TUGAS AKHIR – KS 141501

ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

ADVANCED TRAVELLER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GREAT DELUGE ITERATIVE LOCAL SEARCH (CASE STUDY: ANGKOT'S ROUTE IN SURABAYA)

DHAMAR BAGAS WISESA
NRP 5213 100 136

Dosen Pembimbing
Wiwik Anggraeni, S.Si., M.Kom
Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KS 141501

ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

DHAMAR BAGAS WISESA
NRP 5213 100 136

Dosen Pembimbing
Wiwik Anggraeni, S.Si., M.Kom.
Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – KS 141501

ADVANCED TRAVELLER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GREAT DELUGE ITERATIVE LOCAL SEARCH (CASE STUDY: ANGKOT'S ROUTE IN SURABAYA)

DHAMAR BAGAS WISESA
NRP 5213 100 136

Supervisors
Wiwik Anggraeni, S.Si., M.Kom.
Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

LEMBAR PENGESAHAN

ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

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

Oleh:

Dhamar Bagas Wisesa
5213 100 136

Surabaya, Juli 2017

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP 19650310 199102 1 001

LEMBAR PERSETUJUAN

ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

TUGAS AKHIR

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

Oleh :

Dhamar Bagas Wisesa
5213 100 136

Disetujui Tim Penguji : Tanggal Ujian : 06 Juli 2017
Periode Wisuda : September 2017

Wiwik Anggraeni, S.Si., M.Kom.	**(Pembimbing 1)**
Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.	**(Pembimbing 2)**
Edwin Riksakomara, S.Kom., M.T.	**(Penguji 1)**
Radityo Prasetianto W., S.Kom., M.Kom.	**(Penguji 2)**

ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

Nama Mahasiswa : Dhamar Bagas Wisesa
NRP : 5213 100 136
Departemen : SISTEM INFORMASI FTIF-ITS
Dosen Pembimbing 1 : Wiwik Anggraeni, S.Si., M.Kom.
Dosen Pembimbing 2 : Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.

ABSTRAK

Kemacetan merupakan salah satu permasalahan terbesar untuk kota – kota besar di dunia, ini disebabkan oleh banyak hal mulai dari urbanisasi, peningkatan populasi dan permasalahan jumlah kendaraan pribadi yang lebih banyak digunakan dibandingkan dengan kendaraan umum yang disediakan

Metode yang digunakan untuk memodelkan permasalahan tersebut adalah Orientering Problem dengan pengambilan jarak dan waktu diambil menggunakan Google Maps dan penentuan skor dengan demand atau jumlah angkutan kota pada trayek tersebut, lalu formulasi permaslahaan akan dibuat sesuai dengan langkah – langkah metode tersebut. Orienteering problem diigunakan dikarenakan penelitian sebelumnya belum ada yang menggunakan terhadap studi kasus saat ini tetapi sudah teruji baik dalam permasalahan sejenis.

Pencarian solusi dari model yang telah dibuat akan dilakukan dengan menggunakan Great Deluge Iterative Local Search untuk mencari solusi terbaik dari model dan bagaimana pencarian skor terbesar dapat dilakukan dengan menggunakan

iterative local search yang disampaikan. Iteratice local search ini menyediakan kecepatan dan keefisienan dalam melakukan pencarian solusi.

Dalam penelitian ini, didapati bahwa orienteering problem dapat memodelkan enam buah trayek angkot menjadi network model yang saling terhubung antara satu sama lain dan mendapatkn rute terpendeknya.

Algoritma Great Deluge Iterative Local Search juga dapat meningkatkan hasil dari pencarian solusi awal yang layak dengan menggunakan cara random dengan waktu tempuh 81 menit dan skor 4010.

Kata Kunci: ***Kemacetan, Orienteering Problem, Iterative Local Search, Great Deluge, Great Deluge Iterative Local Search, Optimasi, Pemodelan***

ADVANCED TRAVELLER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GREAT DELUGE ITERATIVE LOCAL SEARCH (CASE STUDY: ANGKOT'S ROUTE IN SURABAYA)

Name	**: Dhamar Bagas Wisesa**
NRP	**: 5213 100 136**
Department	**: INFORMATION SYSTEM FTIF-ITS**
Supervisor 1	**: Wiwik Anggraeni, S.Si., M.Kom.**
Supervisor 2	**: Ahmad Mukhlason, S.Kom., M.Sc., Ph.D.**

ABSTRACT

Congestion is one of the biggest problems for big cities in the world, this is caused by many things ranging from urbanization, population increase and the problem of the number of private vehicles that are more widely used than public transport provided

The method used to model the problem is Orientering Problem with distance taking and time taken using Google Maps and determining the score with the demand or the number of city transport on the route, then the formulation of the permaslahaan will be made in accordance with the steps of the method. Orienteering problem is used because previous research has not been applied to the current case study but it has been well tested in similar problems.

The search for a solution of the model that has been created will be done using Great Deluge Iterative Local Search to find the best solution of the model and how the largest scoring search can be done using iterative local search submitted. Iteratice local search provides the speed and efficiency in searching for solutions.

In this study, it was found that the orienteering problem can model six angkot routes into network models that are interconnected with each other and find the shortest route.

The Great Deluge Iterative Local Search algorithm can also improve results from searching for a feasible initial solution using a random manner with an 81 minute travel time and a 4010 score.

Keywords: ***Traffic, Orienteering Problem, Iterative Local Search, Great Deluge, Great Deluge Iterative Local Search Optimization, Modelling***

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Allah SWT atas segala berkat dan rahmat-Nya lah penulis dapat menyelesaikan buku tugas akhir dengan judul "***ADVANCED TRAVELLER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GREAT DELUGE ITERATIVE LOCAL SEARCH (STUDI KASUS: TRAYEK ANGKOT SURABAYA)*"** sebagai salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Ibu Ernawati dan Bapak Ali Maksum Anwar selaku kedua orang tua, Ghina Aulia Megaputri selaku kakak , serta segenap keluarga penulis yang selalu memberikan doa, dukungan, dan motivasi selama penulis menempuh studi hingga menyeleaikan penelitian Tugas Akhir ini.
2. Ibu Wiwik Anggraeni, S.Si., M.Kom dan Pak Ahmad Mukhlason, S.Kom, M.Sc., Ph.D. selaku dosen pembimbing yang telah mengarahkan, membimbing, memberikan nasihat, dan dukungan kepada penulis dalam mengerjakan tugas akhir ini hingga selesai.
3. Bapak Edwin Riksakomara, S.Kom., M.T. dan Bapak Radityo Prasetianto W., S.Kom., M.Kom. selaku dosen penguji penulis yang selalu memberikan masukan yang meningkatkan kualitas dari Tugas Akhir ini.
4. Bapak Rully Agus Hendrawan, S.Kom, M.Eng. selaku dosen wali penulis yang selalu memberikan motivasi, dan saran selama penulis menempuh pendidikan S1.
5. Tim Optimasi Trayek Angkot Surabaya I Wayan Angga Kusuma Yoga, dan Jockey Satria Wijaya yang selalu Bersama dalam suka, duka dan saling membantu dalam pengerjaan tugas akhir ini.

6. Arifianita Febrina Putri yang selalu menemani, memotivasi, dan saling membantu dalam keadaan suka, duka dalam perkuliahan di Institut Teknologi Sepuluh Nopember ini.
7. Octgi Ristya Perdana, Mochammad Rizki Wicaksono, Febrian Anggoro Harimurti, Valliant Verliando, Muhammad Alam Pasisullah, dan Caesar Gilang sebagai teman dekat segame, yang selalu memotivasi, menjadi inspirasi dan sumber canda tawa selama pengerjaan tugas akhir dan perkuliahan.
8. Teman – teman senior dan junior dari Solaris, Beltranis, Osiris yang telah membantu dalam masa perkuliahan di Sistem Informasi ITS.
9. Seluruh dosen pengajar, staff, dan karyawan di Departemen Sistem Informasi, FTIF ITS Surabaya yang telah memberikan ilmu dan bantuan kepada penulis selama ini.
10. Serta semua pihak yang telah membantu dalam pengerjaan Tugas Akhir ini yang belum mampu penulis sebutkan diatas.

Terima kasih atas segala bantuan, dukungan, serta doanya. Semoga Allah SWT senantiasa memberkati dan membalas kebaikan-kebaikan yang telah diberikan kepada penulis.

Penulis pun menyadari bahwa Tugas Akhir ini masih belum sempurna dengan segala kekurangan di dalamnya. Oleh karena itu penulis memohon maaf atas segala kekurangan yang ada di dalam Tugas Akhir ini dan bersedia menerima kritik dan saran. Semoga Tugas Akhir ini dapat bermanfaat bagi seluruh pembaca.

Surabaya, Juli 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bab pendahuluan akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat kegiatan tugas akhir dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir dapat dipahami.

1.1 Latar Belakang Masalah

Kemacetan merupakan masalah yang sudah umum di dunia, menurut CNN, ibukota Indonesia, DKI Jakarta menempati posisi ke empat di dunia dalam tingkat kemacetan terburuk pada jam sibuk di siang hari pada tahun 2017 [1]. Masalah kemacetan merupakan massalah yang paling sering ditemukan di kota – kota besar di Indonesia. Di Indonesia Sendiri, kemacetan biasanya menjadi masalah utama dalam kota yang memiliki penduduk lebih dari dua juta jiwa seperti Jakarta, Bandung, Surabaya, Medan dan Yogyakarta [2]. Kemacetan memiliki beberapa penyebab utama yaitu urbanisasi, terbatasnya jaringan jalan, dan kecenderungan pemakai kendaraan menggunakan kendaraan pribadi dibandingkan dengan transportasi umum meningkat .

Pada tahun 2015 silam, kemacetan yang terjadi di ibukota DKI Jakarta sudah sangat buruk sampai pada saat itu mencapai pada posisi pertama di dunia sebagai kota termacet di dunia versi Castrol Magnatec [3]. Sedangkan untuk kemacetan di Surabaya sudah tergolong buruk sampai pada saat itu mencapai pada posisi ke empat dalam berita tersebut. Ini diperkirakan disebabkan oleh setiap tahunnya penambahan moto baru dapat mencapai sekitar 15% dan untuk mobil mencapai 6%. Didapati

sekitar 14.000 motor baru setiap bulan yang beredar di jalan sedangkan untuk mobil bisa mencapai 4.000 unit per bulannya [4]. Ranking kemacetan yang 3.128 kasus, dan Sidoarjo sebanyak 2.292 kasus [5]. digunakan oleh Castrol Magnatec ini dihitung dari jumlah *stop – start* yang berarti jumlah jalan dan berhenti pada suatu kota, yakni pada Jakarta sebanyak 33.240 kali dan Surabaya dengan 29.880 kali. Untuk mengatasi kemacetan ini, Surabaya sendiri sudah memiliki beberapa solusi seperti menambahkan jalur, menertibkan lalu lintas, memberikan jalur alternative. Penelitian tugas akhir ini akan memodelkan *ATIS (Automated Transportation Information System)* yang nantinya dapat digunakan sebagai solusi dalam mengatasi kemacetan di Surabaya ini.

Pada Penelitian tugas akhir ini, pemodelan *ATIS* akan menggunakan metode *Orienteering Problem* dimana metode tersebut digunakan karena berdasarkan penelitian sebelumnya metode ini dapat memodelkan permasalahan tentang kemacetan secara baik dan akan diteskan menggunakan *iterative local search* yang akan dipakai.

Sedangkan pemakaian Algrotima *Great Deluge Iterative Local Search* digunakan karena setelah membaca beberapa penelitian sebelumnya, didapati bahwa *Great Deluge Iterative Local Search* ini terbukti lebih cepat dan lebih efisien dalam pencarian hasil yang diinginkan, dan metode ini sudah diaplikasikan ke beberapa permasalahan *Combinatorial Optimization* seperti *Job-Scheduling Problem* [6] [7], *Flow-shop Problem* [8], *Vehicle Routing Problem* [9], dan lain – lain.

Namun, dikarenakan kurangnya referensi *Orienteering Problem* dengan menggunakan *Great Deluge Iterative local search*, maka ini menjadi peluang untuk penelitian baru dan diharapkan dapat menghasilkan hasil yang memuaskan. Meskipun demikian, *Great Deluge Iterative Local Search* terbukti dapat memberikan solusi yang cepat dan efisien pada permasalahan yang lain.

1.2 Rumusan Permasalahan

Berdasarkan uraian latar belakang yang telah dijelaskan, maka rumusan permasalahan yang dapat diselesaikan dalam tugas akhir ini adalah:

1. Model Seperti apa yang dapat dipakai untuk mencari jalur yang optimum?
2. Bagaimana *Orienteering Problem* dapat membuat model penentuan rute perjalanan yang paling optimum?
3. Bagaimana cara *Great Deluge Iterative Local Search* dapat memberikan solusi dari model yang telah dibuat?

1.3 Batasan Permasalahan

Batasan pemasalahan dalam tugas akhir yang dilakukan untuk melakukan Optimasi Trayek Angkot di Surabaya adalah:

1. Studi kasus yang digunakan hanya meliputi sebagian kecil dari kota Surabaya, yaitu trayek mikrolet dengan kode M, S, U, UBB, TWM, WB.
2. Jarak dan waktu yang digunakan masing – masing trayek diperoleh menggunakan Google Maps.
3. Dalam Pemodelan penelitian tugas akhir ini tidak melibatkan faktor psikologis dan finansial.
4. Hasil dari penelitian akhir ini ditujukan kepada masyarakat dan pengunjung kota Surabaya yang dikhususkan menggunakan transportasi angkutan umum.

1.4 Tujuan Penelitian

Tujuan dilakukannya tugas akhir ini sebagai berikut:

1. Membuat model matematis untuk trayek mikrolet di Surabaya menggunakan metode *Orienteering Problem.*
2. Menemukan solusi dari model yang telah dibuat menggunakan *Great Deluge Iterative Local Search*.

3. Menghasilkan dataset yang siap digunakan untuk penelitian berikutnya.

1.5 Manfaat Penelitian

Manfaat yang dapat diambil dari penelitian tugas akhir ini adalah:

Untuk Penulis:

1. Menambah wawasan tentang metode *Orienteering Problem* dan *Great Deluge Iterative Local Search* pada bidang transportasi umum.

Untuk Instansi Terkait:

1. Memberi masukan model untuk *Intelligent Transport Systems (ITS)* di kota Surabaya khususnya pada bidang *Advanced Traveler Information Systems* (ATIS).
2. Hasil penelitian dan implementasi system ini bisa dijadikan contoh dan diterapkan pada kota – kota lain di Indonesia.

Untuk Pengguna:

1. Dapat memberikan rekomendasi rute beserta mikrolet mana yang digunakan dari tempat awal dan ke tempat tujuan .
2. Membuat model dasar yang dapat dikembangkan lebih lanjut untuk pengembangan Surabaya Intelligent Transport System.
3. Di bidang kita dapat menyelesaikan masalah routing kendaraan.

1.6 Relevansi

Transportasi umum merupakan salah satu solusi terbaik yang dapat dilakukan oleh pemerintah setempat terkait dengan permasalahan kecametan, namun minat masyarakat untuk menggunakan transportasi umum masih rendah. Untuk meningkatkan minat masyarakat menggunakan transportasi umum, dibuatlah Surabaya *Intelligent Transport System* (Surabaya ITS) yang memiliki *Advanced Traveller Information System* (ATIS) sebagai salah satu bagiannya. Dengan menggunakan model dari *Orienteering Problem* dan *Great Deluge Iterative Local Search* untuk memberikan rekomendasi rute untuk pengguna yang ingin menggunakan transportasi umum, khususnya mikrolet untuk solusi permasalahan ATIS. Tugas akhir ini diharapkan dapat menjadi masukan untuk implementasi Surabaya ITS dan menjadi referensi untuk penelitian yang berhubungan dengan optimasi, dan optimasi terkait transportasi, khususnya untuk transportasi mikrolet.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan mengenai penelitian terkait dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini. Landasan teori akan memberikan gambaran secara umum dari landasan penjabaran tugas akhir ini.

2.1 Studi Sebelumnya

Berikut adalah daftar penelitian yang telah dilakukan sebelumnya yang mendasari penelitian tugas akhir ini.

Tabel 2.1 Penelitian Terdahulu 1

Judul Penelitian	Iterated local search algorithm for solving the orienteering problem with soft time windows
Penulis dan Tahun	Brahim Aghezzaf, Hassan El Fahim, 2016
Deskripsi Umum Penelitian	Penelitian ini berisi tentang memberikan solusi dari orienteering problem dengan soft time windows, dimana waktu perjalanan tidak bisa secara akurat didapatkan dan juga dapat memberikan algoritma hybrid dari aloritma yang sudah ada dikembangkan dan membandingkan dengan beberaa algoritma yang ada untuk menyelesaikan permasalahannya.
Keterkaitan dan kesimpulan	Metode yang digunakan hamper sama yaitu Orienteering problem, meskipun pada paper menggunakan soft time windows, dan pada penelitian ini, Great Deluge belum digunakan sehingga dapat menjadi peluang baru dalam penelitian.

Tabel 2.2 Penelitian Terdahulu 2

Judul Penelitian	Orienteering Problem: A Survey of Recent Variants, Solution Approaches and Applications
Penulis dan Tahun	Aldy Gunawan, Hoong Chuin Lau, Pieter Vansteenwegen, 2015
Deskripsi Umum Penelitian	Penelitian ini menjelaskan secara detail tentang Orienteering problem, mulai dari penjelasan permasalahan, formulasi permasalahan, jenis – jenis *Orienteering Problem*, pengaplikasian permasalahan terhadap dunia nyata serta benchmark dan algoritma yang digunakan dalam penyelesaian permasalahan *Orienteering Problem*
Keterkaitan dan kesimpulan	Metode *Orienteering Problem* pada penelitian ini sama dengan yang akan digunakan dengan penelitian tugas akhir ini, serta penelitian ini belum menggunakan *Great deluge Iterative Local Search* sehingga menjadi peluang utnuk menjadi penelitian baru.

Tabel 2.3 Penilitan Terdahulu 3

Judul Penelitian	An iterated local search algorithm for solving the Orienteering Problem with Time Windows
Penulis dan Tahun	Aldy Gunawan, Hoong Chuin LAU, and Kun Lu, 2015
Deskripsi Umum Penelitian	Penelitian ini menggunakan salah satu jenis dari orienteering problem, yaitu orienteering problem with time windows. Pada penelitian ini pemodelan dilakukan menggunakan metode tersebut dan dilakukan pencarian solusi dengan iterated local search, dan setelah dilakukan pencarian hasil solusi ditingkatkan dengan

	menggunakan AcceptanceCriterion dan Perturbation.
Keterkaitan dan kesimpulan	Metode yang diapaki merupakan variansi dari *orienteering problem*, yang digunakan dalam penelitian ini dan menggunakan local search, namun *great deluge* belum digunakan dalam local search tersebut.

Tabel 2.4 Penelitian Terdahulu 4

Judul Penelitian	New Formulations for the Orienteering Problem
Penulis dan Tahun	Imdat Kara, Papatya Sevgin Bicakc, Tusan Derya, 2016
Deskripsi Umum Penelitian	Penelitian ini mengusulkan formulasi baru terhadap *Orienteering Problem* dengan menambahkan konstrain baru untuk mempercepat komputasi permasalahannya.
Keterkaitan dan kesimpulan	Metode yang dipakai sama dengan penelitian tugas akhir ini, meskipun hasil yang diberikan tidak terlalu relevan Karena mempercepat komputasi hanya akan berguna apabila node yang digunakan sangatlah banyak.

2.2 Dasar Teori

Pada sub bab ini berisi teori-teori yang mendukung serta berkaitan dengan tugas akhir yang dikerjakan.

2.2.1 Optimasi

Dalam ilmu matematika, *computer science*, dan *operation research*, optimasi matematis merupakan pemilihan dari elemen terbaik (dalam beberapa kriteria) dari beberapa set alternative yang ada [10]. Dalam kasus termudahnya, masalah optimasi terdiri dari memaksimalkan atau meminimalkan sebuah fungsi tujuan dengan memiih inputan secara sistematis

dalam sebuah set yang diperbolehkan dalam fungsi tersebut. Dewasa ini, optimasi sudah sangat mudah dilakukan dengan menggunakan alat bantu seperti computer, sehingga dapat mempercepat proses perhitungan dengan waktu yang singkat. Pada kehidupan nyata, optimasi dapat di aplikasikan terhadap mesin, ekonomi dan finansial, pelistirkan, riset operasi atau manajemen sains, perkontrolan, geofisika dan pemodelan molekul.

2.2.1.1. *Orienteering Problem*

Orienteering merupakan sebuah olahraga *outdoor* yang biasanya dimainkan di tempat yang banyak pepohonannya, di dalam pepohonan tersebut terdapat beberapa "*control point*" yang memiliki skor [11]. Sedangkan nama *Orienteering Problem* sendiri berasal dari olahraga *orienteering* tersebut [12]. Di dalam game ini, seorang individu peserta mulai dari sebuah *control point*, dengan tujuan mengunjungi sebanyak mungkun *checkpoint* dan kembali ke *control point* masing – masing dalam sejumlah waktu yang diberikan [13].

Ada beberapa jenis *Orienteering Problem* (selanjutnya akan disebut sebagai OP), antara lain *Orienteering Problem* biasa, *Team orienteering problem, Orienteering Problem with Time Windows, Team Orienteerign Problem with Time Windows, Time Dependent Orienteering Problem*, ada pula permasalahan yang diekstensikan dari OP yaitu *Stochastic Orienteering Problem, Generalized Orienteering Problem, Arc Orienteering Problem, Capacitated Team Orienteering Problem, Orienteering with Variable Profits, Clustered Orienteering Problem, Cooperative Orienteering Problem, Multi-agent Orienteering Problem,* dan *Orienteering problem with multiple aspect* [14]. Yang akan dibahas dalam penelitian tugas akhir ini hanya *Orienteering problem*.

2.2.1.1.1. Tahapan Orienteering Problem

Dalam OP, sebuah set yang terdiri dari beberapa node N telah diberikan, masing – masing memiliki Skor S_i , dimana node awal dan node akhirnya telah ditentukan. Waktu yang dibutuhkan (t_{ij}) untuk menempuh dari node i ke j telah diketahui untuk tiap node. Tidak semua node dapat dikunjungi Karena keterbatasan waktu(T_{max}). Skornya diasumsikan ditambahkan secara keseluruhan dan tiap node hanya bisa dikunjungi sekali [13].

OP dapat dibantu dengan menggunakan grafik G=(V,A) dimana $V = \{v_1,\ldots,v_n\}$ adalah se set node nya dan A adalah semua node yang berhubungan. Dalam konteks ini, skor S_i tidak boleh negative dan berhubungan dengan setiap node $v_i \in V$ dan waktu tempuh t_{ij} berhubungan dengan tiap node yang berhubungan $a_{ij} \in A$. OP terdiri dari menentukan *Hamiltonian Path* G' ($\subset$G) dari subset V. termasuk dari node awal (v_1) dan akhir (v_n) dan memiliki waktu tempuh dibawah $T_{max,}$ dengan tujuan untuk memaksimalkan skor yang telah dikumpulkan. Oleh Karena itu, OP kebih popular atau dapat dikatakan sebagai sebuah cara untuk menari jalan tercepat antara dua node daripada sebuah tur atau sirkuit [13].

Menggunakan notasi yang telah dijelaskan tadi, OP dapat diformulasi menjadi sebuah *Integer Problem*. Variable keputusan berikut akan digunakan: x_{ij} = 1 jika mengunjungi node i diikuti dengan mengunjungi kunjungan ke node j = 0 jika tidak u_i = posisi dari node i pada jalur nya.

$$MAX \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i X_{ij}, \qquad (0)$$

$$\sum_{j=2}^{N} X_{ij} = \sum_{i=2}^{N-1} X_{iN} = 1, \qquad (1)$$

$$\sum_{i=2}^{N-1} X_{ik} \sum_{j=2}^{N} X_{kj} \leq 1; \quad \forall k = 2, \dots, N-1 \qquad (2)$$

$$\sum_{i=2}^{N-1} \sum_{j=2}^{N} t_{ij} X_{ij} \leq T_{max}, \qquad (3)$$

$$2 \leq u_i \leq N; \quad \forall i = 2, \dots, N, \qquad (4)$$

$$u_i - u_j + 1 \leq (N-1)(N - X_{ij}; \quad \forall i, j = 2, \dots, N, \qquad (5)$$

$$X_{ij} \in \{0,1\}; \quad \forall i = 1, \dots, N, \qquad (6)$$

Fungsi tujuan (0) adalah untuk memaksimalkan skor yang telah dikumpulkan. Konstrain (1) memastikan node awal dimulai dari node 1 dan berakhir di node N. Konstain (2) memastikan semua node terhubung dan maksimal dikunjungi sebanyak satu kali. Konstrain (3) memastikan alokasi waktu yang terbatas. Konstrain (4) dan (5) diperlukan untuk mencegah *subtour*. Konstrain pencegahan *subtour* diformulasikan berdasarkan formulasi Miller-Tucker -Zemlin (MTZ) untuk TSP [15].

2.2.1.2. Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma pencarian rute terpendek antara *node* dalam *graph* yang ditemukan oleh Edsger W. Dijkstra pada tahun 1956 yang diterbitkan pada sebuah buku pada tahun 1959 [16]. Algoritma ini tersedia dalam beberapa jenis seperti jenis asli permasalahan Dijkstra untuk mencari rute terpendek antara dua *node* [16] , tetapi yang lebih sering ditemukan adalah jenis yang memastikan sebuah *node* sebagai *node*sumber atau awal dan mencari rute terpendek ke semua *node* lain dalam *graph* sehingga menghasilkan *shortest path tree* [17]. Dalam beberapa bidang, algoritma Dijkstra dikenal sebagai *uniform-cost search* dan diformulasikan

sebagai sebuah bagian dari ide yang lebih general dalam *best-first-search* [18].

Kode 2.1 merupakan pseudocode algoritma Dijkstra [19].

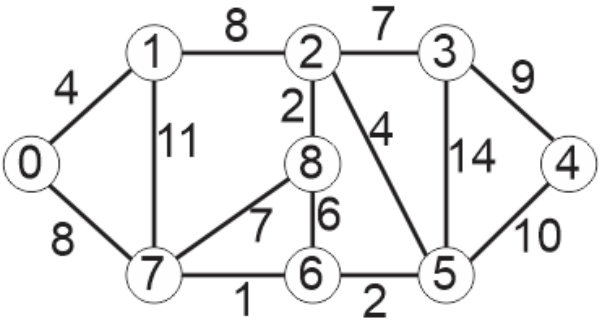
```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:
6          dist[v] ← INFINITY
7          prev[v] ← UNDEFINED
8          add v to Q
9
10      dist[source] ← 0
11
12      while Q is not empty:
13          u ← vertex in Q with min dist[u]
14          remove u from Q
15
16          for each neighbor v of u:
17              alt ← dist[u] + length(u, v)
18              if alt < dist[v]:
19                  dist[v] ← alt
20                  prev[v] ← u
21
22      return dist[], prev[]
```

Kode 2.1 Pseudocode Dijkstra

Algoritma Dijkstra dimulai dengan memberikan sebuah nilai jarak yang tentative ke semua *node*, berikan nilai 0 untuk *node* awal dan tak terhingga untuk *node* lainnya. Kemudian tandai *node* awal sebagai *node* saat ini dan tandai *node* lainnya sebagai *node* yang belum dikunjungi. Buat sebuah *set* untuk semua *node*yang belum dikunjungi dengan nama *unvisited set*. Setelah itu, dari *node* saat ini, hitung jarak sementara ke *node* bersebelahan yang berhubungan dan bandingkan hasil perhitungan jarak sementara tadi tersebut dengan nilai yang

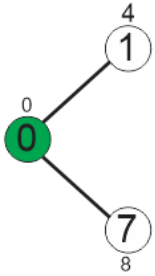
telah dimiliki oleh *node* tersebut dan pilih yang lebih kecil, jika nilai nya lebih besar, maka simpan nilai yang sudah ada. Ketika sudah mempertimbangkan dan melihat *node* yang bersebelahan dan berhubungan dari *node* awal, maka tandai *node* awal tersebut sebagai *visited* dan keluarkan dari *unvisited set* tadi. *Node* yang ditandai dengan *visited* tidak akan dilakukan pengecekan ulang. Jika *node* tujuan sudah ditandai dengan *visted* atau tidak terhubung sama sekali dengan *node* saat ini, maka algoritma berhenti. Kalau tidak, pilih *node* yang belum ditandai dengan *visited* dengan nilai jarak sementara terkecil, jadikan itu sebagai *node* awal (*node* saat ini) dan kembali lakukan perhitungan jarak sementara.

Jika dilihat dari kode 2.1, baris ke lima adalah untuk inisialisasi, baris ke enam adalah deklarasi jarak yang tidak diketahui dari *node* awal ke v, baris ke 7 adalah untuk deklarasi bahwa semua *node* pada awalnya dimasukkan kedalam *unvisited node* kecuali *node* awal. Baris ke 10 untuk menghitung jarak dari *node* awal ke *node* tujuan, baris ke 13 untuk memilih *node* dengan jarak terkecil atau terdekat akan dipilih terlebih dahulu untuk dijadikan *node* awal. Baris ke 16 adalah *looping* untuk menghitung jarak terpendek apabila *node* awal tersebut masih termasuk dalam *unvisited set* dan baris ke 18 adalah untuk mencari rute terpendeknya. Untuk lebih mudahnya dapat dilakukan contoh seperti pada gambar 2.1



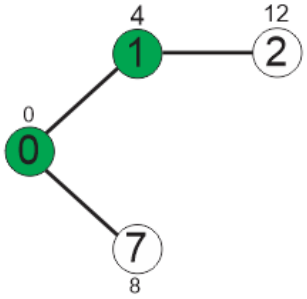
Gambar 2.1 Contoh Permasalah Dijkstra 1

Pada gambar 2.1, *node* awalnya adalah 0 dan tujuannya adalah mencari rute terpendek ke semua *node*yang ada, maka hitung dari *node* awal ke *node* bersebelahan yang berhubungan dari *node* awal tersebut seperti pada gambar 2.2.



Gambar 2.2 Contoh Permasalahan Dijkstra 2

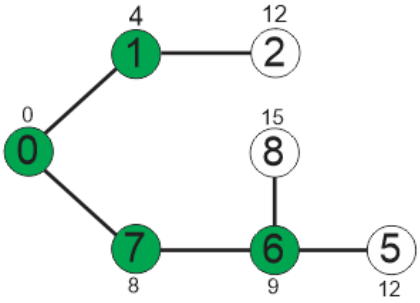
Karena jarak terpendek yang ditemukan adalah 4 seperti pada gambar 2.2, maka *node* 1 menjadi titik awal dan dilanutkan untuk mencari *node* lainnya yang bersebelahan seperti pada gambar 2.3



Gambar 2.3 Contoh Permasalahan Dijkstra 3

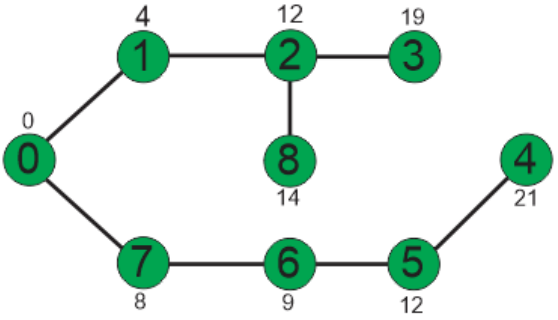
Sama seperti pada tahapan sebelumnya, rute terpendek yang ditemukan adalah ke *node* tujuh dengan jarak sementara 8,

maka *node* tujuh digunakan sebagai *node* awal berikutnya seperti pada gambar 2.4



Gambar 2.4 Contoh Permasalahan Dijkstra 4

Ulangi kembali untuk semua node dan didapatkan hasil rute terpendek seperti pada gambar 2.5.



Gambar 2.5 Contoh Permasalahan Dijkstra 5

Gambar 2.5 memperlihatkan hasil akhir dari pencarian ke semua node dengan jalur terpendek yang dimulai dari *node 0*sebagai *node* awal dan dilakukan pencarian ke seluruh *node*.

2.2.1.3. *Iterative Local Search*

Iterative Local Search atau *iterated local search* merupakan sebuah istilah dalam *computer science* unuk mendifinisikan sebuah modifikasi dari *local search* untuk menyelesaikan permasalahan diskrit [20].

Metode *local search* biasa dapat tersangkut dalam *local minimum*, dimana tidak ada *neighbor* atau solusi yang berdekatan yang dapat meningkatkan hasilnya ditemukan [6].

Kode 2.2 merupakan pseudo code yang umum digunakan untuk *iterated local search* [20].

```
1: s0 = GenerateInitialSolution
2: s* = LocalSearch(s0)
3: repeat
4: s_ = Perturbation(s*, history)
5: s*_ = LocalSearch(s_)
6: s* = AcceptanceCriterion(s*, s*_, history)
7: until termination condition met
```

Kode 2.2 Pseudocode Iterative Local Search

Dalam pemakaiannya, tingkat kesulitan dari *iterative ocal search* terdapat pada penyimpanan sejarah atau *history*. Jika tidak memiliki hal tersebut, maka pemakaian *iterative local searcgnya* tidak memiliki memori atau ingatan. Secara umum, *iterative local search* tanpa ingatan ini yang sering digunakan, meskipun penelitian membuktikan bahwa menggunakan ingatan atau memori dapat meningkatkan hasil pencarian [21].

Pada penelitian tugas akhir ini, tiga buah metode iterative local search digunakan, yaitu *insert*, *swap* dan *delete.* Insert berfungsi untuk menambahkan *node* kedalam jalur yang telah dianggap solusi layak awal, sehingga dapat menambah skor meskipun

menambah waktu tempuh. Swap berfungsi untuk menukar antara satu *node* dalam jalur dengan *node* lainnya sehingga ada kemungkinan dapat meningkatkan skor dan mengurangi waktu tempuh. Dan yang terakhir adalah delete, yang berfungsi untuk menghapus *node* dalam suatu jalur yang dapat mengurangi waktu tmepuh dalam *iterative local search.*

2.2.1.3.1. *Great Deluge Iterative Local Search*

Algoritma *Great deluge* ini ditemukan oleh Gunter Dueck pada 1990 di jerman, lalu dipublikasikan secara internasional pada tahun 1993, merupakan sebuah algoritma umum yang diaplikasikan terhadap permasalahan optimasi [22]. Algoritma ini sangat mirip dengan algoritma *hill climbing* dan *simulated annealing* [22]. *Great Deluge* dinamai berdasarkan ketika terjadi terjadi banjir besar, seseorang yang sedang mendaki gunung akan mencoba mendaki ke arah mana saja yang tidak membasahi kakinya dengan harapan dapat mendapatkan jalan ke atas di saat permukaan air terus naik [23].
Kode 2.3 merupakan pseudo code algoritma *Great deluge* oleh Gunter Dueck.

```
1  Choose an initial configuration
2  Choose the “rain speed” UP > 0
3  Choose the initial WATER-LEVEL > 0
4  Opt:  choose a new configuration which is a stochastic
5          small perturbation of the old configuration
6          compute E :=quality (new configuration)
7          IF E > WATER-LEVEL
8             THEN old configuration := new configuration
9                    WATER-LEVEL := WATER-LEVEL + UP
10 IF a long time no increase in quality or too many iterations
11         THEN stop
12 GOTO Opt
```

Kode 2.3 Pseudocode Great Deluge

Pada penelitian tugas akhir ini, rain speed adalah berapa persen toleransi yang digunakan, dan water-level adalah batas bawah yang akan digunakan dalam penelitian.

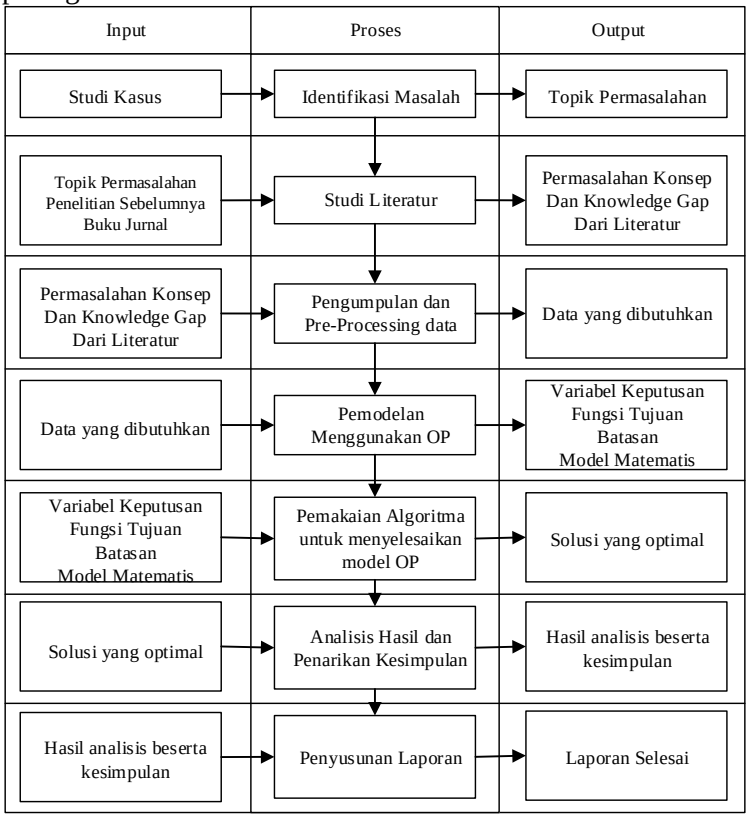
(Halaman ini sengaja dikosongkan)

BAB III
METODE PENELITIAN

Pada bab ini menjelaskan terkait metodologi yang akan digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir ini.

3.1. Diagram Penelitian

Diagram Metodologi penelitian tugas akhir ini dapat dilihat pada gambar 3.1.



Gambar 3.1 Metodologi Penelitian

3.2 Identifikasi Permasalahan

Pada proses ini, identifikasi permasalahan dilakukan, dimana observasi langsung dilakukan terhadap kemacetan yang ada di Surabaya. Kemacetan merupakan masalah utama dalam kota – kota besar yang tidak hanya di Indonesia, bahkan di dunia. Di Surabaya sendiri, sudah ada program lokal yaitu *Advanced Transportation Information System* untuk menangani kemacetan dengan cara memberitahu penggunanya rute atau trayek mikrolet manakah yang paling cepat atau dibutuhkan saat ini. Permasalahan yang ditemukan adalah pemodelan dan optimasi dari rute mikrolet di Surabaya. Pada penelitian ini, permasalahan hanya dibatasi untuk cakupan wilayah sebagian dari Surabaya, dan transportasi Mikrolet.

3.3 Studi Literatur

Pada proses ini dilakukan pengumpulan berbagai jenis referensi baik tertulis, lisan, wawancara, penelitian terdahulu, dan dokumen terkait. Studi Literatur didasarkan pada penelitian terdahulu yang telah dilakukan sebelumnya. Pada tugas akhir ini diusulkan topik tentang pemodelan dan optimasi dari trayek mikrolet di Surabaya dengan kode – kode tertentu dengan metode pemodelan dan algoritma optimasi tertentu. Serta dilakukan pencarian penelitian terdahulu, membaca buku acuan, buku pustaka, tentang metode yang akan dibandingkan dengan metode lain dan juga algoritma yang akan digunakan dibandingkan dengan algoritma lain dan menjadi dasar untuk menentukan metode dan algoritma yang akan dipakai dalam tugas akhir ini dengan menggunakan *gap analysis*. Sesuai dengan yang telah dijelaskan sebelumnya, tugas akhir ini diusulkan dengan metode *Orienteering Problem* dan *Great Deluge Iterative Local Search*.

3.4 Pengumpulan dan Pre-Processing Data

Setelah Proses Studi literature dan mendapatkan yang sesuai, proses berikutnya adalah melakukan pengumpulan data. Data adalah merupakan factor utama dalam pelaksanaan tugas akhir ini. Data untuk penelitian tugas akhir ini bisa didapati melalui situs dishub dan juga menggunakan Google Maps. Data yang dibutuhkan adalah:

- Data node didapatkan dari rute jalan dari trayek – trayek yang dipilih
- Data waktu dalam menit Diperoleh dari Google Maps
- Data Jarak dalam meter diperoleh dari Google Maps
- Data Jumlah mikrolet untuk Skor

Data yang sudah dikumpulkan kemudian di proses untuk dapat digunakan untuk dipakai untuk optimasi nantinya, pre prosesing yang dilakukan adalah mencari rute terpendek dari masing – masing node menggunakan algoritma Dijkstra.

3.5 Pemodelan Menggunakan OP

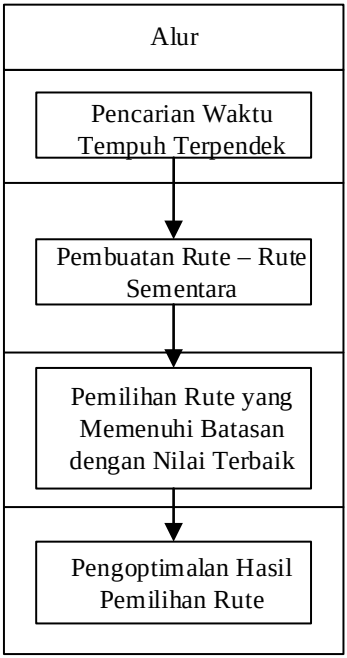
Pada tahapan ini dilakukan pemodelan masalah dengan menggunakan metode *orienteering problem* seperti yang sudah dibahas pada bab sebelumnya, dimana untuk melakukan pemodelan dengan metode tersebut dibutuhkan beberapa hal yang diperlukan:

1. Variabel Keputusan :
    a. Kunjungan dari *node* awal ke *node* berikutnya, dimana pada penelitian ini berarti dari satu tempat pemberhentian ke tempat pemberhentian berikutnya.
2. Fungsi Tujuan:
    a. Memaksimalkan jumlah skor yang didapat, dimana skor diasumsikan merupakan jumlah mikrolet yang dimiliki satu trayek.
    b. Meminimalkan waktu yang ditempuh yaitu T_{max}

3. Fungsi Batasan:
   a. Waktu tempuh tidak boleh melebihi dari yang dialokasikan
   b. Semua node maksimum dikunjungi sekali.
   c. Tidak perlu mengunjungi semua *node*
   d. Semua *node* terhubung antar satu sama lain.

3.6 Pemakaian Algoritma untuk menyelesaikan model OP

Pada proses ini, digunakan beberapa algoritma untuk melakukan pencarian solusi yang optimal dari permasalahan yang dibuat dalam model *orienteering problem*, alur pengerjaan dan algoritma yang digunakan dapat dilihat pada gambar 3.2.



Gambar 3.2 Alur Pemakaian Algoritma untuk menyelesaikan model OP

3.6.1. Pencarian Waktu Tempuh Terpendek

Pada tahapan ini dilakukan pencarian waktu tempuh terpendek antara *node* asal dengan *node* tujuan, dimana dilakukan pencarian jarak terpendek untuk semua *node* dalam bentuk matriks 2x2 dengan menggunakan algoritma Dijkstra. Algortima Dijkstra digunakan karena antara dua buah titik atau *node* terdapat banyak sekali kemungkinan jalur alternatifnya, sehingga yang dicari adalah jalur yang terpendek, secara detilnya akan dijelaskan pada subbab 5.2.1.

3.6.2. Pembuatan Rute – Rute Sementara

Setelah melakukan tahapan sebelumnya, tahapan berikutnya adalah menentukan variable seperti waktu tempuh maksimum, berapa kali melakukan iterasi dan sebagainya, kemudian, dibuatlah kode untuk membuat rute – rute dari *node* awal ke *node* tujuan dengan *node*diantaranya dibuat secara acak. Pembuatan kode untuk pencarian solusi awal yang layak dapat dilihat paa sub bab 4.4.3 pada gambar 4.3.

3.6.3. Pemilihan Rute yang Memenuhi Batasan dengan Nilai Terbaik.

Jika pada tahapan sebelumnya hanya untuk mencari kemungkinan rute yang dapat dilalui secara acak, maka pada tahapan ini dilakukan pemilahan atau seleksi dari hasil pencarian pada tahapan sebelumnya. Pemilahan atau seleksi dilakukan dengan menggunakan algoritma *Roulette wheel*, dimana seleksi dilakukan secara acak, namu jalur yang memiliki nilai yang lebih besar memiliki kemungkinan terpilih sebagai calon jalur yang akan di optimalkan semakin besar pula. Nilai pada konteks ini merupakan skor yang lebih detilnya dijelaskan pada sub bab5.3.3.

3.6.4. Pengoptimalan Hasil Pemilihan Rute

Tahapan terakhir adalah mengoptimalkan hasil pemilihan rute yang sudah dilakukan pada tahapan – tahapan sebelumnya, dimana rute yang optimal pada bahasan ini adalah rute yang memiliki waktu tempuh terpendek, tetapi dengan nilai atau skor yang paling tinggi, dengan urutan prioritas skor terlebih dahulu baru waktu tempuh terpendek. Detil penjelasan tahapan ini dapat dilihat pada sub bab5.3.4.

3.7 Analisis Hasil dan Penarikan Kesimpulan

Pada tahap ini dilakukan Analisa terhadap model yang telah dibuat dan juga algoritma yang digunakan untuk model tersebut. Setelah dianalisa, diambillah kesimpulan terhadap model dan algoritma yang dipakai untuk solusi dari model tersebut dan bagaimana hasilnya terhadap pencarian rute tercepat.

3.8 Penyusunan Laporan

Setelah semua proses dilakukan, tahap terakhir dari proses ini adalah pembuatan laporan tugas akhir sebagai bukti dan dokumentasi atas pengerjaan tugas akhir ini. Proses ini mencakup:

1. **Bab I Pendahuluan**
   Bab ini menjelaskan tentang latar belakang, rmusan masalah, batasan masalah, tujuan penelitian manfaat penelitian serta relevansi penelitian ini.
2. **Bab II Dasar Teori**
   Pada bab ini dijelaskan tentang penelitian terdahulu, studi literature dan juga dasar teori yang digunakan dalam pengerjaan tugas akhir ini.
3. **Bab III Metodologi**

Bab ini menjelaskan langkah – langkah apa yang harus dilakukan dalam pengerjaan tugas akhir ini.

4. **Bab IV Perancangan**
   Bab ini menjelaskan tentang rancangan penelitian, rancangan bagaimana penelitian dilakukan, subyek dan obyek penelitian, pemilihan obyek dan subyek penelitian dan sejenisnya.
5. **Bab V Implementasi**
   Bab ini menjelaskan tentang proses pelaksanaan penelitian, bagaimana penelitian dilakukan, hambatan dan rintangan dalam pelaksanaan tugas akhir ini dan sejenisnya.
6. **Bab VI Analisis dan Pembahasan**
   Bab ini Menjelaskan tentang hasil Analisa dan pembahasan dari permasalahan yang diangkat dalam penelitian tugas akhir ini.
7. **Bab VII Kesimpulan dan Saran**
   Bab ini erisi tentang kesimpulan yang didapat dalam penelitian tugas akhir ini dan saran untuk penelitian – penelitian kedepannya.

(Halaman ini sengaja dikosongkan)

BAB IV
PERANCANGAN

Bab ini menjelaskan tentang perancangan penelitian tugas akhir dalam membuat model peramalan. Bab ini berisikan proses pengumpulan data, praproses data, serta pengolahan dan pemodelan data.

4.1 Pengumpulan Data

Dalam proses memodelkan *Orienteering Problem* dabat dibuat dengan merancang *Network Model* dalam bentuk *graph.* Data – data yang digunakan langsung diambil dari situs resmi Dinas Perhubungan Kota Surabaya, yaitu data trayek dan jumlah angkot.

4.1.1 Pengumpulan data

Tabel 4.1 berisi tentang rute yang ditempuh masing -masing angkot beserta kodenya, semua jalur diambil dari situs resmi dinas perhubungan Surabaya, mulai dari rute berangkat sampai ke tujuan akhir, lalu kembali lagi ke tempat awal.

Tabel 4.1 Rute Masing - Masing Trayek

Kode Trayek	Rute
M	**Berangkat:** Terminal Joyoboyo – Jl. Diponegoro – U Turn – Jl. Raya Darmo – Jl. Marmoyo – Jl. Darmo Kali – Jl. Dinoyo – Jl. Polisi Istimewa – U Turn – Jl. Pajajaran – Jl. Sriwijaya – Jl. Pandegiling – Jl. Keputran – Jl. Embong Sono Kembang – Jl. Embong Cerme – Jl. Embong Kemiri – Jl. Kayun – Jl. Pemuda – Jl. Yos Sudarso – Jl. Walikota Mustadjab – Jl. Genteng Kali – Jl. Undaan Kulon Jl.

	Pengampon – Jl. Bunguran – Jl. Stasiun Kota – Jl. Kebon Rojo – Jl. Pahlawan – Jl. Tembaan – Jl. Bubutan – Jl. Indrapura – Jl. Krembangan Barat – Jl. Krembangan Timur – Jl. Rajawali – Jl. Kasuari – Jl. Kalimas Barat – Pangkalan Kalimas Barat (Pangkalan Akhir) **Kembali:** Pangkalan Kalimas Barat – Jl. Kalimas Barat – Jl. Kasuari – Jl. Garuda – Taman Jayanegara (JMP) – Jl. Jembatan Merah – Jl. Veteran – Jl. Kebon Rojo – Jl. Stasiun Kota – Jl. Bunguran – Jl. Gembong – Jl. Pecindilan – Jl. Undaan Wetan – Jl. Ambengan – Jl. Jakgung Suprapto – Jl. Walikota Mustadjab – Jl. Pemuda – Jl. Kayun – Jl. Sono Kembang – Jl. Panglima Sudirman – Jl. Urip Sumoharjo – Jl. Pandegiling – Jl. Keputran – Jl. Dinoyo – Jl. Bung Tomo – Jl. Upojiwo – Jl. Ratna – Jl. Ngagel – Jl. Darmo Kali – Jl. Marmoyo – Jl. Raya Darmo – Jl. Raya Wonokromo – U. Turn – RSI – Jl. Raya Wonokromo – Terminal Joyoboyo.
S	**Berangkat:** Terminal Joyoboyo – Jl. Raya Wonokromo – U Turn Kebun Binatang – Jl. Raya Wonokromo – Jl. Jagir Wonokromo – Jl. Ngagel – Jl. Ngagel Rejo Kidul – Jl. Bratang Gede – Jl. Barata Jaya – Jl. Bratang Binangun – Jl. Manyar – U Turn – Jl. Raya Manyar – Jl. Raya Nginden – U Turn – Jl. Raya Nginden – Jl. Terminal Bratang (Pangkalan Akhir). **Kembali**: Terminal Bratang – Jl. Barata Jaya – Jl. Bratang Gede – Jl. Ngagel Rejo Kidul – Jl. Raya Ngagel – Jl. Stasiun Wonokromo – U Turn RSI – Jl. A. Yani – Jl. Raya Wonokromo – Terminal Joyoboyo (Pangkalan Akhir).
U	**Berangkat:** Terminal Joyoboyo – Jl. Raya Wonokromo – U Turn – Jl. Raya Wonokromo – Jl. A. Yani – Jl. Bendul Merisi – Jl. Jagir Sidoresmo XII – Jl. Jagir Wonokromo – Jl. Panjang Jiwo – Jl. Raya Kedung Baruk – Jl. Kedung Baruk – Jl. Kedung Asem – Jl. Raya Kedung Asem – Jl. Penjaringan Sari – Jl. Kedal Sari – Jl. Wonorejo (Pangkalan Akhir). **Kembali**:

	Pangkalan Wonorejo – Jl. Wonorejo – Jl. Kendal Sari – Jl. Penjarigan Sari – Jl. Raya Kedung Asem – Jl. Kedung Asem – Jl. Kedung Baruk – Jl. Raya Kedung Baruk – Jl. Panjang Jiwo – Jl. Jagir Wonokromo – Jl. Stasiun Wonokromo – U Turn – Jl. Raya Wonokromo – Terminal Joyoboyo.
UBB	**Berangkat:** Pangk. Ujung Baru – Jl. Perak Timur – Jl. Prapat Kurung – Jl. Kalimas Baru – Jl. Jakarta – Jl. Petekan – Jl. Kebalen Timur – Jl. Indrapura Pasar – Jl. Rajawali – Jl. Branjangan – Jl. Cendrawasih – Jl. Veteran – Jl. Kebon Rojo – Jl. Stasiun Kota – Jl. Semut Kali – Jl. Peneleh – Jl. Makam Peneleh – Jl. RP. Soenaryo Gondokusumo – Jl. Undaan Kulon – Jl. Undaan Wetan – Jl. Kalisari – Jl. Jagung Suprapto – Jl. Ambengan – Jl. Wijaya Kusuma – Jl. Walikota Mustajab – Viaduk – Gubeng pojok – Jl. Sumatera – Jl. Nias – Jl. Jawa – Jl. Biliton – Jl. Sulawesi – Jl. Lombok – Jl. Ngagel – Jl. Upojiwo – Jl. Ngagel Kencono (BAT) – Jl. Ngagel Selatan – Jl. Manyar – Jl. Nginden Kota – Terminal Bratang. **Kembali**: Term. Bratang – Jl. Barata Jaya – Jl. Bratang Binangun – Jl. Ngagel Jaya Selatan – Jl. Krukah Utara – Jl. Ngagel Jaya Barat – Jl. Ngagel Timur IV – Jl. Ngagel Timur – Jl. Kalibokor – Jl. Pucang Anom – Jl. Kalibokor I – Jl. Ngagel – Jl. Sulawesi – Jl. Raya Gubeng – Jl. Karimun Jawa – Jl. Embong Cerme – Jl. Kayun – Jl. Pemuda – Jl. Yos Sudarso – Jl. Walikota Mustajab – Jl. Wijaya Kusuma – Jl. Ambengan – Jl. Jaksa Agung Suprapto – Jl. Kalisari – Jl. Kalianyar – Jl. Undaan Wetan – Jl. Undaan Kulon – Jl. Pengampon – Jl. Bunguran – Jl. Psr. Atom – Jl. Waspada – Jl. Karet – Jl. Jemb. Merah – Jl. Veteran – Jl. Niaga Samping – Jl. Sikatan – Jl. Krembangan – Jl. Krembangan Makam – Jl. Indrapura – Jl. Rajawali – Jl. Pesapen – Jl. Pesapen Selatan – Jl. Indrapura Pasar – Jl. Kalimas Baru – Jl. Ujung Baru (Pangkalan Akhir).
TWM	**Berangkat:** Pangkalan Tambak Wedi – Jl. Kedinding Lor – Jl. Kedung Cowek – Jl. Kenjeran – Jl. Tambak Rejo – Jl. Kapas Krampung – Jl. Karang Asem – Jl. Bronggalan – Jl. Tambang Boyo – Jl. Pacar Keling – Jl. Tambang

	Boyo – Jl. Prof Dr Moestopo – Jl. Raya Dharmahusada Indah – Jl. Raya Kertajaya Indah – U Turn – Jl. Raya Kertajaya Indah – Jl. Kertajaya Indah Tengah – Jl. Manyar Kertoadi – Jl. Gebang Putih – Jl. Arif Rahman Hakim – Jl. Keputih Tegal – Jl. Keputih – Pangkalan Keputih (Pangkalan Akhir). **Kembali**: Pangkalan Keputih – Jl. Keputih Tegal – Jl. Arif Rahman Hakim – Jl. Manyar Kertoadi – Jl. Kertajaya Indah Tengah – Jl. Raya Kertajaya – Jl. Dharmahusada – Jl. Prof Dr Moestopo – Jl. Kedung Sroko – Jl. Pacar Keling – Jl. Kalasan – Jl. Jolotundo – Jl. Bronggalan – Jl. Karang Asem – Jl. Kapas Krampung – U Turn – Jl. Kapas Krampung – Jl. Putro Agung Wetan – Jl. Kedung Cowek – Jl. Kedinding Lor – Jl. Tambak Wedi – Pangkalan Tambak Wedi (Pangkalan Akhir).
WB	**Berangkat:** Terminal Bratang – Jl. Bratang Jaya – Jl. Raya Bratang Binangun – Jl. Ngagel Jaya Selatan – Jl. Manyar – Jl. Menur Pumpungan – Jl. Klampis Jaya – Jl. Raya Kertajaya Indah Tengah – Jl. Raya Manyar Kertoarjo – Jl. Kertajaya – Jl. Gubeng Kertajaya – Jl. Karang Menur Timur – Jl. Dharmawangsa – Jl. Prof. Dr. Mustopo – Jl. Tapak Siring – Jl. Residen Sudirman – Jl. Kusuma Bangsa – Jl. Kapasari – Jl. Simokerto – Jl. Simolawang Baru – Jl. Sidodadi – Jl. Pegirian – Jl. Nyamplungan – Jl. Dana Karya – Jl. Karang Tembok – Jl. Wonosari – Jl. Wonosari Lor – Jl. Wonokusumo – Jl. Bulak Sari – Jl. Bulak Banteng (Pangkalan Akhir). **Kembali**: Pangkalan Jl. Bulak Banteng – Jl. Bulak Sari – Jl. Wonokusumo – Jl. Wonosari Wetan – Jl. Wonosari Lor – Jl. Wonosari – Jl. Karang Tembok – Jl. Sododadi – Jl. Simolawang Baru – Jl. Simokerto – Jl. Kapasari – Jl. Ngaglik – Jl. Tambaksari – Jl. Residen Sudirman – Jl. Pacar Keling – Jl. Tapak Siring – Jl. Prof. Dr. Mustopo – Jl. Dharma Husada – Jl. Gubeng Kertajaya – Jl. Kertajaya – Jl. Manyar Kertoarjo – Jl. Raya Kertajaya Indah – Jl. Klampis Jaya – Jl. Menur Pumpungan – Jl. Manyar – Jl.

	Raya Nginden – U Turn – Jl. Raya Nginden – Terminal Bratang (Pangkalan Akhir).

4.1.2 Data Jumlah Angkot

Berdasarkan dari situs resmi dinas perhubungan Surabaya, dapat diketahui kode, rute dan jumlah angkot yang terdapat dalam trayek tersebut, dimana jumlah angkot dijadikan sebagai skor untuk tiap *node* yang dilewati. Tabel 4.2 merupakan jumlah angkot yang melewati masing – masing trayek tersebut.

Tabel 4.2 Jumlah Angkot Masing - Masing Trayek

Kode Trayek	Asal - Tujuan	Jumlah Angkot
M	Kalimas Barat / Petekan - Manukan Kulon (PP)	133
S	Dukuh Kupang - Benowo (PP)	86
U	Kalimas Barat - Bratang (PP)	124
UBB	Joyoboyo - Tubanan - Manukan (PP)	32
TWM	Kalimas Barat - Benowo (PP)	18
WB	Benowo - Ujung Baru (PP)	71

4.2 Perancangan *Network Model*

Perancangan *Network Model* diawali dengan menggambakan rute trayek yang digunakan, setiap rute trayek digambarkan di atas peta digital menggunakan Corel Draw sebagai gambaran awal jalur tiap trayek sekaligus untuk mengetahui pada jalur mana saja tiap trayek bersinggungan. Sebelum membuat gambaran lintasan atau jalur trayek tersebut, terdapat beberapa asumsi yang dipakai untuk menggambarkan jalur tersebut, asumsi tersebut dapat dilihat pada tabel 4.3

Tabel 4.3 Asumsi Pada Tiap Trayek

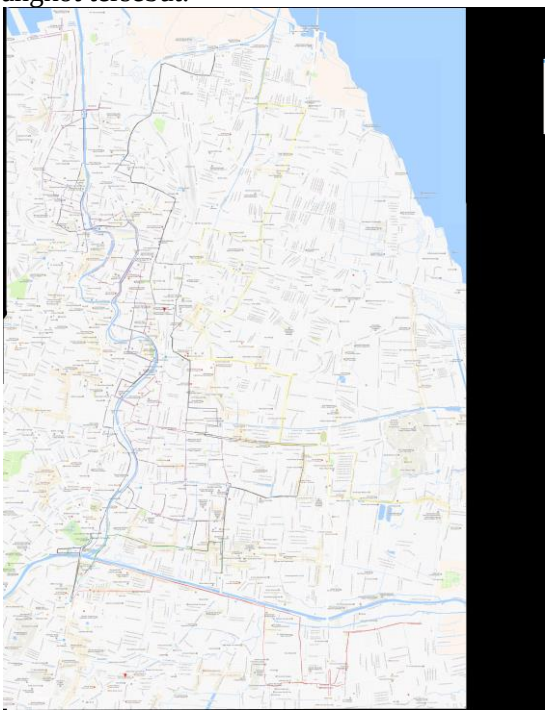
1	**Asumsi:** Pangkalan Tambak Wedi di asumsikan terdapat pada jalan Tambak Wedi Barat dekat toko suroso
	Alasan: Pada google maps tak ada lokasi pangkalan tambak wedi
	Trayek: TWM
2	**Asumsi:** Pangkalan Bulak Banteng diasumsikan terdapat pada Jl. Bulak Banteng Baru No.17
	Alasan: Pada google maps tak ada lokasi pangkalan bulak banteng
	Trayek: WB
3	**Asumsi:** Pangkalan Keputih diasumsikan terdapat pada Jl. Keputih Tegal No.8
	Alasan: Pada google maps tak ada lokasi pangkalan keputih
	Trayek: TWM
4	**Asumsi:** Pangkalan Wonorejo diasumsikan terdapat pada Jl. Wonorejo Rungkut No.1
	Alasan: Pada google maps tak ada lokasi pangkalan Wonorejo
	Trayek: U

Setelah ditambahkan asumsi dari tabel 4.3, dibuatlah rute dari masing – masing kode angkot dengan keterangan warna dan kode seperti pada tabel 4.4.

Tabel 4.4 Warna Trayek

Kode Trayek	Warna Pada Model
M	Biru
S	Hijau
TWM	Kuning
U	Merah
UBB	Ungu
WB	Hitam

Gambar 4.1 merupakan gambaran dari jalur yang dilewati keenam angkot tersebut.



Gambar 4.1 Jalur Trayek

4.2.1. Penentuan Node, Jarak Antar Node dan Score Tiap Node

Setelah membuat gambaran jalur dari keenam trayek yang telah disebutkan, dapat ditentukan beberapa *node*, jarak antar node dan *score* dengan beberapa asumsi sebagai berikut.

a. Setiap jalur trayek yang saling bersinggungan akan menjadi *node* dimana *node* tidak ditempatkan tepat di persimpangan jalan. Hal ini bertujuan untuk mencegah timbulnya kemacetan lalu lintas.
b. Penempatan node diprioritaskan di tempat yang berpotensi padat pengunjung, misalnya pusat perbelanjaan, bank, kantor.
c. Jarak antar *node* tidak boleh lebih dari satu kilo meter (< 1 km). Hal ini diadaptasi dari Keputusan Direktur Jenderal Perhubungan Darat Nomor 271/HK.105/DJRD/96 dimana jarak maksimal antara halte dan/atau Tempat Perhentian Bus (TPB) adalah 1 km untuk daerah pinggiran.
d. *Score* tiap *node* dihitung berdasarkan jumlah angkot dari keenam trayek yang melewati *node* tersebut.

Dengan beberapa asumsi di atas, terbentuklah 235 node dari keenam trayek yang digunakan pada penelitian tugas akhir ini dan dapat dilihat pada Gambar 4.2.

Gambar 4.2 Node Trayek

Deskripsi dan keterangan tambahan tiap node dapat dilihat pada LAMPIRAN A.

4.3 Model *Orienteering Problem*

Setelah *network model* yang telah dibuat sebelumnya, model yang sesuai dapat terbentuk dengan menggunakan model *Orienteering Problem* (OP). Model OP diformulasikan ke

dalam pemrograman integer dengan penentuan variabel keputusan, fungsi tujuan, dan batasan.

4.3.1. Variabel Keputusan

Untuk menyelesaikan permasalahan dalam penelitian tugas akhir ini, didapati bahwa variable keputusannya adalah kunjungan dari satu *node* ke *node* lainnya, yang dapat direpresentasikan dengan x_{ij}. Dimana x_{ij} akan bernilai 1 apabila kunjungan dimulai dari *node i* dan berakhir di *node j*. Jika tidak, maka x_{ij} bernilai 0. Tabel variable keputusan untuk model *Orienteering Problem* dapat dilihat pada LAMPIRAN B.

4.3.2. Fungsi Tujuan

Fungsi Tujuan dalam penelitian tugas akhir ini didasari dari fungsi tujuan *Orienteering Problem* yaitu mencari rute terpendek dengan memaksimumkan skor. Rumus dasar fungsi tujuan OP dapat dilihat sebagai berikut.

$$Maximize \sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} S_i X_{ij}$$

Dimana:
S_i: Skor pada *node* i
X_{ij}: Kunjungan dari *node i* ke *node j*
i: *node* 2, 3, 4, …, N-1
j: *node* 2, 3, 4, …, N

dengan memasukkan N sejumlah node yang dibuat sebelumnya yaitu 235, dan menentukan node awal dan akhir untuk rute perjalanan, model matematis dapat dibuat pada bab berikutnya.

4.3.3. Batasan Model

Orienteering Problem memiliki empat buah Batasan yaitu:

a. Batasan pertama adalah rute harus dimulai dari *node* n dan berakhir di *node* m. yang berarti *node* awal dan akhir harus berbeda, dan untuk memastikan bahwa rute yang dipilih dimulai dari lokasi n dan berakhir di lokasi m. berikut merupakan rumus dasar Batasan pertama *Orienteering Problem*

$$\sum_{j=2}^{|N|} X_{1j} = \sum_{i=1}^{|N|-1} X_{i|N|} = 1$$

b. Batasan kedua adalah memastikan setiap jalur terhubung dan paling banyak dikunjungi sekali. Batasan ini memastikan seluruh *node* dalam *network model* terhubung dan hanya dapat dikunjungi sekali. Terhubungnya *node* ditandai dengan percabangan dan pertemuan. Rumus dasar dari Batasan ini adalah sebagai berikut.

$$\sum_{i=1}^{|N|-1} X_{ik} = \sum_{j=2}^{|N|} X_{kj} \leq 1; \forall k = 2, \dots, (|N| - 1)$$

c. Batasan ketiga adalah total waktu tempuh dari tempat awal ke tempat akhir tidak boleh melibihi alokasi waktu yang ditentukan. Berikut merupakan rumus dasarnya.

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} X_{ij} \leq TMax$$

d. Tidak boleh ada dan terjadinya subtour, kondisi dimana *node* awal merupakan *node* akhir juga.

4.4 Pencarian Solusi dari Model

Pada tahap ini, pencarian solusi model dengan *Great Deluge Iterative Local Search* dilakukan. Dimana pencarian *initial solution* nya dilakukan secara stochastic atau random. *Initial solution* dilakukan dengan tujuan untuk mencari solusi awal yang layak, kemudian di ambil beberapa solusi dan dari beberapa solusi tersebut akan di optimalkan. Untuk mencari solusi dari model, dibutuhkan perancangan pre-processing data, perancangan solusi awal yang layak, perancangan Seleksi Solusi, dan perancangan *Iterative local search*.

4.4.1. Perancangan Pre-Processing Data

Untuk melakukan pencarian solusi, data waktu tempuh yang sudah dibuat dalam tabel tadi harus diubah kedalam matriks dua dimensi dan disimpan dalam format *comma separated value* atau .csv agar dapat diolah di dalam java, pembuatan matriks dua dimensi dilakukan dengan menggunakan algoritma Dijkstra untuk mencari rite terpendek antara semua *node* untuk mendapatkan jarak waktu dari *node* awal ke semua *node* sehingga dapat dilakukan optimasi di tahapan berikutnya.

4.4.2. Perancangan Solusi Awal yang Layak

Solusi awal yang layak merupakan pembentukan solusi sementara, pembentukan populasi awal dan pemilihan solusi awal yang layak ini mengacu pada gambar 4.3 yaitu pseudo code yang diusulkan oleh Tasgetiren [24]. dimana *node* awal dan akhir, tMax, telah ditetapkan terlebih dahulu. Setelah menentukan ketiga hal tersebut, tetapkan berapa kali iterasi

yang akan dilakukan, serta buat populasi awalnya menjadi 0 karena nantinya akan diisi atau ditambahkan. Kemudian, buatlah angka random bernilai Rn antara 1 sampai N dimana N adalah *node* akhir yang sudah didefinisikan sebelumnya. Kemudian buatlah sebuah list R_l, kemudian, antara 1 sampai N buatlah secara random sebuah sub list Rs dimana Rs mengambil sebagian dari random list R_l secara sebagian awal Rn, kemudian hitung total waktu tempuh, dari sub list Rs tadi, jika waktu tempuh kurang dari sama dengan tMax, maka iterasi dilanjutkan, dan populasi awal ditambah 1, dan looping akan terus berjalan sampai loop mencapai batas loop yang sudah ditentukan di awal.

```
Define tMax
Define maxLoop
Define jumlahPop
Set loop = 0 and pop = 0;
Do {
    Produce a random number, Rn, between [1,N]
    Produce a list, Rl, between [1,N] randomly
    Generate a sub list, Rs, by taking the first
        Rn part of the random list Rl
    Compute the total traveling time, tRs, of the
        sub list Rs
    If tRs <= tMax, loop = loop + 1
        pop = pop + 1
} while (loop < maxLoop or pop < jumlahPop)
```

Gambar 4.3 Pseudocode Initial Population

4.4.3. Perancangan Seleksi Solusi

Pada tahap ini, setelah melakukan pembentukan populasi awal dan pemilihan solusi awal yang layak di optimasikan, dilakukan pemilihan atau seleksi secara random untuk menjadi calon yang akan di optimasikan, pemilihan dilakukan secara random, akan

tetapi, jalur dengan skor yang tinggi akan memiliki kemungkinan yang besar untuk terpilih dikarenakan pemilihan dilakukan dengan menggunakan *roulette wheel selection*, dimana pemilihan akan didasarkan pada skor, dengan begitu, skor yang besar kemungkinan besar akan terpilih dan dilanjutkan untuk di optimasikan dengan *iteratie local search*

4.4.4. Perancangan *Iterative Local Search*

Tahapan berikutnya adalah melakukan *iterative local search*, dimana proses *insert, swap,* dan *delete* dilakukan, optimasi dilakukan terhadap hasil seleksi dari populasi awal yang layak, dengan harapan ditemukan solusi yang lebih baik dan menjadi solusi terbaik. Dalam *iterative local search* sendiri terdapat loop sendiri untuk melakukan berapa kali *insert, swap,* dan *delete* yang dipilih secara random terhadap satu jalur, yang kemudian dilanjutkn ke jalur berikutnya dan seterusnya sampai jalurnya habis. Sedangkan untuk *Great deluge* sendiri, disini semua solusinya tidak diambil yang terbaik secara langsung, namun diberikan level toleransi, dimana pada penelitian tugas akhir ini digunakan nilai statis toleransinya yaitu sebanyak 5%.

(Halaman ini sengaja dikosongkan)

BAB V
IMPLEMENTASI

5.1 Model Matematis

Bagian ini menjelaskan tentang model matematis dari tujuan dan Batasan dari model *Orienteering Problem*. Skenario yang digunakan adalah dengan *node* awal 1 dan *node* akhir 125 (Pangkalan Ujung Baru - Jl. Raya Wonorejo No.2 D (pangkalan wonorejo)), pemilihan titik awal dan akhir tersebut dipilih Karena kedua titik tersebut memiliki 4 buah kode angkot yang melewati kedua *node* tersebut dan jarak yang panjang.

5.1.1. Fungsi Tujuan

Pada penelitian tugas akhir kali ini, *node* awal yang digunakan adalah 1 dan *node* akhir yang digunakan adalah 125. Dan terdapat 235 *node* secara keseluruhan. Maka rumus matematis nya dapat dituliskan sebagai berikut.

$$Maximize \sum_{i=1}^{234} \sum_{j=2}^{235} S_i X_{ij}$$

5.1.2. Batasan Model

Seperti yang sudah dijabarkan sebelumnya, terdapat empat Batasan, namun yang memiliki model matemtais hanya ada tiga, berikut merupakan model matematis dari tiap Batasan

a. Batasan pertama, rute harus dimulai dari *node* 1 dan berakhir di *node* 125. Dengan rumus seperti berikut untuk *node* awal.

$$\sum_{j=2}^{125} X_{12} = 1$$

Dan *node* akhirnya

$$\sum_{i=1}^{124} X_{1.125} = 1$$

b. Batasan kedua, Setiap jalur terhubung dan setiap *node* dapat dikunjungi maksimal satu kali. Dengan rumus dasar untuk pertemuan antar *node* sebagai berikut.

$$\sum_{i=1}^{124} X_{ik} \leq 1; \forall k = 2, \dots, 124$$

Dan untuk percabangan antar *node* sebagai berikut.

$$\sum_{i=2}^{125} X_{ik} \leq 1; \forall k = 2, \dots, 125$$

c. Batasan ketiga, Total waktu tempuh tidak boleh melebih waktu yang telah ditentukan. Dengan rumus untuk permasalahan model ini adalah sebagai berikut.

$$\sum_{i=1}^{234} \sum_{j=2}^{235} t_{ij} X_{ij} \leq 120$$

5.2 Pre-Processing Data

Tahap pre-processing data dilakukan dengan mengolah data waktu tempuh antar node agar dapat dibaca dalam *java*.

Pengolahan dimulai dengan membentuk matriks waktu tempuh antar *node* menggunakan Microsoft Excel, kemudian dilakukan pencarian waktu tempuh terpendek dari 1 *node* ke seluruh *node* dengan menggunakan algoritma Dijkstra.

5.2.1. Pembuatan Matriks Waktu Tempuh

Rekapan data berupa tabel yang telah dilakukan tidak bisa langsung saja dibaca oleh java, harus diubah terlebih dahulu bentuknya kedalam matriks dua dimensi agar bisa diolah ke tahap selanjutnya.

Dikarenakan banyaknya *node* yang dipakai dalam penelitian tugas akhir ini sebanyak 235, maka matriks yang harus dibuat adalah sbesar 235x235 yang berisikan waktu dari *node* 1 ke *node 2*, dan seterusnya.

Tabel 5.1 Matriks dua dimensi yang akan di dijkstra

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	0	0	0	0	0	0	0	0
2	1	0	2	3	5	6	7	9	10	11
3	17	16	0	1	3	4	5	7	8	9
4	16	15	0	0	2	3	4	6	7	8
5	14	13	0	0	0	1	2	4	5	6
6	13	12	0	0	0	0	1	3	4	5
7	11	10	0	0	0	0	0	1	2	3
8	11	10	0	0	0	0	0	0	2	3
9	9	8	0	0	0	0	0	0	0	1
10	8	7	0	0	0	0	0	0	0	0

Tabel 5.1 merupakan potongan dari matriks 235x235, dimana yang berisi angka merupakan waktu dari *node* awal ke *node* akhir. Yang berisikan 0 belum diketahui jaraknya terkecuali apabila tujuannya adalah *node* itu sendiri yang berarti tidak ada. Setelah dibuat matriksnya disimpan dalam bentuk .csv.

5.2.2. Pemakaian Algoritma Dijkstra

Untuk mendapatkan *initial feasible solution* yang terbaik, alangkah lebih baik untuk diketahui waktu antar *node* dibandingkan hanya mengetahui sebagian *node* yang ada, dengan begitu, dapat meningkatkan jumlah *feasible solution* yang ditemukan dan meningkatkan kemungkinan ditemukannya *feasible solution* dikarenakan pencarian *feasible solution* dilakukan dengan cara random atau stokastk. Untuk mengimplementasikan hal tersebut, dilakukan algoritma Dijkstra menggunakan java dengan kodingan sebagai berikut.

```
1. static final int V = 235;
2. dist[src] = 0;
```

Kode 5.1 Kode Dijkstra 1

Kode 5.1 menjelaskan bahwa V adalah jumlah *node* yang terdapat dan src merupakan variable source sebagai *node* sumber yang akan dicari waktu tempuhnya.

```
1. int minDistance(int dist[], Boolean sptSet[]
   {
   // Initialize min value
2. int min = Integer.MAX_VALUE, min_index = -1;
3.      for (int v = 0; v < V; v++) {
4.         if (sptSet[v] == false && dist[v] <=
               min) {
5.                min = dist[v];
6.                min_index = v;
7.             }
8.          }
9.          return min_index;
10.     }
```

Kode 5.2 Kode Dijkstra 2

Kode 5.2 memiliki fungsi untuk menemukan *node* dengan nilai waktu tempuh minimum pada node yang belum memiliki waktu tempuh terpendek.

```
1. void printSolution(int dist[], int n) {
2. //System.out.println("Vertex   Distance from
   Source");
3. for (int i = 0; i < V; i++) {
4. //System.out.println(i + " \t\t " +
   dist[i]);
5. System.out.print(dist[i]+",");
6.          }
7.      }
```

Kode 5.3 Kode Dijkstra 3

Kode 5.3 memiliki fungsi untuk mencetak seluruh solusi yang dihasilkan oleh algoritma Dijkstra dengan koma sebagai pemisahnya.

```
1. void dijkstra(int graph[][], int src) {
2. int dist[] = new int[V]; // The output ar-
   ray. dist[i] will hold
3. // the shortest distance from src to i
4. // sptSet[i] will true if vertex i is in-
   cluded in shortest
5. // path tree or shortest distance from src
   to i is finalized
6. Boolean sptSet[] = new Boolean[V];
7. // Initialize all distances as INFINITE and
   stpSet[] as false
8. for (int i = 0; i < V; i++) {
9.       dist[i] = Integer.MAX_VALUE;
10.   sptSet[i] = false;
11.       }
12.   // Distance of source vertex from itself
   is always 0
13. dist[src] = 0;
14.   // Find shortest path for all vertices
15. for (int count = 0; count < V - 1; count++)
   {
16.   // Pick the minimum distance vertex from
   the set of vertices not yet processed. u is
   always equal to src in first iteration.
17. int u = minDistance(dist, sptSet);
18.   // Mark the picked vertex as processed
19. sptSet[u] = true;
20.   // Update dist value of the adjacent ver-
   tices of the picked vertex.
21. for (int v = 0; v < V; v++) // Update
   dist[v] only if is not in sptSet, there is
   an edge from u to v, and total weight of
   path from src to v through u is smaller than
   current value of dist[v]
22.             {
23. if (!sptSet[v] && graph[u][v] != 0
```

```
24. && dist[u] != Integer.MAX_VALUE
25.  && dist[u] + graph[u][v] < dist[v]) {
26.  dist[v] = dist[u] + graph[u][v];
27.                     }
28.                 }
29.             }
30.  // print the constructed distance array
31.             printSolution(dist, V);
32.         }
```

Kode 5.4 Kode Dijkstra 4

Kode 5.4 merupakan inti dari algoritma Dijkstra, pada kodingan ini akan dilakukan pembuatan array satu dimensi untuk menyimpan nilai waktu tempuh. Lalu dilakukan proses algoritma Dijkstra tersebut untuk mencari jalur terpendek.

```
1.  static int[][] intArray = new
    int[235][235];
2.  // Driver method
3.  public static void main(String[] args)
    throws FileNotFoundException, IOException {
4.  String Array;
5.  BufferedReader bufRdr = new Buff-
    eredReader(new FileReader("src/tes1.csv"));
6.  ArrayList<String[]> lines = new Ar-
    rayList<String[]>();
7.  while ((Array = bufRdr.readLine()) != null)
    {
8.  lines.add(Array.split(","));
9.          }
10. //Convert our list to a String array
11.  String[][] array = new
     String[lines.size()][0];
12.  lines.toArray(array);
13. //Convert String array to Integer array
```

```
14. for (int i = 0; i < 235; i++) {
15.  for (int j = 0; j < 235; j++) {
16.  intArray[i][j] = Integer.parseInt(ar-
     ray[i][j]);
17.              }
18.         }
```

Kode 5.5 Kode Dijkstra 5

Kode 5.5 berfungsi untuk membaca file csv yang merupakan inputan berupa matrix dua dimensi yang telah dibuat dan akan dilakukan algoritma Dijkstra. File inputan yang digunakan dalam kasus ini adalah tes1.csv

```
1. ShortestPath t = new ShortestPath();
2. for (int i = 0; i < 235; i++) {
3. for (int j = 0; j < 30; j++) {
4. //int graph[][] = new int[][] {{intAr-
   ray[i][j]}};
5. t.dijkstra(intArray, i);
6. System.out.println();
7.          }
```

Kode 5.6 Kode Dijkstra 6

Kode 5.6 berfungsi untuk memanggil metode Dijkstra yang sudah dijabarkan pada kodingan sebelumnya. Hasil outputan dari ini adalah matriks dua dimensi yang nanti dapat di kopi ke dalam file excel baru. Dalam kasus ini outputannya disimpan dalam file Dijkstra.csv.

5.3 Pencarian Solusi awal Model OP

Setelah mendapatkan waktu tempuh antar *node* secara keseluruhan, solusi dapat dicari. Pencarian solusi dilakukan engan mencari solusi awal sementara yang dilakukan secara random atau stokastik. Setelah dilakukan random, dilakukan seleksi menggunakan *roulette wheel section, setelah dilakukan seleksi*, dapat dilakukan optimasi menggunakan *Great Deluge Iterative Local Search.*

5.3.1 Penentuan Variabel

Bagian ini menjelaskan apa saja yang harus didefinisikan terlebih dahulu sebelum membuat metode, *main class* dan *looping*. Variabel yang didefinisikan disini akan terpakai sepanjang pencarian hasil yang optimal.

```
1. static int nAwal = 1;
2. static int nAkir = 125;
3. static int jmlNode = 235; //Sementara
   menganggap node awal pasti 1 dan node
   akhir, N, adalah jumlah node
4. static int[][] untukjalur = new
   int[jmlNode][jmlNode];
5. static int[] untukscore = new int[jmlNode];
```

Kode 5.7 Deklarasi Variabel

Kode 5.7 mendefinisikan berapa banyak *node* yang digunakan, *node* awalnya darimana, dan dimana *node* itu berakhir. Lalu definisikan array dua dimensi untuk dimasukkan inputtan dari Dijkstra, serta array satu dimensi untuk menyimpan skor dari masing – masing *node.*

```
1. static ArrayList<ArrayList<Integer>> feasSo-
   lution = new ArrayList<ArrayList<Inte-
   ger>>();
2. static ArrayList<Integer> feasSolution_Jarak
   = new ArrayList<Integer>();
3. static ArrayList<Integer> feasSolution_Score
   = new ArrayList<Integer>();
4. static ArrayList<ArrayList<Integer>> pemili-
   hanHasil = new ArrayList<ArrayList<Inte-
   ger>>();
5. static ArrayList<Integer> pemilihanHasil-
   Waktu = new ArrayList<Integer>();
6. static ArrayList<Integer> pemili-
   hanHasilScore = new ArrayList<Integer>();
```

Kode 5.8 Deklarasi Arraylist dan Array of Arraylist

Kode 5.8 merupakan list dari arraylist apa saja yang digunakan dalam pencarian *initial feasible solution*.

```
1.  String ini;
2.  BufferedReader jalur = new Buff-
    eredReader(new FileReader("src/generated-
    path.csv"));
3.  ArrayList<String[]> lines = new Ar-
    rayList<String[]>();
4.  while ((ini = jalur.readLine()) != null) {
5.  lines.add(ini.split(","));
6.          }
7.  //Convert list to a String array
8.  String[][] array = new
    String[lines.size()][0];
9.  lines.toArray(array);
10. //Convert String array to Integer array
11. for (int i = 0; i < array.length; i++) {
12. for (int j = 0; j < array.length; j++) {
13. untukjalur[i][j] = Integer.parseInt(ar-
    ray[i][j]);
14.             }
15. }
```

Kode 5.9 Kode Reader CSV

Kode 5.9 berfungsi untuk membaca jalur yang telah dihasilkan oleh algoritma Dijkstra yang sudah dilakukan sebelumnya, kodingan ini membacanya dan waktu tempuh antar *node* sudah siap digunakan.

```
1.       BufferedReader skor = new Buff-
   eredReader(new FileReader("src/skor.csv"));
2.
3.          String itu;
4.          int o = 0;
5.          while ((itu = skor.readLine()) !=
   null) {
6.              untukscore[o] = Integer.par-
   seInt(itu);
7.              o++;
8.  }
```

Kode 5.10 Kode Reader Skor.csv

Sedangkan kode 5.10 merupakan pengambilan dan memasukkan skor ke masing – masing *node* ke dalam arraylist yang sudah dibuat.

5.3.2 Pencarian Solusi Awal yang Layak

Setelah melakukan tahap pendefinisian kebutuhan variable dan membaca data data dari file .csv, maka tahap selanjutnya adalah menemukan solusi awal yang layak digunakan untuk di optimalkan dengan *Great Deluge Iterative Local Search*. Langkah pertama adaah mendefinisikan beberapa variable sebagai berikut.

```
1. int tMax = 120; //Menetapkan nilai tMax (da-
   lam menit)
2. int maxLoop = 10000; //Menetapkan batas it-
   erasi (m8x loop)
3. int count = 0; //Mengatur count = 0
4. int loop = 0; //Mengatur loop = 0
```

Kode 5.11 Deklarasi Variabel initial Solution

Kode 5.11mendeklarasikan bahwa waktu tempuh maksimum yang diperbolehkan hanya 120, dan iterasi akan berjalan sebanyak 1.000.000 kali, dan juga telah memenuhi Batasan dari model OP yang telah dibuat sebelumnya.

```
1. do {
2. int Rn = new Random().nextInt(jmlNode - 2) +
   1;
3. ArrayList<Integer> listRl = new ArrayList();
4. ArrayList<Integer> listRl_rand = new Ar-
   rayList();
5. for (int i = 1; i < jmlNode; i++) {
6. if (i == nAwal || i == nAkir) {
7. continue;
8. }
9. listRl_rand.add(i);
10.                 }
11.    Collections.shuffle(listRl_rand);
12.    listRl.add(nAwal);
13.    for (int i = 0; i < listRl_rand.size();
   i++) {
14. listRl.add(listRl_rand.get(i));
15.                 }
16. listRl.add(nAkir);
```

Kode 5.12 Kode List Rl

Pada kode 5.12 Array listRl digunakan untuk menyimpan seluruh *node* dalam bentuk arraylist. Lalu listRl1 merupakan tempat penyimpanan *node* yang nanti akan diacak, listRl1 tak mungkin berisi *node* awal dan akhir Karena yang akan di acak merupakan jalur antara kedua *node* tersebut.

```
1. ArrayList<Integer> subListRs = new Ar-
   rayList();
2. subListRs.add(listRl.get(0));
3. for (int i = 1; i < (Rn + 1); i++) {
4. subListRs.add(listRl.get(i));
5.                 }
6. subListRs.add(listRl.get(listRl.size() -
   1));
```

Kode 5.13 Kode subList Rs

Kode 5.13 merupakan pembuatan arraylist subListRs dimana subListRs menyimpan calon solusi yang layak yang nantinya akan di optimasi.

```
1.     public static int hitungWaktuTempuh(Ar-
   rayList<Integer> varListRl) { //Perhitungan
   waktu tempuh
2.         int sum = 0; //Inisiasi nilai awal
   sum adalah 0 untuk penjumlahan
3.         for (int i = 0; i <
   varListRl.size() - 1; i++) { //Loop untuk
   mengambil isi dari isi list Rl
4.             sum = sum + un-
   tukjalur[varListRl.get(i) -
   1][varListRl.get(i + 1) - 1]; //Menjumlah
   setiap waktu tempuh pada 1 list Rl
5.         }
6.         return sum;
7.     }
```

Kode 5.14 Kode Hitung Waktu Tempuh

Kode 5.14 merupakan metode untuk menghitung waktu temuh yang dilalui dari *node* awal ke *node* akhir dengan jalur di antara keduanya dipilih secara random, sehingga, dengan menggunakan kode di atas, dapat secara otomatis menghitung berapa waktu tempuh dengan berapapun *node* yang dilalui.

```
1.  for (int i = 0; i < subListRs.size(); i++) {
2.  if (hitungWaktuTempuh(subListRs) < tMax &&
    !feasSolution.contains(subListRs)) {
3.  feasSolution.add(subListRs);
4.                      }
5.                  }
6.                  loop++; //loop=loop+1
7.              } while (loop <= maxLoop);
```

Kode 5.15 Kode subList Rs

Kode 5.15 akan melakukan pengecekan apakah dari calon solusi yang dihasilkan sebelumnya telah memenuhi konstrain tmax dan memastikan tidak ada solusi yang sama persis dapat terpilih. Hasil seleksi dimasukkan kedalam feasSolution.

```
1.     public static int hitungScore(Ar-
    rayList<Integer> varListRl) { //Perhitungan
    waktu tempuh
2.          int sum = 0; //Inisiasi nilai awal
    sum adalah 0 untuk penjumlahan
3.          for (int i = 0; i <
    varListRl.size(); i++) { //Loop untuk
    mengambil isi dari isi list Rl
4.                  sum = sum + un-
    tukscore[varListRl.get(i) - 1]; //Menjumlah
    setiap waktu tempuh pada 1 list Rl
5.          }
6.          return sum;
7.      }
```

Kode 5.16 Kode Kalkulasi Skor

Jika kode 5.15 untuk mencari waktu tempuh, maka kode 5.16 adalah metode untuk perhitungan skor untuk mendampingi informasi selain dari berapa waktu tempuh dari *node* awal ke akhir dengan isi yang di acak ditambahkan pula dengan skornya secara dinamis.

```
1.          for (int i = 0; i < feasSolu-
   tion.size(); i++) {
2. //             System.out.println(feasSolu-
   tion.get(i) + "(" + hitungWak-
   tuTempuh(feasSolution.get(i)) + ", " + hi-
   tungScore(feasSolution.get(i)) + ")");
3.              feasSolution_Jarak.add(hi-
   tungWaktuTempuh(feasSolution.get(i)));
4.              feasSolution_Score.add(hitung-
   Score(feasSolution.get(i)));
5.          }
```

Kode 5.17 Kode Pemanggil waktu tempuh dan skor

Setelah membuat perhitungan dan metode perhitungan skor dan waktu, maka kedua metode itu dapat dipanggil dengan kode 5.17 dan hasil perhitungannya tersimpan di feasSolution. Lalu untuk melihat hasil dari jalannya kodingan yang pada sub bagian ini dapat dilihat dengan kode 5.18.

```
1. for (int i = 0; i < feasSolution.size();
   i++) {
2. System.out.println(feasSolution.get(i) + "("
   + feasSolution_Jarak.get(i) + ", " + feasSo-
   lution_Score.get(i) + ")");
3.          }
```

Kode 5.18 Kode yang Mencetak Hasi Solusi Awal

5.3.3 Seleksi *Roulette Wheel*

Hasil dari pembuatan solusi awal yang layak menghasilkan banyak sekali kemungkinan atau calon solusi yang layak. Pemilihan atau seleksi dari *roulette wheel* ini netral, Karena memiliki probabilitas yang berbeda – beda, Karena perhitungan *roullete wheel* adalah dengan menggunakan total skor secara keseluruhan dari *initial fasible solution*. Sehingga, semakin besar skor sebuah *node*, semakin besar pula terpilihnya *node* tersebut dalam seleksi *roulete wheel* ini. Tahap pertama adalah membuat metode yang dapat menghitung skor total dari *initial feasible solution* seperti pada kode 5.19 .

```
1.      public static int totalScore() {
2.          int sum = 0;
3.          for (int i = 0; i < feasSolu-
   tion_Score.size(); i++) {
4.              sum = sum + feasSolu-
   tion_Score.get(i);
5.          }
6.          return sum;
7.      }
```

Kode 5.19 Kode Perhitungan Total Skor

Lalu total skor yang dihasilkan dari kode 5.19 digunakan pada potongan kode 5.20.

```
1.      public static int rouletteWheelSelec-
   tion() { //Perhitungan waktu tempuh
2.          double r = new Random().nextDou-
   ble();
3.          int totScore = totalScore();
4.          int k = 0;
5.          double sum = (double) feasSolu-
   tion_Score.get(k) / totScore;
6.          while (sum < r) {
7.              k++;
8.              sum = sum + (double) feasSolu-
   tion_Score.get(k) / totScore;
9.          return k;
10.     }
```

Kode 5.20 Kode Metode Roulette Wheel

Seperti yang dapat dilihat pada kode 5.20 bahwa pertama dimulai dengan double yang di random, lalu di ambil index pertama yaitu index ke 0 apabila skor pada index ke 0 dibagi dengan total skor lebih rendah nilainya daripada double rand, maka solusi tersebut dilewati, jika dia melebihi, maka nilai sebelumnya akan ditambahkan dengan nilai saat ini.

```
1. int size = 30;
2. do {
3. pemilihanHasil.add(feasSolution.get(rou-
   letteWheelSelection()));
4. } while (pemilihanHasil.size() < size);
```

Kode 5.21 Kode inisiasi roulette wheel

Setelah mendefinisikan metode roullete wheel, maka saatnya mengimplementasikannya. Pada kode 5.21 batasan *size* digunakan untuk membatasi jumlah solusi yang diterima yang akan dilakukan optimasi nya.

```
1. for (int i = 0; i < pemilihanHasil.size();
   i++) {
2. pemilihanHasilWaktu.add(hitungWak-
   tuTempuh(pemilihanHasil.get(i)));
3. pemilihanHasilScore.add(hitungScore(pemili-
   hanHasil.get(i)));
4. System.out.println(pemilihanHasil.get(i) +
   "(" + pemilihanHasilWaktu.get(i) + ", " +
   pemilihanHasilScore.get(i) + ")");
5.          }
```

Kode 5.22 Kode looping hasil Roulette Wheel

Kode 5.22 berfungsi untuk melakukan looping untuk memasukkan hasil seleksi dari *roullete wheel* kedalam array pemilihanHasil dan mencetak hasilnya.

5.3.4 Iterative Local Search

Setelah melakukan seleksi dengan *roullete wheel*, maka optimasi dapat dilakukan, dimana solusi yang ada akan dilakukan metode *insert*, *delet* dan *swap*, dimana *node* antara *node* awal dan akhir akan di tambahkan, dikurangi dan ditukar posisinya untuk mendapatkan skor yang lebih baik dan atau waktu tempuh yang lebih singkat, lalu, algoritma *great deluge* tidak langsung mengabil semua yang terbaik, tetapi memiliki level toleransi, dimana level toleransi yang digunakan pada kasus kali ini dimulai dari 5%.

```
1.          for (int i = 0; i < pemili-
   hanHasil.size(); i++) {
2.              //looping untuk mengecek semua
   isi pemilihanHasil
3.              int counterGD = 1;
4.              int maxGDLoop = 1000000;
```

Kode 5.23 Kode inisiasi Iterative Local Search

Langkah pertama adalah mengambil nilai dari hasil seleksi dan menetapkan variable untuk looping. Disini kita menetapkan looping local searchnya sebanyak 1.000.000 seperti pada kode 5.23.

```
1. do {
2. int randCase = new Random().nextInt(3) +
   1;
3. switch (randCase) {
4. case 1:
5.  int swap1 = new Random().nextInt(pemili-
   hanHasil.get(i).size() - 2) + 1;
6. int swap2 = new Random().nextInt(pemili-
   hanHasil.get(i).size() - 2) + 1;
7.  Collections.swap(pemilihanHasil.get(i),
   swap1, swap2);
8.  int currentJarakSwap = hitungWak-
   tuTempuh(pemilihanHasil.get(i));
9. int currentScoreSwap = hitungScore(pemili-
   hanHasil.get(i));
10.   if (currentJarakSwap < pemilihanHasil-
   Waktu.get(i) ) {
11.     pemilihanHasilWaktu.set(i, currentJa-
   rakSwap);
12.   pemilihanHasilScore.set(i, cur-
   rentScoreSwap);
13.   counterGD = 1;
14.   } else {
15.   Collections.swap(pemilihanHasil.get(i),
   swap2, swap1);
16.                          }
17.                        break;
```

Kode 5.24 Inisialisasi Looping dan kode Swap

Seperti yang dapat dilihat pada kode 5.24, metode pertama adalah *swap*, dimana pemilihan nilai dari salah satu index dari sebuah jalur akan ditukar dengan nilai dari index lainnya yang

bukan *node* awal dan akhir, untuk melakukan index, pertama dilakukan angka random terlebih dahulu antara satu sampai tiga untuk menentukan antara *swap, insert* atau *delete* yang akan dipakai, dimana pada kasus ini *swap* adalah angka pertama, dan jika waktu tempuh berkurang atau skor yang dihasilkan paling tidak lebih baik 5% dari skor awal, maka proses *swap* akan dilakukan, jika tidak, proses *swap* tidak terjadi.

```
1. case 2:
2. int delRandom = new Random().nextInt(pem-
   ilihanHasil.get(i).size() - 2) + 1;
3. if (pemilihanHasil.get(i).size() > 3) {
4. int temp = pemili-
   hanHasil.get(i).get(delRandom);
5. pemilihanHasil.get(i).remove(delRandom);
6. int currentJarakDel = hitungWak-
   tuTempuh(pemilihanHasil.get(i));
7. int currentScoreDel = hitungScore(pemili-
   hanHasil.get(i));
8. if (currentJarakDel <= pemilihanHasil-
   Waktu.get(i) && currentScoreDel >= 1.05 *
   pemilihanHasilScore.get(i)) {
9. pemilihanHasilWaktu.set(i, currentJarak-
   Del);
10.    pemilihanHasilScore.set(i, cur-
   rentScoreDel);
11.    counterGD = 1;
12.    } else {
13.    pemilihanHasil.get(i).add(delRandom,
   temp);
14.    }
15.    break;
16.    }
```

Kode 5.25 Kode Local Search Delete

Kode 5.25 adalah kode untuk algoritma delete, ditandai dengan apabilan pembuatan angka random sama dengan angka dua.

Sebelum mendelete, dipastikan terlebih dahulu bahwa paling tidak tedapat satu *node* diantara *node* awal dan *node* akhir. Semua solusi akan di coba untuk di *delete* terlebih dahulu, apabila kriteria waktu tempuh menjadi lebih pendek atau apabila paling tidak skor yang dihasilkan lebih baik dari 5% dari solusi awal, maka hasil *delete* akan disimpan, jika tidak, maka *node* yang dihapus akan dikembalikan ke posisi semula.

```
1. case 3:
2. int rndInsert = new Random().nextInt(pem-
   ilihanHasil.get(i).size() - 2) + 1;
3. int rndNilaiInsert = new Random().nex-
   tInt(jmlNode) + 1;
4. if (rndNilaiInsert == nAwal || rndNilai-
   Insert == nAkir || pemili-
   hanHasil.get(i).contains(rndInsert)) {
5. continue;
6.     }
7. pemilihanHasil.get(i).add(rndInsert,
   rndNilaiInsert);
8. int currentWaktuTempuhInsert = hitungWak-
   tuTempuh(pemilihanHasil.get(i));
9. int currentScoreInsert = hitungScore(pem-
   ilihanHasil.get(i));
10.    if (currentWaktuTempuhInsert <= pemili-
   hanHasilWaktu.get(i) && currentScoreInsert
   >= 1.05 * pemilihanHasilScore.get(i)) {
11.    pemilihanHasilWaktu.set(i, currentWak-
   tuTempuhInsert);
12.    pemilihanHasilScore.set(i, currentScore-
   Insert);
13.    counterGD = 1;
14.    } else {
15.    pemilihanHasil.get(i).remove(rndInsert);
16.                              }
17.                              break;
```

```
18.                 }
19.                 counterGD++;
20.
21.         } while (counterGD <= maxGDLoop);
22.
23.         }
```

Kode 5.26 Kode Local Search Insert

Kode 5.26 adalah potongan kode untuk *insert*,dimana akan terpilih apabilan pemilihan angka random tiga, sebelum melakukan *insert*, dipastikan terlebih dahulu bahwa angka random yang dihasilkan tidak mengandung *node* awal dan akhir, jika sudah, maka diambil index secara random, dan pada posisi tersebut lah penambahan *node* akan disispkan dan apabila melewati kriteria mengurangi waktu tempuh atau paling tidak skor lebih baik 5% daripada solusi awal, maka *insert* akan berhasil.

5.3.5 Pengambilan Solusi Terbaik

Setelah melakukan *local search* dari 30 solusi awal yang telah di seleksi, maka dari ke 30 solusi tersebut sudah ditingkatkan hasilnya dan dapat diambil solusi terbaik, dimana kriteria solusi terbaik adalah dengan skor yang paling banyak, berikut merupakan kode untuk pengambilan solusi terbaik.

```
1.         int bestIndex = 0;
2.          int bestScore = pemili-
   hanHasilScore.get(0);
3.          for (int i = 1; i < pemili-
   hanHasil.size(); i++) {
4.              if (pemilihanHasilScore.get(i)
   >= bestScore) {
5.                  bestIndex = i;
6.                  bestScore = pemili-
   hanHasilScore.get(i);
7.              }
8.          }
9.          System.out.println();
10.            System.out.println("Solusi ter-
   baik :" + pemilihanHasil.get(bestIndex) +
   "(" + pemilihanHasilWaktu.get(bestIndex) +
   "," + pemilihanHasilScore.get(bestIndex) +
   ")");
11.     }
```

Kode 5.27 Kode Pencarian Solusi Terbaik

Kode 5.27 dapat mengecek dan membandingkan mana skor yang paling tinggi dan akan diambil sebagai solusi terbaik. Serta mencetak hasil pencarian, yaitu jalur, beserta skor dan waktu tempuhnya.

5.4 Uji Coba

Pada bagian ini akan membahas kodingan yang ditambahkan untuk melakukan uji coba terhadap pencarian solusi dengan cara melakukan beragai percobaan untuk mendapatkan hasil yang terbaik, lalu hasil – hasil tersebut akan dibandingkan dan ditarik kesimpulannya.

5.4.1 Penggunaan *Reinforcement Learning*

Uji coba dengan menggunakan *Reinforement Learning* ini mengubah pemilihan *local* search yang tadinya bersifat random, menjadi bergantung dengan skor, artinya adalah setiap pilihan akan digunakan terus menerus selama hasil yang dihasilkan lebih baik dari hasil sebelumnya, akan tetapi jika hasilnya sama atau lebih buruk maka skornya akan dikurangi, sehingga probabilitas terpilihnya akan berkurang. Pertama adalah dengan membuat array dengan size 3 seperti pada kode 5.28.

```
1.      static int[] reinforce = new int[3];
```

Kode 5.28 deklarasi variabel reinforcement

Berikutnya adalah pembuatan metode reinfocrementLearning untuk menghitung skor pilhan *local search*yang terpilih.

```
1.      public static int reinforcementLearn-
   ing() {
2.          int max = reinforce[0];
3.          int id = 0;
4.          if (reinforce[1] > max) {
5.              id = 1;
6.              max = reinforce[1];
7.          } else if (reinforce[2] > max) {
8.              id = 2;
9.              max = reinforce[2];
10.            }
11.            if (reinforce[0] == reinforce[1]
   && reinforce[1] == reinforce[2]) {
12.                id = new Random().nex-
   tInt(2);
13.            }
14.
15.            return id;
16.     }
```

Kode 5.29 Metode Reinforcement Learning

Kode 5.29 menjelaskan bahwa yang akan terpilih untuk dilakukan *local search* berikutnya adalah yang memiliki nilai lebih dari max, dimana max secara default merupakan index ke 0, namun apabila index pertama dan kedua lebih bagus dari index ke 0, maka akan digantikan pemilihan *local search*nya, dan jika pada kasus nilai nya sama, maka dipilih secara random.

```
1. //int randCase = new Random().nextInt(3) +
   1;
2. int randCase = reinforcementLearning();
```

Kode 5.30 Penggantian Cara Random

Tahapan berikutnya adalah mengganti randCase yang berfungsi untuk merandom angka yang digunakan untuk *local search* seperti pada kode 5.30.

```
1. reinforce[0]++;
2.     } else {
3. reinforce[0]--;
```

Kode 5.31 Pemasukan metode reinforce

Dan tahapan terkahir adalah membuat indeks masing – masing kasus bertambah setiap berhasil atau dikurangi bila gagal seperti pada kode 5.31.

5.4.2 Penggunaan *Decay Rate*

Decay Rate berfungsi untuk secara dinamis mengubah level toleransi yang sebelumnya ditentukan secara statis, dimana kriteria untuk *local search* bukan lagi pada skor, namun menggunakan batas bawah, dimana batas bawah tersebut merupakan skor terbaik dari *initial feasible solution* yang telah di seleksi menggunakan *roulette wheel selection*. Untuk

memulai *Decay Rate*, pertama harus menentukan beberapa variable seperti pada kode 5.32.

```
1.      static double decayrate = 1;
2.      static int bestBatasBawah = 0;
```

Kode 5.32 Deklarasi Variabel DecayRate

Setelah mendeklarasikan variable seperti pada kode 5.32, setelah melakukan *roulette wheel selection*, ditambahkan sepotong kode seperti kode 5.33.

```
1. if (pemilihanHasilScore.get(i) > bestBatas-
   Bawah) {
2. bestBatasBawah = pemili-
   hanHasilScore.get(i);
3.              }
4.          }
5.          System.out.println("Solusi terbaik
   :" + bestBatasBawah);
```

Kode 5.33 Penambahan Batas Bawah

Kodingan di atas adalah untuk menentukan batas bawah dengan nilai skor *roulette wheel selection* terbaik. Setelah itu, semua Batasan dalam *local search* diganti menjadi seperti kode 5.34.

```
1. if (currentJarakDel <= pemilihanHasil-
   Waktu.get(i) && currentScoreDel >= best-
   BatasBawah)
```

Kode 5.34 Kode Jika Maka

(Halaman ini sengaja dikosongkan)

BAB VI
HASIL DAN PEMBAHASAN

Bab ini berisikan hasil dan pembahasan setelah melakukan implementasi. Hasil yang akan dijelaskan adalah hasil uji coba model, validasi model, dan hasil peramalan untuk periode yang akan datang.

6.1 Lingkungan Uji Coba

Lingkungan uji coba merupakan kriteria perangkat pengujian yang digunakan dalam implementasi *Great Deluge Iterative Local Search* pada model *Orienteering Problem* untuk menyelesaikan permasalahan pada tugas akhir ini. Lingkungan uji coba mencakup perangkat keras dan juga perangkat lunak yang digunakan. Spesifikasi dari perangkat keras yang digunakan dalam implementasi metode ditunjukkan pada tabel di bawah.

PERANGKAT KERAS	SPESIFIKASI
Jenis	Laptop
Processor	Intel® Core™ i3 3127U
RAM	4096MB
Hard Disk Drive	500GB

Ada pula lingkungan pernagkat lunak yang digunakan dalam pengerjaan dan implementasi penelitian tugas akhir ini sebagai berikut.

PERANGKAT LUNAK	FUNGSI
Windows 8.1	Sistem Operasi
NetBeans IDE 8.2	Membuat model matematis
	Melakukan optimasi
Microsoft Office Excel 2016	Mengolah data

6.2 Penggambaran *Network Model*

Network Model dapat digunakan untuk mempermudah penggambaran permasalahan penelitian tugas akhir ini yaitu pembuatan model *Orienteering Problem.* Dalam *Network Model* tesebut, *node* melambangkan pemberhentian angkot, lalu anak panah melambangkan jalur angkot yang digunakan beserta arahnya kemana. Gambar *network model* secara keseluruhan dapat dilihat pada LAMPIRAN G.

Jarak beserta waktu dan trayek apa saja yang melewati *node* tersebut dapat dilihat pada tabel 6.1, dan tabel secara lengkapnya dapat dilihat pada LAMPIRAN C1.

Tabel 6.1 Potongan Tabel Jarak dan Waktu

Node Awal	Node Akhir	Jarak (Meter)	Waktu (Menit)	Trayek
1	2	550	1	UBB
2	3	600	2	UBB
2	1	550	1	UBB
2	16	700	2	UBB
3	4	400	1	UBB
4	5	1000	2	UBB
5	6	400	1	UBB
6	7	400	1	UBB
7	8	350	1	UBB
8	9	700	2	UBB
9	10	450	1	UBB
10	11	500	1	UBB

Seperti yang bisa dilihat pada tabel 6.1, masing – masing *node* dapat dikunjungi lebih dari satu buah trayek, oleh Karena itu,

masing – masing *node* memiliki *score* yaitu jumlah angkot yang lewat pada *node* tersebut. Tabel 6.2 merupakan *score* pada masing – masing *node* dan secara lengkapnya dapat dilihat pada LAMPIRAN C2

.

Tabel 6.2 Potongan Skor tiap Node

Node	Skor
1	32
2	165
3	32
4	32
5	32

Node	Skor
61	32
62	32
63	104
64	104
65	32

Node	Skor
121	124
122	124
123	124
124	124
125	124

6.3 Hasil Algoritma Diijkstra

Apabila rancangan algoritma Dijkstra sudah benar dan tanpa error, serta memastikan bahwa semua node inputan benar – benar terhubung, Karena apabila ada yang tidak terhubung maka jaraknya akan bernilai nilai maksimum integer, program akan membuat matriks seperti potongan tabel di bawah.

Tabel 6.3 Hasil Algoritma Dijkstra

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	3	4	6	7	8	9	10	11
2	1	0	2	3	5	6	7	8	9	10
3	16	15	0	1	3	4	5	6	7	8
4	15	14	16	0	2	3	4	5	6	7
5	13	12	14	15	0	1	2	3	4	5

6	12	11	13	14	16	0	1	2	3	4
7	11	10	12	13	15	16	0	1	2	3
8	11	10	12	13	15	16	17	0	2	3
9	9	8	10	11	13	14	15	16	0	1
10	8	7	9	10	12	13	14	15	16	0

Dapat dilihat dari potongan tabel 6.3, bahwa semua jalur sudah didapati berapa jaraknya terkecuali dengan *node* awal dan akhir yang sama.

6.4 Hasil Pencarian Solusi Dengan *Great Deluge iterative Local Search*

Hasil *running* program dapat dibagi menjadi tiga, yaitu hasil pencarian *initial feasible solution,* hasil *roulette wheel selection* dan hasil *great deluge iterative local search* dan pengambil solusi terbaik. Percobaan dilakukan dengan menggunakan tMax 120, *node* awal 1 dan *node*akhir 125, yaitu dari pangkalan ujung baru sampai pangkalan wonorejo.

6.4.1. Hasil Pencarian Solusi Layak Awal

Hasil pencarian solusi awal yang layak ini dilakukan sebanyak 1.000.000 iterasi dan dengan tMax 120, *node* awal 1 dan *node*akhir 125. Gambar 6.2 merupakan potongan hasil pencarian solusi awal tersebut.

1. [1, 156, 125](78, 289)
2. [1, 64, 125](118, 260)
3. [1, 121, 125](74, 280)
4. [1, 90, 13, 16, 125](105, 385)
5. [1, 73, 125](111, 260)

---Hasil---

total jumlah skor : 725197

jumlah jalur yang feasible : 1835

Gambar 6.1 Hasil Pencarian Solusi Awal

Gambar 6.2 merupakan hasil calon solusi yang dapat dioptimalkan, terdapat 1.835 jalur yang feasible dengan total skor sebanyak 725.197 yang nantinya hanya akan di ambil 30 jalur dari 1.835 jalur yang ada.

6.4.2. Hasil Seleksi *Roullete Wheel*

Setelah melakukan pencarian calon solusi pada tahapan sebelumnya, tahapan berikutnya adalah memilih 30 calon solusi yang akan di optimasikan dengan menggunakan roulette wheel, dimana kemungkinan terpilihnya jalur dengan skor yang besar akan lebih tinggi disbanding dengan jalur dengan skor yang lebih rendah. Gambar 6.3 merupakan ke 30 solusi yang akan di optimasi yang terpilih oleh roulette wheel.

1. [1, 30, 125](80, 188)
2. [1, 124, 112, 131, 125](115, 528)
3. [1, 9, 10, 107, 125](93, 344)
4. [1, 20, 156, 125](78, 321)
5. [1, 39, 60, 120, 125](102, 477)
6. [1, 132, 135, 125](91, 413)
7. [1, 138, 50, 125](84, 321)
8. [1, 50, 101, 125](88, 274)
9. [1, 118, 109, 125](100, 404)
10. [1, 135, 81, 125](111, 454)
11. [1, 156, 81, 50, 125](117, 486)
12. [1, 91, 119, 125](80, 312)
13. [1, 147, 39, 27, 100, 125](107, 705)
14. [1, 148, 137, 125](81, 422)
15. [1, 2, 134, 125](74, 454)
16. [1, 22, 82, 125](78, 353)
17. [1, 7, 93, 125](101, 531)
18. [1, 142, 20, 110, 125](116, 445)
19. [1, 127, 125](81, 280)
20. [1, 2, 144, 125](81, 454)
21. [1, 153, 84, 125](83, 454)
22. [1, 152, 46, 155, 125](87, 454)
23. [1, 153, 83, 20, 133, 125](111, 610)
24. [1, 143, 140, 125](89, 422)
25. [1, 143, 93, 125](84, 632)
26. [1, 44, 98, 125](79, 398)
27. [1, 142, 153, 125](104, 422)
28. [1, 18, 154, 125](74, 321)
29. [1, 81, 101, 110, 125](85, 531)
30. [1, 59, 125](100, 188)

Gambar 6.2 Hasil Roulette Wheel Selection

Potongan Hasil running tersebut mengambil hanya 30 saja dari 1.835 jalur yang ada dengan menggunakan *roulette wheel selection*. Hasil ini harus di validasi terlebih dahulu sebelum dapat dilakukan optimasi, untuk mencegah adanya solusi yang melanggar Batasan model.

6.4.2.1. Validasi Algoritma

Validasi algoritma dilakukan secara manual, yaitu dengan mengecek ke 30 solusi yang terpilih apakah melanggar Batasan apa tidak. Berikut merupakan hasil validasi:

a. Batasan pertama: Berdasarkan gambar 6.3, seluruh solusi dimulai dari *node* 1 dan diakhiri dengan *node* 125.
b. Batasan kedua: Berdasarkan gambar 6.2, dapat disimpulkan dengan banyaknya jumlah feasible solution, dapat dipastikan bahwa semua *node* terhubung dan berdasarkan gambar 6.3, dapat dilihat bahwa semua *node* tidak ada yang dilewati lebih dari sekali.
c. Batasan ketiga: Berdasarkan gambar 6.3, dapat disimpulkan bahwa semua solusi yang akan di optimasikan tidak melebihi tmax yaitu 120.
d. Batasan keempat : Berdasarkan gambar 6.3, dapat disimpulkan bahwa tidak ada subtour pada solusinya, sehingga tidak ada yang melanggar Batasan ini.

6.4.3. Hasil *Great Deluge Iterative Local Search* dan Pengambilan Solusi Terbaik

Setelah seleksi seperti pada gambar 6.3, berikutnya dilakukan optimasi terhadap semua solusi dengan melakukan metode *insert*, *swap* dan *delete*, dimana proses itu akan di iterasikan sebanyak satu juta kali dengan harapan semakin banyak proses iterasi tersebut maka makin tinggi pula skor yang didapatkan.

Tabel 6.4 Great Deluge Iterative Local Search

Solusi ke-	Rute	Waktu	Skor
1	[1, 2, 16, 17, 18, 19, 22, 23, 25, 28, 29, 30, 125]	80	1306

Solusi ke-	Rute	Waktu	Skor
2	[1, 16, 22, 25, 28, 34, 36, 40, 41, 42, 138, 135, 134, 96, 97, 98, 113, 116, 130, 131, 112, 115, 118, 124, 125]	88	3928
3	[1, 2, 7, 9, 10, 16, 22, 28, 152, 153, 34, 36, 143, 39, 41, 139, 138, 135, 96, 97, 104, 105, 107, 125]	93	3647
4	[1, 2, 16, 17, 20, 22, 23, 25, 28, 152, 34, 36, 143, 39, 40, 41, 42, 154, 155, 138, 52, 53, 156, 125]	78	3328
5	[1, 22, 28, 153, 36, 143, 39, 40, 154, 155, 138, 52, 60, 53, 156, 96, 97, 98, 109, 110, 115, 117, 118, 120, 125]	102	3781
6	[1, 2, 16, 18, 22, 23, 25, 28, 34, 36, 143, 39, 40, 41, 139, 137, 135, 134, 96, 97, 98, 132, 125]	74	3688
7	[1, 2, 16, 22, 23, 25, 28, 152, 153, 34, 36, 143, 39, 40, 41, 139, 138, 137, 52, 50, 125]	84	2966
8	[1, 16, 18, 22, 23, 25, 28, 34, 35, 36, 143, 39, 40, 42, 50, 52, 53, 136, 96, 97, 98, 100, 101, 125]	88	3566

Solusi ke-	Rute	Waktu	Skor
9	[1, 2, 16, 25, 28, 34, 40, 41, 42, 138, 96, 97, 98, 108, 109, 110, 113, 114, 115, 118, 125]	74	3373
10	[1, 2, 16, 17, 19, 20, 22, 23, 25, 28, 152, 34, 36, 143, 41, 81, 39, 40, 42, 154, 155, 138, 136, 135, 125]	79	3328
11	[1, 2, 16, 22, 23, 25, 28, 152, 153, 34, 36, 143, 41, 81, 39, 40, 42, 50, 52, 53, 156, 125](87, 3195)	87	3195
12	[1, 22, 91, 23, 28, 34, 36, 143, 41, 139, 138, 136, 134, 96, 97, 98, 108, 109, 110, 112, 115, 117, 119, 125]	80	3607
13	[1, 2, 22, 25, 147, 148, 23, 27, 28, 36, 39, 41, 42, 154, 155, 138, 135, 134, 96, 97, 98, 100, 125]	87	3719
14	[1, 2, 16, 17, 19, 22, 23, 25, 147, 148, 28, 152, 34, 36, 39, 40, 41, 42, 155, 137, 125]	81	28665
15	[1, 2, 16, 19, 22, 23, 25, 28, 152, 153, 34, 35, 36, 143, 39, 40, 41, 139, 137, 136, 134, 125]	74	2966
16	[1, 2, 16, 17, 18, 22, 23, 25, 26, 28, 152, 153, 34, 35, 82, 125]	78	1737

Solusi ke-	Rute	Waktu	Skor
17	[1, 7, 2, 20, 22, 23, 25, 28, 152, 34, 36, 143, 39, 40, 139, 135, 96, 97, 98, 133, 93, 125]	101	3600
18	[1, 2, 16, 20, 22, 23, 28, 152, 34, 39, 41, 142, 154, 137, 96, 97, 98, 132, 108, 110, 125]	80	3308
19	[1, 2, 16, 23, 25, 28, 34, 36, 143, 39, 40, 41, 42, 155, 135, 96, 97, 98, 111, 113, 114, 118, 126, 127, 125]	81	4010
20	[1, 2, 16, 22, 23, 24, 25, 28, 152, 153, 34, 36, 143, 144, 125]	81	2040
21	[1, 2, 16, 22, 23, 25, 26, 28, 152, 153, 34, 84, 125]	83	1774
22	[1, 2, 16, 22, 23, 25, 28, 152, 153, 34, 36, 143, 39, 40, 41, 42, 46, 154, 155, 125]	87	2833
23	[1, 2, 16, 20, 22, 23, 24, 25, 152, 153, 34, 83, 36, 39, 40, 41, 42, 155, 96, 97, 98, 133, 125]	86	3619
24	[1, 2, 16, 20, 22, 23, 25, 28, 152, 153, 34, 36, 143, 39, 40, 41, 42, 142, 141, 140, 125]	89	2966
25	[1, 2, 16, 22, 23, 34, 36, 143, 39, 40, 41, 42, 154,	84	3701

Solusi ke-	Rute	Waktu	Skor
	138, 137, 96, 97, 98, 133, 93, 125]		
26	[1, 2, 19, 22, 23, 25, 28, 152, 153, 34, 143, 39, 40, 41, 44, 42, 137, 136, 96, 97, 98, 125]	79	3431
27	[1, 2, 16, 22, 23, 24, 25, 28, 152, 153, 34, 36, 143, 39, 40, 41, 42, 142, 125]	80	2700
28	[1, 2, 16, 18, 22, 23, 24, 25, 28, 152, 153, 34, 36, 143, 39, 40, 41, 42, 154, 125]	74	2732
29	[1, 2, 28, 34, 36, 143, 39, 40, 81, 41, 42, 155, 134, 96, 97, 98, 101, 99, 108, 110, 125]	85	3356
30	[1, 2, 16, 21, 22, 23, 25, 28, 153, 34, 35, 36, 143, 39, 40, 41, 42, 155, 138, 137, 52, 53, 56, 59, 125]	100	3259

Tabel 6.4 merupakan hasil eksekusi *iterative local search* dan dari tabel tersebut dapat kita simpulkan bahwa seluruh solusi ditingkatkan skornya, meskipun hanya sebagian yang dikurangi waktu perjalanannya, lalu solusi terbaik berdasarkan tabel 6.3 adalah solusi ke 19 yaitu dengan rute [1, 2, 16, 23, 25, 28, 34, 36, 143, 39, 40, 41, 42, 155, 135, 96, 97, 98, 111, 113, 114, 118, 126, 127, 125] dengan waktu tempuh 81 menit dan skor sebanyak 4010.

6.5 Hasil Uji Coba menggunakan *Reinforcement Learning*

Dengan menggunakan reinforcement learning sebagai perandom angka satu sampai tiga untuk melakukan metode *insert*, *swap* dan delete, dapat diperoleh hasil seperti pada gambar 6.4.

Solusi ke = 25

[1, 7, 96, 125](91, 531)

Solusi ke = 26

[1, 91, 28, 35, 125](80, 385)

Solusi ke = 27

[1, 88, 36, 108, 125](90, 477)

Solusi ke = 28

[1, 88, 127, 125](97, 312)

Solusi ke = 29

[1, 2, 147, 22, 125](81, 619)

Solusi ke = 30

[1, 86, 89, 125](91, 220)

Solusi terbaik :[1, 18, 149, 87, 97, 125](99,696)

Gambar 6.3 Hasil Reinforcement Learning

Tahapan nya sama dengan sebelumnya hanya perbedaannya adalah penentuan kapan memakai *insert*, *delete* dan *swap*. Setelah di jalnkan, solusi yang paling baik adalah [1, 18, 149, 87, 97, 125] dengan waktu tempuh 99 menit dan skor 696,

dimana tidak terjadi penambahan skor dikarenakan terjebak dalam pilihan yang sama secara terus menerus.

6.6 Hasil Uji Coba menggunakan *Decay Rate*

Penggunaan decay rate kurang lebih sama eperti pada *Great Deluge* pada sub bab 6.5, perbedaannya disini melakukan uji coba dengan *decay rate* yang berbeda beda dan diperoleh hasilnya. Hasil dari perbandingan solusi terbaik antara *decay rate* dapat dilihat pada tabel 6.5.

Tabel 6.5 Tabel Hasil Decay Rate

Decay Rate	Rute	Waktu	Skor
1	[1, 16, 151, 22, 23, 40, 125]	87	949
3	[1, 150, 153, 94, 125]	95	765
5	[1, 27, 97, 115, 125]	76	788

Uji coba yang ketiga ini mengganti level toleransi yang tadinya 5% menjadi batas bawah, dimana batas bawah adalah skor terbaik dari *roulette wheel* ditambahkan dengan *decay rate* nya yang pada studi kasus ini adalah 1, 3 dan 5. Solusi terbaiknya adalah [1, 16, 151, 22, 23, 40, 125] dengan waktu tempuh 87 menit dan skor 949 dengan decay rate 1. Namun, ketiga decay rate tersebut tidak mengalami peningkatan dari solusi awal mereka, dengan kata lain tidak mengalami perubahan.

6.7 Analisis Hasil Pengambilan Solusi Terbaik dan Uji Coba

Setelah melakukan *running* program nya dan mendapatkan hasil dari masing – masing, diperoleh hasil yang berbeda – beda. Berikut merupakan rekapan hasil pada tabel 6.6.

Tabel 6.6 Hasil Perbandingan

	Random	Reinforcement learning	Decay rate
Panjang jalur	25	6	7
Waktu tempuh	81	99	87
Skor	4010	696	949

Berdasarkan tabel 6.6, dapat disimpulkan bahwa penggunaan pencarian terbaik dihasilkan oleh random dengan panjang rute sebanyak 25, dengan waktu tempuh terpendek 81 menit dan skor tertinggi yaitu 4010, mengalahkan keduanya dengan sangat jauh dikarenakan kedua pencarian tersebut tidak ditemukan peningkatan dari solusi awal keduanya.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan dari semua proses yang telah dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik.

7.1 Kesimpulan

1 Pada penelitian ini dapat disimpulkan bahwa model *Orienteering Problem* dapat digunakan untuk memodelkan Jalur trayek angkot untuk kota Surabaya.
2 Algoritma Dijkstra dapat membantu pencarian waktu terpendek antara node dan pencarian solusi awal yang layak.
3 Pemilihan Solusi awal dengan roulette wheel sangat berpengaruh terhadap solusi yang akan dioptimasikan. Dibuktikan dengan perbandingan antara ketiga cara yang digunakan.
4 Pemilihan *insert, swap* dan *delete* sangat berpengaruh seperti yang dilihat pada metode random, reinforcement learning dan decay rate dimana apabilan tersangkut pada insert atau swap dan atau delete saja akan berpengaruh signifikan terhadap hasil akhir.
5 Pada penelitian ini, penggunaan algoritma *great deluge* sudah dapat dibuktikan mampu membuat skor yang lebih optimal dari solusi awal yang ditemukan.
6 Penggunaan *random* memberikan kontribusi terbesar sebagai pemilik skor tertinggi daripada penggunaan metode lain.

7.2. Saran

1 Pada penelitian ini, optimasi hanya berada mencakup 6 trayek yaitu trayek M, S, TWM, U, UBB, dan WB. Pada

penelitian berikutnya, dapat ditambahkan jalur trayek yang lain untuk memperluas cakupan optimasi angkot di Kota Surabaya.

2 Penelitian ini menggunakan algoritma random untuk mencari solusi layak awal. Sangat disarankan untuk menggunakan algoritma yang lebih baik seperti Particle Swarm Optimization, Memetic Algorithm, dan algoritma opimasi lainnya yang dapat digunakan untuk pencarian solusi sehingga solusi yang dihasilkan dapat lebih optimal.

3 Skor model Orienteering Problem pada penelitian ini menggunakan jumlah trayek yang lewat pada satu titik. Skor dapat diubah menjadi seberapa sering titik tersebut dilewati, atau seberapa populer titik tersebut di Kota Surabaya.

DAFTAR PUSTAKA

[1] A. Petroff, “The 15 worst cities for rush hour traffic,” CNN, 20 2 2017. [Online]. Available: http://money.cnn.com/2017/02/20/autos/traffic-rush-hour-cities/.

[2] Y. A. Gundayaini, “SOLUSI PERMASALAHAN DI INDONESIA DI BIDANG TRANSPORTASI”.

[3] A. Matteo, “Top Ten Cities With the World's Worst Traffic,” Voanews, 3 3 2015. [Online]. Available: http://learningenglish.voanews.com/a/top-ten-cities-for-worst-traffic-jakarta-istabul-gridlock-stress-public-transportation/2665040.html.

[4] Zainuddin, “Kemacetan Surabaya Masuk Empat Besar Dunia,” 6 2 2015. [Online]. Available: http://surabaya.tribunnews.com/2015/02/06/kemacetan-surabaya-masuk-empat-besar-dunia.

[5] S. Sofiana, “Penderita TB di Surabaya Tertinggi di Jatim, Kedua Jember, dan Ketiga Sidoarjo, Ini Data Lengkapnya,” 23 Maret 2016. [Online]. Available: http://surabaya.tribunnews.com/2016/03/23/penderita-tb-di-surabaya-tertinggi-di-jatim-kedua-jember-dan-ketiga-sidoarjo-ini-data-lengkapnya. [Diakses 19 September 2016].

[6] H. Lourenço, “ Job-Shop Scheduling: computational study of local search and large-step optimization methods,” *European Journal of Operational Research.*, vol. 83, no. 2, p. 347–364, 1995.

[7] H. Lourenço dan Z. M., “Iterated Local Search,” *"Combining the large-step optimization with tabu-search: application to the job-shop scheduling problem",* pp. 219-236, 1996.

[8] A. Juan, H. Lourenço, M. Mateo, R. Luo dan Q. Castella, "Using Iterated Local Search for solving the Flow-Shop Problem: parametrization, randomization and parallelization issues," *International Transactions in Operational Research*, 2013.

[9] P. H. Penna, L. Satori Ochi dan A. Subramanian, "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem," *Journal of Heuristics,* vol. 19, no. 2, pp. 201-232, 2013.

[10] I. C. Society, The Nature of Mathematical Programming, Mathematical Programming Glossary.

[11] L. L. a. R. V. Bruce L. Golden, The Orienteering Problem, Maryland: College of Business and Management, University of Maryland.

[12] I. G. B. W. E. Chao, "Theory and methodology – a fast and effective," *European Journal of Operation Research,* vol. 88, p. 14, 1996.

[13] W. S. D. V. O. Pieter Vansteenwegen, "The orienteering problem: A survey," *European Journal of Operational Research,* p. 10, 2010.

[14] H. C. L. P. V. Aldy Gunawan, "Orienteering Problem: A Survey of Recent Variants, Solution," *European Journal of Operational Research,* p. 54, 2015.

[15] C. T. A. Z. R. Miller, "Integer programming formulations and," *Journal of the ACM,* vol. 7, pp. 326 - 329, 1960.

[16] E. W. Djikstra, A note on two problems in connexion with graphs, Numerische Mathematik, 1959.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest dan C. Stein, "Section 24.3: Dijkstra's algorithm," dalam *Introduction to Algorithms*, MIT Press and McGraw–Hill, 2001, p. 595–601.

[18] A. Felner, " Position Paper: Dijkstra's Algorithm versus Uniform Cost Search or a Case Against Dijkstra's Algorithm," 2011.

[19] F. B. Zhan dan C. E. Noon, "Shortest Path Algorithms: An Evaluation Using Real Road Networks," *Transportation Science,* vol. 32, no. 1, p. 65–73, 1998.

[20] M. G. ·. J.-Y. Potvin, Handbook of Metaheuristics, Montreal: Springer Science+Business Media, 2010.

[21] T. St¨utzle, "Local Search Algorithms for Combinatorial Problems: Analysis, Improvements,," dalam *Dissertations in Artificial Intelligence*, Amsterdam, 1999.

[22] P. McMullan, "An Extended Implementation of the Great Deluge," dalam *Lecture Notes in Computer Science book series* , Berlin, 2007.

[23] GunterDueck, "New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel," *Journal of Computational Physics,* vol. 104, no. 1, pp. 86-92, 1993.

[24] M. F. Tasgetiren, "A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem," *Journal of Economic and Social Research* , vol. 4, no. 2, pp. 1-26, 2002.

[25] S. L. Shanthi Muthuswamy, "DISCRETE PARTICLE SWARM OPTIMIZATION FOR THE ORIENTEERING PROBLEM," *International*

Journal of Industrial Engineering, vol. 18, no. 2, pp. 92 - 102, 2011.

[26] R. S. a. K. M. K. Rameshkumar, "Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan," India, 2005.

[27] J. E. R. C. Kennedy, "Particle swarm optimization.," dalam *Proceedings of IEEE International Conference on Neural Networks*, 1942-1948.

[28] J. E. R. C. Kennedy, Swarm intelligence, Morgan Kaufmann, San Mateo, 2001.

[29] Y. E. R. Shi, "Empirical study of particle swarm optimiz," dalam *Proceedings of Congress of Evolutionary Computation*, 1999.

[30] Q.-K. T. M. F. &. L. Y.-C. Pan, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem.," *Computers & Operations Research* , vol. 35, pp. 2807 - 2839, 2008.

[31] J. E. R. C. Kennedy, "A discrete binary version of the particel swarm algorithm," dalam *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, 1997.

[32] C. T. P. L. C-J. Liao, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers & Operations Research,* vol. 34, pp. 3099 - 3111, 2007.

[33] B. C. M. S. P. &. R. A. Jarboui, "Combinatorial particle swarm optimization (CPSO) for partitional clustering problem.," *Applied Mathematics and Computation,* vol. 192, pp. 337 - 345, 2007.

[34] B. D. N. S. P. &. R. A. Jarboui, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation,* vol. 195, pp. 299 - 308, 2008.

[35] B. I. S. S. P. &. R. A. Jarboui, "A combinatorial particle swarm optimization for solving permutation flowshop problems," *Computers & Industrial Engineering,* vol. 54, pp. 526 - 538, 2008.

[36] A. J. B. &. G. X. Lian, "A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan," *Applied Mathematics and Computation,* vol. 183, pp. 1008 - 1017, 2006.

[37] Z. G. X. &. J. B. Lian, " A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos Solitons & Fractal,* vol. 35, pp. 851 - 861, 2008.

[38] A. Y. G. &. W. Z. Chen, "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem," *Journal of Zhejiang University Science A,* vol. 7, no. 4, pp. 607 - 614, 2006.

[39] M. F. L. Y.-C. S. M. &. G. G. Tasgetiren, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research,* vol. 177, pp. 930 - 947, 2007.

[40] W. W. K.-P. Z. C.-G. &. D. L.-J. Pang, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem," dalam *Proceedings of*

the Fourth International Conference on Computer and Information Technology, 2004.

[41] Z. S. F. E. Sevkli, "Variable neighborhood search for the orienteering problem," dalam *Proceedings of International Symposium on Computer and Information Sciences*, Istanbul, Turkey, 2006.

[42] Z. S. F. E. K. O. Sevkli, "Particle swarm optimization for the orienteering problem," dalam *International Symposium on Innovations in Intelligent Systems and Applications*, Istanbul, Turkey, 2007.

BIODATA PENULIS

Penulis lahir di Jakarta, 19 Maret 1995, dengan nama lengkap Dhamar Bagas Wisesa. Penulis merupakan anak kedua dari dua bersaudara. Riwayat pendidikan penulis yaitu TK An-Nisaa' Tangerang, SD An-Nisaa' Tangerang, SMP An-Nisaa' Tangerang, dan SMA Labschool Kebayoran Jakarta Selatan, dan akhirnya penulis masuk menjadi salah satu mahasiswa Sistem Informasi angkatan 2013 melalui jalur Seleksi Bersama Masuk Perguruan Tinggi Negeri (SBMPTN) dengan NRP 5213100136.

Selama kuliah penulis bergabung dalam organisasi kemahasiswaan, yaitu Himpunan Mahasiswa Sistem Informasi ITS selama dua periode kepengurusan yaitu 2015-2016. Pada organisasi tersebut penulis mengikuti berbagai kegiatan dengan menjadi Kepada Divisi Aplikasi Departemen Aplikasi Teknologi pada periode kepengurusan 2015-2016. Di Departemen Sistem Informasi penulis mengambil bidang minat Rekayasa Data dan Intelegensi Bisnis. Penulis dapat dihubungi melalui email dhamar.bagas@gmail.com.

.

(Halaman ini sengaja dikosongkan)

LAMPIRAN A DESKRIPSI DAN KETERANGAN TAMBAHAN NODE

Tabel A.1 Deskripsi dan Keterangan Tambahan Node

Node	Nama Node	Keterangan
1	Pangkalan Ujung Baru	di jalan komplek hangtuah
2	Jembatan Petekan	bagian kanan kirinya terpisah
3	Jalan sisingamangaraja No 53	Dengan Jalan jakarta No66
4	Jalan Perak Barat No 121	Belokan dari jalan perak timur
5	Jalan Perak Barat No 193a	Dekat ATM BCA
6	Jalan Perak Barat no 245	Dekat Indomaret
7	Jalan Perak Barat no 277	Sebelum Kantor Pos
8	Jalan Perak Barat No 375	Sesudah Bank BRI cabang tanjung perak
9	Berlian, Perak Utara	Ke arah Prapat Kurung Utara
10	Jalan Prapat Kurung Utara no 112	Perempatan prapat kurung dan terminal mirah

11	Jalan Perak Barat No 435	Putaran Balik ke jalan perak timur
12	Jalan Kalianget no 7	Pertigaan ke arah jalan kalianget dan jalan kalimas baru
13	Jalan Kalimas Baru 7-53	Dekat Hidayah Tour and Travel CV
14	Jalan Kalimas Baru 78-85	Dekat Majelis Taklim Nurul Amanah
15	Jl. Kalimas Baru No.26	Setelah Masjid Salafiyah
16	Jl. Pati Unus No.12 A	Dekat Pasar Burung Petekan
17	Jl. Kebalen Tim. No.133	Perempatan indrapura-kebalen timur
18	Jl. Indrapura No.73	Dekat pertigaan indrapura-Dapuan baru
19	Jl. Indrapura Ps. No.119	Pertigaan Persapen Selatan-indrapura
20	Jl. Pesapen No.64	Pertigaan Rajwali-Persapen selatan
21	Jl. Branjangan No.14,	Perempatan Jalan Branjangan -Miliwis
22	Jl. Cendrawasih No.4	Pertigaan Jalan Cnedrawasih-Veteran
23	Jl. Jembatan Merah No.26	Pertigaan Jalan Vetera-Niaga Tambang

24	Jl. Kebon Rojo No.10	Dekat PT Bank Mandiri Taspen Pos Kantor Cabang Surabaya
25	Jl. Krembangan Barat No.2	Pertigaan Jalan Indrapura - Krembangan Barat
26	Jl. Gatotan No.37	Dekat Mess Gatotan
27	Jl. Kebon Rojo No.4	Pertemuan dari Jalan veteran ke jalan kebon rojo
28	Jl. St. Kota Blok B-12 No.24	Pertigaan Jalan Siaga-Stasiun Kota
29	Jl. Semut Kali 20-22	Dekat Indomaret Semut Kali
30	Gg. II No.25-C	Pertigaan Jalan Peneleh-Gang dua
31	Gg. I	Pertigaan ganga1-jalan pandean 5
32	Jl. Undaan Peneleh No.5	Setelah Pertigaan Undan peneleh-undan peneleh IV
33	Jl. Undaan Kulon No.55	Sebelum Pukis Bikang Undaan
34	Jl. Undaan Wetan No.82,	Dikedua undan wetan dan undan kulon
35	Jl. Kalisari Pesarean	pertigaan jalan kalisari-kalisari pesarean

36	Jl. Jaksa Agung Suprapto No.32	Perempatan Jalan Jaksa agung suprapto-ambengan
37	Jl. Ambengan No.41	Pertigaan Jalan Ambengan-Wijaya Kusuma
38	Jl. Wijaya Kusuma 36-38	Sebelum Bakery Assih Snack
39	Jl. Wijaya Kusuma No.5	Pertigaan jalan walikota mustajab-wijaya kusuma
40	Jl. Gubeng Pojok No.	Jalan tengah dari 3 jalur
41	Jl. Kayon No.1	Pertigaan jalan kayun-pemuda
42	Jl. Embong Kemiri No.24,	Pertigaan kayun-emobng kemiri
43	Jl. Karimun Jawa No.4	Dekat Larici Ice cream
44	Jl. Raya Gubeng No.68-D	Pertigaan Jalan Raya Gubeng - Karimun Jawa
45	Jl. Kalimantan No.32	Perempatan jalan raya gubeng-kalimantan
46	Jl. Lingga No.5	Pertigaan Jalan nias-lingga
47	Jl. Nias No.72A	Pertigaan Jalan nias-Jawa
48	Jl. Biliton 55 No.83	Pertigaan Jalan Sulawesi-Biliton

49	Jl. Sulawesi 31-33	Perpecahan jalan ke arah raya gubeng dan sulawesi
50	Jl. Lombok No.2	Pertigaan Jalan raya ngagel-lombok
51	Jl. Raya Ngagel No.109	Dekat Ratu Computer
52	Jl. Raya Ngagel No.121	Pertigaan Jalan Upajiwa-Raya ngagel
53	Jl. Upa Jiwa No.3	Dekat Masjid Ridha Allah
54	Jl. Ngagel Jaya Sel. No.23B	Dekat BCA KCP Ngagel Jaya Selatan
55	Jl. Ngagel Jaya Sel. No.85	Pertigaan jalan ngagel jaya selatan - raya ngagel selatan
56	Jl. Raya Ngagel Jaya No.66	Dekat Terang Bulan Hepi
57	Jl. Raya Ngagel Jaya No.32	Perempatan Jalan Raya Ngagel Jaya - Ngagel Jaya Utara
58	Jl. Pucang Anom Tim. No.70	Sebelum Bumiputera 1912
59	Jl. Pucang Anom Tim. No.30	Pertigaan Jalan Pucang Anom timur - Pucang Adi
60	Jl. Pucang Anom Tim. No.4	Pertigaan Jalan Pucang Anom timur - Gubeng Kertajaya

61	Jl. Kertajaya No.110 B	Dekat Shooping Mall Sari Tama
62	Jl. Kertajaya No.2	Ujung Jalan kertajaya. Untuk Putar Balik
63	Jl. Kertajaya No.97	Dekat Toko Sahabat
64	Jl. Raya Menur No.44	Perempatan Jalan Kertajaya - Mawar
65	Jl. Raya Menur No.17	Dekat Warkop 217
66	Jl. Raya Menur No.111	Pertigaan Jalan Raya Menur - Manyar
67	Jl. Raya Menur No.127	Puteran Jalan Menur Pumpungan, Raya Manyar, Kalibokor Selatan, Raya Menur
68	Jl. Raya Manyar No.27	Dekat Electronic Store UD Camar
69	Jl. Raya Manyar No.57B	Dekat Depot Golden Wok
70	Jl. Raya Nginden No.20-A	Putaran Balik Jalan Raya Nginden
71	Terminal Bratang	Di Jalan Raya Manyar
72	Jl. Raya Manyar No.80A	Pertigaan Jalan Raya Manyar - Bratang Binangun
73	Jl. Ngagel Jaya Selatan No.168	Pertigaan jalan ngagel jaya selatan -Bratang Binangun

74	Jl. Ngagel Timur IV No.33	Ujung Jalan Ngagel Jaya Utara
75	Jl. Ngagel Timur IV No.1-B,	Pertigaan Ngagel Timur 4 - Ngagel tim.
76	Jl. Kalibokor Timur No.33	Setelah Sabrina Mode
77	Jl. Pucang Anom No.22	Pertigaan Jalan Pucang Anom- Pucan Anom 1
78	Jl. Kali Bokor I No.2	Dekat Jimboo
79	Jl. Bagong Ginayan IV No.9 B	Pertigaan Jalan Bagong Ginayan - Bagong Karitama
80	Jl. Nias No.150	Setelah Lotus
81	Jl. Pemuda No.17	Perempatan Jalan Yos sudarso - pemuda
82	Jl. Kalisari I No.8	Pertigaan Jalan Kalisari 1 - Kalisari 3
83	Jl. Ambengan No.1B	Perempatan Jalan Ngemplak - Ambengan
84	Jl. Pengampon No.35	Dekat Nasi Empal Pengampon
85	Jl. Bunguran No.3	Sebelum Shopping Mall Varia
86	Jl. St. Kota No.4	Dekat UD. Seni Hiburan
87	Jl. Slompretan No.2	Pertigaan Jalan Waspada - Slompretan

88	Jl. Karet No.39	Dekat PT. Metro Plastik Nusantara
89	Jl. Kepanjen No.28	Pertigaan Jalan Kepanjen-Sikatan
90	Jl. Kunir No.5	Perempatan Jalan Kunir-Krembangan
91	Jl. Krembangan Makam No.45	Pertigaan Jalan Krembangan Buyut - Krembangan Makam
92	Terminal Joyoboyo	Terletak di Jalan Joyoboyo
93	Jl. Gunung Sari No.24-C	Putar Balik ke arah Jalan Gunung Sari
94	Jl. Raya Darmo No.86	Sebelum Kebun Binatang Surabaya
95	Jl. Raya Darmo No.139	Putaran Balik Ke Jalan Raya Malang
96	Jl.Jembatan Wonokromo	Serta Jalan Raya Malang
97	Jl. Raya Wonokromo No.76	Putaran Balik ke Arah Jalan Stasiun Wonokromo
98	Jl. St. Wonokromo No.71	Perempatan Jalan Jagir Wonokromo-Stasiun Wonokromo
99	Jl. Ngagel Rejo Kidul No.7	Pertigaan Jalan Raya Ngagel - Ngagel Raya Kidul

100	Jl. Ngagel Rejo Kidul 28-32	Dekat Masjid Muslih Cholil
101	Jl. Bratang Gede 15-17	Dekat Warung Pak Achmad
102	Jl. Bratang Gede No.127	Pertigaan Jalan Bratang Wetan 2-Bratang Gede
103	Jl. Bratang Jaya No.42	Dekat Angkringan Ngeng Ngeng
104	Jl. Frontage Ahmad Yani Siwalankerto No.19	Pertigaan Jalan Bendul Merisis - Frontage Ahmad Yani Siwalankerto
105	Jl. Bendul Merisi 11-49	Dekat SJ Café
106	Jl. Bendul Merisi No.113	Pertigaan Jalan Bendul Merisi - Bentul 6
107	Jl. Bendul Merisi No.2A,	Pertigaan Jalan Jagir Sidosermo 11 - Jagir Sidosermo 12 - Bendul Merisi
108	Jl. Jagir Wonokromo No.148	Pertigaan Jalan Jagir Wonokromo - Jagir Sidosermo1
109	Jl. Jagir Wonokromo 187-189	Dekat Toko Antasari
110	Jl. Jagir Wonokromo No.285	Dekat Atm Bank Jatim

111	Jl. Panjang Jiwo No.38	Dekat Alfamart Panjang Jiwo
112	Ruko Panji Makmur	Setelah Agen Beras Taj Mahal
113	Jl. Raya Rungkut Lor No.5	Sebelum Abadi Bengkel
114	Jl. Raya Kalirungkut No.20F	Dekat Wiratama Perus
115	Jl. Raya Kalirungkut No.76	Dekat Pasar Kedaung
116	Rungkut Puskesmas No.1	Dekat Rachmat Electro
117	Jl. Raya Kedung Asem No.7	Pertigaan Jalan Raya Kedung Asem
118	Jl. Raya Kedung Asem No.140	Pertigaan Jalan Raya Kedung Asem - Baruk Bar 6
119	Jl. Dr. Ir. H. Soekarno No.104	Putaran balik ke Jl. Dr. Ir. H. Soekarno No.104
120	Jl. Pandugo No.27 A	Dekat Toko Bejo
121	Jl. Penjaringan Tim. No.22	Dekat Agung Bakso Reog
122	Penjaringan Timur No.18	Dekat Ayam Bakar Cak Bram
123	Jl. Kendal Sari No.27	Dekat Coto Makassar
124	Jl. Wonorejo Sel. No.5	Dekat Century Surabaya

125	Jl. Raya Wonorejo No.2 D	Pangkalan Wonorejo
126	Jl. Penjaringan No.8	Sesudah Tiara UD
127	Jl. Dr. Ir. H. Soekarno No.20	Setelah Wolf's Lair
128	Jl. Raya Kedung Asem 70-78	Sebelum Devil Cell
129	Jl. Raya Kedung Baruk No.141	Dekat Warung Bu Sun
130	Jl. Raya Kedung Baruk No.8	Setelah Warung Cak to
131	Jl. Raya Kedung Baruk Blok Q No.23	Dekat Bank Perkreditan Rakyat Bintang Mitra
132	Ruko Mangga Dua, Jl. Jagir Wonokromo No.100	Dekat Mall Mangga Dua
133	Jagir Wonokromo Gg. VIII	Dekat Pusat Agen Ayam
134	Jl. Darmokali No.77	Dekat Dunia Grafindo
135	Jl. Darmokali No.66	Setelah Ayam Bakar Tongkat Solo
136	Jl. Darmokali No.26	Dekat Kb Tk Al-Hikmah
137	Jl. Raya Dinoyo No.141	Dekat Soto Ayam Pak Lan

138	Jl. Raya Dinoyo No.79 B	Dekat Restoran Minang roso
139	Jl. Raya Dinoyo 37-41	Pertigaan Jalan Raya Dinoyo- Mojopahit
140	Jl. Sriwijaya No.9	Dekat Wahana Persada Nusantara
141	Jl. Keputeran No.74	Dekat Gunawan Cell
142	Jl. Kayon No.90	Sebelum Turin Italian Ice Cream
143	Jl. Jaksa Agung Suprapto No.1	Pertigaan Jalan Walikota Mustajab- Jaksa agung Suprapto
144	Jl. Genteng Kali 37-49	Perbatasan antara Jalan Genteng kali- Undaan Kulon
145	Jl. Tembaan No.A-17	Dekat Bebek Tugu Pahlawan
146	Jl. Bubutan No.141	Dekat Ampera Masakan Padang
147	Jl. Krembangan Barat No.63	Sebelum Pertigaan Jalan Krembangan Barat - Merak
148	Jl. Kasuari No.3	Dekat De Javasche Bank
149	Jl. Kalimas Barat 3-5	Sebelum Pertigaan Jalan Kalimas Barat - Kelasi
150	Jl. Kalimas Barat No.53	Sebelum Bengkel Body Repair ARM

151	Jl. Kalimas Barat No.67	Pangkalan Kalimas barat
152	Jl. Gembong No.30	Dekat ITC Surabaya
153	Jl. Pecindilan No.46-48	Pertigaan Jalan Pencicilan -Pencicilan 4
154	Jl. Sono Kembang	Pertigaan Panglima Sudirman - Jalan Karimun Jawa
155	Jl. Urip Sumoharjo No.112	Dekat Toko Duta Abadi
156	Jl. Ratna No.2B	Pertigaan Jalan Ratna-Raya Ngagel
157	Krukah Tim. IV No.7	Dekat Toko Ibu Fauzan
158	Jl. Menur Pumpungan No.67,	Dekat Waring Asri
159	Jl. Menur Pumpungan No.167,	Dekat Apotek Menur
160	Jl. Klampis Jaya No.A8	Dekat Metro Star Komputer
161	Jl. Klampis Jaya No.29	Dekat Restoran Selera Makmur
162	Jl. Klampis Jaya No.47E	Perempatan Jalan Klampis Jaya - Mleto - Manyar Kertoadi
163	Jl. Dr. Ir. H. Soekarno No.98	Dekat Restoran Padang Sederhana

164	Jl. Manyar Kertoarjo No.96	Dekat Hotel Swiss-Belin Manyar
165	Jl. Gubeng Kertajaya XI No.22	Dekat Mata Art Photography
166	Jl. Gubeng Kertajaya XV No.13	Dekat Sekolah Dasar Negeri Airlangga 3 No.200
167	Jl. Srikana No.38	Dekat ABS Pet Gallery
168	Dharmawangsa IX No.2	Dekat Parahita Diagnostic Center
169	Jl. Dharmawangsa No.6 B	Dekat Warung Bu "Rus"
170	Jl. Prof. Dr. Mustopo No.2	Pertigaan Jalan Prof. Dr. Mustopo - Tapak Siring
171	Jl. Tapak Siring,	Percabangan Jalan Tapak Siring - Residen Sudirman - Indrakila
172	Jl. Ambengan No.68	Dekat Kantin Mekar
173	Jl. Kusuma Bangsa No.29	Dekat Bebek Semangat
174	Jl. Kusuma Bangsa No.130	Dekat RR Depot Nasi Pecel & Nasi Campur
175	Jl. Kapasari No.45	Dekat Toko Astika
176	Jl. Kapasari No.126-128	Dekat SMA Yppi-ii
177	Jl. Simolawang Baru 28-30	Dekat Harapan Samudra CV

178	Jl. Sidodadi Baru 2-4	Dekat Depot Banyuwangi
179	Jl. Sombo No.21	Dekat Toko Fatri Barokah
180	Jl. Pegirian No.230	Dekat Almeira Homestay
181	Jl. Karang Tembok No.1	Dekat SMP Khm. Nur
182	Jl. Wonosari Lor No.10A	Dekat Toko Rahmad
183	Jl. Wonosari Lor No.19	Dekat Giras Goyang 25
184	Jl. Endrosono No.116	Pertigaan Jalan Wonosari Lor - Bulak Sari
185	Wonosari Mulyo Gg. II No.1	Dekat Toko Rejo Putra
186	Jl. Bulak Rukem No.19	Pertigaan Jalan Bulak Rukem - Tenggumung Wetan
187	Jl. Bulak Banteng Baru No.17	Pangkalan Bulak Banteng
188	Jl. Ngaglik 17-23	Dekat Warteg Fatimah
189	Jl. Tambaksari 87-89	Dekat Restoran Pak Di
190	Jl. Tambaksari No.6	Dekat Kampung Roti
191	Jl. Residen Sudirman No.1	Dekat Mie Mangkok Melet

192	Jl. Dharmawangsa No.128	Dekat Pattaya16
193	Jl. Tambak Wedi Barat	Dekat Amir Cell
194	Jl. Dukuh Bulak Banteng Tim. No.5	Dekat Aneka Bakery
195	Jl. Kedinding Lor No.8-A	Dekat Warung Madura
196	Jl. Kedinding Lor 4-5	Dekat Warteg Pecel Lele Handayani
197	Jl. Kedung Cowek No.370	Pertigaan Jalan Kedung Cowek - Kedinding Tengah Jaya IV
198	Jl. Kedung Cowek No.274	Dekat Warung Pak Agus
199	Jl. Kedung Cowek No.220	Dekat Restoran Bu Rinda
200	Jl. Kedung Cowek No.138	Dekat Aris Gordyn
201	Jl. Kedung Cowek No.84D	Pertigaan Jalan Kedung Cowek - Gading Karya
202	Jl. Kedung Cowek No.19	Dekat Apotek Super Apotik
203	Jl. Kenjeran No.226	Dekat Grand Kenjeran Resto
204	Jl. Kenjeran 131-133	Dekat Toko Meubel New Anugerah

205	Jl. Tambak Rejo No.85	Toko Natasya
206	Jl. Tambak Rejo No.1	Pertigaan Jalan Tambak Rejo - Tambak Arum gang 1
207	Jl. Kapas Krampung 142-144	Dekat Bank ShinHan Indonesia
208	Jl. Kapas Krampung 109-111	Dekat Warung Empal Bu Yudi
209	Jl. Raya Karang Asem 16-20	Dekat Kantor Notaris Ani Widyasari Rr SH
210	Jl. Bronggalan 18-20	Dekat Soto Ayam Santoso
211	Jl. Tambang Boyo No.146	Pertigaan Jalan Tambang Boyo - Pacarkembang
212	Jl. Kedung Sroko II No.3	Pertigaan Jalan Tambang Boyo - Kedung Sroko 2
213	Jl. Prof. Dr. Mustopo No.29	Dekat Universitas Kedokteran Airlangga
214	Jl. Dharmahusada No.87	Dekat Supermarket MM Jaya
215	Jl. Dharmahusada 36-48	Dekat Futsalholic Shop
216	Jl. Prof. Dr. Mustopo No.195	Dekat Superindo Dharmahusada

217	Jl. Raya Dharma Husada Indah Blok A No.10	Pertigaan Jalan Raya Dharmahusada - Dharmahusada Indah Utara 1
218	Jl. Raya Dharma Husada Indah No.57	Ke arah Greja Kristus Yesus Surabaya
219	Jl. Manyar Kertoadi No.55	Warung Mbok'e Dhewe
220	Manyar Kerta Adi No.83	Dekat Unicorn Rental
221	Jl. Gebang Putih	Dekat Warkop Barcelona
222	Jl. Gebang Putih No.10	Dekat Kantin Kedai Cerita
223	Jl. Arief Rachman Hakim No.179	Dekat Araya Family Club
224	Jl. Arief Rachman Hakim	Dekat Medical Center ITS
225	Jl. Arief Rachman Hakim No.19	Dekat SMP Vita
226	Jl. Keputih Tegal No.48	Dekat Toko Bintang Abadi
227	Jl. Medokan Keputih	Dekat Toko ULFAH Jaya
228	Jl. Arief Rachman Hakim No.8 A	Dekat Warkop Hendro
229	Jl. Dr. Ir. H. Soekarno No.103C	Dekat SMPN 19
230	Jl. Dr. Ir. H. Soekarno	Dekat Renggenis

231	Jl. Raya Kertajaya Indah No.88	Setelah Gedung Yayasan Bhakti Raga
232	Jl. Kedung Sroko No.74	Dekat Depot Cocok
233	Jl. Pacar Keling No.18	Perempatan Jalan Pacar Keling - Pacar Keling 3
234	Jl. Pacarkeling III No.59	Dekat Rumah Susu Surabaya
235	Jl. Putro Agung Kulon No.23	Dekat Ayama & Bebek Goreng

(Halaman ini sengaja dikosongkan)

LAMPIRAN B VARIABEL KEPUTUSAN MODEL ORIENTEERING PROBLEM

Tabel B.1 Variabel Keputusan OP

Variabel	Deskripsi
x1.2	Kunjungan dari node 1 ke node 2
x2.3	Kunjungan dari node 2 ke node 3
x2.1	Kunjungan dari node 2 ke node 1
x2.16	Kunjungan dari node 2 ke node 16
x3.4	Kunjungan dari node 3 ke node 4
x4.5	Kunjungan dari node 4 ke node 5
x5.6	Kunjungan dari node 5 ke node 6
x6.7	Kunjungan dari node 6 ke node 7
x7.8	Kunjungan dari node 7 ke node 8
x8.9	Kunjungan dari node 8 ke node 9
x9.10	Kunjungan dari node 9 ke node 10
x10.11	Kunjungan dari node 10 ke node 11
x11.12	Kunjungan dari node 11 ke node 12
x12.13	Kunjungan dari node 12 ke node 13
x13.14	Kunjungan dari node 13 ke node 14
x14.13	Kunjungan dari node 14 ke node 13
x14.15	Kunjungan dari node 14 ke node 15
x15.14	Kunjungan dari node 15 ke node 14
x15.2	Kunjungan dari node 15 ke node 2
x16.17	Kunjungan dari node 16 ke node 17
x17.18	Kunjungan dari node 17 ke node 18
x17.2	Kunjungan dari node 17 ke node 2
x18.17	Kunjungan dari node 18 ke node 17

x18.19	Kunjungan dari node 18 ke node 19
x18.17	Kunjungan dari node 18 ke node 17
x19.18	Kunjungan dari node 19 ke node 18
x19.20	Kunjungan dari node 19 ke node 20
x19.18	Kunjungan dari node 19 ke node 18
x20.21	Kunjungan dari node 20 ke node 21
x20.19	Kunjungan dari node 20 ke node 19
x21.22	Kunjungan dari node 21 ke node 22
x22.23	Kunjungan dari node 22 ke node 23
x22.89	Kunjungan dari node 22 ke node 89
x23.24	Kunjungan dari node 23 ke node 24
x23.27	Kunjungan dari node 23 ke node 27
x24.25	Kunjungan dari node 24 ke node 25
x25.26	Kunjungan dari node 25 ke node 26
x25.147	Kunjungan dari node 25 ke node 147
x26.27	Kunjungan dari node 26 ke node 27
x26.23	Kunjungan dari node 26 ke node 23
x27.28	Kunjungan dari node 27 ke node 28
x27.145	Kunjungan dari node 27 ke node 145
x28.29	Kunjungan dari node 28 ke node 29
x28.152	Kunjungan dari node 28 ke node 152
x28.27	Kunjungan dari node 28 ke node 27
x29.30	Kunjungan dari node 29 ke node 30
x30.31	Kunjungan dari node 30 ke node 31
x31.32	Kunjungan dari node 31 ke node 32
x32.33	Kunjungan dari node 32 ke node 33
x33.34	Kunjungan dari node 33 ke node 34

x34.84	Kunjungan dari node 34 ke node 84
x34.35	Kunjungan dari node 34 ke node 35
x34.83	Kunjungan dari node 34 ke node 83
x35.82	Kunjungan dari node 35 ke node 82
x36.37	Kunjungan dari node 36 ke node 37
x36.143	Kunjungan dari node 36 ke node 143
x36.83	Kunjungan dari node 36 ke node 83
x36.82	Kunjungan dari node 36 ke node 82
x37.38	Kunjungan dari node 37 ke node 38
x37.36	Kunjungan dari node 37 ke node 36
x38.39	Kunjungan dari node 38 ke node 39
x38.37	Kunjungan dari node 38 ke node 37
x39.40	Kunjungan dari node 39 ke node 40
x39.38	Kunjungan dari node 39 ke node 38
x40.41	Kunjungan dari node 40 ke node 41
x41.42	Kunjungan dari node 41 ke node 42
x41.81	Kunjungan dari node 41 ke node 81
x42.43	Kunjungan dari node 42 ke node 43
x42.142	Kunjungan dari node 42 ke node 142
x42.154	Kunjungan dari node 42 ke node 154
x42.41	Kunjungan dari node 42 ke node 41
x43.44	Kunjungan dari node 43 ke node 44
x43.42	Kunjungan dari node 43 ke node 42
x44.45	Kunjungan dari node 44 ke node 45
x44.43	Kunjungan dari node 44 ke node 43
x45.46	Kunjungan dari node 45 ke node 46
x46.47	Kunjungan dari node 46 ke node 47

x47.46	Kunjungan dari node 47 ke node 46
x47.48	Kunjungan dari node 47 ke node 48
x48.49	Kunjungan dari node 48 ke node 49
x49.50	Kunjungan dari node 49 ke node 50
x49.44	Kunjungan dari node 49 ke node 44
x50.51	Kunjungan dari node 50 ke node 51
x51.50	Kunjungan dari node 51 ke node 50
x51.52	Kunjungan dari node 51 ke node 52
x52.51	Kunjungan dari node 52 ke node 51
x52.53	Kunjungan dari node 52 ke node 53
x53.156	Kunjungan dari node 53 ke node 156
x53.54	Kunjungan dari node 53 ke node 54
x54.55	Kunjungan dari node 54 ke node 55
x54.74	Kunjungan dari node 54 ke node 74
x55.56	Kunjungan dari node 55 ke node 56
x55.54	Kunjungan dari node 55 ke node 54
x56.55	Kunjungan dari node 56 ke node 55
x56.57	Kunjungan dari node 56 ke node 57
x57.56	Kunjungan dari node 57 ke node 56
x57.58	Kunjungan dari node 57 ke node 58
x58.57	Kunjungan dari node 58 ke node 57
x58.59	Kunjungan dari node 58 ke node 59
x58.77	Kunjungan dari node 58 ke node 77
x59.58	Kunjungan dari node 59 ke node 58
x59.60	Kunjungan dari node 59 ke node 60
x60.59	Kunjungan dari node 60 ke node 59
x60.61	Kunjungan dari node 60 ke node 61

x61.62	Kunjungan dari node 61 ke node 62
x62.63	Kunjungan dari node 62 ke node 63
x63.64	Kunjungan dari node 63 ke node 64
x64.65	Kunjungan dari node 64 ke node 65
x64.164	Kunjungan dari node 64 ke node 164
x64.165	Kunjungan dari node 64 ke node 165
x65.66	Kunjungan dari node 65 ke node 66
x65.64	Kunjungan dari node 65 ke node 64
x66.65	Kunjungan dari node 66 ke node 65
x66.67	Kunjungan dari node 66 ke node 67
x67.66	Kunjungan dari node 67 ke node 66
x67.68	Kunjungan dari node 67 ke node 68
x67.158	Kunjungan dari node 67 ke node 158
x68.69	Kunjungan dari node 68 ke node 69
x68.67	Kunjungan dari node 68 ke node 67
x69.68	Kunjungan dari node 69 ke node 68
x69.70	Kunjungan dari node 69 ke node 70
x70.71	Kunjungan dari node 70 ke node 71
x71.72	Kunjungan dari node 71 ke node 72
x72.71	Kunjungan dari node 72 ke node 71
x72.73	Kunjungan dari node 72 ke node 73
x73.55	Kunjungan dari node 73 ke node 55
x73.69	Kunjungan dari node 73 ke node 69
x73.157	Kunjungan dari node 73 ke node 157
x73.72	Kunjungan dari node 73 ke node 72
x74.75	Kunjungan dari node 74 ke node 75
x75.74	Kunjungan dari node 75 ke node 74

x75.76	Kunjungan dari node 75 ke node 76
x76.57	Kunjungan dari node 76 ke node 57
x76.58	Kunjungan dari node 76 ke node 58
x77.78	Kunjungan dari node 77 ke node 78
x78.77	Kunjungan dari node 78 ke node 77
x78.79	Kunjungan dari node 78 ke node 79
x79.78	Kunjungan dari node 79 ke node 78
x79.80	Kunjungan dari node 79 ke node 80
x80.79	Kunjungan dari node 80 ke node 79
x80.48	Kunjungan dari node 80 ke node 48
x81.39	Kunjungan dari node 81 ke node 39
x81.143	Kunjungan dari node 81 ke node 143
x82.34	Kunjungan dari node 82 ke node 34
x82.35	Kunjungan dari node 82 ke node 35
x83.33	Kunjungan dari node 83 ke node 33
x83.36	Kunjungan dari node 83 ke node 36
x84.85	Kunjungan dari node 84 ke node 85
x85.86	Kunjungan dari node 85 ke node 86
x85.28	Kunjungan dari node 85 ke node 28
x86.87	Kunjungan dari node 86 ke node 87
x87.86	Kunjungan dari node 87 ke node 86
x87.88	Kunjungan dari node 87 ke node 88
x88.22	Kunjungan dari node 88 ke node 22
x89.90	Kunjungan dari node 89 ke node 90
x89.23	Kunjungan dari node 89 ke node 23
x90.91	Kunjungan dari node 90 ke node 91
x90.89	Kunjungan dari node 90 ke node 89

x91.20	Kunjungan dari node 91 ke node 20
x92.93	Kunjungan dari node 92 ke node 93
x93.94	Kunjungan dari node 93 ke node 94
x93.92	Kunjungan dari node 93 ke node 92
x94.95	Kunjungan dari node 94 ke node 95
x95.96	Kunjungan dari node 95 ke node 96
x95.134	Kunjungan dari node 95 ke node 134
x96.97	Kunjungan dari node 96 ke node 97
x97.98	Kunjungan dari node 97 ke node 98
x97.104	Kunjungan dari node 97 ke node 104
x98.99	Kunjungan dari node 98 ke node 99
x98.132	Kunjungan dari node 98 ke node 132
x98.133	Kunjungan dari node 98 ke node 133
x99.98	Kunjungan dari node 99 ke node 98
x99.100	Kunjungan dari node 99 ke node 100
x100.99	Kunjungan dari node 100 ke node 99
x100.101	Kunjungan dari node 100 ke node 101
x101.100	Kunjungan dari node 101 ke node 100
x101.102	Kunjungan dari node 101 ke node 102
x102.103	Kunjungan dari node 102 ke node 103
x102.101	Kunjungan dari node 102 ke node 101
x103.72	Kunjungan dari node 103 ke node 72
x103.157	Kunjungan dari node 103 ke node 157
x104.105	Kunjungan dari node 104 ke node 105
x105.106	Kunjungan dari node 105 ke node 106
x106.105	Kunjungan dari node 106 ke node 105
x106.107	Kunjungan dari node 106 ke node 107

x107.106	Kunjungan dari node 107 ke node 106
x107.108	Kunjungan dari node 107 ke node 108
x108.107	Kunjungan dari node 108 ke node 107
x108.109	Kunjungan dari node 108 ke node 109
x108.132	Kunjungan dari node 108 ke node 132
x109.110	Kunjungan dari node 109 ke node 110
x109.108	Kunjungan dari node 109 ke node 108
x110.109	Kunjungan dari node 110 ke node 109
x110.111	Kunjungan dari node 110 ke node 111
x111.110	Kunjungan dari node 111 ke node 110
x111.112	Kunjungan dari node 111 ke node 112
x112.111	Kunjungan dari node 112 ke node 111
x112.113	Kunjungan dari node 112 ke node 113
x113.112	Kunjungan dari node 113 ke node 112
x113.114	Kunjungan dari node 113 ke node 114
x114.113	Kunjungan dari node 114 ke node 113
x114.115	Kunjungan dari node 114 ke node 115
x115.114	Kunjungan dari node 115 ke node 114
x115.116	Kunjungan dari node 115 ke node 116
x116.115	Kunjungan dari node 116 ke node 115
x116.117	Kunjungan dari node 116 ke node 117
x117.116	Kunjungan dari node 117 ke node 116
x117.118	Kunjungan dari node 117 ke node 118
x118.119	Kunjungan dari node 118 ke node 119
x118.128	Kunjungan dari node 118 ke node 128
x119.120	Kunjungan dari node 119 ke node 120
x120.121	Kunjungan dari node 120 ke node 121

x121.120	Kunjungan dari node 121 ke node 120
x121.122	Kunjungan dari node 121 ke node 122
x121.126	Kunjungan dari node 121 ke node 126
x122.121	Kunjungan dari node 122 ke node 121
x122.123	Kunjungan dari node 122 ke node 123
x123.122	Kunjungan dari node 123 ke node 122
x123.124	Kunjungan dari node 123 ke node 124
x124.123	Kunjungan dari node 124 ke node 123
x124.125	Kunjungan dari node 124 ke node 125
x125.124	Kunjungan dari node 125 ke node 124
x126.121	Kunjungan dari node 126 ke node 121
x126.127	Kunjungan dari node 126 ke node 127
x127.118	Kunjungan dari node 127 ke node 118
x128.129	Kunjungan dari node 128 ke node 129
x129.128	Kunjungan dari node 129 ke node 128
x129.130	Kunjungan dari node 129 ke node 130
x130.129	Kunjungan dari node 130 ke node 129
x130.131	Kunjungan dari node 130 ke node 131
x131.130	Kunjungan dari node 131 ke node 130
x131.112	Kunjungan dari node 131 ke node 112
x132.108	Kunjungan dari node 132 ke node 108
x132.98	Kunjungan dari node 132 ke node 98
x133.93	Kunjungan dari node 133 ke node 93
x134.135	Kunjungan dari node 134 ke node 135
x134.96	Kunjungan dari node 134 ke node 96
x135.136	Kunjungan dari node 135 ke node 136
x135.134	Kunjungan dari node 135 ke node 134

x136.137	Kunjungan dari node 136 ke node 137
x136.135	Kunjungan dari node 136 ke node 135
x137.138	Kunjungan dari node 137 ke node 138
x137.52	Kunjungan dari node 137 ke node 52
x137.136	Kunjungan dari node 137 ke node 136
x138.139	Kunjungan dari node 138 ke node 139
x138.137	Kunjungan dari node 138 ke node 137
x139.140	Kunjungan dari node 139 ke node 140
x139.138	Kunjungan dari node 139 ke node 138
x140.141	Kunjungan dari node 140 ke node 141
x141.142	Kunjungan dari node 141 ke node 142
x142.42	Kunjungan dari node 142 ke node 42
x142.141	Kunjungan dari node 142 ke node 141
x142.43	Kunjungan dari node 142 ke node 43
x143.144	Kunjungan dari node 143 ke node 144
x143.39	Kunjungan dari node 143 ke node 39
x144.33	Kunjungan dari node 144 ke node 33
x145.146	Kunjungan dari node 145 ke node 146
x146.25	Kunjungan dari node 146 ke node 25
x147.148	Kunjungan dari node 147 ke node 148
x147.21	Kunjungan dari node 147 ke node 21
x148.149	Kunjungan dari node 148 ke node 149
x148.22	Kunjungan dari node 148 ke node 22
x149.148	Kunjungan dari node 149 ke node 148
x149.150	Kunjungan dari node 149 ke node 150
x150.149	Kunjungan dari node 150 ke node 149
x150.151	Kunjungan dari node 150 ke node 151

x151.150	Kunjungan dari node 151 ke node 150
x152.153	Kunjungan dari node 152 ke node 153
x153.34	Kunjungan dari node 153 ke node 34
x154.155	Kunjungan dari node 154 ke node 155
x154.142	Kunjungan dari node 154 ke node 142
x154.43	Kunjungan dari node 154 ke node 43
x155.139	Kunjungan dari node 155 ke node 139
x156.52	Kunjungan dari node 156 ke node 52
x156.137	Kunjungan dari node 156 ke node 137
x156.136	Kunjungan dari node 156 ke node 136
x157.73	Kunjungan dari node 157 ke node 73
x158.67	Kunjungan dari node 158 ke node 67
x158.159	Kunjungan dari node 158 ke node 159
x159.158	Kunjungan dari node 159 ke node 158
x159.160	Kunjungan dari node 159 ke node 160
x160.159	Kunjungan dari node 160 ke node 159
x160.161	Kunjungan dari node 160 ke node 161
x161.160	Kunjungan dari node 161 ke node 160
x161.162	Kunjungan dari node 161 ke node 162
x162.161	Kunjungan dari node 162 ke node 161
x162.163	Kunjungan dari node 162 ke node 163
x162.219	Kunjungan dari node 162 ke node 219
x163.164	Kunjungan dari node 163 ke node 164
x163.162	Kunjungan dari node 163 ke node 162
x163.218	Kunjungan dari node 163 ke node 218
x164.163	Kunjungan dari node 164 ke node 163
x164.64	Kunjungan dari node 164 ke node 64

x165.166	Kunjungan dari node 165 ke node 166
x166.165	Kunjungan dari node 166 ke node 165
x166.167	Kunjungan dari node 166 ke node 167
x167.166	Kunjungan dari node 167 ke node 166
x167.168	Kunjungan dari node 167 ke node 168
x168.169	Kunjungan dari node 168 ke node 169
x168.192	Kunjungan dari node 168 ke node 192
x169.168	Kunjungan dari node 169 ke node 168
x169.170	Kunjungan dari node 169 ke node 170
x170.169	Kunjungan dari node 170 ke node 169
x170.171	Kunjungan dari node 170 ke node 171
x171.170	Kunjungan dari node 171 ke node 170
x171.172	Kunjungan dari node 171 ke node 172
x172.173	Kunjungan dari node 172 ke node 173
x173.172	Kunjungan dari node 173 ke node 172
x173.174	Kunjungan dari node 173 ke node 174
x174.173	Kunjungan dari node 174 ke node 173
x174.175	Kunjungan dari node 174 ke node 175
x175.174	Kunjungan dari node 175 ke node 174
x175.176	Kunjungan dari node 175 ke node 176
x175.188	Kunjungan dari node 175 ke node 188
x176.175	Kunjungan dari node 176 ke node 175
x176.177	Kunjungan dari node 176 ke node 177
x177.176	Kunjungan dari node 177 ke node 176
x177.178	Kunjungan dari node 177 ke node 178
x178.177	Kunjungan dari node 178 ke node 177
x178.179	Kunjungan dari node 178 ke node 179

x179.178	Kunjungan dari node 179 ke node 178
x179.180	Kunjungan dari node 179 ke node 180
x180.179	Kunjungan dari node 180 ke node 179
x180.181	Kunjungan dari node 180 ke node 181
x181.180	Kunjungan dari node 181 ke node 180
x181.182	Kunjungan dari node 181 ke node 182
x182.181	Kunjungan dari node 182 ke node 181
x182.183	Kunjungan dari node 182 ke node 183
x183.182	Kunjungan dari node 183 ke node 182
x183.184	Kunjungan dari node 183 ke node 184
x184.183	Kunjungan dari node 184 ke node 183
x184.185	Kunjungan dari node 184 ke node 185
x185.184	Kunjungan dari node 185 ke node 184
x185.186	Kunjungan dari node 185 ke node 186
x186.185	Kunjungan dari node 186 ke node 185
x186.187	Kunjungan dari node 186 ke node 187
x187.186	Kunjungan dari node 187 ke node 186
x188.175	Kunjungan dari node 188 ke node 175
x188.189	Kunjungan dari node 188 ke node 189
x189.190	Kunjungan dari node 189 ke node 190
x190.191	Kunjungan dari node 190 ke node 191
x191.171	Kunjungan dari node 191 ke node 171
x192.63	Kunjungan dari node 192 ke node 63
x192.168	Kunjungan dari node 192 ke node 168
x193.194	Kunjungan dari node 193 ke node 194
x194.193	Kunjungan dari node 194 ke node 193
x194.195	Kunjungan dari node 194 ke node 195

x195.194	Kunjungan dari node 195 ke node 194
x195.196	Kunjungan dari node 195 ke node 196
x196.195	Kunjungan dari node 196 ke node 195
x196.197	Kunjungan dari node 196 ke node 197
x197.196	Kunjungan dari node 197 ke node 196
x197.198	Kunjungan dari node 197 ke node 198
x198.197	Kunjungan dari node 198 ke node 197
x198.199	Kunjungan dari node 198 ke node 199
x199.198	Kunjungan dari node 199 ke node 198
x199.200	Kunjungan dari node 199 ke node 200
x200.199	Kunjungan dari node 200 ke node 199
x200.201	Kunjungan dari node 200 ke node 201
x201.200	Kunjungan dari node 201 ke node 200
x201.202	Kunjungan dari node 201 ke node 202
x202.201	Kunjungan dari node 202 ke node 201
x202.203	Kunjungan dari node 202 ke node 203
x203.204	Kunjungan dari node 203 ke node 204
x204.203	Kunjungan dari node 204 ke node 203
x204.205	Kunjungan dari node 204 ke node 205
x205.204	Kunjungan dari node 205 ke node 204
x205.206	Kunjungan dari node 205 ke node 206
x206.205	Kunjungan dari node 206 ke node 205
x206.207	Kunjungan dari node 206 ke node 207
x207.206	Kunjungan dari node 207 ke node 206
x207.208	Kunjungan dari node 207 ke node 208
x208.207	Kunjungan dari node 208 ke node 207
x208.209	Kunjungan dari node 208 ke node 209

x209.208	Kunjungan dari node 209 ke node 208
x209.210	Kunjungan dari node 209 ke node 210
x209.235	Kunjungan dari node 209 ke node 235
x235.209	Kunjungan dari node 235 ke node 209
x210.211	Kunjungan dari node 210 ke node 211
x210.234	Kunjungan dari node 210 ke node 234
x234.210	Kunjungan dari node 234 ke node 210
x211.233	Kunjungan dari node 211 ke node 233
x211.212	Kunjungan dari node 211 ke node 212
x212.213	Kunjungan dari node 212 ke node 213
x213.214	Kunjungan dari node 213 ke node 214
x213.232	Kunjungan dari node 213 ke node 232
x214.213	Kunjungan dari node 214 ke node 213
x214.215	Kunjungan dari node 214 ke node 215
x214.232	Kunjungan dari node 214 ke node 232
x232.214	Kunjungan dari node 232 ke node 214
x215.216	Kunjungan dari node 215 ke node 216
x216.215	Kunjungan dari node 216 ke node 215
x216.217	Kunjungan dari node 216 ke node 217
x217.216	Kunjungan dari node 217 ke node 216
x217.218	Kunjungan dari node 217 ke node 218
x218.217	Kunjungan dari node 218 ke node 217
x218.162	Kunjungan dari node 218 ke node 162
x219.220	Kunjungan dari node 219 ke node 220
x220.219	Kunjungan dari node 220 ke node 219
x220.221	Kunjungan dari node 220 ke node 221
x221.220	Kunjungan dari node 221 ke node 220

x221.222	Kunjungan dari node 221 ke node 222
x222.221	Kunjungan dari node 222 ke node 221
x222.223	Kunjungan dari node 222 ke node 223
x222.228	Kunjungan dari node 222 ke node 228
x223.222	Kunjungan dari node 223 ke node 222
x223.224	Kunjungan dari node 223 ke node 224
x223.228	Kunjungan dari node 223 ke node 228
x228.223	Kunjungan dari node 228 ke node 223
x224.225	Kunjungan dari node 224 ke node 225
x225.224	Kunjungan dari node 225 ke node 224
x225.226	Kunjungan dari node 225 ke node 226
x226.225	Kunjungan dari node 226 ke node 225
x226.227	Kunjungan dari node 226 ke node 227
x227.226	Kunjungan dari node 227 ke node 226
x226.227	Kunjungan dari node 226 ke node 227
x228.222	Kunjungan dari node 228 ke node 222
x228.229	Kunjungan dari node 228 ke node 229
x229.228	Kunjungan dari node 229 ke node 228
x229.230	Kunjungan dari node 229 ke node 230
x230.229	Kunjungan dari node 230 ke node 229
x230.231	Kunjungan dari node 230 ke node 231
x231.162	Kunjungan dari node 231 ke node 162
x162.231	Kunjungan dari node 162 ke node 231
x231.162	Kunjungan dari node 231 ke node 162
x232.233	Kunjungan dari node 232 ke node 233
x233.234	Kunjungan dari node 233 ke node 234
x234.210	Kunjungan dari node 234 ke node 210

x210.234	Kunjungan dari node 210 ke node 234
x235.202	Kunjungan dari node 235 ke node 202
x235.209	Kunjungan dari node 235 ke node 209

(Halaman ini sengaja dikosongkan)

LAMPIRAN C PENGGAMBARAN NETWORK MODEL

Tabel C.1 Tabel Jarak Waktu dan Trayek

Node Awal	Node Akhir	Jarak (Meter)	Waktu (Menit)	Trayek
1	2	550	1	UBB
2	3	600	2	UBB
2	1	550	1	UBB
2	16	700	2	UBB
3	4	400	1	UBB
4	5	1000	2	UBB
5	6	400	1	UBB
6	7	400	1	UBB
7	8	350	1	UBB
8	9	700	2	UBB
9	10	450	1	UBB
10	11	500	1	UBB
11	12	350	1	UBB
12	13	450	1	UBB
13	14	350	1	UBB
14	13	350	1	UBB
14	15	800	2	UBB
15	14	800	2	UBB
15	2	350	1	UBB
16	17	400	1	UBB
17	18	600	3	UBB
17	2	500	1	UBB

18	17	600	3	UBB
18	19	550	2	UBB
18	17	600	3	UBB
19	18	550	3	UBB
19	20	300	2	UBB
19	18	550	3	UBB
20	21	600	1	UBB, M
20	19	280	2	UBB
21	22	300	1	UBB
22	23	300	1	UBB
22	89	500	1	UBB
23	24	350	1	UBB
23	27	260	1	UBB
24	25	500	1	UBB, M
25	26	400	1	UBB, M
25	147	550	2	UBB
26	27	650	2	UBB, M
26	23	350	1	UBB
27	28	500	1	UBB, M
27	145	500	1	M
28	29	1000	4	UBB
28	152	500	1	M
28	27	450	1	UBB, M

29	30	500	2	UBB
30	31	400	1	UBB
31	32	300	2	UBB
32	33	300	1	UBB, M
33	34	300	1	UBB, M
34	84	240	1	UBB, M
34	35	400	1	UBB
34	83	750	1	UBB, M
35	82	95	1	UBB
36	37	200	1	UBB
36	143	500	1	M
36	83	350	1	M
36	82	550	1	UBB
37	38	400	1	UBB
37	36	220	1	UBB
38	39	350	1	UBB
38	37	400	1	UBB
39	40	400	1	UBB,M
39	38	350	1	UBB
40	41	400	2	UBB, M
41	42	550	1	UBB, M
41	81	450	1	UBB, M

42	43	500	1	UBB
42	142	450	1	UBB, M
42	154	600	2	UBB, M
42	41	550	1	UBB, M
43	44	210	1	UBB
43	42	800	2	UBB, M
44	45	400	1	UBB
44	43	250	1	UBB
45	46	650	3	UBB
46	47	400	1	UBB
47	46	400	1	UBB
47	48	500	1	UBB
48	49	300	1	UBB
49	50	250	1	UBB
49	44	350	1	UBB
50	51	850	2	UBB
51	50	850	2	UBB
51	52	350	1	UBB, M
52	51	350	1	UBB, M
52	53	210	1	UBB, M
53	156	550	1	UBB, M

53	54	900	3	UBB, M
54	55	500	1	UBB
54	74	550	1	UBB
55	56	280	1	UBB
55	54	500	1	UBB
56	55	280	1	UBB
56	57	240	1	UBB
57	56	240	1	UBB
57	58	300	1	UBB
58	57	300	1	UBB
58	59	350	1	UBB
58	77	550	1	UBB
59	58	350	1	UBB
59	60	280	1	UBB
60	59	280	1	UBB
60	61	300	1	UBB
61	62	500	1	UBB
62	63	650	1	UBB, WB
63	64	650	2	UBB, WB
64	65	250	1	UBB, WB
64	164	600	1	UBB
64	165	500	1	UBB
65	66	300	1	UBB
65	64	250	1	UBB, WB

66	65	300	1	UBB
66	67	350	1	UBB, WB
67	66	350	1	UBB, WB
67	68	500	1	UBB, WB
67	158	500	1	UBB, WB
68	69	500	1	UBB, WB
68	67	500	1	UBB, WB
69	68	500	1	UBB, WB
69	70	800	1	UBB, WB, S
70	71	450	1	UBB, WB, S
71	72	400	1	UBB, WB, S
72	71	400	1	UBB, WB, S
72	73	400	1	UBB
73	55	550	1	UBB, WB
73	69	500	1	UBB, WB
73	157	550	3	WB
73	72	400	1	UBB
74	75	500	1	UBB
75	74	500	1	UBB

75	76	500	1	UBB
76	57	700	2	UBB
76	58	350	1	UBB
77	78	550	2	UBB
78	77	550	2	UBB
78	79	550	2	UBB
79	78	550	2	UBB
79	80	400	1	UBB
80	79	400	1	UBB
80	48	350	1	UBB
81	39	650	1	UBB, M
81	143	400	1	UBB, M
82	34	500	2	UBB
82	35	95	1	UBB
83	33	650	1	UBB, M
83	36	350	1	M
84	85	500	1	UBB, M
85	86	500	1	UBB, M
85	28	600	2	UBB, M
86	87	500	1	UBB
87	86	500	1	UBB
87	88	500	1	UBB

88	22	650	1	UBB, M
89	90	500	1	UBB, M
89	23	300	1	UBB, M
90	91	350	1	UBB
90	89	500	1	UBB, M
91	20	350	1	UBB
92	93	500	1	M, U , S
93	94	650	1	M, U , S
93	92	550	2	M, U , S
94	95	500	1	M, U , S
95	96	550	1	M, U , S
95	134	500	1	M, S
96	97	500	1	M, U , S
97	98	550	3	U, S
97	104	350	1	M, U , S
98	99	500	1	S
98	132	500	1	U
98	133	400	1	U, S
99	98	500	1	S
99	100	500	1	S

100	99	500	1	S
100	101	500	1	S
101	100	500	1	S
101	102	500	2	S
102	103	500	1	S
102	101	500	2	S
103	72	300	1	WB, S
103	157	500	2	WB, S
104	105	650	3	U
105	106	500	1	U
106	105	500	1	U
106	107	500	1	U
107	106	500	1	U
107	108	500	1	U
108	107	500	1	U
108	109	500	1	U
108	132	600	2	U
109	110	500	1	U
109	108	500	1	U
110	109	500	1	U
110	111	500	3	U
111	110	500	3	U
111	112	500	2	U
112	111	500	2	U
112	113	500	1	U
113	112	500	1	U
113	114	500	1	U

114	113	500	1	U
114	115	500	1	U
115	114	500	1	U
115	116	500	1	U
116	115	500	1	U
116	117	500	1	U
117	116	500	1	U
117	118	500	1	U
118	119	550	2	U
118	128	500	2	U
119	120	500	1	U
120	121	500	1	U
121	120	500	1	U
121	122	500	1	U
121	126	500	1	U
122	121	500	1	U
122	123	500	1	U
123	122	500	1	U
123	124	500	1	U
124	123	500	1	U
124	125	600	2	U
125	124	600	2	U
126	121	500	1	U
126	127	500	1	U
127	118	400	1	U
128	129	500	2	U
129	128	500	2	U

129	130	550	2	U
130	129	550	2	U
130	131	550	1	U
131	130	550	1	U
131	112	550	1	U
132	108	600	1	U
132	98	600	2	U
133	93	900	2	M, U , S
134	135	550	2	M
134	96	300	1	M, S
135	136	500	2	M
135	134	550	2	M
136	137	500	2	M
136	135	500	2	M
137	138	500	1	M
137	52	450	2	UBB, M
137	136	500	2	M
138	139	450	2	M
138	137	500	1	M
139	140	500	2	M
139	138	450	2	M
140	141	500	3	M
141	142	400	2	M
142	42	800	5	M, UBB
142	141	400	2	M

142	43	140	1	M, UBB
143	144	400	1	M
143	39	300	1	UBB, M
144	33	600	1	M
145	146	600	2	M
146	25	450	1	M
147	148	500	1	M, UBB
147	21	500	1	M
148	149	500	1	M
148	22	500	1	M
149	148	500	1	M
149	150	500	1	M
150	149	500	1	M
150	151	400	1	M
151	150	400	1	M
152	153	500	2	M
153	34	750	2	M
154	155	600	1	M
154	142	210	1	M
154	43	300	1	UBB. M
155	139	500	3	M
156	52	500	1	M
156	137	600	2	M
156	136	600	2	M

157	73	550	3	WB
158	67	500	1	WB
158	159	500	2	WB
159	158	500	2	WB
159	160	500	1	WB
160	159	500	1	WB
160	161	500	1	WB
161	160	500	1	WB
161	162	500	1	WB
162	161	500	1	WB
162	163	500	1	WB
162	219	500	1	WB, TWM
163	164	500	1	WB, TWM
163	162	500	1	WB, TWM
163	218	500	1	TWM
164	163	500	1	WB, TWM
164	64	600	1	WB, UBB
165	166	500	3	WB
166	165	500	3	WB
166	167	500	2	WB
167	166	500	2	WB
167	168	500	2	WB
168	169	500	1	WB
168	192	500	1	WB

169	168	500	2	WB
169	170	500	1	WB
170	169	500	1	WB
170	171	500	1	WB
171	170	500	1	WB
171	172	500	1	WB
172	173	500	2	WB
173	172	500	2	WB
173	174	500	1	WB
174	173	500	1	WB
174	175	500	1	WB
175	174	500	1	WB
175	176	500	2	WB
175	188	500	1	WB
176	175	500	1	WB
176	177	500	3	WB
177	176	500	3	WB
177	178	500	2	WB
178	177	500	2	WB
178	179	500	2	WB
179	178	500	2	WB
179	180	500	1	WB
180	179	500	1	WB
180	181	500	1	WB
181	180	500	1	WB
181	182	500	2	WB
182	181	500	2	WB

182	183	500	2	WB
183	182	500	2	WB
183	184	550	2	WB
184	183	550	2	WB
184	185	450	2	WB
185	184	450	2	WB
185	186	500	2	WB
186	185	500	2	WB
186	187	450	2	WB
187	186	450	2	WB
188	175	500	1	WB
188	189	500	1	WB
189	190	500	1	WB
190	191	450	1	WB
191	171	350	1	WB
192	63	350	1	WB
192	168	500	1	WB
193	194	500	2	TWM
194	193	500	2	TWM
194	195	500	2	TWM
195	194	500	2	TWM
195	196	500	2	TWM
196	195	500	2	TWM
196	197	500	2	TWM
197	196	500	2	TWM
197	198	500	1	TWM
198	197	500	1	TWM

198	199	500	1	TWM
199	198	500	1	TWM
199	200	500	1	TWM
200	199	500	1	TWM
200	201	500	1	TWM
201	200	500	1	TWM
201	202	500	1	TWM
202	201	500	1	TWM
202	203	500	2	TWM
203	204	500	1	TWM
204	203	500	1	TWM
204	205	500	1	TWM
205	204	500	1	TWM
205	206	500	1	TWM
206	205	500	1	TWM
206	207	500	1	TWM
207	206	500	1	TWM
207	208	500	1	TWM
208	207	500	1	TWM
208	209	500	1	TWM
209	208	500	1	TWM
209	210	500	1	TWM
209	235	550	2	TWM
235	209	550	2	TWM
210	211	500	1	TWM
210	234	650	2	TWM
234	210	650	2	TWM

211	233	350	1	TWM
211	212	500	1	TWM
212	213	500	1	TWM
213	214	500	1	TWM
213	232	500	3	TWM
214	213	500	1	TWM
214	215	500	1	TWM
214	232	500	1	TWM
232	214	500	1	TWM
215	216	500	1	TWM
216	215	500	1	TWM
216	217	500	1	TWM
217	216	500	1	TWM
217	218	500	1	TWM
218	217	500	1	TWM
218	162	700	1	TWM
219	220	500	2	TWM
220	219	500	2	TWM
220	221	500	1	TWM
221	220	500	1	TWM
221	222	500	1	TWM
222	221	500	1	TWM
222	223	500	1	TWM
222	228	150	1	TWM
223	222	500	1	TWM
223	224	500	1	TWM
223	228	500	1	TWM

228	223	500	1	TWM
224	225	500	1	TWM
225	224	500	1	TWM
225	226	500	1	TWM
226	225	500	1	TWM
226	227	450	1	TWM
227	226	450	1	TWM
226	227	450	1	TWM
228	222	150	1	TWM
228	229	500	3	TWM
229	228	500	3	TWM
229	230	500	1	TWM
230	229	500	1	TWM
230	231	500	1	TWM
231	162	300	1	TWM
162	231	300	1	TWM
231	162	400	1	TWM
232	233	500	1	TWM
233	234	650	2	TWM
234	210	650	2	TWM
210	234	650	2	TWM
235	202	500	2	TWM
235	209	550	2	TWM

Tabel C.2 Tabel Skor

Node	Skor	Node	Skor	Node	Skor	Node	Skor
1	32	**61**	32	**121**	124	**181**	71
2	165	**62**	32	**122**	124	**182**	71
3	32	**63**	104	**123**	124	**183**	71
4	32	**64**	104	**124**	124	**184**	71
5	32	**65**	32	**125**	124	**185**	71
6	32	**66**	32	**126**	124	**186**	71
7	32	**67**	104	**127**	124	**187**	71
8	32	**68**	104	**128**	124	**188**	71
9	32	**69**	104	**129**	124	**189**	71
10	32	**70**	190	**130**	124	**190**	71
11	32	**71**	190	**131**	124	**191**	71
12	32	**72**	190	**132**	124	**192**	71
13	32	**73**	104	**133**	124	**193**	18
14	32	**74**	32	**134**	133	**194**	18
15	32	**75**	32	**135**	133	**195**	18
16	165	**76**	32	**136**	133	**196**	18
17	32	**77**	32	**137**	133	**197**	18
18	32	**78**	32	**138**	133	**198**	18
19	32	**79**	32	**139**	133	**199**	18
20	32	**80**	32	**140**	133	**200**	18
21	32	**81**	165	**141**	133	**201**	18
22	165	**82**	32	**142**	133	**202**	18
23	165	**83**	165	**143**	133	**203**	18
24	32	**84**	165	**144**	133	**204**	18

25	165
26	32
27	165
28	165
29	32
30	32
31	32
32	32
33	165
34	165
35	32
36	165
37	32
38	32
39	165
40	165
41	165
42	165
43	32
44	32
45	32
46	32
47	32
48	32
49	32
50	32
51	32

85	165
86	32
87	32
88	32
89	32
90	32
91	32
92	343
93	343
94	343
95	343
96	343
97	343
98	210
99	86
100	86
101	86
102	86
103	157
104	343
105	124
106	124
107	124
108	124
109	124
110	124
111	124

145	133
146	133
147	133
148	133
149	133
150	133
151	133
152	133
153	133
154	133
155	133
156	133
157	71
158	71
159	71
160	71
161	71
162	89
163	89
164	71
165	71
166	71
167	71
168	71
169	71
170	71
171	71

205	18
206	18
207	18
208	18
209	18
210	18
211	18
212	18
213	18
214	18
215	18
216	18
217	18
218	18
219	18
220	18
221	18
222	18
223	18
224	18
225	18
226	18
227	18
228	18
229	18
230	18
231	18

52	165
53	165
54	32
55	32
56	32
57	32
58	32
59	32
60	32

112	124
113	124
114	124
115	124
116	124
117	124
118	124
119	124
120	124

172	71
173	71
174	71
175	71
176	71
177	71
178	71
179	71
180	71

232	18
233	18
234	18
235	18

(Halaman ini sengaja dikosongkan)

LAMPIRAN D KODE DIJKSTRA

```
/*
 * To change this license header, choose
License Headers in Project Properties.
 * To change this template file, choose Tools
| Templates
 * and open the template in the editor.
 */
// A Java program for Dijkstra's single source
shortest path algorithm.
// The program is for adjacency matrix
representation of the graph

import java.util.*;
import java.lang.*;
import java.io.*;

/**
 *
 * @author Dhamar
 */
class ShortestPath {

    // A utility function to find the vertex
with minimum distance value,
    // from the set of vertices not yet
included in shortest path tree
    static final int V = 235;

    int minDistance(int dist[], Boolean
sptSet[]) {
        // Initialize min value
        int min = Integer.MAX_VALUE, min_index
= -1;

        for (int v = 0; v < V; v++) {
            if (sptSet[v] == false && dist[v]
<= min) {
                min = dist[v];
                min_index = v;
```

```
            }
        }

        return min_index;
    }

    // A utility function to print the
constructed distance array
    void printSolution(int dist[], int n) {
        //System.out.println("Vertex
Distance from Source");
        for (int i = 0; i < V; i++) {
            //System.out.println(i + " \t\t "
+ dist[i]);
            System.out.print(dist[i] + ",");
        }
    }

    // Funtion that implements Dijkstra's
single source shortest path
    // algorithm for a graph represented using
adjacency matrix
    // representation
    void dijkstra(int graph[][], int src) {
        int dist[] = new int[V]; // The output
array. dist[i] will hold
        // the shortest distance from src to i

        // sptSet[i] will true if vertex i is
included in shortest
        // path tree or shortest distance from
src to i is finalized
        Boolean sptSet[] = new Boolean[V];

        // Initialize all distances as
INFINITE and stpSet[] as false
        for (int i = 0; i < V; i++) {
            dist[i] = Integer.MAX_VALUE;
            sptSet[i] = false;
        }
```

```
        // Distance of source vertex from
itself is always 0
        dist[src] = 0;

        // Find shortest path for all vertices
        for (int count = 0; count < V - 1;
count++) {
            // Pick the minimum distance
vertex from the set of vertices
            // not yet processed. u is always
equal to src in first
            // iteration.
            int u = minDistance(dist, sptSet);

            // Mark the picked vertex as
processed
            sptSet[u] = true;

            // Update dist value of the
adjacent vertices of the
            // picked vertex.
            for (int v = 0; v < V; v++) //
Update dist[v] only if is not in sptSet, there
is an
            // edge from u to v, and total
weight of path from src to
            // v through u is smaller than
current value of dist[v]
            {
                if (!sptSet[v] && graph[u][v]
!= 0
                        && dist[u] !=
Integer.MAX_VALUE
                        && dist[u] +
graph[u][v] < dist[v]) {
                    dist[v] = dist[u] +
graph[u][v];
                }
            }
```

```
        }

        // print the constructed distance
array
        printSolution(dist, V);
    }
    static int[][] intArray = new
int[235][235];

    // Driver method
    public static void main(String[] args)
throws FileNotFoundException, IOException {
        String Array;
        BufferedReader bufRdr = new
BufferedReader(new
FileReader("src/tes1.csv"));
        ArrayList<String[]> lines = new
ArrayList<String[]>();
        while ((Array = bufRdr.readLine()) !=
null) {
            lines.add(Array.split(","));
        }

        //Convert our list to a String array
        String[][] array = new
String[lines.size()][0];
        lines.toArray(array);

        //Convert String array to Integer
array
        for (int i = 0; i < 235; i++) {
            for (int j = 0; j < 235; j++) {
                intArray[i][j] =
Integer.parseInt(array[i][j]);
            }
        }

        /* Let us create the example graph
discussed above */
        ShortestPath t = new ShortestPath();
```

```
        for (int i = 0; i < 235; i++) {
            //for (int j = 0; j < 30; j++) {
            //int graph[][] = new int[][]
{{intArray[i][j]}};

            t.dijkstra(intArray, i);
            System.out.println();
        }
    }
}
```

Kode D.1 Kode Dijkstra

(Halaman ini sengaja dikosongkan)

LAMPIRAN E KODE LOCAL SEARCH

```
/*
 * To change this license header, choose
License Headers in Project Properties.
 * To change this template file, choose Tools
| Templates
 * and open the template in the editor.
 */
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

/**
 *
 * @author Dhamar
 */
public class array {

    static int nAwal = 1;
    static int nAkir = 125;
    static int jmlNode = 235; //Sementara
menganggap node awal pasti 1 dan node akhir,
N, adalah jumlah node
    static int[][] untukjalur = new
int[jmlNode][jmlNode];
    static int[] untukscore = new
int[jmlNode];
    static int[] reinforce = new int[3];

    static ArrayList<ArrayList<Integer>>
feasSolution = new
ArrayList<ArrayList<Integer>>();
    static ArrayList<Integer>
feasSolution_Jarak = new ArrayList<Integer>();
    static ArrayList<Integer>
feasSolution_Score = new ArrayList<Integer>();
```

```
    static ArrayList<Integer>
pemilihanHasilWaktu = new
ArrayList<Integer>();
    static ArrayList<Integer>
pemilihanHasilScore = new
ArrayList<Integer>();
    static ArrayList<ArrayList<Integer>>
pemilihanHasil = new
ArrayList<ArrayList<Integer>>();
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
throws FileNotFoundException, IOException {

        //KONVERSI ARRAY 2D UNTUK NILAI ARC
DARI FILE .CSV
        String ini;
        BufferedReader jalur = new
BufferedReader(new
FileReader("src/generatedpath.csv"));

        ArrayList<String[]> lines = new
ArrayList<String[]>();
        while ((ini = jalur.readLine()) !=
null) {
            lines.add(ini.split(","));
        }

        //Convert list to a String array
        String[][] array = new
String[lines.size()][0];
        lines.toArray(array);

        //Convert String array to Integer
array
        for (int i = 0; i < array.length; i++)
{
            for (int j = 0; j < array.length;
j++) {
```

```
                untukjalur[i][j] =
Integer.parseInt(array[i][j]);
            }
        }

        //KONVERSI ARRAY 1D UNTUK SCORE TIAP
NODE DARI FILE .CSV
        BufferedReader skor = new
BufferedReader(new
FileReader("src/skor.csv"));

        String itu;
        int o = 0;
        while ((itu = skor.readLine()) !=
null) {
            untukscore[o] =
Integer.parseInt(itu);
            o++;
        }

        int tMax = 120; //Menetapkan nilai
tMax (dalam menit)
        int maxLoop = 1000000; //Menetapkan
batas iterasi (m8x loop)
        int count = 0; //Mengatur count = 0
        int loop = 0; //Mengatur loop = 0

        do {
            //Produce a random number, Rn ,
between [1,N]
            int Rn = new
Random().nextInt(jmlNode - 2) + 1;

            //Produce a list, Rl , between
[1,N] randomly
            ArrayList<Integer> listRl = new
ArrayList();
            ArrayList<Integer> listRl_rand =
new ArrayList(); //Array yang digunakan untuk
```

```
random node diantara node awal (1) dan akhir
(N)
            for (int i = 1; i < jmlNode; i++)
{ //Fungsi looping mengambil node 2 sampai
jumlah Node / Node Akhir (N)
                if (i == nAwal || i == nAkir)
{
                    continue;
                }
                listRl_rand.add(i); //Meng-
generate node (angka) 2 sampai N-1
            }
            Collections.shuffle(listRl_rand);
//Mengacak node (angka) 2 sampai N-1

            listRl.add(nAwal);
//Merepresentasikan node awal yaitu 1
            for (int i = 0; i <
listRl_rand.size(); i++) {

listRl.add(listRl_rand.get(i));
//Merepresentasikan node 2 ke N-1 yang telah
dibuat acak (random)
            }
            listRl.add(nAkir);
//Merepresentasikan node akhir, N

            //System.out.println(listRl);
//cetak list Rl secara random
            //Generate a sub list, Rs, by
taking the first Rn part of the random list Rl
            ArrayList<Integer> subListRs = new
ArrayList();
            subListRs.add(listRl.get(0));
            for (int i = 1; i < (Rn + 1); i++)
{
                subListRs.add(listRl.get(i));
            }

subListRs.add(listRl.get(listRl.size() - 1));
```

```
            for (int i = 0; i <
subListRs.size(); i++) { //looping mengambil
nilai index pada array cobaList
                if
(hitungWaktuTempuh(subListRs) < tMax &&
!feasSolution.contains(subListRs)) {

feasSolution.add(subListRs);
                }
            }

            loop++; //loop=loop+1

        } while (loop <= maxLoop);

        for (int i = 0; i <
feasSolution.size(); i++) {
//
System.out.println(feasSolution.get(i) + "(" +
hitungWaktuTempuh(feasSolution.get(i)) + ", "
+ hitungScore(feasSolution.get(i)) + ")");

feasSolution_Jarak.add(hitungWaktuTempuh(feasS
olution.get(i)));

feasSolution_Score.add(hitungScore(feasSolutio
n.get(i)));
        }

        for (int i = 0; i <
feasSolution.size(); i++) {

System.out.println(feasSolution.get(i) + "(" +
feasSolution_Jarak.get(i) + ", " +
feasSolution_Score.get(i) + ")");
        }
        System.out.println("---Hasil---");
        System.out.println("total jumlah skor
: " + totalScore());
```

```
        System.out.println("jumlah jalur yang
feasible : " + feasSolution.size());
        System.out.println("\n---Mulai
Algoritma---");

        int size = 30;
        do {

pemilihanHasil.add(feasSolution.get(rouletteWh
eelSelection()));
        } while (pemilihanHasil.size() <
size);
        for (int i = 0; i <
pemilihanHasil.size(); i++) {

pemilihanHasilWaktu.add(hitungWaktuTempuh(pemi
lihanHasil.get(i)));

pemilihanHasilScore.add(hitungScore(pemilihanH
asil.get(i)));

System.out.println(pemilihanHasil.get(i) + "("
+ pemilihanHasilWaktu.get(i) + ", " +
pemilihanHasilScore.get(i) + ")");
        }
        System.out.println("--Mulai Great
Deluge--");
        for (int i = 0; i <
pemilihanHasil.size(); i++) {
            //looping untuk mengecek semua isi
pemilihanHasil
            int counterGD = 1;
            int maxGDLoop = 100;
            do {
                //int randCase = new
Random().nextInt(3) + 1;
                int randCase =
reinforcementLearning();
                switch (randCase) {
                    case 1:
```

```
                            int swap1 = new
Random().nextInt(pemilihanHasil.get(i).size()
- 2) + 1;
                            int swap2 = new
Random().nextInt(pemilihanHasil.get(i).size()
- 2) + 1;

Collections.swap(pemilihanHasil.get(i), swap1,
swap2);
                            int currentJarakSwap =
hitungWaktuTempuh(pemilihanHasil.get(i));
                            int currentScoreSwap =
hitungScore(pemilihanHasil.get(i));
                            if (currentJarakSwap <
pemilihanHasilWaktu.get(i) ) { //currentjarak
lebih kecil atau currentscore lebih besar

pemilihanHasilWaktu.set(i, currentJarakSwap);

pemilihanHasilScore.set(i, currentScoreSwap);
                                //
System.out.println("index yang di swap:" +
swap1 + " dan " + swap2);
                                //
System.out.println("swap2:" + swap2);
                                //
System.out.println(pemilihanHasil.get(i) + "("
+ currentJarakSwap + ", " + currentScoreSwap +
")");
                                counterGD = 1;
                                reinforce[0]++;
                            } else {
                                reinforce[0]--;

Collections.swap(pemilihanHasil.get(i), swap2,
swap1);

System.out.println(pemilihanHasil.get(i) + "("
+ currentJarakSwap + ", " + currentScoreSwap +
")");
```

```
                        }
                        break;
                    case 2:

//System.out.println("---delete---");
                        int delRandom = new
Random().nextInt(pemilihanHasil.get(i).size()
- 2) + 1;
                        if
(pemilihanHasil.get(i).size() > 3) {
                            int temp =
pemilihanHasil.get(i).get(delRandom);

pemilihanHasil.get(i).remove(delRandom);
                            int
currentJarakDel =
hitungWaktuTempuh(pemilihanHasil.get(i));
                            int
currentScoreDel =
hitungScore(pemilihanHasil.get(i));

//System.out.println(pemilihanHasil.get(i));
                            if
(currentJarakDel <= pemilihanHasilWaktu.get(i)
&& currentScoreDel >= 1.05 *
pemilihanHasilScore.get(i)) {

pemilihanHasilWaktu.set(i, currentJarakDel);

pemilihanHasilScore.set(i, currentScoreDel);
                                //
System.out.println(pemilihanHasil.get(i));
                                counterGD = 1;

reinforce[1]++;
                            } else {
                                reinforce[1]--
;

pemilihanHasil.get(i).add(delRandom, temp);
```

```
                            }
                            break;
                        }
                    case 3:
                        int rndInsert = new
Random().nextInt(pemilihanHasil.get(i).size()
- 2) + 1;
                        int rndNilaiInsert =
new Random().nextInt(jmlNode) + 1;
                        if (rndNilaiInsert ==
nAwal || rndNilaiInsert == nAkir ||
pemilihanHasil.get(i).contains(rndNilaiInsert)
) {
                            continue;
                        }

pemilihanHasil.get(i).add(rndInsert,
rndNilaiInsert);

                        int
currentWaktuTempuhInsert =
hitungWaktuTempuh(pemilihanHasil.get(i));
                        int currentScoreInsert
= hitungScore(pemilihanHasil.get(i));
                        if
(currentWaktuTempuhInsert <=
pemilihanHasilWaktu.get(i) &&
currentScoreInsert >= 1.05 *
pemilihanHasilScore.get(i)) { //currentjarak
lebih kecil atau currentscore lebih besar

pemilihanHasilWaktu.set(i,
currentWaktuTempuhInsert);

pemilihanHasilScore.set(i,
currentScoreInsert);
                            counterGD = 1;
                            reinforce[2]++;
                        } else {
                            reinforce[2]--;
```

```
pemilihanHasil.get(i).remove(rndInsert);
                        }
                        break;
                }
                counterGD++;

            } while (counterGD <= maxGDLoop);

        }
        for (int i = 0; i <
pemilihanHasil.size(); i++) {
            System.out.println("Solusi ke = "
+ (i + 1));

System.out.println(pemilihanHasil.get(i) + "("
+ pemilihanHasilWaktu.get(i) + ", " +
pemilihanHasilScore.get(i) + ")");
        }
        int bestIndex = 0;
        int bestScore =
pemilihanHasilScore.get(0);
        for (int i = 1; i <
pemilihanHasil.size(); i++) {
            if (pemilihanHasilScore.get(i) >=
bestScore) {
                bestIndex = i;
                bestScore =
pemilihanHasilScore.get(i);
            }
        }
        System.out.println();
        System.out.println("Solusi terbaik :"
+ pemilihanHasil.get(bestIndex) + "(" +
pemilihanHasilWaktu.get(bestIndex) + "," +
pemilihanHasilScore.get(bestIndex) + ")");
    }
```

```
    public static int
hitungWaktuTempuh(ArrayList<Integer>
varListRl) { //Perhitungan waktu tempuh
        int sum = 0; //Inisiasi nilai awal sum
adalah 0 untuk penjumlahan
        for (int i = 0; i < varListRl.size() -
1; i++) { //Loop untuk mengambil isi dari isi
list Rl
            sum = sum +
untukjalur[varListRl.get(i) -
1][varListRl.get(i + 1) - 1]; //Menjumlah
setiap waktu tempuh pada 1 list Rl
        }
        return sum;
    }

    public static int
hitungScore(ArrayList<Integer> varListRl) {
//Perhitungan waktu tempuh
        int sum = 0; //Inisiasi nilai awal sum
adalah 0 untuk penjumlahan
        for (int i = 0; i < varListRl.size();
i++) { //Loop untuk mengambil isi dari isi
list Rl
            sum = sum +
untukscore[varListRl.get(i) - 1]; //Menjumlah
setiap waktu tempuh pada 1 list Rl
        }
        return sum;
    }

    public static int totalScore() {
//Perhitungan waktu tempuh
        int sum = 0; //Inisiasi nilai awal sum
adalah 0 untuk penjumlahan
        for (int i = 0; i <
feasSolution_Score.size(); i++) { //Loop untuk
mengambil isi dari isi list Rl
```

```
            sum = sum +
feasSolution_Score.get(i); //Menjumlah setiap
waktu tempuh pada 1 list Rl
        }
        return sum;
    }

    public static int rouletteWheelSelection()
{ //Perhitungan waktu tempuh
        double r = new Random().nextDouble();
        int totScore = totalScore();
        int k = 0;
        double sum = (double)
feasSolution_Score.get(k) / totScore;
        while (sum < r) {
            k++;
            sum = sum + (double)
feasSolution_Score.get(k) / totScore;
//            System.out.println(sum);
//            System.out.println(totScore);
        }
//        System.out.println("index ke :" +
k);
//
System.out.println(feasSolution_Score.size());
        return k;
    }

    public static int reinforcementLearning()
{
        int max = reinforce[0];
        int id = 0;
        if (reinforce[1] > max) {
            id = 1;
            max = reinforce[1];
        } else if (reinforce[2] > max) {
            id = 2;
            max = reinforce[2];
        }
```

```
        if (reinforce[0] == reinforce[1] &&
reinforce[1] == reinforce[2]) {
            id = new Random().nextInt(2);
        }

        return id;
    }
}
```

Kode E.1 Kode Local Search

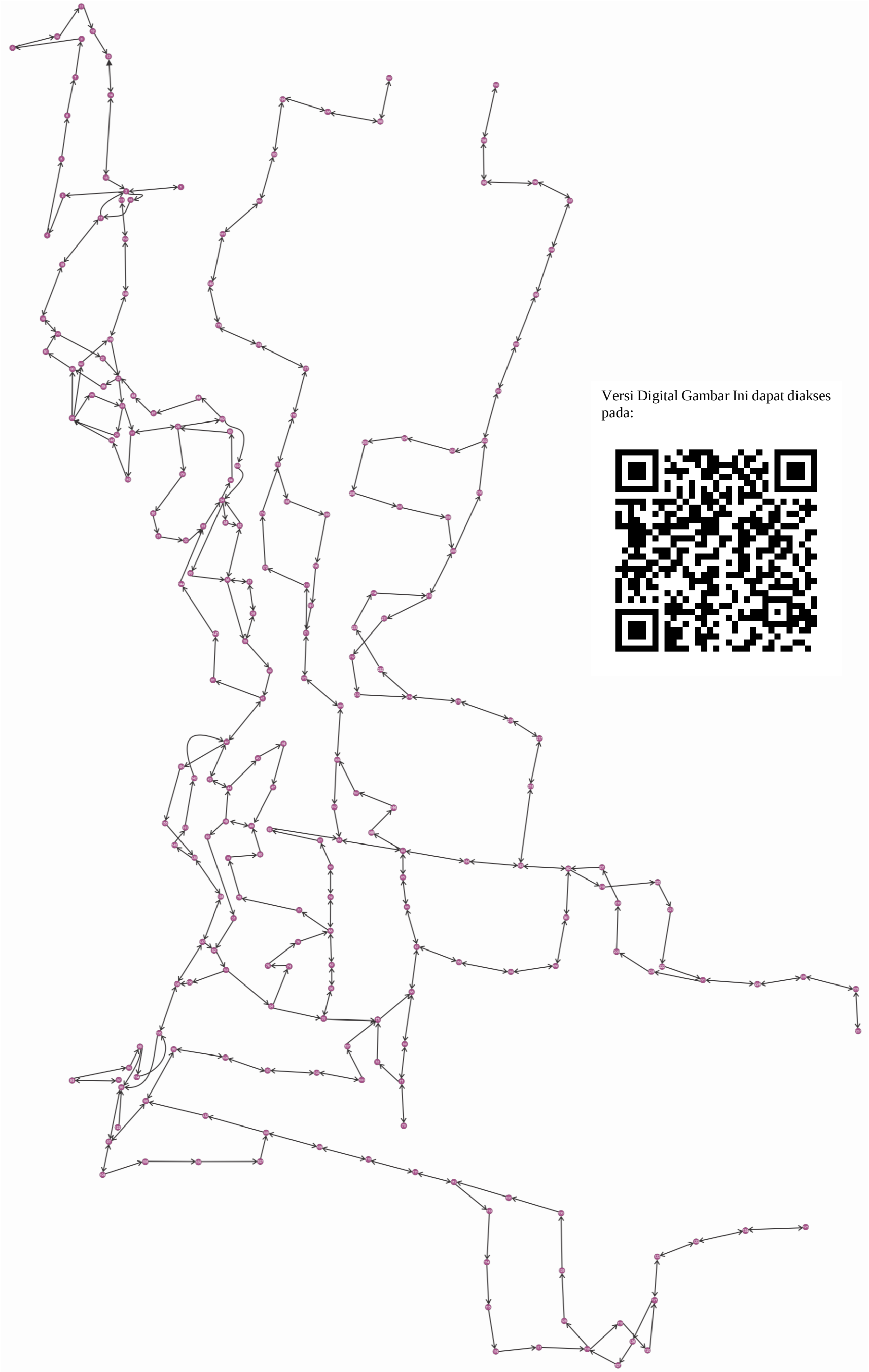
(Halaman ini sengaja dikosongkan)

LAMPIRAN F PETA NODE

Gambar F.1 Trayek dengan Node

(Halaman ini sengaja dikosongkan)

LAMPIRAN G NETWORK MODEL



Versi Digital Gambar Ini dapat diakses pada:

Gambar G.1 Network Model