8                 } else if (slot[array_route[0][i]] == true) {
9                     i--;
10                } else {
11                    slot[array_route[0][i]] = true;
12                }
13            }
14            for (int j = 1; j < popSize; j++) {
15               System.arraycopy(array_route[0], 0,
      tempArray[j], 0, array_route[0].length);
16                acakRute(tempArray);
17                System.arraycopy(tempArray[j], 0,
      array_route[j], 0, tempArray[j].length);
18            }
19     }
```

Source Code 5.1 *Listing Code Generated Route*

Penjelasan *source code* 5.1 adalah:

1. Baris 5 menunjukkan penentuan rute untuk *colony* pertama
2. Baris 6-12 menunjukkan proses pengecekan rute apabila terjadi duplikasi
3. Baris 15-17 menunjukan proses penentuan rute untuk *colony* setelah *colony* pertama.

5.2.1.2 Fungsi Hitung *Fitness*

```
1     public static double[] hitungFitness(int[][] route_fit) {
2             fitness = new double[route_fit.length];
3             int src, dst;
4             for (int i = 0; i < route_fit.length; i++) {
5                 double total_jarak = 0;
6                 for (int j = 0; j < route_fit[i].length; j++) {
7                     if (j == 0 || j == (route_fit[i].length / 2)) {
8                         src = 1;
9                         dst = route_fit[i][j];
10                    } else if (j == (route_fit[i].length - 1) || j
   == ((route_fit[i].length / 2) - 1)) {
11                        src = route_fit[i][j - 1]+1;
12                        dst = route_fit[i][j];
13                    } else {
14                        src = route_fit[i][j - 1]+1;
```

```
15                          dst = route_fit[i][j];
16                      }
17                      total_jarak += value[src][dst];
18                  }
19                  fitness[i] = 1 / (1 + total_jarak);
20              }
21              return fitness;
22      }
```

Source Code 5.2 *Listing Code* Hitung *Fitness*

Penjelasan *Source Code* 5.2 adalah:

1. Baris 5 menjelaskan setiap perulangan i, total_jarak di set menjadi 0
2. Baris 7-10 menunjukkan jika posisi index j berada pada posisi 0 atau pada array dibagi banyak sales maka jarak awalnya akan dimulai dari 0.
3. Baris 10- 13 menunjukkan jika posisi index j berada pada akhir Panjang array maka perhitungan jaraknya berada pada titik rute yang sama
4. Baris 13- 16 menunjukkan perhitungan jarak dari index j-1 sampai index j
5. Baris 17 menunjukkan perhitungan total jarak
6. Baris 19 menunjukkan perhitungan nilai *fitness*

5.2.1.3 Fungsi Acak Rute

```
1   public static void acakRute(int[][] arShuffle) {
2           for (int i = 0; i < popSize; i++) {
3               for (int j = arShuffle[i].length - 1; j > 0; j--) {
4                   int index = rand.nextInt(j + 1);
5                   int a = arShuffle[i][index];
6                   arShuffle[i][index] = arShuffle[i][j];
7                   arShuffle[i][j] = a;
8               }
9           }
10      }
```

Source Code 5.3 *Listing Code* Acak Rute

Penjelasan *Source Code* 5.3 adalah:

1. Baris 5-7 menunjukkan proses *shuffling* atau pengacakan

5.2.2 Proses Fase *Employeed Bee*

Pada fase *employeed bee* terdapat 2 proses yaitu proses *swap operator* dan proses *swap sequences.*

5.2.2.1 Fungsi *Swap Operator*

Pada proses ini setiap titik rute akan dilakukan *swapping* berdasarkan nilai *operator.*

```
1   public static int[][] swapOperator(int[][] route_b, double[]
    ran_col_f) {
2           int[][] swap_op = new int[popSize][2];
3           hasil_swap = new int[popSize][size_prob];
4           double[] swap_fit;
```

```
5               for (int i = 0; i < popSize; i++) {
6                   Arrays.fill(slot, false);
7                   for (int j = 0; j < 2; j++) {
8                       swap_op[i][j] = rand.nextInt(size_prob - 1);
9                       if (slot[swap_op[i][j]] == true) {
10                          j--;
11                      } else {
12                          slot[swap_op[i][j]] = true;
13                      }
14                  }
15              }
16
17              System.out.println("===== SWAP OPERATOR =====");
18              System.out.println("Hasil swap:");
19              for (int p = 0; p < popSize; p++) {
20                  int a = swap_op[p][0], b = swap_op[p][1];
21                  if (swap_op[p][0] > swap_op[p][1]) {
22                      swap_op[p][0] = b;
23                      swap_op[p][1] = a;
24                  }
25                  System.out.print("X" + (p + 1) + " (" +
    (swap_op[p][0] + 1) + "," + (swap_op[p][1] + 1) + ") | \t");
26                  for (int q = 0; q < size_prob; q++) {
27                      int Op1 = swap_op[p][0];
28                      int Op2 = swap_op[p][1];
29                      if (q == Op1) {
30                          hasil_swap[p][q] = route_b[p][Op2];
31                      } else if (q == Op2) {
32                          hasil_swap[p][q] = route_b[p][Op1];
33                      } else {
34                          hasil_swap[p][q] = route_b[p][q];
35                      }
36                      System.out.print(hasil_swap[p][q] + "\t");
37                  }
38                  System.out.println("");
39              }
40              System.out.println("");
41              swap_fit = hitungFitness(hasil_swap);
42              for (int r = 0; r < popSize; r++) {
43                  System.out.print("Fitness X" + (r + 1) + " | ");
44                  System.out.println(swap_fit[r]);
45              }
46              trial_empbee = new int[popSize];
47              System.out.println("");
48              System.out.println("Individu Terpilih:");
49              for (int s = 0; s < popSize; s++) {
50                  if (swap_fit[s] > ran_col_f[s]) {
50                      trial_empbee[s] = 0;
52                  } else {
53                      trial_empbee[s] = 1;
54                      hasil_swap[s] = route_b[s];
55                  }
56                  for (int t = 0; t < size_prob; t++) {
57                      System.out.print(hasil_swap[s][t] + "\t");
58                  }
59                  System.out.print("Trial : " + trial_empbee[s] + " ");
60                  System.out.println("");
61              }
```

```
62              System.out.println("");
63              return hasil_swap;
64      }
```

Source Code 5.4 *listing Code Swap Operator*

Penjelasan *source code 5.4* adalah:

1. Baris 5-15 menjelaskan proses untuk menentukan nilai *operator swap* dan dilakukan pengecekan agar tidak terjadi duplikasi
2. Baris 26-37 menjelaskan proses penukaran rute berdasarkan *operator swap*
3. Baris 29-31 menjelaskan proses penukaran jika nilai q = Op1 maka rute pada
4. Baris 41 menjelaskan perhitungan nilai *fitness* dari rute hasil *swap*
5. Baris 49-50 menjelaskan proses penentuan nilai *trial*. Jika nilai *fitness* hasil *swap* lebih besar dari nilai *fitness* sebelumnya maka nilai *trial* di set menjadi 0, tetapi jika lebih kecil maka nilai *trial* akan ditambah 1.

5.2.2.2 Fungsi *Swap Sequences*

Pada fungsi ini akan dilakukan *swapping* pada masing-masing rute sebanyak *number of sequences (nse).*

```
1   public static int[][] swapSeq(int[][] bee_swap, double[] fit_awal,
    int index) {
2           int[][] hasil = new int[popSize][size_prob];
3           double fit_hasil = 0;
4           fit_best = fit_awal;
5           int nse = 2 * size_prob;
6           int[][] swap_op = new int[nse][2];
7           int p = 0, a = 0, b = 0;
8           int[][] temp = new int[1][6];

9           for (int i = 0; i < popSize; i++) {
10              for (int j = 0; j < size_prob; j++) {
11                  hasil[i][j] = bee_swap[i][j];
12                  best_bee[i][j] = bee_swap[i][j];
13              }
14          }

15          System.out.println("===== SWAP SEQUENCES =====");
16          for (int h = 0; h < popSize; h++) {
17              int margin = 0;
18              temp[0] = bee_swap[h];
19              for (int i = 0; i < size_prob; i++) {
20                  for (int j = i + 1; j < size_prob; j++) {
21                      a = temp[0][i];
22                      b = temp[0][j];
23                      temp[0][i] = b;
24                      temp[0][j] = a;
25                      if (fit_best[h] < fit_hasil) {
26                          trial_empbee[h] = 0;
27                          for (int k = 0; k < size_prob; k++) {
28                              best_bee[h][k] = temp[0][k];
```

```
29                              }
30                              fit_best[h] = fit_hasil;
31                          } else {
32                              trial_empbee[h]++;
33                          }
34                          if (margin == nse) {
35                              break;
36                          }
37                          margin++;
38                      }
39                      if (margin == nse) {
40                          break;
41                      }
42                  }
43                  index++;
44              }
45              System.out.println("Route Best : ");
46              for (int k = 0; k < popSize; k++) {
47                  for (int l = 0; l < size_prob; l++) {
48                      System.out.print(best_bee[k][l] + "\t");
49                  }
50                  System.out.print("Fitness: " + fit_best[k] + "\tTrial
    : " + trial_empbee[k]);
51                  System.out.println("");
52              }
53              System.out.println("");
54
55              return best_bee;
56          }
```

Source Code 5.5 *Listing Code Swap Sequences*

Penjelasan *source code 5.5* adalah:

1. Baris 17 menjelaskan nilai margin yang diset 0 untuk menjadi pembatas agar perulangan tidak keluar dari index
2. Baris 18 menjelaskan nilai array temp yang didapatkan dari nilai bee_swap
3. Baris 21-24 menjelaskan proses penukaran atau *swap* pada rute
4. Baris 27-35 menjelaskan proses penentuan nilai *trial*. Jika nilai *fitness* hasil *swap* lebih besar dari nilai *fitness* sebelumnya maka nilai *trial* di set menjadi 0, tetapi jika lebih kecil maka nilai *trial* akan ditambah 1
5. Baris 34-37 menjelaskan pengecakan nilai margin apakah sudah sama dengan nse. Apabila iya maka perulangan akan dihentikan, jika tidak nilai margin akan ditambah 1
6. Baris 39-41 menjelaskan pengecakan nilai margin apakah sudah sama dengan nse. Apabila iya maka perulangan akan dihentikan.

5.2.3 Fungsi Seleksi Probabilitas

Sebelum masuk fase *onlooker bee,* rute terpilih dari hasil *swap sequences* harus diseleksi menggunakan metode *roulette wheel* yaitu metode seleksi yang

mengambil nilai probabilitas berdasarkan nilai *fitness*nya yang dibagi dengan total nilai *fitness* semua rute.

```
1   public static int[][] seleksi(int[][] beeRoute, double[] beeRouteFit) {
2           double[] prob = new double[popSize];
3           double[] prob_kum = new double[popSize];
4           double[] random = new double[popSize];
5           int[] temp_trial = new int[popSize];
6           double total_fitness = 0;
7
8           for (int i = 0; i < popSize; i++) {
9               total_fitness += beeRouteFit[i];
10          }
11
12          System.out.println("===== SELEKSI ROULETTE WHEEL =====");
13          System.out.println("Probabilitas: ");
14          for (int j = 0; j < popSize; j++) {
15              prob[j] = beeRouteFit[j] / total_fitness;
16              if (j == 0) {
17                  prob_kum[j] = prob[j];
18                  System.out.printf("Prob %.2f | Kumulatif %.2f | Range
    0-%.2f \n", prob[j], prob_kum[j], prob_kum[j]);
19              } else {
20                  prob_kum[j] = prob_kum[j - 1] + prob[j];
21                  System.out.printf("Prob %.2f | Kumulatif %.2f | Range
    %.2f-%.2f \n", prob[j], prob_kum[j], prob_kum[j - 1] + 0.01,
    prob_kum[j]);
22              }
23          }
24
25          for (int tmp = 0; tmp < popSize; tmp++) {
26              temp_trial[tmp] = trial_empbee[tmp];
27          }
28
29          System.out.println("");
30          System.out.println("Individu Terpilih:");
31          for (int k = 0; k < popSize; k++) {
32              random[k] = Math.random();
33              for (int l = 0; l < popSize; l++) {
34                  if (random[k] <= prob_kum[l]) {
35                      for (int i = 0; i < size_prob; i++) {
36                          seleksiBee[k][i] = beeRoute[l][i];
37                          array_route[k][i] = ar_seleksi[l][i];
38                          bestBee_inseq[k][i] = seleksiBee[k][i];
39                      }
40                      fit_best = hitungFitness(seleksiBee);
41                      trial_empbee[k] = temp_trial[l];
42                      System.out.printf("%.2f | ", random[k]);
43                      for (int m = 0; m < size_prob; m++) {
44                          System.out.print(seleksiBee[k][m] + "\t");
45                      }
46                      beeRouteFit = hitungFitness(seleksiBee);
47                      System.out.print(beeRouteFit[k]);
48                      System.out.println("");
49                      break;
50                  }
51              }
52          }
53          return best_bee;
```

```
54      }
```

Source Code 5.6 *Listing Code* Seleksi Probabilitas

Penjelasan *source code* 5.6 adalah:

1. Baris 8-10 menjelaskan proses penjumlahan semua nilai *fitness*
2. Baris 14-23 menjelaskan proses perhitungan probabilitas
3. Baris 34-52 menjelaskan proses seleksi.

5.2.4 Proses Fase *Onlooker Bee*

Pada fase *onlooker bee* terdapat 2 proses yaitu *insert operator* dan *insert sequences.*

5.2.4.1 Fungsi *Insert Operator*

Pada fungsi ini akan dilakukan proses *insert* pada masing-masing rute berdasarkan nilai *operator.*

```
1   public static int[][] insertOp(int[][] bee_col, double[] bee_col_f) {
2           int[][] insert_op = new int[popSize][2];
3           int[][] srcArray = new int[popSize][size_prob];
4           double[] insert_fitness;
5           int a, b;
6
7           for (int i = 0; i < bee_col.length; i++) {
8               for (int j = 0; j < bee_col[i].length; j++) {
9                   srcArray[i][j] = bee_col[i][j];
10              }
11          }
12
13          for (int i = 0; i < popSize; i++) {
14              Arrays.fill(slot, false);
15              for (int j = 0; j < 2; j++) {
16                  insert_op[i][j] = rand.nextInt(size_prob - 1);
17                  if (slot[insert_op[i][j]] == true) {
18                      j--;
19                  } else {
20                      slot[insert_op[i][j]] = true;
21                  }
22              }
23          }
24
25          System.out.println("");
26          System.out.println("===== INSERT OPERATOR =====");
27          System.out.println("Hasil Insert: ");
28          for (int p = 0; p < popSize; p++) {
29              System.out.print("X" + (p + 1) + " (" + (insert_op[p][0] + 1) + "," + (insert_op[p][1] + 1) + ") | ");
30              for (int q = 0; q < size_prob; q++) {
31                  a = insert_op[p][0];
32                  b = insert_op[p][1];
33                  if (a > b) {
34                      if (q > a || q < b) {
35                          hasil_insert[p][q] = srcArray[p][q];
36                      } else if (q == b) {
```

```
37                          hasil_insert[p][q] = srcArray[p][a];
38                      } else {
39                          hasil_insert[p][q] = srcArray[p][q - 1];
40                      }
41                  } else if (b > a) {
42                      if (q < a || q > b) {
43                          hasil_insert[p][q] = srcArray[p][q];
44                      } else if (q == a) {
45                          hasil_insert[p][q] = srcArray[p][b];
46                      } else {
47                          hasil_insert[p][q] = srcArray[p][q - 1];
48                      }
49                  }
50                  System.out.print(hasil_insert[p][q] + "\t");
50              }
52              System.out.println("");
53          }
54          System.out.println("");
55          System.out.println("Fitness Insert Operator: ");
56          insert_fitness = hitungFitness(hasil_insert);
57          for (int r = 0; r < popSize; r++) {
58              System.out.println(insert_fitness[r] + " ");
59          }
60          System.out.println("");
61          System.out.println("Individu Terpilih: ");
62          for (int s = 0; s < popSize; s++) {
63              if (insert_fitness[s] > bee_col_f[s]) {
64                  trial_empbee[s] = 0;
65              } else {
66                  trial_empbee[s]++;
67                  hasil_insert[s] = bee_col[s];
68              }
69              for (int t = 0; t < size_prob; t++) {
70                  System.out.print(hasil_insert[s][t] + "\t");
71              }
72              System.out.print("Trial : " + trial_empbee[s] + "");
73              System.out.println("");
74          }
75          return hasil_insert;
76      }
```

Source Code 5.7 *Listing Code Insert Operator*

Penjelasan *source code* 5.7 adalah:

1. Baris 13-23 menjelaskan proses untuk menentukan nilai *operator insert* dan dilakukan pengecekan agar tidak terjadi duplikasi
2. Baris 30-51 menjelaskan proses *insertion* rute berdasarkan *operator.*
3. Baris 56 menjelaskan perhitungan nilai *fitness* dari rute hasil *insert*
4. Baris 62-74 menjelaskan proses penentuan nilai *trial*. Jika nilai *fitness* hasil *insert* lebih besar dari nilai *fitness* sebelumnya maka nilai *trial* di set menjadi 0, tetapi jika lebih kecil maka nilai *trial* akan ditambah 1.

5.2.4.2 Fungsi *Insert Sequences*

Pada fungsi ini akan dilakukan *insertion* pada masing-masing rute sebanyak *number of sequences (nse).*

```
1   public static int[][] insertSequences(int[][] bee_awal, double[]
    fit_awal) {
2           int[][] hasil = new int[popSize][size_prob];
3           double fit_hasil = 0;
4           fit_best = fit_awal;
5           int nse = 2 * size_prob;
6           int[][] temp = new int[1][6];
7
8           for (int i = 0; i < bee_awal.length; i++) {
9               for (int j = 0; j < bee_awal[i].length; j++) {
10                  hasil[i][j] = bee_awal[i][j];
11              }
12          }
13          System.out.println("");
14          System.out.println("== INSERT SEQUENCES ==");
15          for (int h = 0; h < popSize; h++) {
16              int margin = 0;
17              for (int cl = 0; cl < size_prob; cl++) {
18                  temp[0][cl] = hasil[h][cl];
19              }
20              for (int i = 0; i < size_prob; i++) {
21                  for (int j = i + 1; j < size_prob; j++) {
22                      for (int k = 0; k < size_prob; k++) {
23                          if (k < i || k > j) {
24                              temp[0][k] = hasil[h][k];
25                          } else if (k == i) {
26                              temp[0][k] = hasil[h][j];
27                          } else { //buat geser
28                              temp[0][k] = hasil[h][k - 1];
29                          }
30                      }
31                      for (int l = 0; l < size_prob; l++) {
32                          hasil[h][l] = temp[0][l];
33                      }
34                      fit_hasil = fitnessSeq(temp);
35                      if (fit_hasil > fit_best[h]) {
36                          trial_empbee[h] = 0;
37                          for (int k = 0; k < size_prob; k++) {
38                              bestBee_inseq[h][k] = temp[0][k];
39                          }
40                          fit_best[h] = fit_hasil;
41                      } else {
42                          trial_empbee[h]++;
43                      }
44                      if (margin == nse) {
45                          break;
46                      }
47                      margin++;
48                  }
49                  if (z == nse) {
50                      break;
50                  }
52              }
53          }
```

```
54          System.out.println("Route Best : ");
55          for (int k = 0; k < popSize; k++) {
56              for (int l = 0; l < size_prob; l++) {
57                  System.out.print(bestBee_inseq[k][l] + "\t");
58              }
59              System.out.print("Fitness: " + fit_best[k] + "\tTrial
    : " + trial_empbee[k]);
60              System.out.println("");
61          }
62          return bestBee_inseq;
63      }
```

Source Code 5.8 *Listing Code Insert Sequences*

Penjelasan *source code 5.8* adalah:

1. Baris 16 menjelaskan nilai margin yang diset 0 untuk menjadi pembatas agar perulangan tidak keluar dari index
2. Baris 17-19 menjelaskan nilai array temp yang didapatkan dari nilai array hasil
3. Baris 22-30 menjelaskan proses *insertion* pada rute
4. Baris 34 menjelaskan proses perhitungan nilai *fitness* hasil *insertion*
5. Baris 35-43 menjelaskan proses penentuan nilai *trial*. Jika nilai *fitness* hasil *insert* lebih besar dari nilai *fitness* sebelumnya maka nilai *trial* di set menjadi 0, tetapi jika lebih kecil maka nilai *trial* akan ditambah 1
6. Baris 44-48 menjelaskan pengecakan nilai margin apakah sudah sama dengan nse. Apabila iya maka perulangan akan dihentikan, jika tidak nilai margin akan ditambah 1
7. Baris 49-51 menjelaskan pengecakan nilai margin apakah sudah sama dengan nse. Apabila iya maka perulangan akan dihentikan.

5.2.5 Proses Fase *Scout Bee*

Pada fase ini akan didapatkan solusi terbaik dan rute baru untuk iterasi selanjutnya.

```
1   public static int[][] scoutBee(int[][] bee_awal, int[][]
    bee_inseq, double[] fit_bee_inseq) {
2           int index_best = 0;
3           double best_fit = 0;
4           int best_trial = trial_empbee[0];
5           System.out.println("");
6           System.out.println("===== SCOUT BEE =====");
7           System.out.println("Individu Awal");
8           fitness = hitungFitness(bee_awal);
9           for (int i = 0; i < popSize; i++) {
10              for (int j = 0; j < size_prob; j++) {
11                  System.out.print(bee_awal[i][j] + "\t");
12              }
13              System.out.println(fitness[i]);
14          }
15          System.out.println("");
16          System.out.println("Individu Hasil Insert Sequences");
```

```
17          for (int k = 0; k < popSize; k++) {
18              if (fit_bee_inseq[k] > best_fit) {
19                  best_fit = fit_bee_inseq[k];
20                  best_trial = trial_empbee[k];
21                  index_best = k;
22              }   else   if   (fit_bee_inseq[k]   ==   best_fit   &&
    trial_empbee[k] < best_trial) {
23                  best_trial = trial_empbee[k];
24                  index_best = k;
25              }
26              for (int l = 0; l < size_prob; l++) {
27                  System.out.print(bee_inseq[k][l] + "\t");
28              }
29              System.out.println(fit_bee_inseq[k]);
30          }
31          System.out.println("");
32          System.out.println("Solusi terbaik");
33          for (int m = 0; m < size_prob; m++) {
34              System.out.print(bee_inseq[index_best][m] + "\t");
35          }
36          System.out.println(fit_bee_inseq[index_best]);
37          System.out.println("");
38          System.out.println("Rute Baru Iterasi Selanjutnya");
39          for (int pop = 0; pop < popSize; pop++) {
40              if (trial_empbee[pop] > limit && fit_bee_inseq[pop] >
    fitness[pop]) {
41                  trial_empbee[pop] = 0;
42                  fitnessRute[pop] = fit_bee_inseq[pop];
43                  for (int rut = 0; rut < size_prob; rut++) {
44                      array_route[pop][rut] = bee_inseq[pop][rut];
45                  }
46              }   else   if   (trial_empbee[pop]   >   limit   &&
    fit_bee_inseq[pop] <= fitness[pop]) {
47                  trial_empbee[pop] = 0;
48                  fitnessRute[pop] = fit_bee_inseq[pop];
49                  for (int rut = 0; rut < size_prob; rut++) {
50                      array_route[pop][rut]  =  array_route[pop][rut]
    = rand.nextInt((sheet.getPhysicalNumberOfRows() - 1) - min) + min;
51                      if (array_route[pop][rut] == 0) {
52                          rut--;
53                      }  else  if  (slot[array_route[pop][rut]]  ==
    true) {
54                          rut--;
55                      } else {
56                          slot[array_route[pop][rut]] = true;
57                      }
58                  }
59              }
60              for (int rut = 0; rut < size_prob; rut++) {
61                  System.out.print(array_route[pop][rut] + "\t");
62              }
63              System.out.println("Trial: " + trial_empbee[pop]);
64          }
65          System.out.println("");
66          return array_route;
67      }
```

Source Code 5.9 *Listing Code Scout Bee*

Penjelasan *source code* 5.9 adalah:

1. Baris 18-25 menjelaskan proses mendapatkan solusi terbaik dengan membandingkan nilai *fitness* dari individu awal dengan nilai *fitness* individu hasil *insert sequences*
2. Baris 39-64 menjelaskan proses mendapatkan rute baru untuk iterasi selanjutnya. Jika nilai *trial* rute > *Limit*, maka hapus individu yang tidak ada perbaikan, ganti dengan individu baru hasil *random*, kemudian reset *trial* menjadi 0, jika *trial* rute > Limit dan terdapat individu yang ada perbaikan, maka tidak perlu digantikan dengan individu baru, kemudian reset *trial* menjadi 0. Jika *trial* rute < Limit, maka tidak perlu digantikan dengan individu baru, dan nilai *trial* tidak perlu di-*reset*.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab pengujian dan analisis akan membahas tentang pengujian terhadap hasil implementasi dan dilakukan analisis. pengujian yang dilakukan meliputi pengaruh banyaknya jumlah iterasi untuk mendapatkan solusi terbaik, pengaruh jumlah *size problem* terhadap hasil solusi terbaik, pengaruh jumlah *limit* terhadap hasil solusi terbaik, dan pengaruh jumlah *pop size* terhadap hasil solusi terbaik.

6.1 Pengujian

Pengujian dilakukan dengan menggunakan data sebanyak 163 titik pengiriman. Pengujian dilakukan dengan 4 model pengujian, yaitu:

1. Pengujian untuk mengetahui pengaruh jumlah *size problem*
2. Pengujian untuk mengetahui pengaruh jumlah *pop size*
3. Pengujian untuk mengetahui pengaruh jumlah *limit*
4. Pengujian untuk mengetahui konvergensi
5. Pengujian global hasil pengujian 4 parameter.

6.2 Hasil Pengujian

Dari 4 model pengujian yang dilakukan, didapatkan 4 hasil pengujian yaitu hasil pengujian untuk mengetahui pengaruh jumlah *pop size,* hasil pengujian untuk mengetahui pengaruh jumlah *size problem,* hasil pengujian untuk mengetahui pengaruh jumlah *limit,* dan hasil pengujian untuk mengetahui konvergensi.

6.2.1 Pengujian Pengaruh Banyaknya Jumlah *Size Problem*

Pengujian pengaruh banyaknya jumlah *size problem* dilakukan untuk melihat tingkat konvergensi algoritme. Jumlah *pop size* yang dimasukkan adalah 8 dengan jumlah *limit* sebanyak 10 dan banyak iterasi adalah 50. Pengujian dilakukan dengan dilakukan percobaan sebanyak lima kali. Hasil pengujian ditunjukkan pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Pengaruh Banyaknya Jumlah *Size Problem*

Jumlah *Size Problem*	Hasil Percobaan Ke-*i*					Rata-rata *fitness*
	1	2	3	4	5	
23	0,044131	0,061538	0,076923	0,067568	0,060241	0,06208
24	0,03723	0,061425	0,081301	0,053763	0,059172	0,058578
25	0,033887	0,060827	0,076336	0,063291	0,05848	0,058564
26	0,03592	0,057143	0,078125	0,056818	0,062112	0,058023

Tabel 6.1 Hasil Pengujian Pengaruh Banyaknya Jumlah *Size Problem* (lanjutan)

Jumlah *Size Problem*	Hasil Percobaan Ke-*i*					Rata-rata *fitness*
	1	2	3	4	5	
27	0,03003	0,060459	0,072993	0,06135	0,0625	0,057466
28	0,0373	0,050659	0,074627	0,05291	0,059172	0,054933
29	0,043178	0,057339	0,074627	0,054348	0,060606	0,05802
30	0,034734	0,044643	0,067114	0,055556	0,056818	0,051773
31	0,027315	0,048591	0,072993	0,046296	0,053191	0,049677
32	0,03003	0,054407	0,046468	0,050251	0,053763	0,046984

Pada Tabel 6.1 terlihat bahwa hasil rata-rata nilai *fitness* terbaik adalah pada jumlah *size problem* sebanyak 23 dengan rata-rata nilai *fitness*-nya sebesar 0,06208.

6.2.2 Pengujian Pengaruh Banyaknya Jumlah *Pop Size*

Pengujian pengaruh banyaknya jumlah *pop size* dilakukan untuk melihat tingkat konvergensi algoritme. Jumlah *size problem* yang dimasukkan adalah 23 dengan jumlah *limit* sebanyak 10 dan banyak iterasi adalah 50. Pengujian dilakukan dengan dilakukan percobaan sebanyak 5 kali. Hasil pengujian ditunjukkan pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Pengaruh Banyaknya Jumlah *Pop Size*

Jumlah *Pop Size*	Hasil Percobaan Ke-*i*					Rata-rata *fitness*
	1	2	3	4	5	
10	0,04583	0,062035	0,083333	0,05988	0,067114	0,063638
20	0,0456	0,059032	0,081967	0,064516	0,070423	0,064307
30	0,046555	0,067204	0,084746	0,070922	0,073529	0,068591
40	0,053648	0,065963	0,084746	0,071429	0,079365	0,07103
50	0,054705	0,070423	0,086957	0,066225	0,072464	0,070155
60	0,054496	0,06812	0,08547	0,074074	0,079365	0,072305
70	0,056465	0,066489	0,084034	0,075188	0,08	0,072435
80	0,060386	0,066845	0,086207	0,078125	0,081967	0,074706
90	0,054348	0,069444	0,084746	0,07874	0,075188	0,072493
100	0,052056	0,066489	0,084746	0,072993	0,076336	0,070524

Pada Tabel 6.2 terlihat bahwa hasil rata-rata nilai *fitness* terbaik adalah pada jumlah *pop size* sebanyak 80 dengan rata-rata nilai *fitness*-nya sebesar 0,074706.

6.2.3 Pengujian Pengaruh Banyaknya Jumlah *Limit*

Pengujian pengaruh banyaknya jumlah *limit* dilakukan untuk melihat tingkat konvergensi algoritme. Jumlah *size problem* yang dimasukkan adalah 23 dengan jumlah *pop size* sebanyak 80 dan banyak iterasi adalah 50 kali. Pengujian dilakukan dengan dilakukan percobaan sebanyak 5 kali. Hasil pengujian ditunjukkan pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Pengaruh Banyaknya Jumlah *Limit*

Jumlah *Limit*	Hasil Percobaan Ke-*i*					Rata-rata *fitness*
	1	2	3	4	5	
10	0,060386	0,066845	0,086207	0,078125	0,081967	0,074706
20	0,055066	0,067114	0,087719	0,074074	0,079365	0,072668
30	0,05534	0,068776	0,087719	0,070922	0,078125	0,072176
40	0,056338	0,0612	0,08547	0,071429	0,075188	0,069925
50	0,0489	0,069444	0,082645	0,071429	0,076336	0,069751
60	0,051282	0,062189	0,082645	0,075758	0,0481	0,063995
70	0,064851	0,064851	0,083333	0,068966	0,077519	0,071904
80	0,05814	0,065963	0,081967	0,064935	0,075758	0,069352
90	0,045746	0,064185	0,08547	0,070922	0,076336	0,068532
100	0,043917	0,057013	0,080645	0,053763	0,068493	0,060766

Pada Tabel 6.3 terlihat bahwa hasil rata-rata nilai *fitness* terbaik adalah pada jumlah *limit* sebanyak 10 dengan rata-rata nilai *fitness*-nya sebesar 0,074706.

6.2.4 Pengujian Konvergensi

Pengujian konvergensi dilakukan untuk melihat tingkat konvergensi algoritme. Jumlah *size problem* yang dimasukkan adalah 23 dengan jumlah *limit* sebanyak 10 dan jumlah *pop size* sebanyak 80. Pengujian dilakukan dengan dilakukan percobaan sebanyak 5 kali. Hasil pengujian ditunjukkan pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian Pengaruh Banyaknya Jumlah Iterasi

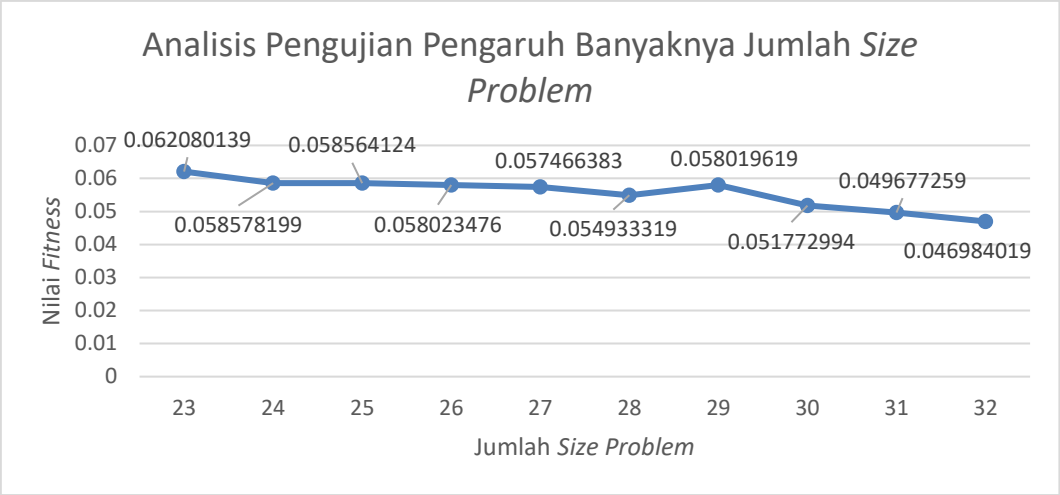
Iterasi	Hasil Percobaan Ke-*i*					Rata-rata *fitness*
	1	2	3	4	5	
1	0,03639	0,05176	0,079745	0,054054	0,065789	0,057548
10	0,045746	0,065963	0,085763	0,068493	0,070922	0,067377
50	0,056306	0,072993	0,088968	0,073529	0,074627	0,073285
100	0,056689	0,074074	0,093458	0,078125	0,076923	0,075854
500	0,056689	0,074074	0,097276	0,07874	0,080645	0,077485
600	0,056689	0,074074	0,097276	0,07874	0,084034	0,078163
700	0,056689	0,074074	0,097276	0,07874	0,084034	0,078163
800	0,056689	0,074074	0,097276	0,07874	0,084034	0,078163
900	0,056689	0,074074	0,097276	0,07874	0,084034	0,078163
1000	0,056689	0,074074	0,097276	0,07874	0,084034	0,078163

6.3 Analisis Hasil Pengujian

Analisis hasil pengujian merupakan analisis yang dilakukan terhadap hasil pengujian yang telah dilakukan. Analisis dilakukan kepada semua hasil pengujian yaitu analisis pengujian pengaruh banyaknya jumlah *pop size*, analisis pengujian pengaruh banyaknya jumlah *size problem,* analisis pengujian pengaruh banyaknya jumlah *limit,* dan analisis pengujian pengaruh banyaknya jumlah iterasi terhadap konvergensi.

6.3.1 Analisis Pengujian Pengaruh Banyaknya Jumlah *Size Problem*

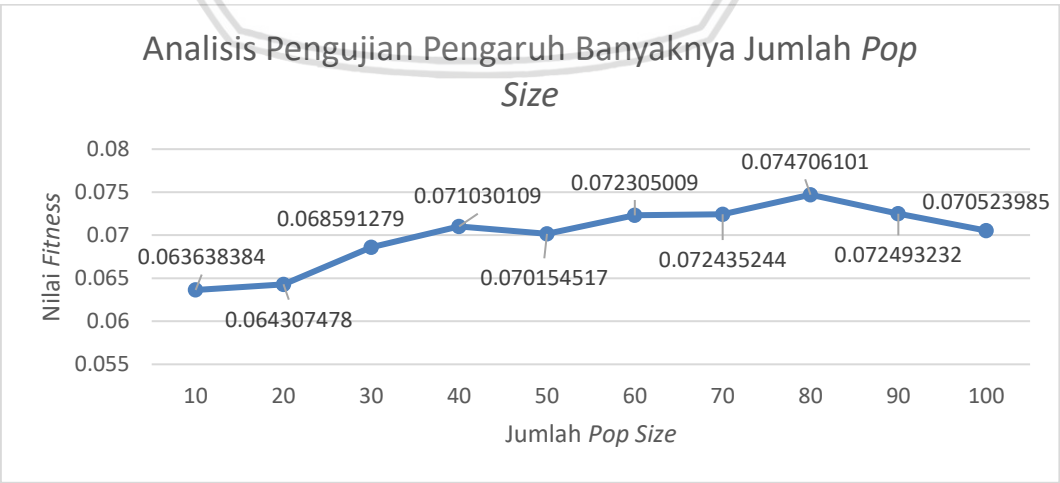
Hasil pengujian yang telah dilakukan terhadap pengaruh banyaknya jumlah *size problem* ditunjukkan pada Gambar 6.1 terlihat bahwa hasil rata-rata nilai *fitness* tertinggi didapatkan dari jumlah *size problem* sebanyak 23 yaitu sebesar 0,06208. Pada Gambar 6.1 juga dapat dilihat bahwa semakin sedikit jumlah *size problem*-nya maka rata-rata nilai *fitness* akan semakin besar, sehingga kemungkinan nilai *fitness* dalam mendapatkan nilai yang konvergen juga semakin tinggi. Hal ini disebabkan karena jumlah titik pengiriman yang sedikit mengakibatkan nilai pembagi yang berupa jarak dalam perhitungan nilai *fitness* akan semakin kecil.



Gambar 6.1 Analisis Pengujian Pengaruh Banyaknya Jumlah *Size Problem*

6.3.2 Analisis Pengujian Pengaruh Banyaknya Jumlah *Pop Size*

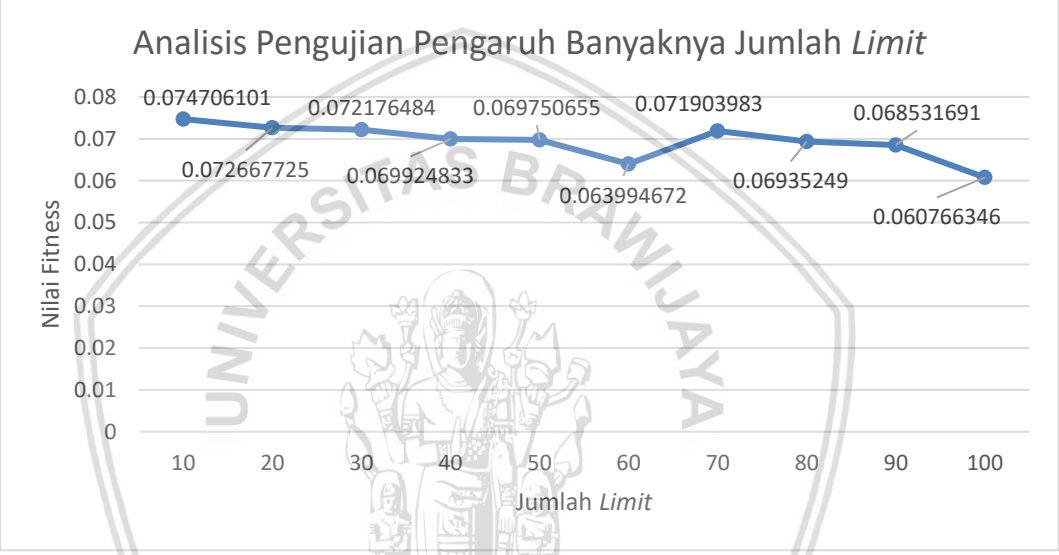
Hasil pengujian pengaruh banyaknya jumlah *pop size* yang ditunjukkan pada Gambar 6.2 terlihat bahwa jumlah *pop size* sebanyak 80 *pop size* didapatkan rata-rata nlai *fitness* tertinggi yaitu sebesar 0,074706101. Jumlah dari *pop size* berpengaruh terhadap hasil solusi terbaik yang dihasilkan karena berdasarkan jumlah *pop size* dengan jumlah yang sedikit cenderung menghasilkan rata-rata nilai *fitness* yang kurang optimal. Gambar 6.2 menunjukkan jumlah *pop size* sebanyak 10 dan 20 menghasilkan rata-rata nilai *fitness* yang kecil. Pada jumlah *pop size* sebanyak 30 mulai menunjukkan peningkatan rata-rata nilai *fitness* dan tidak mengalami penurunan rata-rata nilai *fitness* yang signifikan. Hal ini karena dipengaruhi oleh pengacakan rute pada masing-masing individu (populasi), sehingga semakin banyak jumlah *pop size* maka rute yang akan dibentuk juga semakin banyak dan kemungkinan didapatkan solusi terbaik juga semakin tinggi. Keragaman rute yang dibentuk juga semakin banyak, sehingga pilihan eksplorasi rute juga banyak.



Gambar 6.2 Analisis Pengujian Pengaruh Banyaknya Jumlah *Pop Size*

6.3.3 Analisis Pengujian Pengaruh Banyaknya Jumlah *Limit*

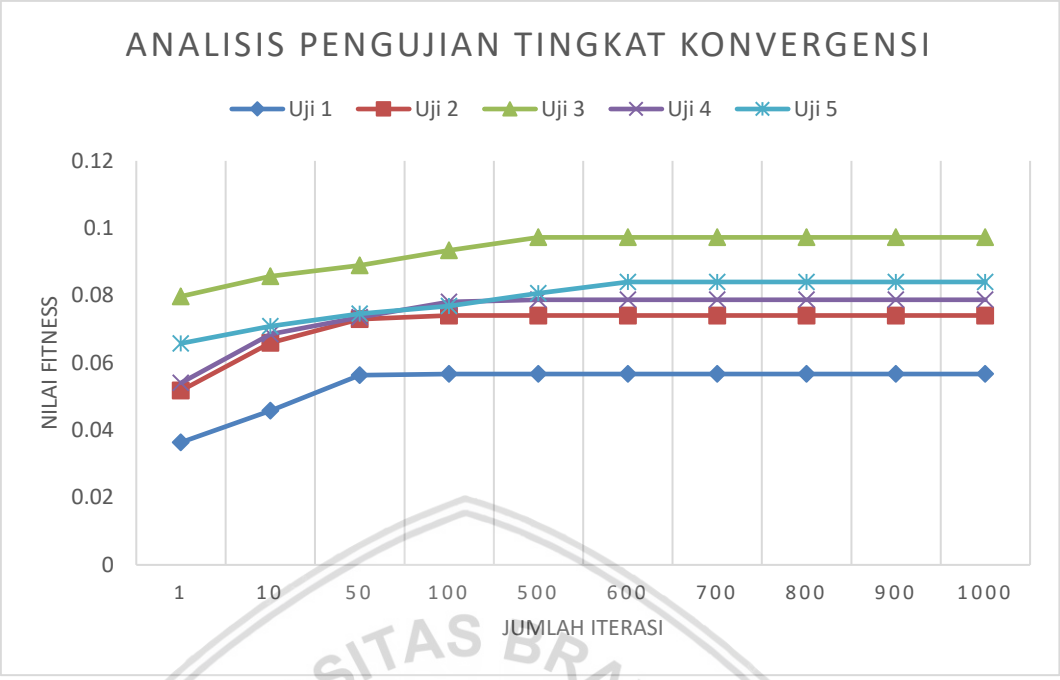
Berdasarkan hasil pengujian pengaruh banyaknya jumlah *limit* pada gambar 6.3 dengan menggunakan panjang *size problem* 23 dan jumlah *pop size* sebanyak 80, terlihat bahwa jumlah *limit* sebanyak 10 menghasilkan rata-rata nilai *fitness* tertinggi yaitu 0,074706. Parameter *limit* menunjukkan batasan *trial* untuk solusi tertentu yang tidak mengalami peningkatan. Sehingga jumlah *limit* yang semakin kecil diharapkan dapat didapatkan rute baru yang lebih optimal. Tetapi jika parameter *size problem* dan *pop size* diatur dengan nilai berbeda akan didapatkan nilai *limit* yang berbeda. Hal ini disebabkan karena masalahnya yang cukup stokastik yang menyebabkan perbedaan besar pada nilai *limit* yang sama (Li, et al., 2012).



Gambar 6.3 Analisis Pengujian Pengaruh Banyaknya Jumlah *Limit*

6.3.4 Analisis Pengujian Konvergensi

Hasil analisis pengujian pengaruh banyaknya jumlah iterasi terhadap tingkat konvergensi ditunjukkan pada Gambar 6.4. Pengujian dilakukan dengan melakukan iterasi sebanyak 1 sampai 1000 iterasi. Berdasarkan pengujian tersebut, terlihat bahwa tingkat konvergensi sudah dimulai pada iterasi sebanyak 600 iterasi. Pada Tabel 6.4 terlihat bahwa jumlah iterasi 600 memiliki rata-rata nilai *fitness* optimal yaitu 0,078163. Tingkat konvergensi sangat dipengaruhi oleh jumlah iterasi. Semakin banyak iterasi semakin mudah menentukan tingkat konvergensi dan rata-rata nilai *fitness* yang didapatkan juga semakin tinggi. Tingkat konvergensi yang baik adalah apabila hanya dibutuhkan sedikit iterasi untuk mencapai konvergensi (Rosita, et al., 2012).



Gambar 6.4 Analisis Pengujian Tingkat Konvergensi

6.3.5 Analisis Global Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan didapatkan parameter-parameter yang optimal dengan rata-rata nilai *fitness* terbaik. Pengujian pertama dilakukan pengujian terhadap jumlah *size problem* yang digunakan dengan hasil yang menunjukkan bahwa jumlah *size problem* memengaruhi rata-rata nilai *fitness* dan didapatkan rata-rata nilai *fitness* terbaik yaitu 0,06208 pada jumlah *size problem* sebanyak 23. Pengujian kedua dilakukan terhadap jumlah *pop size* dengan hasil optimal pada jumlah *pop size* 80 dengan rata-rata nilai *fitness* 0,074706. Pengujian ketiga dilakukan terhadap pengaruh jumlah *limit* dengan *limit* yang menghasilkan rata-rata nilai *fitness* optimal sebesar 0,074706 pada *limit* 10. Dan pengujian yang terakhir adalah pengujian konvergensi dengan melihat dari banyaknya iterasi. Banyak iterasi yang diperlukan untuk mencapai nilai konvergen adalah pada iterasi 600 dengan rata-rata nilai *fitness* 0,078163.

Dari hasil pengujian di atas, dilakukan analisis dari hasil perbandingan antara optimasi sistem dengan pemilihan jalur yang dipilih oleh *sales* dalam 1 hari pengiriman. Analisis dilakukan berdasarkan hasil pengujian sebanyak 5 percobaan. Jalur yang dilalui oleh *sales* adalah sebagai berikut:

Rute 1 : Jus Sutami - Ro (Tidar) - Cing Jus – Fomori - Cumi Hitam (Galunggung) - Padang Murah - Jupe (Trs. SBY) - Jupe (Ambarawa) – Melacca - Wr Kecik - Jo Juice - Cha-Cha Juice - Bakso Ikip - Oshin Jus - Wr Shinchan - Deprot Gang Djangkrik - 9 Dedik (Sutami) - Kaw Kaw - Nyoklat (Wilis) - Jus Ayu - Mi Sabar - Shao Kao - CK (Candi)

Rute 2 : Bebek Doeloer - Wr Bu Indri - Ayam Geprak - Kedai 0341 - Raja Juice – Balipuccino - The Library - Jupe (Bondowoso) - Wr Bu Yayuk - Teh Poci

Smp Lab - CK (Telkom) - Mocca Float - Hotel Salimar - Wr SD Lab - Bunga Bali - Ok Jus – Jupe Telkom - Wr SMP Lab - Wr Bu Didik – Cincau - Madam Tea - Wr Bu Erna - Café Warna

Rute 3 : Ice Coklat - Meni-Meni I Mog - Fourty Eight - Mommy Mango – Seventea - Dum Dum Mog - Kako Susu - CK Mog – Dandees - Ayam Tenes - Rujak Semeru - Oh My Plate (Kawi) – Sumbing - Ichiban Sushi - Wr Taman Slamet - Teh Poci SMA Lab – Moshi-Moshi Ramen Barat - En. Hachi Hachi Mog – Moshi-Moshi Ramen Timur – Bubble – Keylabs - Istana Mie Mog - Surya Kuring

Rute 4 : Toast Story – Gekvec - Bunch Bead - Es Cemot - Nelongso (Trs. SBY) - En. Soju Bar (Jakarta) - Twin Dessert - Bingsoo Sutami - Wr Sedap - Alice Tea Room - Nakam Dulu - Joglo Dau Ijen – Kakao - Pin Balt - Capcin Bogor - Bakso Bakar Trip - Travel Mie - Ayam Dempo - Java Dancer (Jakarta) - JJJ Sutami - Kopi Sino - Lup Lup Tidar - Roti Bakar

Rute 5 : Nasgor 69 Mog - My Kopi O - Mie Kudusan Mog - C2C - D Cost – Yono - En Taiwan Mog - Teh Racek SPBU - Ed Miami Wilis - Ling Ling Mog – Rotbuck – Kafein - GFC Pulosari – Holland - Duta Catering - Paper Town - Kopi Retjeh – Sonokembang – Crochet – Inul Vizta - Ranch Market - Ed Mie Kudusan Bondowoso - Niki Kopitiam

Hasil pengujian ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian Global

	1	2	3	4	5	Total Jarak	Rata-rata *fitness*
Jarak Oleh Sales (*fitness*)	40,12 (0,024319)	23,22 (0,041288)	14 (0,06667)	27,3 (0,035336)	19,1 (0,049751)	123,74	0,043472
Jarak Oleh Sistem (*fitness*)	16,64 (0,056689)	12,5 (0,074074)	9,28 (0,097276)	11,7 (0,07874)	10,9 (0,084034)	61,02	0,078163
Selisih	23,48	10,72	4,72	15,6	8,2	62,72	0,034691

Dari hasil pengujian pada Tabel 6.5 dapat dibuktikan bahwa hasil perhitungan sistem jauh lebih optimal dibandingkan dengan hasil perhitungan jarak yang dilalui oleh *sales* di jalan. Hasil perbandingan yang memiliki selisih terbesar ada pada rute 1 dengan selisih 23,48. Dari semua perhitungan jarak yang dihasilkan oleh sistem selalu lebih kecil dari hasil perhitungan jarak yang dilalui oleh *sales* di jalan. Dengan demikian sistem ini dapat dijadikan sebuah solusi oleh *sales* dalam menentukan jalur yang akan dilewati, sehingga waktu yang dibutuhkan akan lebih sedikit dan meminimalkan biaya transportasi.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian implementasi optimasi rute distribusi *Multiple Traveling Salesman Problem* pada distribusi es batu dengan algoritme *Artificial Bee Colony* (ABC), didapatkan kesimpulan sebagai berikut:

1. Berdasarkan hasil perbandingan sistem algoritme *artificial bee colony (*ABC*)* dengan kondisi aktual yang dipilih oleh *sales* di lapangan adalah hasil oleh sistem mendapatkan total jarak yang lebih pendek dibandingkan dengan total jarak yang dipilih oleh *sales*, sehingga hasil yang dihasilkan sistem lebih optimal.
2. Rata-rata nilai *fitness* tertinggi diperoleh pada pengujian dengan jumlah *pop size* sebanyak 80, jumlah *size problem* sebanyak 23, *limit* 10, dan iterasi yang mencapai konvergensi saat iterasi sebanyak 600 kali dengan rata-rata nilai *fitness 0,0*78163. Sehingga dapat disimpulkan bahwa parameter optimal untuk kasus ini adalah dengan *pop size* sebanyak 80, *size problem* sebanyak 23, *limit* pada *trial* ke-10, dan iterasi sebanyak 600.

7.2 Saran

Adapun saran yang diberikan oleh penulis untuk penelitian lebih lanjut adalah:

1. Pada penelitian selanjutnya terkait perkembangan data agar dapat diimplementasikan dengan menggunakan *database* untuk data yang lebih besar karena pada kenyataannya perusahaan memiliki data yang lebih besar tidak hanya kota Malang saja. Data yang digunakan pada penelitian ini hanya menggunakan data pada cakupan wilayah kecil dari kota Malang.
2. Pada penelitian selanjutnya agar melakukan penelitian dengan dilakukan pengujian terhadap jumlah *sales* lebih dari 2 karena jumlah *sales* pada penelitian ini hanya berjumlah 2 *sales.*
3. Pada penelitian selanjutnya sebaiknya diperhatikan kondisi jalannya sebagai parameter.

DAFTAR PUSTAKA

Amri, F., Nababan, E. B. & Syahputra, M. F., 2012. Artificial Bee Colony Algorithm untuk Menyelesaikan Travelling Salesman Problem. *Jurnal Dunia Teknologi Informasi,* 1(1), pp. 8-13.

Cholissodin, I. & Riyandani, E., 2016. *Swarm Intelligence (Teori & Case Study).* Malang: Fakultas Ilmu Komputer.

Herawati, Y. & Mahmudy, W. F., 2017. Optimasi Menu Makanan Bagi Pasien Gagal Ginjal Menggunakan Algoritme Lebah. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(4), pp. 1698-1703.

Hermawan, M. A., Hidayat, N. & Setiawan, B. D., 2017. Sistem Optimasi Rute Tempat Wisata Kuliner Di Malang Menggunakan Algoritma Bee Colony. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1(3), pp. 215-223.

Irawan, H., 2015. Perencanaan Mesin Penyerut Es Sederhana Berkapasitas 126 Kg/Jam. *Skripsi.*

Karaboga, D. & Basturk, B., 2007. A Powerful and Efficient Algorithm for Numerical Function Optimiziation : Artificial Bee Colony Algorithm. *Journal of Global Optimization,* 39(3), pp. 459-471.

Karimah, S., Widodo, A. W. & Cholissodin, I., 2017. Optimasi Multiple Traveling Salesman Probelm Pada Pendistribusian Air Minum menggunakan Algoritma Genetika (Studi Kasus: UD. Tosa Malang). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1(9), pp. 849-858.

Li, L., Cheng, Y., Tan, L. & Niu, B., 2012. A Discrete Arificial Bee Colony Algorithm for TSP Problem. *Springer-Verlag Berlin Heidelberg,* pp. 566-573.

Otri, S., 2011. Improving The Bees Algorithm For Complex Optimisation Problems.

Rosita, A., Purwanto, Y. & Soelaiman, R., 2012. Implementasi Algoritma Particle Swarm untuk Menyelesaikan Sistem Persamaan Nonlinear. *Jurnal teknik ITS,* 1(1), pp. 211-215.

Segetlija, Z., Mesarić, J. & Dujak, D., 2010. Importance of Distribution Channels - Marketing Channels - For National Economy. pp. 785-809.

Sembiring, A. C., 2014. Penentuan Rute Distribusi Produk Yang Optimal Dengan Menggunakan Algoritma Heuristik Pada PT. Coca-cola Bottling Indonesia Medan. *Jurnal Sains dan Teknologi,* 1(2), pp. 76-84.

Singh, S. & Lodhi, E. A., 2014. Comparison Study of Multiple Traveling Salesman Problem using Genetic Algorithm. *IJCSNS International Journal of Computer Science and Network Security,* 14(7), pp. 107-110.

Wiyanti, D. T., 2013. Algoritma Optimasi Untuk Penyelesaian Traveling Salesman Problem. *Jurnal Transformatika,* 11(1), pp. 1-6.

Wong, L.-P., Low, M. Y. H. & Chong, C. S., 2008. *A Bee Colony Optimization Algorithm for Traveling Salesman Problem.* Kuala Lumpur, 2008 Second Asia International Conference on Modelling & Simulation (AMS), pp. 818-823.

Xue, M. H., Wang, T. Z. & Mao, S., 2016. *Double Evolutional Artificial Bee Colony Algorithm for Multiple Traveling Salesman Problem.* Hong Kong, EDP Sciences.