TUGAS AKHIR – IS184853

OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*

OPTIMIZATION OF TRIP ROUTES SCHEDULING AEROPLANE USING XU FUZZY GENETIC ALGORITHM TO SOLVE THE PROBLEM IN TRAVELING SALESMAN CHALLENGE 2.0

YAYAN IRFAN FERDIAWAN
NRP 05211540000062

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IS184853

OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*

YAYAN IRFAN FERDIAWAN
NRP 05211540000062

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

FINAL PROJECT – IS184853

OPTIMIZATION OF TRIP ROUTES SCHEDULING AEROPLANE USING XU FUZZY GENETIC ALGORITHM TO SOLVE THE PROBLEM IN TRAVELING SALESMAN CHALLENGE 2.0

YAYAN IRFAN FERDIAWAN
NRP 05211540000062

Supervisor
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information and Communication Technology
Institute of Technology Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Oleh:

YAYAN IRFAN FERDIAWAN
NRP. 05211540000062

Surabaya, 17 Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi ER, S.T., M.Sc., Ph.D
NIP. 19761011 200604 2 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

YAYAN IRFAN FERDIAWAN
NRP. 05211540000062

Disetujui Tim Penguji : Tanggal Ujian : 8 Juli 2019
Periode Wisuda : September 2019

Ahmad Muklason, S.Kom., M.Sc., Ph.D. **(Pembimbing I)**

Edwin Riksakomara, S.Kom, MT. **(Penguji I)**

Raras Tyasnurita, S.Kom, M.BA., Ph.D. **(Penguji II)**

Halaman ini sengaja dikosongkan

OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*

Nama Mahasiswa	**:**	**Yayan Irfan Ferdiawan**
NRP	**:**	**05211540000062**
Departemen	**:**	**Sistem Informasi**
Dosen Pembimbing	**:**	**Ahmad Muklason, S.Kom., M.Sc., Ph.D**

ABSTRAK

Permasalahan penjadwalan rute perjalanan atau traveling salesman problem (TSP) adalah salah satu permasalahan optimasi. Tujuan dari TSP adalah menemukan biaya atau jarak minimal yang ditempuh dalam melakukan perjalanan dengan aturan yang ditetapkan. Aturan tersebut adalah harus melalui kota tepat satu kali dalam rute tersebut dan kembali lagi ke kota awal. Traveling salesman challenge 2.0 (TSC 2.0) adalah kompetisi dalam menyelesaikan permasalahan TSP dengan pesawat terbang. Dalam TSC 2.0 seorang salesman harus mengunjungi semua wilayah yang ada, dimana setiap wilayah memiliki satu atau lebih kota tetapi jika salah satu kota dalam wilayah tersebut sudah dikunjungi maka salesman tidak boleh mengunjungi wilayah tersebut lagi. Banyaknya batasan yang ada dalam permasalahan TSC 2.0, membuat tantangan tersendiri untuk menyelesaikan permasalahan ini. Untuk menyelesaikan permasalahan TSC 2.0 digunakan algoritma fuzzy genetika yang mana merupakan gabungan antara fuzzy

logic dan algoritma genetika. Karena algoritma genetika memiliki kelemahan dalam menentukan range parameter maka untuk menentukan parameter dalam algoritma genetika digunakanlah fuzzy logic model Xu. Fuzzy logic model Xu sendiri adalah model yang memiliki dua input dan dua output. Dalam implementasi algoritma ini keanggotaan fuzzy yang digunakan adalah berbentuk kurva S. Logika fuzzy akan berperan dalam menentukan parameter probabilitas crossover dan probabilitas mutasi. Untuk algoritma genetika dalam melakukan pemilihan parent akan dipilih dari 10 individu terbaik dalam populasi. Dari hasil penelitian yang dilakukan ternyata algoritma fuzzy-genetika terbukti lebih baik daripada algoritma genetika. Hasil pengujian algoritma fuzzy-genetika menghasilkan penghematan biaya sebesar 21,51% dari biaya awal sedangkan algoritma genetika sebesar 16,06%.

Kata kunci: *traveling salesman problem, genetic algorithm, fuzzy logic*

OPTIMIZATION OF TRIP ROUTES SCHEDULING AEROPLANE USING XU GENETIC FUZZY ALGORITHM TO SOLVE THE PROBLEM IN TRAVELING SALESMAN CHALLENGE 2.0

Name	**:**	**Yayan Irfan Ferdiawan**
NRP	**:**	**05211540000062**
Department	**:**	**Information Systems**
Supervisior	**:**	**Ahmad Muklason, S.Kom., M.Sc., Ph.D**

ABSTRACT

The problem of scheduling travel routes or traveling salesman problems (TSP) is one of the optimization problems. The purpose of the TSP is to find the minimum cost or distance taken in traveling with the rules set. The rule is to go through the city exactly once in the route and return to the initial city. Traveling challenge 2.0 salesman (TSC 2.0) is a competition in solving TSP problems with airplanes. In TSC 2.0, a salesman must visit all existing regions, where each region has one or more cities, but if one city in the area has been visited, salesmen may not visit the area again. The many limitations that exist in the TSC 2.0 problem, create a challenge to solve this problem. To solve TSC 2.0 problems, fuzzy genetic algorithms are used which are a combination of fuzzy logic and genetic algorithms. Because genetic algorithms have a weakness in determining the range of parameters, to determine parameters in the genetic algorithm,

Xu's fuzzy logic model is used. Xu's Fuzzy logic model itself is a model that has two inputs and two outputs. In implementing this algorithm fuzzy membership used is in the form of an S curve. Fuzzy logic will play a role in determining the probability of crossover parameters and the probability of mutations. For the genetic algorithm in selecting parents, 10 of the best individuals in the population will be selected. The results of testing the fuzzy-genetic algorithm resulted in cost savings of 21.51% of the initial cost while the genetic algorithm was 16.06%.

***Keywords**: traveling salesman problem, genetic algorithm, fuzzy logic*

KATA PENGANTAR

Puji syukur atas karunia yang telah diberikan Allah SWT. selama ini sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul: "**OPTIMASI PENJADWALAN RUTE PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN *ALGORITMA FUZZY GENETIKA MODEL XU* UNTUK MENYELESAIKAN PERMASALAHAN PADA *TRAVELING SALESMAN CHALLENGE 2.0*"**

Pengerjaan tugas akhir ini tidak lepas dari bantuan, bimbingan, dukungan dan doa dari berbagai pihak. Oleh karena itu, izinkan penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada,

1. Allah SWT. yang telah memberikan kemudahan dan jalan dalam mengerjakan tugas akhir ini.
2. Orang tua penulis yaitu Bapak dan Ibu yang telah memberikan dukungan baik segi material, semangat dan doa agar bisa menyelesaikan tugas akhir ini dengan lancar.
3. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing yang telah meluangkan banyak waktu untuk membimbing serta mengarahkan penulis dalam pengerjaan tugas akhir ini.
4. Bapak Edwin Riksakomara, S.Kom., MT. dan Ibu Raras Tyasnurita, S.Kom., MBA. selaku dosen penguji yang telah memberikan kritik dan saran yang membangun agar tugas akhir ini menjadi lebih baik.

5. I Gusti Agung Premananda, selaku teman dan merangkap menjadi dosen pembimbing 2 yang telah membantu dalam proses tugas akhir ini.
6. Saudara Gilang, Edwin, Irul, Supri, Kotak dan lainnya yang dilupakan karena telah memberikan motivasi dan arahan selama proses pengerjaan tugas akhir ini.

Penulis juga menyadari pengerjaan penelitian tugas akhir ini masih jauh dari kata sempurna. Oleh karena itu, penulis menerima pertanyaan, saran, dan kritik yang membangun untuk menjadi masukan penulis dan masukan penelitian selanjutnya. Semoga penelitian tugas akhir dapat memberikan manfaat bagi pembaca.

Surabaya, 02 Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Pada bab pendahuluan ini akan menjelaskan tentang latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan relevansi. Dengan penjelasan singkat tersebut diharapkan dapat membantu dalam memperkenalkan masalah yang sedang dihadapi.

1.1. Latar Belakang

Permasalahan rute perjalanan atau lebih sering dengan *Travelling Salesman Problem (TSP)* merupakan permasalahan metaheuristik yang penyelesaiannya diselesaikan dengan waktu yang cukup lama atau dapat dibilang sulit[1]. Untuk menemukan sebuah solusi yang optimal diperlukan waktu yang sangat lama jika memiliki kompleksitas yang besar. Banyak pakar berlomba-lomba dalam menemukan solusi yang optimal. Sejak ditemukannya metode Branch-and-Bound oleh Danztig, Fulkerson dan Johnson pada tahun 1954 hingga saat ini belum ada yang dapat menyelesaikan permasalahan *TSP* secara optimal dengan hitungan detik[2].

Pada tahun 2018, sebuah perusahaan travel online dengan nama kiwi.com mengadakan kompetisi dalam menyelesaikan permasalahan mengenai *TSP* yang diberi nama *Traveling Salesman Challenge 2.0* (*TSC* 2.0). Dalam kompetisi tersebut terdapat 14 buah dataset yang disediakan 10 adalah data artificial dan 4 data lainnya adalah data asli penerbangan dari perusahaan kiwi.com. Dari dataset tersebut akan diolah sehingga mengeluarkan hasil yang optimal yaitu sebuah rute perjalanan yang memiliki biaya seminimal mungkin. Selain itu pada dataset ini setiap kota hanya boleh dikunjungi satu kali

saja. Jika sebuah kota di suatu daerah sudah dikunjungi maka kota lain di daerah yang sama tidak boleh dikunjungi. Selain dari permasalahan tersebut terdapat juga jadwal penerbangan antara setiap kota yang tidak tersedia setiap harinya. Hal tersebutlah yang mendasari mengapa permasalahan tersebut dapat diselesaikan dengan menggunakan konsep *TSP*.

Salah satu metode yang marak digunakan dalam menyelesaikan permasalahan TSP adalah menggunakan algoritma genetika. Algoritma genetika sendiri sudah ada sejak tahun 1988 dikenalkan oleh D. E. Golberg dan John H. Holland dalam jurnal ilmiahnya yang berjudul *"Genetic Algorithms in Search, Optimization, and Machine Learning"*[3]. Algoritma genetika adalah metode pencarian solusi heuristik yang terinspirasi oleh teori evolusi dan dapat diterapkan pada berbagai masalah pembelajaran dan optimisasi[4]. Namun, terdapat aspek yang sulit dalam menggunakan algoritma genetika yaitu menemukan parameter yang tepat untuk membantu dalam menemukan solusi yang optimal[5]. Sehingga algoritma genetika saja dirasa kurang dalam menyelesaikan permasalahan secara cepat dan mendapatkan solusi yang baik.

Untuk menyempurnakan algoritma genetika perlu adanya suatu metode yang digunakan untuk membantu dalam menemukan parameter terbaik. Penggunaan teori fuzzy melalui perkiraan adalah alat yang sangat kuat untuk memperluas kemampuan logika biner dengan cara yang memungkinkan representasi pengetahuan manusia[6]. Penalaran kualitatif, interpolasi dan penalaran analog dapat diuntungkan dari penanganan yang tepat dari hubungan kesamaan fuzzy dan dapat berguna dalam sistem yang lebih maju[7]. Dalam langkah menyelesaikan permasalahan *TSP* sebuah penelitian menggabungkan fuzzy

logic dengan algoritma genetika yang dinamai dengan algoritma fuzzy genetika[8]. Dari penelitian tersebut ternyata membuktikan bahwa algoritma fuzzy dapat memberikan nilai probabilitas rekombinasi dan mutasi untuk algoritma genetika. Selain itu, dalam penelitian tersebut mengatakan bahwa metode tersebut mampu memberikan hasil yang lebih optimum dalam menyelesaikan permasalahan yang ada[8].

Berdasarkan penelitian yang sudah dilakukan, maka dipilihlah algoritma fuzzy genetika sebagai penyelesaian permasalahan *TSC* 2.0. Dengan mengimplementasikan algoritma tersebut, diharapkan dapat membantu dalam menyelesaikan permasalahan *TSP* dan menemukan solusi yang paling optimal.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, berikut adalah rumusan masalah berupa pertanyaan-pertanyaan yang akan terjawab dengan adanya tugas akhir ini:

1. Bagaimana model matematis permasalahan optimasi penjadwalan rute perjalanan pesawat terbang dari *TSC* 2.0?
2. Bagaimana menerapkan algoritma *Fuzzy Evolusi* dalam menyelesaikan permasalahan optimasi penjadwalan rute perjalanan pesawat terbang dari *TSC* 2.0?
3. Bagaimana performa algoritma *Fuzzy Evolusi* dalam menyelesaikan permasalahan optimasi penjadwalan rute perjalanan pesawat terbang dari *TSC* 2.0?

1.3 Batasan Masalah

Dari permasalahan yang disebutkan di atas, batasan masalah dalam tugas akhir ini adalah:

1. Data yang digunakan dalam menyelesaikan permasalahan optimasi penjadwalan rute perjalanan pesawat terbang berasal dari dataset *TSC* 2.0.
2. Aplikasi yang dibangun menggunakan Bahasa pemrograman java.
3. Aplikasi yang dibangun menggunakan algoritma fuzzy-genetika yang mana fungsi keanggotaan yang dipakai didapatkan dari referensi sebelumnya.

1.4 Tujuan Tugas Akhir

Dari permasalahan yang disebutkan sebelumnya, tujuan yang ingin dicapai dalam tugas akhir ini adalah:

1. Membuat model matematis dari permasalahan optimasi penjadwalan rute perjalanan pesawat dari *TSC* 2.0.
2. Menerapkan algoritma *Fuzzy Evolusi* untuk menyelesaikan permasalahan optimasi penjadwalan rute perjalanan pesawat terbang dari *TSC* 2.0.
3. Melakukan Analisa performa algoritma *Fuzzy Evolusi* dalam menyelesaikan permasalahan optimasi penjadwalan rute perjalanan pesawat terbangdari *TSC* 2.0.

1.5 Manfaat Tugas Akhir

Beberapa manfaat yang diharapkan dapat tercapai pada penelitian ini adalah:

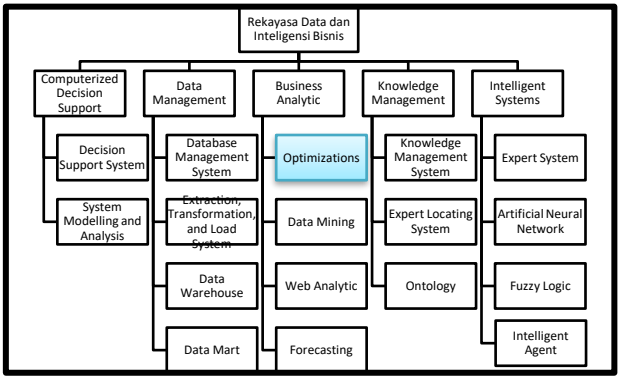
1. Penelitian ini diharapkan dapat menambah wawasan di bidang keilmuan sistem informasi khususnya terkait

dengan penyelesaian permasalahan *TSP* menggunakan algoritma *Fuzzy* Genetika.

2. Penelitian ini diharapkan dapat menjadi rujukan untuk penelitian selanjutnya berkaitan dengan penyelesaian *TSP* menggunakan algoritma *Fuzzy* Genetika.
3. Penelitian ini diharapkan dapat memberikan gambaran kepada penelitian selanjutnya berkaitan dengan penyelesaian *TSP* menggunakan algoritma *Fuzzy* Genetika.

1.6 Relevansi

Penelitian ini berada pada bidang optimasi pada pokok bahasan *Business Analytic* dan merupakan rumpun ilmu penelitian di salah satu lab Sistem Informasi yaitu Rekayasa Data dan Intelegensi Bisnis (RDIB). Penelitian ini berkaitan erat dengan salah satu matakuliah pilihan yaitu Optimasi Kombinatorik dan Heuristik (OKH). Untuk detailnya dapat dilihat pada Gambar 1.1.



Gambar 1.1 Roadmap Penelitian Laboratorium Rekayasa Data dan Intelegensi Bisnis

BAB 2
TINJAUAN PUSTAKA

Dalam bab ini, akan dijelaskan mengenai penelitian sebelumnya dan landasan teori yang digunakan sebagai acuan dalam pengerjaan tugas akhir.

2.1. Studi Sebelumnya

Sebelum penelitian ini dilakukan, sudah banyak penelitian permasalahan *TSP* dengan menggunakan metode fuzzy dan genetika. Pada Tabel 2.1 dapat dilihat mengenai beberapa studi literatur sebelumnya yang menjadi dasar penelitian tugas akhir ini.

Tabel 2.1 Studi Sebelumnya

No	Studi Sebelumnya	
1	Nama Peneliti	Syafiul Muzid
	Tahun Penelitian	2014
	Judul Penelitian	Dinamisasi Parameter Algoritma Genetika Menggunakan *Population Resizing On Fitness Improvement Fuzzy Evolutionary Algorithm (Profifea)*[9]
	Penjelasan Singkat	Dalam penelitian ini digunakan *fuzzy logic* dan algoritma genetika yang menerapkan nilai perubahan

		pada setiap iterasi nilai populasi pada algoritma genetika. Perubahan tersebut dikarenakan penulis menganggap besar atau kecilnya parameter populasi sangat berpengaruh pada hasil yang didapatkan. Jika populasi terlalu kecil maka akan terjadi konversi dini, sedangkan jika terlalu besar akan mengakibatkan lamanya waktu dalam menghasilkan solusi. Metode *fuzzy* sendiri disini digunakan untuk mendapatkan nilai parameter crossover dan mutasi. Dalam penelitian ini menggunakan model Xu dimana memiliki 2 buah masukan yaitu parameter populasi dan iterasi, sedangkan 2 buah keluaran yaitu parameter crossover dan mutasi.
	Hasil Penelitian	Dalam penelitian ini, penulis mengimplementasikan algoritma *fuzzy* genetika untuk 20 kota yang posisinya digambarkan dengan titik koordinat x dan y. Dalam penelitian ini penulis juga membandingkan hasil dari algoritma *fuzzy* genetika dengan algoritma genetika. Dari perbandingan tersebut ternyata

		algoritma *fuzzy* genetika mampu menghasilkan solusi lebih baik daripada algoritma genetika.
	Keterkaitan Penelitian	Hubungan dari penelitian ini dengan sebelumnya adalah algoritma dan model yang dipakai akan sama. Algoritma *fuzzy* genetika model Xu akan diimplementasikan untuk menyelesaikan permasalahan *TSC 2.0*. Terdapat 300 destinasi area yang mana tujuannya adalah menemukan biaya minimal untuk melakukan perjalanan mengelilingi area yang ditentukan. Bentuk permasalahan *TSC* 2.0 bersifat asimetris dimana biaya A ke B tidak sama dengan biaya B ke A.
2	Nama Peneliti	Antonia P. Plerou
	Tahun Penelitian	2017
	Judul Penelitian	*Fuzzy Genetic Algorithms: Fuzzy Logic Controllers and Genetics Algorithms*[10]
	Penjelasan Singkat	Dalam jurnal ini berisikan tentang konsep *fuzzy logic* dan algoritma genetika digunakan. Dalam paper ini menjelaskan tentang pengaruh parameter yang digunakan serta

		kelebihan dari penggunaan *fuzzy logic* di dalam algoritma genetika. Dalam algoritma genetika pemilihan parameter sangat berpengaruh kepada hasil yang didapatkan. Dari permasalahan tersebut maka digunakan konsep fuzzy untuk mendapatkan hasil yang lebih baik. Akan tetapi perlu adanya analisis model yang teliti untuk model fuzzy yang disebut dengan *fuzzy logic controller*.
	Hasil Penelitian	Jurnal ini tidak melakukan implementasi tentang penggunaan *fuzzy logic* dalam algoritma genetika. Jurnal ini hanya berisi tentang konsep dasar *fuzzy logic* dan penggunaannya dalam algoritma genetika.
	Keterkaitan Penelitian	Dalam jurnal tersebut berisikan konsep dasar mengenai algoritma gabungan antara *fuzzy logic* dan algoritma genetika yang mana akan digunakan dalam membantu memahami konsep algoritma ini.
3	Nama Peneliti	Nicholas Ernest dan Kelly Cohen
	Tahun Penelitian	2014

	Judul Penelitian	*Fuzzy Logic Clustering of Multiple Traveling Salesman Problem for Self-Crossover Based Genetic Algorithm*[11]
	Penjelasan Singkat	Dalam penelitian ini, algoritma genetika dan metode *fuzzy logic* diterapkan dalam *Multiple Traveling Salesman Problem* atau lebih dikenal dengan *Vehicle Routing Problem.* Dalam penelitian ini penulis menyebut nama algoritmanya dengan *UNCLE (UNburdening through CLustering Efficiently)* dan *SCOORAGE (Self-CROssover GEnetic algorithm).* Pada kasus ini terdapat 5 sales yang akan melalui 100 kota. Dalam algoritma ini pertama-tama system akan melakukan clustering sebanyak 5 cluster berdasarkan koordinat kota terdekat. 5 cluster tersebut dikarenakan banyaknya salesman sebanyak 5 buah. Setelah didapatkan cluster tersebut maka dicarilah rute terdekat dengan menggunakan algoritma genetika.
	Hasil Penelitian	Dari hasil penelitian ini peneliti menggunakan algoritma lain untuk melakukan perbandingan algoritma

		tersebut antara lain *Raw Fuzzy C-Means Clustering, Fixed Membership Function Width, Density-based Membership Function Width dan UNCLE-SCOORAGE*. Dari penelitian tersebut didapatkan hasil bahwa *UNCLE-SCOORAGE* lebih optimal dibandingkan 4 baik sisi jarak dan waktu.
	Keterkaitan Penelitian	Dalam penelitian tersebut algoritma genetika dan fuzzy logic diterapkan pada 100 kota dan 5 buah kendaraan sedangkan pada tugas akhir ini hanya 1 salesman yang mengelilingi lebih dari 200 wilayah. Dalam tugas akhir ini juga memiliki batasan yang lebih rumit dibandingkan *TSP* biasa karena harus mempertimbangkan jadwal pesawat terbang seperti di kehidupan nyata.

Berdasarkan studi review diatas, algoritma fuzzy dan genetika memang memiliki kelebihan dan kekurangan. Salah satu kekurangannya adalah harus mengatur model pada fuzzy logic sebelum menjalankan algoritma genetika.

3.2 Dasar Teori

Bagian ini menjelaskan dasar teori apa saja yang dijadikan sebagai acuan atau landasan dalam pengerjaan tugas akhir ini.

2.2.1. Travelling Salesman Problem

Permasalahan TSP adalah salah satu permasalahan paling marak dalam bidang optimasi kombinatorik. Dimana tujuannya adalah mencari solusi terbaik dalam waktu yang singkat[12]. Permasalahan ini muncul sudah jauh hari sebelum komputer ditemukan. Akan tetapi baru populer pada tahun 1931 setelah munculnya buku yang berjudul "The Traveling Salesman Problem, how he should be and what he should do to be successful in his business. By a veteran traveling salesman[2]".

Permasalahan TSP sendiri adalah permasalahan yang dihadapi salesman saat bepergian. Salesman tersebut harus mengunjungi serangkain kota dan tiap kota tersebut harus dikunjungi sebanyak satu kali kemudian kembali lagi ke kota awal salesman tersebut berada[1]. Banyak sekali contoh permasalahan TSP dalam kehidupan sehari seperti penjadwalan rute kereta, pengaturan rute bus dan masih banyak lainnya. Adapun model matematis TSP yang normal biasanya ditulis seperti persamaan berikut[12]:

$$X_{ij} = \begin{cases} 0 \\ 1 \end{cases} \qquad j \neq i \qquad (1)$$

Fungsi tujuan

$$Min \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ij} \qquad j \neq i \qquad (2)$$

Batasan

$$\sum_{i=0, i\neq j}^{n} X_{ij} = 1 \qquad j = 0, \dots, n \qquad (3)$$

$$\sum_{j=0,\ j\neq i}^{n} X_{ij} = 1 \qquad i = 0, \dots, n \qquad (3)$$

$$u_i - u_j + nX_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n \qquad (4)$$

Penjelasan:
X_{ij}= Rute perjalanan melewati kota i dan menuju kota j
C_{ij}= Biaya perjalanan yang dikeluarkan untuk berpindah dari kota i ke kota j
u_i= Bilangan dummy yang diberikan saat rute melewati kota ke-i
u_j= Bilangan dummy yang diberikan saat rute melewati kota ke-j
n= jumlah kota yang ada pada permasalahan

Dalam keempat persamaan yang telah dituliskan, sudah mencakup semua aturan permasalahan *TSP* baik variabel keputusan, fungsi tujuan dan Batasan-batasan yang ada. Persamaan satu (1) menunjukan variabel keputusan bahwa apakah salesman melalui perjalanan dari kota i ke kota j yang bernilai bilangan biner benar atau salah. Persamaan dua (2) menunjukan fungsi tujuan dimana rute yang dilalui salesman merupakan biaya yang minimal. Persamaan tiga (3) menunjukan batasan yaitu setiap kota hanya boleh dilalui sebanyak satu kali saja. Persamaan empat (4) menunjukan batasan dimana semua kota terhubung menjadi sebuah rute yang dilalui oleh seorang *salesman* dengan menambahkan bilangan dummy untuk mengecek tidak ada sub rute.

2.2.2. Dataset Traveling Salesman Challenge 2.0 (TSC 2.0)

Pada tahun 2018, perusahaan travel online dengan nama kiwi.com mengadakan sebuah kompetisi penyelesaian

masalah *TSP*. Kompetisi *TSP* tersebut diberi nama dengan *Traveling Salesman Challenge* 2.0 (*TSC* 2.0). Sama seperti permasalahan *TSP* pada umumnya, pada *TSC* 2.0 ini bertujuan untuk meminimalkan biaya yang dikeluarkan seorang salesman untuk mengelilingi kota yang ada dan kembali ke kota asal dengan menggunakan pesawat terbang. Tetapi, permasalahan *TSC* 2.0 ini agak berbeda dengan permasalahan *TSP* pada umumnya. Adapun model yang ada pada *TSC* 2.0 yaitu:

- Tujuan: Menemukan biaya minimal yang digunakan untuk mengelilingi seluruh area yang menjadi destinasi seorang salesman dengan menggunakan pesawat terbang.
- Variabel keputusan: Kota mana saja yang dipilih dalam melakukan rute perjalanan menggunakan pesawat terbang.
- Batasan:
    1. Perjalanan dimulai dari kota yang sudah ditentukan.
    2. Salesman harus mengunjungi setiap area tepat satu kali.
    3. Pada tiap area salesman harus mengunjungi salah satu kota dalam area tersebut tepat satu kali.
    4. Perpindahan salesman antar area dilakukan setiap hari.
    5. Setiap hari seorang salesman hanya boleh mengunjungi satu area.

6. Perjalanan harus berakhir di area yang sama dengan area salesman pertama kali memulai perjalanan.
7. Perjalanan salesman ke area selanjutnya harus berlanjut dari kota di area sebelumnya yang telah dikunjungi pada satu hari sebelumnya.

Dalam *TSC* 2.0 disediakan 14 data yang memiliki ukuran berbeda-beda. Adapun rincian data tersebut yaitu 3 dataset berukuran kecil, 3 dataset berukuran medium dan 8 lainnya berukuran besar. Selain dari ukurannya terdapat 2 jenis data yaitu *artificial* data dan data real. Untuk keterangan lebih detail terkait data yang akan diselesaikan dapat dilihat pada Tabel 2.2.

Gambar 2.1 Baris pertama data input berisikan informasi banyaknya area dan dari kota mana perjalanan tersebut dimulai. Sebagai contoh tertulis “13 GDN” menunjukan terdapat 13 area yang harus dikunjungi oleh salesman dan perjalanan dimulai dari kota GDN. Sedangkan setiap 2 baris setelah baris pertama berisikan informasi nama wilayah dan kota apa saja yang terdapat pada area tersebut.

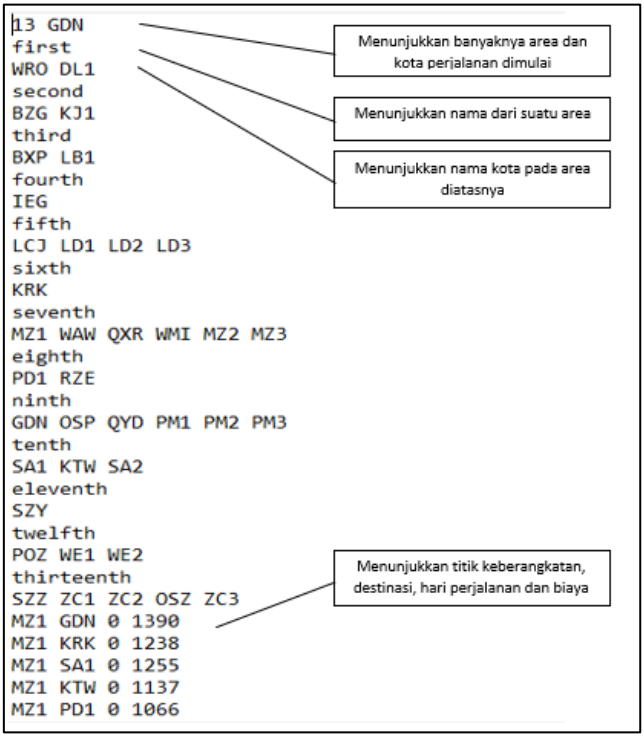
Pada baris ke-2 tertulis “first” dan pada baris ke-3 tertulis “WR0 DL1” menunjukan nama area dan nama kota pada area tersebut. Pada baris kota dipisahkan spasi untuk menandakan bahwa bukan sebuah kota tetapi lebih dari satu (contoh: “WR0 DL1” yang berarti terdapat 2 buah kota yaitu WR0 dan DL1). Setelah semua area sudah dituliskan, baris selanjutnya berisikan rincian informasi perjalanan pesawat terbang yang dituliskan dengan format “d e f g” yang dimana “d” menandakan titik keberangkatan

pesawat terbang, “e” menandakan kota tujuan pesawat terbang, “f” menandakan hari perjalanan berapakah jadwal pesawat tersebut tersedia dan “g” menandakan biaya yang dikeluarkan untuk menempuh perjalanan menggunakan pesawat terbang tersebut. Untuk notasi “f” jika tertulis dengan nilai “15” maka pesawat terbang tersebut hanya tersedia pada hari perjalanan ke-15 dan jika jika tertulis “0” maka pesawat terbang tersebut tersedia setiap hari.

Pada baris yang lain tertulis “MZ1 GDN 0 1390” yang berarti titik keberangkatan pesawat terbang adalah kota “MZ1”, kota tujuan pesawat terbang adalah kota “GDN”, “0” menunjukan bahwa pesawat tersedia setiap hari dan biaya yang dikeluarkan untuk menempuh perjalanan menggunakan pesawat terbang adalah “1390”.

Tabel 2.2 Dataset TSC 2.0

Dataset	Ukuran	Tipe Data	Jumlah Area	Jumlah Kota	Banyaknya Hari
1	Small	*Artificial*	10	10	10
2	Small	*Artificial*	10	15	10
3	Small	*Artificial*	13	38	13
4	Medium	*Artificial*	40	99	40
5	Medium	*Artificial*	46	138	46
6	Medium	*Artificial*	96	192	96
7	Big	Artificial	150	300	150
8	Big	Artificial	200	300	200
9	Big	Artificial	250	250	250
10	Big	Artificial	300	300	300
11	Big	Real	150	200	150
12	Big	Real	200	250	200
13	Big	Real	250	275	250
14	Big	Real	300	300	300



Gambar 2.1 Format dataset *TSC* 2.0

2.2.3. Hyper Heuristik

Perbedaan mendasar dari heuristik dan metaheuristik adalah heuristik secara tegas hanya menggunakan *domain knowledge* (pengetahuan tentang masalah) untuk mempercepat solusi. Misalnya, jika memecahkan masalah TSP, dan berada di kota A, maka logika heuristik bisa saja "kota terdekat dengan A". Biasanya heuristik memberikan solusi yang sangat cepat (konvergensi sangat cepat), namun dapat dengan mudah terjebak pada optimal lokal.

Untuk mengatasi hal tersebut muncullah metaheuristik yang dimana menggabungkan konsep domain knowledge dan menghindari terjebak pada solusi optimal lokal.

Hyper heuristik adalah penggabungan metode heuristik(meta) yang digunakan untuk melengkapi kelemahan heuristik dengan heuristik lainnya[13]. Untuk menyempurnakan sebuah heuristik perlu dilakukan penggabungan dengan heuristik lainnya dengan maksud untuk mendapatkan hasil yang dapat mendekati optimal. Selain untuk mendapatkan hasil yang lebih optimal penggunaan *hyper* heuristik bertujuan untuk mengarah pada sistem yang lebih umum yang mampu menangani berbagai domain masalah daripada teknologi metaheuristik saat ini yang cenderung disesuaikan dengan masalah tertentu [13].

2.2.4. Algoritma Fuzzy Genetika

Algoritma *fuzzy* genetika adalah sebuah algoritma gabungan dari konsep fuzzy dan algoritma genetika dimana parameter yang digunakan dalam algoritma genetika didapatkan dari sistem fuzzy[9]. Tujuan dari penggabungan algoritma ini adalah untuk menggunakan pengontrol logika *fuzzy* yang masukannya merupakan kombinasi dari parameter kontrol awal algoritma genetika dan keluarannya adalah parameter kontrol yang digunakan dalam algoritma genetika. Parameter kontrol awal dari algoritma genetika dikirim ke *fuzzy logic controller*, yang menghitung nilai parameter kontrol baru yang akan digunakan oleh algoritma genetika[10].

Terdapat beberapa tahapan dalam proses algoritma fuzzy genetika yang sama dengan algoritma genetika yaitu[8]:

1. Representasi kromosom
2. Inisialisasi populasi
3. Fungsi evaluasi
4. Seleksi
5. Operator genetika meliputi probabilitas crossover dan mutasi
6. Penentuan parameter, yaitu parameter kontrol algoritma genetika, yaitu: ukuran populasi (popsize), peluang crossover (pc), dan peluang mutasi (pm). Dalam penentuan parameter ini dilakukan proses sistem fuzzy untuk mendapatkan nilai yang akan digunakan sebagai parameter.

Pada dasarnya konsep *fuzzy logic* mengandung 3 aturan dasar, yaitu *fuzzification, Interference system dan defuzzification*[10]. *Fuzzification* merupakan suatu proses untuk mengubah suatu masukan dari bentuk tegas menjadi *fuzzy* yang biasanya disajikan dalam bentuk himpunan-himpunan *fuzzy* dengan suatu fungsi keanggotaannya masing-masing.

Dalam tahapan ini dijelaskan terkait nilai domain dari setiap parameter yang digunakan dalam sistem *fuzzy*. *Interference system* merupakan acuan untuk menjelaskan hubungan antara variable-variabel masukan dan keluaran dimana variabel yang diproses dan dihasilkan berbentuk *fuzzy*. Untuk menjelaskan hubungan antara masukan dan keluaran biasanya menggunakan "*if-then*". Defuzzification merupakan proses pengubahan variabel berbentuk *fuzzy*

menjadi data-data pasti. Dalam fase defuzzification ini nilai keluaran dari interference system akan diubah kembali menjadi bentuk data pasti (nominal) yang dapat diproses oleh algoritma genetika.

Kode 2.2 menunjukkan psoudocode algoritma fuzzy genetika secara garis besar. Pada potongan kode tersebut ditunjukkan bahwa untuk menjalankan algoritma ini, perlu adanya aturan yang ditetapkan untuk algoritma fuzzy logic.

```
Start
Initialize population in P
Evaluate individuals in P
Initialize generasi in G
FuzzySystem input
    if (populasi is ...) and (generasi is ...)
    then (crossover is ...) and (mutasi is ...)
    ....
    ....
FuzzySystem ouput
while(condition=true){
Select a and b individuals from P
do crossover a and b to produce x
do mutasi in x
Evaluate individuals (P+x)
Select individuals P from (P+x) in P
}
End while
End Algoritm
```

Kode 2.1 Psoudocode *fuzzy*-genetika

2.2.5. Fuzzy Model XU

Dalam algoritma *fuzzy* sendiri, sebenarnya terdapat beberapa model yang digunakan dalam konsep optimasi.

Salah satu model yang digunakan adalah model Xu. Model Xu adalah model yang menggunakan sistem inferensi *fuzzy* Mamdani.

Karena logika fuzzy tidak hanya bernilai benar (1) atau salah (0) maka terdapat rumus khusus dalam melakukan kalkulasi tergantung grafik keanggotaan fuzzy. Salah satu grafik keanggotaannya adalah grafik S dimana rumusnya dapat dituliskan dengan:

Kurva S naik:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 2\left((x-\alpha)/(\gamma-\alpha)\right)^2 & \rightarrow \ \alpha \le x \le \beta \\ 1 - 2\left(\dfrac{\gamma - x}{\gamma - \alpha}\right)^2 & \rightarrow \ \beta \le x \le \gamma \end{cases}$$

Kurva S turun:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 1 - 2\left(\dfrac{\gamma - x}{\gamma - \alpha}\right)^2 & \rightarrow \ \alpha \le x \le \beta \\ 2\left((x-\alpha)/(\gamma-\alpha)\right)^2 & \rightarrow \ \beta \le x \le \gamma \end{cases}$$

Keterangan:

α = nilai keanggotaan nol (0)

β = nilai keanggotaan tengah (0,5)

γ = nilai keanggotaan lengkap (1)

x = nilai yang terdapat dalam keanggotaan.

Di dalam model Xu sendiri terdapat 2 buah masukan dan 2 buah keluaran[8]. 2 buah masukan tersebut yaitu jumlah populasi dan jumlah iterasi. Sedangkan untuk keluaran dari model Xu yaitu nilai probabilitas *crossover* dan nilai

probabilitas mutasi. Terdapat aturan yang ditetapkan dalam model Xu dimana melibatkan variabel masukan dan keluaran yang dapat dilihat pada Tabel 2.2 dan 2.3[9].

Tabel 2.3 Aturan yang ditetapkan untuk crossover

Probabilitas *crossover*		Ukuran populasi		
		Small	Medium	Large
Ukuran iterasi	Short	Medium	Small	Small
	Medium	Large	Large	Medium
	Long	Verylarge	Verylarge	Large

Tabel 2.4 Aturan yang ditetapkan untuk mutasi

Probabilitas mutasi		Ukuran populasi		
		Small	Medium	Large
Ukuran iterasi	Short	Large	Medium	Small
	Medium	Medium	Small	Verysmall
	Long	Small	Verysmall	Verysmall

Dari tabel 2.2 dan 2.3 didapatkan sembilan (9) aturan dalam menentukan nilai sistem *fuzzy* yaitu:

- *if (populasi is small) and (iterasi is short) then (crossover is medium) and (mutasi is large).*
- *if (populasi is medium) and (iterasi is short) then (crossover is small) and (mutasi is medium).*
- *if (populasi is large) and (iterasi is short)*

then (crossover is small) and (mutasi is small).

- *if (populasi is small) and (iterasi is medium)*

then (crossover is large) and (mutasi is medium).

- *if (populasi is medium) and (iterasi is medium)*

then (crossover is large) and (mutasi is small).

- *if (populasi is large) and (iterasi is medium)*

then (crossover is medium) and (mutasi is verysmall).

- *if (populasi is small) and (iterasi is long)*

then (crossover is verylarge) and (mutasi is small).

- *if (populasi is medium) and (iterasi is long)*

then (crossover is verylarge) and (mutasi is verysmall).

- *if (populasi is large) and (iterasi is long)*

then (crossover is large) and (mutasi is verysmall).

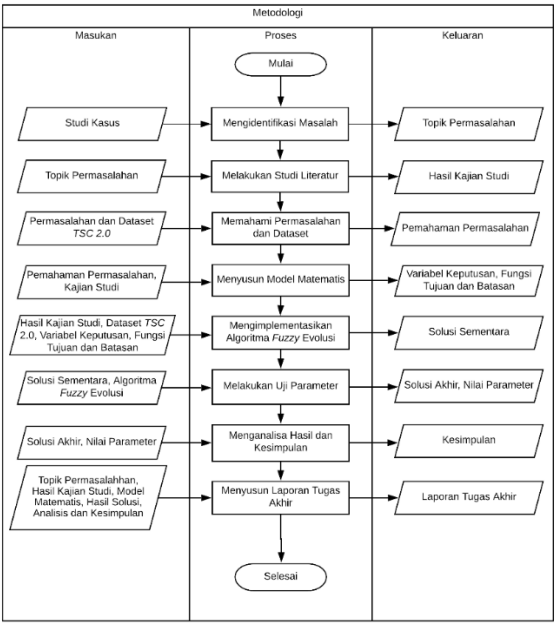
Dalam logika *fuzzy* model Xu terdapat dua (2) variabel masukan dan dua (2) variabel keluaran. Untuk mendapatkan hasil yang maksimal diperlukan nilai domain dari setiap variabel yang telah ditetapkan. Penentuan nilai domain dari setiap parameter sangat penting dalam proses ini. Nilai domain yang terlalu tinggi dan rendah akan mempengaruhi hasil yang didapatkan.

BAB 3
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai tahap-tahap pelaksanaan tugas akhir dan perencanaan jadwal dari pelaksanaan kegiatan.

3.1. Tahapan Pelaksanaan Tugas Akhir

Pada bab ini akan dijelaskan mengenai metodologi yang akan digunakan dalam mengerjakan tugas akhir ini. Metodologi ini akan dipakai sebagai dasar acuan dan langkah-langkah pengerjaan yang sistematis yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Pengerjaan Tugas Akhir

3.1.1. Identifikasi Masalah

Tahap pertama dalam menyelesaikan tugas ini adalah melakukan identifikasi masalah yang diangkat menjadi topik tugas akhir. Pada tahapan ini topik yang diangkat adalah *TSC* 2.0. Perlu dilakukan pemahaman lebih lanjut terkait *TSC* 2.0, yang bertujuan untuk memahami permasalahan yang terjadi dan apa yang ingin dicapai.

3.1.2. Studi Literatur

Pada tahap ini kedua melakukan studi literatur terkait permasalahan yang dihadapi. Dalam rangka membantu menyelesaikan permasalahan yang ada, maka perlu melakukan kajian terkait penelitian-penelitian yang sudah dilakukan sebelumnya. Penelitian sebelumnya diharapkan dapat memberikan masukan dan gambaran terkait penyelesaian permasalahan yang dihadapi, baik bersumber pada buku, paper, jurnal dan sumber bacaan lainnya. Pada tahap ini, adapun hasil yang didapatkan yaitu algoritma yang digunakan dalam menyelesaikan permasalahan *TSC* 2.0, adapun algoritma tersebut adalah *fuzzy* genetika.

3.1.3. Memahami Permasalahan dan Dataset

Setelah mendapatkan algoritma dan permasalahan yang ingin dihadapi, hal yang perlu dilakukan adalah memahami permasalahan tersebut. Dalam kasus ini permasalahan yang dirujuk adalah *TSC* 2.0. Pada tahapan ini secara garis besar akan didapatkan pemahaman permasalahan TSC 2.0. Secara garis besar akan didapatkan informasi terkait tujuan dari permasalahan *TSC* 2.0 dan hal-hal yang tidak boleh dilakukan dalam menyelesaikan permasalahan tersebut.

Keluaran dari proses ini akan menjadi dasar dalam menyusun model matematis.

3.1.4. Penyusunan Model Matematis

Setelah mendapatkan algoritma dan permasalahan yang ingin dihadapi, hal yang perlu dilakukan adalah memahami permasalahan tersebut. Dalam kasus ini permasalahan yang dirujuk adalah *TSC* 2.0. Pada tahapan ini secara garis besar akan didapatkan pemahaman permasalahan TSC 2.0. Secara garis besar akan didapatkan informasi terkait tujuan dari permasalahan *TSC* 2.0 dan hal-hal yang tidak boleh dilakukan dalam menyelesaikan permasalahan tersebut. Keluaran dari proses ini akan menjadi dasar dalam menyusun model matematis.

3.1.5. Implementasi Algoritma *Fuzzy* Genetika

Pada tahap implementasi yang pertama kali menetapkan nilai domain setiap parameter untuk sistem *fuzzy* yang akan dibuat. Setelah nilai domain dibuat barulah system dapat dibagun. Hal yang pertama dilakukan saat membangun sistem adalah menyesuaikan input pada sistem dengan data yang ada. Setelah input data terbangun, data perlu dipecah sesuai dengan kebutuhan dan disimpan di dalam *array*. Data yang terpecah inilah yang nantinya akan dikelola dengan algoritma *fuzzy* genetika untuk mendapatkan hasil yang terbaik. Nilai domain parameter dimasukkan dalam sistem fuzzy yang nantinya akan menjadi *range* masukan dan keluaran sistem *fuzzy*. Nilai keluaran sistem fuzzy akan menjadi masukan dalam algoritma genetika untuk mendapatkan hasil. Pada tahap ini kita perlu melakukan implementasi algoritma yang sudah dibuat kepada semua

(14) dataset *TSC* 2.0. Pada tahapan ini akan didapatkan solusi awal yang terbaik.

3.1.6. Pengujian Parameter

Dalam tahapan ini, yang perlu dilakukan adalah melakukan perubahan parameter yang diperlukan serta mencatat hasil dan perubahan parameter yang dilakukan. Perubahan parameter ini bertujuan untuk mendapatkan solusi yang terbaik. Pada tahap ini kita perlu melakukan pengujian parameter kepada semua (14) dataset *TSC* 2.0. Pada tahap ini akan didapatkan solusi terbaik saat melakukan uji parameter.

3.1.7. Analisis Hasil dan Kesimpulan

Pada tahap ini dilakukan Analisa terkait hasil yang telah didapatkan dari penelitian. Pada tahap ini didapatkan 3 buah hasil yaitu solusi awal terbaik, solusi terbaik saat perubahan parameter dan nilai parameter yang diubah. Dari hasil tersebut dilakukan analisis dan kesimpulan yang didapatkan.

3.1.8. Penyusunan Laporan Tugas Akhir

Pada tahapan ini dilakukan penyusunan laporan tugas akhir yang didapatkan dari semua dokumentasi, proses dan keluaran dari penelitian yang dilakukan. Hasil dari tahapan ini berupa Buku Laporan Tugas Akhir yang diharapkan mampu menyumbangkankan ilmu serta menjadi masukan dan saran untuk penelitian selanjutnya.

BAB 4
PERANCANGAN

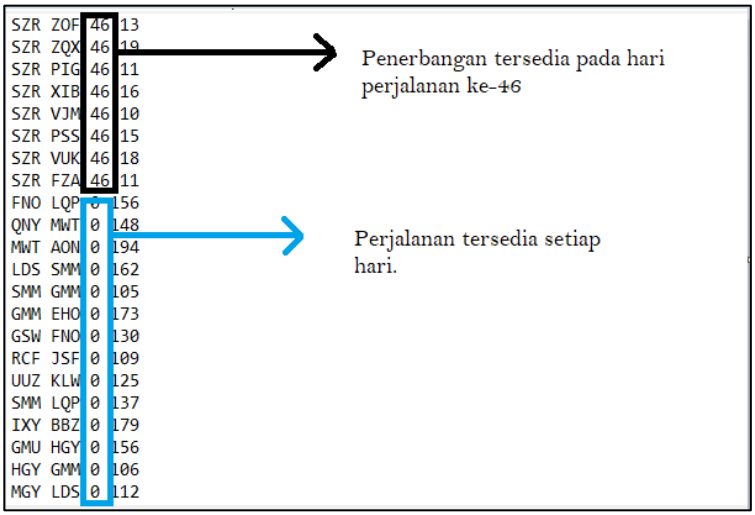
Pada bab ini akan dijelaskan terkait tentang pemahaman dataset yang akan digunakan, model matematis dari fungsi tujuan terhadap permasalahan yang dihadapi, dan batasan-batasan dalam perencanaan implementasi algoritma *fuzzy-genetic*.

4.1. Pemahaman Dataset

Pada bagian ini, berisikan pemahaman dataset khususnya terhadap rincian input untuk algoritma *fuzzy-genetic*. Terdapat 14 dataset yang akan digunakan untuk menguji algoritma yang digunakan. Dari 14 data tersebut terdapat 4 buah data yang digunakan untuk menguji parameter yang cocok. 4 Data tersebut adalah data 1, 6, 11 dan 14 dimana merupakan data pada tahap *development phase*.

Terdapat 2 jenis penulisan hari penerbangan pesawat dalam dataset yang digunakan sebagai input. Jenis yang pertama adalah bilangan asli {1, 2, …., x} yang menunjukkan bahwa pesawat terbang tersedia pada hari perjalanan ke-x. Jenis yang kedua adalah hari penerbangan pesawat dituliskan dengan bilangan 0 yang menunjukkan bahwa pesawat terbang tersedia pada semua hari perjalanan yang dapat dilihat pada Gambar 4.1.

Pada 14 dataset input terdapat kombinasi antara 2 jenis penulisan hari penerbangan pesawat. Jika dalam dataset input tersebut mengandung 2 jenis penulisan tersebut maka perlu dilakukan pengecekan sambungan rute yang bertuliskan hari selanjutnya dan 0 (h+1 dan 0).



Gambar 4.1 Gambar penulisan format hari

4.2. Formulasi Model Matematis Permasalahan

Untuk mempermudah dalam memahami lebih dalam permasalahan yang dihadapi, permasalahan perlu dikonversikan dalam bentuk model matematis umum menjadi khusus sehinga fungsi tujuan, variabel keputusan dan batasan menjadi jelas guna mempermudah dalam implementasi.

4.2.1 Fungsi Tujuan

Tujuan dari permasalahan *TSC 2.0* adalah untuk meminimalkan biaya yang dikeluarkan seorang wisatawan dalam mengunjungi setiap area yang ada dimana setiap area memiliki beberapa kota. Jika wisatawan tersebut sudah mengunjungi kota dalam sebuah area maka kota lain dalam area tersebut tidak boleh dikunjungi. Untuk model matematika fungsi tujuan dapat dilihat pada persamaan

$$\min \sum_{c=1}^{m} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} C_{ijc} X_{ijc} \quad \{4.1\}$$

Penjelasan:

C_{ijc} =biaya perjalanan dari kota i menuju j pada hari ke c

X_{ijc} = Variabel keputusan pada hari ke c

n = jumlah kota

m=jumlah hari

4.2.2 Variabel keputusan

Variabel keputusan sendiri adalah variabel yang digunakan dalam memenuhi fungsi tujuan. Dalam *TSC 2.0* variabel keputusannya adalah menemukan kota pada area mana saja yang dilalui oleh seorang wisatawan. Untuk model matematis dapat dilihat pada persamaan 4.2.

$$X_{ijc} \begin{cases} 1 & Berangkat\ dari\ kota\ i\ menuju\ j\ pada\ hari\ ke\ c \\ 0 & Sebaliknya \end{cases} \quad \{4.2\}$$

Penjelasan:

i = {1.....n} merupakan indeks kota keberangkatan
j = {1.....n} merupakan indeks kota tujuan
c= {1.....p} merupakan index hari perjalanan

4.2.3 Batasan Permasalahan

Permasalahan *TSC 2.0* memiliki beberapa batasan yang berbeda dengan permasalahan *TSP* biasanya. Untuk model matematis presentasi dari batasan dapat dilihat pada persamaan 4.3 sampai dengan 4.8

$$\sum_{c}^{m} \sum_{i=0, i \neq j}^{n} X_{ijc} = 1 \qquad i = 0, \dots, n \quad \{4.3\}$$

$$\sum_{c}^{m}\sum_{j=0,i\neq j}^{n} X_{ijc} = 1 \qquad j = 0,\dots,n \qquad \{4.4\}$$

$$\sum_{j=0}^{n}\sum_{i=0,i\neq j}^{n} X_{ijc} = 1 \qquad c = 1,\dots,m \qquad \{4.5\}$$

$$u_i \in \boldsymbol{Z} \qquad i = 0,\dots,n \qquad \{4.6\}$$

$$u_i - u_j + nX_{ijc} \leq n-1 \quad 1 \leq i \neq j \leq n \quad c = 1,\dots m \quad \{4.7\}$$

$$T_0 = T_m \qquad \{4.8\}$$

Penjelasan:

u_i= variabel buatan

u_j= variabel buatan

T_0= Area yang dikunjungi pada hari perjalanan ke-0

T_m= Area yang dikunjungi pada hari perjalanan ke-m

Persamaan 4.3 dan 4.4 membatasi bahwa hanya ada 1 kota yang dikunjungi dalam sebuah area. Persamaan 4.5 merujuk bahwa dalam 1 hari hanya terdapat 1 buah perjalanan dari kota i ke kota j pada hari c. Persamaan 4.6 dan 4.7 adalah persamaan yang memastikan bahwa tidak ada subrute. Sedangkan untuk persamaan 4.8 merujuk bahwa rute harus berawal dan berakhir pada area yang sama.

4.3. Rancangan Algoritma *Fuzzy*-Genetika

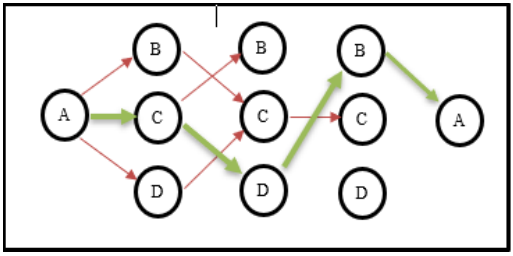
Pada tahapan ini akan dijelaskan terkait rancangan tahapan implementasi algoritma *fuzzy* genetika.

4.3.1 Solusi Awal

Dalam 14 dataset input memiliki permasalahan yang berbeda-beda, seperti tidak ditemukannya kelanjutan dari

sebuah rute yang ada seperti gambar 4.2. Sehingga dataset tersebut perlu diolah untuk menghapus rute yang menimbulkan *error* atau tidak menghasilkan solusi. Untuk mengatasi permasalahan tersebut digunakan bantuan graph yang mana nantinya akan membantu dalam menemukan solusi dan penguraian data secara cepat.

Graph akan dibuat dalam array yang berdimensi 2 dimensi dimana berisikan *node* dan *edge*. *Node* sendiri adalah kota yang tersedia pada perjalanan hari-n, sedangkan *edge* merujuk pada sambungan rute atau jadwal perjalanan kota tersebut. Setelah semua rute disimpan dalam berntuk *graph* selanjutnya dibuatlah solusi awal dari *graph* yang sudah dibuat. Selanjutnya solusi tersebut di optimasi menggunakan algoritma *fuzzy*-genetika.



Gambar 4.2 Ilustrasi perjalanan dari tiap kota

4.3.2 Model *Fuzzy*-genetika

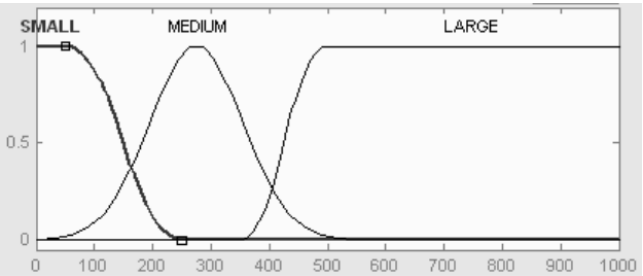
Dalam penyelesaian permasalahan ini digunakan model *fuzzy* yang sudah ada, sehingga tidak perlu melakukan pemodelan *fuzzy*. Dalam permasalahan ini, logika *fuzzy* berfungsi untuk mengatur parameter yang ada dalam algoritma genetika. Dalam penyelesaian permasalahan ini digunakan model *fuzzy* Mamdani dimana memiliki 2 buah

input yaitu populasi dan iterasi, sedangkan 2 output yaitu probabilitas mutasi dan probabilitas *crossover*. Untuk fungsi sistem *fuzzy* sudah dijelaskan pada bab 2. Tabel 4.1 adalah semesta keanggotaan *fuzzy* dalam algoritma ini.

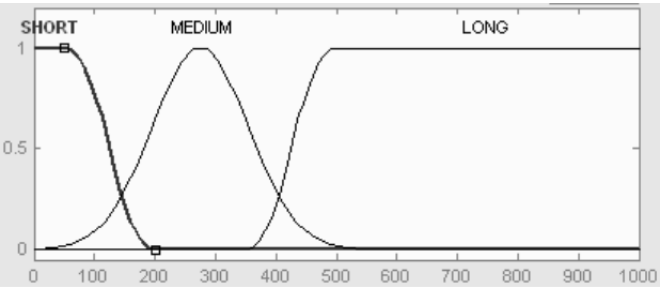
Tabel 4.1 Keanggotaan fuzzy parameter input dan output

Jenis	Parameter	Kategori	Anggota
Input	Populasi	Small	[50 250]
		Medium	[80 275]
		Large	[350 500]
	Iterasi	Short	[50 200]
		Medium	[80 275]
		Long	[350 500]
Output	Probmutasi	Very small	[0.02 0.08]
		Small	[0.08 0.14]
		Medium	[0.14 0.2]
	Probcrossover	Large	[0.2 0.25]
		Small	[0.6 0.7]
		Medium	[0.7 0.75]
		Large	[0.75 0.8]
		Very Large	[0.8 0.9]

Gambar 4.3 dan 4.4 menunjukkan grafik keanggotaan fuzzy untuk populasi dan grafik keanggotaan fuzzy untuk iterasi. Dalam grfaik keanggotaan tersebut menggambarkan bahwa kurva keanggotaan yang dipakai adalah kurva S naik dan turun.



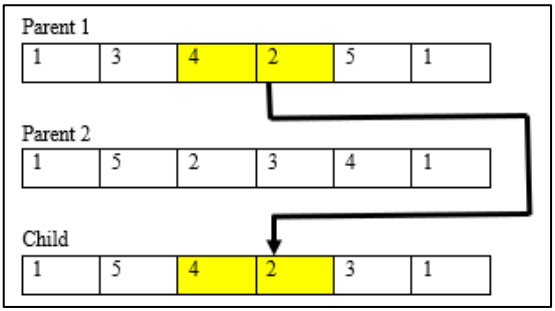
Gambar 4.3 Kurva keanggotaan populasi



Gambar 4.4 kurva keanggotaan iterasi

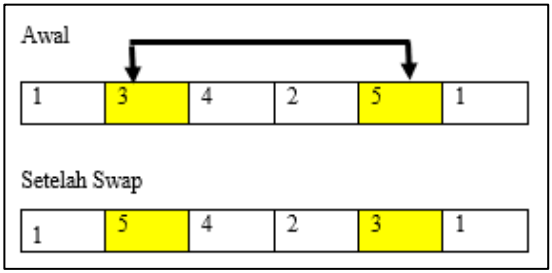
Setelah model didapatkan barulah algoritma genetika berjalan. Dalam setiap iterasi akn dipilih 2 solusi dari banyaknya populasi, yang nantinya akan menjadi induk. Setelah dipilih 2 anggota populasi, akan dihasilkan nilai random dari keanggotaan probabilitas kawin silang (*crossover*). Jika nilai memenuhi syarat maka akan dilakukan crossover. Untuk melakukan *crossover*, hal yang perlu dilakukan adalah mendapatkan sambungan genetika dari induk pertama. Posisi sambungan genetika tersebut didapatkan secara random. Setelah itu sambungan genetika dimasukkan ke dalam induk kedua dan

menghasilkan individu baru. Untuk proses perkawianan silang dapat dilihat pada Gambar 4.3.



Gambar 4.5 Ilustrasi kawin silang algoritma genetika

Dari individu yang tercipta dilakukan proses mutasi berdasarkan probabilitas yang dihasilkan secara random. Gambar 4.4 akan menunjukkan proses mutasi pada algoritma *fuzzy*-genetika, dimana proses mutasi yang dipilih adalah *swap*. Swap sendiri artinya menukar, sehingga mutasi yang dilakukan adalah menukar dua posisi kota. Setelah proses perkawinan silang dan mutasi selesai dilakukan pengecekan bahwa rute tersebut benar-benar ada.



Gambar 4.6 Ilustrasi fungsi mutasi algoritma genetika

Setelah dicek bahwa rute yang tercipta adalah rute yang memiliki penerbangan sesuai selanjutnya adalah membandingkan individu baru dengan populasi yang ada saat ini. Jika nilai fitness populasi lebih besar maka akan digantikan dengan individu yang baru tersebut.

4.3.3 Uji Parameter

Dari 14 dataset yang ada, dilakukan pengujian parameter guna mendapatkan parameter yang terbaik. Untuk pengujian parameter ini tidak semua data digunakan tetapi hanya data 1, 6, 11 dan 14. Pemilihan dataset ini dikarenakan dataset tersebut adalah dataset yang digunakan dalam fase *development*. Setelah uji parameter dilakukan maka akan dipilih parameter terbaik dari penilaian solusi yang dikeluarkan tiap parameter.

Untuk menilai solusi tersebut dijumlahkan nilai solusi terbaik dan rata-rata dari setiap dataset pada tiap parameter. Akumulasi solusi terkecil dari parameter akan dipilih sebagai pengujian berikutnya.

Dalam pengujian parameter ini terdapat 2 parameter input dan parameter output. Adapun paramener input

tersebut adalah populasi dan iterasi. 2 parameter inilah yang nantinya akan dicari nilai yang cocok dalam *tunning* parameter. Dalam tunning parameter ini, nilai parameter didapatkan dari range nilai keanggotaan fuzzy. Dalam penelitian ini keanggotaan fuzzy untuk populasi dan iterasi ada 3. Dari tiga golongan tersebut dipilih 1 buah nilai dari setiap golongan untuk dijadikan parameter. Skenario tersebut dapat dilihat pada Tabel

Tabel 4.2 Skenario yang dilakukan

Skenario	Populasi	Iterasi
A	50	50
B	50	200
C	50	400
D	200	50
E	200	200
F	200	400
G	400	50
H	400	200
I	400	400

Skenario tersebut diujikan kepada 4 buah dataset yaitu 1, 6, 11 dan 14. Pemilihan dataset tersebut dikarenakan data tersebut adalah data uji saat *development phase*. Dalam pengujian ini dataset dijalankan sebanyak 10 kali untuk memastikan bahwa hasil dapat dipercaya.

BAB 5
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi pembuatan solusi yang layak dengan menggunakan algoritma *fuzzy*-genetika yang menggunakan Bahasa pemrograman *java*.

5.1. Pembentukan Solusi

Dalam tahapan ini akan dijelaskan mengenai pembuatan solusi awal dari dataset input. Adapun beberapa aktivitas yang penting dalam subbab ini, yaitu membaca dan menyimpan dataset, membuat *graph*, melakukan optimasi *graph*, membuat rute awal dan melakukan pengecekan tour yang dibuat apakah sudah cocok atau belum

5.1.1. Membaca dan Menyimpan Dataset

Dataset input perlu dibaca dan disimpan agar dapat dengan mudah diolah. Untuk membaca dataset ini digunakan fungsi *BufferedReader*. *BufferedReader* akan membaca *file* yang sudah diatur lokasinya lalu menyimpannya dalam bentuk data string. Data berbentuk string tersebut selanjutnya dipisahkan berdasarkan golongan. Terdapat 2 golongan yaitu area-kota dan penerbangan. Untuk Kode dalam memecah dataset input dapat dilihat pada Kode 5.1.

```
public static void input() throws FileNotFoundException, IOException {
    String dataset = "E:\\TA\\FuzzyGenetic\\Data\\" + dataPenerbangan + ".in";
    BufferedReader reader = new BufferedReader(new FileReader(dataset));
    String baris1[] = (reader.readLine()).split(" ");
    jumlahArea = Integer.parseInt(baris1[0]);
    kotaAsal = baris1[1];
    namaArea = new String[jumlahArea];
    detailArea = new int[jumlahArea][2];
    String kota = "";
    day = new Cost[jumlahArea + 1];
    int index = -1;
    for (int i = 0; i < (jumlahArea * 2); i++) {
        if (i % 2 == 0) {
            namaArea[i / 2] = reader.readLine();
        } else {
            baris1 = (reader.readLine()).split(" ");
            int j;
            for (j = 0; j < baris1.length; j++) {
                kota = kota + (namaArea[(i - 1) / 2] + "." + baris1[j]) + "&";
                if (baris1[j].equals(kotaAsal)) {
                    areaAsal = namaArea[(i - 1) / 2];
                }
            }
            index = index + j;
            detailArea[(i - 1) / 2][1] = j;
            detailArea[(i - 1) / 2][0] = index;
        }
    }
```

Kode 5.1 Kode membaca dataset input

Untuk area dan kota disimpan bersama terlebih dahulu. Sedangkan untuk penerbangan akan dimasukkan kedalam wadah yang nantinya disimpan dengan array 2 dimensi yaitu kota keberangkatan dan kota tujuan dengan memanggil indeks kota yang sudah ada. Selanjutnya biaya perjalanan dari kota keberangkatan dan kota tujuan disimpan kedalam array dimana nantinya jika ingin digunakan hanya perlu memanggil indeks kota keberangkatan dan kota tujuan. Setelah dataset input dipecah selanjutnya adalah menyimpan kota awal dan area akhir dimana nantinya digunakan untuk membuat graph didalam class node dan cost yang dapat dilihat pada Kode 5.2.

```
while (true) {
    line = reader.readLine();
    if (line == null) {
        break;
    } else {
        line2 = line.split(" ");
        int hari = Integer.parseInt(line2[2]);
        if (day[hari] == null) {
            day[hari] = new Cost(Kota.length);
        }
        if (bahanGraph[hari] == null) {
            bahanGraph[hari] = new ArrayList<>();
        }
        if (!line2[0].equals(kota1)) {
            Awal = cariIndex(line2[0]);
        }
        if (!line2[1].equals(kota2)) {
            Tujuan = cariIndex(line2[1]);
        }
        kota1 = line2[0];
        kota2 = line2[1];
        Integer bahan[] = {Awal, Tujuan};
        bahanGraph[hari].add(bahan);
        if (hari == 0) {
            indexHari0++;
        }
        day[hari].add(Awal, Tujuan, Integer.parseInt(line2[3]));

    }
```

Kode 5.2 Kode memecah jadwal penerbangan

5.1.2. Node

Dalam class node berisikan tentang kumpulan kota keberangkatan dan kota tujuan yang berbentuk indeks yang nantinya dibuat rute. Dalam class *node* ini memiliki beberapa fungsi yaitu menambahkan, mengecek kota berikutnya dan mengganti isi kota berikutnya dalam *graph* dan membuat rute awal. Untuk potongan kode kelas node dapat dilihat pada Kode 5.3.

```java
public class Node {
    int indexKota;
    int indexArea;
    List<Node> kotaBerikutnya;
    public Node(int a, int b) {
        indexKota = a;
        indexArea = b;
        kotaBerikutnya = new ArrayList<>();
    }
    public boolean checkKotaBerikutnya(int a) {
        for (int i = 0; i < kotaBerikutnya.size(); i++) {
            if (kotaBerikutnya.get(i).indexKota == a) {
                return true;
            }
        }
        return false;
    }
```

Kode 5.3 Kode class node

5.1.3. Membuat *Graph*

Dari hasil membaca *file* dan menyimpan data yang diperlukan, langkah selanjutnya adalah menyambungkan kota-kota menjadi bandara kedatangan dengan bandara keberangkatan dengan menggunakan *graph* dimana disini berbentuk *arraylist* di dalam array. Banyaknya *array* yang ada disini sama dengan durasi hari perjalanan seorang wisatawan. *Array* akan berindeks hari dan arraylist berisikan rute pada hari tersebut. Hal ini berguna dalam mempermudah mengacak solusi agar *feasible* atau rute tersebut benar-benar ada.

Dalam pembuatan graph ini terdapat beberapa elemen yang penting yaitu kota keberangkatan dan kedatangan. semua kota yang ada di *graph* akan disambungkan berdasarkan jadwal penerbangan yang sudah ada. Untuk

potongan kode membuat *graph* dapat dilihat pada Kode 5.4.

```
for (int i = 1; i < graph.length - 1; i++) {
    graph[i] = new ArrayList<Node>();
    if (bahanGraph[0] != null) {
        for (int k = 0; k < kotaAkhir[0].size(); k++) {
            boolean test = true;
            for (int j = 0; j < graph[i].size(); j++) {//check apakah di graph sudah ada?
                if (kotaAkhir[0].get(k) == graph[i].get(j).indexKota) {
                    test = false;
                    break;
                }
```

Kode 5.4 kode dalam membuat graph

Semua kota yang dimasukkan ke dalam graph ini, karena terdapat beberapa kasus dimana kota tidak memiliki sambungang sebelumnya atau sesudahnya. Kota keberangkatan akan dicek apakah memiliki sambungan sebelumnya. Pengecekan dilakukan dengan jadwal hari sebelumnya yang berindeks i-1 dan 0. Karena dalam kasus *TSC* ini terdapat 2 jenis penulisan jadwal keberangkatan yaitu i dan 0. Sedangkan untuk kota kedatangan akan dicek dengan jadwal pada hari sesudahnya yang berindeks i+1 dan 0. Semua jadwal yang lolos pengecekan akan dimasukkan dalam *graph.* Untuk potongan kode mengecek kelanjutan kota dapat dilihat pada Kode 5.5.

```
for (int k = 0; k < bahanGraph[pengganti].size(); k++) {
    //bandingkan dengan kemarin
    boolean test3 = true;
    for (int j = 0; j < graph[i].size(); j++) {
        //check apakah di graph sudah ada?
        if (graph[i].get(j).indexKota == bahanGraph[pengganti].get(k)[1]) {
            test3 = false;
            break;
        }
    }
    if (test3) {
        for (int j = 0; j < graph[i - 1].size(); j++) {
            boolean t = false;
            if (kotaAwal[i] != null) {
                if (kotaAwal[i].contains(graph[i - 1].get(j).indexKota)) {
                    t = true;
                }
            }
            if (kotaAwal[0] != null) {
                if (kotaAwal[0].contains(graph[i - 1].get(j).indexKota)) {
                    t = true;
                }
            }
```

Kode 5.5 Kode mengecek kelanjutan penerbangan dalam graph

5.1.4. Optimasi Graph

Untuk melakukan pengecekan ulang bahwa rute yang disimpan benar-benar memiliki sambungannya, maka dilakukan pengecekan ulang dari banyaknya panjang *graph*-2 hingga *graph* pertama. Jika ternyata ditemukan bahwa terdapat kota yang tidak memiliki rute selanjutnya maka akan dihapus dari *graph* yang ada saat ini. Untuk melihat potongan kode mengecek rute-rute dalam optimasi graph dapat dilihat pada Kode 5.6.

```
for (int i = graph.length - 2; i > 0; i--) {
    for (int j = 0; j < graph[i].size(); j++) {
        boolean cek = true;
        for (int k = 0; k < graph[i + 1].size(); k++) {
            if (day[i + 1] != null) {
                if (day[i + 1].check(graph[i].get(j).indexKota, graph[i + 1].get(k).indexKota)) {
                    cek = false;
                    break;
                }
            }
```

Kode 5.6 Kode mengecek rute dalam graph

5.1.5. Pembuatan Tour Awal

Langkah pertama yang dilakukan untuk membuat tour adalah melakukan inisiasi dimanakah tour akan dimulai

dan berakhir. Setelah kota di inisiasi maka melakukan pengisian array tour dengan melakukan random rute yang ada di *graph* dimana kota dalam *graph* tersebut memiliki kota yang berkesinambungan dengan tour yang sudah terisi, dapat dilihat pada Kode 5.7.

```
public void makeTour(int aa) {
    int tourArea[] = new int[graph.length];
    int index = 0;
    Random r = new Random();
    tourKota[0] = graph[0].get(0).indexKota;
    tourArea[0] = graph[0].get(0).indexArea;
    int hasil = r.nextInt(graph[graph.length - 2].size());
    Node b = graph[graph.length - 2].get(hasil);
    tourKota[graph.length - 2] = b.indexKota;
    tourArea[graph.length - 2] = b.indexArea;
    tourArea[graph.length - 1] = tourArea[0];
    for (int i = 1; i < graph.length - 2; i++) {
        Node a = graph[i - 1].get(index).getRandom(tourKota, i, tourArea, aa);
        for (int j = 0; j < graph[i].size(); j++) {
            if (graph[i].get(j).indexKota == a.indexKota) {
                index = j;
                break;
            }
        }
```

Kode 5.7 Kode membuat tour baru

5.1.6. Menghitung Biaya

Untuk menghitung biaya yang dikeluarkan dalam perjalanan ini dihitung dengan menggunakan fungsi *calculateFitness().* Fungsi ini akan mengecek tour dari awal hingga akhir. Jika ternyata terdapat *cost* yang tidak ditemukan maka solusi tersebut dianggap tidak baik sehingga fitness di set menjadi -1. Untuk potongan Kode dalam menghitung biaya dapat dilihat pada Kode 5.8.

```
public void calculateFitness() {
    fitness = 0;
    try {
        for (int i = 0; i < tourKota.length - 1; i++) {
            int cost1 = -1;
            int cost2 = -1;
            if (day[i + 1] != null) {
                cost1 = day[i + 1].get(tourKota[i], tourKota[i + 1]);
            }
            if (day[0] != null) {
                cost2 = day[0].get(tourKota[i], tourKota[i + 1]);
            }
            if (cost1 > 0 && cost2 > 0) {
                if (cost1 < cost2) {
                    fitness = fitness + cost1;
                } else {
                    fitness = fitness + cost2;
                }
            } else if (cost1 > 0) {
                fitness = fitness + cost1;
            } else if (cost2 > 0) {
                fitness = fitness + cost2;
            } else {
                fitness = -1;
                break;
            }
        }
```

Kode 5.8 Kode untuk menghitung biaya perjalanan

5.1.7. Menyimpan Solusi dalam bentuk file

Dalam memudahkan melakukan pengecekan dalam pengerjaan tugas akhir, solusi dari hasil *running* kode program disimpan dalam bentuk *file* berformat.csv. Dalam kode tersebut digunakan *class PrintWritter* dalam melakukan penyimpanan kedalam file tersebut yang dapat dilihat pada Kode 5.9.

```
PrintWriter pw = null;
try {
    pw = new PrintWriter(new File("E:\\TA\\FuzzyGenetic\\Data\\" + dataPenerbangan + ".csv"));
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
StringBuilder builder = new StringBuilder();
```

Kode 5.9 Pembuatan file penyimpanan output

Dalam file tersebut tersimpan informasi berupa biaya rute awal, rute perjalanan awal, biaya rute algoritma optimasi, rute perjalanan algoritma optimasi dan waktu dalam melakukan optimasi. Untuk potongan kode dalam menyimpan solusi kedalam file dapat dilihat pada Kode 5.10.

```
System.out.println(dataPenerbangan+" Fuzzy");
builder.append(fittest + "\n");
builder.append(Pop[index].getTour()+ "\n");
builder.append(fitawal+ "\n");
builder.append(awal+ "\n");
builder.append(waktu+ "\n");
pw.write(builder.toString()+ "\n");
pw.close();
```

Kode 5.10 Penambahan isi dari file *output*

5.2. Algoritma Fuzzy-Genetika

Pada subbab ini akan dijelaskan mengenai implementasi algoritma yang dipakai untuk menyelesaikan permasalahan *TSC 2.0* yaitu *fuzzy* genetika. Pada subbab ini penjelasan akan dibagi menjadi 2 bagian yaitu *fuzzy* dan algoritma genetika.

5.2.1. Fuzzy Logic

Dalam implementasi algoritma ini, *fuzzy* genetika berperan sebagai pengatur parameter probabilitas kawin silang dan probabilitas mutasi. *Output* dari *fuzzy* ini sendiri didapatkan dari input banyaknya populasi dan populasi yang diberikan. Populasi dan iterasi akan dicek dan digolongkan berdasarkan rumus keanggotaan sistem *fuzzy*. Untuk mengecek keanggotaan fuzzy dari populasi dapat dilihat pada Kode 5.11.

```
public static void cekpop(int p){
    if(p<=250){
        if(p<=50){
            Populasi[0]=1;
        }
        if(150>=p && p>50){
            Populasi[0]=1-2*((p-50)/(250-50)^2);
        }
        if(250>p && p>150){
            Populasi[0]=2*((250-p)/(250-50)^2);
        }
    }
```

Kode 5.11 Kode menemukan posisi dalam keanggotaan fuzzy

Setelah nilai fuzzy dari populasi atau iterasi didapatkan maka akan digolongkan berdasarkan nilai yang didapatkan. Terdapat 3 nilai yang disimpan di dalam array, isi [0] mempresentasikan anggota *SMALL/SHORT*, isi [1] mempresentasikan *MEDIUM* dan isi [2] mempresentasikan *LARGE/LONG*. Dari ketiga *array* isi tersebut dibandingkan mana yang paling besar dan diambil untuk melakukan penggolongan. Untuk menggolongkan nilai dari keanggotaan fuzzy dapat dilihat pada Kode 5.12.

```
    if(index==0){
    pop="SMALL";
    }else if(index==1){
    pop="MEDIUM";
    }else if(index==2){
    pop="LARGE";
    }
}
```

Kode 5.12 Kode dalam defuzzifikasi

Langkah selanjutnya adalah menentukan *output* dari system yang dimana telah didapatkan dari studi literatur sebelumnya. *Output* inilah yang nantinya akan melakukan inisiasi parameter probabilitas *crossover* dan probabilitas mutasi.

5.2.2. Genetika Algoritm

Setelah didapatkan parameter probabilitas *crossover* dan probabilitas mutasi maka algoritma genetika baru dapat berjalan. Langkah pertama adalah membuat rute yang telah di*input* pertama kali atau yang disebut populasi. Dari populasi tersebut akan dilakukan perkawinan silang dan mengasilkan rute baru yang nantinya akan dimutas. Untuk potongan kode algoritma genetika dapat dilihat pada Kode 5.13.

```
for (int i = 1; i < Pop.length; i++) {
    Pop[i] = new Tour(graph, Kota, day, false);
    if (Pop[i].getFitness() < 0) {
        Random r = new Random();
        int[] s = Pop[r.nextInt(i)].tourKota;
        Pop[i].setTour(s);
        int fi = Pop[i].getFitness();
        for (int j = 0; j < 50; j++) {
            random(i, true);
            if (fi != Pop[i].getFitness()) {
                break;
            }
        }
    }
    fitness[i] = Pop[i].getFitness();
    listSolusi[i] = Pop[i].getTour();
}
```

Kode 5.13 Kode dalam membuat populasi *GA*

Selanjutnya akan dilakukan iterasi untuk melakukan perkawinan silang. Dalam perkawinan silang ini akan dipilih 2 individu secara random dari 10 populasi yang terbaik. Jika nilai dari probabilitas kawin silang lebih besar dari 0.7 maka akan menjalankan fungsi *crossover*. Saat perintah *crossover* berjalan akan dipilih range indeks array dari individu pertama. Range indeks array di buat secara random dan disimpan didalam array baru yang ukurannya

sama dengan tour. Sisa *array* yang masih kosong selanjutnya diisi dengan individu kedua. Untuk potongan kode dalam melakukan perkawinan silang dapat dilihat pada Kode 5.14.

```
int tourP1[] = a.tourKota;
int tourP2[] = b.tourKota;
if (startPos > endPos) {
    int aa = startPos;
    startPos = endPos;
    endPos = aa;
}
if (endPos <= a.tourKota.length-1) {
    endPos--;
}
for (int i = startPos; i <= endPos; i++) {
    tourBaru[i] = tourP1[i];
}
int in = 0;
for (int i = 0; i < a.tourKota.length - 1; i++) {

    boolean cek = true;

    for (int j = startPos; j <= endPos; j++) {
        if (tourP2[i] == tourBaru[j]) {
            cek = false;
            break;
        }
    }
    if(in>tourBaru.length-1){
                    break;
                }
```

Kode 5.14 Kode kawin silang *GA*

Setelah terbentuk individu baru, individu tersebut akan dimutasi memenuhi kriteria yang berlaku. Dalam penelitian ini digunakan fungsi *swap* untuk menukar dua buah kota. Pemilihan kota mana yang akan ditukar dilakukan dengan menggunakan random. Setelah ditukar maka akan dicek rute tersebut ada atau tidak dengan menggunakan fungsi *calculate fitness*. Untuk potongan kode dalam melakukan mutasi dapat dilihat pada Kode 5.15.

```
public void swap(int a, int b, boolean s) {
    int c1=0,c2=0,c3=0,c4=0;
    boolean test=true;
    c1=getCost(a-1,a, a);
    c2=getCost(a,a+1, a+1);
    c3=getCost(b-1,b, b);
    c4=getCost(b,b+1, b+1);
    if(c1<0)test=false;
    if(c2<0)test=false;
    if(c3<0)test=false;
    if(c4<0)test=false;
    int costAwal=c1+c2+c3+c4;
    int c11=getCost(a-1,b, a);
    int c21=getCost(b,a+1, a+1);
    int c31=getCost(b-1,a, b);
    int c41=getCost(a,b+1, b+1);
    if(c11<0)test=false;
    if(c21<0)test=false;
    if(c31<0)test=false;
    if(c41<0)test=false;
    int costAkhir=c11+c21+c31+c41;
    if((costAwal>costAkhir || !s )&& test){
        int c = tourKota[a];
        tourKota[a] = tourKota[b];
        tourKota[b] = c;
        fitness=fitness+(costAkhir-costAwal);

    }
    calculateFitness();
}
```

Kode 5.15 Kode untuk mutasi algoritma *GA*

5.3. Mengecek *Output*

Untuk memastikan bahwa semua batasan sudah dipenuhi dalam solusi yang telah dibuat, pengecekan dilakukan dengan kode yang telah dibuat menggunakan Bahasa pemrograman java. Kode tersebut membaca file *output* yang berisikan solusi yang telah terbuat. Kode tersebut akan membaca *file* berformat .csv dan mengecek apakah terdapat area yang sama, kota yang sama dan biaya yang dikeluarkan sudah cocok atau belum.

```
for(int i=0;i<a.length-1;i++){
    String[] b=a[i].split("\\.");
    if(!area.contains(b[0]))area.add(b[0]);
    else{
        System.out.println("area double");
        cek=false;
        break;
    }
    if(!kota.contains(b[1]))kota.add(b[1]);
    else{
        System.out.println("kota double");
        cek=false;
        break;
    }
    for(int k=0;k<areaKota.length;k++){
        if(("("+areaKota[k][0]).equals(b[0])){
            String c[]=areaKota[k][1].split(" ");
            cek=false;
```

Kode 5.16 Mengecek area dan kota dalam file output.

BAB 6
HASIL DAN PEMBAHASAN

Pada bab ini berisikan tentang hasil pengujian parameter, solusi semua dataset dari parameter yang terpilih dengan algoritma *fuzzy*-genetika dan genetika, serta analisa perbandingan antara algoritma *fuzzy*-genetika dan genetika.

6.1. Data Uji Coba

Data yang digunakan sebagai data uji coba adalah semua data (14 data) yang digunakan dalam kompetisi *TSC* 2.0. Untuk melakukan melakukan *tunning* parameter digunakan data 1, 6, 11 dan 14 yang mana merupakan data yang digunakan saat kompetisi *TSC* saat fase *developmet*.

6.2. Lingkungan Uji Coba

Dalam membantu melakukan implementasi dan pengujian penelitian ini, terdapat beberapa perangkat keras dan perangkat lunak yang digunakan. Adapun spesifikasi perangkat keras tersebut dapat ditunjukkan pada Tabel 6.1.

Tabel 6.1 Spesifikasi perangkat keras

Perangkat Keras	Spesifikasi
Jenis	Laptop ASUS seri X454Y
Processor	AMD Quad Core A8-7410 up to 2.5 GHz
RAM	8 GB
Hard Disk Drive	500 GB

Sedangkan Untuk spesifikasi perangkat lunak yang digunakan dalam pengerjaan penelitian tugas akhir ditunjukkan pada Tabel 6.2.

Tabel 6.2 Spesifikasi perangkat lunak

Perangkat Lunak	Fungsi
Windows 10 64 bit	Sistem Operasi
Netbeans 8.2	Implementasi dan Penjalan Algoritma
Microsoft Excel 2019	Pengolahan Hasil Uji Coba
Microsoft Word 2019	Penulisan Laporan

6.3. Hasil Solusi Awal

Dalam menemukan solusi awal data masukan perlu diolah agar mempermudah dalam melakukan proses implementasi. Pertama kali yang dilakukan adalah menyimpan data masukan kedalam sebuah wadah. Pada penelitian ini data masukan dimasukkan dalam sebuah array dua buah dimensi yang mana berisikan *node* (kota) dan *edge* (rute/jadwal). Dari array tersebut kemudian diacak dan menghasilkan solusi yang memenuhi batasan yang ada. Jika solusi tidak memenuhi batasan yang ada maka akan diulang lagi hingga didapatkan solusi yang layak. Berikut ini adalah contoh *output* dari inisial solusi program yang dikembangkan.

Tabel 6.3 Keluaran dari solusi awal

Dataset	Solusi
1	Cost : 12686 (Zona_0.AB0)-(Zona_5.AB5)-(Zona_8.AB8)-(Zona_1.AB1)- (Zona_3.AB3)-(Zona_7.AB7)-(Zona_4.AB4)-(Zona_9.AB9)- (Zona_6.AB6)-(Zona_2.AB2)-(Zona_0.AB0)
2	Cost : 1498 (Area_0.EBJ)-(Area_3.NBP)-(Area_7.OMG)-(Area_4.NCA)- (Area_5.NUJ)-(Area_6.OHT)-(Area_2.GSM)-(Area_1.EFZ)- (Area_8.QKK)-(Area_9.SSC)-(Area_0.TKT)

6.4. Hasil Uji Parameter Algoritma Fuzzy-genetika

Untuk mendapatkan solusi yang terbaik, pengujian parameter dilakukan dengan menggunakan 9 skenario yang berbeda. Pengujian parameter ini mengubah nilai populasi dan iterasi yang terdapat di dalam algoritma *fuzzy*-genetika.

6.4.1 Skenario A

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=50 dan iterasi 50. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.5.

Tabel 6.4 Hasil Skenario A

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1782	7333	3966.6
6	11929	12365	12150.4
11	91729	95767	93837.8
14	216292	221861	219245.2

6.4.2 Skenario B

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=50 dan iterasi 200. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.6.

Tabel 6.5 Hasil Skenario B

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1688	5790	2577.2
6	12078	12369	12216.5

11	86909	94988	92853.7
14	214420	223576	218822.7

6.4.3 Skenario C

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=50 dan iterasi 400. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.7.

Tabel 6.6 Hasil Skenario C

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1493	2047	1849.3
6	11845	12349	12165.1
11	86909	94988	92853.7
14	214814	222344	219095.8

6.4.4 Skenario D

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=200 dan iterasi 50. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.8

Tabel 6.7 Hasil Skenario D

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1663	4733	2122.4
6	11692	12085	11880.4
11	88122	93811	91248.3
14	211853	219973	217193.9

6.4.5 Skenario E

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=200 dan iterasi 200. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.9

Tabel 6.8 Hasil Skenario E

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1697	4539	2162.3
6	11744	12087	11893.4
11	88730	94139	91482
14	21435	220151	217278.5

6.4.6 Skenario F

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=200 dan iterasi 400. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.10

Tabel 6.9 Hasil Skenario F

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1553	4366	2024.8
6	11828	12176	11935
11	89360	92641	91168.4
14	214629	219002	217235.7

6.4.7 Skenario G

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=400 dan iterasi=50.

Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.11

Tabel 6.10 Hasil Skenario G

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1526	2039	1774.2
6	11427	12019	11793.3
11	88223	91917	90372.9
14	213961	219276	216405.8

6.4.8 Skenario H

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=400 dan iterasi=200. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.12

Tabel 6.11 Hasil Skenario H

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1662	2143	1818.7
6	11566	12077	11823.6
11	88268	92373	90849.7
14	213065	218080	215406.9

6.4.9 Skenario I

Dalam pengujian ini dataset 1, 6, 11 dan 14 diuji menggunakan parameter populasi=400 dan iterasi=400. Keempat dataset tersebut dijalankan sebanyak 10 kali dan didapatkan hasil seperti Tabel 6.13

Tabel 6.12 Hasil Skenario I

Data	Biaya Perjalanan		
	Best	Worst	Average
1	1396	1573	1467.8
6	11523	12060	11811.7
11	87871	92045	89995.9
14	215650.7	218738	212381

Dari hasil pengujian skenario parameter keempat data didapatkan hasil rata-rata solusi. Berikut adalah hasil rata-rata solusi yang dihasilkan dari pengujian 9 skenario dapat dilihat pada Tabel

Tabel 6.13 Hasil uji 9 Skenario

Skenario	Biaya Perjalanan			
	Data 1	Data 6	Data 11	Data14
A	3966.6	12150.4	93837.8	219245.2
B	2577.2	12216.5	92853.7	218822.7
C	1849.3	12165.1	92853.7	219095.8
D	2122.4	11880.4	91248.3	217193.9
E	2162.3	11893.4	91482	217278.5
F	2024.8	11935	91168.4	217235.7
G	1774.2	11793.3	90372.9	216405.8
H	1818.7	11823.6	90849.7	215406.9
I	1467.8	11811.7	89995.9	215650.7

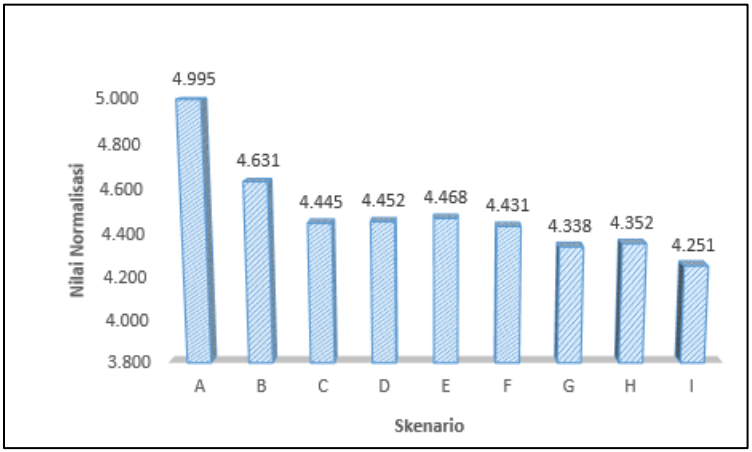
Untuk memilih solusi mana yang cocok maka dilakukan kalkulasi sederhana dimana nilai tiap solusi bagi dengan maksimal solusi yang keluar lalu dikalikan dengan bobot dataset. Untuk dataset 1 dan 6 memiliki bobot 1 sedangkan dataset 11 dan 14 memiliki bobot 1,5. Besar pembobotan ini sendiri mengacu pada bobot nilai dataset saat development

phase. Hasil terkecil dari kalkulasi akan dijadikan parameter dalam menguji semua dataset. Dari Perhitungan yang dilakukan ternyata skenario I memiliki kalkulasi yang kecil sehingga dipilihlah Skenario I. Untuk rincian perhitungan dapat dilihat pada Tabel 6.15 dan Gambar 6.1.

Tabel 6.14 Hasil Penilaian 9 Skenario

Skenario	Nilai Normalisasi				
	Data1	Data6	Data11	Data14	Hasil*
A	1.000	0.995	1.000	1.000	4.995
B	0.650	1.000	0.990	0.998	4.631
C	0.466	0.996	0.990	0.999	4.445
D	0.535	0.972	0.972	0.991	4.452
E	0.545	0.974	0.975	0.991	4.468
F	0.510	0.977	0.972	0.991	4.431
G	0.447	0.965	0.963	0.987	4.338
H	0.459	0.968	0.968	0.982	4.352
I	0.370	0.967	0.959	0.984	4.251

*Perhitungan ini, digunakan untuk mendapatkan nilai dengan skala yang sama. Banyak perhitungan lainnya yang dapat digunakan



Gambar 6.1 Grafik Penilaian Skenario

6.5. Hasil Uji Coba Algoritma Fuzzy-genetika

Dalam pengujian kali ini semua dataset dijalankan menggunakan skenario terpilih yaitu skenario I. Dalam pengujian ini semua dataset dijalankan sebanyak 10. Berikut ini adalah hasil dari penggunaan *fuzzy*-genetika.

Tabel 6.15 Hasil Pengujian Algoritma Fuzzy-genetika

Data	Biaya Perjalanan(Solusi Akhir)		
	Average	Best	Worst
1	1504.3	1396	1660
2	1498	1498	1498
3	10632	8824	12107
4	36148.4	32980	38549
5	3765.6	3559	4143
6	11526.8	11394	11690
7	80270.9	78154	82864
8	13384.5	12330	14068

Data	Biaya Perjalanan(Solusi Akhir)		
	Average	Best	Worst
9	265688.5	251849	272033
10	366164.4	357114	374173
11	91018.4	88979	92435
12	133283.5	128460	135606
13	179637.6	177099	182881
14	215036.6	211190	218483

6.6. Performa Algoritma Fuzzy-genetika Vs Genetika

Dalam menguji performa yang algoritma yang dibuat, dilakukan pengujian dengan algoritma genetika dengan parameter populasi=400 iterasi=400 probabilitas *crossover*=0,75 dan probabilitas mutasi=0,015. Dari algoritma genetika tersebut didapatkan hasil seperti pada Tabel 6.17.

Tabel 6.16 Hasil Pengujian Algoritma Genetika

Data	Biaya Perjalanan(Solusi Akhir)		
	Average	Best	Worst
1	3072.8	1714	5798
2	1498	1498	1498
3	13359.2	11710	14863
4	43705.1	38041	48498
5	4569.8	4428	4676
6	11828.4	11377	12063
7	83805.5	80236	85744
8	13891.2	13204	14645
9	268411.9	258604	274182
10	370475.8	370433	370531
11	91316.1	90466	92581

Data	Biaya Perjalanan(Solusi Akhir)		
	Average	Best	Worst
12	133939.5	131286	135580
13	180393.8	176483	184410
14	214841.9	207768	217547

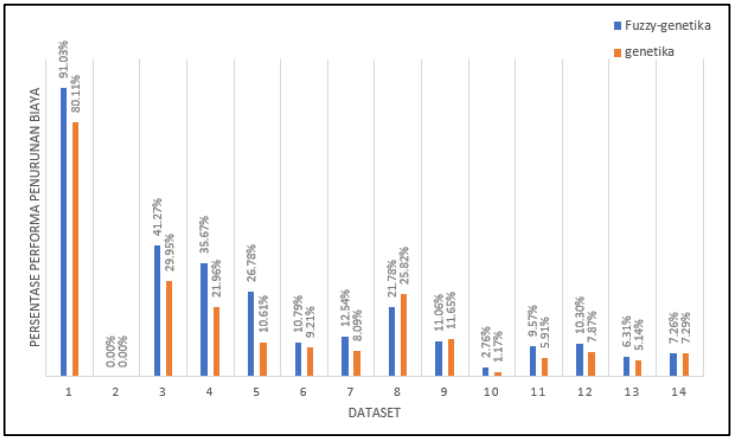
Dari hasil tersebut dilakukan penilaian performa dari algoritma *fuzzy* genetika dan genetika. Untuk melakukan penilaian ini dilakukan penghitungan selisih dari solusi awal dengan solusi akhir lalu membaginya dengan nilai solusi awal dan dikalikan dengan 100%. Untuk hasil perhitungan dapat dilihat pada Tabel 6.18.

Tabel 6.17 Hasil Perbandingan Rata-rata(Average) Algoritma Fuzzy-genetika dan Genetika

Data	Fuzzy-Genetika			Genetika			Selisih
	Solusi awal	Solusi akhir	Performa	Solusi awal	Solusi akhir	Performa	
1	16763.6	1504.3	91.03%	15449.2	3072.8	80.11%	10.92%
2	1498	1498	0.00%	1498	1498	0.00%	0.00%
3	18104.7	10632	41.27%	19070.1	13359.2	29.95%	11.33%
4	56190.4	36148.4	35.67%	56005.7	43705.1	21.96%	13.70%
5	5143.1	3765.6	26.78%	5112.2	4569.8	10.61%	16.17%
6	12920.3	11526.8	10.79%	13029	11828.4	9.21%	1.57%
7	91775.4	80270.9	12.54%	91186.9	83805.5	8.09%	4.44%
8	17111.6	13384.5	21.78%	18727.5	13891.2	25.82%	-4.04%
9	298714.7	265688.5	11.06%	303821.3	268411.9	11.65%	-0.60%
10	376567.7	366164.4	2.76%	374863.2	370475.8	1.17%	1.59%
11	100654.1	91018.4	9.57%	97049.6	91316.1	5.91%	3.67%
12	148588.6	133283.5	10.30%	145381	133939.5	7.87%	2.43%

13	191727.3	179637.6	6.31%	190174.3	180393.8	5.14%	1.16%
14	231881.8	215036.6	7.26%	231747	214841.9	7.29%	-0.03%
Rata-Rata			20.51%	Rata-Rata		16.06%	4.45%

Dari hasil yang didapatkan bahwa performa *fuzzy*-genetika lebih baik daripada algoritma genetika biasa. Dalam menghitung performa pada penelitian ini dihitung dengan cara membagi selisih penghematan biaya lalu dibagi dengan biaya awal. Sehingga jika nilai performa semakin besar menunjukkan bahwa semakin menghemat biaya. Kalkulasi rata-rata performa didapatkan bahwa performa algoritma *fuzzy*-genetika memiliki kalkulasi 20.51% lebih baik dari solusi awal, sedangkan algoritma genetika memiliki kalkulasi 16.06% lebih baik daripada solusi awal. Pada dataset 8, 9 dan 14 performa algoritma genetika lebih baik daripada algoritma fuzzy-genetika. Untuk dataset 2 karena hanya terdapat 1 rute maka nilai solusi awal dan solusi akhir tidak berpengaruh. Gambar 6.2 menunjukan grafik perbandingan algoritma fuzzy-genetika dan genetika.



Gambar 6.2 Grafik Perbandingan Fuzzy-genetika dan Genetika

Halaman ini sengaja dikosongkan

BAB 7
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dan saran untuk penetilian selanjutnya. Kesimpulan dan saran ini diambil dari hasil penetitian yang sudah dilakukan.

7.1. Kesimpulan

Setelah dilakukan implementasi algoritma fuzzy-genetika dalam menyelesaikan permasalahan *TSC* 2.0 dapat diambil kesimpulan bahwa:

1. Algoritma *fuzzy*-genetika adalah salah satu algoritma yang dapat digunakan untuk menyelesaikan permasalahan pencarian rute perjalanan *TSC* 2.0.
2. Perubahan parameter populasi dan iterasi dapat mempengaruhi performa dari algoritma genetika. Semakin besar nilai populasi dan iterasi akan menghasilkan solusi lebih baik.
3. Dalam penelitian ini algoritma *fuzzy*-genetika terbukti mampu memberikan solusi lebih baik dari algoritma genetika dengan nilai performa rata-rata yaitu 21,51% sedangkan algoritma genetika berkisaran 16,06% dari solusi awal.
4. Dilihat dari nilai solusi akhir yang ditawarkan, algoritma fuzzy genetika menawarkan biaya lebih sedikit hampir pada semua dataset kecuali dataset 14. Namun dilihat dari nilai performa penghematan biaya, algoritma *fuzzy*-genetika tidak dapat melebihi algoritma genetika tidak hanya pada dataset 14 tetapi juga pada dataset 2, 8 dan 9. Untuk dataset 2 tidak terjadi

penurunan biaya, dikarenakan hanya terdapat 1 rute yang memenuhi semua batasan.

7.2. Saran

Berdasarkan kesimpulan yang diuraikan oleh penulis, saran yang dapat diberikan kepada penelitian selanjutnya adalah:

1. Dalam penelitian ini pengujian dilakukan dengan parameter yang sama sebanyak 10 kali. Untuk penelitian selanjutnya alangkah baiknya untuk meningkatkan jumlah pengujian untuk mendapatkan hasil yang akurat.
2. Dalam penelitian ini pengujian dilakukan dengan menggunakan 9 skenario dari 2 parameter input yang berbeda. Untuk penelitian selanjutnya alangkah baiknya untuk menambahkan jumlah skenario yang ada untuk mendapatkan pengaturan parameter yang terbaik
3. Pada penelitian ini dilakukan dengan keanggotaan *fuzzy* yang telah ada pada penelitian yang sebelumnya. Alangkah baiknya untuk mencoba range keanggotaan *fuzzy* yang lainnya untuk mengetahui *range fuzzy* yang cocok.

DAFTAR PUSTAKA

[1] S. Puspitorini, "Penyelesaian Masalah Traveling Salesman Problem dengan Jaringan Saraf Self Organizing," *Media Inform.*, vol. 6, no. 1, pp. 39–55, 2008.

[2] G. Laporte, "A short history of the traveling salesman problem," *Canada Res. Chair Distrib. Manag. ...*, p. 26, 2006.

[3] U. Hacizade and I. Kaya, "GA Based Traveling Salesman Problem Solution and its Application to Transport Routes Optimization," *IFAC-PapersOnLine*, vol. 51, no. 30, pp. 620–625, 2018.

[4] J. McCall, "Genetic algorithms for modelling and optimisation," *J. Comput. Appl. Math.*, vol. 184, no. 1, pp. 205–222, 2005.

[5] K. Deb, "An Effcient Constraint Handling Method for Genetic Algorithms," *Millenn. J. Int. Stud.*, vol. 44, pp. 311–338, 2000.

[6] R. R. Yager, "Fuzzy logics and artificial intelligence," *Fuzzy Sets Syst.*, vol. 90, no. 2, pp. 193–198, 1997.

[7] D. Dubois, H. Prade, and P. Sabatier, "An introduction to fuzzy systems," *Clin. Chim. Acta*, vol. 270, pp. 3–29, 1998.

[8] S. Muzid, "Pemanfaatan Algoritma Fuzzy Evolusi Untuk Penyelesaian," *Semin. Nas. Apl. Teknol. Inf.*, no. Snati, pp. 33–38, 2008.

[9] S. Muzid, "Dinamisasi Parameter Algoritma Genetika Menggunakan Population Resizing On Fitness

Improvement Fuzzy Evolutionary Algorithm," *Pros. SNATIF Ke-1*, vol. 1, pp. 145–152, 2014.

[10] A. P. Plerou, "Fuzzy Genetic Algorithms : Fuzzy Logic Controllers and Genetics Algorithms," *Glob. J. Res. Anal.*, vol. 5, no. 11, pp. 497–500, 2016.

[11] N. Ernest and K. Cohen, "Fuzzy Logic Clustering of Multiple Traveling Salesman Problem for Self-Crossover Based Genetic Algorithm," no. January, pp. 1–5, 2014.

[12] M. Cárdenas-Montes, "Creating hard-to-solve instances of travelling salesman problem," *Appl. Soft Comput. J.*, vol. 71, pp. 268–276, 2018.

[13] E. Hart, S. Schulenburg, P. Ross, E. Burke, G. Kendall, and J. Newall, "Hyper-Heuristics: An Emerging Direction in Modern Search Technology," *Springer, Boston, MA*, vol. 57, pp. 457–474, 2003.

BIODATA PENULIS

Yayan Irfan Ferdiawan, lahir di Blora, 30 April 1997. Biasa dipanggil Yayan, Menenempuh pendidikan dasar di SD Ngelo 4 yang sekarang sudah gabungkan dengan SD Ngelo 2 pada tahun 2003 hingga 2009. Setelah lulus dari sekolah dasar, penulis melanjutkan pendidikan formal di bangku sekolah menengah pertama di SMP Negeri 2 Cepu dan lulus pada tahun 2012. Setelah itu penulis melanjutkan pendidikan formal di SMA Negeri 1 Cepu dan lulus pada tahun 2015. Pada tahun 2015, penulis melanjutkan pendidikan tinggi di Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Insitut Teknologi Sepuluh Nopember (ITS) Surabaya, melalui jalur Seleksi Nasional Masuk Perguruan Tinggi (SNMPTN) tahun 2015. Selama menjalani pendidikan tinggi di ITS, penulis menjalankan kegiatan yang berkaitan dengan perkuliahan dan mengikuti beberapa kegiatan organisasi maupun kepanitiaan.

Untuk mengetahui informasi lebih lanjut mengenai penelitian ini maupun terkait dengan penulis, dapat menghubungi melalui email yayanirfanferdiawan@gmail.com.

Halaman ini sengaja dikosongkan

LAMPIRAN A

Hasil Algoritma Fuzzy-genetika

Tabel A-1 Hasil Fuzzy-genetika Dataset 1

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
1	400	400	20148	1396	330
1	400	400	16341	1479	121
1	400	400	18131	1548	51
1	400	400	14654	1472	49
1	400	400	12138	1481	37
1	400	400	17841	1554	24
1	400	400	21535	1479	24
1	400	400	12936	1493	21
1	400	400	22000	1660	21
1	400	400	11912	1481	24
AVG			16764	1504	70.2
BEST			11912	1396	21
Worst			22000	1660	330

Tabel A-2 Hasil Fuzzy-genetika Dataset 2

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
2	400	400	1498	1498	223
2	400	400	1498	1498	79
2	400	400	1498	1498	75
2	400	400	1498	1498	62
2	400	400	1498	1498	32

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
2	400	400	1498	1498	21
2	400	400	1498	1498	48
2	400	400	1498	1498	35
2	400	400	1498	1498	21
2	400	400	1498	1498	18
AVG			1498	1498	61.4
BEST			1498	1498	18
Worst			1498	1498	223

Tabel A-3 Hasil Fuzzy-genetika Dataset 3

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
3	400	400	16967	10850	529
3	400	400	18560	10247	251
3	400	400	15900	8824	128
3	400	400	18458	9919	180
3	400	400	19979	11372	142
3	400	400	18416	12101	78
3	400	400	18583	9990	67
3	400	400	18789	12107	53
3	400	400	17555	11348	48
3	400	400	17840	9562	53
AVG			18105	10632	152.9
BEST			15900	8824	48
Worst			19979	12107	529

Tabel A-4 Hasil Fuzzy-genetika Dataset 4

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
4	400	400	54325	38549	1317
4	400	400	51348	37471	752
4	400	400	56768	32980	545
4	400	400	51514	36746	633
4	400	400	59270	34033	531
4	400	400	56370	34743	548
4	400	400	55576	35730	597
4	400	400	61581	38148	575
4	400	400	57437	37009	571
4	400	400	57715	36075	498
AVG			56190	36148	656.7
BEST			51348	32980	498
Worst			61581	38549	1317

Tabel A-5 Hasil Fuzzy-genetika Dataset 5

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
5	400	400	5364	3589	6452
5	400	400	5347	3600	6718
5	400	400	4882	4002	5949
5	400	400	5213	3696	5231
5	400	400	4758	3734	5228
5	400	400	5293	3914	5232
5	400	400	5083	3832	5226
5	400	400	5095	4143	5209
5	400	400	4956	3587	5257
5	400	400	5440	3559	5219

AVG	5143.1	3765.6	5572.1
BEST	4758	3559	5209
Worst	5440	4143	6718

Tabel A-6 Hasil Fuzzy-genetika Dataset 6

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
6	400	400	13002	11690	77240
6	400	400	12858	11688	77146
6	400	400	13003	11530	79485
6	400	400	13026	11471	84046
6	400	400	12617	11573	79144
6	400	400	12831	11394	78468
6	400	400	12853	11442	75587
6	400	400	13030	11506	74522
6	400	400	12812	11402	72631
6	400	400	13171	11572	71645
AVG			12920	11527	76991
BEST			12617	11394	71645
Worst			13171	11690	84046

Tabel A-7 Hasil Fuzzy-genetika Dataset 7

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
7	400	400	92812	80728	117643
7	400	400	97373	78626	113815
7	400	400	90357	82864	118865
7	400	400	91921	82185	113551
7	400	400	88865	79890	109293
7	400	400	90396	78154	105594

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
7	400	400	93508	81311	103319
7	400	400	90947	79955	94782
7	400	400	94203	80303	95438
7	400	400	87372	78693	93799
AVG			91775	80271	106610
BEST			87372	78154	93799
Worst			97373	82864	118865

Tabel A-8 Hasil Fuzzy-genetika Dataset 8

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
8	400	400	19463	13626	449003
8	400	400	15285	13824	432296
8	400	400	18676	13878	398111
8	400	400	16354	13911	411728
8	400	400	16002	14068	388984
8	400	400	19632	13191	357802
8	400	400	18090	13027	348414
8	400	400	17644	13094	349777
8	400	400	13470	12896	347845
8	400	400	16500	12330	351813
AVG			17111.6	13384.5	383577
BEST			13470	12330	347845
Worst			19632	14068	449003

Tabel A-9 Hasil Fuzzy-genetika Dataset 9

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
9	400	400	309224	272033	1207170
9	400	400	298445	267178	1297761
9	400	400	294060	251849	1294191
9	400	400	296169	271095	1166105
9	400	400	293603	267256	1160510
9	400	400	295959	267270	1051103
9	400	400	313310	266823	933869
9	400	400	288952	263659	991744
9	400	400	308220	263295	1000314
9	400	400	289205	266427	1057206
AVG			298714.7	265688.5	1115997
BEST			288952	251849	933869
Worst			313310	272033	1297761

Tabel A-10 Hasil Fuzzy-genetika Dataset 10

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
10	400	400	375448	370081	4271594
10	400	400	374433	358181	4308455
10	400	400	378486	373305	3273664
10	400	400	378471	369967	3668877
10	400	400	374447	357114	2871851
10	400	400	374444	374173	4234838
10	400	400	378480	358139	3074895
10	400	400	378443	365222	3837134

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
10	400	400	378471	365179	3661535
10	400	400	374554	370283	3882751
AVG			376568	366164	3708559
BEST			374433	357114	2871851
Worst			378486	374173	4308455

Tabel A-11 Hasil Fuzzy-genetika Dataset 11

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
11	400	400	106572	92435	192044
11	400	400	98819	92026	233477
11	400	400	99444	90173	208576
11	400	400	100461	89870	195383
11	400	400	99043	91383	214192
11	400	400	96172	92308	214697
11	400	400	101686	88979	224317
11	400	400	103926	91053	230045
11	400	400	100141	90884	216067
11	400	400	100277	91073	218443
AVG			100654.1	91018.4	214724.1
BEST			96172	88979	192044
Worst			106572	92435	233477

Tabel A-12 Hasil Fuzzy-genetika Dataset 12

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
12	400	400	149591	132712	427416
12	400	400	152931	135247	386440

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
12	400	400	154234	131151	475313
12	400	400	149112	134294	522726
12	400	400	145506	134409	461578
12	400	400	147581	128460	452937
12	400	400	150231	135606	417626
12	400	400	149837	133021	405790
12	400	400	144358	132519	428478
12	400	400	142505	135416	407120
AVG			148588.6	133283.5	438542.4
BEST			142505	128460	386440
Worst			154234	135606	522726

Tabel A-13 Hasil Fuzzy-genetika Dataset 13

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
13	400	400	194573	178632	2360966
13	400	400	188608	179782	2486013
13	400	400	190914	182881	2418183
13	400	400	191381	177099	2553506
13	400	400	190622	178632	2602653
13	400	400	197747	181652	2439983
13	400	400	188233	177523	2396956
13	400	400	192125	180431	2414935
13	400	400	188699	180958	2429984
13	400	400	194371	178786	2204022
AVG			191727.3	179638	2430720
BEST			188233	177099	2204022

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
Worst			197747	182881	2602653

Tabel A-14 Hasil Fuzzy-genetika Dataset 14

DATA	Parameter		Solusi awal	Solusi akhir	Time(Mili)
	Populasi	Iterasi			
14	400	400	238681	211696	181673
14	400	400	227943	215648	187049
14	400	400	234303	214592	224657
14	400	400	240691	215189	207219
14	400	400	228778	216041	257650
14	400	400	241081	214688	223058
14	400	400	234112	217605	214107
14	400	400	224615	211190	224211
14	400	400	226167	218483	189627
14	400	400	222447	215234	195016
AVG			231881.8	215036.6	210426.7
BEST			222447	211190	181673
Worst			241081	218483	257650