SISTEM OPTIMASI RUTE TEMPAT WISATA KULINER DI MALANG MENGUNAKAN ALGORITMA BEE COLONY

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Arif Hermawan
NIM: 115060807111050

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

SISTEM OPTIMASI RUTE TEMPAT WISATA KULINER DI MALANG MENGGUNAKAN ALGORITMA BEE COLONY

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Arif Hermawan
NIM: 115060807111050

Skripsi ini telah diuji dan dinyatakan lulus pada
13 April 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Nurul Hidayat, S.Pd, M.Sc	Budi Darma Setiawan, S.Kom, M.Cs
NIP: 19680430 200212 1 001	NIP: 198410152014041002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan , S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

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, 13 April 2017

Muhammad Arif Hermawan

NIM: 115060807111050

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada seluruh umat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "SISTEM OPTIMASI RUTE TEMPAT WISATA KULINER DI MALANG MENGGUNAKAN ALGORITMA *BEE COLONY*" dengan baik. Sholawat serta salam kami haturkan kepada junjungan kami Nabi Muhammad SAW, semoga kami selalu istiqomah dalam meneladaninya. Skripsi ini merupakan serangkaian mata kuliah dan syarat kelulusan yang harus ditempuh oleh mahasiswa sebagai pengaplikasian dari beberapa mata kuliah yang telah di pelajari di bangku kuliah.

Dalam pelaksanaan dan penulisan skripsi ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moral maupun material. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Nurul Hidayat, S.Pd, M.Sc selaku dosen pembimbing I skripsi yang telah dengan sabar membimbing dan mengarahkan penulis, sehingga dapat menyelesaikan skripsi ini.
2. Budi Darma Setiawan, S.Kom, M.Cs selaku dosen pembimbing II skripsi yang telah dengan sabar membimbing dan mengarahkan penulis, sehingga dapat menyelesaikan skripsi ini.
3. Wayan Firdaus Mahmudy , S.Si, M.T, Ph.D., Ir. Heru Nurwasito, M.Kom., Marji , Drs., M.T, dan Eddy Santoso, S.Kom. selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, dan Wakil Ketua 3 Fakultas Ilmu Komputer Universitas Brawijaya.
4. Tri Astoto Kurniawan , S.T, M.T, Ph.D dan M. Tanzil Furqon, S.Kom, M.CompSc selaku Ketua dan Sekretaris Jurusan Teknik Informatika Universitas Brawijaya.
5. Denny Sagita R., S.Kom, M.Kom selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Budi Darma Setiawan, S.Kom, M.Cs selaku Ketua Laboratorium Komputasi dan Sistem Cerdas.
7. Orang tua dan seluruh keluarga atas segenap dukungan dan kasih sayang yang telah diberikan.

8. Seluruh Dosen Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmu dan bantuan yang telah diberikan.
9. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
10. Teman-teman angkatan 2011 dan Konsentrasi Cerdas dan Visualisasi dan seluruh pihak yang telah membantu kelancaran penulisan skripsi yang tidak dapat penulis sebutkan satu persatu.
11. Seluruh pihak yang telah membantu dalam penyelesaian penulisan skripsi ini yang tidak dapat penulis sebut satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik yang membangun dapat disampaikan melalui email penulis ershav@gmail.com. Penulis berharap skripsi ini dapat memberikan manfaat bagi semua pihak.

Malang, 13 April 2017

Penulis

ershav@gmail.com

ABSTRAK

Banyaknya tempat wisata kuliner di Malang yang dapat dijangkau membuat kesulitan para pecinta kuliner untuk mencari rute optimum, baik dari segi jarak, waktu, maupun biaya yang dikeluarkan untuk berpergian dari satu tempat kuliner ke tempat kuliner yang lain. Salah satu faktor yang mempengaruhi besarnya pengeluaran saat wisata kuliner adalah biaya transportasi. Hal yang berhubungan erat dengan transportasi adalah jarak tempuh yang dilalui. Banyak penikmat kuliner yang merasa terlalu banyak membuang waktu dalam perjalanan menuju tempat kuliner dikarenakan salah dalam pemilihan rute yang ditempuh. Karena Malang memiliki tempat wisata kuliner yang banyak, maka dibutuhkan optimasi dalam pencarian rute optimum dari posisi awal menuju posisi tujuan. Dipilihnya algoritma *bee colony* dikarenakan algoritma ini dirasa memiliki kemampuan untuk keluar dari *local minimum* dan dapat secara efisien digunakan untuk optimasi. Algoritma *bee colony* dirasa mampu menyelesaikan permasalahan *Traveling Salesman Problem* lebih baik dibandingkan dengan algoritma lain yang juga didasarkan pada kecerdasan berkelompok. Pada pengujian, didapatkan hasil bahwa penggunaan algoritma *bee colony* telah mengalami konvergensi dalam pencarian solusi terbaiknya yang dapat dilihat dari *fitness* yang dihasilkan. Salah satu yang terbaik telah mengalami konvergensi pada jumlah *bee colony* sebanyak 20 dari 50 jumlah *bee colony* . Selain itu konvergensi juga dapat dilihat pada jumlah iterasi 20 dari jumlah maksimum iterasi 50.

Kata Kunci: Algoritma, Algoritma Bee Colony, Optimasi, Rute Terpendek, Traveling Salesman Problem

DAFTAR ISI

KATA PENGANTAR..... iv
DAFTAR ISI..... vii
DAFTAR TABEL..... x
DAFTAR GAMBAR..... xi
BAB 1 PENDAHULUAN..... 1
1.1 Latar belakang..... 1
1.2 Rumusan masalah..... 3
1.3 Tujuan..... 3
1.4 Manfaat..... 3
1.5 Batasan masalah..... 3
1.6 Sistematika pembahasan..... 4
BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI..... 5
2.1 Kajian Pustaka..... 5
2.2 Wisata Kuliner..... 6
2.3 Wisata Kuliner Malang..... 7
2.4 Optimasi..... 7
2.5 Travelling Salesman Problem (TSP)..... 8
2.6 Algoritma Bee Colony..... 9
BAB 3 METODOLOGI PENELITIAN DAN PERANCANGAN..... 16
3.1 Metodologi..... 16
3.1.1 Studi Literatur..... 17
3.1.2 Pengumpulan Data..... 17
3.1.3 Analisis dan Perancangan Sistem..... 18
3.1.4 Implementasi Sistem..... 18
3.1.5 Pengujian dan Analisis..... 18
3.1.6 Pengambilan Kesimpulan..... 19
3.2 Perancangan Sistem..... 19
3.2.1 Deskripsi Sistem..... 19
3.2.2 Perancangan Perangkat Lunak..... 20
3.2.2.1 Fase Initial..... 21

3.2.2.2 Hitung Nilai Fitness 22
3.2.2.3 Fase Employed Bee 23
3.2.2.4 Fungsi Swap Operator atau Random Swap 24
3.2.2.5 Fungsi Swap Sequence atau Random Swap Sequence.....26
3.2.2.6 Fase Onlooker Bee 27
3.2.2.7 Fase Scout Bee 28
3.2.3 Perhitungan Manual 28
BAB 4 IMPLEMENTASI 40
4.1 Lingkungan Implementasi 40
4.1.1 Lingkungan Perangkat Keras 40
4.1.2 Lingkungan Perangkat Lunak 40
4.2 Implementasi Program 40
4.2.1 Proses Fase Initial 40
4.2.1.1 Proses Generated Individu 41
4.2.1.2 Proses Check Duplicate 41
4.2.1.3 Proses Random Colony 42
4.2.1.4 Proses Shuffle Array 43
4.2.1.5 Proses Menghitung Fitness 43
4.2.2 Proses Fase Employeed Bee 45
4.2.2.1 Proses Swap Operator 45
4.2.2.2 Proses Swap Sequences 47
4.2.3 Proses Seleksi Roulette Wheel 49
4.2.4 Proses Fase Onlooker Bee 51
4.2.4.1 Proses Insert Operator 51
4.2.4.2 Proses Insert Sequences 53
4.2.5 Proses Scout Bee 55
BAB 5 PENGUJIAN DAN ANALISIS 58
5.1 Pengujian..... 58
5.2 Hasil Pengujian 58
5.2.1 Pengujian Pengaruh Jumlah Bee Colony Terhadap Hasil Solusi Terbaik..... 58

5.2.2 Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik 59
5.2.3 Pengujian Pengaruh Banyaknya Size Problem Terhadap Hasil Solusi Terbaik 60
5.3 Hasil Pengujian 60
5.3.1 Analisis Pengujian Pengaruh Jumlah Bee colony Terhadap Hasil Solusi Terbaik 60
5.3.2 Analisis Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik 61
5.3.3 Analisis Pengujian Pengaruh Banyaknya Size Problem Terhadap Hasil Solusi Terbaik 62
BAB 6 KESIMPULAN 64
6.1 Kesimpulan 64
6.2 Saran 64
DAFTAR PUSTAKA 65

DAFTAR TABEL

Tabel 2.1 Kajian Pustaka 5
Tabel 2.2 Daftar Wisata Kuliner di Malang 7
Tabel 2.3 Pseudocode Algoritma Bee Colony 10
Tabel 3.1 Nama Wisata Kuliner 28
Tabel 3.2 Jarak antar tempat wisata kuliner Kuliner 29
Tabel 3.3 Pembentukan rute dengan metode nearest neighbor 30
Tabel 3.4 Perhitungan swap operator 30
Tabel 3.5 Nilai fitness hasil swap operator 30
Tabel 3.6 Perbandingan nilai fitness awal dengan fitness hasil swap operator 31
Tabel 3.7 Individu Terpilih 31
Tabel 3.8 Perhitungan Swap Sequence Bee ke 1 31
Tabel 3.9 Perhitungan Swap Sequence Bee ke 2 32
Tabel 3.10 Perhitungan Swap Sequence Bee ke 3 32
Tabel 3.11 Perhitungan Swap Sequence Bee ke 4 33
Tabel 3.12 Individu Terpilih 33
Tabel 3.13 Perhitungan Probabilitas 34
Tabel 3.14 Hasil Range 34
Tabel 3.15 Seleksi Roulette Wheel 34
Tabel 3.16 Perhitungan Insert Operator 35
Tabel 3.17 Perhitungan Fitness Hasil Insert Operator 35
Tabel 3.18 Perbandingan Nilai Fitness Hasil Insert Operator 35
Tabel 3.19 Individu Terpilih 36
Tabel 3.20 Perhitungan insert Sequence Bee ke 1 36
Tabel 3.21 Perhitungan insert Sequence Bee ke 2 36
Tabel 3.22 Perhitungan insert Sequence Bee ke 3 37
Tabel 3.23 Perhitungan insert Sequence Bee ke 4 37
Tabel 3.24 Individu Awal 38
Tabel 3.25 Individu Terpilih 38
Tabel 3.26 Solusi Terbaik 38
Tabel 5.1 Hasil Pengujian Pengaruh Jumlah Bee Colony Terhadap Hasil

Solusi 59
Tabel 5.2 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik 59
Tabel 5.3 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik 60

DAFTAR GAMBAR

Gambar 2.1 Contoh Graf Rute Perjalanan 8
Gambar 3.1 Diagram Alur Penelitian 17
Gambar 3.2 Diagram Alur Perancangan Sistem 19
Gambar 3.3 Diagram Alur Bee Colony 20
Gambar 3.4 Diagram Alur Generated Individu 21
Gambar 3.5 Diagram Alur Perhitungan Fitness 22
Gambar 3.6 Diagram Alur Fase Employeed Bee 23
Gambar 3.7 Diagram Alur Fungsi Swap Operator 24
Gambar 3.8 Diagram Alur Fungsi Swap Sequence 26
Gambar 3.9 Diagram Alur Fase Onlooker Bee 27
Gambar 3.10 Diagram Alur Fase Scout Bee 28
Gambar 5.1 Analisis Pengujian Pengaruh Jumlah Bee Colony Terhadap Hasil Solusi..... 61
Gambar 5.2 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik..... 62
Gambar 5.3 Hasil Pengujian Pengaruh Banyaknya Size Problem Terhadap Hasil S Solusi Terbaik.....63

BAB 1 PENDAHULUAN

1.1 Latar belakang

Wisata kuliner saat ini sudah menjadi sebuah jenis wisata yang diminati masyarakat dan memliki dampak yang sangat banyak bagi perkembangan sebuah daerah. Salah satu nilai penting dari wisata kuliner adalah menumbuh kembangkan potensi makanan asli daerah di Indonesia yang sepertinya sudah mulai tergeser oleh produk-produk asing ataupun makanan asing. Kekayaan budaya Indonesia pada bidang kuliner dapat ditandai dengan beragamnya jenis masakan dengan citarasa dan sajian khas. Sejalan dengan itu munculah pusat-pusat wisata kuliner di berbagai kawasan di Indonesia, salah satunya adalah Kota Malang.

Wisata kuliner merupakan aktifitas yang biasa dilakukan untuk menghilangkan rasa jenuh akan makanan dan minuman yang biasa disantap sehari-hari. Wisata kuliner dapat menemukan suasana baru yang dapat memberikan atau bahkan menambah selera dan nafsu makan. Maka tidak jarang beberapa orang rela menghabiskan biaya yang tidak sedikit untuk sekedar mencicipi rasa makanan pada suatu daerah atau bahkan negara tertentu. Namun ada juga sebagian orang yang sangat teliti dengan pengeluarannya saat akan melakukan wisata kuliner. Salah satu faktor yang dapat mempengaruhi besarnya pengeluaran saat melakukan wisata kuliner adalah biaya transportasi. Hal yang berhubungan erat dengan transportasi adalah jarak tempuh yang dilalui oleh para penikmat kuliner menuju tempat wisata kuliner yang dituju. Umumnya masalah yang timbul ialah jauhnya jarak antar tempat wisata kuliner.

Masalah setiap penikmat wisata kuliner yang ingin melakukan kunjungan wisata kuliner biasanya membutuhkan informasi-informasi yang dapat mendukung suatu perjalanan. Beberapa diantaranya adalah informasi tentang tujuan tempat wisata kuliner dan rute perjalanan. Informasi mengenai rute perjalanan ini menjadi sangat penting bagi para penikmat wisata kuliner, karena Indonesia memiliki gambaran yang kurang baik dalam akses layanan transportasi. Kondisi jalan di Indonesia yang kebanyakan terjal dan pembangunan jalan yang kurang merata merupakan salah satu faktor penyebab ketidaknyamanan para penikmat wisata kuliner. Banyak penikmat wisata kuliner yang merasa terlalu jauh dalam menempuh perjalanan menuju tempat wisata kuliner dikarenakan salah dalam pemilihan rute yang ditempuh. Karena Malang memiliki tempat wisata kuliner yang lumayan banyak, maka dibutuhkan optimasi dalam pencarian rute optimum jalur perjalanan dari posisi awal menuju posisi tujuan pada suatu peta lokasi. Hal ini dapat dimanfaatkan oleh para penikmat wisata kuliner dalam memilih jalur yang akan ditempuh agar perjalanan menjadi lebih nyaman dan efisien dari segi jarak tempuh serta biaya.

Dalam perjalanan, penentuan rute terpendek sangat dibutuhkan agar perjalanan menjadi lebih efisien dari segi jarak tempuh, waktu dan biaya yang dikeluarkan sampai tempat tujuan. Namun dalam penentuan rute terpendek terdapat kesulitan yang timbul, yakni terdapat banyak pilihan rute yang ada pada tiap daerah karena dalam kondisi yang sebenarnya setiap perjalanan menuju tempat wisata memiliki banyak rute yang dapat dilalui. Untuk membantu dalam menentukan rute terpendek dapat digunakan peta konvensional dan memilih mana rute yang dianggap terpendek dari posisi awal ke posisi tujuan. Namun hal ini dirasa kurang maksimal dan dapat memperlambat waktu karena harus memilih dan menentukan sendiri dari banyak rute yang ada dan melakukan perhitungan sendiri manakah rute yang terpendek dari posisi awal ke posisi tujuan. Melihat dari permasalahan yang ada maka diperlukan sebuah sistem yang dapat mengoptimasi rute terpendek sebuah perjalanan dari posisi awal sampai posisi tujuan.

Pada penelitian sebelumnya pernah dilakukan penelitian mengenai Artificial Bee Colony Algorithm untuk Menyelesaikan Travelling Salesman Problem. Dalam penelitian ini disimpulkan bahwa 3 dari 7 data benchmark mencapai nilai optimal dalam hal ini jarak terpendek. Secara garis besar, semakin besar size problem, dalam hal ini jumlah kota yang diproses, tingkat kesalahan juga semakin meningkat (Amri, et al., 2012). Pada penelitian lain Penerapan Bee Colony Optimization Algorithm untuk Penentuan Rute Terpendek (Studi Kasus: Objek Wisata Daerah Istimewa Yogyakarta) dijelaskan bahwa penggunaan algoritma bee colony dapat menentukan rute terpendek dengan baik (Danuri & Prijodiprodjo, 2013). Sebelumnya pernah dilakukan penelitian mengenai Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO). Dalam penelitian ini disimpulkan bahwa Algoritma Ant Colony tidak dapat memperkirakan jumlah iterasi yang terbaik. Karena jika jumlah iterasi terlalu kecil, maka hasil yang didapat tidak akurat, sebaliknya jika jumlah iterasi terlalu besar, maka nilai lintasan terbaik semakin lama semakin besar dan tidak semakin kecil (Maria, Sinaga, & Helena, 2012).

Melihat dari beberapa penelitian sebelumnya, maka dalam skripsi ini diusulkan judul "Sistem Optmasi Rute Tempat Wisata Kuliner di Malang menggunakan Algoritma Bee Colony". Algoritma Bee Colony dipilih karena sederhana, fleksibel (Karaboga, 2005) dan memiliki kemampuan untuk keluar dari local minimum dan dapat secara efisiensi digunakan untuk multimodal dan multivariable optimasi fungsi (Karaboga & Basturk, 2007). Menurut Chong, Algoritma Bee Colony mampu menyelesaikan permasalahan TSP lebih baik dibandingkan dengan algoritma lain yang juga didasarkan pada kecerdasan berkelompok. Selain itu Algortima Bee Colony telah banyak diterapkan dalam beberapa permasalahan diantaranya Fuzzy Clustering, Flexible Job Shop Scheduling Problem, Numerical Function dan Job Shop Scheduling (Chong, et al., 2006).

Penulis berharap dari penelitian ini dapat dihasilkan optimasi dengan nilai akurasi yang tinggi sehingga penerapan algortima Bee Colony pada optimasi rute tempat wisata kuliner ini bisa benar-benar membantu para penikmat wisata kuliner saat melakukan perjalanan. Sehingga para penikmat wisata kuliner dapat

merasakan efisiensi saat berwisata dari waktu, tenaga, dan biaya yang dikeluarkan.

1.2 Rumusan masalah

Adapun Rumusan Masalah yang ada pada penelitian ini berdasarkan latar belakang diatas adalah:

1. Bagaimana merancang sistem dengan menggunakan metode Algoritma *Bee Colony* untuk optimasi rute tempat wisata kuliner di Malang.
2. Bagaimana nilai fitness yang diperoleh dari hasil implementasi Algoritma *Bee Colony* untuk optimasi rute tempat wisata kuliner di Malang.

1.3 Tujuan

Adapun tujuan yang ada pada penelitian ini dilihat dari rumusan masalahnya adalah:

1. Merancang sistem Algoritma *Bee Colony* untuk optimasi rute tempat wisata kuliner di Malang.
2. Menghitung nilai *fitness* yang diperoleh dari hasil implementasi Algoritma *Bee Colony* untuk optimasi rute tempat wisata kuliner di Malang.

1.4 Manfaat

Harapannya dalam penelitian ini dapat memberikan manfaat bagi semua pihak. Adapun manfaat yang diperoleh dari pengimplementasian algoritma *Bee Colony* ini adalah:

1. Dapat merancang sistem Algoritma Bee Colony untuk optimasi rute tempat wisata kuliner di Malang
2. Dapat menghitung dan mengetahui nilai fitness yang diperoleh dari hasil implementasi Algoritma Bee Colony untuk optimasi rute tempat wisata kuliner di Malang.

1.5 Batasan masalah

Dalam menghindari pembahasan yang tidak terfokus pada tujuan maka penulis membuat batasan-batasan ruang lingkup sebagai berikut:

1. Studi kasus ditujukan pada wisata kuliner di Kota Malang dan sekitarnya (Kabupaten Malang)
2. Optimasi dilakukan dengan menggunakan jarak antar tempat wisata kuliner di Kota Malang dan sekitarnya (Kabupaten Malang)
3. Optimasi dilakukan pada perencanaan perjalanan dalam waktu satu hari

1.6 Sistematika pembahasan

Dalam penulisan skripsi ini terdiri dari 6 bab dan beberapa sub bagian. Untuk memudahkan dalam penulisan dan pembahasan pada tiap-tiap bab agar sesuai dengan rumusan masalah yang telah dibuat maka digunakan sistematika pembahasan seperti dibawah ini.

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, manfaat,batasan masalah, dan sistematika pembahasan.

BAB II Kajian Pustaka dan Dasar Teori

Menguraikan landasan teori dan teori penunjang serta membahas teori-teori yang mendukung dalam optimasi rute tempat wisata kuliner di Malang menggunakan Algoritma *Bee Colony*.

BAB III Metodologi Penelitian dan Perancangan

Membahas metode yang digunakan dalam penelitian yang terdiri dari Studi Literatur, Metode Pengumpulan Data, Pengolahan Data dan Analisis Data.

Membahas tentang analisa kebutuhan dari aplikasi optimasi rute tempat wisata kuliner di Kota Malang menggunakan Algoritma *Bee Colony* dan kemudian merancang hal-hal yang berhubungan dengan analisa tersebut.

BAB IV Implementasi

Membahas tentang hasil skenario dan analisis pada setiap skenario pengujian yang ada menggunakan *Algoritma Bee Colony*.

BAB V Pengujian dan Analisis

Memuat tentang hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

Bab VI Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab dua berisi uraian dan pembahasan tentang kajian literatur ilmiah dan teori yang berkaitan dengan optimasi tempat wisata kuliner di Malang menggunakan algoritma *bee colony*. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian, yaitu penelitian sebelumnya yang berjudul "*Artificial Bee Colony Algorithm* untuk Menyelesaikan *Travelling Salesman Problem"* (Amri, Nababan, & Syahputra, 2012), "Penerapan *Bee Colony Optimization Algorithm* untuk Penentuan Rute Terpendek (Studi Kasus : Objek Wisata Daerah Istimewa Yogyakarta)" (Danuri, Prijodiprodjo, 2013), dan "*Hybrid Artificial Bee Colony* : Penyelesaian Baru Pohon Rentang Berbatas Derajat" (Izzah, Dewi, Mutrofin, 2013). Penelitian saat ini adalah penerapan *Algoritma Bee Colony* untuk optimasi rute wisata kuliner di Malang. Dasar teori yang diperlukan adalah: wisata kuliner, wisata kuliner di Malang, optimasi, *travelling salesman problem* (TSP), dan *algoritma bee colony.*

2.1. Kajian Pustaka

Pada kajian pustaka membahas mengenai penelitian sebelumnya yang berkaitan dengan skripsi ini. Kajian literatur ilmiah ditunjukkan pada Tabel 2.1.

Tabel 2. 1 Kajian Pustaka

No	Judul	Obyek (*Input* dan Parameter)	Metode	*Output* (Hasil dan Akurasi)
1	*Artificial Bee Colony Algorithm* untuk Menyelesaikan *Travelling Salesman Problem*(Amri, Nababan, & Syahputra, 2012)	-Banyaknya *size problem* (kota) -Titik koordinat tiap kota -*Colony size* -Limit -Maksiterasi	*Artificial Bee Colony Algorithm*	-Jalur terpendek dari TSP -Waktu tercepat
2	Penerapan *Bee Colony Optimization Algorithm* untuk Penentuan Rute Terpendek (Studi Kasus : Objek Wisata Daerah Istimewa Yogyakarta) (Danuri, Prijodiprodjo, 2013)	-Posisi asal -Posisi tujuan -NBee -Nilai probabilitas untuk jalur yang dipilih -Parameter pengontrol tingkat signifikan jarak -Faktor skala *waggle dance*	*Bee Colony Optimization Algorithm*	-Rute terpendek
3	*Hybrid Artificial Bee Colony* : Penyelesaian Baru Pohon Rentang Berbatas Derajat (Izzah, Dewi, Mutrofin, 2013)	-NBee (*employee* dan *onlooker*) -*Pc* (probabilitas terpilihnya induk untuk *crossover*)	*Hybrid Artificial Bee Colony*	-Perbandingan kinerja dari algoritma ABC dengan algoritma HABC

No	Judul	Obyek (*Input* dan Parameter)	Metode	*Output* (Hasil dan Akurasi)
		-dc (*degree constrained*) *-MaxIterasi*		

Pada penelitian sebelumnya pernah dlakukan penelitian mengenai *Artificial Bee Colony Algorithm* untuk Menyelesaikan *Travelling Salesman Problem*. Dalam penelitian ini disimpulkan bahwa 3 dari 7 data benchmark mencapai nilai optimal dalam hal ini jarak terpendek. Secara garis besar, semakin besar size problem, dalam hal ini jumlah kota yang diproses, tingkat kesalahan juga semakin meningkat (Amri, et al., 2012). Pada penelitian lain Penerapan *Bee Colony Optimization Algorithm* untuk Penentuan Rute Terpendek (Studi Kasus : Objek Wisata Daerah Istimewa Yogyakarta) dijelaskan bahwa penggunaan algoritma *bee colony* dapat menentukan rute terpendek dengan baik. Terdapat dua proses utama pada saat penelusuran jalur yaitu *forward* dan backward. Algoritma *bee colony optimization* bekerja pada proses *forward*. Nilai probabilitas suatu jalur dijadikan dasar pada proses transisi alur kemudian durasi *waggle dance* dari tiap lebah yang berhasil menemukan posisi tujuan akan dijadikan rute pilihan. Jumlah lebah yang dilepas sangat mempengaruhi dalam menemukan rute-rute yang bisa dilalui. Semakin banyak jumlah lebah yang lepas semakin besar peluang ditemukannya rute terpendek (Danuri & Prijodiprodjo, 2013). Kemudian pada penelitian *Hybrid Artificial Bee Colony* : Penyelesaian Baru Pohon Rentang Berbatas Derajat dijelaskan bahwa algoritma *bee colony* memiliki kelebihan dibandingan dengan algoritma yang lain (G*enetic Algorithm, Hybrid Artificial Bee Colony*) yaitu sangat sederhana dan fleksibel (Izzah, et al., 2013).

Melihat dari beberapa penelitian sebelumnya, maka dalam skripsi ini diusulkan judul "Optimasi Rute Tempat Wisata Kuliner di Malang menggunakan Algoritma *Bee Colony*". Algoritma *Bee Colony* dipilih karena sederhana, fleksibel (Karaboga, 2005) dan memiliki kemampuan untuk keluar dari *local minimum* dan dapat secara efisiensi digunakan untuk multimodal dan multivariable optimasi fungsi (Karaboga & Basturk, 2007). Algoritma *Bee Colony* merupakan algoritma yang digunakan untuk memecahkan masalah yang terinspirasi oleh tingkah laku kolektif dari koloni sosial serangga yaitu lebah. Permasalahan *Travelling Salesman Problem* (TSP) akan diselesaikan dengan algoritma *Bee Colony* karena algoritma tersebut dapat mendapatkan jarak tempuh yang minimal dari sebuah *trail* yang tertutup (Amri, et al., 2012).

2.2. Wisata Kuliner

Kuliner adalah hasil olahan bahan (bahan mentah ataupun jadi) berupa masakan. Masakan tersebut berupa lauk pauk, makanan, dan minuman. Setiap daerah di Indonesia memiliki citarasa dan ciri khas makanan tersendiri, maka dari itu setiap daerah memiliki tradisi kuliner yang berbeda. Setiap daerah juga memiliki nama masakan yang berbeda-beda. Apalagi Indonesia sangat kaya akan resep masakan kuliner khas yang secara turun temurun selalu diwariskan dalam

setiap keluarga. Sehingga wisata kuliner dapat diartikan sebagai jenis wisata minat khusus yang menitik beratkan atau mengkhususkan pada kegiatan perjalanan untuk menikmati kuliner atau makanan sehingga mendapatkan kepuasan. Wisata kuliner merupakan salah satu pengembangan wisata minat khusus yang mengutamakan berwisata untuk menikmati makanan dan minuman dengan tujuan bersenang senang (Permata, 2011).

Wisata kuliner akan menawarkan pengalaman masakan lokal yang memiliki cita rasa khas pada tiap daerah. Bahan-bahan yang diolah, tampilan makanan, dan cita rasa yang nikmat menurut selera penikmat kuliner juga menjadi tolak ukur dari masakan lokal, sehingga masakan tersebut pada umumnya diterima dengan baik oleh penikmat kuliner. Potensi yang besar pada masakan lokal ini dapat menjadi sebuah terobosan baru untuk memulai atau melanjutkan upaya pembangunan kepariwisataan.

2.3. Wisata Kuliner Malang

Malang merupakan wilayah yang berada di Provinsi Jawa Timur yang memiliki potensi wisata kuliner tidak kalah dengan kota-kota tetangganya di Jawa Timur. Banyak tempat tujuan yang dapat dijadikan rujukan wisata kuliner. Pada Tabel 2.2 ditunjukkan beberapa daftar wisata kuliner di Malang yang menarik untuk dikunjungi.

Tabel 2. 2 Daftar Wisata Kuliner di Malang

No	Wisata Kuliner	Alamat	Menu Favorit
1	Toko Oen	Jl. Basuki Rahmat No. 5, Malang	Es Krim
2	Mie Gajah Mada Malang	Jl. Pasar Besar No.17 A, Malang	Mie pangsit
3	Bakso President	Jl. Batanghari No. 5, Malang	Bakso
4	Inggil Museum Resto	Jl. Gajahmada No. 2, Malang	Sate komoh
5	Warung Ronde Titoni	Jl. Zainal Arifin No.17, Malang	Angsle dan ronde
6	Rumah Makan Cairo	Jl. Kapten Piere Tenedan Malang	Olahan daging kambing
7	Pecel Kawi	Jl. Kawi Atas No.43B, Malang	Pecel
8	Sate Landak Bu Ria	JL. Raja Bugis No. 47, Malang	Sate landak
9	Depot Hok Lay	Jl. KH Achmad Dahlan No. 10, Malang	Lumpia dan pangsit cwiemie
10	Puthu Lanang Celaket	Jl. Jaksa Agung Suprapto, Malang	Puthu

2.4. Optimasi

Optimasi adalah proses memaksimasi atau meminimasi suatu fungsi tujuan dengan tetap memperhatikan pembatas yang ada. Optimasi memegang peranan penting dalam mendesain suatu sistem. Dengan optimasi, suatu sistem dapat

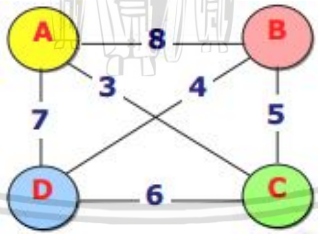
menghasilkan ongkos yang lebih murah atau profit yang lebih tinggi, menurunkan waktu proses dan sebagainya. Optimasi dalam waktu sekarang memerlukan bantuan software untuk menyelesaikan permasalahan yang ditemukan untuk mendapatkan solusi yang optimal dengan waktu komputasi yang lebih singkat. Teknik-teknik optimasi tidak banyak berubah dalam beberapa tahun belakangan ini. Tetapi aplikasi dari teknik optimasi telah menjamur di berbagai bidang secara cepat (Iqbal, 2016).

Sedangkan menurut definisi, Optimasi adalah sebuah proses produksi yang lebih Efisien (lebih kecil dan atau lebih cepat). Sehingga Optimasi memang sangat diperlukan untuk hal apapun. Optimasi bisa juga diartikan, untuk membuat sesuatu sebaik-baiknya agar berjalan semaksimal mungkin (Dyardian, 2015).

2.5. *Travelling Salesman Problem* (TSP)

Travelling Salesman Problem (TSP) merupakan masalah klasik mencari rute terpendek yang bisa dilalui salesman ketika ingin mengunjungi beberapa kota tanpa harus mendatangi kota yang sama lebih dari satu kali (Kusrini & Istiyanto, 2008). Penyelesaian eksak untuk masalah TSP ini mengharuskan perhitungan terhadap semua kemungkinan rute yang dapat diperoleh, kemudian memilih salah satu rute yang terpendek.

Apa yang dilakukan dalam TSP adalah membentuk sebuah tour. Operator yang bisa digunakan untuk masalah TSP adalah pencarian urutan semua lokasi untuk memilih lokasi yang belum pernah terpilih satu demi satu sehingga dihasilkan satu rute kunjungan yang lengkap dari lokasi awal kemudian mengunjungi semua lokasi yang lain tepat satu kali dan akhirnya kembali ke lokasi awal. Sebagai ilustrasi ditunjukkan pada Gambar 2.1.



Gambar 2.1 Contoh Graf Rute Perjalanan

Diasumsikan bahwa simpul awal dan simpul akhir adalah 1. Suatu graf TSP dengan 4 simpul tersebut dikonversi menjadi sebuah pohon pencarian yang menghasilkan (4-1)! = 6 kemungkinan kunjungan. Sedangkan untuk 50 kota, terdapat sebanyak (50-1)! = 6,08×1062 kemungkinan urutan kunjungan.

Pada TSP asimetris, biaya dari kota 1 ke kota 2 tidak sama dengan biaya dari kota 2 ke kota 1. Dengan *n* kota, besarnya ruang pencarian adalah

$$\frac{n!}{n} = (n-1)! \qquad (2.1)$$

Jumlah jalur yang mungkin adalah permutasi dari jumlah node dibagi dengan jumlah node. Misalkan terdapat jalur 1, 2, 3. Jika dipahami secara siklus, sebuah

jalur dengan urutan $n123 - n231 - n312$. Tetapi jalur dengan urutan $n123 \neq n321$.

2.6. Algoritma *Bee Colony*

Bee Colony merupakan kecerdasan buatan yang termasuk dalam golongan *swarm intelligence* dengan menirukan cara kerja koloni lebah dalam mencari nektar. Dalam menentukan sumber makanan kemampuan koloni lebah dibagi menjadi tiga kelompok yaitu lebah pengintai, lebah penjelajah dan lebah pekerja. Ketiga lebah ini melakukan suatu pekerjaan untuk menentukan besar dan letak sumber nektar kemudian mengingat dan membandingkannya dengan sumber lain. Pada akhir fungsi dipilih nectar dengan jarak yang optimal.

Kehidupan lebah yang melakukan seleksi pada saat mencari makan terdiri dari tiga komponen penting yaitu sumber makanan, lebah pekerja dan lebah *unemployed*. Seleksi tersebut mendefinisikan dua perilaku pada lebah yang dianggap penting yaitu pencarian sumber nektar dan rute optimal yang diperoleh.

1. Sumber makanan. Nilai dari sumber makanan tergantung dari tiga factor yaitu jarak sarang, banyaknya sumber makanan, dan tingkat kemudahan saat mengumpulkan sumber makanan.
2. Lebah pekerja. Lebah ini bertugas untuk mencari sumber makanan. Lebah pekerja juga bertugas untuk membawa informasi penting tentang letak dari sumber makanan, jarak, dan arah sumber makanan dari sarang.
3. Lebah *unemployed*. Ada dua jenis lebah *unemployed* yaitu lebah *scout* yang bertugas mencari letak sumber makanan di sekitar sarang untuk mendapatkan sumber makanan baru dan lebah *onlooker* yang bertugas menunggu di sarang kemudian mendapatkan sumber makanan melalui berbagai informasi yang didapat dari lebah pekerja. Jumlah rata-rata lebah *scout* hanya sekitar 5-12% dari jumlah lebah secara keseluruhan.

Pertukaran informasi di antara lebah adalah kejadian yang paling penting. Saat memeriksa semua sarang, lebah dapat membedakan beberapa bagian yang terdapat dalam sarangnya. Dan bagian yang penting dalam sarang adalah pertukaran informasi yang disebut dengan *dancing area*. Komunikasi di antara para lebah yang berkaitan dengan sumber makanan terjadi di *dancing area* disebut juga tarian *waggle dance*.

Karena informasi tentang sumber makanan tersedia untuk lebah *onlooker* di *dancing area*, maka memungkinkan lebah tersebut untuk dapat menonton berbagai tarian lebah dan memutuskan untuk bekerja pada sumber yang paling menguntungkan.

Ada peluang yang lebih besar bagi lebah *onlooker* untuk memilih sumber-sumber makanan yang lebih menguntungkan, karena lebih banyak informasi sumber makanan yang lebih menguntungkan tersebut. Lebah pekerja membagi

informasi sesuai dengan probabilitas yang sebanding dengan profitability dari sumber makanan, dan membagi informasi melalui *waggle dance* dengan durasi yang lebih lama. Oleh karena itu, rekruitmen sebanding dengan *profitability* dari sumber makanan.

Diasumsikan bahwa ada dua sumber makanan yang ditemukan, yaitu Y dan Z kemudian lebah pencari makan yang potensial akan mulai sebagai lebah *unemployed*. lebah *unemployed* tidak memiliki informasi tentang sumber makanan di sekitar sarang. Ada dua opsi pilihan untuk lebah tersebut yaitu:

1. **Bisa menjadi *scout***

   Menjadi *scout* dan mulai mencari sumber makanan disekitar sarang secara spontan.

2. **Bisa menjadi rekrutan**

   Menjadi rekrutan setelah menonton *waggle dance* dan mulai mencari sumber makanan.

Setelah menemukan sumber makanan, lebah tersebut menggunakan kemampuannya sendiri untuk mengingat lokasi sumber makanan dan kemudian mulai mengeksploitasinya segera. Oleh karena itu, lebah tersebut akan menjadi lebah pekerja. Lebah pekerja mengambil nektar lalu kembali ke sarang dan membongkar nektar pada tempat persediaan makanan. Setelah pembongkaran makanan, lebah pekerja tersebut memiliki tiga opsi yaitu menjadi pengikut tidak terikat setelah meninggalakan sumber makanan, melakukan *waggle dance* dan kemudian merekrut lebah lainnya sebelum kembali ke sumber makanan yang sama, atau meneruskan untuk mencari makanan di sumber makanan semula tanpa merekrut lebah lainnya. Pada Tabel 2.3 dijelaskan mengenai *pseudocode* yang digunakan untuk menyelesaikan permasalahan TSP

Tabel 2.3 *Pseudocode* Algoritma *Bee Colony*

Langkah	Keterangan
1	**Inisialisasi semua parameter yang diperlukan**, yaitu *colony size,* maksiterasi, *limit.*
2	Fase *initial* - Untuk tiap *population size, initial solution* dihasilkan secara *random* - Hitung *fitness* menggunakan persamaan (2.3)
3	**Iterasi = 1**
4	**Fase Employed bee** - Untuk tiap *employed bee, update employed bee* dengan menggunakan *neighborhood operator, swap operator* dan swap sequence. - Hitung *fitness* setiap *employed bee* menggunakan persamaan 2.3. - Jika solusi yang baru hasilnya lebih baik dari sebelumnya gantikan solusi lama dengan solusi baru, jika tidak tambahkan *limit* dengan 1.

Tabel 2.3 *Pseudocode* Algoritma *Bee Colony*

Langkah	Keterangan
5	Hitung probabilitas tiap *employed bee.*
6	Fase *onlooker bee* - Untuk tiap *onlooker bee,* pilih solusi dari *employed bee* dengan menggunakan persamaan 2.7 dan gunakan teknik *roulette wheel selection.* - Tentukan solusi yang baru *employed bee* terpilih dengan menggunakan *neighborhood operator, insert operator* dan *insert sequence.* - Hitung *fitness* setiap *onlooker bee* menggunakan persamaan 2.3
7	**Fase *Scout Bee*** - Hitung jumlah trial dan simpan jumlah maksimal, M, untuk tiap bee yang tidak mengalami peningkatan solusi. - Jika M > *limit,* tinggalkan solusi yang tidak mengalami peningkatan, jika M < *limit, scout bee* memilih solusi sebelumnya
8	**Hapalkan solusi terbaik dicapai saat ini**
9	***Iterasi = iterasi* + 1**
10	Sampai *iterasi* = maksiterasi

Sumber: Diadaptasi dari Amri, Nababan, Syahputra (2012)

Sesuai dengan Tabel 2.3, penjabaran mengenai langkah-langkah *pseudocode* dari algoritma *Bee Colony* (Amri, et al., 2012):

1. ***Input* parameter**

   Tahapan yang pertama dilakukan, yaitu memasukkan parameter-parameter yang dibutuhkan dalam penyelesaian permasalahan TSP dan memasukkan nama file dataset.tsp yang akan digunakan. Parameter-parameter yang dimaksud meliputi:

   a. File TSP, yang memuat dataset TSP untuk selanjutnya diolah di dalam sistem. *File* tsp ini memiliki tiga informasi yang digunakan dalam pengolahan sistem, yaitu banyaknya *size problem* (kota), titik koordinat tiap kota. Untuk tiap titik koordinat dapat dihasilkan jarak tiap titik untuk mempermudah pengolahan data.

   b. *Colony size,* merupakan jumlah dari *employed* bee ditambah *onlooker bee* yang akan digunakan dalam *system.* Dimana jumlah *employed bee = onlooker bee* dalam hal ini disebut *population size.*

   c. *Limit,* merupakan batasan dari *population size* yang tidak mengalami peningkatan kualitas untuk sejumlah *iterasi.*

d. Maksiterasi, merupakan banyaknya iterasi yang akan dilakukan. Dalam hal ini disebut juga kriteria berhenti

2. **Fase *Initial***

Fase *initial* merupakan proses untuk mendapatkan *initial solution* untuk tiap *employed bee*. Dimana tiap *bee* mereperesentasikan kandidat perjalanan (*tour*), dan dimana elemen pertama dan terakhir merepresentasikan kota asal. Untuk menghasilkan *initial solution* dilakukan secara *random* (Otri, 2011).

Sebagai contoh kota yang harus dilalui adalah kota (1-3-5-7-9) untuk menghasilkan bee ke 1 (1-5-3-7-9), pembangkit bilangan acak digunakan untuk menghasilkan rute yang dapat dilalui, dan sekali bilangan dihasilkan tidak diperbolehkan muncul lagi. Bilangan pertama yang dihasilkan akan dimunculkan pada akhir *tour* untuk memberikan *tour* tertutup. Kemudian pembangkit bilangan acak menghasilkan bilangan lain sampai semua bilangan dihasilkan dan membentuk sebuah *tour* penuh dan tertutup.

Setelah setiap *Employeed Bee* menghasilkan *initial solution*, proses berikutnya adalah menghitung *fitness* dari tiap *Employeed Bee* dengan menggunakan persamaan 2.3.

$$fit_i = \begin{cases} \frac{1}{1+f_i} & if\,(f_i \geq 0) \\ 1 + abs\,(f_i) & if\,(f_i < 0) \end{cases} \qquad (2.3)$$

Keterangan:

fit_i = Nilai *Fitness Bee* ke *i*

f_i = Jumlah jarak tiap tempat wisata kuliner

Kemudian dideklarasikan juga *limit* dan status awal dengan nilai 0. *Initial solution* yang terbentuk pada fase *initial* selanjutnya akan diperbaiki pada *improvement solution*. Pada *improvement solution* terdiri dari fase *Employeed Bee* dan fase *Onlooker Bee*.

3. Improvement Solution

Improvement solution berada pada fase *Employeed Bee* dan *Onlooker Bee*. *Improvement solution* digunakan untuk memperbarui solusi dengan metode *neighborhood operator*. *Neighborhood operator* yang akan digunakan yaitu *Swap Operator* dan *Swap Sequence* pada fase *Employeed Bee* dan *Insertion Operator* dan *Insertion Sequence* pada fase *Onlooker Bee.*

a. Fase *Employeed Bee*

- *Swap Operator* (SO) atau *Random Swap*

Swap Operator digunakan untuk menentukan 2 wisata kuliner dari solusi yang akan di tukar, dimana wisata kuliner1 dan wisata kuliner2 yang

ditukar menggunakan teknik random dan bilangan yang dihasilkan untuk wisata kuliner1 ≠ wisata kuliner2. Sebagai contoh bilangan random yang dihasilkan adalah SO(2, 4). Maka solusi baru yang dihasilkan dari initial solution dengan SO adalah sebagai berikut:

$$\text{Bee}_i = \text{Bee}_i + \text{SO} \quad (2.4)$$

Keterangan :

Bee_i = Individu Bee ke i

SO = Swap Operator (wisata kuliner1, wisata kuliner2)

Wisata kuliner1 ≠ Wisata kuliner2

Sebagai contoh untuk bee-1, secara random nilai dari swap operator adalah SO(2, 4), maka proses swap yang terjadi dapat dilihat pada penjelasan berikut ini.

$$\text{Bee}_1 = (0, 2, 3, 4, 1, 5) + \text{SO}(2, 4)$$

0	2	3	4	1	5
0	4	3	2	1	5

- *Swap sequence*(SS) atau *Random Swap sequence*

Swap sequence merupakan kumpulan dari *swap operator*, solusi baru untuk tiap *Employeed Bee* dihasilkan berdasarkan *tour* sebelumnya dan *Swap sequence*.

$$\text{Bee}_i = \text{Bee}_i + \text{SS} \quad (2.5)$$
$$= \text{Bee}_i + (\text{SO}_1, \text{SO}_2, \text{SO}_3, \dots, \text{SO}_n)$$
$$= (\text{Bee}_i + \text{SO}_1) + \text{SO}_2) + \text{SO}_3) \dots +)\text{SO}_n)$$

Keterangan :

SS = Swap Sequence

Son = Swap Operator ke-n

n = Banyaknya Swap Sequence

Setelah kriteria berhenti dari proses *Employeed Bee* telah terpenuhi, maka dilakukan perhitungan probabilitas dengan menggunakan persamaan 2.7 untuk tiap Employeed Bee. Perhitungan ini akan digunakan pada fase berikutnya.

$$P_{ij} = \frac{F(\theta_i)}{\sum_{k=1}^{s} F(\theta_k)} \quad (2.6)$$

Keterangan :

Pij =Kemungkinan memilih Employeed Bee ke-i

F(θi) = Fitness Value dari Employeed Beeke-i

S = Jumlah Employeed Bee

θi = Posisi dari Employeed Bee

b. Fase Onlooker Bee

- *Insert Operator* (IO) atau *Random Insertion*

Insert Operator digunakan untuk menentukan sebuah wisata kuliner dari solusi yang akan di tukar posisinya, dimana wisata kuliner awal dan posisi tujuan yang ditukar menggunakan teknik random dimana bilangan random yang dihasilkan untuk wisata kuliner1 ≠ tujuan, dan kemudian sisa urutan solusi akan digeser. Sebagai contoh bilangan random yang dihasilkan adalah IO(2, 4). Maka solusi baru yang dihasilkan dari initial tour dengan IO adalah sebagai berikut :

$$Bee_i = Bee_i + IO \quad (2.7)$$

Keterangan :

Bee_i = Individu Bee ke i

IO =Insert Operator (wisata kuliner1, Tujuan)

Sebagai contoh untuk bee-1, secara random nilai dari insert operatoradalah IO(2, 4), maka proses insert yang terjadi dapat dilihat pada penjelasan berikut ini.

$$Bee_1 = (0, 2, 3, 4, 1, 5) + IO(2, 4)$$

0	2	3	4	1	5
0	4	2	3	1	5

- *Insert Sequence*(IS) atau *Random Insertion Sequence*

Insert sequence merupakan kumpulan dari insert operator, solusi baru untuk tiap Employeed Bee dihasilkan berdasarkan tour sebelumnya dan insert sequence.

$$\begin{aligned} Bee_i &= Bee_i + IS \\ &= Bee_i + (IO_1, IO_2, IO_3, \dots, IO_n) \\ &= (Bee_i + IO_1) + IO_2) + IO_3) \dots +)IO_n) \end{aligned} \quad (2.8)$$

Keterangan :

IS = Insert Sequence

IOn = Insert Operator ke-n

n = Banyaknya Insert Sequence

Setelah kriteria berhenti dari proses *Onlooker Bee* telah terpenuhi, selanjutnya memasuki fase *Scout Bee*.

c. Fase Scout Bee

Setelah melalui dua fase improvement solution, fase Employeed Bee dan fase Onlooker Bee, maka akan dilakukan perhitungan kualitas dari masing-masing Employeed Bee. Jumlah Scout Bee disini bersifat dinamis, tergantung pada jumlah Employeed Bee yang telah melebihi limit. Apabila limit dari bee yang melakukan improvement solution melebihi maximum limit yang ditetapkan, maka solusi dari bee tersebut akan dihilangkan dan diganti dengan solusi baru dengan menggunakan teknik random, memperbarui jarak yang dihasilkan, dan menyetel ulang limit kembali menjadi 0.

4. Kriteria Berhenti

Kriteria berhenti dari sistem ini akan dilakukan sebanyak maksiterasi yang didefinisikan. Iterasi akan dilakukan sampai kriteria berhenti terpenuhi, dan selama belum terpenuhi, maka akan mengulang langkah 3 pada tabel 2.3.

BAB 3 METODOLOGI PENELITIAN DAN PERANCANGAN

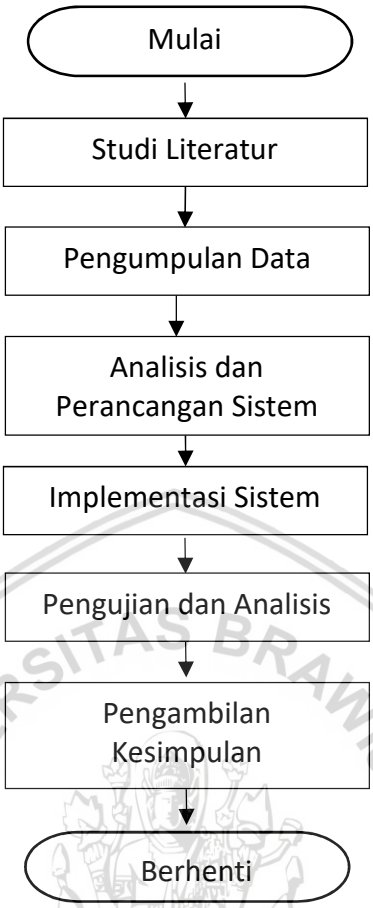
Pada bab metodologi akan dibahas mengenai metode yang akan digunakan dan perancangan yang akan dibuat, meliputi langkah-langkah penelitian hingga gambaran penggunaan metode pada implementasi Algoritma *Bee Colony* untuk optimasi rute wisata kuliner di Malang

3.1 Metodologi

Metodologi penelitian skripsi ini dilakukan dengan enam tahap yaitu:

1. Mempelajari literatur yang terkait dengan penelitian ini, yaitu Algoritma *Bee Colony* dan *Travelling salesman Problem* (TSP).
2. Mengumpulkan data penelitian yang berupa dataset hasil observasi untuk digunakan sebagai data training dan data testing.
3. Melakukan perancangan sistem implementasi Algoritma *Bee Colony* untuk optimasi rute wisata kuliner di Malang.
4. Mengimplementasikan hasil dari perancangan yang telah dibuat sebelumnya menjadi sistem optimasi yang terkomputerisasi.
5. Melakukan pengujian dan analisis terhadap sistem yang dibuat, memeriksa apakah hasil implementasi telah sesuai dengan perancangan yang telah dibuat sebelumnya.
6. Pengambilan kesimpulan dari hasil implementasi sistem.

Pada Gambar 3.1 ditunjukkan diagram alir dari desain metodologi dengan implementasi Algoritma *Bee Colony* untuk optimasi rute wisata kuliner di Malang.



Gambar 3.1 Diagram alur penelitian

3.1.1 Studi Literatur

Metode ini digunakan untuk mendapatkan dasar teori sebagai sumber acuan untuk penulisan skripsi dan pengembangan aplikasi pendukung skripsi. Studi literatur ini meliputi beberapa teori yang nantinya digunakan untuk mengolah data mentah menjadi data yang siap digunakan untuk penelitian. Teori penunjang dan pendukung skripsi ini meliputi jarak antar tempat wisata kuliner kuliner di Kota Malang dan Kabupaten Malang. Sumber atau referensi yang digunakan antara lain buku, jurnal, laporan penelitian dan bantuan *search engine* internet. Kemudian data yang telah diolah tersebut diproses dengan menggunakan Algoritma *Bee Colony* sehingga dapat diimplementasikan dalam sebuah program.

3.1.2 Pengumpulan Data

Metode ini digunakan untuk mendapatkan data sebagai acuan untuk pengembangan *Software*. Data sampel yang dimaksud adalah data jarak antar tempat wisata kuliner kuliner di Kota Malang dan Kabupaten Malang. Data jarak antar tempat wisata kuliner didapat dari *Google Maps*, yang kemudian diolah menjadi data matriks sehingga dapat digunakan dalam perhitungan Algoritma *Bee Colony*.

3.1.3 Analisis dan Perancangan Sistem

Analisis dan perancangan sistem adalah tahap dimulainya merancang suatu sistem yang mampu memenuhi semua kebutuhan fungsional aplikasi. Teori-teori dari pustaka dan data dari sample digabungkan dengan ilmu yang didapat, diimplementasikan untuk merancang dan mengembangkan sistem aplikasi optimasi rute tempat wisata kuliner kuliner di Malang. Perancangan sistem meliputi perancangan proses, data, dan pengujian.

Pada tahap analisis, pertama menyiapkan data numerik yang akan digunakan untuk proses inisialisasi pada perhitungan Algoritma Bee Colony yaitu data jarak antar tempat wisata kuliner kuliner. Selanjutnya membuat model pemecahan masalah dengan membuat model metode yang digunakan dalam sebuah diagram alir agar permasalahan mudah untuk dipecahkan. Selanjutnya menganalisis kebutuhan pada proses penelitian, yaitu kebutuhan hardware dan software.

Kebutuhan hardware yang digunakan meliputi Intel® insideTM 2.50 GHz, RAM 4.00 GB, hardisk 500 GB, dan Monitor 14". Sedangkan kebutuhan software yang digunakan adalah operating system windows 8 64 bit, microsoft office 2013, bahasa pemrograman Java, NetBeans IDE 7.4, dan Enterprise Architect.

Tahap perancangan sistem merupakan tahap pembuatan alur kerja pemecahan masalah optimasi dengan menggunakan algoritma bee colony, penggunaan parameter dan variabel, penggunaan data kedalam algoritma dan formula matematika, desain pengujian dan evaluasi sistem sesuai dengan parameter dan variabel yang berpengaruh terhadap kinerja sistem.

3.1.4 Implementasi Sistem

Implementasi adalah proses pembuatan sistem sesuai dengan rancangan yang telah dibuat sebelumnya. Implementasi aplikasi dilakukan dengan mengacu kepada perancangan sistem aplikasi optimasi rute wisata kuliner di Malang untuk membantu para wisatawan dan penikmat kuliner saat melakukan perjalanan wisata dan mencicipi hidangan kuliner di Kota Malang dan Kabupaten Malang. Sistem tersebut selanjutnya akan diuji dan dianalisis untuk mendapatkan hasil yang terbaik.

3.1.5 Pengujian dan Analisis

Pengujian dilakukan dengan cara memasukkan data uji ke sistem untuk mendapatkan hasil optimasi rute terpendek antar tempat wisata kuliner kuliner dan menganalisis ketepatan sistem dalam menentukan rute terpendek. Pada tahap ini sistem diuji tingkat keberhasilannya yaitu menguji semua fitur apakah sudah sesuai dengan rancangan awal, serta menguji parameter yang telah ada.

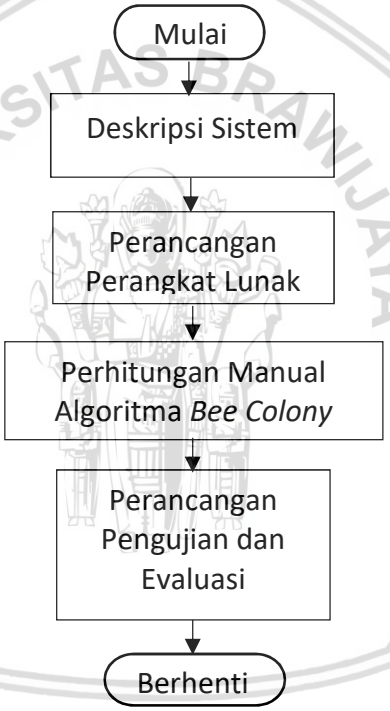
Pengujiannya meliputi pengujian pengaruh jumlah *bee colony* terhadap hasil solusi yang paling baik, pengaruh banyaknya iterasi terhadap hasil solusi terbaik, pengaruh banyaknya *size problem* terhadap hasil solusi terbaik dan pengaruh jenis neighborhood operator terhadap hasil solusi terbaik.

3.1.6 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan telah selesai dilakukan. Kesimpulan diambil untuk menjawab rumusan masalah yang telah dibuat sebelumnya. Tahap paling akhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi selama penelitian dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

3.2 Perancangan Sistem

Tahapan perancangan sistem terdiri dari deskripsi umum sistem, perancangan program aplikasi, perhitungan manual, dan perancangan pengujian dan evaluasi. Gambar 3.2 merupakan model perancangan sistem yang dilakukan pada proses penelitian.



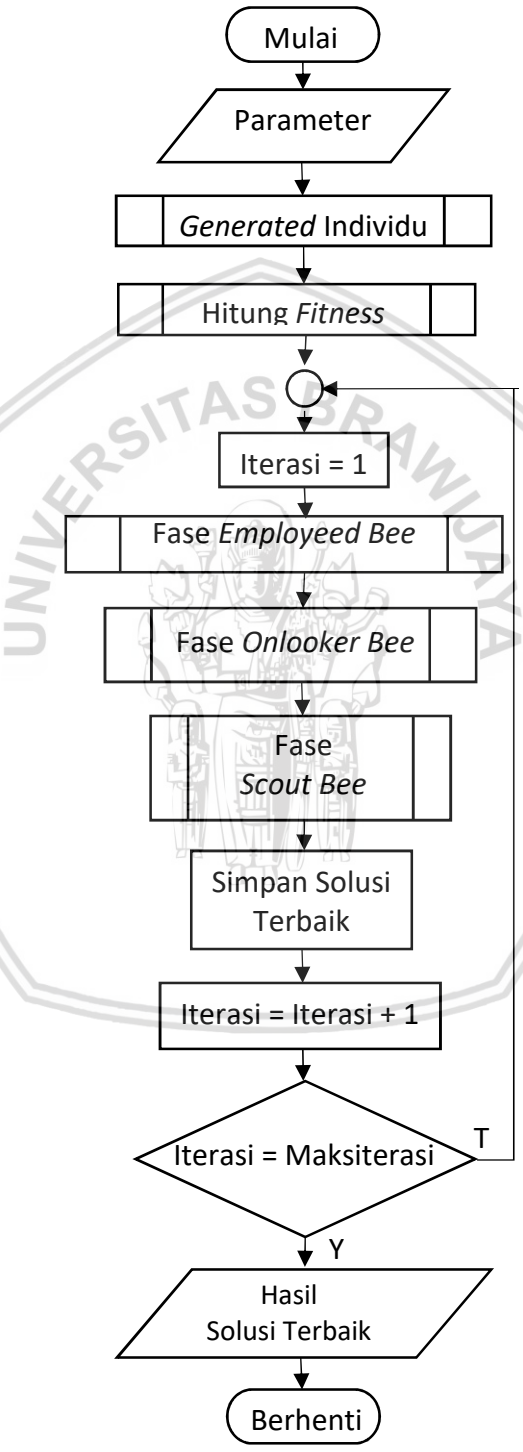
Gambar 3.2 Diagram Alir Perancangan Sistem

3.2.1 Deskripsi Sistem

Sistem ini dibuat untuk menerapkan Algortima *Bee Colony* pada permasalahan optimasi rute tempat wisata kuliner kuliner ke dalam sebuah program, dimana rute tempat wisata kuliner kuliner yang dicari adalah di daerah Kota Malang dan Kabupaten Malang. Sistem akan mengolah data *input* yang dimasukkan oleh user berupa tempat wisata kuliner kuliner yang ingin dikunjungi oleh user. kemudian, sistem akan menghasilkan *output* berupa rute yang akan dilewati oleh para penikmat kuliner dan jarak yang akan dilalui. *Output* merupakan hasil optimasi rute yang dapat dilalui oleh para penikmat kuliner dengan mencari jarak terpendek.

3.2.2 Perancangan Perangkat Lunak

Pada tahap ini proses penerapan metode Algoritma *Bee Colony* untuk permasalahan optimasi yang dijalankan pada program aplikasi digambarkan dalam diagram alir sistem yaitu tahapan proses sistem dengan metode Algortima *Bee Colony* seperti pada Gambar 3.3.

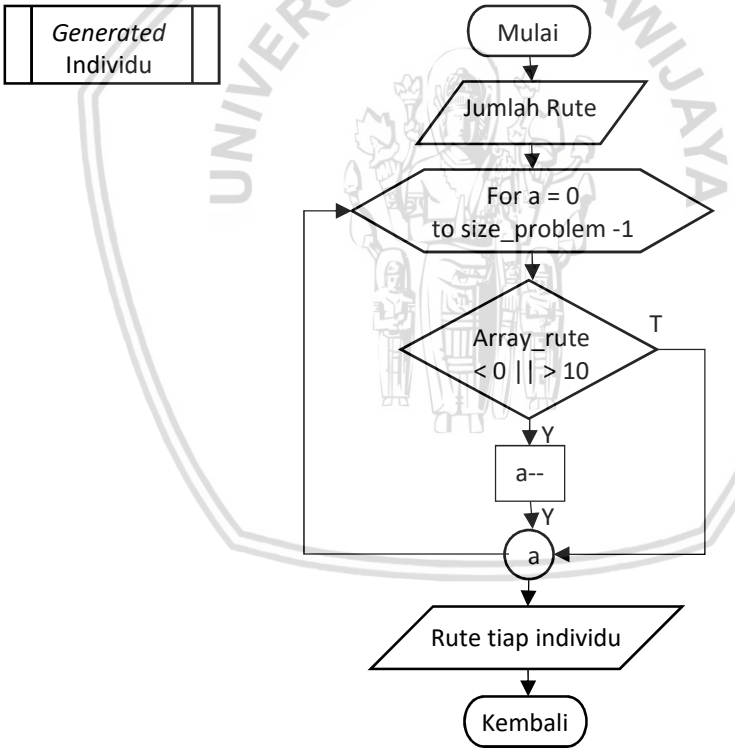


Gambar 3.3 Diagram Alir Bee Colony

Langkah pertama adalah inisialisasi parameter, yang kemudian memasuki fase inisialisasi yaitu *menggenerated* individu dan menghitung *fitness*, selanjutnya data diiteriasi, dilanjutkan dengan fase *employeed bee* kemudian dihitung probabilitasnya di tiap *employeed bee*. Setelah dihitung probabilitasnya, data menuju pada fase *onlooker bee* kemudian ke fase *scout bee* hingga mendapatkan solusi terbaik kemudian disimpan. Data akan terus diiterasi hingga mencapai maksiterasi yang telah ditentukan. Iterasi akan dilakukan sampai kriteria berhenti terpenuhi, dan selama belum terpenuhi, maka akan mengulang langkah 3 pada Gambar 2.3.

3.2.2.1 Fase *Initial*

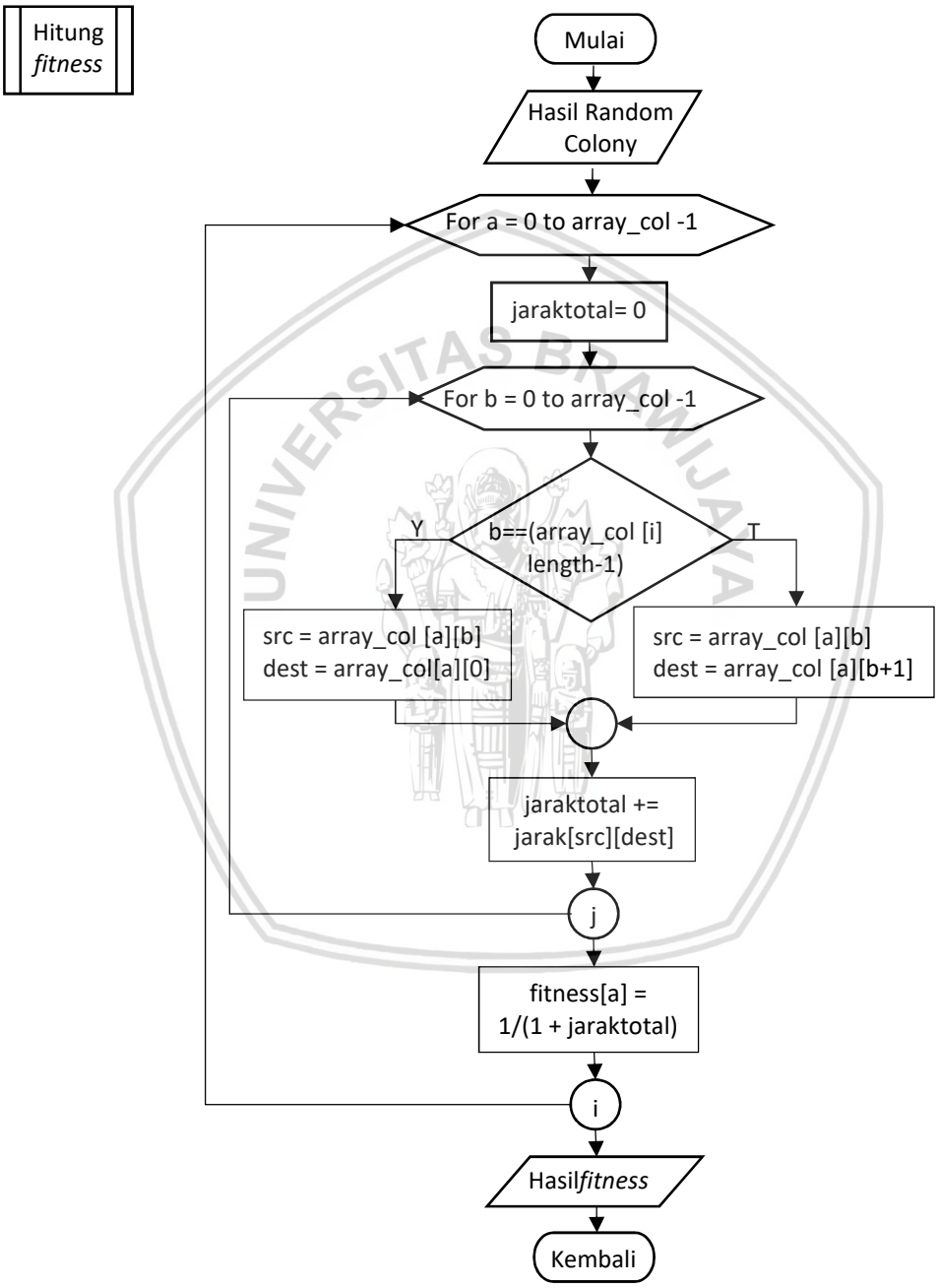
Fase *initial* merupakan proses untuk mendapatkan *initial solution* pada tiap *Employeed Bee*. Pada fase ini dilakukan *generated* individu. Inputan rute yang dimasukkan akan dilakukan pengecekan duplikasi dan kemudian dirandom. Kemudian dihitung nilai *fitness*-nya dengan rumus yang ada pada persamaan 2.3 pada masing-masing *bee*. Diagram alir fungsi *generated* individu dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Alir *Generated* Individu

3.2.2.2 Hitung Nilai *Fitness*

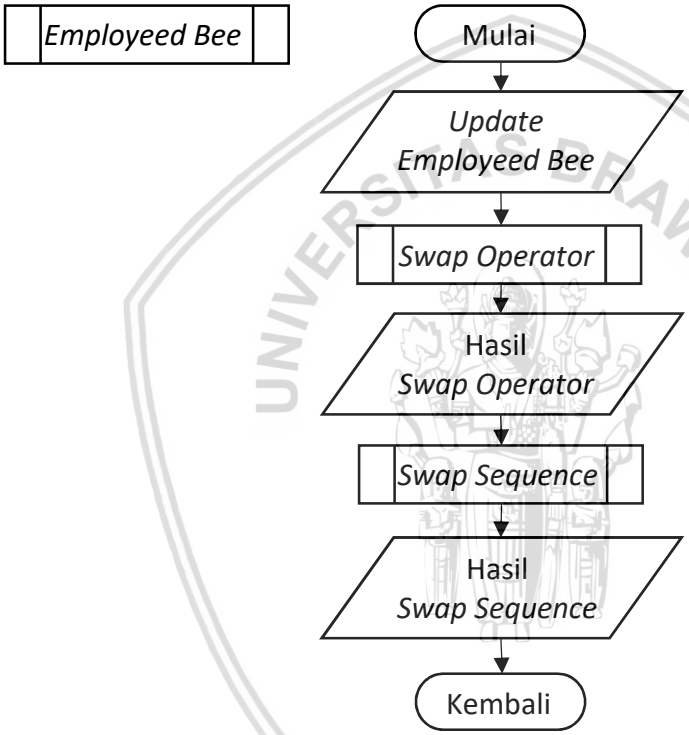
Pada Gambar 3.5 dijelaskan tentang proses perhitungan fitness. Langkah pertama dilakukan inisialisasi terlebih dahulu, kemudian dilakukan looping penghitungan total jarak sepanjang *size problem* sesuai dengan banyaknya *bee colony*. Hingga ditemukan total jarak masing-masing *bee colony* dan kemudian dihitung nilai *fitness*-nya.



Gambar 3.5 Diagram Alir perhitungan Fitness

3.2.2.3 Fase *Employed Bee*

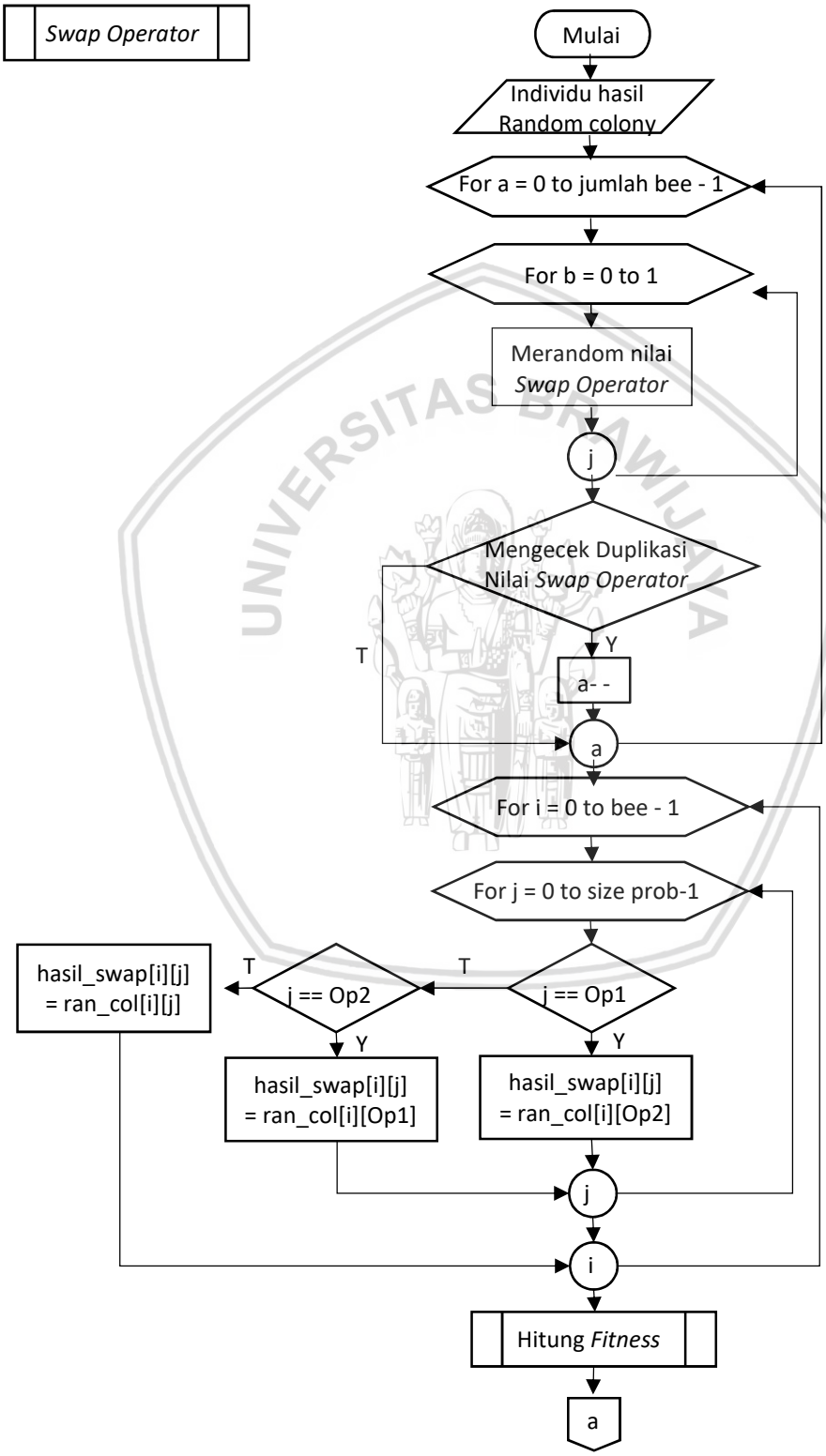
Pada Gambar 3.6 dijelaskan tentang fase Employeed Bee. Pada fase ini digunakan untuk memperbarui solusi dengan metode *neighborhood operator*. Neighborhood operator yang digunakan adalah *Swap Operator* dan *Swap Sequence*. Saat dihasilkan solusi baru dari dari swap operator, akan dihitung nilai fitness-nya dan dibandingkan dengan solusi sebelumnya. Jika solusi baru lebih baik, maka nilai trial = 0 dan menggantikan solusi sebelumnya. Namun jika solusi baru tidak lebih baik dari solusi sebelumnya, maka trial ditambahkan dengan nilai 1. Maka otomatis solusi yang dipilih adalah solusi sebelumnya. Individu terpilih dari swap operator akan dilanjutkan ke swap sequence untuk diacak lagi hingga menemukan solusi yang paling baik.

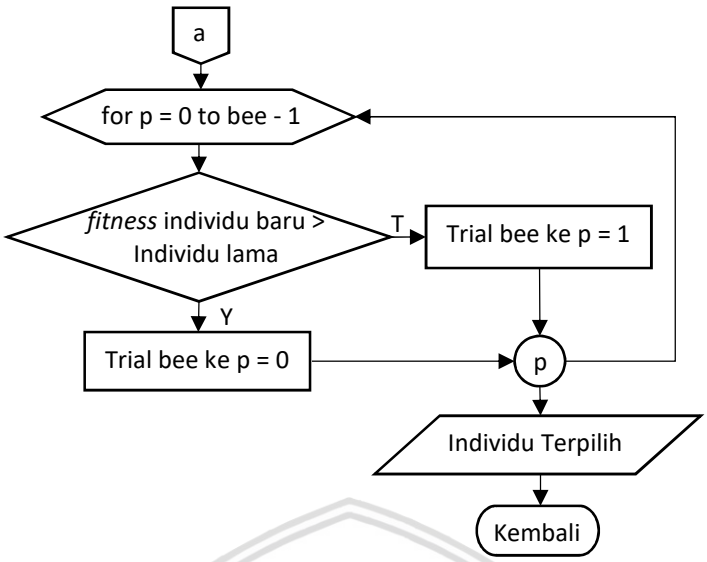


Gambar 3.6 Diagram Alir fase *Employed Bee*

3.2.2.4 Fungsi *Swap Operator* atau *Random Swap*

Pada *swap operator,* ditentukan proses penukaran solusi tempat wisata kuliner, dimana tempat wisata kuliner yang ditukar menggunakan teknik *random* dan bilangan yang dihasilkan tempat wisata kuliner yang satu dengan yang lainnya tidaksama.





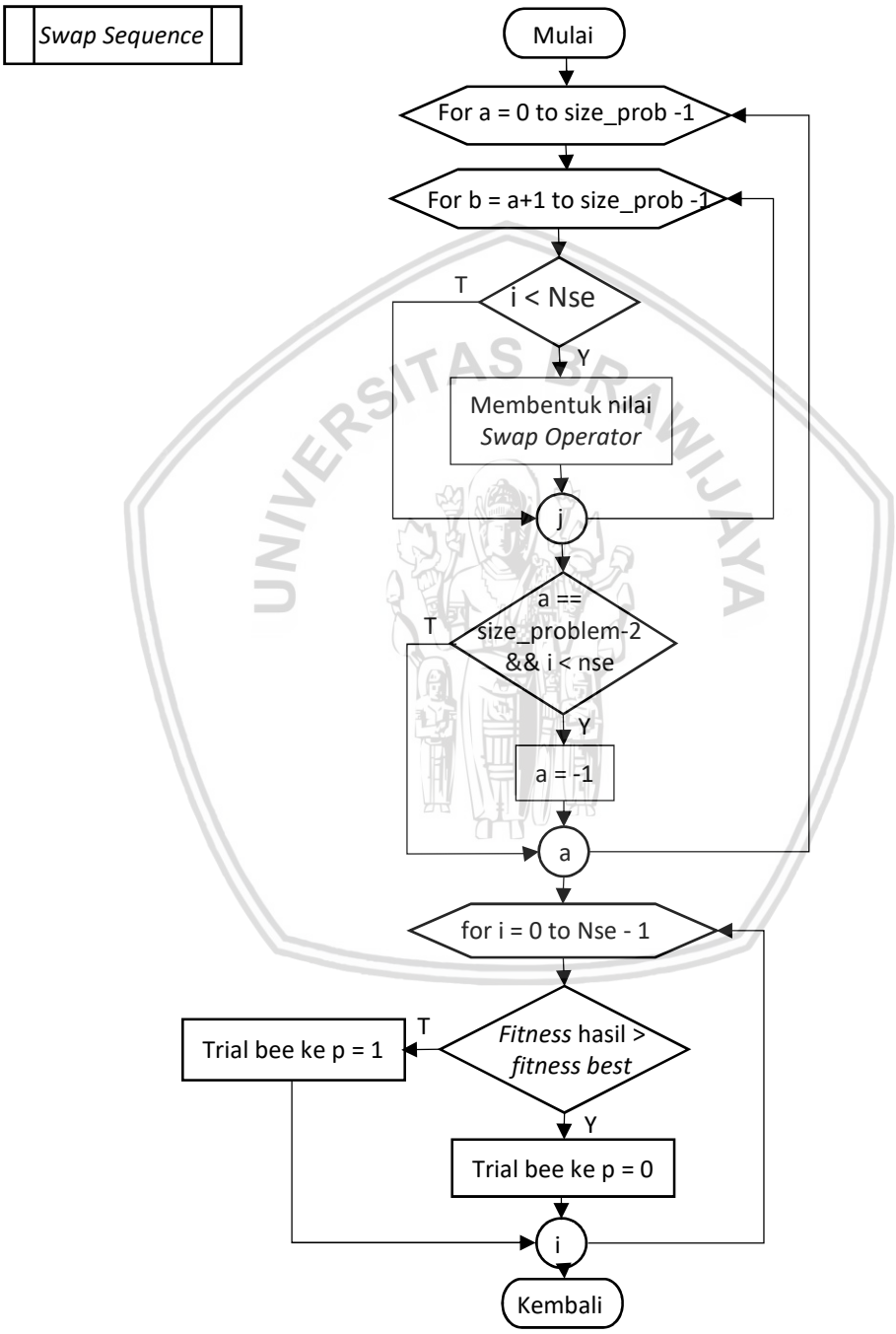
Gambar 3.7 Diagram Alir Fungsi *Swap Operator*

Proses perhitungan pada fungsi swap operator yang ada pada diagram alir Gambar 3.7 dapat dijabarkan sebagai berikut:

1. Melakukan inisialisasi
2. Merandom nilai swap operator sesuai dengan jumlah bee dan nilai dari tempat wisata kuliner yang dimasukkan, kemudian akan dicek apakah nilai randomnya terjadi duplikasi atau tidak, jika terjadi duplikasi maka akan dirandom lagi nilainya.
3. Setelah mendapat nilai random untuk nilai operator, maka nilai tersebut akan dimasukkan ke dalam index i dan j.
4. Perulangan pada index i dan j dilakukan sebanyak jumlah bee dan disesuaikan dengan inputan tempat wisata kuliner.
5. Index x dan y menunjukkan nilai operator 1 dan nilai operator 2
6. Jika nilai j = a, maka yang dimasukkan ke hasil adalah nilai dari index b. Sedangkan jika nilai j = b, maka yang dimasukkan ke hasil adalah nilai dari index a.Sehingga nilai index di a dan nilai index di b akan ditukar.
7. Jika sudah dihasilkan nilai swap operator, makan akan dihitung nilai fitness dari swap operator.
8. Jika nilai fitness dari hasil swap operator lebih besar dari nilai fitness individu sebelumnya, maka trial = 0 dan nilai fitness sebelumnya akan digantikan. Namun jika nilai fitness dari hasil swap operator tidak lebih besar dari nilai fitness individu sebelumnya, maka trial ditambahkan dengan nilai 1. Kemudian akan dipilih individu untuk dipakai ke tahap selanjutnya.

3.2.2.5 Fungsi *Swap Sequence* atau *Random Swap Sequence*

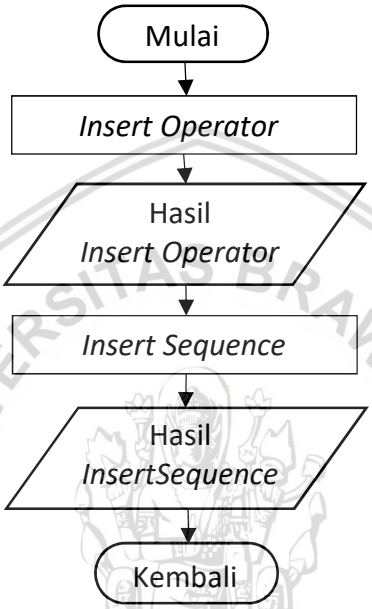
Individu swap sequence diperoleh dari hasil swap operator. Rute tiap individu diacak lagi sebanyak Nse. Setiap diacak akan diperiksa hasil fitness-nya, jika hasil fitness baru lebih besar dari hasil fitness lama, maka trial tidak akan ditambah, dan solusi baru tersebut akan dipilih untuk di swap lagi. Namun jika tidak, maka yang di swap adalah hasil rute sebelumnya dan trial ditambahkan dengan 1.



Gambar 3.8 Diagram Alir Fungsi *Swap Sequence*

3.2.2.6 Fase *Onlooker Bee*

Pada Gambar 3.9 ditunjukkan proses fase *Onlooker Bee.* Pada fase ini digunakan untuk memperbarui solusi menggunakan metode *neighborhood operator*. *Neighborhood operator* yang digunakan yaitu *Insert Operator* dan Insert Sequence.

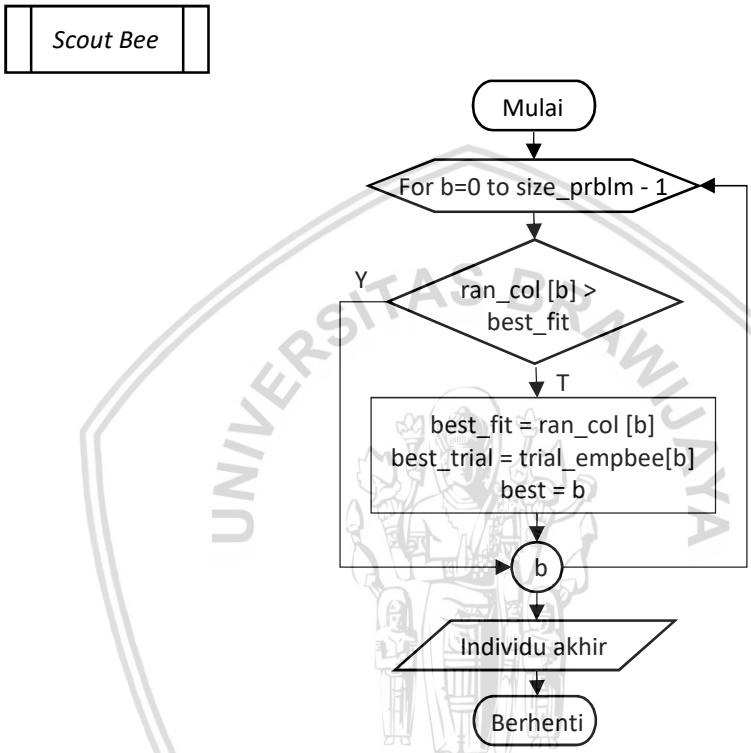


Gambar 3.9 Diagram Alir Fase *Onlooker Bee*

Saat dihasilkan solusi baru dari dari insert operator, akan dihitung nilai fitness-nya dan dibandingkan dengan solusi sebelumnya. Jika solusi baru lebih baik, maka nilai trial = 0 dan menggantikan solusi sebelumnya. Namun jika solusi baru tidak lebih baik dari solusi sebelumnya, maka trial ditambahkan dengan nilai 1 dan solusi yang dipilih adalah solusi sebelumnya. Individu terpilih dari insert operator akan dilanjutkan ke insert sequence untuk diacak lagi, hingga menemukan solusi yang terbaik.

3.2.2.7 Fase *Scout Bee*

Pada fase scout bee, dilakukan perhitungan kualitas dari masing-masing *employeed bee.* Jumlah scout bee disini bersifat dinamis, tergantung pada jumlah *emloyeed bee* yang melebihi *limit.* Apabila limit dari *bee* yang yang melakukan *improvement solution* melebihi *maximum limit* yang ditentukan, maka solusi dari *bee* tersebut akan dihilangkan dan diganti dengan solusi baru menggunakan teknik random, memoerbarui jarak yang dihasilkan, dan menyetel ulang limit kembali menjadi 0.



Gambar 3.10 Diagram Alir Fase *Scout Bee*

Pada fase scout bee ditampilkan individu awal dan individu akhir. Kemudian ditampilkan solusi terbaik dari perhitungan algoritma *bee colony.*

3.2.3 Perhitungan Manual

Travelling salesman problem disini akan diselesaikan menggunakan Algorrima Bee Colony. Pada tahap pertama, yakni inisialisasi seluruh parameter yang dibutuhkan. Parameter yang digunakan adalah file TSP, colony size, limit dan maksiterasi. Tabel 3.1 merupakan keterangan dari file TSP, yaitu keterangan dari nama wisata kuliner yang ada di Malang. Sedangkan Tabel 3.2 merupakan file TSP yang akan digunakan dalam Algoritma *Bee Colony*, yaitu jarak antar tempat wisata kuliner kuliner.

Tabel 3. 1 Nama Wisata Kuliner

No	Wisata Kuliner
0	Alun-alun Malang
1	Toko Oen
2	Mie Gajah Mada Malang
3	Bakso President
4	Inggil Museum Resto
5	Warung Ronde Titoni
6	Rumah Makan Cairo
7	Pecel Kawi
8	Sate Landak Bu Ria
9	Depot Hok Lay
10	Puthu Lanang Celaket

Tabel 3. 2 Jarak antar tempat wisata kuliner

	0	1	2	3	4	5	6	7	8	9	10
0	**0**	0,35	0,35	3,3	1,1	0,55	0,75	1,8	9,5	0,5	2,8
1	0,35	**0**	0,7	2,6	0,8	0,6	0,75	2,1	9,2	0,85	1,6
2	0,35	0,7	**0**	3,4	1,4	0,7	0,55	3,1	9,4	0,55	2,9
3	3,3	2,6	3,4	**0**	2,6	3,6	3,6	3,7	6,8	3,4	1
4	1,1	0,8	1,4	2,6	**0**	1	1,9	2,3	7,8	1	2,4
5	0,55	0,6	0,7	3,6	1	**0**	1,2	2,4	8,8	0,4	2,2
6	0,75	0,75	0,55	3,6	1,9	1,2	**0**	1,9	9,9	1	2,9
7	1,8	2,1	3,1	3,7	2,3	2,4	1,9	**0**	10	2,1	2,5
8	9,5	9,2	9,4	6,8	7,8	8,8	9,9	10	**0**	9,5	9,4
9	0,5	0,85	0,55	3,4	1	0,4	1	2,1	9,5	**0**	3,4
10	2,8	1,6	2,9	1	2,4	2,2	2,9	2,5	9,4	3,4	**0**

Dimisalkan pada sebuah studi kasus terdapat seseorang yang akan memilih 5 tempat wisata yang ingin dikunjungi.

I. Inisialisasi

Tempat wisata kuliner yg ingin dikunjungi :

0	1	2	3	4

Colony Size = 8, Maksiterasi = 2, Size Problem = 4, Limit = 4, Colony Size = Population Size

II. Pembentukan Rute

1. **Menggunakan Metode *Nearest Neighbour***

Pembentukan rute menggunakan perhitungan metode *neighborhood operator* yang ditunjukkan oleh Tabel 3.3.

Tabel 3. 3 Pembentukan rute dengan metode *nearest neighbor*

Bee Ke -i	Rute					Jarak					Total Jarak	*Fitness*
1	0	1	2	3	4	0,35	0,7	3,4	2,6	1,1	8,15	**0,10929**
2	0	3	1	2	4	3,3	2,6	0,7	1,4	1,1	9.1	**0,09901**
3	0	1	4	3	2	0,35	0,8	2,6	3,4	0,35	7,5	**0,11765**
4	0	4	3	1	2	1,1	2,6	2,6	0,7	0,35	7,35	**0,11976**

III. Iterasi = 1

IV. Fase *Employeed Bee*

Perhitungan *Swap Operator*

Pada Tabel 3.4 ditunjukkan contoh perhitungan *swap operator*. Rute pada sisi kiri menunjukkan rute awal yang telah diacak pada saat pembentukan rute dengan metode *nearest neighbor* dan rute pada sisi kanan merupakan rute yang telah diberlakukan *swap operator*.

Tabel 3. 4 Perhitungan *swap operator*

Bee Ke - i	Rute					(SO)	Bee Ke - *i*	Rute				
1	0	1	2	3	4	(1,3)	**1**	0	3	2	1	4
2	0	3	1	2	4	(1,4)	**2**	0	4	1	2	3
3	0	1	4	3	2	(2,4)	**3**	0	1	2	3	4
4	0	4	3	1	2	(3,4)	**4**	0	4	3	2	1

Nilai *fitness* hasil *swap operator*

Tabel 3.5 merupakan nilai *fitness* pada masing-masing individu *Bee* yang dihitung dengan persamaan 2.3.

Tabel 3. 5 Nilai fitness hasil *swap operator*

Jarak					Total Jarak	*Fitness*
3,3	3,4	0,7	0,8	1,1	9,3	**0,09709**
1,1	0,8	0,7	3,4	3,3	9,3	**0,09709**
0,35	0,7	3,4	2,6	1,1	8,15	**0,10929**
1,1	2,6	3,4	0,7	0,35	8,15	**0,10929**

Perbandingan nilai *Fitness*

Setelah mendapatkan nilai *fitness* yang baru, maka dibuatlah perbandingan antara nilai *fitness* individu awal dengan nilai *fitness* yang baru. Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka *trial* pada masing-masing individu diberi nilai 0. Jika tidak, maka *trial* ditambahkan dengan 1. Perbandingan nilai *fitness* dapat dilihat di Tabel 3.6.

Tabel 3. 6 Perbandingan nilai fitness awal dengan fitness hasil *swap operator*

Bee Ke -i	Awal		Hasil (SO)		*Trial*
	Total Jarak	*Fitness*	Total Jarak	*Fitness*	
1	8,15	**0,10929**	9,3	**0,09709**	1
2	9.1	**0,09901**	9,3	**0,09709**	1
3	7,5	**0,11765**	8,15	**0,10929**	1
4	7,35	**0,11976**	8,15	**0,10929**	1

Individu yang dipilih

Pada Tabel 3.7 merupakan individu yang terpilih. Individu yang dipilih merupakan hasil dari perbandingan nilai *fitness* individu awal dengan individu yang baru dengan nilai *fitness* yang lebih besar.

Tabel 3.7 Individu terpilih

Bee Ke -i	Rute				
1	0	1	2	3	4
2	0	3	1	2	4
3	0	1	4	3	2
4	0	4	3	1	2

Number of sequence (*Nse*) = 2**Size Problem* = 2*4 = 8

a. Perhitungan *Swap Sequence*

Setelah dipilihnya individu baru dari hasil *swap operator*, dilakukan perhitungan *swap sequence.* Perhitungan *swap sequence* hampir sama dengan perhitungan *swap operator*. Namun perbedaannya adalah jika *swap sequence* tiap individu dilakukan *swap* sebanyak *Nse*. Kolom *trial* pada tabel menunjukkan jika nilai *fitness* yang baru mengalami peningkatan, maka nilai *trial* akan diganti menjadi 0, jika tidak mengalami peningkatan, maka nilai *trial* akan ditambahkan dengan 1. Pada Tabel 3.8 ditunjukkan perhitungan *swap sequence* pada individu pertama.

Tabel 3.8 Perhitungan *swap sequence* bee ke 1

Bee Ke 1	0	1	2	3	4	*Fitness*	0,10929	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	2	1	3	4	7,35	0,11976	0
(1,3)	0	3	1	2	4	9,1	0,09901	1
(1,4)	0	4	1	3	2	8,25	0,10811	2
(2,3)	0	2	3	1	4	8,25	0,10811	3
(2,4)	0	2	4	3	1	7,3	0,12048	0
(3,4)	0	2	4	1	3	8,45	0,10582	1
(1,2)	0	4	2	3	1	8,85	0,10152	2
(1,3)	0	3	4	2	1	8,35	0,10695	3
Terbaik	0	2	4	3	1	**Fitness**	0,12048	

Pada Tabel 3.9 ditunjukkan perhitungan *swap sequence* pada individu *bee* ke 2.

Tabel 3.9 Perhitungan swap sequence bee ke 2

Bee Ke 2	0	3	1	2	4	*Fitness*	0,09901	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	1	3	2	4	8,85	0,10152	0
(1,3)	0	2	3	1	4	8,25	0,10811	0
(1,4)	0	4	3	1	2	7,35	0,11976	0
(2,3)	0	4	1	3	2	8,25	0,10811	1
(2,4)	0	4	2	1	3	9,1	0,09901	2
(3,4)	0	4	3	2	1	8,15	0,10929	3
(1,2)	0	3	4	1	2	7,75	0,11429	4
(1,3)	0	1	3	4	2	7,3	0,12048	0
Terbaik	0	1	3	4	2	**Fitness**	0,12048	

Pada Tabel 3.10 ditunjukkan perhitungan *swap sequence* pada individu *bee* ke 3.

Tabel 3.10 Perhitungan swap sequence bee ke 3

Bee Ke 3	0	1	4	3	2	*Fitness*	0,11765	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	4	1	3	2	8,25	0,10811	1
(1,3)	0	3	4	1	2	7,75	0,11429	2
(1,4)	0	2	4	3	1	7,3	0,12048	0
(2,3)	0	2	3	4	1	7,5	0,11765	1
(2,4)	0	2	1	3	4	7,35	0,11976	2

Bee Ke 3	0	1	4	3	2	*Fitness*	0,11765	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(3,4)	0	2	4	1	3	8,45	0,10582	3
(1,2)	0	4	2	3	1	8,85	0,10152	4
(1,3)	0	3	4	2	1	8,35	0,10695	5
Terbaik	0	2	4	3	1	**Fitness**	0,12048	

Pada Tabel 3.11 ditunjukkan perhitungan *swap sequence* pada individu *Bee* ke 4.

Tabel 3.11 Perhitungan *swap sequence* bee ke 4

Bee Ke 4	0	4	3	1	2	*Fitness*	0,11976	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	3	4	1	2	7,75	0,11429	1
(1,3)	0	1	3	4	2	7,3	0,12048	0
(1,4)	0	2	3	4	1	7,5	0,11765	1
(2,3)	0	1	4	3	2	7,5	0,11765	2
(2,4)	0	1	2	4	3	8,35	0,10695	3
(3,4)	0	1	3	2	4	8,85	0,10152	4
(1,2)	0	3	1	4	2	8,45	0,10582	5
(1,3)	0	4	3	1	2	7,35	0,11976	6
Terbaik	0	1	3	4	2	**Fitness**	0,12048	

Individu yang dipilih

Pada Tabel 3.12 merupakan Individu terpilih. Individu terpilih merupakan hasil perbandingan nilai *fitness* individu awal dengan *fitness* individu baru dengan nilai *fitness* yang lebih besar.

Tabel 3.12 Individu terpilih

Bee Ke - *i*	**Rute**					***Fitness***	**Trial**
1	0	2	4	3	1	0,12048	3
2	0	1	3	4	2	0,12048	0
3	0	2	4	3	1	0,12048	5
4	0	1	3	4	2	0,12048	6

V. Solusi Baru

Sebelum memasuki fase *onlooker bee*, individu yang terpilih akan diseleksi menggunakan metode *roulette wheel*. Langkah pertama adalah menghitung

probabilitas dari setiap individu. Pada Tabel 3.13 ditunjukkan perhitungan probabilitas.

Tabel 3.13 Perhitungan probabilitas

Bee Ke - *i*	Rute					*Fitness*	*Prob*
1	0	2	4	3	1	0,12048	0,25
2	0	1	3	4	2	0,12048	0,25
3	0	2	4	3	1	0,12048	0,25
4	0	1	3	4	2	0,12048	0,25
						0,48192	

VI. Fase *Onlooker Bee*

Setelah tiap individu dihitung probabilitasnya, kemudian dilakukan perhitungan probabilitas kumulatifnya (*probcum*) untuk mendapatkan *range* pada tiap individu. Pada Tabel 3.14 ditunjukan hasil range.

Tabel 3.14 Hasil range

Bee Ke -*i*	*Prob*	*ProbCum*	*Range*
1	0,25	0,25	1 - 25
2	0,25	0,5	26 - 50
3	0,25	0,75	51 - 75
4	0,25	1	76 - 100

Individu Terpilih (*Roulette Wheel Selection*)

Untuk menentukan individu baru yang akan dipilih, dicari dengan cara *random* nilai sebanyak individu yang ada. Nilai *random* yang muncul akan digunakan untuk menyeleksi individu yang akan dipilih sesuai dengan *range*-nya. Pada Tabel 3.15 dijelaskan nilai *random* yang muncul dan individu yang dipilih.

Tabel 3.15 Seleksi *roulette wheel*

Bee Ke - *i*	Random	Terpilih	Rute					*Fitness*
1	33	**2**	0	1	3	4	2	0,12048
2	93	**4**	0	1	3	4	2	0,12048
3	73	**3**	0	2	4	3	1	0,12048
4	23	**1**	0	2	4	3	1	0,12048

Perhitungan *Insert Operator*

Individu yang sudah dipilih saat fase *onlooker bee* merupakan hasil seleksi *roulette wheel*. Kemudian langkah selanjutnya adalah melakukan perhitungan *insert operator* sesuai dengan persamaan 2.8. Kolom rute pada Tabel 3.16 pada sisi kiri ditunjukkan rute pada individu awal dan kolom rute pada sisi kanan menunjukkan hasil dari *insert operator*.

Tabel 3.16 Perhitungan *insert operator*

Bee Ke -*i*	Rute					(IO)	Bee Ke -*i*	Rute				
1	0	1	3	4	2	(1,4)	**1**	0	2	1	3	4
2	0	1	3	4	2	(2,4)	**2**	0	1	2	3	4
3	0	2	4	3	1	(3,4)	**3**	0	2	4	1	3
4	0	2	4	3	1	(1,3)	**4**	0	3	2	4	1

Nilai *fitness* hasil *insert operator*

Setelah mendapatkan rute yang baru, selanjutnya dilakuan perhitungan nilai *fitness* dengan persamaan 2.3. pada Tabel 3.17 ditunjukkan nilai *fitness* rute yang baru.

Tabel 3.17 Perhitungan *fitness* hasil *insert operator*

Bee Ke -*i*	Rute					Jarak	*Fitness*
1	0	2	1	3	4	7,35	0,11976
2	0	1	2	3	4	8,15	0,10929
3	0	2	4	1	3	8,45	0,10582
4	0	3	2	4	1	9,25	0,09756

Perbandingan Nilai *Fitness*

Pada perbandingan nlai *fitnes,* nilai *fitness* yang baru akan dibandingkan dengan nilai *fitness* yang lama. Jika nilai *fitness* yang baru memiliki nilai yang lebih tinggi, maka *trial* pada masing-masing individu diberi nilai 0. Jika tidak, maka *trial* ditambahkan dengan 1. Perbandingan nilai *fitness* dapat dilihat di Tabel 3.18.

Tabel 3.18 Perbandingan nilai *fitness* hasil *insert operator*

Bee Ke -*i*	Awal		Hasil (IO)		*Trial*
	Total Jarak	*Fitness*	Total Jarak	*Fitness*	
1	7,3	0,12048	7,35	0,11976	1
2	7,3	0,12048	8,15	0,10929	7
3	7,3	0,12048	8,45	0,10582	6

Bee Ke -*i*	Awal		Hasil (IO)		*Trial*
	Total Jarak	*Fitness*	Total Jarak	*Fitness*	
4	7,3	0,12048	9,25	0,09756	4

Individu tepilih

Individu terpilih merupakan hasil perbandingan dari nilai *fitness* individu awal dengan individu baru yang nilai *fitness*-nya lebih besar. Individu terpilih ditunjukkan pada Tabel 3.19.

Tabel 3.19 Individu terpilih

Bee Ke -i	Rute					*Fitness*
1	0	1	3	4	2	0,12048
2	0	1	3	4	2	0,12048
3	0	2	4	3	1	0,12048
4	0	2	4	3	1	0,12048

Perhitungan *Insert Sequence*

Setelah dipilih individu baru yang berasal dari hasil *insert operator*, kemudian dilakukan perhitungan *insert sequence* yang cara perhitungannya hampir sama dengan perhitungan *insert operator*. Namun yang membedakan pada *insert sequence* ialah tiap individu dilakukan *swap* sebanyak *Nse*. Kolom *trial* pada tabel menunjukkan jika nilai *fitness* baru mengalami peningkatan maka nilai *trial* akan dirubah menjadi 0, jika tidak meningkat maka nilai *trial* akan ditambahkan dengan 1. Pada Tabel 3.20 ditunjukkan perhitungan *insert sequence* pada individu *Bee* pertama.

Tabel 3.20 Perhitungan *insert sequence* bee ke 1

Bee Ke 1	0	1	3	4	2	*Fitness*	0,12048	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	3	1	4	2	8,45	0,10582	2
(1,3)	0	4	1	3	2	8,25	0,10811	3
(1,4)	0	2	1	3	4	7,35	0,11976	4
(2,3)	0	1	4	3	2	7,5	0,11765	5
(2,4)	0	1	2	3	4	8,15	0,10929	6
(3,4)	0	1	3	2	4	8,85	0,10152	7
(1,2)	0	3	1	4	2	8,45	0,10582	8
(1,3)	0	4	1	3	2	8,25	0,10811	9

Bee Ke 1	0	1	3	4	2	*Fitness*	0,12048	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
Terbaik	0	1	3	4	2	**Fitness**	0,12048	

Pada Tabel 3.21 ditunjukkan perhitungan *insert sequence* pada individu *Bee* ke 2.

Tabel 3.21 Perhitungan *insert sequence* bee ke 2

Bee Ke 2	0	1	3	4	2	*Fitness*	0,12048	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	3	1	4	2	8,45	0,10582	8
(1,3)	0	4	1	3	2	8,25	0,10811	9
(1,4)	0	2	1	3	4	7,35	0,11976	10
(2,3)	0	1	4	3	2	7,5	0,11765	11
(2,4)	0	1	2	3	4	8,15	0,10929	12
(3,4)	0	1	3	2	4	8,85	0,10152	13
(1,2)	0	3	1	4	2	8,45	0,10582	14
(1,3)	0	4	1	3	2	8,25	0,10811	15
Terbaik	0	1	3	4	2	**Fitness**	0,12048	

Pada Tabel 3.22 ditunjukkan perhitungan *insert sequence* pada individu *Bee* ke 3.

Tabel 3.22 Perhitungan *insert sequence* bee ke 3

Bee Ke 3	0	2	4	3	1	*Fitness*	0,12048	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	4	2	3	1	8,85	0,10152	7
(1,3)	0	3	2	4	1	9,25	0,09756	8
(1,4)	0	1	2	4	3	8,35	0,10695	9
(2,3)	0	2	3	4	1	7,5	0,11765	10
(2,4)	0	2	1	4	3	7,75	0,11429	11
(3,4)	0	2	4	1	3	8,45	0,10582	12
(1,2)	0	4	2	3	1	8,85	0,10152	13
(1,3)	0	3	2	4	1	9,25	0,09756	14
Terbaik	0	2	4	3	1	**Fitness**	0,12048	

Pada Tabel 3.23 ditunjukkan perhitungan *insert sequence* pada individu *Bee* ke 4.

Tabel 3.23 Perhitungan insert sequence bee ke 4

Bee Ke 4	0	2	4	3	1	*Fitness*	0,12048	*Trial*
(SO)	**Rute**					**Jarak**	***Fitness***	
(1,2)	0	4	2	3	1	8,85	0,10152	5
(1,3)	0	3	2	4	1	9,25	0,09756	6
(1,4)	0	1	2	4	3	8,35	0,10695	7
(2,3)	0	2	3	4	1	7,5	0,11765	8
(2,4)	0	2	1	4	3	7,75	0,11429	9
(3,4)	0	2	4	1	3	8,45	0,10582	10
(1,2)	0	4	2	3	1	8,85	0,10152	11
(1,3)	0	3	2	4	1	9,25	0,09756	12
Terbaik	0	2	4	3	1	**Fitness**	0,12048	

VII. Fase *Scout Bee*

Individu Awal

Pada Tabel 3.24 ditunjukkan rute dan nilai *fitness* pada setiap individu awal. Pada fase *scout bee* ditampilkan individu awal yang pertama dimasukkan.

Tabel 3.24 Individu awal

Bee Ke -*i*	**Rute**					***Fitness***
1	0	1	2	3	4	0,10929
2	0	3	1	2	4	0,09901
3	0	1	4	3	2	0,11765
4	0	4	3	1	2	0,11976

Individu Terpilih

Pada Tabel 3.25 ditunjukkan rute dan nilai *fitness* pada tiap individu terpilih dari hasil perhitungan *bee colony*.

Tabel 3.25 Individu terpilih

Bee Ke -*i*	**Rute**					***Fitness***
1	0	1	3	4	2	0,12048
2	0	1	3	4	2	0,12048
3	0	2	4	3	1	0,12048
4	0	2	4	3	1	0,12048

Solusi terbaik

Solusi terbaik merupakan hasil perhitungan yang menghasilkan nilai *fitness* tertinggi. Jika nilai *fitness* tertinggi berjumlah lebih dari satu, maka akan dilakukan pemeriksaan *trial*. Salah satu dari hasil nilai *fitness* tertinggi dan memiliki nilai *trial* terendah, akan dipilih menjadi solusi yang terbaik. Tabel 3.26 merupakan solusi terbaik.

Tabel 3.26 Solusi Terbaik

Bee Ke – i	**Rute**					***Fitness***
1	0	1	3	4	2	0,12048

BAB 4 IMPLEMENTASI

Pada bab implementasi akan dibahas mengenai implementasi algoritma *bee colony* untuk optimasi rute tempat wisata kuliner di Kota Malang berdasarkan perancangan yang telah dijelaskan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Adapun lingkungan implementasinya yaitu lingkungan perangkat keras (*hardware*) dan lingkungan perangkat lunak (*software*). Selanjutnya implementasi program.

4.1.1 Lingkungan Perangkat Keras

Untuk spesifikasi perangkat keras yang digunakan adalah:

1. Prosesor *Intel® Core™* i5 2.50*GHz*
2. RAM 4.00 GB
3. *Hardisk* 500 GB
4. Monitor 14”
5. *Keyboard*

4.1.2 Lingkungan Perangkat Lunak

Untuk spesifikasi perangkat lunak yang digunakan adalah:

1. *Operating System Windows* 10 64 *bit*
2. *Microsoft office* 2013
3. NetBeans IDE 7.4
4. Bahasa pemrograman *java*

4.2 Implementasi Program

Pada sub-bab implementasi program akan dibahas mengenai implementasi berdasarkan perancangan sistem pada bab sebelumnya, yaitu implementasi algoritma *bee colony* untuk optimasi rute tempat wisata kuliner di Malang. Implementasi pembuatan program aplikasi dibuat dengan menggunakan bahasa pemrograman *java*. Program aplikasi yang dibuat mempunyai 4 proses utama yaitu proses fase *initial*, proses fase *employeed bee*, proses fase *onlooker bee*, dan proses fase *scout bee*.

4.2.1 Proses Fase Initial

Pada fase *initial* terdapat inputan rute yang dimasukkan akan dilakukan pengecekan duplikasi yaitu apakah input yang dimasukkan memiliki kesamaan atau tidak kemudian kedudukannnya dirandom. Setelah itu dihitung jarak antar

tempat wisata dan ditotal seluruh jaraknya untuk mengetahui nilai fitness-nya pada masing-masing *bee.*

4.2.1.1 Proses Generated Individu

Proses generated individu digunakan untuk memasukkan nilai yang berupa tempat wisata kuliner yang ingin dikunjungi. Kemudian inputan tersebut digunakan untuk membentuk individu yang baru.

```
1  public static void col_size() {
2          int rute_input;
3
4          System.out.print("Masukkan Jumlah Rute :");
5          size_problem = input.nextInt();
6          int jumlah_size_problem;
7          array_rute = new int[size_problem];
8
9          for (jumlah_size_problem = 0; jumlah_size_problem
10  < size_problem; jumlah_size_problem++) {
11             System.out.print("Masukkan rute " +
12  (jumlah_size_problem + 1 + ": "));
13             rute_input = input.nextInt();
14             array_rute[jumlah_size_problem] = rute_input;
15             if (array_rute[jumlah_size_problem] < 0 ||
16  array_rute[jumlah_size_problem] > 14) {
17                 jumlah_size_problem--;
18             }
19         }
20     }
```

Source Code 4. 1 Proses Input Data

Penjelasan *source code* 4.1 adalah :

1. Baris 9 - 10 menunjukkan bahwa inputan yang boleh dimasukkan adalah hanya angka 0 – 10

4.2.1.2 Proses *Check Duplicate*

Pada *check duplicate* dilakukan pengecekan pada inputan yang dimasukkan agar tidak terjadi duplikasi.

```
1  public static String checkDuplicate(int[] array_cek) {
```

```
2
3           ArrayList<Integer> listRute = new
4           ArrayList<Integer>();
5           for (int j = 0; j < array_cek.length; j++) {
6               listRute.add(j, array_cek[j]);
7           }
8           HashSet setRute = new HashSet(listRute);
9           if(setRute.size() != listRute.size()){
10              colony_size();
11              return checkDuplicate(array_rute);
12          }else{
13              return "ok";
14          }
15      }
```

Source Code 4. 2 Proses *Check Duplicate*

Penjelasan *source code* 4.2 adalah :

1. Baris 10 – 11 berarti jika inputan rute yang tidak sama dengan list rute, maka akan diulangi fungsi colony_size kembali

4.2.1.3 Proses *Random Colony*

Pada *random colony* ditampilkan pengacakan rute sesuai dengan jumlah koloni yang telah diinput.

```
1   public static int[][] random_colony(int[] array_rute) {
2           Random col = new Random();
3           System.out.print("Masukkan jumlah koloni : ");
4           bee = input.nextInt();
5           int[][] acak_colony = new
6           int[bee][array_rute.length];
7
8           for (int y = 0; y < bee; y++) {
9               int[] temp_rute = array_rute.clone();
10              ShuffleArray(temp_rute);
11              acak_colony[y] = temp_rute.clone();
12              for (int x = 0; x < (array_rute.length); x++) {
13                  System.out.print(acak_colony[y][x] +" ");
14              }
15              System.out.println();
```

```
16          }
17          return acak_colony;
18      }
```

Source Code 4. 3 Proses *Random Colony*

Penjelasan *source code* 4.3 adalah :

1. Baris 9 merupakan inputan rute dimasukkan ke dalam temp_rute
2. Baris 10 merupakan nilai pada temp_rute diproses dengan fungsi ShuffleArray
3. Baris 12 -14 merupakan menampilkan hasil dari acak_colony

4.2.1.4 Proses *Shuffle Array*

Pada *Shuffle Array* dilakukan pengacakan rute sesuai dengan jumlah koloni yang telah diinput.

```
1   private static void ShuffleArray(int[] ar)
2       {
3           Random rnd = new Random();
4           for (int i = ar.length - 1; i > 0; i--)
5           {
6             int index = rnd.nextInt(i + 1);
7             int a = ar[index];
8             ar[index] = ar[i];
9             ar[i] = a;
10          }
11      }
```

Source Code 4. 4 Proses *Shuffle Aray*

Penjelasan *source code* 4.4 adalah :

1. Baris 4 – 10 dilakukan pengacakan isi *array* dengan *swap* satu persatu.

4.2.1.5 Proses Menghitung *Fitness*

Menghitung *fitness* dilakukan perhitungan total jarak hingga nilai *fitness*. Total jarak diset bernilai 0. Kemudian jarak dihitung dari index terakhir hingga index 0.

```
1   public static double[] hitung_fitness(int[][] array_colony)
2       {
```

```
3           double[] fitness = new double[array_colony.length];
4
5           for(int i=0;i<array_colony.length;i++)
6           {
7               double total_jarak = 0;
8               for(int j=0;j<array_colony[i].length;j++)
9               //panjang untuk size problem
10              {
11                  int src, dest;
12                  if(j==(array_colony[i].length-1))
13                  {
14                     src = array_colony[i][j];
15                     dest = array_colony[i][0];
16                  }
17                  else
18                  {
19                      src = array_colony[i][j];
20                     dest = array_colony[i][j+1];
21                  }
22                  total_jarak=total_jarak+jarak[src][dest];
23              }
24              fitness[i] = 1/(1 + total_jarak);
25          }
26          return fitness;
27      }
```

Source Code 4. 5 Proses Menghitung *Fitness*

Penjelasan *source code* 4.5 adalah :

1. Baris 7 menjelaskan untuk awal total jarak diset menjadi 0
2. Baris 12 – 16 jika pada perhitungan jarak sudah pada index terakhir, maka jarak dihitung dari index terakhir hingga index 0
3. Baris 19 – 20 menjelaskan perhitungan jarak pada index ke j hingga index ke j+1
4. Baris 22 menjelaskan perhitungan total jarak
5. Baris 24 menjelaskan perhitungaan nilai fitness yaitu 1 dibagi dengan 1 ditambah total jarak

4.2.2 Proses Fase *Employeed Bee*

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

4.2.2.1 Proses *Swap Operator*

Pada proses menghitung *swap operator* dilakukan *swap* pada masing-masing *bee colony*.

```
1   private static int[][] swapOperator(int[][] random_colony,
2   double[] random_colony_f)
3       {
4           int[][] swap_operator = new int[bee][2];
5           // simpan nilai operator
6           int[][] hasil_swap = new int[bee][size_problem];
7           double[] swap_fitness;
8           Random random = new Random();
9
10          for(int i = 0;i < bee;i++)
11          {
12              for(int j = 0;j < 2;j++)
13              {
14          swap_operator[i][j]=random.nextInt(size_problem-1);
15              }
16              if(duplicates(swap_operator[i]))
17              {
18                  i--;
19              }
20          }
21          System.out.println("Hasil swap:");
22          for(int p = 0;p < bee;p++)
23          {
24              for(int q = 0;q < size_problem; q++)
25              {
26                  int Op1 = swap_operator[p][0];
27                  int Op2 = swap_operator[p][1];
28                  if(q == Op1)
29                  {
30                     hasil_swap[p][q]=random_colony[p][Op2];
31                  }
```

```
                else if(q == Op2)
                {
                   hasil_swap[p][q]=random_colony[p][Op1];
                }
                else
                {
                   hasil_swap[p][q]=random_colony[p][q];
                }
                System.out.print(hasil_swap[p][q] + " ");
            }
            System.out.println();
        }
        swap_fitness = hitung_fitness(hasil_swap);
        for(int r = 0;r < bee;r++)
        {
            System.out.println(swap_fitness[r]);
        }
        trial_empbee = new int[bee];
        System.out.println("Individu terpilih:");
        for(int s = 0;s < bee;s++)
        {
           if(swap_fitness[s] > random_colony_f[s])
         {
                trial_empbee[s] = 0;
         }
            else
         {
            trial_empbee[s] = 1;
            hasil_swap[s] = random_colony[s];
         }
            for(int t = 0;t < size_problem;t++)
         {
            System.out.print(hasil_swap[s][t] + " ");
         }
         System.out.print("trial" + trial_empbee[s] + " ");
         System.out.println();
        }
        return hasil_swap;
```

```
70      }
```

Source Code 4. 6 Proses *Swap Operator*

Penjelasan *source code* 4.6 adalah :

1. Baris 10 - 20 menentukan nilai *random* yang akan digunakan untuk nilai *operator* dan tidak boleh ada duplikasi
2. Baris 26 merupakan nilai pada operator 1
3. Baris 27 merupakan nilai pada operator 2
4. Baris 22 – 42 menampilkan hasil *swap operator*
5. Baris 28 – 31 menjelaskan jika q = nilai operator 1, maka hasil pada operator 2 dimasukkan ke index q
6. Baris 32 – 35 menjelaskan jika q = nilai operator 2, maka hasil pada operator 1 dimasukkan ke index q
7. Baris 45 merupakan perhitungaan nilai *fitness* dari hasil *swap operator*
8. Baris 51 – 69 merupakan perhitungan *trial*. Jika nilai *fitness* hasil *swap operator* lebih besar dari nilai *fitness* dari individu sebelumnya,maka nilai *trial* menjadi 0, jika lebih kecil, nilai *trial* menjadi 1

4.2.2.2 Proses *Swap Sequences*

Pada proses menghitung swap sequences dilakukan swap pada masing-masing bee colony sebanyak nilai Nse.

```
1   public static int[] swapSequence(int[] bee_swap,
2   double fitness_awal, int index)
3       {
4           int[] hasil = new int[size_problem];
5           hasil = bee_swap.clone();
6           int[] best_bee = new int[size_problem];
7           best_bee = bee_swap.clone();
8           double fitness_hasil = fitness_awal;
9           double fitness_best = fitness_awal;
10          int nse = size_prob * 2;
11          int[][] swap_operator = new int[nse][2];
12          int p = 0;
13
14          for(int i = 0;i < size_problem;i++)
```

```
15          {
16            for(int j = i+1;j < size_problem;j++)
17            {
18               if(p < nse)
19               {
20                       swap_operator[p][0] = i;
21                       swap_operator[p][1] = j;
22                       p++;
23               }
24            }
25              if(i == size_problem-2 && p < nse)
26                  i = -1;
27          }
28          System.out.println("Bee ke-"+(index+1));
29          for(p = 0;p < nse;p++)
30          {
31              int a = swap_operator[p][0];
32              int b = swap_operator[p][1];
33              hasil = best_bee.clone();
34              int temp = hasil[a];
35              hasil[a] = hasil[b];
36              hasil[b] = temp;
37              fitness_hasil = hitung_fitness (hasil);
38              if(fitness_hasil > fitness_best)
39              {
40                  trial_empbee[index] = 0;
41                  best_bee = hasil;
42                  fitness_best = fitness_hasil;
43              }
44              else
45              {
46                  trial_empbee[index]++;
47              }
48              System.out.print(a+","+b+" ");
49              for(int k = 0;k<size_problem;k++)
50              {
51                  System.out.print(best_bee[k] + " ");
52              }
```

```
53              System.out.print("hasil swap -> ");
54              for(int l = 0;l<size_problem;l++)
55              {
56                  System.out.print(hasil[l] + " ");
57              }
58  System.out.println(fitness_best+"trial"+trial_empbee[index]);
59          }
60          return best_bee;
61      }
```

Source Code 4. 7 Proses *Swap Sequences*

Penjelasan *source code* 4.7 adalah :

1. Baris 10 menjelaskan perhitungan nilai Nse
2. Baris 16 - 27 menjelaskan nilai operator 1 dan 2 sebanyak nilai Nse
3. Baris 34 menjelaskan bahwa hasil dari *best bee* dimasukkan ke hasil
4. Baris 35 - 37 menjelaskan nilai pada index a disimpan ke dalam temporary, nilai pada index b dimasukkan ke dalam index a, kemudian nilai pada temporary dimasukkan ke dalam index b.
5. Baris 38 - 48 menjelaskan jika nilai *fitness* hasil *swap* lebih baik dari nilai *fitness* sebelumnya, maka *trial* menjadi 0. Jika tidak, *trial* ditambahkan dengan 1.

4.2.3 Proses Seleksi *Roulette Wheel*

Individu diseleksi dengan metode *roulette wheel* Sebelum memasuki fase *onlooker bee*.

```
1   public static int[][] rouletteWheel(int[][] bee_colony,
2   double[] bee_colony_f)
3       {
4           double[] prob = new double[bee];
5           double[] prob_cum = new double[bee];
6           double[] random = new double[bee];
7           int[][] hasil = bee_colony.clone();
8           double total_fitness = 0;
9
10          for(int i = 0;i < bee;i++)
11          {
12              total_fitness += bee_colony_f[i];
13          }
```

```
14          System.out.println("Menghitung probabilitas:");
15          for(int j = 0;j < bee;j++)
16          {
17              prob[j] = bee_colony_f[j]/total_fitness;
18              if(j == 0)
19              {
20               prob_cum[j] = prob[j];
21               System.out.println("Prob " + prob[j] + "
22  kumulatif " + prob_cum[j] + " range 0-" + prob_cum[j]);
23              }
24              else
25              {
26                prob_cum[j] = prob_cum[j-1] + prob[j];
27                System.out.println("Prob " + prob[j] + "
28  kumulatif " + prob_cum[j] + " range " + prob_cum[j-1] + "-
29  " + prob_cum[j]);
30              }
31          }
32          int[] temp_trial = trial_empbee.clone();
33          System.out.println("Individu terpilih:");
34          for(int k = 0;k < bee;k++)
35          {
36              random[k] = Math.random();
37              for(int l = 0;l < bee;l++)
38              {
39                  if(random[k] <= prob_cum[l])
40                  {
41                    hasil[k] = bee_colony[l];
42                    trial_empbee[k] = temp_trial[l];
43                    System.out.print(random[k] + " ");
44                    for(int m = 0;m < size_prob;m++)
45                    {
46                     System.out.print(bee_colony[l][m]  +  "
47  ");
48                    }
49  System.out.print(hitung_fitness (hasil[k]));
50                    System.out.println();
51                    break;
                  }
```

```
52              }
53          }
54          return hasil;
55      }
```

Source Code 4. 8 Proses Seleksi *Roulette Wheel*

Penjelasan *source code* 4.8 adalah :

1. Baris 10 - 13 menghitung nilai total *fitness*
2. Baris 17 menghitung nilai *probabilitas*
3. Baris 34 - 53 seleksi *roulette wheel*

4.2.4 Proses Fase *Onlooker Bee*

Pada fase *onlooker bee* terdapat perhitungan *insert operator* dan *insert sequences.*

4.2.4.1 Proses *Insert Operator*

Pada *Insert Operator* dilakukan *insert* pada masing-masing *bee colony.*

```
1   public static int[][] insertOperator(int[][] bee_colony,
2   double[] bee_colony_f)
3       {
4           int[][] insert_operator = new int[bee][2];
5           int[][] hasil_insert = new int[bee][size_problem];
6           double[] insert_fitness;
7           Random random = new Random();
8
9           for(int i = 0;i < bee;i++)
10          {
11              for(int j = 0;j < 2;j++)
12              {
13  insert_operator[i][j]=random.nextInt(size_problem - 1);
14              }
15              if(duplicates(insert_operator[i]) ||
16  insert_operator[i][0]>insert_operator[i][1])
17              {
18                  i--;
19              }
20          }
```

```
21          System.out.println("Hasil insert:");
22          for(int p = 0;p < bee;p++)
23          {
24              for(int q = 0;q < size_problem; q++)
25              {
26                  int a = insert_operator[p][0];
27                  int b = insert_operator[p][1];
28                  if(q < a || q > b)
29                      hasil_insert[p][q] = bee_colony[p][q];
30                  else if(q == a)
31                      hasil_insert[p][q] = bee_colony[p][b];
32                  else
33                      hasil_insert[p][q]=bee_colony[p][q-1];
34                  System.out.print(hasil_insert[p][q] + " ");
35              }
36              System.out.println();
37          }
38          insert_fitness = hitung_fitness(hasil_insert);
39          for(int r = 0;r < bee;r++)
40          {
41              System.out.println(insert_fitness[r]);
42          }
43          System.out.println("Individu terpilih:");
44          for(int s = 0;s < bee;s++)
45          {
46              if(insert_fitness[s] > bee_colony_f[s])
47              {
48                  trial_empbee[s] = 0;
49              }
50              else
51              {
52                  trial_empbee[s]++;
53                  hasil_insert[s] = bee_colony[s];
54              }
55              for(int t = 0;t < size_problem;t++)
56              {
57                  System.out.print(hasil_insert[s][t] + " ");
58              }
```

```
59              System.out.print("trial" + trial_empbee[s] + " ");
60              System.out.println();
61             }
62             return hasil_insert;
63         }
```

Source Code 4.9 Proses *Insert Operator*

Penjelasan *source code* 4.9 adalah :

1. Baris 9 - 20 menentukan nilai *random* yang akan digunakan untuk nilai *operator* dan tidak boleh terjadi nilai yang sama
2. Baris 22 - 35 menjelaskan perhitungan *insert operator*
3. Baris 29 nilai pada operator 2
4. Baris 46 - 61 menjelaskan perhitungan *trial*. Jika nilai *fitness* hasil *insert operator* lebih besar dari nilai *fitness* dari individu sebelumnya,maka nilai *trial* menjadi 0, jika lebih kecil, nilai *trial* menjadi 1

4.2.4.2 Proses *Insert Sequences*

Pada *insert sequences* dilakukan *insert* pada masing-masing *bee colony* sebanyak nilai Nse.

```
1   public static int[] insertSequence(int[] bee_awal,
2   double fitness_awal, int index)
3       {
4           int[] hasil = new int[size_problem];
5           hasil = bee_awal.clone();
6           int[] best_bee = new int[size_problem];
7           best_bee = bee_awal.clone();
8           double fitness_hasil = fitness_awal;
9           double fitness_best = fitness_awal;
10          int nse = size_problem * 2;
11          int[][] insert_operator = new int[nse][2];
12          int p = 0;
13
14          for(int i = 0;i < size_problem;i++)
15          {
16              for(int j = i+1;j < size_problem;j++)
17              {
18                  if(p < nse)
19                  {
```

```
                    insert_operator[p][0] = i;
                    insert_operator[p][1] = j;
                    p++;
                }
            }
            if(i == size_problem-2 && p < nse)
                i = -1;
        }
        System.out.println("Bee ke-"+(index+1));
        for(p = 0;p < nse;p++)
        {
            int a = insert_operator[p][0];
            int b = insert_operator[p][1];
            int[] temp = best_bee.clone();
            for(int q = 0;q < size_problem; q++)
            {
                if(q < a || q > b) hasil[q] = temp[q];
                else if(q == a) hasil[q] = temp[b];
                else hasil[q] = temp[q-1];
            }
            fitness_hasil = hitung_fitness(hasil);
            if(fitness_hasil > fitness_best)
            {
                trial_empbee[index] = 0;
                best_bee = hasil;
                fitness_best = fitness_hasil;
            }
            else
            {
                trial_empbee[index]++;
            }
            System.out.print(a+","+b+" ");
            for(int k = 0;k<size_problem;k++)
            {
                System.out.print(best_bee[k] + " ");
            }
            System.out.print("hasil insert -> ");
            for(int l = 0;l<size_problem;l++)
```

```
58              {
59                  System.out.print(hasil[l] + " ");
60              }
61              System.out.println
62  (fitness_best+"trial"+trial_empbee[index]);
63          }
64          return best_bee;
65      }
```

Source Code 4. 10 Proses *Insert Sequences*

Penjelasan *source code* 4.10 adalah :

1. Baris 10 perhitungan nilai Nse
2. Baris 16 - 27 menjelaskan bahwa nilai operator 1 dan 2 sebanyak nilai Nse
3. Baris 33 menjelaskan bahwa hasil dari *best bee* dimasukkan ke *temp*
4. Baris 34 - 39 menjelaskan jika nilai q kurang dari a atau lebih dari b maka nilai pada *temporary* index ke q disimpan ke dalam hasil index ke q, jika nilai q = a, maka nilai pada *temporary* index ke b dimasukkan ke dalam hasil index ke q. Kemudian nilai pada index ke q di -1 dan dimasukkan ke dalam hasil index q. Sehingga terjadi pergeseran tempat
5. Baris 41 - 50 menjelaskan jika nilai *fitness* hasil *insert* lebih baik dari nilai *fitness* sebelumnya, maka *trial* menjadi 0. Jika tidak lebih baik, *trial* ditambah dengan 1

4.2.5 Proses *Scout Bee*

Pada *scout bee* ditampilkan individu awal dan individu yang terpilih. Kemudian solusi dengan nilai *fitness* tertinggi dan nilai *trial* terendah akan menjadi solusi terbaik.

```
1           int j,k;
2           int index_best = 0;
3           double best_fitness = 0;
4           int best_trial = trial_empbee[0];
5           System.out.println();
6           System.out.println("Individu awal");
7           for(j = 0;j < bee;j++)
8               {
9                   for(k = 0;k < size_problem;k++)
10                  {
11                  System.out.print(colony_awal[j][k] + " ");
```

```
12                  }
13                  System.out.println(colony_awal_f[j]);
14              }
15              System.out.println();
16              System.out.println("Individu terpilih");
17              for(j = 0;j < bee;j++)
18              {
19                  if(random_colony_f[j] > best_fitness)
20                  {
21                      best_fitness = random_colony_f[j];
22                      best_trial = trial_empbee[j];
23                      index_best = j;
24                  }
25                 else if(random_colony_f[j]==best_fitness &&
26                      trial_empbee[j] < best_trial)
27                  {
28                      best_trial = trial_empbee[j];
29                      index_best = j;
30                  }
31                  for(k = 0;k < size_problem;k++)
32                  {
33                  System.out.print(random_colony[j][k]+" ");
34                  }
35                  System.out.println(random_colony_f[j]);
36              }
37              System.out.println();
38              System.out.println("Solusi terbaik");
39              for(j = 0;j < size_problem;j++)
40              {
41        System.out.print(random_colony[index_best][j] + " ");
42              }
43        System.out.println(random_colony_f[index_best]);
44          }
45      }
```

Source Code 4. 11 Proses *Scout Bee*

Penjelasan *source code* 4.11 adalah :

1. Baris 6 - 14 menampilkan individu awal
2. Baris 16 - 36 menampilkan individu terpilih

3. Baris 19 - 24 dipilihnya individu terbaik dengan nilai *fitness* yang tertinggi
4. Baris 25 – 30 menjelaskan bahwa jika terdapat lebih dari satu solusi yang dihasilkan dengan nilai *fitness* yang tinggi, maka akan dicari diantara solusi tersebut yang memiliki nilai *trial* terendah

BAB 5 PENGUJIAN DAN ANALISIS

Pada bab pengujian dan analisis akan dibahas mengenai analisis pengujian yang meliputi pengaruh jumlah *bee colony* terhadap hasil solusi terbaik, pengaruh banyaknya iterasi terhadap hasil solusi terbaik, dan pengaruh banyaknya s*ize problem* terhadap hasil solusi terbaik.

5.1 Pengujian

Pengujian ini dilakukan sebanyak 3 model pengujian yaitu :

1. Pengujian untuk mengetahui pengaruh jumlah *bee colony* terhadap hasil solusi terbaik.
2. Pengujian untuk mengetahui pengaruh banyaknya iterasi terhadap hasil solusi terbaik.
3. Pengujian untuk mengetahui pengaruh banyaknya s*ize problem* terhadap hasil solusi terbaik.

5.2 Hasil Pengujian

Terdapat 3 hasil pengujian yaitu pengaruh jumlah *bee colony* terhadap hasil solusi terbaik, pengaruh banyaknya iterasi terhadap hasil solusi terbaik, dan pengaruh banyaknya *size problem* terhadap hasil solusi terbaik.

5.2.1 Pengujian Pengaruh Jumlah *Bee Colony* Terhadap Hasil Solusi Terbaik

Pengujian pengaruh jumlah *bee colony* terhadap hasil solusi terbaik dilakukan untuk melihat tingkat konvergensi. Jumlah *size problem* adalah 8, nilai yang digunakan adalah (0, 2, 3, 5, 6, 8, 9, 10), dengan menggunakan *bee colony* (5, 10, 15, 20, 25, 30, 35, 40, 45, 50). Iterasi yang digunakan sebanyak 50 kali. Pada pengujian ini dilakukan uji coba sebanyak 5 kali. Pengujiannya ditunjukkan pada tabel 5.1

Tabel 5. 1 Hasil Pengujian Pengaruh Jumlah *Bee Colony* Terhadap Hasil Solusi

Jumlah *Bee Colony*	Hasil Percobaan Ke - i					Rata – Rata *Fitness*
	1	2	3	4	5	
5	0,04081	0,04338	0,04	0,04396	0,04367	0,04236
10	0,04367	0,04425	0,04367	0,04	0,04228	0,04277
15	0,03883	0,04008	0,04405	0,04008	0,04405	0,04142
20	0,04008	0,04032	0,04405	0,04405	0,04396	0,04249
25	0,04405	0,04405	0,04405	0,04425	0,04032	0,04334
30	0,04008	0,04386	0,04405	0,04008	0,04008	0,04263
35	0,04032	0,04032	0,04405	0,04405	0,04032	0,04281
40	0,04405	0,04396	0,04405	0,04405	0,04405	0,04403
45	0,04008	0,04032	0,04405	0,04032	0,04032	0,04102
50	0,04008	0,04032	0,04396	0,04405	0,04008	0,0417

Pada Tabel 5.1 dapat dilihat bahwa hasil dari rata-rata nilai *fitness* yang paling baik adalah pada jumlah *bee colony* sebanyak 40 dimana nilai rata-rata *fitness*-nya sebesar 0,04403.

5.2.2 Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik

Pengujian pengaruh banyaknya iterasi terhadap hasil solusi terbaik dilakukan untuk melihat tingkat konvergensi. Jumlah *size problem* adalah 8, nilai yang digunakan adalah (0, 2, 3, 5, 6, 8, 9, 10), dan jumlah bee colony yang digunakan adalah 40. Menggunakan jumlah bee colony sebanyak 40 dikarenakan pada pengujian sebelumnya dihasilkan nilai rata-rata fitness tertinggi pada bee colony sebanyak 40. Iterasi yang dilakukan sebanyak (5, 10, 15, 20, 25, 30, 35, 40, 45, 50) kali. Pengujiannya ditunjukkan pada Tabel 5.2.

Tabel 5. 2 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik

Iterasi	Hasil Percobaan Ke - i					Rata – Rata *Fitness*
	1	2	3	4	5	
5	0,04357	0,04032	0,04386	0,04283	0,04405	0,04293
10	0,04405	0,04405	0,04405	0,04405	0,04386	0,04201
15	0,04032	0,04425	0,04386	0,04246	0,04425	0,04303
20	0,04405	0,04405	0,04405	0,04032	0,04405	0,0433
25	0,04396	0,04032	0,04405	0,04032	0,04008	0,04275
30	0,04405	0,04405	0,04405	0,04405	0,04405	0,04305
35	0,04032	0,04032	0,04405	0,04405	0,04032	0,04281
40	0,04405	0,04425	0,04396	0,04405	0,04032	0,04335
45	0,04032	0,04396	0,04386	0,04032	0,04032	0,04376
50	0,04396	0,04405	0,04405	0,04405	0,04386	0,0441

Pada Tabel 5.2 dapat dilihat bahwa hasil rata-rata nilai *fitness* terbaik adalah pada jumlah iterasi ke 50, dimana menghasilkan nilai rata-rata sebesar 0,04410.

5.2.3 Pengujian Pengaruh Banyaknya *Size Problem* Terhadap Hasil Solusi Terbaik

Pengujian pengaruh banyaknya s*ize problem* terhadap hasil solusi terbaik dilakukan untuk melihat tingkat konvergensi. Jumlah *Size problem* adalah (6, 7, 8, 9, 10, 11), dengan rute yang digunakan adalah (0, 2, 4, 6, 8, 10), (1, 2, 3, 4, 7, 8, 9), (0, 3, 5, 6, 7, 8, 9, 10), (1, 2, 3, 4, 5, 6, 7, 9, 10), (0, 1, 2, 3, 4, 5, 7, 8, 9, 10) dan (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10). Jumlah *bee colony* yang digunakan adalah 40. Iterasi yang dilakukan sebanyak 50 kali. Adapun pengujiannya ditunjukkan pada Tabel 5.3.

Tabel 5. 3 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik

Jumlah Size Problem	Hasil Percobaan Ke - i					Rata – Rata Fitness
	1	2	3	4	5	
6	0,04082	0,04082	0,04082	0,04082	0,04082	0,04082
7	0,04073	0,04264	0,04073	0,04073	0,04264	0.04049
8	0,04073	0,03745	0,04073	0,04073	0,04073	0.04007
9	0,04071	0,0408	0,04079	0,04072	0,04079	0.04076
10	0,04132	0,03650	0,04237	0,04193	0,03891	0.04021
11	0,03540	0,03534	0,03559	0,04008	0,04396	0.03807

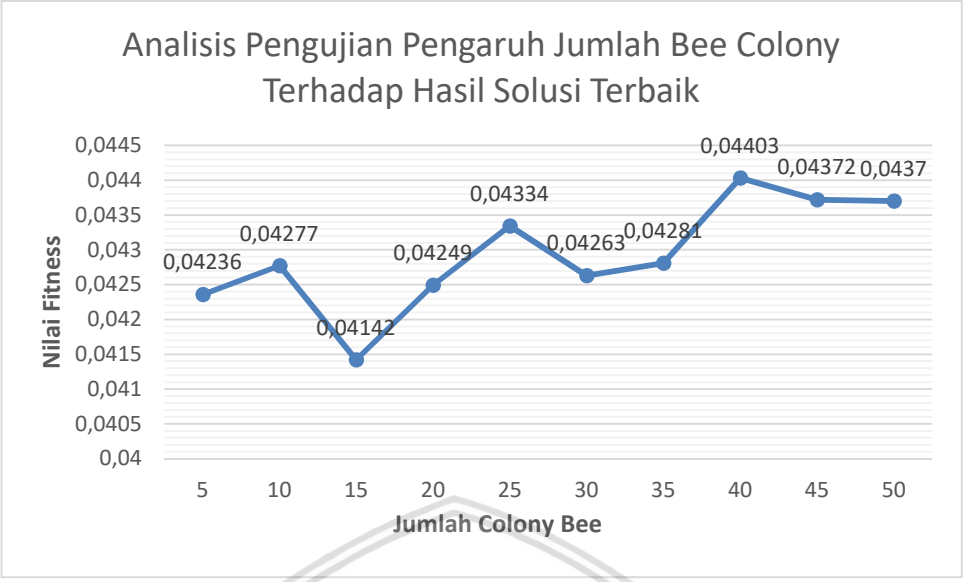
Pada Tabel 5.3 dapat dilihat bahwa jumlah *size problem* sebanyak 6 memiliki perhitungan nilai *fitness* yang konstan, dimana menghasilkan nilai rata-rata sebesar 0,04082.

5.3 Analisis Hasil Pengujian

Analisis hasil pengujian merupakan analisis dari hasil pengujian sebelumnya. Analisis yang dilakukan sebanyak pengujian yang dilakukan yaitu analisis pengujian pengaruh jumlah *bee colony* terhadap hasil solusi terbaik, analisis pengaruh banyaknya iterasi terhadap hasil solusi terbaik, dan analisis pengaruh banyaknya s*ize problem* terhadap hasil solusi terbaik.

5.3.1 Analisis Pengujian Pengaruh Jumlah *Bee Colony* Terhadap Hasil Solusi Terbaik

Pada pengujian terhadap pengaruh jumlah *bee colony* terhadap hasil solusi terbaik yang ditunjukkan pada gambar 5.1, terlihat bahwa nilai *interval* antara pengujian 1 sampai 10 memberikan rata-rata nilai *fitness* tertinggi yaitu 0,04403 dengan jumlah *bee colony* sebanyak 40. Pada pengujian, jumlah *bee colony* berpengaruh pada hasil solusi terbaik karena pada jumlah *bee colony* yang sedikit, nilai *fitness* yang dihasilkan cenderung kurang optimal.

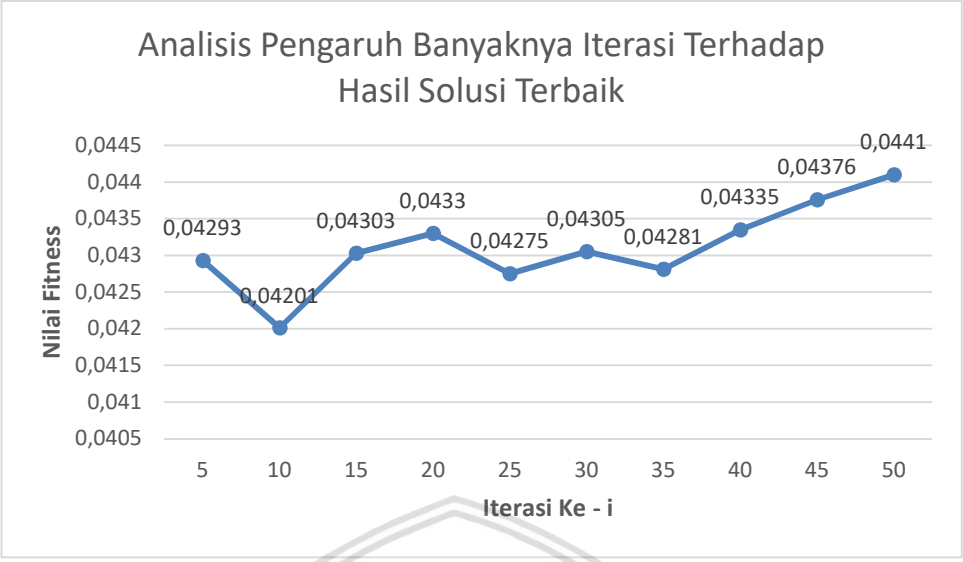


Gambar 5. 1 Analisis Pengujian Pengaruh Jumlah *Bee Colony* Terhadap Hasil Solusi

Dapat dilihat bahwa jumlah *bee colony* antara 5 – 15, menghasilkan nilai *fitness* yang kecil. Hasil nilai *fitness* mulai menunjukkan konvergensi pada jumlah *bee* colony ke 20, dan tidak memperlihatkan penurunan nilai *fitness* yang signifikan. Hal ini juga dipengaruhi oleh pengacakan rute pada tiap individu. Semakin banyak individu yang digunakan, maka rute yang terbentuk juga akan semakin banyak. Sehingga kemungkinan mendapatkan solusi terbaik juga semakin tinggi.

5.3.2 Analisis Pengujian Pengaruh Jumlah *Iterasi* Terhadap Hasil Solusi Terbaik

Pengujian pengaruh jumlah iterasi terhadap hasil solusi terbaik menggunakan jumlah *bee colony* sebanyak 40, dengan 8 *size problem*. Pengujian ini dilakukan dengan banyak iterasi yang berbeda dan dilakukan percobaan pada tiap iterasi sebanyak lima kali. Berdasarkan pada pengujian terhadap pengaruh jumlah iterasi terhadap hasil solusi terbaik yang ditunjukkan pada Gambar 5.2, terlihat bahwa nilai *interval* antara percobaan 1 sampai 10 memberikan rata-rata nilai *fitness* tertinggi yaitu 0,0441 dengan jumlah iterasi sebanyak 50.

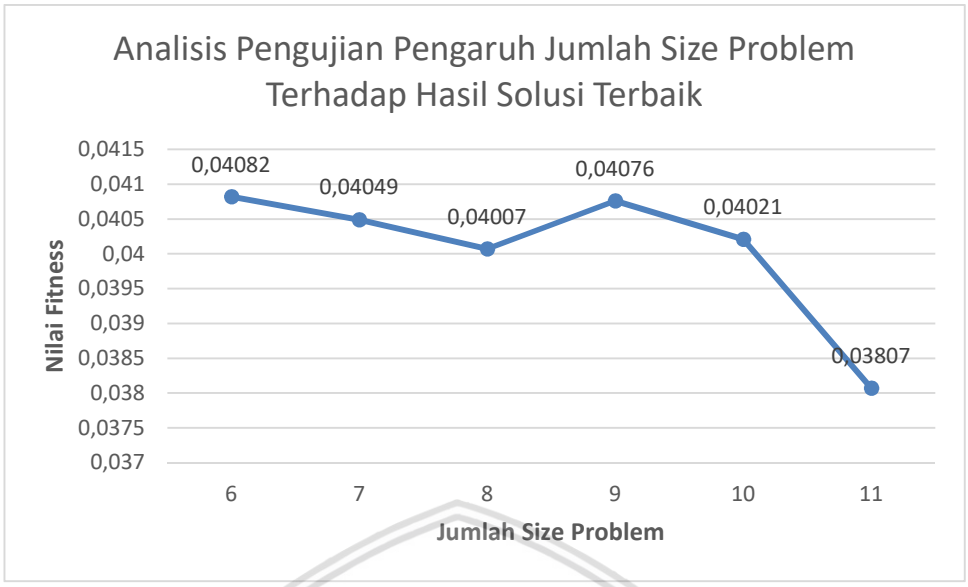


Gambar 5. 2 Hasil Pengujian Pengaruh Banyaknya Iterasi Terhadap Hasil Solusi Terbaik

Pada pengujian, jumlah iterasi berpengaruh pada hasil solusi terbaik karena jumlah iterasi pada rentang yang rendah, nilai *fitness* yang dihasilkan cenderung kurang optimal. Dapat dilihat bahwa jumlah iterasi sebanyak 10, menghasilkan nilai *fitness* yang kecil. Hasil nilai *fitness* mulai menunjukkan konvergensi pada jumlah iterasi sebanyak 20, dan tidak memperlihatkan penurunan hasil *fitness* yang signifikan. Pada iterasi yang tinggi, dapat menghasilkan kemungkinan solusi yang lebih baik karena selalu dilakukan perhitungan secara terus – menerus.

5.3.3 Analisis Pengujian Pengaruh Banyaknya *Size Problem* Terhadap Hasil Solusi Terbaik

Pengujian pengaruh banyaknya *size problem* terhadap hasil solusi terbaik dilakukan untuk melihat tingkat kovergensi. Jumlah *Size problem* adalah (6, 7, 8, 9, 10, 11), dengan rute yang digunakan (0, 2, 4, 6, 8, 10), (1, 2, 3, 4, 7, 8, 9), (0, 3, 5, 6, 7, 8, 9, 10), (1, 2, 3, 4, 5, 6, 7, 9, 10), (0, 1, 2, 3, 4, 5, 7, 8, 9, 10) dan (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10). Jumlah *bee colony* yang digunakan adalah 40. Iterasi yang dilakukan sebanyak 50 kali. Adapun rata – rata hasil pengujiannya ditunjukkan pada Gambar 5.3.



Gambar 5. 3 Hasil Pengujian Pengaruh Banyaknya Size Problem Terhadap Hasil Solusi Terbaik

Dari hasil pengujian, banyaknya *size problem* yang digunakan berpengaruh pada hasil *fitness* yang dihasilkan. Semakin sedikit *size problem* yang digunakan, kemungkinan mendapatkan nilai *fitness* juga semakin tinggi. Hal ini disebabkan, perhitungan rute yang dilakukan tidak terlalu banyak. Sehingga pengacakan yang dilakukan juga sedikit.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian sistem optimasi rute tempat wisata kuliner di Malang menggunakan algoritma *bee colony*, dapat diambil kesimpulan sebagai berikut:

1. Algoritma *bee colony* dapat diterapkan pada sistem optimasi rute tempat wisata kuliner dengan hasil yang lebih optimal. Adapun langkah-langkah dalam merancang sistem optimasi ini adalah:
   a. Pertama adalah melakukan proses fase *initial*. Pada fase ini, dibentuk rute pada setiap *bee colony*. Setiap *bee colony* memiliki rute dan total jarak dari setiap rute, sehingga dapat dihasilkan nilai yang disebut nilai *fitness*.
   b. Kedua adalah melakukan perhitungan yang terdiri dari proses fase *employeed bee*, *onlooker bee* dan *scout bee*.
   c. Ketiga adalah uji hasil dari perhitungan *bee colony* yaitu diujikan jumlah *bee* colony, jumlah iterasi, dan jumlah *size problem* yang digunakan untuk mendapatkan nilai rata-rata *fitness* agar dapat dianalisis.
2. Nilai rata-rata hasil *fitness* maksimum diperoleh pada pengujian jumlah *bee colony* ke 40 dengan nilai 0,04403 dan iterasi 50 dengan nilai 0,04410.

6.2 Saran

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

1. Pada penelitian selanjutnya terkait dengan perkembangan data agar metode ini dapat diimplementasikan dengan menggunakan *database* untuk data yang lebih besar.
2. Pada penelitian selanjutnya agar melakukan penelitian dengan pengujian pada parameter limit dan algoritma *bee colony* dapat diterapkan pada objek lain.

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.

Chong, C. S., Low, M. Y. H., Sivakumar, A. I. & Gay, K. L., 2006. *A BEE COLONY OPTIMIZATION ALGORITHM TO JOB SHOP SCHEDULING.* Nanyang, Winter Simulation Conference.

Chong, C. S., Low, M. Y. H., Sivakumar, A. I. & Gay, K. L., 2007. Using a bee colony algorithm for neighborhood search in job shop scheduling problems," in Proc. of 21st European Conference on Modeling and Simulation (ECMS2007).

Danuri & Prijodiprodjo, W., 2013. Penerapan Bee Colony Optimization Algorithm untuk Penentuan Rute Terpendek (Studi Kasus : Objek Wisata Daerah Istimewa Yogyakarta). *IJCCS,* Volume 7, pp. 65-76.

Darto Paulus Siregar, 2011. Optimasi Penjadwalan Kuliah Dengan Metode *Tabu Search.* S1. Universitas Sumatera Utara.

Desiani, A., Arhami, M.2005. Konsep Kecerdasan Buatan. Yogyakarta: Andi.

Dyardian, 2015. *DYARDIAN BLOG.* [Online] Available at: http://dyardian.heck.in/ini-adalah-pengertian-dasar-optimasi.xhtml [Diakses 11 Agustus 2015].

Iqbal, M., 2016. *D3 Teknik Telekomunikasi, Telkom University.* [Online] Available at: http://miqbal.staff.telkomuniversity.ac.id/optimasi-convex/ [Diakses 20 Mei 2016].

Izzah, A., Dewi, R. K. & Mutrofin, S., 2013. *HYBRID ARTIFICIAL BEE COLONY :PENYELESAIAN BARU POHON RENTANG BERBATAS DERAJAT.* Yogyakarta, Seminar Nasional Teknologi Informasi dan Multimedia.

Karaboga, D., 2005. *AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION,* Kayseri: Erciyes University.

Karaboga, D. & Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J Glob Optim ,* p. 459–471.

Karaboga. S. and Basturk. D., 2008, On the performance of artificial bee colony (abc) algorithm, Applied Soft Computing, vol. 8, no. 1, pp. 687-697.

Kusrini, K. & Istiyanto, J. E., 2008. PENYELESAIAN TRAVELLING SALESMAN PROBLEM DENGAN ALGORITMA CHEAPEST INSERTION HEURISTICS DAN BASIS DATA. *Jurnal Informatika,* Volume 8, pp. 109-114.

Maria, A., Sinaga, E. Y. & Helena, M., 2012. Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO).

Nakrani, S. and C. Tovey, 2004. On honey bees and dynamic allocation in an internet server colony. Adaptive Behavior, 12(3-4):p. 223-240.

Otri, S., 2011. *IMPROVING THE BEES ALGORITHM FOR COMPLEX OPTIMISATION PROBLEMS ,* Cardiff: s.n.

Permata, Y. I., 2011. *Manajemen Pengembangan Wisata Kuliner di Gladag Langen Bogan Surakarta,* Semarang: s.n.

Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S and Zaidi M., 2006. Thee Bees Algorithm – A Novel Tool for Complex Optimisation Problems. Intelligent Systems Laboratory, Manufacturing Engineering Centre, Cardiff University, UK.

Saidah, N. H., Er, M., & Soelaiman, R. 2014. Penyelesaian Masalah travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO).

Singh, Anshul, 2012. Augmentation of Travelling Salesman Problem using Bee Colony Optimization.

Suyanto, 2010. *Algoritma Optimasi :Deterministik atau Probabilistik.* Yogyakarta: Graha Ilmu.

Teodorovic, D and Dell'orco, M, 2005. Bee colony optimization - A cooperative learning approach to complex transpor-tation problems, Advanced OR and AI Methods in Transportation, pp. 51-60.

Teodorovic. D., 2008. Swarm Intelligence Systems for transportation engineering:Principles and Applications, Transportation Research Part C: Emerging Technologies, vol 16, no. 6, pp. 651-657.