Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IS184853

OPTIMASI RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELLING SALESMAN CHALLENGE 2.0*

OPTIMIZATION OF ITINERARIUM BY AIRPLANE USING TABU-SIMULATED ANNEALING ALGORITHM TO SOLVE THE PROBLEM IN TRAVELING SALESMAN CHALLENGE 2.0

EDWIN DWI AHMAD
NRP 05211540000060

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 RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELING SALESMAN CHALLENGE 2.0*

EDWIN DWI AHMAD
NRP 05211540000060

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

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

FINAL PROJECT - IS184853

OPTIMIZATION OF ITINERARIUM BY AIRPLANE USING TABU-SIMULATED ANNEALING ALGORITHM TO SOLVE THE PROBLEM IN TRAVELLING SALESMAN CHALLENGE 2.0

EDWIN DWI AHMAD
NRP 05211540000060

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

LEMBAR PENGESAHAN

OPTIMASI RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELLING SALESMAN CHALLENGE 2.0*

TUGAS AKHIR

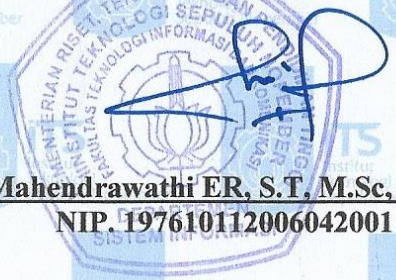
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

EDWIN DWI AHMAD
0521 15 4000 0060

Surabaya, Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI



Mahendrawathi ER, S.T, M.Sc, Ph.D
NIP. 197610112006042001

LEMBAR PERSETUJUAN

OPTIMASI RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELLING 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 :

EDWIN DWI AHMAD
0521 15 4000 0060

Disetujui Tim Penguji :

Tanggal Ujian : Juli 2019
Periode Wisuda : September 2019

Ahmad Muklason, S.Kom, M.Sc, Ph.D **(Pembimbing 1)**

Edwin Riksakomara, S.Kom, M.T **(Penguji 1)**

Raras Tyasnurita, S.Kom, MBA, Ph.D **(Penguji 2)**

OPTIMASI RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELLING SALESMAN CHALLENGE 2.0*

Nama Mahasiswa	**: Edwin Dwi Ahmad**
NRP	**: 05211540000060**
Departemen	**: Sistem Informasi**
Pembimbing	**: Ahmad Muklason, S.Kom, M.Sc, Ph.D**

ABSTRAK

Travelling Salesman Problem (TSP) termasuk ke dalam golongan permasalahan Non-deterministic Polynomial NP-hard yang berarti belum ada algoritma eksak yang dapat menyelesaikannya dalam waktu polynomial. Tujuan TSP adalah untuk mencari rute perjalanan terpendek dimana perjalanan tersebut dimulai dari sebuah kota dan harus berakhir pada kota yang sama dengan kota keberangkatan dan setiap kota yang dilewati harus dikunjungi tepat satu kali. Selain menemukan rute perjalanan terpendek, TSP juga digunakan untuk mencari rute perjalanan dengan biaya seminimal mungkin. Salah satu permasalahan TSP terdapat di dalam kompetisi Travelling Salesman Challenge (TSC) 2.0 yang diselenggarakan pada tahun 2018. Tujuan TSC 2.0 adalah menemukan rute perjalanan menggunakan pesawat terbang dengan biaya termurah. Setiap rute yang diambil harus melewati seluruh area yang ditentukan. Masing-masing area terdapat satu atau lebih kota dan dimana dalam setiap area hanya boleh satu kota saja yang boleh dikunjungi. Tantangan dalam permasalahan ini adalah banyaknya batasan-batasan soft constraint maupun hard constraint yang harus dipenuhi. Oleh karena itu, untuk menyelesaikan permasalahan pada TSC 2.0 dalam tugas akhir ini akan digunakan algoritma hybrid Tabu Search dan Simulated Annealing. Pemilihan algoritma

hybrid Tabu Search dan Simulated Annealing karena algoritma ini telah banyak dikembangkan untuk mengatasi permasalahan NP-hard. Hasil yang diperoleh dari penelitian tugas akhir ini menunjukkan algoritma tabu-simulated annealing dapat menyelesaikan permasalahan pada TSC 2.0 dengan mereduksi 48.54% biaya dari solusi awal dan mampu bersaing dengan algoritma great deluge yang hanya mampu mereduksi 41.33% dari solusi awal.

Kata Kunci: Algoritma Simulated Annealing, Algoritma Tabu Search, Travelling Salesman Problem

OPTIMIZATION OF ITINERARIUM BY AIRPLANE USING TABU-SIMULATED ANNEALING ALGORITHM TO SOLVE THE PROBLEM IN TRAVELLING SALESMAN CHALLENGE 2.0

Name : Edwin Dwi Ahmad
NRP : 05211540000060
Department : Information System
Supervisor : Ahmad Muklason, S.Kom, M.Sc, Ph.D

ABSTRACT

Traveling Salesman Problem (TSP) belongs to a group of Non-deterministic Polynomial NP-hard problems, which means there is no exact algorithm that can solve it in polynomial time. The purpose of TSP is to find the shortest distance where the journey starts from a city and must end in the same city as the city of departure and each city that is passed must be visited exactly once. In addition to finding the shortest distance, TSP is also used to find routes at minimum cost. One of the TSP problems is in the Traveling Salesman Challenge (TSC) 2.0 competition held in 2018. The purpose of TSC 2.0 is to find routes using airplanes at the lowest cost. Each route taken must pass the entire specified area. Each area has one or more cities and where in each area only one city can be visited. The challenge in this problem is the many constraints of soft constraints and hard constraints that must be met. Therefore, to solve problems in TSC 2.0 in this final project, a hybrid Tabu Search and Simulated Annealing algorithm will be used. The selection of hybrid Tabu Search and Simulated Annealing algorithms because this algorithm has been widely developed to overcome NP-hard problems. The results obtained from this final assignment show that the tabu-simulated annealing algorithm can solve the problems in TSC 2.0 by reducing 48.54% of the cost of the initial solution and able to compete with the great deluge algorithm which can only reduce 41.33% of the initial solution.

Kata Kunci: Algoritma Simulated Annealing, Algoritma Tabu Search, Travelling Salesman Problem

KATA PENGANTAR

Puji syukur atas karunia yang telah diberikan Allah SWT. selama ini sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul: "OPTIMASI RUTE RENCANA PERJALANAN DENGAN PESAWAT TERBANG MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING* UNTUK MENYELESAIKAN PERMASALAHAN *TRAVELLING SALESMAN CHALLENGE 2.0*"

Pengerjaan tugas akhir ini tidak lepas dari bantuan, bimbingan, semangat, motivasi dan dukungan baik moral maupun doa. Oleh karena itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada,

1. Bapak Mat Ikhsan dan Ibu Mariyati selaku orang tua yang telah memberikan dukungan baik segi material, semangat dan doa agar bisa menyelesaikan tugas akhir ini dengan lancar.
2. Keluarga dan sanak saudara penulis yang telah mendukung dan mendoakan tiada henti untuk menyelesaikan tugas akhir ini.
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, M.T dan Ibu Raras Tyasnurita, S.Kom, MBA, Ph.D selaku dosen penguji yang telah memberikan kritik dan saran yang membangun agar tugas akhir ini menjadi lebih baik.
5. Prema, Yayan, dan Gilang selaku teman satu topik penelitian yang telah membantu dalam proses tugas akhir ini.
6. Teman-teman laboratorium RDIB, ADDI, dan MSI yang selalu menemani penulis dalam mengerjakan tugas akhir.
7. Teman – teman Lannister, khususnya yang menjadi anggota kontrakan Lamtoro jilid II.

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, Juli 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan relevansi dengan laboratorium RDIB.

1.1 Latar Belakang

Travelling Salesman Problem (TSP) merupakan permasalahan yang dihadapi oleh seorang salesman dimana orang tersebut harus mengunjungi semua kota. Salesman tersebut memulai perjalanannya dari sebuah kota dan harus kembali ke kota keberangkatanya [1]. Hasil yang diharapkan pada TSP adalah menemukan rute terpendek yang dapat ditempuh untuk mengunjungi setiap kota yang telah disediakan,dengan syarat semua kota harus dikunjungi tepat satu kali dalam setiap perjalanannya [2]. Selain digunakan untuk menemukan rute perjalanan terpendek, TSP juga digunakan untuk menemukan rute perjalanan dengan total biaya paling sedikit [3].

Permasalahan pada TSP merupakan permasalahan klasik di dalam optimasi kombinatorial. Seperti yang telah diketahui, TSP tergolong ke dalam permasalahan *NP-hard* yang berarti permasalahan yang sulit untuk diselesaikan dalam waktu *polynomial*. Kompleksitas komputasional pada TSP akan meningkat seiring dengan bertambahnya jumlah kota [2]. Banyak penelitian yang telah dilakukan untuk menyelesaikan permasalahan TSP, penelitian-penelitian tersebut menggunakan algoritma *heuristic*, seperti *simulated annealing*, *ant colony optimiz*ation dan *tabu search* [4].

Travelling Salesman Challange 2.0 (TSC 2.0) mengadakan sebuah kompetisi dengan mengangkat permasalahan pada TSP. Hasil yang diharapkan dari TSC 2.0 adalah rute rencana perjalanan menggunakan pesawat terbang dengan biaya seminimal mungkin. Pada kasus ini, rute yang dipilih harus melewati seluruh area yang ada. Di dalam setiap area sedikitnya terdapat satu kota, namun apabila terdapat satu area yang

memiliki beberapa kota maka hanya satu kota saja yang boleh dikunjungi. Pemilihan kota pada setiap area dalam perencanaan rute perjalanan ini dibebaskan, adapun dengan syarat untuk mengunjungi seluruh area telah terpenuhi.

Berbagai macam penelitian telah dilakukan terkait TSP dengan menggunakan pendekatan yang berbeda. Salah satunya adalah penelitian dengan menerapkan algoritma *hybrid* s*imulated annealing* dan *tabu search* [5]. Penelitian yang dilakukan menunjukkan algoritma yang dikembangkan dapat memperoleh hasil yang optimal. Selain itu dengan menggabungkan dua algoritma dapat mengatasi kelemahan masing-masing algoritma.

Penelitian berikutnya menggunakan algoritma *simulated annealing* dan *gene-expression* [4]. Pada penelitian ini algoritma *simulated annealing* digunakan untuk meningkatkan keanekaragaman populasi pada *gene-expression* serta memperbaiki dalam global search. Hasil yang diperoleh pada penelitian ini adalah algoritma yang digunakan berhasil mengungguli algoritma *heuristic* lainnya dalam hal mencari solusi terbaik.

Berdasarkan penelitian yang telah dilakukan sebelumnya, maka metode yang digunakan untuk menyelesaikan permasalahan pada optimasi rute rencana perjalanan dengan pesawat terbang adalah dengan menggunakan algoritma *hybrid* yaitu *tabu search* dan *simulated annealing*. Algoritma tersebut telah memberikan bukti dapat memberikan solusi yang optimal pada permasalahan TSP.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang yang telah dijelaskan, maka rumusan masalah dari tugas akhir ini, yaitu:

1. Bagaimana model matematis untuk permasalahan rute rencana perjalanan dengan pesawat terbang pada TSC 2.0?
2. Bagaimana menerapkan algoritma *tabu-simulated annealing* untuk menyelesaikan permasalahan rute

rencana perjalanan dengan pesawat terbang pada TSC 2.0?
3. Bagaimana performa algoritma *tabu-simulated annealing* dalam menyelesaikan permasalahan rute rencana perjalanan dengan pesawat terbang pada TSC 2.0?

1.3 Batasan Masalah

Berdasarkan uraian permasalahan di atas, batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Data yang digunakan dalam melakukan optimasi rute rencana perjalanan dengan pesawat terbang adalah dataset TSC 2.0.
2. Hasil dari tugas akhir ini adalah aplikasi yang dibangun dengan menggunakan bahasa pemrograman Java.

1.4 Tujuan Tugas Akhir

Tujuan yang hendak dicapai dalam pengerjaan tugas akhir ini, yaitu:

1. Membuat model matematis untuk menyelesaikan permasalahan rute rencana perjalanan dengan pesawat terbang.
2. Mengoptimalkan rute rencana perjalanan dengan pesawat terbang menggunakan algoritma *tabu-simulated annealing*.
3. Melakukan analisa performa algoritma *tabu-simulated annealing* dalam menyelesaikan permasalahan rute rencana perjalanan dengan pesawat terbang.

1.5 Manfaat Tugas Akhir

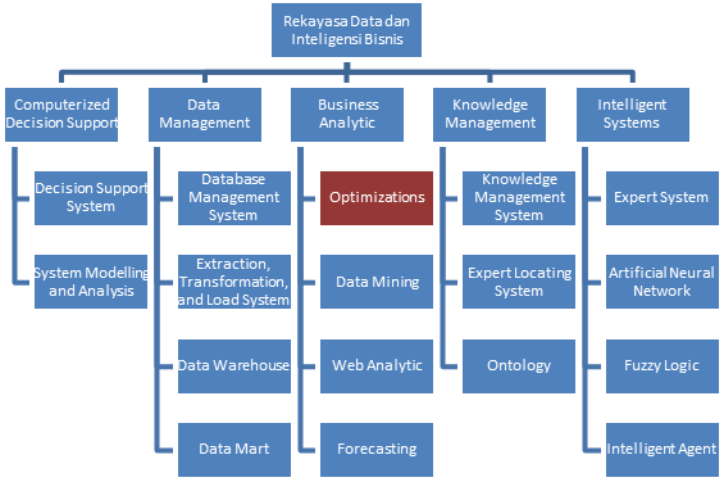
Manfaat yang dapat diberikan dengan adanya tugas akhir ini adalah sebagai berikut:

1. Dapat dijadikan sebagai pengetahuan dalam bidang optimasi khususnya pada rute rencana perjalanan dengan pesawat terbang.
2. Dapat membantu menyelesaikan permasalahan rute rencana perjalanan dengan pesawat terbang menggunakan algoritma *tabu-simulated annealing*.

3. Dapat dijadikan sebagai bahan rujukan atau referensi tambahan dalam penelitian selanjutnya.

1.6 Relevansi

Topik yang diangkat dalam penyusunan tugas akhir ini adalah optimasi yang berkaitan dengan mata kuliah Optimasi Kombinatorial dan Heuristik. Selain itu topik tersebut juga termasuk dalam bidang keilmuan yang menjadi *research roadmap* pada Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB). Bagan 1.1 merupakan bidang keilmuan yang terdapat pada Laboratorium RDIB.



Gambar 1.1 *Roadmap* Penelitian Laboratorium RDIB

BAB II
TINJAUAN PUSTAKA

Pada bagian ini dilakukan studi literatur yang digunakan sebagai landasan dan referensi objek penelitian serta teori-teori yang mendukung.

2.1 Studi Sebelumnya

Pada sub bab ini akan dijelaskan mengenai beberapa penelitian terdahulu yang telah dilakukan dan memiliki relevansi dengan tugas akhir. Penelitian terdahulu tersebut dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Penelitian Terdahulu	
1	Judul Penelitian	*Developing a Dynamic Neighborhood Structure for an Adaptive Hybrid Simulated Annealing – Tabu Search Algorithm to Solve the Symmetrical Travelling Salesman Problem* [5]
	Tahun Penelitian	2016
	Nama Peneliti	Yu Lin PhD, Zheyong Bian PhD, Xiang Liu PhD
	Deskripsi	Pada penelitian ini, peneliti mengembangkan algoritma untuk mengatasi permasalahan *Travelling Salesman Problem* (TSP), yaitu gabungan antara algoritma s*imulated annealing* dan *tabu search*. Penelitian dilakukan dengan mengembangkan struktur lingkungan yang dinamis pada algoritma *hybrid* untuk meningkatkan efisiensi pencarian dengan cara mereduksi pengacakan pada konvensional *2-opt neighborhood*.

No	Penelitian Terdahulu	
		Peneliti mengusulkan parameter adaptif yang dapat menyesuaikan secara otomatis terhadap algoritma *hybrid*. Selain itu, peneliti juga membandingkan hasil eksperimen dengan algoritma lain.
	Hasil Penelitian	Penelitian ini menunjukan bahwa algoritma yang diusulkan dapat menghasilkan solusi yang memuaskan. Selain itu, algoritma yang dikembangkan dapat mengatasi kelemahan dari masing-masing algoritma *simulated annealing* dan *tabu search*. Penelitian yang dilakukan juga menunjukan struktur lingkungan yang dinamis lebih efisien dan akurat daripada *classical 2-opt*.
	Keterkaitan dengan Tugas Akhir	Penelitian ini telah membuktikan bahwa algoritma *tabu search dan simulated annealing* memberikan solusi yang optimal dalam permasalahan TSP sehingga dapat dijadikan sebagai rujukan dalam pengerjaan tugas akhir.
2	Judul Penelitian	*Solving the Travelling Salesman Problem Based on an Adaptive Simulated Annealing Algorithm with Greedy Search* [6].
	Tahun	2011
	Nama Peneliti	Geng Xiutang, Chen Zhihua, Yang Wei, Deqian Shi, Zhao Kai
	Deskripsi	Penelitian ini mengusulkan algoritma *local search* yang efektif berdasarkan teknik *simulated annealing* dan *greedy search* untuk menyelesaikan permasalahan pada TSP. Algoritma *simulated annealing* digunakan untuk

No	Penelitian Terdahulu	
		memperoleh solusi yang lebih akurat. Kemudian teknik *greedy search* digunakan untuk mempercepat tingkat konvergensi pada algoritma yang diusulkan.
	Hasil Penelitian	Penelitian ini menunjukkan bahwa algoritma yang diusulkan memberikan hasil yang lebih baik antara waktu dan akurasi CPU di antara beberapa algoritma terbaru untuk TSP.
	Keterkaitan dengan Tugas Akhir	Pemanfaatan algoritma *simulated annealing* pada penelitian ini akan dijadikan sebagai rujukan dalam tugas akhir.
3	Judul Penelitian	*Travelling Salesman Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming* [4]
	Tahun	2018
	Nama Peneliti	Zhou Ai-Hua, Zhu Li-Peng, Bin Hu, Song Deng, Yan Song, Hongbin Qiu
	Deskripsi	Penelitian ini mengusulkan algoritma baru untuk menyelesaikan permasalahan TSP, algoritma tersebut adalah *simulated annealing* dan *gene-expression* (GEP). *simulated annealing* digunakan untuk meningkatkan keanekaragaman populasi pada *gene-expression* (GEP) dan untuk memperbaiki kemampuan dalam *global search.* Peneliti juga membandingkan performa algoritma dengan menggunakan enam *instance* yang berbeda.
	Hasil Penelitian	Algoritma yang dikembangkan berhasil mengungguli algoritma *heuristic* lainnya

No	Penelitian Terdahulu	
		dalam hal solusi terbaik, solusi terburuk, dan *running time*.
	Keterkaitan dengan Tugas Akhir	Penyelesaian permasalahan TSP pada penelitian ini akan dijadikan sebagai rujukan dalam mengerjakan tugas akhir ini.

2.2 Dasar Teori

Pada sub bab ini akan dijabarkan mengenai dasar teori yang digunakan untuk mendukung pengerjaan tugas akhir.

2.2.1 *Travelling Salesman Problem*

Travelling Salesman Problem (TSP) merupakan salah satu permasalahan klasik dalam optimasi kombinatorial yang bertujuan untuk mencari rute terpendek dalam mengunjungi semua kota yang ada di dalam peta wisata [2]. Rute yang dipilih bermula dari sebuah kota dan harus berakhir di kota yang sama dengan kota keberangkatan. Setiap kota hanya boleh dikunjungi tepat satu kali.

Permasalahan TSP merupakan masalah yang sulit di dalam komputasional, banyak penelitian yang telah dilakukan dan diumumkan dengan menggunakan metode yang bermacam-macam. Metode tersebut seperti *Ant Colony Optimization* (ACO), *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), *Artificial Bee Colony* (ABC), dan lain-lain [6]. Namun, sampai saat ini belum ada algoritma yang mampu memberikan solusi optimal yang tepat untuk permasalahan TSP [7]. Salah satu contoh penerapan TSP adalah bagaimana menghubungkan komponen komputer dengan menggunakan panjang kabel paling minimum.

Selain bertujuan mencari rute terpendek, pencarian rute perjalanan dengan total biaya minimum juga harus diperhatikan di dalam TSP [3]. Hal ini bertujuan untuk menghemat biaya

yang digunakan di dalam perjalanan, karena seorang *traveller* pada umumnya memiliki uang saku yang terbatas.
Formula yang digunakan untuk menyelesaikan permasalahan TSP adalah seperti persamaan berikut [8]:

Objective function:

$$Min \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ij} \qquad j \neq i \qquad (1)$$

Decision variable:

$$X_{ij} = \{ {}^{0}_{1} \qquad j \neq i \qquad (2)$$

Subject to:

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

$$\sum_{j=0,\ j\neq i}^{n} X_{ij} = 1 \qquad i = 0, \dots, n \qquad (4)$$

$$u_i - u_j + nX_{ij} \leq n - 1 \qquad 1 \leq i \neq j \leq n \qquad (5)$$

Gambar 2.1 Persamaan TSP

Keterangan:

C_{ij} : Biaya perjalanan yang dihabiskan untuk berpindah dari kota i ke kota j

X_{ij} : Rute perjalanan dari kota i ke kota j

n : Jumlah kota pada permasalahan TSP

Ui : *Dummy variable* yang diberikan saat rute melewati ke-i

Uj : *Dummy variable* yang diberikan saat rute melewati ke-j.

Formula (1) merupakan fungsi tujuan dimana rute yang dipilih memiliki biaya paling minimum. Formula (2) menunjukan variabel keputusan dimana salesman dari kota i melalui kota j yang bernilai bilangan biner. Formula (3) dan (4) merupakan batasan yaitu setiap kota hanya boleh dikunjungi satu kali. Formula (5) merupakan batasan yang menunjukkan semua kota terhubung menjadi sebuah rute yang dipilih oleh salesman

dengan menambahkan *dummy variable* untuk memastikan tidak ada *sub route*.

2.2.2 *Travelling Salesman Challenge 2.0*

Travelling Salesman Challenge (TSC) 2.0 merupakan sebuah kompetisi yang diselenggarakan oleh perusahaan travel *online* yaitu Kiwi. Tujuan diselenggarakannya kompetisi ini adalah untuk memecahkan permasalahan dalam perencanaan rute perjalanan dengan pesawat terbang. Perjalanan yang harus dilakukan adalah mengunjungi satu kota yang terdapat di dalam beberapa area yang telah ditentukan. Di dalam satu area bisa terdapat satu kota atau lebih.

Pemilihan rute perjalanan yang dipilih adalah berdasarkan rute yang memiliki biaya paling rendah. Sementara itu, terdapat beberapa *hard constraint* yang harus dipenuhi dalam menentukan rencana perjalanan dengan pesawat terbang pada TSC 2.0. Hard constraint yang harus dipenuhi adalah:

1. Perjalanan harus dimulai dari kota yang telah ditentukan
2. Setiap area harus dikunjungi tepat satu kali
3. Dalam setiap area terdapat satu atau lebih kota dan setiap area salesman harus mengunjungi tepat satu kota saja
4. Setiap hari salesman harus berpindah dari satu area menuju area yang lain
5. Salesman hanya boleh mengunjungi satu area setiap hari
6. Perjalanan salesman ke area berikutnya harus berlanjut dari kota pada area yang dikunjungi sehari sebelumnya
7. Perjalanan harus berakhir pada area yang sama dengan area keberangkatan pertama kali.

Variabel keputusan pada permasalahan ini adalah kota mana saja yang harus dipilih untuk dikunjungi supaya sesuai dengan tujuan dan batasan-batasan yang ada.

Dataset yang digunakan dalam permasalahan ini adalah dataset yang telah disediakan oleh TSC 2.0. Di dalam dataset berisi data tentang jumlah area, titik kota pertama untuk memulai perjalanan, daftar bandara, serta jadwal penerbangan beserta harganya untuk masing-masing penerbangan.

Dataset yang diberikan terdiri dari 14 jenis dataset yang berbeda. Dataset dikelompokkan berdasarkan tipe data dan ukuran datanya. Berdasarkan tipe data dikelompokkan menjadi *artificial data* dan data yang berasal dari penerbangan asli. Sementara itu berdasarkan ukuran datanya, dataset dikelompokkan menjadi *small data*, *medium data*, dan *large data*. Berikut ini merupakan rincian karakteristik dari dataset TSC 2.0 yang dijelaskan dalam Tabel 2.2.

Tabel 2.2 Dataset TSC 2.0

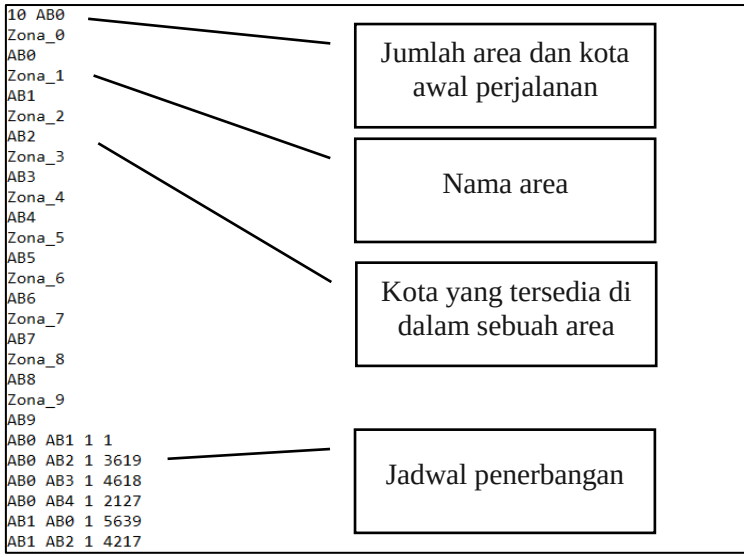
Dataset	Tipe Data	Ukuran Data	Jumlah Area	Jumlah Kota	Jumlah Hari
1	*Artificial*	*Small*	10	10	10
2	*Artificial*	*Small*	10	15	10
3	*Artificial*	*Small*	13	38	13
4	*Artificial*	*Medium*	40	99	40
5	*Artificial*	*Medium*	46	138	46
6	*Artificial*	*Medium*	96	192	96
7	*Artificial*	*Large*	150	300	150
8	*Artificial*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

Dataset	Tipe Data	Ukuran Data	Jumlah Area	Jumlah Kota	Jumlah Hari
13	*Real kiwi.com*	*Large*	250	275	250
14	*Real kiwi.com*	*Large*	300	300	300

Penulisan dataset TSC 2.0 memberikan informasi penting pada setiap baris-barisnya. Baris pertama memberikan dua informasi yaitu jumlah area yang ada di dalam dataset dan titik awal lokasi perjalanan. Baris kedua hingga beberapa baris selanjutnya memberikan informasi nama area yang tersedia.

Setiap area dijelaskan dalam dua baris, baris pertama berisi nama area dan baris kedua berisi daftar kota yang terdapat di dalam area tersebut. Baris selanjutnya sampai baris terakhir memberikan informasi jadwal penerbangan yang tersedia. Informasi yang dapat diperoleh adalah bandara keberangkatan, bandara tujuan, hari pada perjalanan ke-n dan biaya penerbangan.

Hari penerbangan yang memiliki nilai 0 mempunyai arti bahwa penerbangan tersebut dilakukan setiap hari. Sebagai contoh, apabila terdapat informasi seperti “AB0 AB2 1 3619” menunjukkan bahwa pada perjalanan hari ke-1 terdapat penerbangan dari bandara AB0 menuju bandara AB1 dengan biaya sebesar 3619. Gambar 2.2 merupakan contoh penulisan dataset TSC 2.0.



Gambar 2.2 Contoh Dataset

2.2.3 *Hyper Heuristic*

Hyper-heuristic adalah metode pencarian *heuristic* yang digunakan untuk melakukan otomasi proses, biasanya menggunakan *machine learning*. *Hyper-heuristic* dapat juga diartikan sebagai kumpulan pendekatan yang digunakan untuk mengkombinasikan beberapa metode *heuristic* untuk menyelesaikan beberapa permasalahan komputasi yang memiliki tingkat kesulitan tinggi [9]. Tujuan *hyper-heuristic* ini dikembangkan untuk meningkatkan ruang lingkup penyelesaian masalah menjadi lebih umum dan sederhana.

Hyper-heuristic berbeda dengan *metaheuristic*, *metaheuristic* sebagian besar diimplementasikan dalam ruang pencarian solusi masalah. Sementara itu, *hyper-heuristic* diimplementasikan diatas *search space* yang berupa *solution space*. *Hyper-heuristic* tidak memiliki jaminan keberhasilan dalam menemukan solusi yang optimal. Parameter tuning di dalam *hyper-heuristic* dilakukan secara otomatis, sehingga *hyper-heuristic* tidak lagi memerlukan paramater tuning manual [10].

2.2.4 Algoritma *Tabu Search*

Tabu search merupakan salah satu algoritma *metaheuristic* yang menerapkan metode pencarian lokal yang digunakan untuk melakukan optimasi matematika [11]. Prinsip dasar algoritma *tabu search* adalah mencegah solusi yang tidak membaik dengan memanfaatkan ruang memori, yaitu daftar tabu (*tabu list*). *Tabu search* akan menyimpan solusi yang pernah ditemukan dan sebisa mungkin menghindari proses pencarian kembali ke kandidat solusi sebelumnya [12]. Atas dasar itulah algoritma *tabu search* dapat menghindarkan diri dari *local optima solution*.

Setiap iterasi yang dilakukan, memori yang digunakan akan selalu diperbaharui [12]. Namun, menyimpan semua solusi yang pernah ditemukan tentunya membutuhkan kapasitas memori dan waktu yang besar karena setiap iterasi yang dilakukan harus dilakukan pencocokan terhadap solusi baru apakah solusi baru tersebut sudah ada di dalam *tabu list*. Untuk menghindari kebutuhan memori dan waktu yang lama, maka *tabu list* akan dibuat konstan dan berisi pergerakan yang tabu.

Di dalam setiap iterasi apabila solusi yang dihasilkan terdapat pada *tabu list*, maka solusi tersebut tidak diambil sebagai solusi baru. Namun, apabila terdapat solusi yang lebih baik pada solusi sebelumnya, maka solusi yang lebih baik akan menggantikan solusi sebelumnya. Konsep yang digunakan dalam *tabu list* adalah menggunakan aturan FIFO (*first in first out*) yang artinya solusi tabu yang paling awal masuk akan digantikan oleh solusi baru [12]. Kode 2.1 merupakan *pseudocode* algoritma *tabu search*.

```
1: procedure Tabu Search
2:   Tabu list T
3:   current ← initial solution
4:   best ← current
5:    repeat
6:      current ← arg min_{x∈N(current)} f(x), x:
        non-tabu
7:      if current < best then
8:         best ← current
9:      end if
10:        record the recent move in T
           delete the oldest entry if
           necessary
11:   until stop condition
12: end procedure
```

Kode 2.1 *Pseudocode Tabu Search*

2.2.5 Algoritma *Simulated Annealing*

Simulated annealing adalah algoritma *metaheuristic* yang mengadopsi konsep pemuaian pada teori fisika [13]. Apabila sebuah materi keras dipanaskan hingga mencair dan kemudian mendinginkannya, maka struktur materi tersebut bergantung pada sifat pendinginannya. Jika materi didinginkan dengan perlahan, maka akan menghasilkan kristal-kristal yang berkualitas baik. Apabila pendinginan dilakukan terlalu cepat, maka kristal yang terbentuk tidak akan sempurna.

Initial solution atau solusi awal dalam metode *simulated annealing* dimulai dengan pilihan acak. Selanjutnya dilakukan proses evaluasi terhadap solusi awal tersebut sehingga menghasilkan solusi baru. Apabila solusi baru yang didapatkan lebih optimal, maka solusi akan diterima sebagai solusi sementara. Namun, jika solusi baru tidak lebih optimal maka akan melewati fungsi pengontrol [14]. Fungsi pengontrol dinyatakan dengan persamaan *Boltzmann*:

$$P = e^{-\frac{c}{t}}$$

Dengan *P* adalah probabilitas, *e* adalah bilangan eksponensial, *c* adalah perbedaan evaluasi fungsi tujuan antara solusi dengan kandidat solusi, dan *t* merupakan parameter suhu.

Pada *simulated annealing* terdapat parameter-parameter yang dapat mempengaruhi iterasi, berikut ini merupakan parameter-parameter tersebut [15]:

1. Suhu awal dan suhu akhir,
2. Jumlah iterasi setiap penurunan suhu,
3. *Cooling rate*, koefisien yang berfungsi untuk menurunkan suhu.

Pseudocode algoritma *simulated annealing* ditunjukkan oleh Kode 2.3.

```
1: procedure Simulated Annealing
2:    current ← initial solution
3:    t ← initial temperature
4:    l ← initial length
5:    repeat
6:     for i = 1 to l do
7:       candidate ϵ N(current)
8:        if f(candidate) <= f(current) then
9:           current ← candidate
10:        else if
11:         exp(f(current) - f(candidate)=t)
          > random[0,1)then
12:          current ← candidate
13:        end if
14:    end for
15:    update l and t
16:   until stop condition
17: end procedure
```

Kode 2.2 *Pseudocode Simulated Annealing*

2.2.6 Algoritma *Tabu-Simulated Annealing*

Algoritma *hybrid tabu-simulated annealing* merupakan pengembangan algoritma *metheuristic* yang digunakan untuk menghasilkan solusi lebih optimal. Algoritma *hybrid* diimplementasikan untuk mengatasi kekurangan dari masing-masing algoritma.

Di dalam pendekatan algoritma *hybrid, simulated annealing* dimanfaatkan untuk *acceptance criteria* pada *local search*. Sementara itu, *tabu search* dimanfaatkan untuk menyimpan struktur lingkungan yang menghasilkan solusi lebih buruk. Struktur lingkungan disimpan ke dalam *tabu list* untuk tidak digunakan kembali [16].

Dengan menggunakan beberapa contoh masalah yang bervariasi, algoritma *simulated annealing* memiliki kinerja lebih baik untuk target kualitas solusi yang lebih tinggi. Sedangkan algoritma *tabu search* memiliki kinerja lebih baik pada target kualitas solusi lebih rendah [17]. Sehingga dengan menggunakan algoritma *hybrid tabu-simulated annealing* kemungkinan untuk mendapatkan solusi lebih baik menjadi semakin besar.

Kode 2.3 merupakan *pseudocode* algoritma *tabu-simulated annealing*.

```
1:      generate initial solution S
2:      set initial temperatur t0
3:      set cooling rate cr
4:      set final tempeterature t1
5:      set tabu list capacity TL
6:      While t0 > t1
7:        define low level heuristic = S*
8:        if (S* < S)
9:              S = S*
18:       else
                if exp(-delta/t0) > Random number
19:                 S = S*;
20:             else
21:                   cek Tabu list capacity
22:                   if TL = 0
23:                     remove first entry
24:                     add low level heuristic
25:                   else
26:                     add low level heuristic
27:     t0 = cr * t0
```

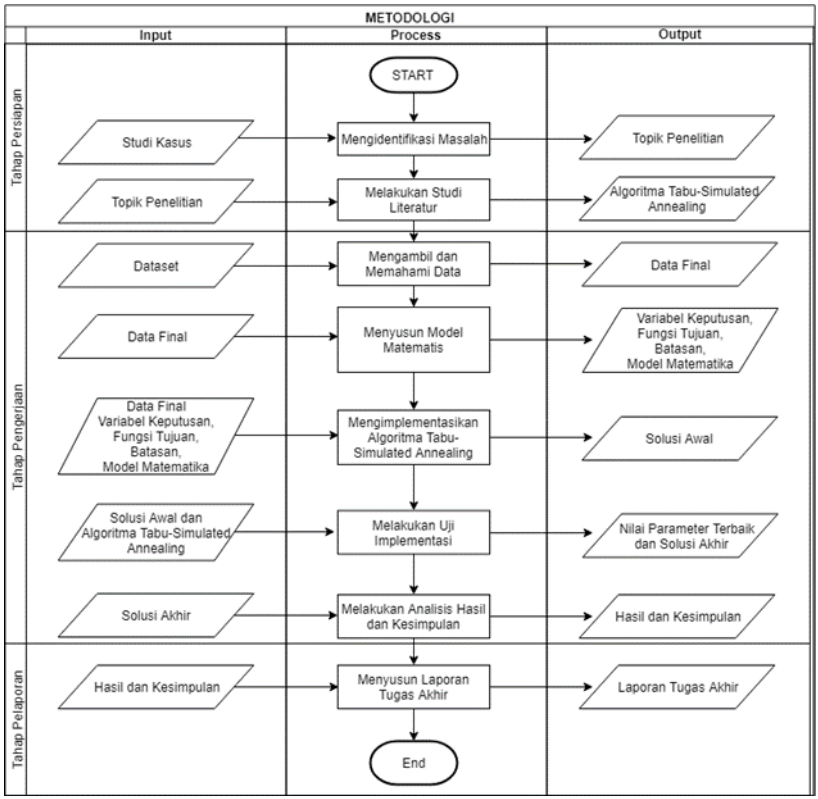
Kode 2.3 *Pseudocode Tabu-Simulated Annealing*

BAB III
METODOLOGI

Pada bab ini akan dijelaskan mengenai alur metodologi yang akan dilakukan dalam tugas akhir ini. Metodologi ini juga digunakan sebagai pedoman untuk melaksanakan tugas akhir agar terarah dan sistematis.

3.1 Metodologi Penelitian

Alur metodologi dalam pengerjaan tugas akhir dapat dilihat pada Gambar 3.1.



Gambar 3.1 Metodologi Penelitian

3.2 Tahapan Pelaksanaan Tugas Akhir

Tahapan pelaksanaan tugas akhir akan menjelaskan terkait segala sesuatu yang akan dikerjakan atau langkah-langkah pengerjaan tugas akhir.

3.2.1 Mengidentifikasi Masalah

Tahap pertama yang dilakukan dalam tugas akhir ini adalah melakukan identifikasi permasalahan dengan cara analisis studi kasus untuk mendapatkan detail topik mengenai suatu permasalahan yang diangkat dalam pengerjaan tugas akhir ini. Hasil yang didapatkan dari tahap ini adalah permasalahan yang diangkat menjadi topik penelitian. Adapun topik penelitian yang diangkat adalah mencari rute perjalanan terbaik dengan pesawat terbang dari beberapa kota yang telah disediakan oleh TSC 2.0.

3.2.2 Melakukan Studi Literatur

Setelah menemukan topik penelitian, tahap selanjutnya adalah melakukan studi literatur dengan mengumpulkan dan mempelajari kembali referensi-referensi seperti jurnal ilmiah, penelitian sebelumnya dan dokumen lainnya yang terkait dengan topik penelitian. Tujuan melakukan studi literatur adalah untuk memperdalam pemahaman dan pengetahuan tentang topik yang akan diteliti. Hasil yang diperoleh pada tahapan ini adalah algoritma yang akan digunakan untuk memecahkan permasalahan pada topik penelitian. Adapun algoritma yang akan digunakan pada penelitian ini adalah gabungan dua algoritma, yaitu *tabu search* dan *simulated annealing*.

3.2.3 Mengambil dan Memahami Data

Pada tahap ini dilakukan pengambilan data yang akan digunakan di dalam penelitian. Data yang digunakan berasal dari TSC 2.0. Setelah data berhasil didapatkan, langkah selanjutnya adalah memahami data, yaitu tipe datanya, variabel yang melekat pada data, dan jumlah data. Hasil yang diperoleh pada tahap ini adalah data final yang digunakan sebagai bahan penelitian.

3.2.4 Menyusun Model Matematis

Pada tahap ini dibuat model matematis untuk menyelesaikan permasalahan yang diangkat. Hasil yang diperoleh pada tahapan ini adalah variabel keputusan, fungsi tujuan, batasan-batasan dan model matematika. Fungsi batasan terdiri dari *hard constraint* yaitu batasan yang harus dipenuhi keseluruhan dan *soft constraint* yaitu batasan yang sebisa mungkin dapat dipenuhi.

3.2.5 Mengimplementasikan Algoritma *Tabu-Simulated Annealing*

Pada tahap ini, algoritma *tabu-simulated annealing* diimplementasikan pada tugas akhir. Algoritma *simulated annealing* diimplementasikan pada *local search* dengan menggunakan *acceptance criteria* solusi lebih baik dari solusi sebelumnya, hal ini bertujuan supaya terjadi perubahan terhadap solusi awal menjadi solusi yang lebih baik. Proses *annealing* akan dilakukan apabila solusi yang dihasilkan tidak lebih baik dari solusi awal.

3.2.6 Melakukan Uji Implementasi

Pada tahapan ini dilakukan uji coba dengan cara memasukkan dataset ke dalam algoritma *tabu-simulated annealing*. Proses uji coba dilakukan untuk melakukan verifikasi dan validasi terhadap algoritma yang telah dirancang. Selain itu, tahapan ini juga digunakan untuk mengetahui apakah algoritma yang dirancang telah sesuai dengan cara kerja dan kebutuhan. Apabila algoritma yang dirancang tidak sesuai dengan cara kerja dan kebutuhan, maka akan dilakukan evaluasi kesalahan dan perbaikan terhadap algoritma.

Pada tahapan ini juga dilakukan eksperimen terhadap parameter yang digunakan untuk mencari nilai parameter terbaik. Di dalam algoritma *tabu search* membutuhkan parameter ukuran panjang *tabu list*. Sementara itu, pada algoritma *simulated annealing* dibutuhkan parameter suhu awal, koefisien penurunan suhu, dan suhu akhir. Nilai parameter terbaik akan ditandai dengan nilai solusi yang dihasilkan lebih optimum.

3.2.7 Analisis Hasil dan Kesimpulan

Pada tahapan ini dilakukan analisis terhadap hasil percobaan yang telah dilakukan. Analisis dilakukan dengan cara mengamati hasil keluaran dari setiap iterasi, kemampuan algoritma dalam menemukan biaya terendah untuk setiap perjalanan, dan kecepatan algoritma dalam mencari rute perjalanan. Selain itu, analisis juga dilakukan dengan membandingkan performa algoritma *tabu-simulated annealing* dengan algoritma lain.

3.2.8 Menyusun Laporan Tugas Akhir

Pada tahap ini dilakukan penyusunan dokumentasi tugas akhir yang dilakukan. Hasil yang diperoleh pada tahap ini adalah Buka Laporan Tugas Akhir. Dengan adanya dokumentasi ini diharapkan dapat menjadi referensi dan dapat dikembangkan pada penelitian selanjutnya.

BAB IV
PERANCANGAN

Dalam bab ini akan dijelaskan persiapan perancangan berkaitan dengan data yang digunakan, model matematis dari fungsi tujuan dan batasan sebagai tahap persiapan implementasi algoritma.

4.1 Pemahaman Data

Dalam penelitian tugas akhir ini, data yang digunakan adalah dataset TSC 2.0. Dataset tersebut dapat diperoleh dari situs www.travellingsalesman.kiwi.com dimana dataset tersebut disediakan oleh www.kiwi.com dan digunakan dalam kompetisi *Travelling Salesman Challenge* 2.0 pada tahun 2018.

Sebagaimana telah dijeskan pada sub bab 2.2.2, terdapat 14 dataset yang berisi jumlah penerbangan dan kota yang berbeda, dataset tersebut juga memiliki karakteristik masing-masing. Berdasarkan ukuran data, dataset dibagi menjadi tiga kategori, yaitu *small*, *medium*, dan *large*. Sementara itu, berdasarkan tipe data, dataset terbagi menjadi data *artificial* dan data penerbangan asli.

Dataset yang tersedia dibagi menjadi dua tahap, yaitu tahap pengembangan dan tahap evaluasi. Dataset yang termasuk ke dalam tahap pengembangan adalah dataset 1, 6, 11, dan 14. Sisanya tergolong sebagai dataset dalam tahap evaluasi. Empat dataset yang termasuk ke dalam tahap pengembangan mewakili karakteristik dari keseluruhan dataset.

Baris pertama pada dataset menunjukkan jumlah area serta kota keberangkatan. Baris selanjutnya memberikan informasi tentang area yang tersedia beserta kota yang ada di dalamnya, setiap area setidaknya memiliki satu kota. Selain itu terdapat baris yang menunjukkan jadwal penerbangan dari sebuah kota ke kota lainnya disertai dengan biaya yang dibutuhkan.

Jadwal penerbangan yang memiliki nilai hari ke-i memiliki arti bahwa penerbangan tersebut hanya terjadi pada hari itu.

Sementara itu, jadwal penerbangan yang terjadi setiap hari ditunjukkan dengan nilai 0. Selain itu, terdapat dataset yang tidak setiap hari memiliki jadwal penerbangan dari kota i menuju kota j.

4.2 Formulasi Model Matematis

Pada penelitian tugas akhir ini, terdapat *hard constraint* dan *soft constraint* yang harus dipenuhi. Oleh karena itu diperlukan model matematika guna mempermudah implementasi algoritma dalam penelitian tugas akhir ini.

Formulasi model matematis yang dilakulan pada penelitian tugas akhir ini adalah variabel keputusan, fungsi tujuan, dan batasan-batasan

4.2.1 Variabel Keputusan

Variabel keputusan pada penelitian ini digunakan untuk mendapatkan hasil optimasi yang optimal dengan tetap memenuhi batasan-batasan yang dibuat. Variabel keputusan yang digunakan dalam permasalahan TSP adalah pemilihan kota sebagai titik keberangkatan dan kota tujuan. Variabel keputusan yang digunakan dalam penelitian ini ditunjukkan pada persamaan 6.

$$X_{ijk} = \begin{cases} 1, & \text{jika berangkat dari kota i menuju kota j pada hari ke k} \\ 0, & \text{sebaliknya} \end{cases} \quad (6)$$

Dimana:
X merupakan variabel keputusan
i = (1, 2, 3, …, m) merupakan indeks keberangkatan sampai dengan m kota
j = (1, 2, 3, …, m) sampai dengan n kota tujuan
k= (1, 2, 3, …., o) o sampai dengan hari perjalanan

4.2.2 Fungsi Tujuan

Fungsi tujuan merupakan fungsi dari variabel keputusan yang nilainya akan dimaksimumkan atau diminimalkan. Fungsi tujuan pada penelitian tugas akhir ini adalah mencari rute perjalanan dengan biaya seminimal mungkin. Persamaan (7)

menunjukkan fungsi tujuan yang digunakan dalam penelitian tugas akhir ini.

$$Minimize \sum_{k=1}^{o} \sum_{i=0}^{m} \sum_{j \neq i, j=0}^{m} C_{ijk} X_{ijk} \quad (7)$$

Dimana,

C_{ijk} = Biaya perjalanan dari kota i menuju kota j pada hari ke k
X_{ijk} = Variabel keputusan pada hari ke k
m = Jumlah kota
o = Jumlah hari

4.2.3 Batasan Masalah

Terdapat beberapa batasan (*hard constrains*) yang perlu diperhatikan agar menghasilkan solusi yang *feasible*. Pada TSC 2.0, batasan yang dimiliki berbeda dengan batasan pada TSP seperti biasa. Persamaan 7 sampai 14 merupakan model matematis dari batasan masalah pada TSC 2.0

Persamaan 7, 8, dan 9 digunakan untuk memastikan perjalanan dari area i menuju area j dan sebalinya hanya dilakukan satu kali. Y_{abc} merupakan variabel yang bernilai satu apabila terdapat perjalanan pada area i menuju area j pada hari ke c.

$$X_{ijk} \in Y_{abc} \quad i, j = 0, \dots, n \quad a, b = 0, \dots, m \quad c = 1, \dots, p \quad (7)$$

$$\sum_{c}^{m} \sum_{a=0, a \neq b}^{n} Y_{abc} = 1 \qquad b = 0, \dots, n \quad (8)$$

$$\sum_{c}^{m} \sum_{b=0, b \neq a}^{n} Y_{abc} = 1 \qquad a = 0, \dots, n \quad (9)$$

Persamaan 10 dan 11 digunakan untuk memastikan tidak ada sub rute pada solusi yang dihasilkan.

$$u_i \in \boldsymbol{Z} \qquad i = 0, \dots, n \quad (10)$$

$$u_i - u_j + nX_{ijc} \leq n - 1 \quad 1 \leq i \neq j \leq n \quad c = 1, \dots m \quad (11)$$

Persamaan 12 digunakan untuk memastikan dalam satu hari terdapat satu kali perjalanan dari kota i menuju kota j.

$$\sum_{j=0}^{n} \sum_{i=0, i \neq j}^{n} X_{ijk} = 1 \qquad k = 1, \dots, o \tag{12}$$

Persamaan 13 dan 14 digunakan untuk memastikan perjalanan akan berakhir pada kota yang memiliki area sama dengan area pada kota pertama. Variabel T_d akan berisi area yang akan dikunjungi pada hari ke d.

$$Y_{abc} \in T_d \tag{13}$$

$$T_0 = T_d \tag{14}$$

4.3 Rancangan Algoritma *Tabu-Simulated Annealing*

Pada bagian ini akan dijelaskan rancangan algoritma *tabu-simulated annealing* mulai dari pembentukan *initial solution* sampai menghasilkan solusi optimum.

Sebelum menerapkan algoritma *tabu-simulated annealing*, terlebih dahulu harus membuat *initial solution*. *Initial solution* dibuat dengan cara mengacak kota maupun area sehingga menghasilkan rute awal. Selanjutnya rute awal akan melewati *low level heuristic*, yaitu *swap* dan *move*. Algoritma *tabu-simulated annealing* digunakan setelah rute awal melewati *low level heuristic* untuk dioptimasi sehingga mencapai fungsi tujuan yang diharapkan.

4.3.1 Pembentukan *Initial Solution*

Initial solution merupakan rute awal yang dibentuk dengan cara mengambil kota secara acak sesuai dengan jadwal penerbangan yang tersedia. Untuk memudahkan dalam membuat *initial solution*, dibuat *graph* yang berisikan *node* dan *edge*. *Node* digunakan untuk menyimpan kota yang terdapat pada hari ke-i. Sementara itu, *edge* digunakan untuk menyimpan jadwal penerbangan ke kota selanjutnya.

4.3.2 Penerapan Algoritma *Tabu-Simulated Annealing*

Penerapan algoritma *tabu-simulated annealing* bertujuan untuk mengoptimasi *initial solution* yang dihasilkan pada tahap sebelumnya. Di dalam algoritma ini, terdapat empat *low level heuristic* yaitu *swap* 2 kota, *swap* 3 kota, *swap* 4 kota dan *move* 1 kota. *Low level heuristic* akan diterapkan pada *initial solution* sehingga dapat menghasilkan solusi sementara.

Selanjutnya solusi sementara akan dibandingkan dengan *initial solution*. Apabila solusi sementara lebih optimal daripada *initial solution*, maka solusi sementara diterima sebagai solusi baru. Namun apabila nilai solusi sementara lebih buruk daripada solusi baru, maka solusi sementara akan melewati proses *annealing*. *Tabu list* akan digunakan apabila solusi sementara yang dihasilkan tidak lebih baik daripada *initial solution* setelah melewati proses ini. *Tabu list* akan menyimpan *low level heuristic* yang digunakan untuk membuat solusi sementara.

Untuk mendapatkan solusi optimum diperlukan nilai parameter yang tepat. Oleh karena itu, diperlukan eksperimen pengujian parameter dengan menggunakan beberapa dataset. Pengujian parameter dilakukan dengan cara membuat kombinasi nilai parameter yang berbeda dalam setiap eksperimen. Kombinasi parameter terbaik akan digunakan untuk mengoptimasi keseluruhan dataset.

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Pada bab ini membahas mengenai implementasi algoritma tabu-simulated annealing yang digunakan dalam optimasi rute perjalanan. Implementasi ini menggunakan *tools* IDE Netbeans dan bahasa pemrograman Java

5.1 Membaca Dataset

Dalam membaca dataset digunakan *BufferedReader*. Baris pertama pada dataset menunjukkan jumlah area dan kota awal. Informasi di dalam dataset dibagi menjadi tiga bagian, yaitu area, kota, dan jadwal penerbangan.

Jumlah area yang terdapat di dalam dataset disimpan dalam variabel jumlahArea. Sementara itu kota awal keberangkatan disimpan dalam variabel kotaAsal. Setiap area dan kota yang terdapat di dalam sebuah area disimpan ke dalam variabel yang berbeda. Kode 5.1 digunakan untuk membaca dataset dan menyimpannya ke dalam variabel.

```
public static void input() throws FileNotFoundException, IOException {
    String dataset = "dataset/" + 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];
```

Kode 5.1 Membaca Dataset

Variabel day digunakan untuk menyimpan jadwal penerbangan yang tersedia beserta biayanya. Kode 5.2 berfungsi untuk menyimpan jadwal penerbangan.

```
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 Menyimpan Jadwal Penerbangan

5.2 Membuat Kelas *Cost*

Kelas *cost* digunakan untuk mencari biaya yang dibutuhkan dalam berpindah dari kota i menuju kota j. Di dalam kelas *cost* terdapat *array* dua dimensi yang menyimpan biaya yang dibutuhkan untuk berangkat dari kota i menuju kota j. Kode 5.3 merupakan kelas *cost*.

```
public class Cost {
    int cost[][];

    public Cost(int jumlah) {
        cost = new int[jumlah][jumlah];
    }

    public void add(int a, int b, int c) {...9 lines }

    public boolean check(int a, int b) {...7 lines }

    public int get(int a, int b) {...3 lines }
}
```

Kode 5.3 *Cost*

5.3 Membuat *Node*

Node dibuat untuk menyimpan kumpulan kota dan area dalam bentuk indeks serta daftar kota yang dapat dijadikan sebuah rute. Di dalam kelas Node terdapat beberapa fungsi seperti menambahkan, menghapus, melihat, mengecek, dan mengubah isi variabel pada graph.

```
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) {...8 lines }

public void gantiIsiKotaBerikutnya(List<Node> a) {...6 lines }

public void add(Node a) {...5 lines }

public void remove(int a) {...8 lines }

public int getKotaBerikutnya(int a) {
    return kotaBerikutnya.get(a).indexKota;
}

public Node getRandom(int a[], int b, int c[], int aa) {...52 lines }
```

Kode 5.4 Membuat *Node*

5.4 Membuat *Graph*

Graph digunakan untuk menyambungkan setiap kota menjadi kota keberangkatan dan kota tujuan. *Graph* dibuat dalam bentuk *arraylist* di dalam *array*. Jumlah *array* yang disimpan sama dengan jumlah hari yang dibutuhkan untuk menyelesaikan rute perjalanan. *Array* memiliki indeks hari dan rute yang jadwalnya sama dengan indeks hari akan dimasukkan ke dalam *arraylist*.

```
graph = new List[namaArea.length + 1];
//menambahkan kota asal
graph[0] = new ArrayList<Node>();
graph[0].add(new Node(indexKotaAsal, indexAreaAsal));
//menambahkan kota berakhirnya perjalanan
graph[namaArea.length] = new ArrayList<Node>();
for (int i = detailArea[indexAreaAsal][1]; i > 0; i--) {
    graph[namaArea.length].add(new Node(detailArea[indexAreaAsal][0] - i + 1, indexAreaAsal));
}
```

Kode 5.5 Membuat *Graph*

Semua kota yang masuk ke dalam *graph* akan disambungkan dengan memperhatikan jadwal penerbangan yang dimilikinya.

```
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++) {
                if (kotaAkhir[0].get(k) == graph[i].get(j).indexKota) {
                    test = false;
                    break;
                }
            }
            if (test) {
                for (int j = 0; j < graph[i - 1].size(); j++) {
                    if (kotaAwal[0].contains(graph[i - 1].get(j).indexKota)) {
                        if (day[0].check(graph[i - 1].get(j).indexKota, kotaAkhir[0].get(k))) {
                            boolean test2 = false;
                            if (kotaAwal[0].contains(kotaAkhir[0].get(k))) {
                                test2 = true;
                            }
                            if (bahanGraph[i] != null) {
                                if (kotaAwal[i].contains(kotaAkhir[0].get(k))) {
                                    test2 = true;
                                }
                            }
                            if (test2) {
                                graph[i].add(new Node(kotaAkhir[0].get(k), cariArea(kotaAkhir[0].get(k))));
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
```

Kode 5.6 Menyambungkan *Graph*

Di dalam sebuah *graph* tidak semua kota memiliki sambungan penerbangan pada hari sebelumnya atau setelahnya. Oleh karena itu diperlukan pengecekan untuk memastikan adanya jadwal penerbangan pada kota tersebut.

Untuk mengecek jadwal pada hari sebelumnya dilakukan dengan mengecek jadwal yang memiliki indeks i-1 dan 0. Sementara itu pengecekan untuk jadwal pada hari setelahnya dilakukan dengan mengecek jadwal yang memiliki indeks i+1 dan 0.

Jadwal penerbangan yang lolos pengecekan akan dimasukkan ke dalam *graph* untuk dijadikan sebuah rute.

```
for (int k = 0; k < bahanGraph[pengganti].size(); k++) {
    //bandingkan dengan kemarin
    boolean test3 = true;
    //check apakah di graph sudah ada?
    for (int j = 0; j < graph[i].size(); j++) {
        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;
                }
            }
            if (t) {
                boolean test1 = false;
                int pengganti2;
                if (pengganti == 0) {
                    pengganti2 = pengganti;
                } else {
                    pengganti2 = i + 1;
                }
                if (kotaAwal[pengganti2].contains(bahanGraph[pengganti].get(k)[1])) {
                    test1 = true;
                }
                if (pengganti2 != pengganti) {
                    if (bahanGraph[0] != null) {
                        if (kotaAwal[0].contains(bahanGraph[pengganti].get(k)[1])) {
                            test1 = true;
                        }
                    }
                }
                //tambah node jika sesuai ketentuan
                if (test1) {
                    graph[i].add(new Node(bahanGraph[pengganti].get(k)[1],
                            cariArea(bahanGraph[pengganti].get(k)[1])));
                    break;
                }
            }
        }
    }
}
```

Kode 5.7 Mengecek Jadwal Penerbangan

5.5 Mengoptimasi *Graph*

Optimasi *graph* dilakukan untuk menyeleksi kembali kota yang masuk ke dalam *graph* dengan cara mengecek banyaknya *graph*-2 hingga *graph* pertama. Kota yang tidak memiliki rute

selanjutnya akan dihapus dari *graph*. Dengan demikian semua kota di dalam *graph* dapat dihubungkan menjadi sebuah rute.

```
public static void optimizeGraph() {
    //check kembali kota kota yang tidak ada rute sambungan
    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;
                    }
                }
                if (day[0] != null) {
                    if (day[0].check(graph[i].get(j).indexKota, graph[i + 1].get(k).indexKota)) {
                        cek = false;
                        break;
                    }
                }
            }
            if (cek) {
                graph[i].remove(j);
                j--;
            }
        }
    }
```

Kode 5.8 Mengoptimasi *Graph*

5.6 Membuat Rute

Kota yang terdapat di dalam *graph* akan dibuat menjadi sebuah rute perjalanan. Rute disimpan dalam variabel tourKota dalam bentuk *array*. Kota awal dan kota akhir perjalanan akan diinisialisasi terlebih dahulu. Untuk mengisi rute perjalanan dilakukan *random* terhadap kota yang ada di dalam *graph* berdasarkan hari. Kode 5.9 digunakan untuk inisialisasi kota awal dan kota akhir. Kode 5.10 digunakan untuk mengisi rute kota.

```
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];
```

Kode 5.9 Inisialisasi Kota Awal Dan Kota Akhir

```
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;
        }
    }
    if (i == graph.length - 3) {
        if (graph[i].get(index).checkKotaBerikutnya(b.indexKota)) {
            tourKota[i] = a.indexKota;
            tourArea[i] = a.indexArea;
        } else {
            tourKota[i] = -1;
        }
    } else {
        tourKota[i] = a.indexKota;
        tourArea[i] = a.indexArea;
    }
    if (tourKota[i] == -1) {
        break;
    }
}
```

Kode 5.10 Mengisi Rute Secara Acak

5.7 Membuat *Initial Solution*

Pembuatan *initial solution* digunakan sebagai rute awal sebelum dilakukan optimasi. *Initial solution* disimpan dalam variabel rute.

```
rute = new Tour(graph, Kota, day, true);
```

Kode 5.11 Menyimpan *Initial Solution*

Initial solution dibuat dengan cara mengambil kota yang ada di dalam *graph* secara acak kemudian menyambungkannya menjadi sebuah rute.

5.8 Menghitung biaya

Biaya yang dibutuhkan dalam setiap rute perjalanan perlu diketahui. Solusi akhir paling optimal adalah rute yang memiliki biaya terkecil. Biaya dihitung dari mulai rute awal hingga terakhir. Untuk menghitung biaya yang dibutuhkan adalah dengan menggunakan fungsi *calculateFitness()*.

```
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;
            }
        }
    } catch (Exception e) {
        fitness = -1;
    }
}
```

Kode 5.12 Menghitung Biaya

5.9 Membuat *Low Level Heuristic*

Pembuatan *low level heuristic* digunakan untuk memindahkan urutan solusi sementara sehingga dapat menghasilkan solusi baru. Solusi baru yang dihasilkan tidak selamanya memiliki biaya yang lebih optimal daripada solusi sementara, sehingga *low level heuristic* yang digunakan dapat mempengaruhi hasil solusi baru.

Pada penelitian tugas akhir ini menggunakan empat *low level heuristic* yang berbeda. Berikut ini merupakan implementasi *low level heuristic*.

5.9.1 *Swap* 2 Kota

Low level heuristic swap 2 kota digunakan untuk menukar urutan dua kota pada solusi sementara. Pemilihan dua kota yang akan ditukar urutannya dilakukan secara acak. Kota sebagai titik keberangkatan pertama dan kota akhir pada solusi sementara tidak akan dipilih sebagai kota yang akan ditukar posisinya. Kode program *swap* 2 kota ditunjukkan

```
public static void swap2Kota(int i, boolean s) {
    Random r = new Random();
    int a = 0, b = 0;
    while (a == b) {
        a = r.nextInt(jumlahArea);
        b = r.nextInt(jumlahArea);
        if (a < 1) {
            a = 1;
        }
        if (b < 1) {
            b = 1;
        }
    }
    rute.swap(a, b, s);
}
```

Kode 5.13 *Swap* 2 Kota

```
if((costAwal>costAkhir || !s )&& test){
    int c = tourKota[a];
    tourKota[a] = tourKota[b];
    tourKota[b] = c;
    fitness=fitness+(costAkhir-costAwal);

}
```

Kode 5.14 Membandingkan Biaya Akhir

Kode 5.12 digunakan untuk mengecek biaya perjalanan. Apabila biaya perjalanan pada solusi baru lebih kecil daripada solusi sementara, maka akan dilakukan *swap* pada kota terpilih sehingga menghasilkan solusi baru.

5.9.2 *Swap* 3 Kota

Low level heuristic swap 3 kota digunakan untuk menukar urutan tiga kota secara bersamaan. Pemilihan kota yang akan

dituka dilakukan secara acak. Kode 5.13 merupakan kode program yang digunakan untuk menukar urutan 3 kota.

```
public static void swap3Kota(int i, boolean s) {
    Random r = new Random();
    int a = 0, b = 0;
    int fits = rute.getFitness();
    int tours[] = rute.tourKota.clone();
    while (a == b) {
        a = r.nextInt(jumlahArea);
        b = r.nextInt(jumlahArea);
        if (a < 1) {
            a = 1;
        }
        if (b < 1) {
            b = 1;
        }
        rute.swap(a, b, false);
        a = r.nextInt(jumlahArea);
        if (a < 1) {
            a = 1;
        }
        rute.swap(a, b, false);
        if (s) {
            if (rute.getFitness() > fits) {
                rute.setTour(tours);
                rute.fitness = fits;
            }
        }
    }
}
```

Kode 5.15 *Swap* 3 Kota

5.9.3 *Swap 4* Kota

Low level heuristic swap 4 kota digunakan untuk menukar urutan empat kota. Kota yang akan ditukar urutannya dipilih secara acak. Kota awal keberangkatan dan kota akhir perjalanan tidak akan dipilih sebagai kota yang ditukar posisinya.

```
public static void swap4Kota(int i, boolean s) {
    Random r = new Random();
    int a = 0, b = 0;
    int fits = rute.getFitness();
    int tours[] = rute.tourKota.clone();
    while (a == b) {
        a = r.nextInt(jumlahArea);
        b = r.nextInt(jumlahArea);
        if (a < 1) {
            a = 1;
        }
        if (b < 1) {
            b = 1;
        }
        rute.swap(a, b, false);
        a = r.nextInt(jumlahArea);
        if (a < 1) {
            a = 1;
        }
        rute.swap(a, b, false);
        b = r.nextInt(jumlahArea);
        if (b < 1) {
            b =1;
        }
        rute.swap(a, b, false);
        if (s) {
            if (rute.getFitness() > fits) {
                rute.setTour(tours);
                rute.fitness = fits;
            }
        }
```

Kode 5.16 *Swap* 4 Kota

5.9.4 *Move 1* Kota

Low level heuristic move 1 kota digunakan untuk menggeser urutan 1 kota yang terpilih. Pemilihan kota yang akan digeser dilakukan secara acak, selain itu penentuan urutan kota yang digeser juga dilakukan pengacakan.

```
public static Tour moves(Tour a) {
    Tour baru = new Tour(graph, Kota, day);
    int tourBaru[] = new int[graph.length];

    for (int i = 0; i < tourBaru.length; i++) {
        tourBaru[i] = -1;
    }

    int startPos = (int) (Math.random() * a.tourKota.length-1);
    int endPos = (int) (Math.random() * a.tourKota.length-1);

    int tourPl[] = a.tourKota;
    tourBaru[startPos] = tourPl[endPos];
```

Kode 5.17 Memilih Index Kota Dan Index Urutan

```
int in = 0;
for (int i = 0; i < a.tourKota.length - 1; i++) {
    boolean cek = true;
    for (int j = 0; j <= tourBaru.length-1; j++) {
        if (tourPl[i] == tourBaru[j]) {
            cek = false;
            break;
        }
    }
    if(in>tourBaru.length-1){
        break;
    }

    if (cek) {
        while (true) {
            if (tourBaru[in] < 0) {
                tourBaru[in] = tourPl[i];
                in++;
                break;
            } else {
                in++;
            }
        }
    }
}
tourBaru[tourBaru.length-1]=tourPl[tourBaru.length-1];
baru.setTour(tourBaru);
return baru;
```

Kode 5.18 *Moves* 1 Kota

5.10 Implementasi Algoritma *Tabu-Simulated Annealing*

Algoritma *tabu-simulated annealing* diimplementasikan untuk mendapatkan nilai solusi paling optimum. *Low level heuristic* diterapkan setelah mendapatkan *initial solution*. *Low level heuristic* yang akan diterapkan dipilih secara acak. Sealanjutnya akan didapatkan solusi sementara.

Selanjutnya solusi sementara akan dibandingkan dengan *initial solution*. Apabila solusi sementara lebih optimal daripada *initial solution*, maka solusi sementara diterima sebagai solusi baru.

Namun apabila nilai solusi sementara lebih buruk daripada solusi baru, maka solusi sementara akan melewati proses *annealing*. *Tabu list* akan digunakan apabila solusi sementara yang dihasilkan tidak lebih baik daripada *initial solution* setelah melewati proses ini. *Tabu list* akan menyimpan *low level heuristic* yang digunakan untuk membuat solusi sementara.

Namun, sebelumnya akan dicek apakah masih tersedia slot di dalam *tabu list*. Apabila slot *tabu list* sudah penuh, maka akan dikeluarkan isi *tabu list* yang masuk lebih dulu. *Low level heuristic* yang sudah masuk ke dalam *tabu list* tidak akan digunakan kembali pada proses selanjutnya sampai dengan *low level heuristic* tersebut keluar dari *tabu list*.

Solusi sementara dapat diterima menjadi solusi baru apabila biaya lebih rendah daripada biaya pada *initial solution*, atau nilai acak lebih besar daripada nilai persamaan Boltzman dan *low level heuristic* yang digunakan tidak terdapat dalam *tabu list*.
Kode 5.17 merupakan penerapan algoritma *tabu-simulated annealing*.

```
public static void tabuSA(int fittest, int a){
    int delta = newSolution - fittest;
    double random = Math.random();
    if (fittest > newSolution && newSolution > 0) {
        fittest = newSolution;
        System.out.println(fittest);
    }
    else {
        if (Math.exp(-(Math.abs(delta)) / t0 ) > Math.random()&& newSolution > 0) {
        fittest = newSolution;
            System.out.println(fittest);
        } //jika lebih buruk
        else{
            if(TL.remainingCapacity() == 0){
                TL.remove();
                TL.add(a);
            }
            else{
                TL.add(a);
            }
        }
    }
    t0 *= coolingRate;
}
```

Kode 5.19 Algoritma *Tabu-Simulated Annealing*

5.11 Implementasi Algoritma *Great Deluge*

Algoritma *great deluge* digunakan sebagai algoritma pembanding dalam menghasilkan sebuah solusi. Kode 5.20 merupakan variabel yang digunakan dalam algoritma *great deluge*. Nilai variabel estimateCost didapatkan dari solusi optimal pada setiap dataset dengan menggunakan algoritma *tabu-simulated annealing*. Variabel numOfIte sama dengan jumlah iterasi eksperimen skenario 7 pada algoritma tabu-simulated annealing.

```
String initialSolution = initialTour;
initialCost = fittest;
String bestSolution = initialSolution;
estimateCost = 7718;
numOfIte = 46051691;
level = initialCost;
decreasingRate = (fittest - estimateCost) / numOfIte;
iteration = 0;
not_improving_counter = 0;
not_improving_counter_GDA = 100;
```

Kode 5.20 Parameter Algoritma *Great Deluge*

Low level heuristic yang digunakan dalam algoritma pembanding ini adalah *swap* 2 kota. Selanjutnya biaya pada solusi sementara akan dibandingkan dengan initialCost, apabila biaya solusi sementara kurang dari biaya initialCost atau kurang dari level, maka solusi sementara tersebut diterima sebagai solusi baru. Kode 5.21 merupakan algoritma pembanding great deluge.

```
while   (iteration < numOfIte){

    swap2Kota(0, true);
    baru = rute;
    costBaru = baru.getFitness();

    if (costBaru < initialCost) {
        initialCost = costBaru;
        bestSolution = baru.getTour();
        not_improving_counter = 0;
    }
    else {
        if (costBaru <= level) {
            initialCost = costBaru;
            not_improving_counter = 0;
        }
        else {
            not_improving_counter++;
            if (not_improving_counter == not_improving_counter_GDA) {
                break;
            }
        }
    }
    level = level - decreasingRate;
    iteration++;
    a++;
}
```

Kode 5.21 Algoritma *Great Deluge*

Halaman sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil yang didapatkan dari pengerjaan tugas akhir ini serta pembahasan secara keseluruhan yang didapatkan dari penelitian.

6.1 Data Uji Coba

Data yang digunakan sebagai uji coba adalah dataset TSC 2.0 bagian 1, 6, 11, dan 14 yang digunakan langsung dalam penelitian tugas akhir ini.

6.2 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam penelitian ini terdiri dari perangkat keras dan perangkat lunak. Spesifikasi perangkat keras yang digunakan dalam implementasi ditunjukkan pada Tabel 6.1.

Tabel 6.1 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi
Jenis	Laptop
Processor	Intel(R) Core™ i3-3227U @ 1.90 GHz (4 CPUs) ~1.90GHz
RAM	6 GB
Hard Disk Drive	500 GB

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

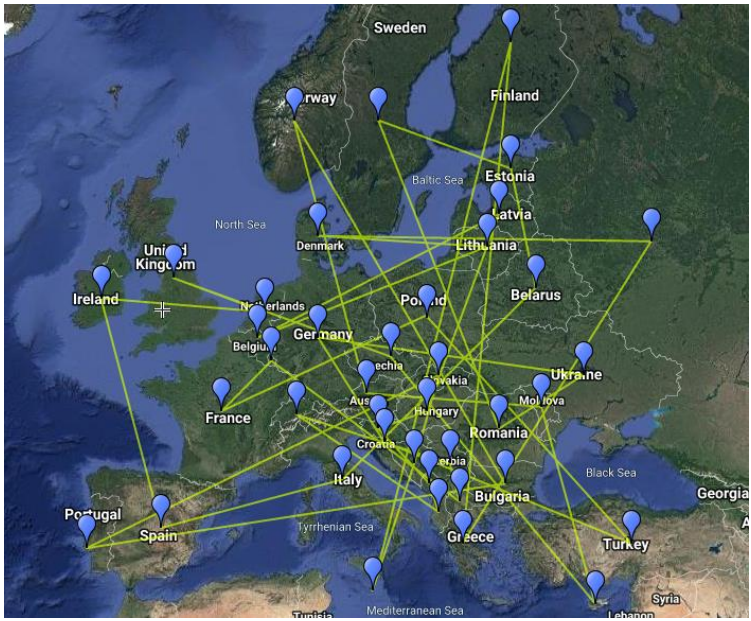
Tabel 6.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	Fungsi
Windows 10 64 bit	Sistem Operasi
Netbeans 8.2	Implementasi Algoritma
Notepad++	Pengolahan Data dan Hasil

Perangkat Lunak	Fungsi
Microsoft Excel 365 Pro Plus	Pengolahan Hasil Uji Coba
Microsoft Word 365 Pro Plus	Penulisan Laporan

6.3 Hasil Uji Coba *Initial Solution*

Pada penelitian tugas akhir ini, rute awal sebagai inisiasi solusi dibuat dengan cara memilih urutan kota secara acak dan selanjutnya kota-kota tersebut dihubungkan menjadi sebuah rute baru. *Initial solution* yang terbentuk harus memenuhi batasan yang berlaku. Selama solusi dapat memenuhi batasan, solusi tersebut tetap akan diterima sebagai solusi awal meskipun memiliki biaya yang besar. Gambar 6.1 menunjukkan contoh *initial solution* rute perjalanan pada dataset 4, biaya yang dibutuhkan untuk menyelesaikan perjalanan tersebut sebesar 50162.

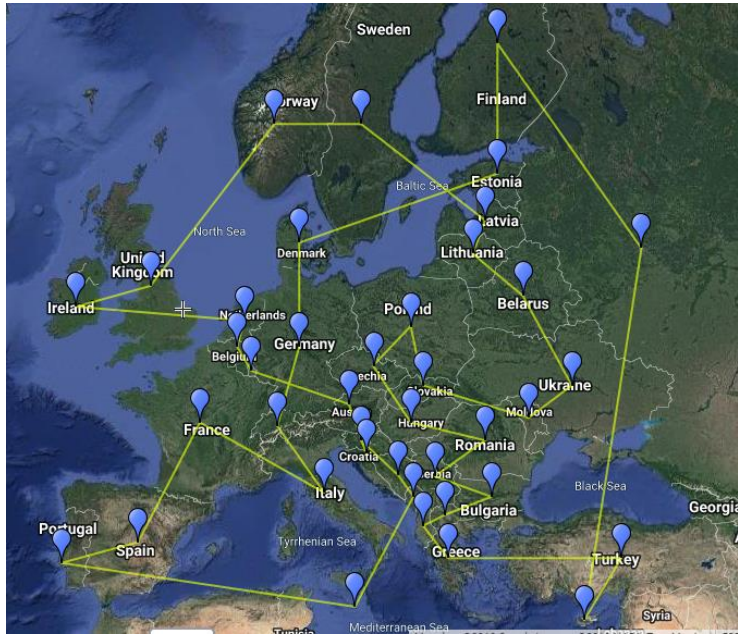


Gambar 6.1 Contoh *Initial Solution* Dataset 4

6.4 Uji Coba Implementasi Algoritma

Algoritma *tabu-simulated annealing* dilakukan untuk mengoptimasi rute perjalanan yang telah diinisiasi sebelumnya. Di dalam algoritma *tabu-simulated annealing* terdapat parameter-parameter yang dapat diubah untuk mendapatkan solusi yang optimal.
Selain itu, terdapat empat *low level heuristic* yang digunakan dalam algoritma *tabu-simulated annealing*. *Low level heuristic* tersebut adalah *swap* 2 kota, *swap* 3 kota, *swap* 4 kota, dan *move* 1 kota.

Gambar 6.2 merupakan visualisasi rute perjalanan hasil optimasi *initial solution* pada dataset 4 dengan menggunakan algoritma *tabu-simulated annealing*. Biaya yang dibutuhkan untuk menyelesaikan perjalanan adalah 20227.



Gambar 6.2 Contoh Hasil Optimasi Dataset 4

6.5 Hasil Eksperimen Implementasi Algoritma

Eksperimen dilakukan dengan mengubah parameter-parameter yang dapat mempengaruhi hasil akhir pada algoritma *tabu-simulated annealing*. Parameter yang berpengaruh pada algoritma *tabu-simulated annealing* adalah suhu awal, *cooling rate*, panjang *tabu list*, dan suhu akhir yang digunakan sebagai *stopping condition*. Tujuan melakukan eskperimen adalah untuk mendapatkan nilai parameter yang menghasilkan solusi optimal.

Untuk mempermudah dalam melakukan eksperimen ini, maka dibuat beberapa skenario. Setiap skenario eksperimen dijalankan sebanyak 11 kali untuk satu dataset. Nilai parameter yang digunakan pada setiap skenario didasarkan pada percobaan parameter secara acak. *Low level heuristic* yang digunakan pada eksperimen ini adalah *swap* 2 kota, *swap* 3 kota, *swap* 4 kota, dan *move* 2 kota.

Tidak semua dataset dijalankan dalam skenario ini, eksperimen hanya dilakukan terhadap dataset yang dapat mewakili karakter dari keseluruhan dataset. Oleh karena itu dataset yang digunakan dalam skenario ini adalah 1, 6, 11, dan 14.

Tabel 6.3 menunjukkan daftar parameter yang digunakan. Sementara daftar nilai parameter pada setiap skenario ditunjukkan oleh Tabel 6.4.

Tabel 6.3 Daftar Parameter

Parameter	Keterangan
LLH	Jumlah *low level heuristic* yang digunakan
T_0	Suhu awal algoritma *simulated annealing*
T_1	Suhu akhir algoritma *simulated annealing*
α	Penurunan suhu
TL	Panjang *tabu list* solusi algoritma *tabu search*

Tabel 6.4 Daftar Skenario

Skenario	T0	T1	α	TL
1	100	0.1	0.95	2
2	10000	0.1	0.9025	2
3	100000000	0.1	0.81450625	2
4	1000000000	0.1	0.9995	2
5	1000000000	0.1	0.9995	3
6	1000000000	0.1	0.999995	3
7	1000000000	0.1	0.9999995	3
8	1500000000	0.1	0.999995	3
9	1500000000	0.001	0.99999995	3
10	1500000000	0.001	0.999995	2
11	1000000000	0.1	0.9999995	2
12	1000000000	0.001	0.9999995	3

6.5.1 Skenario 1

Pada skenario 1, percobaan dilakukan menggunakan parameter suhu awal sebesar 100, suhu akhir 0.1, dan *cooling rate* 0.95. Semua jenis *low level heuristic* dipakai dalam skenario ini serta panjang *tabu list* yang digunakan adalah 2. Adapun hasil eksperimen 1 ditunjukkan oleh Tabel 6.5.

Tabel 6.5 Biaya Hasil Skenario 1

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	4299	11788	95947	233047
2	1533	11862	102617	229980
3	1729	11357	101914	228777
4	3921	11756	90786	229717
5	3765	12197	101224	229151
6	1431	12041	100156	225641
7	2290	12100	103722	227104
8	4104	12057	97381	227851
9	2321	11752	101905	233377

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
10	3008	11899	101663	230774
11	2373	11861	101438	231818
Worst	4299	12197	103722	233377
Average	2797.636	11879.09	99886.64	229748.8
Best	1431	11357	90786	225641
Time (ms)	962.54	67708.27	9709.27	68726.91

Berdasarkan parameter yang digunakan pada skenario 1, terjadi 135 iterasi dalam setiap proses untuk menghasilkan solusi akhir. Semakin besar dataset yang diuji, *running time* algoritma semakin besar.

6.5.2 Skenario 2

Pada skenario 2, nilai parameter suhu awal ditingkatkan sementara itu nilai parameter *cooling rate* dikecilkan. Dengan adanya perubahan skenario ini diharapkan jumlah iterasi yang dilakukan semakin banyak. Adapun hasil eksperimen 2 ditunjukkan oleh Tabel 6.6

Tabel 6.6 Biaya Hasil Skenario 2

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	4515	12250	103820	235121
2	3889	11698	101118	234502
3	1824	11703	97870	229931
4	6271	12571	102344	226805
5	9563	11659	99047	222967
6	3838	11821	102382	228525
7	1630	12156	100111	227977
8	1526	12286	99701	235776
9	4050	12103	101855	227860
10	2645	12257	105867	223840

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
11	3767	11891	102041	233589
Worst	9563	12571	105867	235776
Average	3956.182	12035.91	101468.7	229717.5
Best	1526	11659	97870	222967
Time (ms)	2176.09	124251.2	8430.18	29886.64

Tabel 6.6 menunjukkan hasil eksperimen skenario 2. Jumlah iterasi yang terjadi pada skenario 2 adalah 113, lebih kecil dari pada skenario 1. Hal ini berdampak terhadap solusi yang dihasilkan belum maksimal. Rata-rata solusi yang dihasilkan lebih buruk daripada skenario 1.

6.5.3 Skenario 3

Pada skenario 3, suhu awal ditambah menjadi 100.000.000 dan *cooling rate* diubah menjadi 0.81450625. Panjang *tabu list* dan jumlah *low level heuristic* yang digunakan masih sama dengan skenario sebelumnya. Hasil eksperimen ditunjukkan pada Tabel 6.7.

Tabel 6.7 Biaya Hasil Skenario 3

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1563	12126	102099	231384
2	1846	11636	100101	231910
3	2366	12020	105894	222952
4	3996	11574	93815	225556
5	4849	11890	97039	234330
6	3957	11944	99849	234409
7	5571	11509	100477	231972
8	6702	12197	103997	237246
9	3634	12340	100147	232915
10	5150	11309	108983	232500

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
11	2322	11339	102658	233426
Worst	6702	12340	108983	237246
Average	3814.182	11807.64	101369	231690.9
Best	1563	11309	93815	222952
Time (ms)	1963.09	243142.3	10671.91	36091.64

Eksperimen skenario 3 menghasilkan 102 iterasi untuk setiap proses. Pada skenario ini, jumlah iterasi lebih sedikit daripada skenario sebelumnya, rata-rata solusi yang dihasilkan juga semakin buruk.

6.5.4 Skenario 4

Pada skenario 4, parameter suhu awal dinaikkan menjadi 1000000000 dan *cooling rate* diubah menjadi 0.9995. Sementara itu, suhu akhir, panjang *tabu list* dan *low level heuristic* yang digunakan tidak berubah dari skenario sebelumnya. Adapun hasil dari eksperimen 4 ditunjukkan oleh Tabel 6.8.

Tabel 6.8 Biaya Hasil Skenario 4

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1481	5803	89023	222538
2	1561	5360	84510	225755
3	2290	5715	83441	220631
4	2285	4867	83764	226997
5	2437	5709	86749	223268
6	1396	5824	83886	227724
7	1472	5907	85849	229139
8	1481	5736	86404	228900
9	1396	5836	84914	223057
10	2290	6001	87415	221339

11	1396	6076	81378	225332
Worst	2437	6076	89023	229139
Average	1771.364	5712.182	85212.09	224970.9
Best	1396	4867	81378	220631
Time (ms)	1627.27	147091.5	16351.45	37915.73

Tabel 6.8 menunjukkan bahwa rata-rata solusi akhir lebih optimal daripada skenario sebelumnya. Jumlah iterasi yang terjadi pada setiap proses optimasi adalah 46.041 iterasi. Jauh lebih banyak daripada iterasi pada skenario 1, 2, dan 3.

6.5.5 Skenario 5

Pada skenario 5, parameter yang diubah hanya panjang *tabu list*. Skenario ini dilakukan untuk mengetahui apakah panjang *tabu list* mempengaruhi hasil solusi akhir. Hasil eksperimen skenario 5 ditunjukkan pada Tabel 6.9.

Tabel 6.9 Biaya Hasil Skenario 5

Percobaan	**Dataset 1**	**Dataset 6**	**Dataset 11**	**Dataset 14**
1	2209	5633	80932	224253
2	1396	5540	77639	226911
3	2290	5828	79713	212678
4	2209	5239	83671	217136
5	2437	6012	77691	222651
6	1396	5649	78363	222485
7	2290	5886	81118	216844
8	2209	5887	80796	215355
9	1472	5514	82588	219894
10	1396	4902	80362	216877
11	2209	5783	81640	225208
Worst	2437	6012	83671	226911
Average	1955.727	5624.818	80410.27	220026.5
Best	1396	4902	77639	212678

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
Time (ms)	1259	87786.5	8297.09	37746.18

Tabel 6.9 menunjukkan hasil solusi akhir yang diperoleh tidak jauh berbeda dari skenario 4. Namun, rata-rata pada dataset 11 dan 14 jauh lebih optimal daripada skenario 5. Jumlah iterasi yang terjadi adalah 46.041.

6.5.6 Skenario 6

Pada skenario 6, dilakukan perubahan pada parameter *cooling rate* dari yang awalnya 0.999995 menjadi 0.9999995. Adapun hasil eksperimen skenario 6 ditunjukkan oleh Tabel 6.10.

Tabel 6.10 Biaya Hasil Skenario 6

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	2255	4661	72840	222693
2	1396	4540	74946	222594
3	1396	4914	76064	219815
4	1396	4902	77837	216921
5	1481	4383	77485	214306
6	2209	4738	74260	223434
7	1481	5244	70402	214087
8	1481	4524	72444	224029
9	2290	4684	74300	213723
10	1481	4145	69240	204451
11	2209	4961	75057	217677
Worst	2290	5244	77837	224029
Average	1734.091	4699.636	74079.55	217611.8
Best	1396	4145	69240	204451
Time (ms)	5654.18	126481	65898.45	231632.3

Tabel 6.10 menunjukkan bahwa skenario 6 menghasilkan rata-rata solusi akhir lebih optimal dari pada skenario yang telah

dilakukan sebelumnya. Jumlah iterasi yang terjadi pada setiap proses optimasi adalah 4.605.159.

6.5.7 Skenario 7

Eksperimen pada skenario 7, dilakukan dengan mengubah parameter *cooling rate* menjadi 0.9999995. Parameter lainnya tidak mengalami perubahan sebagaimana ditunjukkan pada Tabel 6.4. Hasil eksperimen dari skenario 7 ditunjukkan pada Tabel 6.11.

Tabel 6.11 Biaya Hasil Skenario 7

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1481	4450	68864	210146
2	1396	4428	65517	203770
3	2209	4635	67842	203037
4	1396	3901	72275	208322
5	2437	4139	73882	208947
6	1396	4708	67593	211367
7	1396	4571	74517	204686
8	1526	4920	72565	216869
9	2366	4383	71965	213083
10	1396	4576	75698	215933
11	2437	4041	74440	212541
Worst	2437	4920	75698	216869
Average	1766.909	4432	71378	209881.9
Best	1396	3901	65517	203037
Time (ms)	64791	418313.1	381567.2	1862812

Eksperimen yang dilakukan pada skenario 7 menunjukkan adanya peningkatan hasil optimasi pada dataset 14 daripada skenario 6. Hanya dataset 1 yang memiliki penurunan nilai hasil akhir jika dibandingkan dengan skenario 6. Jumlah iterasi yang terjadi dalam setiap proses pada skenario 7 adalah 46.051.691.

6.5.8 Skenario 8

Pada skenario 8 parameter suhu awal dinaikkan menjadi 1.500.000.000 dan *cooling rate* diubah menjadi 0.999995. Sementara itu, tidak dilakukan perubahan terhadap parameter lainnya sebagaimana ditunjukkan pada Tabel 6.4. Hasil eksperimen skenario 8 ditunjukkan pada Tabel 6.12.

Tabel 6.12 Biaya Hasil Skenario 8

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	2290	4754	80845	224677
2	1936	5071	74050	215902
3	2285	5023	73332	224243
4	1396	4999	78786	214348
5	1396	4767	73900	215076
6	2285	4561	72471	219717
7	1481	4565	76029	213395
8	2209	4197	69478	215939
9	2437	4965	76288	214060
10	2437	4312	76012	224129
11	2285	4504	75077	216978
Worst	2437	5071	80845	224677
Average	2039.727	4701.636	75115.27	218042.2
Best	1396	4197	69478	213395
Time (ms)	25169.82	286296.8	249601.6	741485.1

Iterasi yang terjadi pada skenario 8 adalah 4.686.252. Terjadi penurunan pada rata-rata hasil akhir apabila dibandingkan dengan skenario 7.

6.5.9 Skenario 9

Pada skenario 9 parameter *cooling rate* diubah menjadi 0.99999995 dan parameter suhu akhir menjadi 0.001. Parameter lainnya tidak dilakukan perubahan seperti yang ditunjukkan

pada Tabel 6.4. Hasil eksperimen skenario 9 ditunjukkan pada Tabel 6.13.

Tabel 6.13 Biaya Hasil Skenario 9

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	2290	4280	65224	204730
2	1396	4677	68811	205843
3	1472	4496	70276	207550
4	1472	4659	71302	208542
5	1472	4583	79024	211321
6	1396	4342	75078	203882
7	1396	4034	69671	216978
8	1396	4295	78352	212268
9	1481	4867	73457	213790
10	1396	4255	67493	223685
11	1396	4617	68280	205945
Worst	2290	4867	79024	223685
Average	1505.727	4464.091	71542.55	210412.2
Best	1396	4034	65224	203882
Time (ms)	334416.6	1732966	2355939	3324670

Eksperimen skenario 9 menunjukkan rata-rata solusi optimal dari setiap dataset memiliki hasil yang lebih baik daripada skenario 8. Jumlah iterasi yang terjadi juga semakin banyak yaitu 560.729.711.

6.5.10 Skenario 10

Pada skenario 10, parameter yang diubah adalah *cooling rate* dan panjang *tabu list*. Parameter *cooling rate* diubah menjadi 0.999995 dan panjang *tabu list* diubah menjadi 2. Parameter lainnya tidak dilakukan perubahan sebagaimana ditunjukkan pada Tabel 6.8. Hasil eksperimen skenario 10 ditunjukkan pada Tabel 6.14.

Tabel 6.14 Biaya Hasil Skenario 10

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1396	4324	77245	223413
2	2209	4714	73745	206453
3	1526	4742	74694	213360
4	1481	4490	71603	227364
5	2290	4913	75644	207155
6	2437	4925	73425	216512
7	1396	5281	74203	219974
8	1396	5118	70533	223441
9	2255	4972	74273	214654
10	1396	4740	78275	218459
11	1396	4708	70081	208652
Worst	2437	5281	78275	227364
Average	1743.455	4811.545	73974.64	216312.5
Best	1396	4324	70081	206453
Time (ms)	39008	426451	271622.1	836509.5

Hasil eksperimen pada skenario 11 menunjukkan hasil yang lebih baik daripada skenario 10. Jumlah iterasi yang terjadi pada skenario 11 adalah 46051691.

6.5.11 Skenario 11

Pada eksperimen skenario 11, panjang *tabu list* diubah menjadi 2 dan suhu akhir menjadi 0.001. Hasil eksperimen pada skenario 12 ditunjukkan pada Tabel 6.15.

Tabel 6.15 Biaya Hasil Skenario 11

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1481	3894	68222	214699
2	2285	4669	76410	210378
3	1472	4452	75301	210536

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
4	1396	3729	69469	222651
5	1396	4517	70639	213516
6	2285	4821	70115	203684
7	2290	4514	69103	211232
8	2285	4209	68795	214284
9	2290	4598	72763	213079
10	2209	4223	70014	201959
11	1396	4861	68976	222699
Worst	2290	4861	76410	222699
Average	1889.545	4407.909	70891.55	212610.6
Best	1396	3729	68222	201959
Time(ms)	127794	722284.5	983666.5	3495951

Hasil eksperimen pada skenario 11 menunjukkan hasil yang lebih baik daripada skenario 10. Jumlah iterasi yang terjadi pada skenario 11 adalah 46051691.

6.5.12 Skenario 12

Pada eksperimen skenario 12, panjang *tabu list* diubah menjadi 2 dan suhu akhir menjadi 0.001. Hasil eksperimen pada skenario 12 ditunjukkan pada Tabel 6.16 .

Tabel 6.16 Biaya Hasil Skenario 12

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
1	1481	3894	68222	214699
2	2285	4669	76410	210378
3	1472	4452	75301	210536
4	1396	3729	69469	222651
5	1396	4517	70639	213516
6	2285	4821	70115	203684
7	2290	4514	69103	211232

Percobaan	Dataset 1	Dataset 6	Dataset 11	Dataset 14
8	2285	4209	68795	214284
9	2290	4598	72763	213079
10	2209	4223	70014	201959
11	1396	4861	68976	222699
Worst	2290	4861	76410	222699
Average	1889.545	4407.909	70891.55	212610.6
Best	1396	3729	68222	201959
Time (ms)	185058.8	984400.8	1309612	4201555

Berdasarkan eksperimen 12, dapat diketahui jika skenario 12 memiliki hasil yang tidak jauh berbeda dengan skenario 11. Pada skenario 12, jumlah iterasi yang terjadi adalah 55.262.029.

6.5.13 Pemilihan Skenario Parameter

Hasil eksperimen pada skenario 1 sampai skenario 12 akan dibandingkan untuk menentukan skenario parameter yang tepat sehingga dapat menghasilkan solusi paling optimum. Selanjutnya, skenario yang terpilih akan digunakan untuk mencari solusi optimum pada semua dataset.

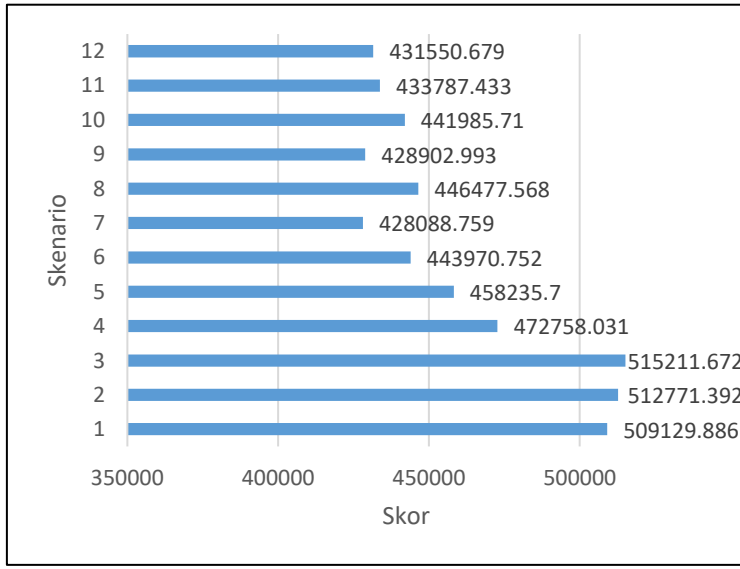
Metode yang digunakan dalam memilih skenario yang tepat adalah memberikan bobot kepada setiap dataset. Kemudian bobot tersebut dikalikan dengan rata-rata solusi setiap dataset. Selanjutnya dijumlahkan berdasarkan setiap skenario. Sehingga dapat diketahui skenario mana yang lebih optimal.

Berdasarkan peraturan yang digunakan pada tahap *development* TSC 2.0, dataset 1 dan 6 akan diberikan bobot sebesar 1. Sementara itu, dataset 11 dan 14 diberikan bobot 1.5. Tabel 6.17 merupakan hasil dari pemilihan parameter.

Tabel 6.17 Perbandingan Skor Biaya Setiap Skenario

Skenario	Dataset 1	Dataset 6	Dataset 11	Dataset 14	Skor
1	2797.636	11879.09	99886.64	229748.8	509129.9

Skenario	Dataset 1	Dataset 6	Dataset 11	Dataset 14	Skor
2	3956.182	12035.91	101468.7	229717.5	512771.4
3	3814.182	11807.64	101369	231690.9	515211.7
4	1771.364	5712.182	85212.09	224970.9	472758
5	1955.727	5624.818	80410.27	220026.5	458235.7
6	1734.091	4699.636	74079.55	217611.8	443970.8
7	1766.909	4432	71378	209881.9	428088.8
8	2039.727	4701.636	75115.27	218042.2	446477.6
9	1505.727	4464.091	71542.55	210412.9	428903
10	1743.455	4811.545	73974.64	216312.5	441985.7
11	1867.545	4560.273	73418.91	211487.5	433787.4
12	1889.545	4407.909	70891.55	212610.6	431550.7



Gambar 6.3 Grafik Perbandingan Skor Setiap Skenario

Berdasarkan Tabel 6.17 dan grafik pada Gambar 6.1, dapat diketahui bahwa skenario yang memiliki skor terbaik adalah skenario 7 dengan total skor 428088.8. Skor pada skenario 9, 11

dan 12 mendekati skor skenario 7 namun tidak dapat menghasilkan solusi lebih optimal meskipun nilai parameter telah ditingkatkan. Selanjutnya skenario 7 akan digunakan untuk mencari solusi optimum pada keseluruhan dataset.

6.6 Performa Algoritma

Setelah melakukan beberapa eksperimen skenario dengan kombinasi parameter yang berbeda-beda, maka diperlukan perbandingan untuk mengetahui performa dari algoritma *tabu-simulated annealing*. Perbandingan dilakukan dengan cara membandingkan biaya yang dibutuhkan pada rute awal dengan biaya yang dibutuhkan setelah dilakukan optimasi terhadap rute awal. Perbandingan yang kedua adalah membandingkan performa algoritma *tabu-simulated annealing* dengan algoritma lain, yaitu algoritma *great deluge* dalam menemukan rute perjalanan dengan biaya seminimal mungkin.

6.6.1 Perbandingan dengan *Initial Solution*

Pada bagian ini dilakukan perbandingan solusi hasil optimasi dengan solusi awal. Perbandingan ini berguna untuk mengetahui kinerja algoritma *tabu-simulated annealing* dalam memberikan solusi yang lebih optimum.

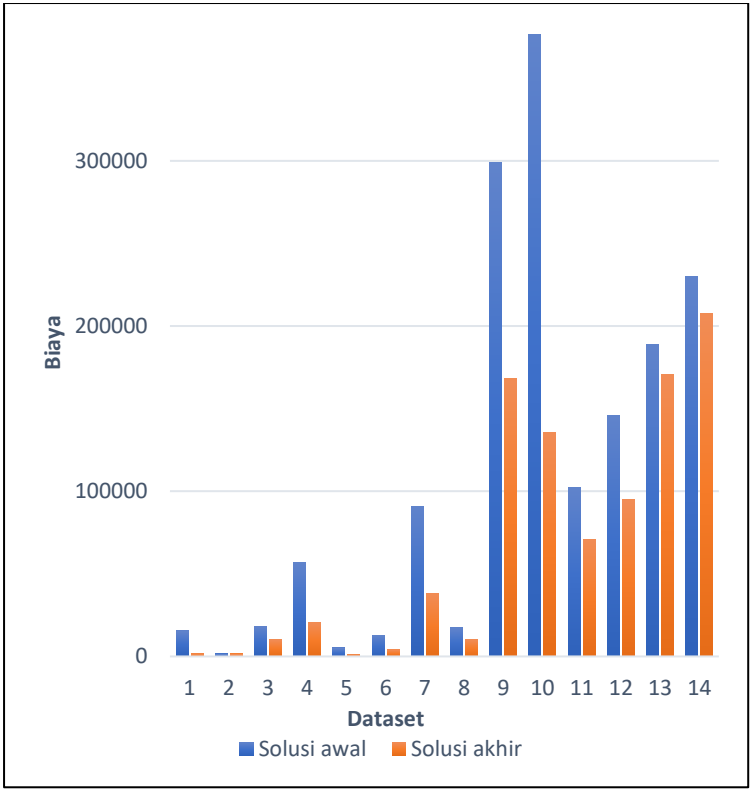
Parameter yang digunakan dalam sub bab ini adalah parameter terbaik yang didapatkan pada eksperimen skenario. Parameter tersebut adalah suhu awal sebesar 1.000.000.000, *cooling rate* 0.9999995, suhu akhir sebesar 0.1, panjang *tabu list* yaitu 3, dan 4 *low level heuristic*.

Dataset yang digunakan dalam sub bab ini adalah semua dataset yang ada, yaitu dataset 1 sampai dengan 14. Setiap dataset dijalankan 11 kali dengan menggunakan parameter yang sama. Tabel 6.18 merupakan hasil perbandingan dari solusi awal dengan solusi akhir.

Tabel 6.18 Perbandingan Biaya Solusi Awal dengan Solusi Akhir

Dataset	Solusi Awal	Solusi Akhir	Presentase
1	15443.91	1710.455	88.92%

Dataset	Solusi Awal	Solusi Akhir	Presentase
2	1498	1498	0.00%
3	18283.36	10255.09	43.91%
4	57140.45	20316.73	64.44%
5	5151.545	1304	74.69%
6	12967.91	4238.636	67.31%
7	91014.36	37873.45	58.39%
8	17528.18	10413.73	40.59%
9	299022	167996.9	43.82%
10	376471.6	135589.9	63.98%
11	102442.5	70881	30.81%
12	145962.5	94873.45	35.00%
13	188607.1	170909.2	9.38%
14	229867.1	207373.2	9.79%
Rata-rata			48.54%



Gambar 6.4 Perbandingan Biaya Solusi Awal dengan Solusi Akhir

Berdasarkan tabel 6.18 dan gambar 6.2, algoritma *tabu-simulated annealing* dapat melakukan optimasi dengan rata-rata mereduksi biaya pada solusi awal sebesar 48.54%. Pada dataset 2 tidak dimasukkan ke dalam perbandingan karena solusi awal dengan solusi akhir yang dihasilkan tidak berubah. Presentase penurunan biaya terbesar terjadi pada dataset 1 dengan presentase 88.92%. Sementara itu, persentase penurunan biaya terkecil terjadi pada dataset 13 dengan 9.38%.

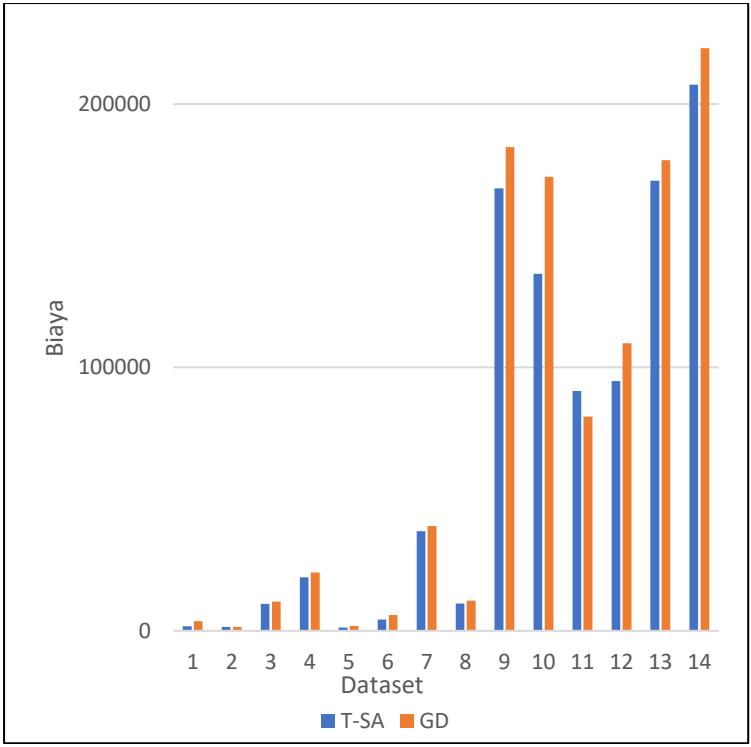
6.6.2 Perbandingan dengan Algoritma Lain

Pada bagian ini algoritma *tabu-simulated annealing* akan dibandingkan dengan algoritma optimasi lainnya yaitu

algoritma *great deluge*. Uji coba dilakukan untuk mengetahui kinerja algoritma dalam menghasilkan solusi terbaik. Percobaan dijalankan 11 kali dengan menggunakan iterasi sebanyak kurang lebih 46.051.691. Berikut ini merupakan hasil dari perbandingan algoritma *tabu-simulated annealing* dengan algoritma *great deluge*.

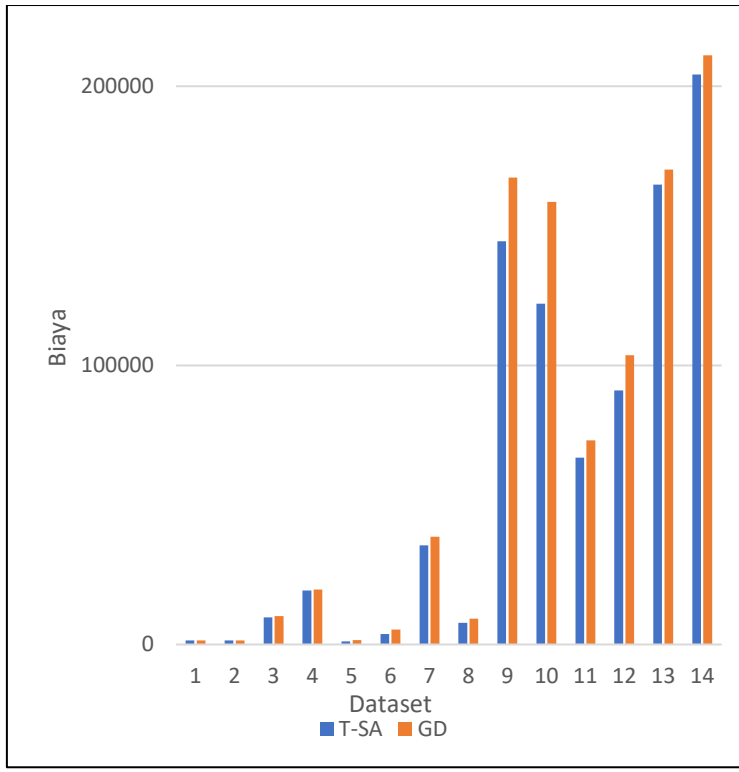
Tabel 6.19 Perbandingan Biaya dengan Algoritma *Great Deluge*

Dataset	***Tabu-Simulated Annealing***		***Great Deluge***	
	Best	**Average**	**Best**	**Average**
1	1396	1710.45	1431	3726.64
2	1498	1498	1498	1498
3	9751	10255.09	10118	11120.91
4	19299	20316.73	19683	22124.27
5	1128	1304	1525	1872.91
6	3759	4238.67	5354	5955.82
7	35565	37873.45	38588	39793.27
8	7718	10413.73	9198	11399.27
9	144419	167996.9	167291	183654.5
10	122088	135589.9	158551	172350.3
11	66939	91008	73181	81354.27
12	91008	94873.45	103641	109145.9
13	164764	170909.2	170116	178625.6
14	204185	207373.2	211069	221193.7



Gambar 6.5 Perbandingan Rata-rata Algoritma Tabu-SA dengan GD

Dari perbandingan rata-rata solusi yang ditunjukkan oleh Gambar 6.3, dapat diketahui bahwa algoritma *tabu-simulated annealing* menghasilkan rata-rata lebih optimal daripada algortima *great deluge*. Namun, perbedaan rata-rata yang dihasilkan tidak terlalu signifikan. Pada dataset 11, algoritma *great deluge* menghasilkan solusi lebih optimum daripada algoritma *tabu-simulated annealing*.



Gambar 6.6 Perbandingan Solusi Optimal Algoritma Tabu-SA dengan GD

Berdasarkan Gambar 6.4, algoritma *tabu-simulated annealing* dapat menghasilkan solusi dengan biaya lebih optimal daripada algoritma *great deluge*. Algoritma *tabu-simulated annealing* memberikan hasil yang lebih baik untuk semua dataset yang digunakan. Perbedaan paling signifikan terjadi pada dataset 10 dimana selisih biaya solusi optimal antara kedua algoritma mencapai 36.463. Pada dataset yang termasuk ke dalam ukuran *small* dan *medium*, solusi optimal yang dihasilkan algoritma *great deluge* mampu mendekati solusi optimal pada algoritma *tabu-simulated annealing*.

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Bab ini akan menjelaskan kesimpulan dari penelitian tugas akhir yang telah dilakukan, beserta saran yang diharapkan dapat membantu meningkatkan hasil pada penelitian selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Algoritma tabu-simulated annealing dapat digunakan untuk menyelesaikan permasalahan pada TSC 2.0.
2. Dari perbandingan solusi awal dan solusi akhir yang telah dilakukan, diketahui bahwa algoritma *tabu-simulated annealing* dapat meningkatkan solusi awal lebih optimal dengan rata-rata 48.54%.
3. Pada dataset 1 dan 2, algoritma *tabu-simulated annealing* dapat menemukan solusi dengan biaya paling optimal yaitu 1396 untuk dataset 1 dan 1498 untuk dataset 2.
4. Pada dataset *small* dan *medium*, performa algoritma *tabu-simulated annealing* dalam menghasilkan solusi optimal lebih baik daripada dataset berukuran *large*.
5. Pada dataset yang termasuk kelompok data *artificial* yaitu data 1 sampai dengan 10, performa algoritma *tabu-simulated annealing* lebih optimal apabila dibandingkan dengan kelompok data *real*. Solusi awal pada dataset *artificial* dapat dioptimalkan dengan presentase 40.59% - 88.92%. Sementara itu, pada dataset *real*, presentase yang dihasilkan lebih kecil yaitu 9.38% - 35%.
6. Dalam mencari solusi optimal untuk keseluruhan dataset, algoritma *tabu-simulated annealing* mampu menghasilkan rata-rata solusi optimal lebih baik daripada algoritma *great deluge*. Namun, pada dataset 11 algoritma *great deluge* mampu mengungguli

algoritma *tabu-simulated annealing* dalam menghasilkan rata-rata solusi optimal lebih baik.

7. Perubahan parameter yang dilakukan pada penelitian ini dapat mempengaruhi performa algoritma. Namun, semakin besar nilai parameter yang digunakan tidak menjamin dapat menghasilkan solusi yang lebih optimal.

7.2 Saran

Berdasarkan kesimpulan yang telah diuraikan di atas, saran yang dapat diberikan dalam penelitian berikutnya adalah:

1. Dalam penelitian ini, pengujian dilakukan dengan parameter yang sama sebanyak 11 kali pada setiap dataset. Untuk penelitian selanjutnya alangkah baiknya untuk meningkatkan jumlah pengujian agar mendapatkan hasil yang lebih akurat.
2. Dalam penelitian ini, pengujian dilakukan dengan menggunakan 12 skenario dari empat parameter input yang berbeda. Dalam penelitian ke depan, ekseperimen yang dilakukan diharapkan lebih banyak dengan kombinasi parameter yang lebih kompleks.
3. Pada penelitian ini, tidak semua kombinasi parameter dilakukan. Oleh karena itu, pada penelitian selanjutnya lebih baik semua skenario kombinasi parameter dilakukan sehingga dapat mengetahui parameter yang paling berpengaruh dalam menghasilkan solusi optimal.
4. *Low level heuristic* yang digunakan pada penelitian ini hanya sebatas *move* dan *swap*. Untuk mendapatkan hasil penelitian lebih variatif, *low level heuristic* yang digunakan dapat ditambah untuk menghasilkan solusi yang lebih optimum.

DAFTAR PUSTAKA

[1] A. A. Ismail and S. Herdjunanto, "Penerapan Algoritma Ant System dalam Menemukan Jalur Optimal pada Traveling Salesman Problem (TSP) dengan Kekangan Kondisi Jalan," vol. 1, no. 3, pp. 1–6, 2012.

[2] Y. Wang, "The Hybrid Genetic Algorithm with two Local Optimization Strategies for Traveling Salesman Problem," *Comput. Ind. Eng.*, 2014.

[3] Kara, I. and Derya, T., "Formulations for Minimizing Tour Duration of the Traveling Salesman Problem with Time Windows". Procedia Economics and Finance, vol. 26, no. 15, pp.1026-1034, 2015.

[4] A. Zhou, L. Zhu, B. Hu, S. Deng, Y. Song, and H. Qiu, "Traveling-Salesman-Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming."

[5] Y. Lin, Z. Bian, and X. Liu, *Developing a Dynamic Neighborhood Structure for an Adaptive Hybrid Simulated Annealing – Tabu Search Algorithm to Solve the Symmetrical Traveling Salesman Problem*. Elsevier B.V., 2016.

[6] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Appl. Soft Comput. J.*, vol. 11, no. 4, pp. 3680–3689, 2011.

[7] S. Suwannarongsri and D. Puangdownreong, "Adaptive Tabu Search for Traveling Salesman Problems," *Int. J. Math. Comput. Simul.*, vol. 6, no. 2, pp. 274–281, 2012.

[8] G.B. Dantzig, D.R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling salesman problem," *Oper. Res.*, vol. 2, p. 393, 1954.

[9] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, and E.

Ozcan, “A Classification of Hyper-heuristic Approaches A Classification of Hyper-heuristic Approaches,” no. August 2016, 2010.

[10] E. K. Burke et al, “Hyper-heuristics: A survey of the state of the art,” *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.

[11] S. E. E. Profile, “A hybrid simulated annealing-tabu search algorithm for the part selection and machine loading problems in flexible manufacturing systems,” no. March 2012, 2014.

[12] F. S. Hillier, *Handbook of Metaheuristics*, 2nd ed. Springer, 2010.

[13] R. W.Eglese, “Simulated annealing A tool for operational research.” European Journal of Operational Research 46, North-Holland, pp. 271–281, 1990.

[14] L. Ingber, “Simulated Annealing : Practice versus Theory,” vol. 18, no. 11, pp. 29–57, 1993.

[15] A. E. Ezugwu, A. Oluyinka, and M. Eduard, “Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem,” *Expert Syst. Appl.*, vol. 77, pp. 189–210, 2017

[16] Arıkan, Murat, and Serpil Erol. "A hybrid simulated annealing-tabu search algorithm for the part selection and machine loading problems in flexible manufacturing systems." The International Journal of Advanced Manufacturing Technology 59.5-8 (2012): 669-679.

[17] Lenin, Kanagasabai, Bhumanapally Ravindhranath Reddy, and Munagala Suryakalavathi. "Hybrid Tabu Search-Simulated Annealing Method to Solve Optimal Reactive Power Problem." International Journal of Electrical Power & Energy Systems 82 (2016): 87-91.

[18] Dueck, Gunter. "New optimization heuristics: The great deluge algorithm and the record-to-record travel." Journal

of Computational physics 104.1 (1993): 86-92.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Edwin Dwi Ahmad, lahir di Banyuwangi, 15 Oktober 1996. Biasa dipanggil Edwin, Menenempuh pendidikan dasar di SD Negeri 5 Jambewangi tahun 2003 hingga 2009. Setelah lulus dari sekolah dasar, penulis melanjutkan pendidikan formal di bangku sekolah menengah pertama di SMP Negeri 2 Genteng dan lulus pada tahun 2012. Setelah itu penulis melanjutkan pendidikan formal di SMA Negeri 1 Genteng 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 aktif mengikuti organisasi mahasiswa dan kegiatan kepanitiaan, diantaranya adalah organisasi Himpunan Mahasiswa Sistem Informasi (HMSI) serta kepanitiaan ITS Expo. Penulis merupakan mahasiswa yang menerima beasiswa bidikmisi.

Untuk mengetahui informasi lebih lanjut mengenai penelitian ini maupun terkait dengan penulis, dapat menghubungi melalui email edwindwiahmad96@gmail.com.

Halaman ini sengaja dikosongkan

LAMPIRAN A: Biaya Hasil Optimasi Rute Perjalanan

Tabel A-1 Biaya Hasil Optimasi Dataset 1

Dataset 1			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	5163	1396	20614
2	15995	1396	39334
3	19535	1526	58114
4	24915	1396	76995
5	18833	2437	95766
6	17949	2437	114814
7	12369	2437	133819
8	12170	1396	152897
9	25375	1526	172156
10	7503	1472	190232
11	10076	1396	209088
Best	5163	1396	20614
Average	15443.91	1710.455	114893.545
Worst	25375	2437	209088

Tabel A-2 Biaya Hasil Optimasi Dataset 2

Dataset 2			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	1498	1498	22763
2	1498	1498	40572
3	1498	1498	58374
4	1498	1498	76189
5	1498	1498	94082
6	1498	1498	147872

Dataset 2			
Percobaan	**Initial**	**Best**	**Time (ms)**
7	1498	1498	129868
8	1498	1498	165762
9	1498	1498	183664
10	1498	1498	201584
11	1498	1498	489570
Best	1498	1498	22763
Average	1498	1498	146390.9
Worst	1498	1498	489570

Tabel A-3 Biaya Hasil Optimasi Dataset 3

Dataset 3			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	19150	10103	23320
2	16797	10705	43117
3	21368	11468	62958
4	15496	10240	82648
5	18202	9885	102443
6	17643	9751	122212
7	19562	10134	142057
8	18786	10318	161744
9	19196	10364	181548
10	15725	9973	201310
11	19192	9865	221034
Best	15496	9751	23320
Average	18283.36	10255.09	122217.364
Worst	21368	11468	221034

Tabel A-4 Biaya Hasil Optimasi Dataset 4

Dataset 4			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	50162	20227	41725
2	58059	22740	72430
3	49784	20636	103244
4	59307	20041	134058
5	56042	19876	164784
6	53588	20353	195576
7	57400	19299	225971
8	63916	19926	225971
9	63916	19926	256088
10	61890	20356	286211
11	54481	20104	316322
Best	49784	19299	41725
Average	57140.45	20316.73	183852.7
Worst	63916	22740	316322

Tabel A-5 Biaya Hasil Optimasi Dataset 5

Dataset 5			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	5066	1382	57793
2	5254	1222	106607
3	5199	1128	155612
4	5046	1322	203766
5	5008	1136	253515
6	5149	1296	302457
7	5013	1572	351304
8	5305	1426	400580

Dataset 5			
Percobaan	**Initial**	**Best**	**Time (ms)**
9	5383	1283	448106
10	5437	1232	497515
11	4807	1345	547351
Best	4807	1128	57793
Average	5151.545	1304	302236.909
Worst	5437	1572	547351

Tabel A-6 Biaya Hasil Optimasi Dataset 6

Dataset 6			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	13000	4088	134260
2	12910	3959	128415
3	12636	4345	384889
4	12588	4600	508451
5	13401	4724	627662
6	12732	3958	742055
7	12863	4451	862251
8	13278	4056	980026
9	12726	3759	1234427
10	13515	4278	1214728
11	12998	4407	1339395
Best	12588	3759	128415
Average	12967.91	4238.636	741505.4
Worst	13515	4724	1339395

Tabel A-7 Biaya Hasil Optimasi Dataset 7

Dataset 7			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	89685	38453	236156
2	94004	36732	467361
3	85947	37394	703587
4	86361	38749	937908
5	87752	38673	1168491
6	89015	38252	1401366
7	93870	39479	1635923
8	93656	36120	1870703
9	93530	35565	2104101
10	95951	39471	2349812
11	91387	37720	2601200
Best	85947	35565	236156
Average	91014.36	37873.45	1406964.36
Worst	95951	39479	2601200

Tabel A-8 Biaya Hasil Optimasi Dataset 8

Dataset 8			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	16188	8649	3914265
2	15452	8742	5897207
3	15310	7718	9635749
4	18671	9894	3223563
5	17255	11679	4003397
6	18198	8815	4872144
7	19222	12784	5694471
8	18044	10741	6508407

Dataset 8			
Percobaan	**Initial**	**Best**	**Time (ms)**
9	18563	11778	7259294
10	17833	11830	8046979
11	18074	11921	8866761
Best	15310	7718	3223563
Average	17528.18	10413.73	6174749
Worst	19222	12784	9635749

Tabel A-9 Biaya Hasil Optimasi Dataset 9

Dataset 9			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	309513	171393	819760
2	315801	180250	1519836
3	305287	172381	2227595
4	296403	153992	2931586
5	275925	144419	3596587
6	308102	180533	4291458
7	290199	159604	4871582
8	304222	173375	5474373
9	291986	157283	6054930
10	295230	186163	6642732
11	296574	168573	7214583
Best	275925	144419	819760
Average	299022	167996.9	4149547.45
Worst	315801	186163	7214583

Tabel A-10 Biaya Hasil Optimasi Dataset 10

Dataset 10			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	378483	130049	8627183
2	375468	126200	2664279
3	378477	130150	3884388
4	378501	138192	5009319
5	378545	142146	6091659
6	374433	142132	7146611
7	374495	142150	7975243
8	375455	122088	7049001
9	374472	138037	9663905
10	378391	146190	10481329
11	374468	134155	11294157
Best	374433	122088	2664279
Average	376471.6	135589.9	7262461
Worst	378545	146190	11294157

Tabel A-11 Biaya Hasil Optimasi Dataset 11

Dataset 11			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	105050	72100	331107
2	104790	70482	632511
3	109235	69992	103668
4	103668	73587	1144786
5	101744	69042	1398259
6	100506	66939	1659162
7	101111	73962	1915526
8	96972	68624	2166913

Dataset 11			
Percobaan	**Initial**	**Best**	**Time (ms)**
9	99156	68068	2428345
10	102069	72782	2680613
11	102566	74113	2928436
Best	96972	66939	103668
Average	102442.5	70881	1580847.82
Worst	109235	74113	2928436

Tabel A-12 Biaya Hasil Optimasi Dataset 12

Dataset 12			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	152663	96385	434156
2	142810	97793	860803
3	150286	95211	1241029
4	147556	97790	1577690
5	147324	91185	1783193
6	142277	92531	2279673
7	142789	94592	2628632
8	142528	96996	2970055
9	147333	93695	3318426
10	138862	91008	3664472
11	151160	96422	4001239
Best	138862	91008	434156
Average	145962.5	94873.45	2250852
Worst	152663	97793	4001239

Tabel A-13 Biaya Hasil Optimasi Dataset 13

Dataset 13			
Percobaan	**Initial**	**Best**	**Time (ms)**
1	194748	175831	430691
2	185508	169865	839585
3	190342	174575	1273164
4	190093	174790	1681072
5	182557	166246	1670662
6	186582	173484	2463922
7	194470	168587	2869489
8	189447	167998	3266237
9	187519	164764	3792440
10	185284	165843	4052556
11	188128	178018	4436038
Best	182557	164764	430691
Average	188607.1	170909.2	2434168.73
Worst	194748	178018	4436038

Tabel A-14 Biaya Hasil Optimasi Dataset 14

Dataset 14			
Percobaan	**Initial**	**Best**	**Time (ms**
1	233804	208667	547225
2	224176	205042	1149034
3	233768	209026	1711337
4	236072	206469	2263508
5	221483	206773	2708260
6	231681	212016	4864766
7	231155	206475	3862316
8	220954	204723	4394504

Dataset 14			
Percobaan	**Initial**	**Best**	**Time (ms**
9	229010	204185	4326073
10	232631	209062	2167562
11	233804	208667	547225
Best	220954	204185	547225
Average	229867.1	207373.2	2594710
Worst	236072	212016	4864766

LAMPIRAN B: Hasil Optimasi Rute Perjalanan

Tabel B-1 Hasil Optimasi Rute Perjalanan Dataset 1

	Rute Perjalanan
Solusi awal	AB0 - AB6 - AB3 - AB7 - AB2 - AB9 - AB1 - AB4 - AB8 - AB5 - AB0
Solusi akhir	AB0 - AB7 - AB4 – AB9 - AB1 - AB6 - AB2 - AB8 - AB3 - AB5 - AB0

Tabel B-2 Hasil Optimasi Rute Perjalanan Dataset 2

	Rute Perjalanan
Solusi awal	EBJ – NBP – OMG – NCA – NUJ – OHT – GSM – EFZ – QKK – SSC - TKT
Solusi akhir	EBJ – NBP – OMG – NCA – NUJ – OHT – GSM – EFZ – QKK – SSC - TKT

Tabel B-3 Hasil Optimasi Rute Perjalanan Dataset 3

	Rute Perjalanan
Solusi awal	GDN – POZ - KJ1 – BXP – WRO – SZZ – RZE – WMI – SZY - LD3 – KTW – IEG – KRK - GDN
Solusi akhir	GDN – SZY – BXP – RZE – KRK – KTW - LD3 – WMI - KJ1 – POZ – WRO – IEG – SZZ - GDN

Tabel B-4 Hasil Optimasi Rute Perjalanan Dataset 4

	Rute Perjalanan
Solusi awal	GDN – IGL – ZAG – SXF – ODS – TSR – BLE – TLL - MHP – TRN – ZAZ – DUB – AMS – BFS – BRQ – AKT – KIV – INN – BGO – SOF - OPO – KSC – SJJ – MLA – KUO – ATH – VOZ – CPH – VNO – ANR – RIX – BUD – LJU – TGD - TIA – GVA – BEG – SKP – LUX – MRS - GDN
Solusi akhir	GDN – KSC – KIV – ODS – MHP – VNO – RIX – BLE – BGO – BFS – DUB – AMS – ANR – LUX – INN – ZAG – LJU – SJJ – TGD – MLA – OPO – ZAZ – MRS – TRN – GVA – SXF – CPH – TLL – KUO – VOZ –

Rute Perjalanan	
	AKT – IGL – ATH – TIA – SKP – SOF – BEG – TSR – BUD – BRQ - GDN

Tabel B-5 Hasil Optimasi Rute Perjalanan Dataset 5

Rute Perjalanan	
Solusi awal	RCF – FNO – CLQ – JSF – BPA – HGY – IXY – MWE – JRJ – EHO – PIG – JAQ – LGI – UCF – WYX – FWF – LZS – RZA – EQP – MLL – MUD – LQP – MGY – UUZ – GMM – GSW – KLW – QNY – PSS – FIE – AVC – JLM – LDS – GMU – QNL – EIF – SMM – MWT – AON – FED – SZR – YMJ – EGM – QUZ – SQZ – BBZ - RCF
Solusi akhir	RCF – MWE – FED – FNO – HGY – YMJ – UCF - JAQ – EIF – EHO – JRJ – BPA – AON – FWF – LZS – LQP – RZA – GSW – JSF – EGM – LGI – MLL – AVC – MUD – SMM – GMU – KLW – QNY – WYX – MGY – EQP – BBZ – IXY – UUZ – LDS – QUZ – SQZ – PIG – SZR – PSS – GMM – MWT – FIE – JLM – QNL – CLQ - RCF

Tabel B-6 Hasil Optimasi Rute Perjalanan Dataset 6

Rute Perjalanan	
Solusi awal	VHK – MUF – GCC – RNK – DUK – QEZ – UDY – EFU – CWC – NTJ – RNS – PYW – UXA – KIS – FVJ – CCR – SAN – BCT – BML – YUT – GYZ – CZA – XEH – URB – LNP – QIU – ZZY – MXV – UWL – TPR – AWR – VTQ – CTQ – IXJ – BJW – RSE – IFS – YQK – PFG – BHK – CJF – QLA – UYZ – IWR – JSP – QXW – WBF – IXQ – XSR – YHH – YCE – NOZ – CUV – BYT – AFF – LGF – WYU – IXR – WTV – BSK – ASN – WJI – ZVP – ZVF – CPQ – UNM – UGD – LDS – EUV – KKE – LFE – UXC – IEI – LTN – VDG – NKE – RLV – TXU – PJI – FPD – HPO – OTN – ZML – IKC – UDQ – UHR – LWM – DQJ – VKQ – RXB – EXJ – AJK – UAX – YMG – GCX – DBD - VHK
Solusi akhir	VHK – FVJ – XEH – IXR – NKE – PYW – JSP – RSE – TPR – NTJ – MXV – OTN – CZA – LWM – UAX – AJK – BJW – BCT – LDS – EFU – ZML – QXW – XSR

Rute Perjalanan	
	– BML – YUT – QIU – VDG – UGD – RNK – UWL – URB – DQJ – WJI – VTQ – QEZ – GYZ – CPQ – ZVP – IFS – YQK – IXQ – NOZ – CUV – WTV – ZZY – YMG – DBD – EXJ – CTQ – PJI – KKE – LFE – AFF – UXA – UHR – MUF – LGF – UYZ – IEI – SAN – YHH – LNP – LTN – WBF – IKC – UDY – CJF – WYU – BYT – CCR – QLA – FPD – HPO – PFG – TXU – KIS – AWR – CWC – BHK – BSK – IXJ – GCC – DUK – RLV – UDQ – GCX – RXB – EUV – VKQ – UXC – UNM – ZVF – ASN – IWR – RNS – YCE - VHK

Tabel B-7 Hasil Optimasi Rute Perjalanan Dataset 7

Rute Perjalanan	
Solusi awal	AHG – GWJ – DKV – DSM – ERX – DIZ – FOS – ATB – FJO – BWF – FXP – DHV – DNK – GWN – BHB – DPO – DAU – ECE – BTY – ETU – FMZ – AZS – BNK – GSL – FOE – FAY – AEN – FIO – FRW – FWO – COH – CWC – DIB – BBW – GMM – FKV – EIH – HCP – BPW – BKI – EZG – GRU – BRF – EFY – BZH – DUH – FJA – FDV – ALX – CRI – HAF – HFU - DER – DWI – DDV – HAP – DVS – BUF – BAQ – BAJ – AKZ – BXV – BZT – FFF – GAR – EPQ – FRJ – BSP – ARO – ALM – FCD – CTU – GUT – AMR – FHS – LOU – KXM – KON – JWN – HUV – MOR – IWU – MCH – LZD – LPA – LHG – HFX – LJA – MRT – MIJ – IBM – MGM – LID – MQY – HON – JHC – HWB – LUI – ITE – HGD – JMP – NIZ – KLO – LBK – JBB – JUZ – NCU – JOO – IVN – ICN – MAS – IRE – IOM – KXF – KKA – LAL – KYK – MON – MLJ – JOQ – NAD – JLI – IRN – NFL – JQD – HXU – LAA – KXN – HMD – NFB – IDG – KXI – JUG – KMH – IIH – LSE – LIR – MDX – HRV – HTJ – KPR – MYY – INT – HID – MED – KLS – HVV – LYI – JGW – MAZ - AHG
Solusi akhir	AHG – AMR – ERX – GWJ – EZG – BKI – DKV – GSL – BAJ – ECE – DNK – FDV – ALM – DSM – FFF – BRF – HAP – BWF – ATB – EPQ – DAU – GAR – FIO – DUH – AZS – CWC – BBW – BZT – FRW – GWN – HFU – FOS – BHB – BNK – DPO – FKV – DHV – DIB – FAY – DDV – ARO – FHS – FWO – DIZ – FCD –

Rute Perjalanan	
	FRJ – EFY – CRI – FMZ – GMM – EIH – FJO – DWI – DVS – DER – HAF – BPW – HCP – BAQ – FOE – ETU – BXV – FXP – BZH – BSP – GUT – ALX – AKZ – BTY – AEN – FJA – COH – CTU – GRU – BUF – IBM – MCH – KPR – HID – KXM – JMP – KLO – MED – IIH – KYK – LAL – LHG – LJA – JOO – MYY – MAS – HRV – NFL – IDG – HXU – HVV – JHC – KMH – MLJ – JWN – IVN – LPA – HGD – JQD – HUV – JBB – IOM – IRE – NFB – KON – LZD – LOU – NCU – JOQ – KXI – HTJ – LBK – MAZ – MQY – JUZ – MRT – JLI – LSE – KKA – ITE – KLS – LAA – JGW – ICN – MIJ – JUG – IRN – LUI – MDX – MON – KXF – LID – HON – MGM – LIR – INT – NAD – KXN – IWU – HMD – NIZ – HWB – MOR – LYI – HFX - AHG

Tabel B-8 Hasil Optimasi Rute Perjalanan Dataset 8

Rute Perjalanan	
Solusi awal	AEW – YSB – CJT – PRN – RLT – VRF – DWQ – DNZ – KIL – QQT – XNG – QQS – JLS – HGU – JSR – VNJ – EHZ – LAZ – EAZ – RUP – LTF – WCU – DSR – HKD – KBN – GHI – JIM – BFZ – RWM – HIN – CHK – BIN – XLY – QXU – BPY – BKB – VZK – RVM – HNP – DUA – CZJ – MAE -WHF – MYR – TRH – NFP – EBY – UGL – MRE – CKW – PDY – OVD – PFU – ULO – JHO – OTQ – ILI – KCY – BQL – QZU – KIC – CJM – IDB – FCJ – JFU – AME – PEU – JQL – QHR – HGV – JEL – IVF – IPE – PNH – RWZ – HTD – NVV – TQN – RZG – XSF – TKC – GVD – DLI – FGF – QRL – QMS – QCW – PJS – IXO – WCS – GKF – JRX – EQV – WFU – JRT – LTP – UEW – EOG – HPZ – XSY – MDB – DRO – ZCX – LKE – CIL – PRO – WJA – JZU – DEG – QUI – FMC – HIX – JBJ – RWB – MSW – IHD – NPF – NPT – OBE – FCP – MVV – LRU – LII – MYZ – OPC – KIW – RAR – QBR – NOR – USB – SVZ – CAA – BMD – EGV – CGR – ZNG – AKF – INI – BNL – EOB – EDA – LRL – PDI – DCB – MFT – FKP – EBK – BJG – RRT – FAO – VFT – EXV – OXH - RAA – CQP – MMN – JCY – BLJ – DCY – ASF – AFH – XGM – ECS – FWA – TGY – NRS – WCD – AUO –

Rute Perjalanan	
	ALA – BCU – BYB – OAE – OVC – DVQ – AZF – IID – EBC – IYZ – JKB – OLQ – DSN – PUW – FYA – NZN – CUU – PEP – PUG – GRH – HMM – AUJ – JAH – PFI – PCD – IUN – LNC – GSC – BAB – EOW – OOM – AOY - AEW
Solusi akhir	AEW – TRH – FCJ – OAE – FMC – CJT – DCB – DSN – MRE – VZK – BCU – QUI – PDY – WCU – RWZ – MVV – NOR – LAZ – EQV – HTD – DSR – JAH – KBN – GRH – INI – HMM – DCY – PUG – CGR – JBJ – OVC – HIN – XNG – HIX – UGL – AFH – QZU – PFU – HNP – JHO – AME – LRL – EDA – DUA – NFP – FCP – RLT – BPY – PRN – IID – EBY – VRF – TGY – JQL – ILI – OTQ – BNL – OOM – EBC – ECS – ALA – CZJ – WHF – XLY – YSB – PEU – EOW – DNZ – MSW – GKF – AKF – JEL – EXV – AUJ – LNC – OXH – RUP – NVV – NZN – JCY – ZNG – RWM – TQN – EGV – WJA – RAR – FGF – ZCX – FYA – RZG – LII – KIW – UEW – MDB – HPZ – ASF – XSY – VNJ – CIL – EOG – HGV – QBR – JSR – EBK – IXO – USB – HKD – PCD – AOY – KCY – BAB – RVM – NRS – DWQ – BIN – RWB – QQT – BYB – BQL – FKP – LTF – DRO – WFU – BJG – QRL – SVZ – TKC – JRT – OPC – IVF – QCW – LRU – FAO – DLI – PEP – PJS – BLJ – QMS – MAE – BKB – QXU – EOB – KIC – MFT – HGU – QQS – QHR – LKE – JRX – JZU – GHI – VFT – EAZ – RAA – GVD – CAA – JIM – MYZ – IUN – BFZ – JFU – MYR – CHK – CJM – DVQ – ULO – JKB – CKW – JLS – PDI – OVD – AZF – OLQ – XGM – IYZ – KIL – WCD – IHD – FWA – NPF – NPT – PUW – PFI – CQP - RRT- PRO – XSF – WCS – PNH – LTP – MMN – IPE – GSC – EHZ – CUU – BMD – IDB – OBE – DEG – AUO - AEW

Tabel B-9 Hasil Optimasi Rute Perjalanan Dataset 9

Rute Perjalanan	
Solusi awal	GVT – IFZ – UDV – GSQ – EGT – MZN – VMN – FNL – TKY – AXR – AHR – VSK – UFP – DPL – JLS – MRO – XIW – FTC – VKO – LSE – UEI – EJK – SAT – RZB – NTM – KKO – SPU – STR – MVE – FKS –

Rute Perjalanan	
	OBN – EYO – ZWM – LSC – SGB – OXE – VGR – ODA – QBZ – YRN – GJO – XXF – JQS – JAV – RUK – QNV – OQL – LMK – IAM – IUU – PSE – XGD – ZZO – TAE – VVL – NRL – RGA – TXQ – NVG – JSU – XIG – HSZ – NZQ – KLB – XSQ – STB – ALC – VIQ – DQF – URP – WQP – NJX – ZTN – KUL – WTB – JLO – JIV – MUL – TMZ – TMW – IUH – WWJ – JVS – YDX – REH – UMH – MRH – TSX – VGF – TQQ – VCI – GRI – WSD – DJT – VLS – QTU – UZH – IQC – JPI – JNF – UCQ – ZHS – JDF – YEK – XMH – LDA – JXV – ZID – JBA – PTC – ZJX – ZLZ – GPH – SZE – AOL – BBM – HHM – HVP – JOF – PQM – RPW – EYQ – JAK – MPH – RZK – ZEA – VLV – BYN – CPU – CAK – YKY – BRA – OBM – XXJ – DHJ – UOY – AXO – VUC – BQX – NDJ – PUN – HDY – QJV – LHE – JXA – OKB – JJD – CYU – IML – WMZ – IRH – RGB – PFD – TWU – VAK – JUJ – IJS – PWO – PYJ – AWV – QRP – ICU – XXV – TVP – YQL – HPH – NJH – LVV – VDX – DHD – QNA – JPK – KKQ – VWB – PSG – UGR – PCH – VIW – ERN – CAB – AEY – EAG – GER – FAK – QGZ – XEJ – KRI – ZUV – LBR – SAW – DLZ – EUJ – BUC – LJS – BRC – AHC – BNC – BLW – XSV – SEW – OLW – HAQ – FWU – AEH – OXU – XPE – LUA – SPH – VEH – PZZ – MGK – RBN – YSI – NRE – RDM – PSS – QBQ – NUO – WSO – BXV – RSX – UCR – UHZ – IZD – WAZ – LOC – UCN – KXR – ZNU – UZF – PAM – HJJ – SIY – UHQ - IMC – FMI – BSZ – NMF – FNQ – CEA – HAC – HVC – SOX – BCI – MLK – XOS – XNA – JGV – SWH – NGB - GVT
Solusi akhir	GVT – IFZ – SPU – NRL – TAE – SAW – SAT – JPI – CAB – KRI – VIQ – JNF – RPW – RDM – MRO – LDA – NDJ – BCI - BUC - LVV - UEI - EJK - RZB - SIY - VMN - JAK - IAM - RSX - QBZ - UHZ - EYO - MZN - ZWM - XIG - SGB - UMH - XXF - NTM - LBR - OBN - EGT - TXQ - JQS - JAV - TMW - PSG - AOL - OLW - WSO - WSD - IUH - PWO - IRH - FKS - AHR - YDX - ZZO - FMI - NVG - JSU - RUK - KLB - XSQ - HSZ - UGR - STB - TMZ - BYN - DQF - URP - XP - ZTN - MUL - ALC - PUN - NUO - JLO - KUL - WQP - NJX -

Rute Perjalanan	
	JDF - PYJ - GRI - VIW - KKO - XOS - WTB - VUC - DPL - ZEA - VCI - KXR - AWV - DJT - IUU - QTU - UFP - SWH - CEA - FNL - MVE - BSZ - WMZ - YEK - XMH - XXJ - SPH - ODA - UZF - VAK - ICU - UCN - VKO - BLW - QNV - AEH - HVP - LMK - JOF - TQQ - XEJ - VGF - QNA - XNA - LHE - JGV - VLV - TVP - CPU - VEH - BRA - OKB - OBM - SZE - EUJ - HVC - UOY - DHJ - XIW - JIV - JBA - MRH - RGB - DHD - HDY - RZK - LOC - TSX - XGD - PSE - UCR - JXA - PFD - TWU - LUA - AXO - IJS - UCQ - VLS - ZLZ - QRP - HAC - PQM - LJS - JUJ - NJH - MLK - VWB - VDX - PZZ - MPH - ZHS - OXU - LSC - GPH - LSE - KKQ - HHM - TKY - ZUV - YRN - AXR - GER - FAK - QGZ - WWJ - VGR - NRE - ERN - VVL - STR - SOX - UZH - IQC - BRC - AHC - BNC - HAQ - PCH - OQL - FWU - JPK - NZQ - BBM - SEW - PTC - XSV - JXV - CAK - YKY - QJV - MGK - GJO - JVS - UHQ - PSS - JLS - ZID - FTC - BXV - IMC - AEY - EAG - IZD - WAZ - CYU - IML - JJD - ZNU - UDV - PAM - HJJ - BQX - REH - DLZ - RGA - GSQ - NMF - FNQ - ZJX - VSK - HPH - NGB - EYQ - QBQ - YSI - RBN - OXE - XXV - YQL - GVT

Tabel B-10 Hasil Optimasi Rute Perjalanan Dataset 10

Rute Perjalanan	
Solusi awal	ECB - OGF - FGT - DSH - SVY - JAP - EVX - UZO - IFL - AAB - CUQ - JRK - DRM - NAP - RQS - QYW - EWE - ZRU - THY - URM - LRY - OQY - SRK - IPX - PWL - STD - LKC - MTV - EAK - NLU - JTU - ICJ - IYX - DRK - LNF - PGR - XXY - BHJ - IVY - KPX - MHI - OJE - SIV - XUA - YEO - XGY - SVT - KAM - FSJ - ABP - JGZ - TUJ - ZDE - OEU - APJ - IWD - LRJ - ZEV - TBG - NMV - AJI - EGP - TPR - DEN - XJE - TRO - RYW - LBS - INX - FCD - FAB - MOM - THP - CJU - GSW - XWU - ZVN - NVJ - KLG - SHI - YOO - NWT - PZG - TGN - TNQ - TNI - URW - TIP - CSH - BHR - SPH - FCV - BFP - EMG - GCP - CHH - ZZW - VRY - WMZ - CHW - AKB - SCB - AGL - DSZ - LTS - KNJ - UUG - BKB - YWI - UXZ - KFW - VBB - DLU

Rute Perjalanan	
	- ERR - AXT - QQA - FJQ - QPO - JKP - MFH - VJP - LNS - SNZ - ETP - RAX - IUM - UPC - HMD - YQZ - ISW - OLW - EVC - VON - URG - TIK - RFP - ZJA - AQG - XGF - UYA - RXJ - ZAZ - CQJ - ESC - EXW - GEH - IUK - LFN - RFS - JZK - ZUN - JEH - CNB - KJC - NPL - BSS - JVK - BOT - LTO - URO - KXM - LXL - CZK - NRS - GER - OQV - HLC - DTO - GFW - YPU - AIN - VWD - CSW - LMG - EJV - WCD - RVW - GLQ - JQC - TFG - KCX - VKI - BUF - BQK - EBZ - NPJ - XPD - AYM - ODK - VEE - VJR - HFK - ZCY - EIN - NTI - OTR - PDB - OQC - NPM - LMY - KWU - EOL - UIA - ASE - JCY - RMN - AHQ - BNO - BTQ - UZM - KWE - IJZ - UJN - KCM - EDC - RWT - WRM - XCA - GLV - QDZ - ZZZ - SPA - XFD - EPP - SLM - UKX - FZB - OXB - YEC - IIK - YRB - DDD - JGU - HZM - BOU - YMC - NHG - YJV - CCB - FJI - TKS - KXQ - BHN - FXQ - EXJ - JDU - KNS - ZIS - WNB - PHT - ACC - OLV - UYZ - QHU - AQO - HTF - WFD - VUW - UVD - YME - QLI - GFY - YHM - DEU - GFO - XOV - ICO - HDZ - WTU - QVC - KOR - XRB - YOV - DDL - WGN - ZSZ - LME - ZIU - QNP - YMJ - GJZ - VQC - LOR - STM - MRZ - ZXF - LEL - QYH - MFM - SZR - ZBE - YMX - HFW - JIE - LQV - GUO - CLG - NEU - QWG - BCB - ECB
Solusi akhir	ECB - ABP - LNF - NTI - IFL - JAP - RXJ - DLU - IJZ - JEH - VKI - AXT - UZO - OGF - ASE - LNS - CQJ - EDC - IPX - TGN - GUO - URM - BHJ - OQV - SVT - THY - UZM - TIP - EBZ - BFP - GJZ - MHI - EVC - ZUN - YQZ - VJR - SRK - WGN - OXB - BOU - VRY - NWT - VEE - KCM - YEO - XXY - FJQ - ODK - FXQ - .EXJ - ETP - AHQ - YPU - IUM - EAK - INX - WRM - SNZ - QLI - QHU - RAX - CSW - VBB - CUQ - AYM - LMY - PZG - PWL - ZZZ - LRY - IUK - YRB - YWI - LOR - WNB - FCD - JGU - NMV - JGZ - OTR - BHR - DRM - YJV - AGL - BKB - LRJ - ZEV - DSZ - XJE - VON - YOO - EGP - LTS - VQC - DDL - LME - ZZW - OQY - EMG - BCB - LBS - LEL - ZSZ - NHG - LTO - OEU - RMN - CSH - PHT - BHN - ZVN - ICJ - RQS - JKP - GLV - JCY - AQO - XGF - SVY - QPO - FGT - KOR - RFP - TBG - JTU - JRK - SCB - STD - GER -

Rute Perjalanan	
	WCD - JDU - URG - UKX - BTQ - KAM - FCV - HMD - LQV - HTF - WFD - KNJ - CZK - TRO - FJI - IIK - HFK - AJI - KJC - ERR - ZXF - ICO - TKS - DRK - EIN - CNB - UPC - QQA - QYH - NPL - KFW - PGR - KWU - VUW - KXM - JVK - UYA - QYW - DTO - PDB - HDZ - KXQ - WMZ - OJE - SHI - MFH - EJV - XOV - MOM - QVC - KPX - GSW - BUF - LKC - CCB - EXW - SPA - CHW - LXL - HZM - ISW - XRB - GFY - UYZ - RVW - ZJA - TPR - TNQ - XFD - BNO - APJ - BSS - CHH - GFW - GEH - LMG - OLW - FAB - YMC - XCA - NVJ - KLG - ZAZ - LFN - TIK - STM - HLC - MTV - AAB - UJN - MRZ - YMJ - THP - EPP - AKB - ZDE - TFG - QWG - XUA - SPH - ZCY - ZIS - DEU - OLV - GCP - DSH - NPM - UXZ - ESC - AQG - AIN - VWD - EOL - NLU - BOT - DEN - BQK - KWE - QDZ - QNP - YHM - GFO - ACC - UVD - IYX - ZRU - GLQ - FSJ - MFM - CLG - XGY - TUJ - IWD - ZBE - ZIU - NEU - VJP - RFS - EVX - NAP - JIE - JQC - CJU - NPJ - YME - URO - OQC - JZK - UIA - NRS - XPD - RYW - URW - EWE - TNI - RWT - SLM - UUG - KCX - IVY - SZR - YOV - YMX - HFW - KNS - DDD - FZB - YEC - XWU - SIV - WTU - ECB

Tabel B-11 Hasil Optimasi Rute Perjalanan Dataset 11

Rute Perjalanan	
Solusi awal	LIJ - IUM - XJU - KJP - FOG - LIR - AUD - OXZ - YYU - JBN - BPL - TLV - YKW - PXN - DTD - HYM - TDZ - LSU - MBQ - LOM - OGM - ZQV - NFQ - TNZ - RYP - EKP - IWT - HLV - OFQ - KRL - KYX - NOP - NPJ - JNL - QSA - NNZ - LSX - BUC - EIV - TTY - SLQ - OIU - PTG - SZD - HIE - KPI - DJI - CLS - UHG - KYF - OYX - LMS - OFA - LRM - TAH - QDL - LRU - TBR - MRN - GGK - UYV - AOU - VXG - NGN - KHM - QWO - BKY - MGY - NUJ - JTA - ZEX - NJQ - NTG - OPJ - CUM - HJQ - CIB - DPA - CHM - BLH - IKX - FKN - DGR - SXJ - QDG - ZWK - HCI - VAH - FWB - EGG - EOW - GAM - LSW - JWL - CJM - EHR - LQH - NRY - IZX - NQM - HMT - MWP - FKQ - UOA - JFR - BXF - EDC - FKL - NMK - IGC - MBV - JUL -

Rute Perjalanan	
	UHH - TSY - ELA - CAW - QAL - QZQ - LTQ - LUF - MFB - JZN - GAB - RMH - AEB - JCP - LVG - NTW - PHY - CPP - KIB - MYB - SRK - SVT - RIK - FIT - BEJ - FTW - FKW - BKR - LGM - NDU - DIW - KIP - CXO - PND - CQT - PKP - FTD - SPS - LIJ
Solusi akhir	LIJ - HMT - TNZ - FIT - CHM - UHG - UHH - DGR - EHR - JNL - SPS - QAL - QWO - IZX - LTQ - LVG - KRL - IUM - CLS - NFQ - GAB - DPA - GGK - KJP - CJM - EOW - LQH - MBQ - GAM - ZWK - JZN - PHY - LUF - NJQ - MYB - LIR - PTG - KPI - HJQ - LOM - YKW - NNZ - FWB - NQM - FTD - NOP - VXG - CXO - LRM - TBR - KHM - MRN - FKQ - KYF - SLQ - AUD - KYX - MBV - JFR - TAH - OYX - LSU - EDC - CQT - JUL - QDL - NUJ - BUC - MFB - CIB - NTG - QZQ - RIK - SZD - OFQ - BLH - EGG - AOU - OFA - BKR - OXZ - CAW - JTA - RMH - PXN - JBN - IGC - MGY - TLV - HCI - LMS - NTW - PKP - LRU - DJI - ZEX - PND - BKY - TDZ - BPL - NRY - JWL - KIB - BXF - DTD - MWP - NMK - NPJ - XJU - CPP - FTW - QSA - YYU - OPJ - VAH - SVT - TTY - OIU - FKL - DIW - OGM - KIP - LGM - CUM - ELA - SXJ - AEB - FKN - EIV - EKP - SRK - LSX - UYV - BEJ - IWT - HYM - ZQV - NGN - UOA - QDG - LSW - RYP - JCP - FKW - HLV - NDU - HIE - TSY - IKX - FOG - LIJ

Tabel B-12 Hasil Optimasi Rute Perjalanan Dataset 12

Rute Perjalanan	
Solusi awal	PJE - KIB - MFA - WQT - GJZ - SDL - PJV - LEX - GHP - KQX - BPT - CWO - BPA - BQP - OJU - ZKT - CZB - IWP - KLC - YBA - JCH - MIS - ZWI - GYD - WVR - OPN - JTO - XPW - IWJ - SBF - LNP - MCD - DKL - EVS - ISW - SUN - LGZ - GEF - TTB - CNF - NUU - KWC - OQD - BFK - QZH - MXC - DFY - EDL - WBJ - VCP - CKH - YMP - MNJ - EKV - MQY - ABC - LRD - GQE - GUI - BAP - DMM - NIK - FNT - NPB - RLV - HQK - ISX - DID - EZN - KSN - GDP - AEF - FWL - HBE - GYX - DUF - YNU - QIP - PIL - NZM - MYZ - HJC - WDP - TPT - LVK - JJR - XPH - UBM - ARK - FXQ - JOX - UTP - FLG - RHH - XIA - NSX -

Rute Perjalanan	
	SFS - SKJ - LNJ - QRB - ZAA - MXX - EIH - ULP - GOF - HKN - OAM - WMN - TZF - RRM - NXV - OUH - JOY - RFV - OHN - KEF - ADH - AXD - KYY - CDF - VSR - NEJ - MWL - PFI - GJY - QNT - RWG - FSR - AWL - OWP - NIU - TBX - JTJ - ISQ - DUV - ZGQ - ODY - RAG - QTT - FBT - QTI - MPH - HUU - OBN - HBC - SAN - JEQ - OXT - NVC - SWO - LKD - KGS - RCF - NXG - SKB - TAR - TLJ - HER - SXO - OYK - KFV - MUP - AVW - RDI - IGO - GSF - EFD - NTP - RNM - JPR - OMN - ILA - MTH - OZM - FDO - KPQ - QPQ - SCI - PLU - DFV - ASV - KBN - MKS - HPQ - NHH - JLG - HZT - FKK - GPQ - KNP - OLX - JOK - BCC - MYC - NSB - GRI - AVL - IHU - CNZ - LZB - PJE
Solusi akhir	PJE - XPH - EZN - IHU - VSR - LEX - GYX - ZWI - RHH - DKL - IWJ - RCF - OPN - ISX - HER - QIP - MNJ - OBN - AXD - LKD - FLG - JOK - ASV - HBC - EFD - GUI - RLV - OHN - SKJ - EKV - CWO - JOX - LVK - NUU - SXO - DFY - FNT - NXV - OMN - GOF - WBJ - OYK - FBT - ULP - NSB - EDL - TBX - HBE - BPA - GJZ - NHH - GSF - ODY - MUP - NPB - UTP - KLC - ISQ - HPQ - JLG - QZH - IGO - UBM - KFV - ZGQ - GQE - AWL - TLJ - LRD - HQK - OAM - CNF - YBA - YMP - MYC - SDL - OZM - RNM - NEJ - TTB - DUV - ADH - GRI - ZKT - GEF - EVS - RRM - JCH - HUU - OWP - DID - NIK - AEF - QRB - OLX - BFK - NVC - OQD - NIU - BCC - CKH - KGS - XPW - TPT - SKB - RWG - PJV - LNJ - ILA - KQX - SWO - GYD - BAP - KSN - JTO - RDI - FXQ - NSX - SFS - HJC - JTJ - LGZ - KBN - YNU - QTT - DMM - QNT - WDP - MQY - ZAA - NTP - SCI - MCD - KYY - GHP - AVL - MYZ - QTI - FKK - RAG - GDP - ISW - TZF - DFV - MWL - JPR - IWP - WVR - JJR - KNP - WMN - OJU - FSR - KWC - HKN - BPT - LNP - FDO - KPQ - CDF - PIL - WQT - PLU - OUH - MTH - NXG - FWL - OXT - SBF - QPQ - KIB - SUN - MFA - MXC - PFI - VCP - LZB - AVW - JOY - ABC - EIH - MIS - ARK - MXX - GPQ - XIA - MKS - CZB - MPH - CNZ - HZT - TAR - GJY - JEQ - KEF - RFV - SAN - DUF - BQP - NZM - PJE

Tabel B-13 Hasil Optimasi Rute Perjalanan Dataset 13

Rute Perjalanan	
Solusi awal	GKU – WPR – INI – SWU – PYX – FTZ – CSO – UXF – IVS – VZY – FIO – LCQ – NIF – GSI – FFX – KSY - XRE – ZPU – IUB – NCV – UOU – DAK – RYO – SOF – RYT – BEB – OGV – BQF – ZSX – NLM – TPX - MOE – NZH – YTJ – NLU – THJ – YSL – LLY – DAH – ZIM – TBP – ZFJ – FCA – POB – TGW – BYN – LWI – HIA – WOZ – GSM – WBE – AVI – RFW – CJZ – GMA – UKA - AMW – PEF - AOH – UOW – ONM - FMJ - NYB – ZTR – SDY – PNF – VKU - OIR – XMX - EUV – YXE – XYJ – BYM – AWP - NSG – ACC - TNL - SQK – EWE - VEZ - PLK - TLG – AVW - XME – AVF – RXQ – ZUO – SOL - VLA – SHH -WOS – UDG - DJY - DLP - GWD - CKZ – IZA - VOV – GRX - DIY - DOO - CYT - LZM - DCH - EPV - SRS – ZEZ – CTG – GAR - NTW – MYF – GSX - BMD – UTM – PUT - KQA - GDN - DLH - GOH – CKA – BZR - IHK – AHE - PLE – TXU – RUA – NEO - UFC – TUU - OIK - NYM – RJJ - TIP - WPW - VCP – MPZ – TDN - UAW - DFV- SAN - OUQ - AON - OEE – BJT - IKD - JCC - OCL – YPG – NUY – RZS - KPN – WDX - VZT - GXK - HGS – KKI – UHC - RVW – XJS - IOA - TGS – IZT – DEB – SAU – IZY - GQJ – AIV - JJB – PGC – UQD – GJX – REB - XNA – YWL - IUF – OOK – TVO – GKJ – YWH – SSF – IQS - NOI - SBY - ACA – AAM – VPY - LQC - UQP - CZD - LZQ – UPM – SEF – AOJ – OFL – SIQ -SNU – OEG - KFW -OFW – SHN – RZM - HML – CFT – LZZ - HTM – BDZ – OAB – LGT - NLW – QUE - LIS - SQP – DCO - BSH - ITH - XMH - KKY – QNF - OTS – VMY - WHN - AUF – EMY - WJL – AHQ - .DYS – TWD - FSI - OEH - JCJ – MGJ - GCB - DSJ - VSE – KZA - HFS - OXW – WTP – AEL – SYL - GLT - TAC - GCK - AQC - LEB - SXE - JBZ- LXH - FFA - VMW - GKU
Solusi akhir	GKU – WPR – INI – SWU – PYX – IOA – NZH - DCO – IVS – LIS – FIO – LCQ – NIF – OOK - FFX – KSY – XRE – ZPU – IUB - NCV – UOU – DAK - RYO – SOF – RYT – ZUO – OGV – BQF - ZSX - NLM - TPX – MOE – DAH – WDX – NLU -THJ - YSL – TVO – XMX – BSH – LZM – ZFJ – FCA - POB – TGW – ACC –

Rute Perjalanan	
	LWI – HIA – WOZ – GSM - WBE – AVI – YTJ – CJZ - GMA – UKA – WJL - PEF – EWE - VSE – ONM - EUV- NYB – ZTR - SDY – PNF - VKU – TAC – MPZ – SQP – YXE - XYJ – BYM - AWP – LQC – BYN - TNL – SQK - WHN – XJS - TDN – TLG – AUF – XME – AVF - RXQ – BEB - SOL – VLA – SHH – VCP -UDG - DJY – DLP - AON – CKZ – IZA - VOV – GRX - DIY – DOO - CYT – FTZ – DCH - EPV – SRS - ZEZ – CTG - GAR – NTW - MYF - GSX -BMD - UTM – PUT - KQA – GDN – DLH - GOH – CKA – BZR - IHK - YWL – PLE - TXU - RUA - NEO - UFC - TUU - OIK – NYM - RJJ - TIP -WPW- WOS – CSO - UAW – VZT - DFV - SAN – OUQ - GWD - OEE - BJT – IKD – JCC - OCL - YPG - NUY – RZS - KPN – RFW – PLK - GXK - HGS – KKI – UHC - RVW – VEZ – WTP - TGS – IZT – DEB - SAU - IZY – GQJ - AIV - JJB -AMW – UQD – KZA - REB – XNA – AHE – IUF – PGC - LLY – GKJ - YWH - AOH - IQS - NOI -SBY – ACA - AAM - FFA - NSG - UQP - CZD - LZQ – UPM – SEF – AOJ – OFL – SIQ – SNU – OEG – KFW – OFW – EMY – RZM – HML - LGT – LZZ – HTM – BDZ – OAB – CFT – NLW - QUE – VZY – FMJ – UXF – ZIM – ITH – XMH - KKY – QNF – OTS – VMY – SSF – AVW – SHN - TBP – AHQ – DYS – TWD – FSI – OEH – JCJ - MGJ -GCB – DSJ – UOW – GJX – HFS – OXW - GCK - AEL – SYL – GLT – OIR – GSI – AQC- LEB – SXE - JBZ - LXH - VPY - VMW - GKU

Tabel B-14 Hasil Optimasi Rute Perjalanan Dataset 14

Rute Perjalanan	
Solusi awal	IXG - IPU - AAJ - SYY - IZW - YLV - XJK - FDC - IBE - AOB - TPF - NNJ - JTI - HHA - EBK - YZU - OQQ - PSK - BFF - SIQ - DOB - YFC - USH - EJY - ZGD - JKA - HHR - XDP - DUB - RGA - KXF - AZZ - DZU - IZC - EXL - MFF - BRR - AOL - QSX - ZLA - WLK - PZF - CEF - KQO - OOW - EWQ - QLD - QVV - ULV - VTF - OON - MYJ - NAW - CCK - EKJ - GIT - TVT - VMP - PXB - PPL - OYM - XZU - XDS - GUO - PMQ - GQV - HYF - FES - RLF - CWL - XGR - QPR

Rute Perjalanan	
	- UPE - ZZV - DWY - JTW - JWW - WZV - EVD - JSE - SWR - MMT - HOH - LYR - EAJ - EKB - HXK - KXT - HIM - AAL - OYJ - RLV - GZZ - ZAH - LKS - MXN - DYS - WNQ - KJN - JDF - JNH - FGK - MXD - QTT - WEN - MER - AUQ - IVF - JPW - NVF - BPO - IMH - IFX - HKT - SRB - IPC - OFE - ZPD - ORA - QHL - RTP - TIL - CDN - CFH - RZZ - AZP - ILP - MWZ - TOG - ZOS - NWX - PQK - KII - GXR - FXG - ITG - OXU - HMF - XIH - BNB - LCC - OUO - RUR - ZHK - JEI - SVV - RDM - MCY - VRH - GSD - ODX - EVS - ZIT - FON - YBJ - OST - JVE - MCB - EWS - PWL - OXB - WCQ - JRO - ZBA - BOW - RUB - SGD - QEM - CKL - WEC - VZA - HFT - QID - QTS - HKP - HVE - JFR - XTR - CEA - VRY - QJO - NHX - NZZ - LWD - PNS - DBP - OGN - FGX - DEI - CNY - NRF - KUA - NXW - XMU - ZTD - FQX - SRE - ZPQ - BCJ - ZAW - SJO - KDT - YDZ - HLT - NGK - PMI - EDF - CFT - WCY - IAV - IES - FPG - JOK - HXV - DRK - DQK - YDC - GZG - VMZ - KPC - SZB - SJX - FSE - QCY - GTW - WUC - ZTM - UVX - FBV - OUL - ZYZ - JQA - QQM - THL - YAH - PGB - SDF - FMC - KBC - ONX - BAH - IER - UIE - JSN - OEX - QQE - IJJ - FSO - XJQ - KEU - LMA - TNR - OXF - YHQ - LXM - FLC - SOM - AKF - MIZ - ZUJ - PAB - WGF - EXG - END - SSN - UDG - SJH - QSS - MMR - JRF - IJS - DAX - GAL - YAF - QYJ - TKN - NNS - PTG - QNQ - MQW - FIC - WBK - GCZ - AGQ - SLF - DLU - DMI - EPH - MTA - GIU - RDO - FKP - KEV - HDB - VQA - TKP - UFE - APY - IKW - JAK - IXG
Solusi akhir	IXG - JRF - ONX - FLC - QJO - HYF - SDF - HLT - MXD - OYJ - PGB - XZU - YHQ - HHA - DUB - OXU - OGN - QCY - IFX - DZU - NRF - GQV - CEA - FKP - WEN - SRB - LXM - ZGD - GIU - KXF - XMU - QTT - BNB - UVX - HXV - NAW - PQK - JSE - NGK - RDM - FDC - ZYZ - GAL - SIQ - ZTM - FMC - RUR - NXW - IMH - OYM - QLD - QQE - DOB - ZLA - KDT - EWS - TOG - BPO - FSE - AOL - AAL - FON - DEI - WBK - YAF - HDB - BOW - RLV - RDO - ZPQ - XIH - SJX - RGA - YAH - YFC - IJS - AUQ - PMQ - MCB - BFF - PSK - JSN - YLV - JQA - GTW - WCQ - CWL - ODX

Rute Perjalanan	
	- HHR - RLF - WLK - EBK - AOB - SYY - OUL - VRY - UDG - OFE - NVF - ZIT - THL - DWY - IAV - XDP - NNS - FQX - OQQ - MWZ - CFT - HXK - VTF - AAJ - IPC - UFE - IER - QYJ - EKB - MER - FBV - SGD - XGR - GIT - PNS - RTP - MCY - PXB - JNH - BRR - LYR - JRO - WEC - JKA - CKL - ZBA - AZP - PPL - ZTD - VMP - UIE - SJH - HKP - EJY - JEI - JPW - QVV - JWW - DMI - IES - EAJ - MIZ - RUB - ILP - LMA - HFT - GSD - FSO - PZF - DAX - VRH - HVE - IVF - SRE - DRK - PAB - KQO - TKP - EKJ - OST - KPC - JDF - DQK - DYS - APY - NZZ - GCZ - FIC - MQW - WNQ - SZB - IZC - OUO - TPF - HIM - YBJ - HMF - ZZV - FGX - FXG - UPE - JTI - NWX - QID - LKS - GZZ - AZZ - WUC - KBC - TVT - AGQ - SSN - WCY - CFH - KJN - JTW - SJO - DLU - QHL - JAK - ZAW - LWD - XJK - IPU - MTA - FES - MMR - MXN - TIL - IBE - GZG - USH - GUO - TNR - OXB - OON - LCC - WGF - ITG - SWR - KUA - RZZ - CEF - HKT - ZAH - KXT - NNJ - OOW - BCJ - PTG - XJQ - NHX - QTS - IKW - VQA - QSS - EXG - QQM - BAH - ZOS - XDS - ULV - ZPD - EPH - XTR - OEX - MMT - SOM - MYJ - IJJ - IZW - ORA - ZHK - YDC - PMI - DBP - QSX - OXF - END - PWL - FGK - GXR - KEV - KEU - VMZ - SLF - ZUJ - EVS - QEM - FPG - JOK - JFR - CNY - AKF - CCK - EWQ - YZU - WZV - CDN - KII - QPR - EVD - JVE - EXL - MFF - YDZ - HOH - EDF - QNQ - VZA - SVV - TKN - IXG