TUGAS AKHIR – KS 141501

ADVANCED TRAVELER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

ADVANCED TRAVELER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GENETIC ALGORITHM (CASE STUDY: ANGKOT'S ROUTES IN SURABAYA)

I WAYAN ANGGA KUSUMA YOGA
NRP 5213 100 006

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 TRAVELER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

I WAYAN ANGGA KUSUMA YOGA
NRP 5213 100 006

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 TRAVELER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GENETIC ALGORITHM (CASE STUDY: ANGKOT'S ROUTES IN SURABAYA)

I WAYAN ANGGA KUSUMA YOGA
NRP 5213 100 006

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

DEPARTMENT OF INFORMATION SYSTEM
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

LEMBAR PERSETUJUAN

ADVANCED TRAVELER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (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:

I Wayan Angga Kusuma Yoga
NRP 5213 100 006

Disetujui Tim Penguji : Tanggal Ujian : 6 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)**

LEMBAR PENGESAHAN

ADVANCED TRAVELER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (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:

I Wayan Angga Kusuma Yoga
NRP 5213 100 006

Surabaya, Juli 2017

KEPALA
DEPARTEMEN SISTEM INFORMASI

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

ADVANCED TRAVELER INFORMATION SYSTEMS: OPTIMASI RENCANA PERJALANAN DENGAN MODEL ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (STUDI KASUS: TRAYEK ANGKOT SURABAYA)

Nama Mahasiswa	**: I Wayan Angga Kusuma Yoga**
NRP	**: 5213 100 006**
Departemen	**: Sistem Informasi FTIF-ITS**
Pembimbing 1	**: Wiwik Anggraeni, S.Si., M.Kom.**
Pembimbing 2	**: Ahmad Mukhlason, S.Kom, M.Sc, Ph.D**

ABSTRAK

Kemacetan lalu lintas merupakan permasalahan umum yang sering terjadi pada kota-kota besar. Salah satu faktor utamanya adalah peningkatan pertumbuhan kendaraan bermotor yang tidak sebanding dengan pertumbuhan jalan. Banyak kerugian yang ditimbulkan oleh kemacetan lalu lintas dan hal tersebut berujung pada terhambatnya pertumbuhan ekonomi. Untuk mengurangi kemacetan lalu lintas, Pemerintah Surabaya sendiri telah melakukan berbagai upaya, salah satunya adalah rencana peningkatan penggunaan transportasi umum termasuk revitalisasi angkot. Salah satu cara menarik minat masyarakat yang saat ini sedang dikembangkan oleh Pemerintah Surabaya adalah penerapan sistem manajemen transportasi yang handal yaitu Surabaya Intelligent Transport Systems (SITS). Untuk mendukung hal tersebut, maka dalam penelitian ini dilakukan optimasi rute perjalanan mikrolet/angkot di Surabaya.

Pembuatan model dilakukan berdasarkan enam trayek angkot yang telah dipilih yaitu kode trayek DP, I, Q, TV2, Z, dan Z1 beserta jumlah angkot masing-masing trayek yang didapat melalui website resmi Dinas Perhubungan Kota Surabaya. Data waktu tempuh didapatkan melalui Google Maps. Berdasarkan data-data tersebut, dibangun model Orienteering

Problem dan selanjutnya dibuat solusi berdasarkan model tersebut menggunakan Genetic Algorithm. Dalam pembuatan solusi dari model, perlu dibuat data set terlebih dahulu untuk membantu proses pencarian solusi yang layak. Terdapat dua dataset yaitu data set waktu tempuh dan skor dimana masing-masing berbentuk matriks. Dalam pembuatan data set waktu tempuh digunakan Algoritma Dijkstra. Dalam penelitian ini dihasilkan network model dari enam trayek tersebut, model matematis, algoritma pembuatan solusi model dengan bahasa pemrograman Java, dan data set yang dapat digunakan untuk pengembangan selanjutnya.

Hasil dari pembuatan solusi dari model OP menunjukkan bahwa Orienteering Problem dapat digunakan untuk memodelkan permasalahan optimasi jalur angkot Kota Surabaya. Genetic Algorithm juga dapat menyelesaikan permasalahan OP dengan hasil fitness yang baik dimana penetapan jumlah generasi, jumlah populasi, probabilitas perkawinan silang, dan probabilitas mutasi juga sangat berpengaruh dalam memberikan hasil yang optimum dengan nilai fitness yang lebih baik. Diharapkan dengan adanya penelitian ini dapat membantu pengembangan Surabaya Intelligent Transport System dan terlaksananya revitalisasi angkot di Kota Surabaya.

Kata Kunci: optimasi, model, transportasi umum, orienteering problem, genetic algorithm

ADVANCED TRAVELLER INFORMATION SYSTEMS: ITINERARY OPTIMISATION USING ORIENTEERING PROBLEM MODEL AND GENETIC ALGORITHM (CASE STUDY: ANGKOT'S ROUTES IN SURABAYA)

Name	**: I Wayan Angga Kusuma Yoga**
NRP	**: 5213 100 006**
Department	**: Information Systems FTIF -ITS**
Supervisor 1	**: Wiwik Anggraeni, S.Si., M.Kom.**
Supervisor 2	**: Ahmad Mukhlason, S.Kom, M.Sc, Ph.D**

ABSTRACT

Traffic congestion is a common problem in major cities. One of the main cause is the uneven growth between motorized vehicle and roads to access. Many losses are caused by traffic congestion and this leads to economic growth inhibition. To reduce traffic congestion, the Government of Surabaya has made various efforts, one of which is the plan to increase the use of public transportation including the revitalization of angkot. One of the interesting ways for the community currently being developed by the Surabaya Government is the implementation of a reliable transportation management system namely Surabaya Intelligent Transport Systems (SITS). To support it, this research will take the topic of optimization angkot's travel route in Surabaya.

The model is based on six existing route that was chosen, with route code DP, I, Q, TV2, Z, and Z1 and the total amount of angkot deployed for every route, that were extracted from the Dept. of Transportation's website and travel time, which were obtained from Google Maps. Based on those data, an Orienteering Problem model was made, and from that model, a solution was discovered using Genetic Algorithm. On the conception of solution based on the model, a dataset need to be created to help discover a feasible solution. There are 2

datasets, travel time and score, both in a matrix form, and for the travel time dataset, dijkstra algorithm was used. In this research, a network model of the 6 routes, a mathematical model, an algorithm for the model's solution in Java programming language, and dataset were created, and can be used for further research.
The result of the solution conception based on the OP model shows that OP can be used to model the problems of Surabaya's angkot route optimization. GA can also be used to solve OP with a good fitness result. The establishment of generation amount, population amount, the probability of cross-breeding and mutation also greatly take effect on giving an optimal result with a better fitness value. This research is expected to help on the development of SITS and the angkot revitalization project in Surabaya.

Keywords: Orienteering Problem, Genetic Algorithm, Public Transport, Optimation, Modeling

KATA PENGANTAR

Puji syukur atas karunia yang telah diberikan Ida Sang Hyang Widhi Wasa selama ini sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul:

OPTIMASI JALUR TRANSPORTASI UMUM MENGGUNAKAN ORIENTEERING PROBLEM DAN GENETIC ALGORITHM (STUDI KASUS: JALUR ANGKOT SURABAYA)

Terima kasih atas pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik materi maupun spiritual demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak dan Mama, yaitu I Ketut Erawan dan Dina Diatmika yang telah mendoakan dan memberikan dukungan kepada penulis sehingga dapat menyelesaikan pendidikan S1 ini dengan baik.
2. Adik tersayang, Nia dan Caya yang telah mendoakan dan memberikan dukungan kepada penulis sehingga dapat menyelesaikan pendidikan S1 ini dengan baik.
3. Ibu Wiwik Anggraeni, S.Si, M.Kom. serta Bapak Ahmad Mukhlason, S.Kom, M.Sc. selaku dosen pembimbing yang telah memberikan ilmu, bimbingan, dan motivasi untuk kelancaran tugas akhir ini.
4. Bapak Ir. Achmad Holil Noor Ali, M.Kom. selaku dosen wali penulis selama menempuh pendidikan di Jurusan Sistem Informasi.
5. Bapak Edwin Riksa Komara, S.Kom., M.T. serta Bapak Radityo Prasetianto W., S.Kom., M.Kom. selaku dosen penguji yang telah memberikan kritik, saran, dan masukan yang dapat menyempurnakan Tugas Akhir ini.
6. Seluruh dosen pengajar beserta staf Tata Usaha yang telah memberikan ilmu dan bantuan kepada penulis selama menempuh pendidikan di Jurusan Sistem Informasi, FTIf ITS Surabaya.

7. Tim optimasi trayek angkot Surabaya yaitu Jockey Satria dan Dhamar Bagas yang selalu bersama dalam suka dan duka selama mengerjakan tugas akhir ini.
8. Kadek Jossy Alandari yang selalu mendoakan, memberikan semangat dan motivasi kepada penulis dalam keadaan apapun.
9. Teman-teman kontrakan: Teja, Anom, Cok Adit, dan Anra, yang selalu berbagi suka dan duka di dalam satu atap yang sama.
10. Teman-teman Jalan-Jalan Men: Iksan, Aboy, Bambang, dan Rhesa yang mewarnai hari-hari yang kosong selama menjalani pekuliahan.
11. Sahabat Suku Air: Nino, Gustisa, Panjul, Gunglir, Cempaka, Ria, Gek in, dan Wulan yang memberikan semangat dan menghibur penulis selama perkuliahan.
12. Teman-teman BELTRANIS, SOLARIS, BASILISK, dan OSIRIS yang selalu menemani penulis dalam melewati hari-hari dan berbagi pengalaman selama pekuliahan.
13. Teman-teman penulis di organisasi Tim Pembina Kerohanian Hindu (TPKH) ITS dan kepanitiaan Information Systems Expo 2015 yang telah banyak memberikan pelajaran dan pengalaman serta membantu penulis selama bekerja.
14. Berbagai pihak yang telah membantu dalam pengerjaan Tugas Akhir ini yang belum mampu penulis sebutkan di atas.

Penyusunan laporan ini masih jauh dari sempurna, untuk itu saya menerima adanya kritik dan saran yang membangun untuk perbaikan di masa mendatang. Semoga buku tugas akhir ini dapat memberikan manfaat pembaca.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

DAFTAR TABEL

"Halaman ini sengaja dikosongkan"

DAFTAR GAMBAR

"Halaman ini sengaja dikosongkan"

DAFTAR KODE

BAB I
PENDAHULUAN

Pada bab pendahuluan akan dijelaskan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, 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

Kemacetan lalu lintas merupakan permasalahan umum yang sering terjadi pada kota-kota besar di berbagai negara, tidak terkecuali kota Jakarta dan Surabaya. Pada tahun 2016, seperti yang diberitakan oleh *wonderlist.com*, kedua kota tersebut termasuk ke dalam sepuluh tempat dengan masalah lalu lintas terburuk di dunia dimana Jakarta dan Surabaya menduduki peringkat pertama dan keempat [1]. Banyak faktor yang mempengaruhi terjadinya kemacetan lalu lintas di dua kota tersebut yaitu kepadatan penduduk yang cukup tinggi dan kurangnya kesadaran masyarakat untuk mematuhi peraturan lalu lintas seperti masih adanya pejalan kaki yang menyeberang sembarangan, penyalahgunaan fungsi trotoar untuk parkir kendaraan, serta angkutan umum yang menaikkan dan menurunkan penumpang dengan tidak tertib.

Salah satu penyebab utamanya adalah peningkatan pertumbuhan kendaraan bermotor yang tidak sebanding dengan pertumbuhan jalan. Berdasarkan penuturan dari Gubernur DKI Jakarta Basuki Tjahaja Purnama, jumlah kendaraan bermotor di Jakarta bertambah sebanyak 1200 unit per harinya [2]. Jika pertumbuhan tersebut dikalkulasikan, maka jumlah kendaraan meningkat hingga sembilan persen dimana tidak sebanding dengan pertumbuhan jalannya yang hanya 0,01 persen [3]. Begitu juga halnya di Surabaya, dimana pertambahan

kendaraan bermotor di Surabaya bertambah sebanyak 17000 unit per bulan dimana setiap bulannya bahkan setiap tahunnya belum tentu bertambah jalan baru [3] [4].

Banyak kerugian yang diperoleh masyarakat selama mengalami kemacetan lalu lintas, seperti kelebihan pengeluaran untuk biaya bahan bakar kendaraan, terganggunya kesehatan akibat polusi udara, dan menurunnya produktivitas tiap individu akibat banyaknya waktu yang dikorbankan selama menghadapi kemacetan di jalan. Hal ini juga dirasakan oleh pihak korporasi (perusahaan) karena kemacetan membuat biaya transportasi barang maupun jasa meningkat. Kemacetan lalu lintas berujung pada terhambatnya pertumbuhan ekonomi di kota-kota besar, bahkan pada tahun 2016 berdasarkan laporan Masyarakat Transportasi Indonesia (MTI), Dinas Perhubungan DKI Jakarta menghitung kerugian masyarakat akibat dampak kemacetan lalu lintas di Jakarta mencapai Rp 150 triliun per tahun [5] [6]. Begitu juga di Surabaya, seperti yang dilansir viva.co.id pada tahun 2010, kerugian bisa mencapai 1 triliun setiap harinya [7].

Berbagai tindakan telah dilakukan pemerintah untuk dapat menyelesaikan atau setidaknya mengurangi kemacetan lalu lintas. Pemerintah Surabaya sendiri telah melakukan penambahan ruas jalan, pengaturan lalu lintas yang lebih tertib, dan peningkatan penggunaan transportasi umum seperti *Mass Rapid Transit* (MRT), monorel, dan termasuk adanya rencana revitalisasi angkot/mikrolet [8]. Peningkatan penggunaan transportasi umum merupakan upaya yang tepat dalam menghadapi salah satu faktor utama penyebab kemacetan yaitu meningkatnya pertumbuhan kendaraan bermotor yang tidak sebanding dengan pertumbuhan jalan seperti yang telah dipaparkan pada paragraf sebelumnya.

Terkait dengan upaya peningkatan penggunaan transportasi umum di Surabaya, maka pemerintah harus dapat menarik minat masyarakat sehingga transportasi umum dapat menjadi pilihan utama masyarakat. Dalam menarik minat masyarakat,

pemerintah harus dapat menyediakan transportasi umum yang nyaman, murah, cepat, dan terjadwal. Adanya perkembangan teknologi informasi yang semakin pesat sudah dimanfaatkan oleh Pemerintah Surabaya untuk mendukung transportasi umum yang semakin efektif dan efisien dengan dibuatnya sistem manajemen transportasi yaitu Surabaya *Intelligent Transport Systems* (Surabaya ITS atau SITS) [9]. Di negara maju seperti Amerika Serikat, Jepang, dan Singapura telah menerapkan ITS untuk manajemen lalu lintas dan transportasinya dengan baik [10].

Penerapan SITS baru sampai pada tahap pemantauan kondisi kemacetan lalu lintas secara *real time* menggunakan CCTV [9]. Berdasarkan hal tersebut, diperlukan pengembangan lebih lanjut dimana dengan mengacu pada *ITS America Strategic Plan* dan penyesuaian kondisi yang ada di Surabaya, maka dibuat rencana strategis dari penerapan SITS yang dibagi menjadi beberapa bidang. Salah satunya yang difokuskan pada tahun ini adalah *Advanced Traveler Information Systems* (ATIS). Sasaran akhir dari ATIS adalah sebuah sistem cerdas yang dapat memberikan informasi termasuk saran yang optimum mengenai rute perjalanan dengan transportasi umum dari satu tempat ke tempat lainnya. Harapannya, penelitian tugas akhir ini dapat mendukung tercapainya sasaran dari ATIS dalam memberikan model dan solusi berupa rekomendasi rute perjalanan kepada masyarakat Surabaya dan para pengunjung khususnya yang ingin berkeliling mengunjungi banyak tempat di Surabaya menggunakan sarana transportasi umum. Untuk penelitian ini sendiri mengambil objek yaitu mikrolet/angkot sehubungan dengan rencana revitalisasi angkot di Surabaya. Angkot dikenal sebagai transportasi umum darat yang cukup favorit di Surabaya dan menjadi ciri khas Indonesia dikarenakan tarifnya yang murah dan hampir di setiap kota di Indonesia memiliki jalur angkot dengan banyak armada di setiap jalurnya [11]. Dengan memanfaatkan perkembangan teknologi agar rencana revitalisasi angkot dapat terwujud, maka

dalam penelitian ini dilakukan optimasi penentuan rute perjalanan dengan mikrolet/angkot di Surabaya.

Pembuatan model diusulkan menggunakan *Orienteering Problem* (OP). Metode ini digunakan pada kasus-kasus optimasi dengan kondisi terbatasnya waktu yang tersedia [12]. Seperti yang disebutkan pada penelitian Tsiligirides pada tahun 1984 dimana kasus ini merupakan kasus pertama yang diselesaikan dengan OP yaitu penjualan yang tidak memiliki cukup waktu untuk mengunjungi semua kota [13]. Untuk itu diperlukan model untuk optimasi dalam memaksimalkan total penjualan di kota-kota yang dianggap memiliki profit tinggi selama waktu yang terbatas tersebut. Permasalahan lainnya yang diselesaikan dengan OP adalah perencanaan perjalanan wisata seperti yang disebutkan dalam penelitian Souffriau dan temannya pada tahun 2008, Wang dan temannya pada tahun 2008, serta Schilde dan temannya pada tahun 2009 dimana para wisatawan tidak mungkin dapat mengunjungi semua tempat wisata yang mereka sukai dikarenakan waktu yang terbatas [14] [15] [16]. Penelitian ini memiliki karakteristik permasalahan yang sama yaitu memaksimalkan nilai yang didapat pengguna, adanya batasan waktu/biaya, dan tidak semua *node* (tempat) perlu dikunjungi.

Setelah model dibentuk, solusi dapat dicari dimana algoritma yang diusulkan dalam penelitian ini adalah *Genetic Algorithm* (GA). Algoritma ini sudah sering digunakan dalam menyelesaikan permasalahan optimasi, dan sudah ada beberapa penelitian yang menggunakan algoritma ini untuk mencari solusi dari model OP seperti pada penelitian Tasgetiren dan Smith [17] [18]. Penelitian tersebut menyelesaikan permasalahan optimasi rute pariwisata menggunakan GA dari model OP. Mereka juga membandingkan GA dengan algoritma yang disusulkan oleh peneliti lain yaitu Chao dan kawan-kawan tahun 1996 dengan *Chao's heuristic*, Tsiligirides tahun 1984 dengan *Tsiligirides's Stochastic Algorithm*, Wang dan kawan-kawan tahun 1995 dengan *Artificial Neural Network* (ANN)

dimana pada kasus ini, solusi yang dihasilkan GA lebih baik dibandingkan hasil dari ketiga algoritma tersebut [17]. Sedangkan penelitian yang dilakukan oleh Ferreira dan kawan-kawan pada tahun 2014 terhadap permasalahan optimasi transportasi pengangkut limbah juga menghasilkan solusi yang baik menggunakan GA [19]. Secara keseluruhan, penelitian-penelitian di atas menunjukkan bahwa GA mampu menghasilkan solusi yang baik untuk permasalahan model OP.

Berdasarkan kelebihan-kelebihan OP dan GA dari penelitian yang telah dilakukan sebelumnya, maka dalam tugas akhir ini akan dilakukan pembuatan model *Orienteering Problem* (OP) dengan menggunakan *Genetic Algorithm* (GA) untuk mencari solusi berdasarkan model tersebut. Solusi dari penelitian ini berupa rekomendasi rute perjalanan menggunakan angkot di Surabaya yang optimum dengan memaksimalkan peluang mendapatkan angkot.

1.2. Perumusan Masalah

Rumusan masalah yang akan diselesaikan dalam tugas akhir ini adalah:

a. Seperti apakah model yang dapat digunakan dalam optimasi penentuan rute perjalanan dengan mikrolet/angkot di Surabaya?
b. Bagaimana *Orienteering Problem* dapat diterapkan untuk memodelkan penentuan rute perjalanan dengan angkot di Surabaya yang optimum?
c. Bagaimana mendapatkan solusi berdasarkan model yang telah ditemukan menggunakan *Genetic Algorithm*?

1.3. Batasan Masalah

Batasan masalah terkait pengerjaan tugas akhir ini adalah:

a. Objek penelitian dalam tugas akhir ini adalah transportasi umum yaitu mikrolet/angkot (angkutan kota) di Surabaya.
b. Luaran dari penelitian ini ditujukan kepada masyarakat Surabaya dan para pengunjung khususnya yang ingin berkeliling mengunjungi banyak tempat di Surabaya menggunakan transportasi umum yaitu angkot.
c. Cakupan tugas akhir ini adalah enam jalur mikrolet di Surabaya dengan daerah asal-tujuan yaitu Kalimas Barat/Petekan - Manukan Kulon (DP), Dukuh Kupang - Benowo (I) Kalimas Barat - Bratang (Q), Joyoboyo - Tubanan - Manukan (TV2), Kalimas Barat - Benowo (Z), Benowo - Ujung Baru (Z1).
d. Data yang digunakan dalam tugas akhir ini adalah jumlah angkot dari masing-masing jalur dan waktu tempuh antar *node* yang merepresentasikan tempat pemberhentian angkot yang masing-masing didapatkan melalui Dinas Perhubungan Kota Surabaya dan *Google Maps*.
e. Dalam penelitian ini tidak melibatkan faktor psikologis dan finansial dari pengguna.

1.4. Tujuan Tugas Akhir

Tujuan dari penelitian tugas akhir ini adalah:

a. Membuat model yang dapat digunakan dalam optimasi penentuan rute perjalanan dengan mikrolet di Surabaya menggunakan *Orienteering Problem*.
b. Mendapatkan solusi berupa rekomendasi rute optimum berdasarkan model yang telah ditemukan menggunakan *Genetic Algorithm*.
c. Membuat data set yang nantinya dapat digunakan untuk penelitian selanjutnya.

1.5. Manfaat Tugas Akhir

Manfaat yang diharapkan dapat dihasilkan dari pengerjaan tugas akhir ini adalah:

Bagi Peneliti

a. Mampu memahami penerapan metode *Orienteering Problem* dan *Genetic Algorithm* dalam melakukan optimasi khususnya pada bidang transportasi umum.

Bagi Pendidikan

a. Menjadikan penelitian ini sebagai sumber pustaka untuk penelitian sejenis terkait *Orienteering Problem* dan *Genetic Algorithm* khususnya pada bidang transportasi.
b. Mempermudah kelanjutan dari penelitian ini dengan dibuatnya data set.

Bagi Pemerintah

a. Mendapatkan masukan untuk penyempurnaan implementasi Surabaya *Intelligent Transport Systems* (SITS) khususnya pada bidang *Advanced Traveler Information Systems* (ATIS).

1.6. Relevansi

Penggunaan transportasi umum merupakan solusi yang tepat dari Pemerintah Surabaya dalam mengatasi kemacetan dimana mengurangi jumlah penggunaan kendaraan pribadi. Untuk menarik minat masyarakat menggunakan transportasi umum, Pemerintah Surabaya telah membangun sistem manajemen transportasi yaitu Surabaya *Intelligent Transport Systems* (SITS) dalam mendukung penyediaan transportasi umum yang semakin efektif dan efisien. Namun SITS perlu dikembangkan dengan adanya *Advanced Traveler Information Systems* (ATIS) sebagai salah satu bidang rencana strategis dengan sasaran akhir sebuah sistem cerdas yang dapat memberikan informasi termasuk saran yang paling optimal mengenai rute perjalanan dengan transportasi umum dari satu tempat ke tempat lainnya.

Dengan menggunakan *Genetic Algorithm*, solusi yang dihasilkan berupa rekomendasi rute perjalanan kepada masyarakat yang ingin menggunakan sarana transportasi umum mikrolet/angkot guna mendukung sasaran akhir ATIS berdasarkan model dari *Orienteering Problem*. Tugas akhir ini diharapkan dapat menjadi masukan untuk penyempurnaan implementasi SITS dan sumber pustaka untuk penelitian selanjutnya terkait optimasi dalam manajemen transportasi khususnya mikrolet.

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan mengenai penelitian sebelumnya 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. Penelitian Sebelumnya

Berikut ini penelitian-penelitian sebelumnya yang dijadikan acuan dalam pengerjaan tugas akhir yang disajikan dalam bentuk tabel.

Tabel 2.1 Studi Literatur 1

Judul Paper	A Genetic Algorithm for the Orienteering Problem [18]
Penulis; Tahun	M. Faith Tasgetiren, Alice E. Smith; 2000
Deskripsi Umum Penelitian	Dalam penelitian ini digunakan *Genetic Algorithm* (GA) untuk mencari solusi dari permodelan OP. Menggunakan GA, peneliti ingin mendapatkan rute wisata optimum dari *node* awal dan akhir yang telah ditetapkan sebelumnya dengan mencari total skor maksimum dan adanya batasan waktu TMAX. Di dalam proses GA, digunakan *penalty function* untuk mencegah beberapa solusi dari populasi yang masuk dengan memperbaiki keturunan (*offspring*) yang tidak layak. Solusi dari GA selanjutnya dibandingkan dengan hasil algoritma-algoritma lain yang diusulkan peneliti sebelumnya yaitu Chao dan kawan-kawan (1996) dengan *Chao's heuristic*, Tsiligirides (1984) dengan

	Tsiligirides's Stochastic Algorithm, Wang dan kawan-kawan (1995) dengan *Artificial Neural Network* (ANN). Melalui empat set ujian dengan 67 kasus, dibandingkan *Tsiligirides's Stochastic Algorithm*, GA mengungguli dalam semua kasus. Dibandingkan dengan ANN, GA menghasilkan hasil yang sama atau lebih baik sebanyak 66 dari 67 kasus. Dibandingkan dengan *Chao's heuristic*, GA menghasilkan hasil yang sama sebanyak 61 dari 67 kasus, sedangkan lima dari enam kasus sisanya diungguli GA. Dari hasil tersebut dapat disimpulkan bahwa GA berhasil mengungguli ketiga algoritma lainnya.
Keterkaitan Penelitian	Kesamaan dalam penelitian ini adalah penggunaan *Genetic Algorithm* yang dapat diterapkan dalam studi kasus optimasi rute perjalanan dengan mikrolet yang juga menggunakan permodelan *Orienteering Problem*. Kasus penelitian ini juga memiliki karakteristik yang sama yaitu, adanya batasan waktu sehingga tidak semua *node* (tempat) perlu dikunjungi.

Tabel 2.2 Studi Literatur 2

Judul Paper	A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem [17]
Penulis; Tahun	M. Faith Tasgetiren; 2001
Deskripsi Umum Penelitian	Penelitian ini merupakan pengembangan dari penelitian Tasgetiren yang sebelumnya pada tahun 2000. Pada algoritma ini masih

	menyelesaikan permasalahan yang sama, namun algoritma yang digunakan yaitu *Genetic Algorithm* dengan pengembangan *adaptive penalty function*. Hal ini terbukti dengan hasil yang lebih baik ketika dibandingkan dengan tiga algoritma yang juga disebutkan dalam penelitian sebelumnya. Dibandingkan *Tsiligirides's Stochastic Algorithm*, GA dengan *adaptive penalty function* mendapatkan hasil yang lebih baik sebanyak 46 dari 67 kasus dan sisanya memiliki hasil yang sama. Dibandingkan dengan ANN, GA dengan *adaptive penalty function* mendapatkan hasil yang lebih baik sebanyak 8 dari 67 kasus, 1 kasus tidak lebih baik dan sisanya memiliki hasil yang sama. Dibandingkan dengan *Chao's heuristic*, GA dengan *adaptive penalty function* menghasilkan hasil yang sama sebanyak 64 dari 67 kasus, sedangkan sisanya diungguli GA. Dari hasil tersebut dapat disimpulkan bahwa GA dengan *adaptive penalty function* berhasil mengungguli ketiga algoritma lainnya dan juga GA itu sendiri.
Keterkaitan Penelitian	Kelebihan dari penelitian ini adalah penggunaan *Genetic Algorithm* yang dikembangkan dengan penambahan *adaptive penalty function* dan mampu mendapatkan hasil yang lebih baik dari sebelumnya. Kasus penelitian ini juga memiliki karakteristik yang sama yaitu, adanya batasan waktu sehingga tidak semua *node* (tempat) perlu dikunjungi.

Tabel 2.3 Studi Literatur 3

Judul Paper	Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach [19]
Penulis; Tahun	J. Ferreira, A. Quintas, dan J. A. Oliveira; 2014
Deskripsi Umum Penelitian	Dalam penelitian ini digunakan *Genetic Algorithm* untuk menyelesaikan permasalahan *Team Orienteering Problem* (TOP) untuk perusahaan pengolah limbah dalam proses pengumpulan sampah ke setiap pelanggan. Tujuannya adalah mencari rute optimum yang memaksimalkan total keuntungan dan meminimalkan sumber daya yang dibutuhkan. Hasil yang didapatkan dari penelitian ini mencapai 60% dari skor terbaik dalam seleksi 24 kasus *benchmark* TOP, dengan rata-rata eror sebesar 18.7% pada sisa kasusnya. Dapat disimpulkan bahwa *Genetic Algorithm* terbukti mampu memecahkan permasalahan TOP secara efisien dengan hasil yang baik.
Keterkaitan Penelitian	Adanya penelitian ini menunjukkan bahwa GA juga dapat digunakan pada varian dari OP yaitu *Team Orienteering Problem* (TOP). Penelitian ini dapat menjadi referensi untuk mempelajari bagaimana optimasi menggunakan GA.

Tabel 2.4 Studi Literatur 4

Judul Paper	The Orienteering Problem: A Survey [12]

Penulis; Tahun	Pieter Vansteenwegen, Wouter Souffriau, Dirk Van Oudheusden; 2011
Deskripsi Umum Penelitian	Penelitian ini menjelaskan secara detail mengenai *Orienteering Problem*. Penjelasan tersebut mencakup definisi permasalahan, formulasi matematika, pengaplikasian dalam dunia nyata, *benchmark instances*, dan pendekatan solusi (algoritma) untuk masing-masing jenis dari OP. Jenis OP yang dibahas dalam penelitian ini adalah *Orienteering Problem* itu sendiri, *Team Orienteering Problem*, *Orienteering Problem with Time Windows*, dan *Team Orienteering Problem with Time Windows*. Selain itu dibahas juga secara singkat jenis OP lainnya seperti *Generalized Orienteering Problem* (GOP) dan *Orienteering Problem with Stochastic Profits* (OPSP). Kesimpulan dari penelitian ini adalah sudah banyak permasalahan dalam dunia nyata yang dapat dimodelkan ke dalam OP atau jenis-jenisnya, seperti model optimasi rute pariwisata, perekrutan atlet, atau militer dan banyaknya algoritma yang digunakan untuk mencari solusi dari model OP yang telah dibuat.
Keterkaitan Penelitian	Karakteristik dari OP cocok digunakan untuk membuat model dari permasalahan pada penelitian ini yaitu optimasi rute perjalanan dengan mikrolet di Surabaya dimana terdapat batasan waktu sehingga tidak semua tempat harus dikunjungi.

Tabel 2.5 Studi Literatur 5

Judul Paper	Genetic Algorithm Solving Orienteering Problem in Large Networks [20]
Penulis; Tahun	J. Karbowska-Chilinska, J. Koszelew, K. Ostrowski, dan P. Zabielski; 2012
Deskripsi Umum Penelitian	Penelitian ini menggunakan GA yang efektif untuk menyelesaikan permasalahan OP dalam sebuah jaringan transportasi yang luas dengan 900 *node* di dalamnya. Tujuan dari penelitian ini tidak hanya mencari total profit maksimum, melainkan juga mencari panjang perjalanan sesuai dengan rute yang dihasilkan. Hasil dari GA juga dibandingkan dengan parallel Genetic Algorithm (pGA) dan Genetic Local Search (GLS). Dibandingkan dengan GLS, GA memberikan hasil yang lebih baik sekitar 2% pada t_{max} = 2500 dan 12% pada t_{max} = 1000. Dibandingkan dengan pGA, GA memberikan hasil yang lebih baik sekitar 1.8% pada t_{max} = 3000 dan 8% pada t_{max} = 2500. Dapat disimpulkan bahwa *Genetic Algorithm* terbukti mampu memecahkan permasalahan OP secara efisien dengan hasil yang baik.
Keterkaitan Penelitian	GA mampu untuk membuat solusi dari permasalahan OP bahkan dalam area yang luas dengan banyak *node*. Penelitian ini dapat menjadi referensi untuk mempelajari bagaimana optimasi dari model OP menggunakan GA.

2.2. Dasar Teori

Berikut ini dijabarkan dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir, yaitu:

2.2.1 Kemacetan Lalu Lintas

Banyak definisi yang telah diusulkan untuk menggambarkan secara jelas mengenai apa itu kemacetan lalu lintas. Namun sampai saat ini belum ada definisi yang diterima secara universal terkait kemacetan lalu lintas. Berdasarkan sumber, definisi kemacetan dapat dikategorikan menjadi tiga kelompok, sebagai berikut [21].

a. Terkait dengan kapasitas, dapat disimpulkan kemacetan adalah suatu kondisi dimana jumlah kendaraan yang padat melebihi kapasitas jalan dengan kecepatan arus lalu lintas yang rendah.
b. Terkait dengan waktu, dapat disimpulkan kemacetan merupakan suatu kondisi arus lalu lintas yang lambat sehingga menimbulkan antrean yang panjang dan menyebabkan kerugian waktu.
c. Terkait dengan biaya, dapat disimpulkan kemacetan merupakan suatu kondisi lalu lintas yang mengacu pada biaya tambahan yang dihasilkan dari gangguan antar pengguna jalan.

Oleh karena itu, berdasarkan poin-poin tersebut, dapat diraih kesimpulan terkait definisi kemacetan lalu lintas, yaitu suatu kondisi terhambatnya arus lalu lintas yang disebabkan jumlah kendaraan yang melebihi kapasitas jalan dan ditandai dengan adanya antrean panjang yang menyebabkan kerugian waktu dan juga adanya biaya tambahan.

2.2.2 Optimasi

Optimasi merupakan pusat dari seluruh permasalahan yang melibatkan pengambilan keputusan, baik dalam bidang teknik atau ekonomi. Pengambilan keputusan melibatkan berbagai alternatif dan pengambil keputusan harus memilih salah satu di antara alternatif tersebut. Ukuran baik tidaknya alternatif digambarkan dalam bentuk fungsi tujuan. Optimasi berguna untuk memilih alternatif terbaik dengan mendapatkan nilai

optimum baik itu maksimum atau minimum berdasarkan fungsi tujuan yang diberikan [22].

Dalam beberapa tahun terakhir, fenomena terkait optimasi telah mendapatkan perhatian yang besar terutama akibat pesatnya kemajuan teknologi komputer, termasuk di dalamnya pengembangan dan ketersediaan aplikasi yang *user-friendly*, dan prosesor paralel serta berkecepatan tinggi. Contoh nyata dari fenomena ini adalah pengembangan yang semakin luas pada *tools* optimasi seperti Optimization Toolbox of MATLAB dan banyak paket aplikasi komersial lainnya [22].

2.2.3 Pre-Processing Data

Pre-processing data merupakan suatu proses yang digunakan untuk mengubah data mentah menjadi data yang berkualitas dimana data tersebut sudah siap digunakan untuk proses pengolahan data. Data mentah belum siap digunakan untuk proses pengolahan data karena memiliki beberapa kekurangan sebagai berikut [23].

a. *Incomplete*, yaitu sebagian data hilang atau kekurangan nilai atribut.
b. *Noisy*, yaitu data masih mengandung error atau memiliki nilai-nilai yang menyimpang (*outlier*).
c. *Inconsistent*, ketidakcocokan pengisian nilai dalam suatu kode atau nama.

Dalam melakukan pre-processing data dapat digunakan beberapa teknik sebagai berikut [23].

a. *Data Cleaning*, digunakan untuk memperkecil *noise*, membetulkan data yang tidak konsisten, mengisi nilai yang kosong, atau mengidentifikasi atau menghapus *outlier*.
b. Data Integration, digunakan untuk menggabungkan data dari berbagai sumber ke dalam satu tempat penyimpanan data yang sesuai.

c. Data Reduction, digunakan untuk menguraikan data ke dalam bentuk yang ukurannya lebih kecil.
d. Data Transformation, digunakan untuk generalisasi data, normalisasi data, atau pengumpulan data ke dalam bentuk yang sama.

2.2.4 Algoritma Dijkstra

Algoritma ini ditemukan oleh E. W. Dijkstra pada tahun 1959 dengan tujuan untuk menyelesaikan *minimal spanning tree problem* dan *shortest path problem* [24]. Algoritma ini digunakan untuk menyelesaikan permasalahan dalam mencari rute terpendek dari titik awal (sumber) ke seluruh titik tujuan dalam sebuah *weighted graph* baik itu memiliki arah (*directed*) atau tidak (*undirected*). Ketentuan dalam menggunakan algoritma ini adalah graph yang digunakan tidak boleh memiliki bobot negatif di setiap garis penghubung antar titik [25].

2.2.5 Orienteering Problem

Orienteering Problem (OP) berasal dari sebuah permainan olahraga yaitu *orienteering* dimana setiap pemain mengunjungi pos-pos (*node*) yang disediakan dan kembali ke pos awal permainan dalam waktu yang telah ditentukan. Tiap pos memiliki skor yang berbeda dan tujuan dari permainan ini adalah mencari jumlah skor sebanyak mungkin. Maka dari itu, dibuatlah metode OP untuk menentukan rute berdasarkan *node-node* yang telah dipilih dengan tujuan untuk memaksimalkan jumlah skor, sekaligus mendapatkan waktu minimum. Berdasarkan tujuan tersebut, OP dilihat sebagai gabungan antara *Knapsack Problem* (KP) dan *Travelling Salesperson Problem* (TSP). Karena dalam OP tidak semua *node* yang tersedia harus dikunjungi, maka metode ini sangat membantu untuk mengunjungi *node* yang ada sebanyak mungkin dalam waktu yang disediakan [12].

2.2.3.1 Konsep Dasar Orienteering Problem

Berikut dijabarkan istilah-istilah yang sering digunakan dalam *Orienteering Problem* [12].

a. Skor (S), merupakan nilai/manfaat yang diperoleh ketika mengunjungi suatu *node*. Setiap *node* memiliki jumlah skor yang berbeda dimana penetapan skor untuk setiap *node* ditentukan sesuai dengan kepentingan dari peneliti.
b. T_{max} (umumnya berupa batasan waktu), merupakan batasan maksimum waktu yang disediakan ketika menghitung total skor maksimum.
c. *Vertex* atau *node*, merepresentasikan titik bersinggungannya dua atau lebih garis (arc). *Node* umumnya dinotasikan dalam bentuk titik atau lingkaran kecil. Contoh *node* dalam kasus nyata yaitu kota dan terminal.
d. Arc, merupakan penghubung suatu *node* dengan *node* lainnya. Arc umumnya dinotasikan dalam bentuk garis (baik berisi panah atau tidak) yang menghubungkan *node*. Contoh arc dalam kasus nyata adalah jalan, jalur transportasi.
e. Lintasan Hamiltonian merupakan lintasan yang melalui tiap *node* dalam suatu graf tepat satu kali. OP digunakan untuk mencari lintasan Hamiltonian terpendek berdasarkan *node-node* yang dipilih, sehingga dalam OP *node* paling banyak dikunjungi satu kali.

2.2.3.2 Model Orienteering Problem

Diberikan sejumlah N *node* i dan masing-masing *node* diberi skor S_i. Ditentukan terlebih dahulu *node* awal (*node* 1) dan *node* akhir (*node* N) beserta beban (waktu tempuh) yang dibutuhkan dari *node* i ke *node* j, t_{ij}. Sesuai dengan penjelasan sebelumnya, tujuan OP adalah menentukan lintasan (*path*) berdasarkan batasan T_{max} dengan mengunjungi beberapa *node* untuk

mendapatkan total skor maksimum dan tidak semua *node* harus dikunjungi.

Berdasarkan notasi yang telah dibahas pada paragraf di atas, OP dapat diformulasikan sebagai pemrograman integer. Variabel keputusan $x_{ij} = 1$ jika kunjungan ke *node* i diikuti dengan kunjungan ke *node* j, dan jika tidak maka $x_{ij} = 0$. Dengan u_i sebagai posisi *node* i dalam lintasan, berikut fungsi tujuan dan batasan-batasan yang diformulasikan.

$$Max \sum_{i=1}^{N-1} \sum_{j=2}^{N} S_i x_{ij}, \quad (0)$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1, \quad (1)$$

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

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

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

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

$$x_{ij} \in \{0,1\}; \ \forall 1, j = 1, \dots, N. \quad (6)$$

Fungsi tujuan OP seperti pada persamaan (0) adalah memaksimalkan total skor. Agar fungsi tujuan tersebut terpenuhi, terdapat beberapa batasan yaitu batasan dengan persamaan (1) menjamin lintasan akan dimulai pada *node* 1 dan berakhir di *node* N. Batasan dengan persamaan (2) memastikan semua *node* dalam lintasan saling terhubung dan setiap *node* paling banyak dikunjungi satu kali. Batasan dengan persamaan (3) memastikan jumlah waktu tempuh tidak melebih batasan

waktu (T_{max}). Batasan dengan persamaan (4) dan (5) mencegah adanya *subtours*, yaitu kondisi dimana lintasan dimulai dan berakhir pada titik yang sama (membentuk sirkuit) [12].

2.2.6 Genetic Algorithm

Genetic Algorithm (GA) merupakan sebuah algoritma *meta-heuristics* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam permasalahan optimasi. Algoritma ini memiliki prinsip seperti pada proses biologi dari seleksi alam dan evolusi. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu dimana masing-masing individu merepresentasikan sebuah alternatif solusi dari permasalahan yang ada. Dalam algoritma ini, individu dilambangkan dengan sebuah nilai fitness dimana nilai tersebut yang nantinya akan digunakan untuk mencari solusi terbaik dari permasalahan tersebut.

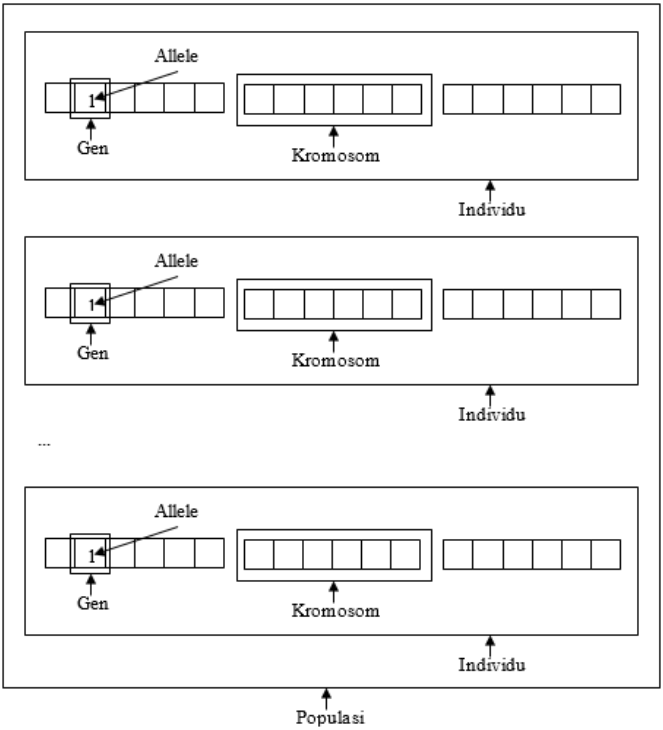
Dalam suatu populasi, individu memiliki kesempatan untuk dapat melakukan reproduksi melalui perkawinan silang dengan individu lainnya. Dari perkawinan silang tersebut terdapat individu baru yang dinamakan keturunan dimana individu baru tersebut membawa beberapa sifat induknya. Sedangkan individu populasi yang tidak terseleksi dalam proses reproduksi akan mati dengan sendirinya. Dengan begitu, generasi dengan karakteristik yang bagus akan bermunculan dalam populasi tersebut untuk kemudian disilangkan dengan karakter lainnya. Dengan melakukan kawin silang sebanyak mungkin, maka akan semakin besar kemungkinan untuk mendapatkan hasil yang terbaik. Beberapa hal yang dilakukan dalam *Genetic Algorithm* sebagai berikut [26].

a. Mendefinisikan Individu

   Individu, merepresentasikan nilai yang menyatakan salah satu solusi yang mungkin dari suatu permasalahan. Berikut dijabarkan istilah-istilah penting terkait dengan tahapan ini.

- Gen, merupakan sebuah nilai yang menyatakan satuan dasar yang membentuk arti tertentu dalam satu kesatuan gen.
- *Allele*, merupakan nilai dari gen. *Allele* dalam gen tersebut dapat direpresentasikan ke dalam berbagai bentuk seperti bit, bilangan real, dan elemen permutasi.
- Kromosom, satu kesatuan gen yang membentuk nilai tertentu.
- Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi (satu iterasi).

Istilah-istilah tersebut dapat didefinisikan seperti pada Gambar 2.1.

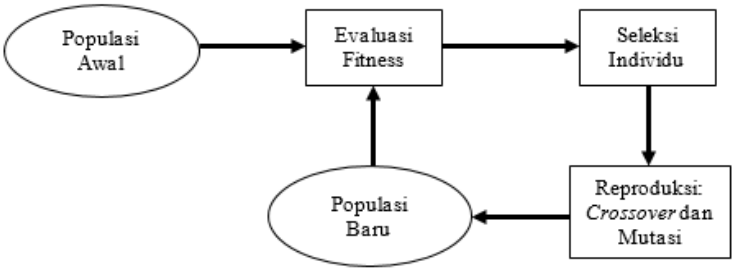


Gambar 2.1 Ilustrasi Pendefinisian Individu

b. Mendefinisikan Nilai Fitness

Nilai *fitness*, merupakan nilai yang merepresentasikan baik tidaknya suatu solusi (individu). Nilai ini dijadikan acuan dalam mencapai solusi optimum dalam *Genetic Algorithm* (GA). GA bertujuan untuk mencari individu dengan nilai *fitness* yang paling tinggi. Nilai fitness diformulasikan sesuai dengan kondisi dari penelitian tersebut. Misalnya jika tujuan penelitian adalah meminimalkan jarak, maka nilai fitness dapat berupa inversi total jarak.

GA memiliki beberapa tahapan dalam satu generasi (iterasi). Tahapan-tahapan tersebut digambarkan ke dalam siklus yang dikemukakan oleh David Goldberg seperti Gambar 2.2 [26].



Gambar 2.2 Siklus Algoritma Genetika oleh David Goldberg

a. Pembangkitan Populasi Awal

Proses ini merupakan tahapan untuk membangkitkan sejumlah individu (solusi) secara acak atau dengan prosedur tertentu. Populasi awal muncul karena adanya proses pengkodean. Proses pengkodean akan menghasilkan suatu deretan atau kumpulan gen yang disebut kromosom seperti ilustrasi pada Gambar 2.1. Ukuran dari populasi awal tergantung pada kondisi permasalahan yang ingin diselesaikan.

b. Evaluasi Fitness

Berdasarkan nilai fitness, pada tahapan ini akan dilakukan evaluasi pada populasi awal maupun populasi baru yang nantinya terbentuk. Untuk mengetahui pencapaian nilai optimum dapat dilihat dari seberapa tinggi nilai fitness. Individu yang memiliki nilai fitness tinggi akan bertahan hidup, sedangkan individu dengan nilai fitness rendah akan gugur.

c. Seleksi

Proses ini digunakan untuk memilih individu-individu dalam suatu populasi untuk dijadikan induk dalam proses kawin silang (*crossover*). Terdapat beberapa teknik yang digunakan dalam proses seleksi, yaitu seleksi dengan Roda Roulette (*Roulette Wheel Selection*), seleksi berdasarkan Ranking Fitness (*Rank-based Fitness*), seleksi *Stochastic Universal Sampling*, seleksi Lokal (*Local Selection*), seleksi dengan Pemotongan (*Truncation Selection*), seleksi dengan Turnamen (*Tournament Selection*).

d. Perkawinan Silang (*Crossover*)

Proses ini melibatkan dua induk yang akan melakukan perkawinan silang (menggunakan operasi pertukaran, aritmatika) untuk membentuk individu (solusi) baru. Individu baru (keturunan) diharapkan dapat memiliki kualitas yang lebih baik dibandingkan induknya. Proses ini disebut juga sebagai proses rekombinasi. Dalam proses ini memiliki beberapa tahapan [27].

a) Menentukan probabilitas dari perkawinan silang (*crossover*).
b) Memunculkan bilangan acak sebanyak i.
c) Membandingkan bilangan acak tersebut dengan probabilitas *crossover* yang telah ditentukan.

d) Jika bilangan acak ke-i kurang dari nilai probabilitas *crossover* maka terpilih calon induk dan *crossover* dapat dilakukan.

Ada beberapa jenis proses perkawinan silang diantaranya yaitu perkawinan silang diskret, perkawinan silang menengah, perkawinan silang garis, perkawinan silang satu titik, perkawinan silang banyak titik, dan perkawinan silang dengan permutasi.

e. Mutasi

Proses ini berperan untuk menggantikan gen yang hilang akibat dari proses seleksi sehingga dapat juga memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Individu dimutasi dengan menambahkan nilai acak yang sangat kecil dengan probabilitas yang rendah. Jika probabilitas mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Namun bila probabilitas mutasi terlalu besar, maka akan terlalu banyak gangguan acak, sehingga individu baru dapat kehilangan kemiripan dengan induknya. Ada beberapa jenis mutasi diantaranya yaitu mutasi biner, mutasi bilangan real, dan mutasi kromosom permutasi [27].

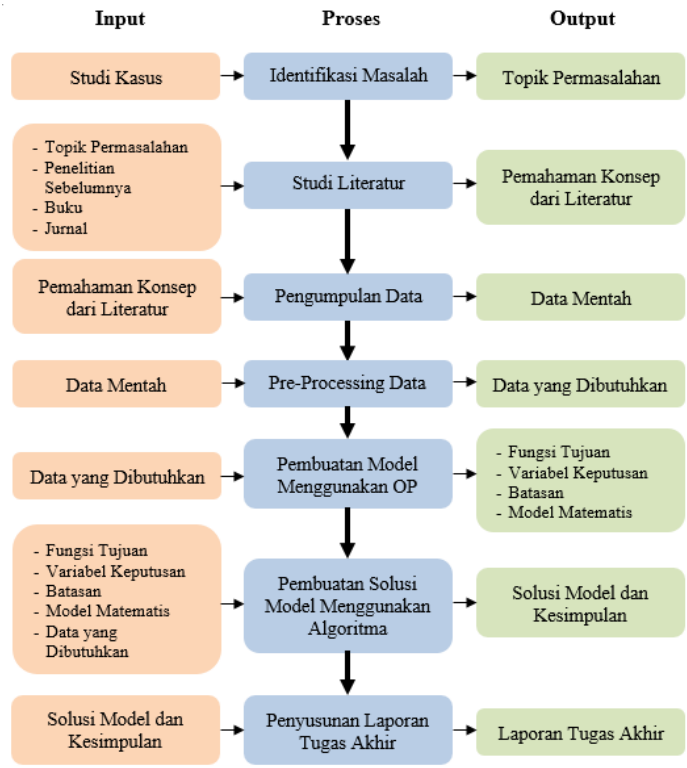
Setelah menyelesaikan semua proses di atas, maka akan terbentuk populasi baru. Populasi baru tersebut akan menjadi populasi awal untuk generasi (iterasi) selanjutnya. Proses tersebut akan terus berulang hingga jumlah parameter generasi sudah sesuai dengan yang ditentukan.

BAB III
METODOLOGI PENELITIAN

Pada bab metode penelitian akan diurutkan tahapan-tahapan apa saja yang dilakukan dalam menyelesaikan penelitian tugas akhir ini. Masing-masing tahapan akan dijelaskan secara rinci. Lalu disertakan jadwal pengerjaan tiap tahapan.

3.1 Tahapan Pelaksanaan Tugas Akhir

Pada sub bab ini akan menjelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi penelitian tersebut dapat dilihat pada Gambar 3.1.



Gambar 3.1 Metodologi Penelitian

3.1.1 Identifikasi Masalah

Proses pertama yang dilakukan adalah mengidentifikasi permasalahan dengan melakukan analisis berdasarkan informasi terkait kemacetan lalu lintas khususnya di Surabaya. Ditemukan bahwa faktor utama yang menjadi penyebab kemacetan adalah pertambahan kendaraan bermotor di Surabaya bertambah sebanyak 17000 unit per bulan dimana setiap bulannya bahkan setiap tahunnya belum tentu bertambah jalan baru [3] [4]. Maka dari itu transportasi umum menjadi solusi yang tepat untuk mengurangi kemacetan.

Selanjutnya diusulkan mikrolet/angkot sebagai objek penelitian dikarenakan transportasi umum ini memiliki rute dan jumlah yang banyak di wilayah Surabaya dan juga adanya rencana pemerintah Surabaya untuk melakukan revitalisasi angkot. Sebagaimana dilansir pada halaman web resmi Dinas Perhubungan Surabaya, jumlah dan rute (trayek) mikrolet di Surabaya pada tahun 2015 masing-masing sebesar 4721 unit dan 58 rute [28]. Pada penelitian ini permasalahan hanya dibatasi untuk enam rute mikrolet.

3.1.2 Studi Literatur

Studi literatur dilakukan dengan mengumpulkan berbagai referensi seperti buku pustaka, penelitian sebelumnya, dan dokumen terkait yang mendukung penyelesaian tugas akhir ini. Studi literatur didasarkan pada topik yang telah dipilih yaitu mengenai optimasi jalur transportasi umum menggunakan metode tertentu untuk membuat model dan mendapatkan solusi berdasarkan model yang dibuat menggunakan algoritma tertentu. Untuk mencari metode tersebut, dilakukan *gap analysis* untuk mencari kelebihan dan kekurangan dari masing-masing metode yang pernah diterapkan dalam kasus yang serupa. Karena karakteristik permasalahan yang sama yaitu adanya batasan waktu dan tidak semua *node* perlu dikunjungi, maka diusulkan metode yang digunakan adalah *Orienteering Problem* (OP). Selanjutnya dilakukan analisis kembali terhadap

algoritma-algoritma yang pernah digunakan untuk mendapatkan solusi dari model yang serupa. Karena *Genetic Algorithm* pernah digunakan untuk pencarian solusi pada model OP dan memberikan hasil yang baik, maka diusulkan algoritma yang digunakan adalah *Genetic Algorithm.*

3.1.3 Pengumpulan Data

Setelah mendapatkan pemahaman konsep terkait metode dan algoritma yang digunakan, dilakukan pengumpulan data trayek berupa enam jalur mikrolet yang digunakan dalam penelitian dengan rincian sebagai berikut.

a. Kalimas Barat/Petekan - Manukan Kulon dengan kode trayek DP.
b. Dukuh Kupang - Benowo dengan kode trayek I.
c. Kalimas Barat - Bratang dengan kode trayek Q.
d. Joyoboyo - Tubanan - Manukan dengan kode trayek TV2.
e. Kalimas Barat - Benowo dengan kode trayek Z.
f. Benowo - Ujung Baru dengan kode trayek Z1.

Berdasarkan enam jalur mikrolet tersebut, melalui Dinas Perhubungan Kota Surabaya dan aplikasi *Google Maps* didapatkan data yang juga diperlukan dalam penelitian sebagai berikut.

a. Data skor yang merupakan jumlah mikrolet yang melewati setiap *node* (tempat pemberhentian mikrolet).
b. Data waktu tempuh di setiap pemberhentian mikrolet.

3.1.4 Pre-Processing Data

Data yang didapatkan pada proses sebelumnya, yaitu data skor dan data hasil pencarian waktu tempuh masih merupakan data mentah. Agar dapat digunakan dalam proses pemodelan, perlu dilakukan *pre-processing* data untuk mengubah data mentah

tersebut menjadi data yang berkualitas. Untuk menjadi data yang berkualitas, pada penelitian ini dilakukan *data transformation* untuk mengubah data mentah tersebut agar sesuai dengan format dari masukan (input) pemodelan dan pencarian solusi model. Sebagai catatan, data skor dan hasil pencarian waktu tempuh akan berbentuk matriks yang detailnya akan dibahas pada sub bab 4.3.

3.1.5 Pembuatan Model Menggunakan OP

Sebelum pembuatan model, ditentukan terlebih dahulu hal-hal yang perlu diperhatikan dalam membuat model.

a. Penentuan *node* awal dan akhir dari permodelan OP. *Node* merepresentasikan setiap tempat pemberhentian mikrolet.
b. Penentuan nilai dari *arc*. *Arc* merepresentasikan waktu tempuh antar *node*.
c. Penentuan skor untuk masing-masing *node* yaitu jumlah angkot yang melewati setiap *node*. Semakin banyak jumlah angkot yang lewat maka semakin besar peluang untuk mendapatkan angkot. Hal ini disesuaikan dengan pengguna yaitu masyarakat Surabaya dan para pengunjung khususnya yang ingin berkeliling mengunjungi banyak tempat di Surabaya menggunakan angkot. Dengan memaksimalkan peluang mendapatkan angkot, maka secara tidak langsung dapat memperkecil waktu tunggu.

Model dibuat berdasarkan variabel-variabel yaitu fungsi tujuan, fungsi batasan, dan variabel keputusan dengan penjabaran sebagai berikut.

a. Variabel Keputusan
   - Kunjungan dari satu tempat ke tempat pemberhentian mikrolet lainnya.

b. Fungsi Tujuan
   - Memaksimalkan jumlah skor yaitu jumlah angkot yang lewat di tiap tempat yang dikunjungi.

c. Fungsi Batasan
   - Penetapan tempat awal dan tujuan akhir.
   - Tiap tempat pemberhentian saling terhubung dan setiap tempat paling banyak dikunjungi satu kali.
   - Adanya kapasitas waktu tempuh.
   - Tidak adanya subtours.

3.1.6 Pembuatan Solusi Model Menggunakan Algoritma

Dalam tahapan ini diterapkan algoritma *Dijkstra* untuk memperbarui pencarian waktu tempuh setiap *node* dan hasil pembaruan tersebut akan dijadikan masukan dari penerapan *Genetic Algorithm* untuk menghasilkan solusi akhir dari model. Penjelasan lebih rincinya dapat dilihat pada sub bab ini.

3.1.6.1 Pembaruan Hasil Pencarian Waktu Tempuh Antar Node

Hasil pencarian waktu tempuh yang dihasilkan pada *pre-processing* data hanya mencari waktu tempuh antar *node* yang terhubung langsung, sedangkan antar *node* yang tidak terhubung secara langsung belum dihitung waktu tempuhnya dimana nilai waktu tempuh tersebut dinyatakan dengan nilai 0. Matriks tersebut masih belum dapat dijadikan masukan untuk *Genetic Algorithm* karena jika dijalankan tidak akan menghasilkan solusi (individu) yang layak sesuai dengan jumlah populasi. Untuk itu setiap nilai nol perlu diubah dengan memberikan nilai baru kepada waktu tempuh antar *node* yang tidak terhubung secara langsung. *Node* yang tidak terhubung secara langsung dapat memiliki lebih dari satu alternatif jalur, maka dari itu ditentukan nilai waktu tempuh baru tersebut diambil berdasarkan jalur dengan waktu tempuh tercepat. Maka dari itu digunakan algoritma *Dijkstra* untuk mencari waktu

tempuh tercepat antar *node* yang tidak terhubung secara langsung. Hasil pencarian waktu tempuh akan diperbarui dimana waktu tempuh tercepat tersebut akan menggantikan nilai nol. Sebagai catatan, hasil pencarian waktu tempuh akan tetap berbentuk matriks yang detailnya akan dibahas pada sub bab 4.3.

3.1.6.2 Pembuatan Solusi Model

Pada tahapan dilakukan pembuatan solusi dari model yang telah dibuat pada tahapan sebelumnya menggunakan *Genetic Algorithm* pada aplikasi NetBeans IDE 8.1 menggunakan bahasa pemrograman Java. Berdasarkan proses dari algoritma tersebut, pada tahapan ini dibagi menjadi beberapa langkah. *Pseudo code* pada Kode 3.1 menunjukkan secara umum bagaimana tahapan-tahapan GA dalam mencari solusi dari model OP [17].

```
Initialize population P of size λ
Evaluate λ individuals in P
While not termination do {
        Select 2*μ individuals from P
        Crossover individuals to produce
        μ offspring
        Mutate some individuals in μ
        Add μ offspring to λ individuals in P
        Evaluate (λ + μ) individuals in P
        Select λ individuals from (λ + μ) individuals in P }
End While
End Algorithm
```

Kode 3.1 Pseudo Code untuk Genetic Algorithm

a. Pendefinisian Individu dan Parameter Pembuatan Solusi

Sebelum menentukan populasi awal, dilakukan terlebih dahulu penentuan individu, dimana individu/kromosom direpresentasikan dalam alternatif rute mikrolet dimana setiap gennya mewakili tempat-tempat (*node*) yang

dikunjungi. Berdasarkan karakteristik dari model OP, maka setiap kromosom memiliki *node* awal dan akhir yang sama. Selain itu ditentukan juga parameter-parameter dalam Genetic Algorithm seperti jumlah populasi, jumlah generasi, probabilitas perkawinan silang, probabilitas mutasi, dan juga Tmax. Setiap parameter tersebut akan mempengaruhi optimumnya solusi akhir yang ditemukan.

b. Pembangkitan Populasi Awal

Untuk membentuk populasi, dipilih secara acak sejumlah individu berdasarkan total waktu tempuh yang tidak boleh melebihi kapasitas waktu tempuh (T_{max}). Pada bab selanjutnya dibuat alur pembangkitan populasi awal dalam bentuk *pseudo code* yang dapat dilihat pada Kode 4.1.

c. Proses Seleksi

Seleksi dilakukan dengan memilih sejumlah individu dari populasi yang akan melanjutkan ke proses berikutnya, untuk dijadikan induk pada proses perkawinan silang. Proses seleksi menggunakan metode *roulette wheel selection* dimana akan diambil sejumlah individu yang *unique* pada populasi. Seleksi tersebut akan memilih individu dari populasi berdasarkan nilai *fitness*. Semakin tinggi nilai fitness maka semakin besar peluang individu tersebut terpilih. Individu-individu yang terbaik dalam kelompok ini akan diseleksi untuk menjadi induk.

d. Proses Perkawinan Silang

Setiap individu yang terpilih pada proses seleksi akan menjadi induk dimana setiap induk akan ditukar gennya dengan cara tertentu untuk mendapat individu baru yang disebut keturunan. Dalam tahap ini dilakukan *modified*

injection crossover pada sejumlah individu yang terpilih di tahap sebelumnya untuk mendapatkan keturunan (solusi) baru berdasarkan probabilitas *crossover* yang telah ditentukan sebelumnya. Proses dilakukan dengan memilih *insert point* pada induk 1 dan sebagian kromosom pada induk 2. Sebagian kromosom pada induk 2 yang terpilih akan dimasukkan ke induk 1 dari *insert point* tersebut untuk mendapatkan *proto child*. *Proto child* tersebut akan disesuaikan ukurannya dengan induk 1 dan didapatkan keturunan hasil perkawinan silang (*offspring*).

e. Proses Mutasi

Proses ini dilakukan pada keturunan hasil perkawinan silang menggunakan pencarian lokal untuk memperkaya populasi dengan menambah (*add*) dan menghilangkan (*omit*) gen pada keturunan (*offspring*) baru yang dibuat pada tahap sebelumnya. Dari proses ini dihasilkan sejumlah keturunan hasil mutasi berdasarkan probabilitas mutasi.

f. Evaluasi Fitness

Pada tahap ini dihitung nilai fungsi *fitness* dengan formulasi tertentu untuk mengevaluasi layak atau tidaknya suatu alternatif rute (individu/kromosom) yang dihasilkan. Fungsi *fitness* merepresentasikan kualitas setiap individu. Individu yang memiliki nilai *fitness* tinggi memiliki kemungkinan yang besar untuk bertahan dalam populasi, sedangkan individu dengan nilai *fitness* rendah memiliki kemungkinan yang kecil untuk bertahan dalam populasi.

g. Pembentukan Populasi Baru

Gabungan dari individu pada populasi awal dan keturunan hasil mutasi akan diseleksi kembali dengan

mengambil individu terbaik (berdasarkan nilai *fitness*) sejumlah parameter jumlah populasi yang telah ditentukan sebelumnya. Populasi baru akan menjadi populasi awal dimana sub bab 3.1.6.2 proses c sampai proses g akan diulang sebanyak parameter jumlah generasi yang telah ditentukan sebelumnya.

h. Pengambilan Solusi Terbaik

Individu terbaik berdasarkan nilai *fitness* hasil populasi dari perulangan terakhir (generasi terakhir) akan terpilih menjadi solusi akhir dari studi kasus permasalahan ini.

3.1.7 Penyusunan Laporan Tugas Akhir

Pada tahapan ini dilakukan penyusunan laporan tugas akhir sebagai bentuk dokumentasi atas terlaksananya tugas akhir ini. Dalam laporan tersebut akan mencakup sebagai berikut.

a. Bab I Pendahuluan

Pada bab pendahuluan akan dijabarkan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan, manfaat, dan relevansi terhadap pengerjaan tugas akhir.

b. Bab II Tinjauan Pustaka

Bab ini akan menjelaskan mengenai penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

c. Bab III Metodologi Penelitian

Pada bab ini dijelaskan mengenai tahapan-tahapan apa saja yang dilakukan dalam pengerjaan tugas akhir ini. Lalu disertakan jadwal pengerjaan tiap tahapan.

d. Bab IV Perancangan

Pada bab ini berisi rancangan penelitian, rancangan bagaimana penelitian akan dilakukan, pemilihan objek penelitian, dan sebagainya.

e. Bab V Implementasi

Pada bab ini berisi proses pelaksanaan penelitian, bagaimana penelitian dilakukan, penerapan strategi pelaksanaan, hambatan, rintangan dalam pelaksanaan, dan sebagainya.

f. Bab VI Analisis dan Pembahasan

Pada bab ini berisi pembahasan tentang penyelesaian permasalahan yang dikerjakan pada penelitian tugas akhir ini.

g. Bab VII Kesimpulan dan Saran

Berisi tentang kesimpulan dan saran yang ditujukan untuk kelengkapan penyempurnaan tugas akhir ini.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan bagaimana rancangan dari penelitian tugas akhir yang meliputi subjek dan objek dari tugas akhir, pemilihan subjek dan objek tugas akhir serta bagaimana tugas akhir akan dilakukan.

4.1. Pengumpulan dan Deskripsi Data

Untuk memudahkan proses pemodelan menggunakan *Orinteering Problem* dibutuhkan perancangan *network model* dalam bentuk *graph*. Data-data yang diproses ke dalam perancangan *network model* merupakan data yang diperoleh dari website resmi Dinas Perhubungan Kota Surabaya meliputi data trayek dan jumlah angkot dengan penjelasan sebagai berikut.

4.1.1. Deskripsi Data Trayek Angkot

Setiap angkot memiliki lintasan, pangkalan awal dan akhir yang tetap sesuai dengan kode trayeknya masing-masing. Data trayek angkot akan digunakan untuk menentukan *node* pada perancangan *network model*. Dalam penelitian ini dipilih enam trayek yang saling bersinggungan sehingga dalam *network model* nantinya setiap *node* akan saling terhubung. Data trayek angkot digambarkan ke dalam Tabel 4.1 yang berisikan kode trayek dan rute.

Tabel 4.1 Data Trayek Angkot

Kode Trayek	Rute
DP	**Berangkat:** Petekan – JMP – Jl. Niaga – Jl. Veteran – Jl. Kebonrojo – Jl. Indrapura – Jl. Rajawali – Jl. Gresik – Jl. Demak – Jl. Kalibutuh – Jl. Tembok Sayuran – Jl. Tidar – Jl. Pacuan Kuda – Jl. Petemon Sidomulyo

Kode Trayek	Rute
	IV – Jl. Petemon II – Jl. Simo Kewagean – Jl. Banyu Urip – Jl. Simo Magersari – Jl. Raya Simo Gunung – Jl. Kupang Jaya – Jl. Sukomanunggal Jaya – Jl. Kupang Baru – Jl. Darmo Baru Barat XII – Jl. Darmo Permai III – Jl. Darmo Permai II – U Turn – Jl. HR. Muhammad – Jl. Darmo Permai Selatan – Jl. Simpang Darmo Permai Utara – – Jl. Simpang Darmo Permai Selatan XV – Jl. Pradah Indah – PTC – Jl. Raya Lontar – Jl. Sambi Kerep – Jl. Kuwukan – Jl. Jilidro – Jl. Wonorejo – Pangkalan Manukan Kulon **Kembali:** Pangkalan Manukan Kulon – Jl. Wonorejo – Jl. Jlidro – Jl. Kuwukan – Jl. Sambikerep – Jl. Raya Lontar – PTC – Jl. Pradah Indah – Jl. Simpang Darmo Permai Selatan – Jl. Simpang Darmo Permai Utara – Jl. Darmo Permai II – Jl. Darmo Permai III – Jl. Darmo Baru Barat XII – Jl. Sukomanunggal Jaya – Jl. Kupang Jaya – Jl. Kupang Baru – Jl. Simo Gunung – Jl. Simo Magersari – Jl. Banyu Urip – Jl. Simo Kewagean – Jl. Petemon II – Jl. Petemon Sidomulyo IV – Jl. Pacuan Kuda – Jl. Tidar – Jl. Tembok Sayuran – Jl. Kalibutuh – Jl. Demak – Jl. Gresik – Jl. Ikan Dorang – Jl. Ikan Kakap – Jl. Perak Barat – U. Turn – Jl. Perak Timur – Jl. Rajawali – Jl. Kasuari – JMP – Petekan.
I	**Berangkat:** Pangkalan Kupang – Jl. Raya Diponegoro – Jl. Banyu Urip – Jl. Simo Magersari – Jl. Simo Kalangan – Jl. Simo Pomahan – Jl. Simo Tambakan – Jl. Simo Jawar – Jl. Raya Sukomanunggal – Jl. Raya Tanjung Sari – Jl. Raya Tandes – Jl. Raya Tandes Kidul – Jl. Raya Tandes Lor – Jl. Raya Balongsari – Jl. Raya Bibis – Jl. Raya Manukan Wetan – Jl. Raya Manukan Kulon – Jl. Raya

Kode Trayek	Rute
	Manukan Madya – Jl. Raya Manukan Krajan – Jl. Manukan Tama – Jl. KH. Amin – Jl. Raya Manukan Kulon – Jl. Banjar Sugihan – Jl. Raya Kandangan – Jl. Moroseneng – Jl. Sememi Jaya – Jl. Sememi – Jl. Raya Babat Jerawat – Jl. Pakal – Jl. Pakal – Jl. Raci – Jl. Benowo (Pangkalan Akhir) **Kembali**: Pangkalan Benowo – Jl. Benowo – Jl. Raci – Jl. Pakal – Jl. Raya Babat Jerawat – Jl. Sememi – Jl. Sememi Jaya – Jl. Moroseneng – Jl. Raya Kandangan – Jl. Banjar Sugihan – Jl. Raya Manukan Kulon – Jl. Raya Manukan Wetan – Jl. Bibis – Jl. Raya Balongsari – Jl. Raya Tandes Lor – Jl. Raya Tandes Kidul – Jl. Raya Tandes – Jl. Raya Tanjungsari – Jl. Raya Sukomanunggal – Jl. Simo Jawar – Jl. Simo Tambakaan – Jl. Simo Pomahan – Jl. Simo Kalangan – Jl. Simo Magersari – Jl. Banyu Urip – Jl. Pasar Kembang – U Turn – Jl. Raya Diponegoro – Jl. Kembang Kuning – Pangkalan Kupang (Pangkalan Akhir).
Q	**Berangkat:** Terminal Bratang – Jl. Barata Jaya – Jl. Bratang Binangun – Jl. Ngagel Jaya Selatan – Jl. Bung Tomo – Jl. Ratna – Jl. Upajiwo – Jl. Ngagel – Jl. Darmo Kali – U Turn – Jl. Dinoyo – Jl. Polisi Istimewa – Jl. Dr. Sutomo – Jl. Diponegoro – U Turn – Jl. Pasar Kembang – Jl. Kedung Doro – Jl. Tidar – Jl. Arjuna – Jl. Semarang – Jl. Pasar Turi – Jl. Bubutan DKA – Jl. Indrapura – Jl. Krembangan Barat – Jl. Krembangan Timur – Jl. Rajawali – Jl. Kasuari -Jl. Kalimas Barat – Pangkalan Kalimas Barat (Petekan) (Pangkalan Akhir). **Kembali**: Pangkalan Kalimas Barat (Petekan) – Jl. Kalmas Barat – Jl. Kasuari – Jl. Garuda – Jl. Taman

Kode Trayek	Rute
	Jayengrono – Jl. Jembatan Merah – Jl. Veteran – Jl. Pahlawan – Jl. Tembaan – Jl. Bubutan – Jl. Pasar Turi – Jl. Semarang – Jl. Arjuna – Jl. Tidar – Jl. Kedungdoro – Jl. Diponegoro – Jl. Dr. Sutomo – Jl. Polisi Istimewa – Jl. Dinoyo -Jl. Bung Tomo – Jl. Ngagel jaya Selatan – Jl. Manyar – Jl. Nginden – Jl. U Turn – Terminal Bratang (Pangkalan Akhir).
TV2	**Berangkat:** Terminal Joyoboyo – Jl. Raya Darmo – Jl. Raya Diponegoro – Jl. Kutai – Jl. Adityawarman – Jl. Mayjen Sungkono – Bundaran Tol – Jl. HR. Muhammad – Jl. Darmo Permai Selatan – Jl. Simpang Darmo Permai Utara – Jl. Tubanan – Jl. Gadel – Jl. Balongsari Tama (Diklat) – Jl. Balongsari Tama Tengah – Jl. Lempung Indah – Jl. Lempung Tama – Jl. Manukan Dalam – Jl. Manukan Tama – Jl. Manukan Kulon (Pangkalan Akhir). **Kembali**: Terminal Manukan Kulon – Jl. Manukan Tama (SMU XI) – Jl. Manukan Dalam – Jl. Lempung Tama – Jl. Lempung Indah – Jl. Balongsari Tama Tengah – Jl. Balongsari Tama (Diklat) – Jl. Gadel – Jl. Tubanan – Jl. Simpang Darmo Permai Utara – Jl. Darmo Permai Selatan – Jl. HR. Muhammad – Bundaran Tol – Jl. Mayjen Sungkono – Jl. Adityawarman – Jl. Hayam Wuruk – Jl. Brawijaya – Terminal Joyoboyo.
Z	**Berangkat:** Pangkalan JMP – Jl. Veteran – Jl. Kebon Rojo – Jl. Indrapura – Jl. Rajawali – Jl. Gresik – Jl. Gresik Gadukan – Jl. Gresik Tambak Asri – J.l Kalianak Timur – Jl. Kalianak Barat – Jl. Greges Timur – Jl Jreges Barat – Jl. Margomulyo – Jl. Raya Balongsari – Jl. Bibis – Jl. Raya Manukan Wetan – Jl. Manukan Kulon – Jl. Manukan Krajan – Jl. Manukan Tama –

Kode Trayek	Rute
	Jl. KH Amir – Jl. Raya Manukan Kulon – Jl. Manukan Krajan – Jl. Manukan Tama – Jl. Manukan Wetan – Jl. Raya Manukan Kulon – Jl. Banjar Sugihan – Jl. Raya Kandangan – Jl. Moro Seneng – Jl. Sememi Jaya – Jl. Raya Babat Jerawat – Jl. Raya Raci – Jl. Raya Benowo (pangkalan akhir). **Kembali**: Pangkalan Benowo – Jl. Benowo – Jl. Raya Raci – Jl. Raya Babat Jerawat – Jl. Raya Sememi – Jl. Sememi Jaya – Jl. Moroseneng – Jl. Raya Kandangan – Jl. Banjar Sugihan – Jl. Raya Manukan Kulon – Jl. Raya Manukan Wetan – Jl. Bibis – Jl. Raya Balongsari – Jl. Margomulyo – Jl. Greges Barat – Jl. Greges Timur – Jl. Kalianak Barat – Jl. Kalianak Timur – Jl. Gresik Tambak Asri – Jl. Gresik Gadukan – Jl. Gresik – Jl. Ikan Dorang – Jl. Ikan Kakap – Jl. Perak Barat – Jl. Perak Timur – Jl. Rajawali – Jl. Kasuari – Jl. Taman Jayengrono – JMP (Pangkalan Akhir)
Z1	**Berangkat:** Pangkalan Pangkalan Benowo – Jl. Benowo – Jl. Raci Benowo – Jl. Raya Pakal – Jl. Mulyo Mukti – Jl. Raya Babat Jerawat – Jl. Raya Sememi – Jl. Moro Seneng – Jl. Raya Kandangan – Jl. Raya Banjar Sugihan – Jl. Raya Manukan Kulon – Jl. Raya Bibis – Jl. Balongsari – Jl. Raya Tandes Lor – Jl. Raya Tandes Kidul – Jl. Tanjung Sari – Jl. Tambak Mayor – Jl. Dupak Rukun – Jl. Gresik – Jl. Ikan Lumba-lumba – Jl. Tanjung Sadari – Jl. Tanjung Priuk – Jl. Nilam Barat – Jl. Prapat Kurung Utara – Jl. Tanjung Perak Barat – Putar – Jl. Tanjung Perak Timur – Jl. Prapat Kurung Utara – Jl. Kalimas Baru – Pangkalan Ujung Baru (Pangkalan Akhir). **Kembali**:

Kode Trayek	Rute
	Pangkalan Ujung Baru – Jl. Perak Timur – Putar – Jl. Perak Barat – Jl. Prapat Kurung Selatan – Jl. Laksad M Nasir – Jl. Tanjung Priuk – Jl. Perak Barat – Putar – Jl. Perak Timur – Jl. Gresik – Jl. Demak – Jl. Dupak Rukun – Jl. Pasar Loak – Jl. Dupak Rukun – Jl. Tambak Mayor – Jl. Tanjung Sari – Jl. Raya Tandes Kidul – Jl. Raya Tandes Lor – Jl. Raya Balongsari – Jl. Raya Bibis – Jl. Raya Manukan Kulon – Jl. Raya Banjar Sugihan – Jl. Raya Kandangan – Jl. Moro Seneng – Jl. Sememi – Jl. Raya Babat Jerawat – Jl. Mulyo Mukti – Jl. Raya Pakal – Jl. Raci Benowo – Jl. Raya Benowo – Pangkalan Benowo (Pangkalan Akhir)

4.1.2. Deskripsi Data Jumlah Angkot

Dinas Perhubungan telah mencatat setiap jumlah angkot yang lewat dalam setiap rute trayek berdasarkan kodenya masing-masing. Data jumlah angkot akan digunakan sebagai penentu nilai skor untuk masing-masing *node*. Data jumlah angkot digambarkan ke dalam Tabel 4.2 yang berisikan kode trayek, asal-tujuan, dan jumlah angkot.

Tabel 4.2 Data Jumlah Angkot

Kode Trayek	Asal - Tujuan	Jumlah Angkot
DP	Kalimas Barat / Petekan - Manukan Kulon (PP)	84
I	Dukuh Kupang - Benowo (PP)	112
Q	Kalimas Barat - Bratang (PP)	115
TV2	Joyoboyo - Tubanan - Manukan (PP)	145
Z	Kalimas Barat - Benowo (PP)	129
Z1	Benowo - Ujung Baru (PP)	119

4.2. Perancangan Network Model

Network model digunakan untuk mempermudah pemodelan menggunakan *Orienteering Problem*. Berdasarkan data trayek angkot dan jumlah angkot, perancangan *network model* terdiri dari berbagai proses sebagai berikut.

Tabel 4.3 Asumsi Penggambaran Rute

Trayek	Keterangan
Z	**Asumsi:** Rute "Jl. Manukan Krajan – Jl. Manukan Tama – Jl. KH Amir – Jl. Raya Manukan Kulon – Jl. Manukan Krajan – Jl. Manukan Tama" dihilangkan. **Alasan:** Seharusnya setelah Jl. Manukan Kulon langsung menuju Jl. Banjar Sugihan sesuai dengan peta jalur yang ditunjukkan oleh Dishub. Rute yang dihilangkan juga tidak saling bertemu.
Z1	**Asumsi:** Jalan Ikan Lumba-Lumba pada rute trayek dihilangkan. **Alasan:** Seharusnya dari Jl. Demak langsung menuju Jl. Tanjung Sadari dengan menyeberangi Jl. Gresik Gradukan Timur dan tidak melewati Jl. Ikan Lumba-Lumba. Jalan Ikan Lumba-Lumba juga tidak terhubung dengan Jl. Demak maupun Jl. Tanjung Sadari.
TV2	**Asumsi:** Rute **"**Jl. Darmo Permai Selatan – Jl. Simpang Darmo Permai Utara – Jl. Tubanan – Jl. Gadel" diganti dengan rute "Jl. Mayjen Yono Suwoyo - Jl. Raya Darmo Permai III - Jl. Pattimura - Jl. Raya Sukomanunggal Jaya - Jl. Raya Satelit Indah - Jl. Raya Satelit Utara - Jl. Raya Darmo Indah Barat" **Alasan:** Deskripsi arahan yang ditunjukkan tidak jelas sehingga diganti dengan rute yang digambarkan dalam peta jalur trayek TV2 pada website resmi Dishub Surabaya.

Trayek	Keterangan
I	**Asumsi:** Penambahan rute berangkat "Jl. Raya Dukuh Kupang Barat - Kl. Mayjen Sungkono - Jl. Indragiri" sebelum Jl. Diponegoro. **Alasan:** Mengikuti peta jalur trayek I pada website resmi Dishub Surabaya.

4.2.1. Penggambaran Rute Trayek

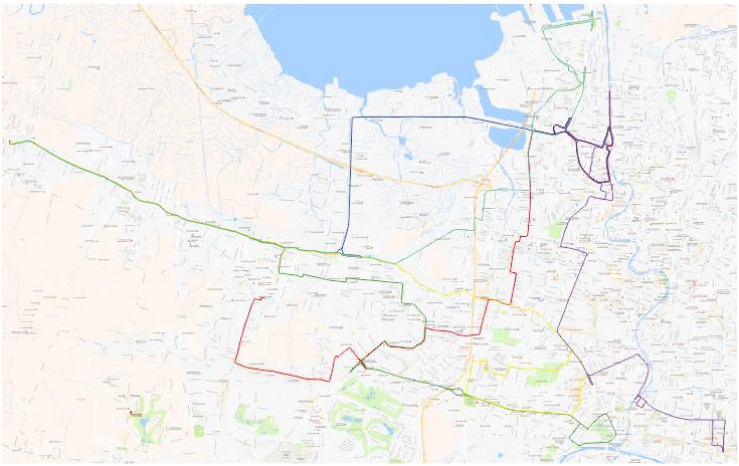
Berdasarkan data trayek angkot, dibuatlah gambar masing-masing lintasan yang dilalui oleh enam trayek yang telah dipilih. Dalam penggambaran rute trayek terdapat beberapa ketidaksesuaian antara data trayek dengan kondisi yang ada di lapangan. Maka dari itu dibuatlah asumsi-asumsi seperti pada Tabel 4.3.

Dengan adanya asumsi tambahan pada Tabel 4.3, maka dibuat penggambaran jalur dimana masing-masing kode trayek memiliki warna yang berbeda seperti rincian pada Tabel 4.4.

Tabel 4.4 Warna Kode Trayek

Kode Trayek	Warna
DP	Merah
I	Kuning
Q	Ungu
TV2	Hijau Tua
Z	Biru Tua
Z1	Hijau Muda

Secara keseluruhan, lintasan dari keenam trayek dapat dilihat pada Gambar 4.1.



Gambar 4.1 Penggambaran Trayek ke dalam Peta

4.2.2. Penentuan Node, Nilai Arc, dan Skor Tiap Node

Berdasarkan penggambaran lintasan dari keenam trayek maka ditentukan sejumlah *node*, nilai *arc*, dan skor dengan asumsi sebagai berikut.

a. Setiap lintasan trayek yang saling bersinggungan akan menjadi *node* yang merupakan representasi dari tempat pemberhentian angkot. Agar sesuai dengan kenyataan yang ada di lapangan, n*ode* tidak ditempatkan tepat di persimpangan jalan. Hal ini bertujuan untuk mencegah timbulnya kemacetan lalu lintas.
b. Dikarenakan keinginan dari pengguna (masyarakat Surabaya dan pengunjung) yang ingin berkeliling ke berbagai tempat di Surabaya, maka penempatan *node* diprioritaskan di tempat yang berpotensi padat pengunjung, misalnya tempat wisata, pusat perbelanjaan, bank, dan kantor.
c. Jarak antar *node* (nilai *arc*) tidak boleh kurang dari dua ratus meter (> 200 m) sehingga jalan yang memiliki panjang kurang dari 200 meter tidak

memiliki *node*. Hal ini diadaptasi dari Keputusan Direktur Jenderal Perhubungan Darat Nomor 271/HK.105/DJRD/96 dimana jarak minimum antara halte dan/atau Tempat Perhentian Bus (TPB) adalah 200-300 meter untuk daerah pusat kegiatan sangat padat.

d. Jarak antar *node* (nilai *arc*) tidak boleh lebih dari satu kilo meter (< 1 km) sehingga jalan yang memiliki panjang lebih dari 1 kilo meter akan memiliki 2 atau lebih *node*. 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 500-1000 meter untuk daerah pinggiran.
e. Skor tiap *node* dihitung berdasarkan jumlah angkot dari keenam trayek yang melewati *node* tersebut. Hal ini bertujuan untuk memperbesar peluang mendapatkan angkot.

Tabel 4.5 Contoh Deskripsi Node

No de	Deskripsi	Keterangan Tambahan
1	Jl. Kalimas Barat No.63	Pangkalan Petekan, dekat AML (Asia Mandiri Lines)
2	Jl. Kalimas Barat No.45	Dekat PT Rajawali Nusindo
3	Jl. Kalimas Barat No.3	Pangkalan JMP, dekat pertigaan Jl. Kalimas Barat dan Jl. Kasuari
4	Jl. Kasuari No.12	Dekat Jembatan Merah Plaza (JMP)
5	Jl. Garuda No.2	Dekat Jembatan Merah Plaza (JMP)

Dari asumsi tersebut didapatkan 191 *node* dan tersebar di setiap lintasan trayek angkot yang digambarkan ke dalam bentuk titik-titik hitam seperti pada LAMPIRAN G. Berikut Tabel 4.5 merupakan contoh deskripsi dari penentuan masing-masing

node berupa alamat dan keterangan tambahan. Secara lengkapnya deksripsi setiap *node* ada pada LAMPIRAN A. Masing-masing skor di setiap *node* ditunjukkan pada Tabel 4.6.

Tabel 4.6 Daftar Skor Tiap Node

Node	Skor	Node	Skor	Node	Skor	Node	Skor
1	199	50	112	99	84	148	145
2	199	51	112	100	84	149	145
3	328	52	112	101	84	150	145
4	328	53	112	102	84	151	145
5	213	54	112	103	203	152	145
6	328	55	112	104	203	153	145
7	328	56	112	105	203	154	145
8	328	57	112	106	203	155	84
9	328	58	257	107	332	156	84
10	115	59	257	108	119	157	84
11	115	60	257	109	129	158	84
12	213	61	257	110	129	159	84
13	213	62	257	111	129	160	84
14	213	63	257	112	129	161	84
15	213	64	112	113	129	162	84
16	332	65	112	114	129	163	84
17	213	66	145	115	129	164	84
18	332	67	145	116	129	165	84
19	115	68	145	117	129	166	84
20	115	69	145	118	129	167	505
21	115	70	145	119	129	168	360
22	115	71	145	120	129	169	360
23	115	72	145	121	129	170	360
24	115	73	145	122	129	171	360
25	115	74	145	123	129	172	360

Node	Skor
26	115
27	115
28	115
29	115
30	227
31	227
32	227
33	115
34	115
35	115
36	115
37	115
38	115
39	115
40	115
41	115
42	115
43	115
44	115
45	115
46	115
47	115
48	115
49	112

Node	Skor
75	145
76	145
77	145
78	145
79	229
80	145
81	229
82	84
83	229
84	229
85	229
86	229
87	84
88	84
89	84
90	84
91	196
92	196
93	112
94	112
95	112
96	84
97	84
98	84

Node	Skor
124	360
125	231
126	231
127	231
128	112
129	112
130	112
131	112
132	119
133	119
134	119
135	119
136	119
137	119
138	119
139	119
140	119
141	360
142	360
143	505
144	145
145	145
146	145
147	145

Node	Skor
173	360
174	360
175	360
176	360
177	360
178	360
179	119
180	119
181	119
182	119
183	119
184	119
185	119
186	119
187	119
188	119
189	119
190	119
191	119

Sedangkan nilai dari *arc* yang menghubungkan setiap *node* ditunjukkan dengan contoh seperti pada Tabel 4.7. Secara lengkapnya deksripsi nilai *arc* yang menghubungkan setiap *node* ada pada LAMPIRAN B.

Tabel 4.7 Contoh Daftar Nilai *Arc* antar *Node*

Node Awal	Node Akhir	Waktu (menit)	Trayek
1	2	1	DP, Q
2	1	1	DP, Q
2	3	1	DP, Q
3	2	1	DP, Q
3	4	1	Q

4.3. Pre-Processing Data

Data yang dibentuk pada tahap 4.2 Perancangan Network Model yaitu daftar nilai *arc* (waktu tempuh) dan skor masih merupakan data mentah. Pada proses ini dilakukan pengolahan data waktu tempuh dan skor masing-masing *node* untuk menjadi data yang berkualitas sebagai masukan (*input*) untuk *Genetic Algorithm* agar dapat dijalankan menggunakan bahasa pemrograman *Java*. Proses ini terdiri dari dua tahapan sebagai berikut.

4.3.1. Pembentukan Matriks Waktu Tempuh

Hasil rekapan data waktu tempuh (nilai *arc*) antar node yang ditunjukkan pada Tabel 4.7 tidak dapat dibaca sebagai input dengan *Java*. Untuk itu, perlu dilakukan *pre-processing* data dengan *data transformation* agar data tersebut dapat digunakan sebagai *input* dalam pencarian solusi dari model OP menggunakan *Genetic Algorithm*.

Dari Tabel 4.7 dilakukan *data transformation* dengan mengubah bentuknya ke dalam matriks dua dimensi pada *Microsoft Office Excel*. Matriks ini akan berukuran sesuai dengan jumlah *node* yang ada sehingga dalam kasus ini dibuat matriks dengan jumlah baris dan kolom sebanyak 191. Matriks tersebut merepresentasikan hubungan tiap *node* sesuai dengan *network model* yang telah dibuat sebelumnya. Setiap *cell* berisikan waktu tempuh antar *node* dimana pada tahap ini hanya

dimasukkan waktu tempuh antar *node* yang saling terhubung secara langsung sesuai dengan Tabel 4.7. Berikut merupakan potongan dari matriks waktu tempuh yang telah dibuat.

Tabel 4.8 menunjukkan potongan matriks 10×10 dimana isi dari matriks tersebut merupakan waktu tempuh antar *node* yang saling terhubung langsung dalam satuan menit. Matriks dua dimensi tersebut dibaca dari baris ke kolom. Dicontohkan seperti kotak merah pada Tabel 4.8, waktu tempuh dari *node* 5 ke *node* 3 adalah 1 menit. Setiap *cell* yang memiliki nilai 0 menunjukkan bahwa kedua *node* tersebut tidak saling terhubung secara langsung, yaitu harus melalui *node* lain untuk menghubungkan *node* tersebut. Begitu juga sebaliknya, setiap *cell* yang berisikan nilai lebih dari 0 menunjukkan node tersebut saling terhubung secara langsung. Nilai 0 tersebut nantinya akan digantikan dengan nilai waktu terpendek pada tahapan selanjutnya.

Tabel 4.8 Potongan Awal Matriks Waktu Tempuh

node	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	0	1	0	1	2	0	0	0	0	0
5	0	0	1	0	0	0	1	0	0	0
6	0	0	0	0	0	0	2	0	0	0
7	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	1

Setelah semua nilai waktu tempuh dimasukkan ke dalam matriks berukuran 191x191, selanjutnya matriks disimpan ke dalam format .csv (*comma delimited*). Keterangan *node* seperti pada Tabel 4.8 yang berada di luar tabel/matriks tidak perlu dimasukkan ke dalam dokumen .csv.

4.3.2. Pembentukan Matriks Skor

Sama seperti sebelumnya, menggunakan *Microsoft Office Excel*, data skor setiap *node* yang ada pada Tabel 4.6 perlu dilakukan *data transformation* dengan mengubah bentuknya ke dalam matriks satu dimensi agar dapat terbaca oleh *Java*. Matriks ini berisi 191 baris sesuai dengan jumlah *node*. Setiap baris pada matriks ini berisikan skor pada masing-masing *node*. Berikut merupakan potongan dari matriks skor yang telah dibuat.

Tabel 4.9 merupakan potongan matriks dengan ukuran 10 baris dimana isi dari matriks tersebut merupakan skor setiap *node* yang merepresentasikan jumlah angkot yang melewati *node* tersebut. Dicontohkan seperti kotak merah pada tabel di atas, skor untuk *node* 9 adalah 328. Sebanyak 191 *node*/baris akan terisi dengan skornya masing-masing.

Tabel 4.9 Potongan Matriks Skor

node	
1	199
2	199
3	328
4	328
5	213
6	328
7	328
8	328
9	328
10	115

Setelah semua nilai skor dimasukkan ke dalam matriks, selanjutnya matriks berukuran satu dimensi tersebut disimpan ke dalam format .csv (*comma delimited*). Keterangan *node* seperti pada Tabel 4.9 yang berada di luar tabel/matriks tidak perlu dimasukkan ke dalam dokumen .csv. Tidak seperti

matriks waktu tempuh, pada matriks skor tidak perlu dilakukan *data cleaning* karena semua *cell* telah terisi.

4.4. Formulasi Model Orienteering Problem

Berdasarkan *network model* dan hasil dari *pre-processing* data yang telah dibuat sebelumnya, dibentuklah model yang sesuai menggunakan bentuk model *Orienteering Problem* (OP). Model OP diformulasikan ke dalam pemrograman integer dengan penentuan variabel keputusan, fungsi tujuan, dan batasan sebagai berikut.

4.4.1. Variabel Keputusan

Permasalahan yang akan diselesaikan dalam tugas akhir ini adalah mengetahui rute terpendek yang harus dilalui dari suatu node ke node lainnya. Sehingga variabel keputusannya adalah kunjungan dari suatu node ke node lainnya (x_{ij}). Variabel keputusan $x_{ij} = 1$ jika kunjungan ke *node* i diikuti dengan kunjungan ke *node* j, dan jika tidak maka $x_{ij} = 0$.

Tabel 4.10 Contoh Variabel Keputusan

Variabel	Keterangan
x1.2	Kunjungan dari node 1 ke node 2
x2.1	Kunjungan dari node 2 ke node 1
x2.3	Kunjungan dari node 2 ke node 3
x3.2	Kunjungan dari node 3 ke node 2
x3.4	Kunjungan dari node 3 ke node 4

Dicontohkan $x_{12} = 1$ (pada studi kasus ini, ditulis sebagai x1.2 = 1) menunjukkan kunjungan ke *node* 1 diikuti dengan ke *node* 2 dan $x_{54} = 0$ (pada studi kasus ini, ditulis sebagai x5.4 = 1) menunjukkan kunjungan ke *node* 5 tidak diikuti dengan kunjungan ke *node* 4. Sebagai contoh, variabel keputusan dapat dilihat pada Tabel 4.10. Secara lengkapnya variabel keputusan ada pada LAMPIRAN C.

4.4.2. Fungsi Tujuan

Dalam penelitian ini, seperti model OP pada umumnya, tujuan yang ingin dicapai adalah mencari rute dengan jumlah skor maksimum. Dalam pembuatan fungsi tujuan, ditentukan terlebih dahulu *node* awal yaitu *node* 1 (Jl. Kalimas Barat No.63, Pangkalan Petekan) dan *node* akhir yaitu *node* 166 (Jl. Wonorejo No.18, Pangkalan Manukon). Fungsi tujuan dalam model OP adalah sebagai berikut.

$$Max \sum_{i=1}^{N-1} \sum_{j=2}^{N} S_i x_{ij}$$

Sehingga pada studi kasus ini, fungsi tujuan tersebut ditulis dalam persamaan berikut.

$$Max \sum_{i=1}^{165} \sum_{j=2}^{166} S_i x_{ij}$$

Dimana,
S_i = Skor dari *node* i
x_{ij} = Kunjungan dari *node* i ke *node* j
i = 1, 2, 3, …, 165
j = 2, 3, 4, …, 166

4.4.3. Perumusan Batasan

Batasan-batasan yang ada dalam model antara lain sebagai berikut.

Batasan 1: Rute dimulai dari *node* 1 dan berakhir di *node* 166. Rute yang terpilih pastinya merupakan jalur yang terhubung dengan *node* awal dan *node* akhir, sehingga variabel keputusan yang terkait dengan kunjungan dari *node* 1 dan kunjungan ke *node* 166 pasti bernilai satu. Batasan 1 dalam model OP adalah sebagai berikut.

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1$$

Sehingga pada studi kasus ini, batasan tersebut ditulis dalam persamaan berikut.

- Batasan *node* awal

$$\sum_{j=2}^{166} x_{1j} = 1$$

- Batasan *node* akhir

$$\sum_{i=1}^{165} x_{i166} = 1$$

Batasan 2: Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali.
Terhubungnya semua *node* dalam suatu *network model* ditunjukkan dari adanya percabangan dan pertemuan antar *node*. Batasan 2 dalam model OP adalah sebagai berikut.

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

Sehingga pada studi kasus ini, batasan tersebut ditulis dalam persamaan berikut.

- Batasan untuk pertemuan antar node

$$\sum_{i=1}^{165} x_{ik} \leq 1;\ \forall k = 2, \dots, 165$$

- Batasan untuk percabangan antar node

$$\sum_{j=2}^{166} x_{kj} \leq 1;\ \forall k = 2, \ldots, 165$$

Batasan 3: Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan.
Penentuan besarnya batasan total waktu tempuh (tMax) adalah fleksibel, tergantung keinginan dari pengguna yaitu masyarakat Surabaya dan para pengunjung khususnya yang ingin berkeliling mengunjungi banyak tempat di Surabaya menggunakan angkot. Sehingga pada studi kasus, sebagai contoh, ditentukan terlebih dahulu bahwa tMax untuk mencari rute dari node 1 ke node 166 adalah maksimum 70 menit. Ini menunjukkan bahwa pengguna ingin berkeliling dari node 1 ke node 166 dengan maksimal waktu 70 menit. Batasan 3 dalam model OP adalah sebagai berikut.

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ij} \leq T_{max},$$

Maka pada studi kasus ini, batasan tersebut ditulis dalam persamaan berikut.

$$\sum_{i=1}^{165} \sum_{j=2}^{166} t_{ij} x_{ij} \leq 70$$

Batasan 4 dan 5: Tidak boleh ada *subtours* dimana lintasan dimulai dan berakhir pada *node* yang sama. Batasan ini sudah tergambarkan dalam batasan lainnya. Sehingga secara tidak langsung, jika batasan lain terpenuhi, maka batasan 4 juga terpenuhi.

4.5. Penentuan Parameter Variabel GA

Pada tahap ini merupakan proses menentukan parameter untuk beberapa variabel yang mempengaruhi solusi model OP dari *Genetic Algorithm* (GA). Proses penentuan parameter tersebut antara lain nilai dari probabilitas perkawinan silang (*crossover*) dan probabilitas mutasi.

4.5.1. Probabilitas Perkawinan Silang (Crossover)

Probabilitas ini akan mempengaruhi jumlah proses perkawinan silang yang terjadi antara sesama individu. Nilai dari parameter tersebut berupa angka persentase. Untuk mendapatkan hasil yang optimum, maka biasanya probabilitas perkawinan silang yang digunakan berkisar diantara 0.8-0.95. [29]

4.5.2. Probabilitas Mutasi

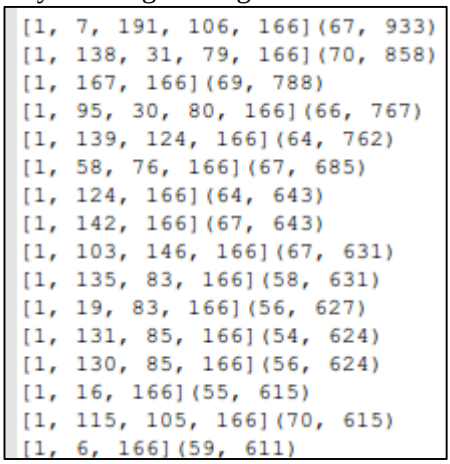
Probabilitas ini akan mempengaruhi jumlah proses mutasi yang terjadi dalam suatu populasi. Nilai dari parameter tersebut berupa angka persentase. Untuk mendapatkan hasil yang optimum, maka rekomendasi probabilitas mutasi yang digunakan nantinya berkisar diantara 0.05-0.1. [29]

4.6. Penentuan Komponen GA

Pada tahap ini dilakukan persiapan agar model OP dapat digunakan dalam proses penyelesaian dengan menggunakan *Genetic Algorithm* (GA). Proses persiapan tersebut terdiri dari penentuan beberapa komponen GA yaitu:

- Populasi
  Pada kasus ini, populasi merupakan sekumpulan solusi dari rute yang dapat dilewati dari *node* awal menuju *node* akhir. Kumpulan dari kromosom/individu membentuk populasi. Kromoson/indivu terdiri dari susunan *node* terurut yang dijalankan dengan bahasa pemrograman Java. Susunan *node* yang terurut bertujuan untuk menjaga node awal dan akhir tetap sama untuk setiap

kromosom. Gambar 4.2 merupakan salah satu contoh populasi yang terdiri dari 16 kromosom dengan *node* awal adalah 1 dan *node* akhir adalah 166. Kromosom memiliki keterangan waktu tempuh dan skornya masing-masing.



Gambar 4.2 Contoh Populasi

Dalam suatu populasi, panjang kromosom dapat berbeda. Namun setiap kromosom yang terdapat dalam suatu populasi harus memiliki node awal dan akhir yang tetap seperti Gambar 4.2.

- Individu/Kromosom

  Pada kasus ini, individu merupakan kromosom yang berisi solusi berupa lintasan yang terdiri dari *node* yang terurut. Setiap *node* yang terdapat pada sebuah kromosom merupakan *node* yang harus dilalui/dikunjungi untuk menuju *node* akhir. Setiap

[*node* awal, *node* mana saja yang dikunjungi, *node* akhir] (waktu tempuh, skor)

Contoh:
[1, 95, 30, 80, 166] (66, 767)

kromosom dalam populasi memiliki format dengan contoh seperti pada Gambar 4.3.

Gambar 4.3 Contoh Kromosom

Kromosom pada Tabel 4.4 merupakan sebuah lintasan/rute yang harus dilalui dari *node* 1 menuju *node* 166. Untuk menuju *node* 166, dari node awal yaitu node 1 harus mengunjungi *node* 95 terlebih dahulu, lalu dilanjutkan dengan *node* 30, selanjutnya *node* 80 dan sampai ke *node* 166. Rute tersebut memerlukan waktu tempuh selama 66 menit dan memiliki skor sejumlah 767.

- Gen
  Pada kasus ini, gen merupakan *node* yang dikunjungi dalam suatu rute/lintasan. Jumlah gen dalam suatu kromosom adalah tidak tetap. Penetapan jumlah gen dalam kromosom akan ditentukan secara acak. Maksimum jumlah gen dalam suatu kromosom adalah N, dimana N merupakan jumlah *node* dalam network model. Seperti pada Gambar 4.3, dalam kromosom tersebut terdapat 5 gen yaitu 1, 95, 30, 80, dan 166.

4.7. Desain Genetic Algorithm

Pada bagian ini akan dijabarkan mengenai alur kerja Genetic Algorithm untuk membuat solusi dari model OP dalam menyelesaikan permasalahan mencari rute yang optimum. Pada tugas akhir ini digunakan bahasa pemrograman Java. Dalam penerapan algoritma ini digunakan masukan (*input*) yaitu hasil dari proses yang dijelaskan pada bagian 4.3.
Berikut tahapan yang dilakukan dalam menerapkan *Genetic Algorithm* untuk menyelesaikan permasalahan pada studi kasus ini.

4.7.1. Inisialisasi Parameter

Pada tahap ini dilakukan penentuan setiap parameter dan komponen dari *Genetic Algorithm*. Sesuai dengan rekomendasi yang diberikan pada tahap sebelumnya, probabilitas untuk perkawinan silang dan mutasi masing-masing berkisar antara 0.8-0.95 dan 0.05-0.1. [29] Maka dari itu, dalam tugas akhir ini diambil probabilitas perkawinan silang, mutasi, dan ukuran populasi sebagai berikut.

- Probabilitas Perkawinan Silang = 0.9
- Probabilitas mutasi = 0.1
- Ukuran populasi = 50

4.7.2. Pembangkitan Populasi Awal

Pada tahap ini dibangkitkan populasi awal dengan panjang kromosom/individu yang bervariasi. Pada tahap ini dilakukan dengan membangkitkan sejumlah individu secara acak dimana waktu tempuh dari individu yang dihasilkan tidak boleh melebihi batasan t_{max} yang telah ditentukan. Setiap individu harus memiliki gen awal dan akhir yang sama. Secara detail, untuk membangkitkan populasi awal, digunakan *pseudocode* seperti Kode 4.1.

```
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, then pop = pop + 1
} while (pop < jumlahPop)
```

Kode 4.1 Pesudo Code untuk Pembangkitan Populasi Awal

Dari *pseudo code* pada Kode 4.1, untuk pembangkitan populasi awal adalah sebanyak 50 individu. Dimana setiap individu tidak boleh sama (unik) dan memenuhi batasan tMax.

4.7.3. Evaluasi Fitness

Pada tahap ini dilakukan dengan mengkalkulasikan jumlah skor dari tiap *node* yang terdapat dalam suatu rute/lintasan. Seperti yang dijelaskan pada bagian 4.4.2, fungsi tujuan yang ditentukan dalam pemodelan OP adalah memaksimalkan skor yang merepresentasikan peluang mendapatkan angkot. Sesuai dengan fungsi tujuan tersebut, fungsi fitness dari algoritma ini adalah jumlah skor. Semakin tinggi jumlah skor, maka semakin besar peluang mendapatkan angkot sehingga secara tidak langsung memungkinkan untuk memperkecil waktu tunggu/tempuh. Fitness ini digunakan untuk mengukur kelayakan dari individu pada populasi awal dan populasi baru yang dibentuk dari hasil perkawinan silang dan mutasi.

4.7.4. Seleksi Individu

Pada proses ini digunakan seleksi untuk menentukan induk pada proses selanjutnya yaitu perkawinan silang (*cross over*) dan juga proses mutasi. Dalam tahap ini digunakan metode *roulette wheel selection* dimana akan diambil sebanyak 40 individu yang *unique* pada populasi. Seleksi tersebut akan memilih individu dari populasi berdasarkan nilai *fitness*. Semakin tinggi nilai fitness maka semakin besar peluang individu tersebut terpilih. Cara kerja metode ini adalah sebagai berikut [26].

a. Hitung nilai *fitness* dari masing-masing individu (jumlah skor tiap *node* yang dikunjungi).
b. Hitung total *fitness* semua individu.
c. Dihitung probabilitas masing-masing individu (nilai *fitness* individu dibagi total *fitness*).
d. Berdasarkan probabilitas tersebut, dihitung jatah masing-masing individu dari angka 0 sampai 1.

e. Dibangkitkan bilangan random dari 0 sampai 1.
f. Dari bilangan random yang dihasilkan, ditentukan individu mana saja yang terpilih dari proses seleksi.

4.7.5. Perkawinan Silang

Setelah ditentukan individu/induk dari proses seleksi, maka dilakukan perkawinan silang (*crossover*) guna mendapatkan keturunan/solusi baru yang diharapkan lebih baik dari individu sebelumnya. Tidak semua individu yang terpilih akan melakukan perkawinan silang, tergantung dari probabilitas perkawinan silang yang ditentukan sebelumnya. Pada kasus ini dilakukan modifikasi *injection crossover* untuk memindah-silangkan posisi gen pada kromosom seperti penjelasan pada bagian d Proses Perkawinan Silang dengan contoh sebagai berikut. [17]

Induk 1 : 1, 9, 18, 16, 15, 13, 2, 21
Induk 2 : 1, 3, 13, 11, 10, 12, 2, 5, 8, 15, 20, 21

Proto-Child : 1, 9, 18, 16, 15, 2, 5, 8, 15, 20, 21
Keturunan : 1, 9, 18, 16, 15, 2, 5, 21

Dalam proses perkawinan silang menggunakan modifikasi *injection crossover* seperti pada contoh di atas, terdapat rentetan proses sebagai berikut [17].

a. Dipastikan induk 1 memiliki jumlah gen yang lebih sedikit atau sama dengan induk 2.
b. Dibuat sebuah *insertion point* untuk induk 1 yang merupakan bilangan acak antara indeks 1 sampai indeks 2 terakhir.
c. *Insertion point* tersebut akan membentuk *sub list* mengambil gen pada induk 1 dimulai dari indeks 0 (*node* awal) sampai indeks *insertion point* berada.
d. Selanjutnya pada induk 2 juga akan dibentuk *sub list* dengan mengambil gen dimulai dari indeks terakhir

(tidak termasuk *node* akhir) sebanyak gen yang diambil pada induk 1.

e. Kemudian *sub list* dari induk 1 (hasil langkah c) digabungkan dengan *sub list* dari induk 2 (hasil langkah d) untuk menghasilkan *proto child*.
f. Jika jumlah gen yang digabungkan lebih sedikit dibandingkan gen pada induk 1, maka keturunan yang dihasilkan pada proses tersebut akan digantikan dengan induk 1 [20].
g. Gen sub list yang sama dengan induk 1 akan dihapus untuk mendapatkan *proto child*.
h. *Proto child* akan disesuaikan ukurannya dengan induk 1 untuk membentuk keturunan. Setiap perkawinan silang hanya menghasilkan 1 keturunan.
i. Jika *Proto child* memiliki jumlah gen yang lebih sedikit dibandingkan dengan induk 1, maka keturunan akan langsung digantikan dengan induk 1 [20].

4.7.6. Mutasi

Pada tahap ini memungkinkan untuk memunculkan solusi baru dari keturunan yang dihasilkan pada tahapan sebelumnya. Namun pada proses ini diusahakan tidak banyak individu yang mengalami mutasi dengan cara menetapkan probabilitas mutasi yang kecil yaitu 0.1 agar keturunan masih memiliki. [29] Seperti penjelasan pada bagian e Proses Mutasi, digunakan operator pencarian lokal dimana solusi yang *feasible* (sesuai dengan batasan t_{max}) akan semakin membaik sedangkan yang tidak *feasible* akan semakin memburuk.

Dicontohkan dalam kasus ini, keturunan yang dimutasi akan dipilih secara acak, dimana digunakan dua operator pencarian lokal yaitu *add* dan *omit*. Prosedur tersebut akan terus diulang secara berurutan sampai mendapatkan jumlah hasil mutasi berdasarkan probabilitas mutasi. Berikut ini adalah contoh penggunaan pencarian lokal.

- **Operator *Add***

  1, 3, 11, 18, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21

  - Temukan *node* yang tidak dalam keturunan secara acak, misalnya **14**.
  - Masukkan node tersebut ke dalam keturunan pada posisi yang acak.

  1, 3, 11, 18, 16, 15, 13, 2, 5, **14**, 6, 12, 8, 10, 20, 21

  - Evaluasi keturunan.
  - Apabila *node* tersebut menghasilkan nilai *fitness* (skor) yang lebih besar dan waktu tempuh yang lebih kecil atau sama dengan sebelumnya, maka *node* tersebut disimpan. Jika tidak, *node* tersebut akan dihapus.
  - Ulangi sampai bilangan random tidak menemukan keturunan yang lebih baik pada iterasi ke 100.
  - Hasil mutasi pada iterasi terakhir dimasukkan ke dalam populasi keturunan dengan menggantikan keturunan sebelumnya.

- **Operator *Omit***

  1, 3, 11, **18**, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21

  - Temukan *node* yang ada dalam keturunan, misalnya **18**.
  - Hilangkan node tersebut ke dalam keturunan.

  1, 3, 11, 16, 15, 13, 2, 5, 6, 12, 8, 10, 20, 21

  - Evaluasi keturunan.
  - Apabila *node* yang dihilangkan menghasilkan waktu tempuh yang lebih kecil dari sebelumnya, maka *node* akan benar-benar terhapus. Jika tidak, *node* tersebut akan kembali dimasukkan.
  - Ulangi sampai bilangan random tidak menemukan keturunan yang lebih baik pada iterasi ke 100.
  - Hasil mutasi pada iterasi terakhir dimasukkan ke dalam populasi keturunan dengan menggantikan keturunan sebelumnya.

4.7.7. Pemberhentian Algoritma

Individu pada populasi awal serta keturunan dari proses perkawinan silang dan mutasi akan digabung dan dipilih 50 individu terbaik berdasarkan nilai *fitness*-nya untuk menjadi populasi baru, karena jika dipilih secara acak ada kemungkinan individu terbaik pada generasi sebelumnya tidak masuk ke populasi baru. Pembuatan populasi baru akan terus berulang hingga algoritma tersebut memenuhi kondisi yang telah ditentukan sebelumnya, yaitu jumlah generasi. Algoritma akan berhenti ketika telah mencapai nilai maksimal generasi/iterasi yang telah ditentukan sebelumnya, yaitu sebanyak 300 generasi, dimana menunjukkan bahwa pemberhentian algoritma terjadi pada iterasi ke 300.

BAB V
IMPLEMENTASI

Pada bab ini berisi tentang proses implementasi algoritma dalam mencari hasil yang optimum dari studi kasus tugas akhir ini menggunakan bahasa pemrograman *Java* dengan *tools* NetBeans IDE 8.1.

5.1. Pembaruan Matriks Waktu Tempuh dengan Algoritma Dijkstra

Seperti yang terlihat pada Tabel 4.8 pada matriks tersebut banyak terisi dengan nilai nol karena banyak *node* yang tidak terhubung secara langsung. Hasil tersebut masih belum dapat dijadikan masukan untuk *Genetic Algorithm* karena jika dijalankan tidak akan menghasilkan solusi (individu) yang layak sesuai dengan jumlah populasi. Seperti penjelasan pada sub bab 3.1.6.1, untuk mengisi nilai nol pada matriks awal waktu tempuh yang telah dibentuk, perlu diterapkan algoritma Dijkstra. Nilai nol akan diisi dengan waktu tempuh tercepat antar *node* yang dihasilkan oleh algoritma tersebut. Matriks awal waktu tempuh yang telah dibuat sebelumnya dengan format. csv akan menjadi *input* untuk algoritma ini yang dijalankan pada NetBeans IDE 8.1. Algoritma ini dibuat dengan penjelasan Kode 5.1 sebagai berikut.

```
19    static int jumlahNode = 191;
20
21  int minDistance(int dist[], Boolean sptSet[]) {
22        // Inisiasi nilai minimum
23        int min = Integer.MAX_VALUE;
24        int min_index = -1;
25
26        for (int i = 0; i < jumlahNode; i++) {
27            if (sptSet[i] == false && dist[i] <= min) {
28                min = dist[i];
29                min_index = i;
30            }
31        }
32        return min_index;
33    }
```

Kode 5.1 Method Pencarian Waktu Tempuh Minimum

Pada baris 19 dalam Kode 5.1 dideklarasikan terlebih dahulu variabel jumlah *node* dalam studi kasus ini yaitu sebanyak 191 *node*. Selanjutnya dari baris 21 sampai 33 merupakan sebuah fungsi untuk menemukan *node* dengan nilai waktu tempuh minimum dari sekumpulan *node* yang belum memiliki nilai waktu tempuh terpendek.

```
36  void printSolution(int dist[], int n) {
37      for (int i = 0; i < jumlahNode; i++) {
38          System.out.print(dist[i] + ",");
39      }
40  }
```

Kode 5.2 Method Cetak Hasil Waktu Tempuh Tercepat

Selanjutnya Kode 5.2 merupakan fungsi untuk mencetak solusi berupa waktu tempuh tercepat yang dihasilkan algoritma Dijkstra. Digunakan tanda pemisah koma (,) untuk memisahkan hasil waktu tempuh antar *node*.

```
45  void dijkstra(int graph[][], int src) {
46      int dist[] = new int[jumlahNode];
47
48      Boolean sptSet[] = new Boolean[jumlahNode];
49
50      for (int i = 0; i < jumlahNode; i++) {
51          dist[i] = Integer.MAX_VALUE;
52          sptSet[i] = false;
53      }
54
55      dist[src] = 0;
56
57      for (int count = 0; count < jumlahNode - 1; count++) {
58
59          int u = minDistance(dist, sptSet);
60          sptSet[u] = true;
61
62          for (int v = 0; v < jumlahNode; v++) {
63              if (!sptSet[v] && graph[u][v] != 0
64                      && dist[u] != Integer.MAX_VALUE
65                      && dist[u] + graph[u][v] < dist[v]) {
66                  dist[v] = dist[u] + graph[u][v];
67              }
68          }
69      }
70      printSolution(dist, jumlahNode);
71  }
```

Kode 5.3 Method Penerapan Dijkstra

Kode 5.3 merupakan sebuah fungsi yang menerapkan algoritma *Dijkstra* (*single source shortest path*) untuk sebuah *graph* yang direpresentasikan dalam bentuk matriks. Fungsi ini akan mencari waktu tempuh tercepat dari *node* sumber ke seluruh *node* lainnya. Pada baris 46, program akan membuat array dist yang akan menyimpan nilai waktu tempuh dari setiap *node*. Selanjutnya pada baris 48 dibuat array sptSet bertipe Boolean yang akan bernilai *true* jika waktu tempuh tercepat dari *node* sumber ke *node* tujuan telah ditemukan (*finalized*).

Baris 50 sampai 53 merupakan sebuah loop yang digunakan untuk menginisialisasi semua waktu tempuh (pada array dist) bernilai *infinite* dan Boolean sptSet bernilai *false*. Khusus waktu tempuh *node* sumber ke *node* sumber itu sendiri akan dibuat bernilai nol seperti yang ditunjukkan baris 55. Baris 57 sampai 71 merupakan fungsi utama untuk mencari waktu tempuh tercepat dari *node* sumber ke seluruh *node*. Update dist hanya jika tidak ada di sptSet, node sumber ke node tujuan terhubung, dan nilai waktu tempuh yang baru lebih kecil dibandingkan nilai sebelumnya. Baris 70 akan mencetak hasil dari algoritma tersebut.

```
74   public static void main(String[] args) throws
75           FileNotFoundException, IOException {
76       //KONVERSI ARRAY 2D UNTUK NILAI ARC DARI FILE .CSV
77       String thisLinel;
78       BufferedReader bufRdr2D = new BufferedReader(
79               new FileReader("data/MatriksAwal.csv"));
80
81       ArrayList<String[]> lines = new ArrayList<String[]>();
82       while ((thisLinel = bufRdr2D.readLine()) != null) {
83           lines.add(thisLinel.split(","));
84       }
85
86       String[][] array = new String[lines.size()][0];
87       lines.toArray(array);
88
89       int[][] graph = new int[jumlahNode][jumlahNode];
90       for (int i = 0; i < array.length; i++) {
91           for (int j = 0; j < array.length; j++) {
92               graph[i][j] = Integer.parseInt(array[i][j]);
93           }
94       }
```

Kode 5.4 Fungsi Import Matriks Waktu Tempuh

Selanjutnya pada kelas *main*, seperti yang ditunjukkan pada Kode 5.8, pada baris 78 dan 79, digunakan fungsi BufferedReader dan FileReader untuk membaca input data yaitu file dengan nama matriks.csv yang disimpan dalam folder data sebagai tempat menyimpan matriks waktu tempuh. Selanjutnya, seperti yang ditunjukkan baris 81 sampai 84, data tersebut dibaca per baris sebagai String dan disimpan ke dalam *ArrayList* lines. Karena memiliki tipe *String*, selanjutnya dari *ArrayList* data dimasukkan terlebih dahulu ke dalam sebuah array bertipe *String*. Selanjutnya nilai waktu tempuh tersebut diubah ke dalam bentuk *Integer* dengan melakukan *parsing* dan menyimpannya ke dalam array 2 dimensi *graph* seperti yang ditunjukkan pada baris 89 sampai 94.

```
96              AlgoritmaDijkstra t = new AlgoritmaDijkstra();
97
98              for (int i = 0; i < jumlahNode; i++) {
99                  t.dijkstra(graph, i);
100                 System.out.println();
101             }
102     }
103 }
```

Kode 5.5 Pemanggilan Method Dijkstra

Kode 5.5 digunakan untuk memanggil method *Dijkstra* dengan matriks 2 dimensi *graph* yang telah dibuat sebelumnya (pada Kode 5.3) dan *node* sumber. Dilakukan *looping* yang ditunjukkan pada baris 98 sampai 101 untuk memanggil method *Dijkstra* secara berulang dengan *node* sumber yang berbeda. Sehingga hasil *looping* tersebut akan dicetak di setiap baris baru dan disimpan ke dalam Arc191x191.csv untuk dijadikan *input* pada penerapan *Genetic Algorithm* (GA).

5.2. Implementasi Genetic Algorithm

Pada sub bab ini menjelaskan implementasi Genetic Algorithm (GA) yang telah dirancang pada bab sebelumnya. Pada setiap pengimplementasian tahapan GA akan disertakan dengan potongan kode yang dikemas dalam satu kelas *Java*

menggunakan aplikasi NetBeans IDE 8.1 dengan file berformat .java beserta penjelasan masing-masing potongan kode tersebut.

5.2.1. Inisialisasi Parameter dan Data

Tahap ini menjelaskan mengenai apa saja yang dijadikan input ke dalam *Java* untuk dilakukan pengimplementasian algoritma sehingga dapat mengeluarkan solusi rute yang optimum. Input tersebut terdiri dari beberapa parameter dan data-data yang akan digunakan. Penentuan parameter yang digunakan untuk *Genetic Algorithm* dalam pencarian solusi dari model *Orienteering Problem* adalah deklarasi *node* awal, *node* akhir, jumlah *node*, batasan waktu (tMax), jumlah populasi, jumlah individu yang diseleksi, jumlah generasi, probabilitas perkawinan silang (*cross over*), dan probabilitas mutasi. Penentuan parameter dapat dilihat pada Kode 5.6.

```
28    static int nodeAwal = 1;
29    static int nodeAkhir = 166;
30    static int jumlahNode = 191;
31    static int tMax = 70; //Menetapkan
32    static int jumlahPop = 50; //Meneta
33    static int jumlahSelect = 40; //Mer
34    static int jumlahGenerasi = 300; //
35    static double probCrossOver = 0.9;
36    static double probMutation = 0.1;
```

Kode 5.6 Inisialisasi Paramater GA

Kode 5.6 menunjukkan bahwa model dari studi kasus ini memiliki jumlah *node* (jumlahNode) sebanyak 191. Seperti penjelasan pada bab sebelumnya, *node* awal (nodeAwal) dan akhir (nodeAkhir) ditentukan terlebih dahulu dimana dalam studi kasus ini dicontohkan pencarian rute optimum dari *node* 1 menuju *node* 166. Dalam pencarian solusi, ditentukan jumlah populasi (jumlahPop) akan menampung sebanyak 50 individu dengan pencarian solusi akan dilakukan sampai 300 iterasi

(jumlahGenerasi). Populasi tersebut akan diseleksi menjadi 40 individu. Probabilitas perkawinan silang (probCrossOver) sebesar 0.9 digunakan untuk menentukan seberapa banyak individu dari hasil seleksi yang akan dikawin silangkan. Begitu juga probabilitas mutasi (probMutasi) akan digunakan sebesar 0.1 digunakan untuk menentukan seberapa banyak dari hasil perkawinan silang yang akan dimutasi. Karena menggunakan pemodelan OP, maka dideklarasikan juga batasan waktu (tMax) sebesar 70 menit.

Selanjutnya adalah mendeklarasikan data-data yang digunakan sebagai masukan dalam pengimplementasian GA, yaitu data waktu tempuh dan data skor. Masing-masing data tersebut dimasukkan ke dalam array yang digunakan sebagai tempat penyimpanan seperti Kode 5.7.

```
38    static int[][] nilaiArc = new int[jumlahNode][jumlahNode];
39    static int[] nilaiSkor = new int[jumlahNode];
```

Kode 5.7 Array Penyimpanan Data Waktu Tempuh dan Skor

```
81  //A. KONVERSI ARRAY 2D UNTUK NILAI ARC DARI FILE .CSV
82  String thisLine2D;
83  BufferedReader bufRdr2D
84          = new BufferedReader(new FileReader("data/Arc191x191.csv"));
85
86  ArrayList<String[]> lines = new ArrayList<String[]>();
87  while ((thisLine2D = bufRdr2D.readLine()) != null) {
88      lines.add(thisLine2D.split(","));
89  }
90
91  //A.1. Mengkonversi list ke dalam String array
92  String[][] array = new String[lines.size()][0];
93  lines.toArray(array);
94
95  //A.2. Mengkonversi String array ke Integer array
96  for (int i = 0; i < array.length; i++) {
97      for (int j = 0; j < array.length; j++) {
98          nilaiArc[i][j] = Integer.parseInt(array[i][j]);
99      }
100 }
```

Kode 5.8 Konversi Data Waktu Tempuh ke dalam Array

Array 2 dimensi dengan nama nilaiArc digunakan untuk menyimpan matriks waktu tempuh antar *node*, sedangkan *array* 1 dimensi dengan nama nilaiSkor digunakan untuk menyimpan matriks skor untuk setiap *node*. Kedua data tersebut disimpan dalam format .csv dan diperlukan fungsi untuk dapat membaca dan memasukkan isi matriks tersebut ke dalam *array* seperti Kode 5.8.

Berdasarkan Kode 5.8, pada baris 83 dan 84, digunakan fungsi BufferedReader dan FileReader untuk membaca input data yaitu file dengan nama Arc191x191.csv yang disimpan dalam *folder* data sebagai tempat menyimpan matriks waktu tempuh. Selanjutnya, seperti yang ditunjukkan baris 86 sampai 89, data tersebut dibaca per baris sebagai String dan disimpan ke dalam *ArrayList* lines. Karena memiliki tipe *String*, selanjutnya dari *ArrayList* data dimasukkan terlebih dahulu ke dalam sebuah array bertipe *String*. Selanjutnya nilai waktu tempuh tersebut diubah ke dalam bentuk *Integer* dengan melakukan *parsing* dan menyimpannya ke dalam array 2 dimensi nilaiArc seperti yang ditunjukkan pada baris 96 sampai 100.

```
102   //B. KONVERSI ARRAY 1D UNTUK SCORE TIAP NODE DARI FILE .CSV
103   BufferedReader bufRdr1D
104           = new BufferedReader(new FileReader("data/Score191.csv"));
105
106   String thisLine1D;
107   int o = 0;
108   //B.1. Membaca setiap baris yang ada pada file .csv dan memasukkan
109   //ke dalam array integer
110   while ((thisLine1D = bufRdr1D.readLine()) != null) {
111       nilaiSkor[o] = Integer.parseInt(thisLine1D);
112       o++;
113   }
```

Kode 5.9 Konversi Data Skor ke dalam Array

Selanjutnya dengan fungsi yang mirip, berdasarkan Kode 5.9, pada baris 103 dan 104, digunakan fungsi BufferedReader dan FileReader untuk membaca input data yaitu file dengan nama Score191.csv yang disimpan dalam *folder* data sebagai tempat menyimpan matriks skor. Karena hanya berbentuk matriks 1 dimensi dan hanya memiliki satu nilai di setiap barisnya, maka

setelah dibaca per baris, data skor dapat langsung diubah ke dalam bentuk *Integer* dengan melakukan *parsing* dan menyimpannya ke dalam *array* 1 dimensi seperti yang ditunjukkan pada baris 110 sampai 113.

5.2.2. Pembangkitan Populasi Awal

Dalam tahapan ini, akan dihasilkan sejumlah individu yang akan disimpan ke dalam ArrayList arrPopulation, beserta waktu tempuh masing-masing individu ke dalam ArrayList dengan nama arrPopulation_WaktuTempuh dan nilai fitness dari masing-masing individu ke dalam ArrayList dengan nama arrPopulation_Fitness seperti Kode 5.10.

```
41  static ArrayList<ArrayList<Integer>> arrPopulation
42          = new ArrayList<ArrayList<Integer>>();
43  static ArrayList<Integer> arrPopulation_WaktuTempuh
44          = new ArrayList<Integer>();
45  static ArrayList<Integer> arrPopulation_Fitness
46          = new ArrayList<Integer>();
```

Kode 5.10 Deklarasi ArrayList sebagai Tempat Penyimpanan Individu dalam Populasi beserta Waktu Tempuh dan Nilai Fitness

Setelah data waktu tempuh dan skor terbaca oleh *Java* serta ArrayList tempat penyimpanan populasi dibuat, maka dibangkitkan sejumlah individu yang merepresentasikan solusi layak berdasarkan variabel tMax (batasan waktu). Selain itu dalam mencari solusi (individu) dilakukan iterasi sampai mencapai jumlah populasi yang telah ditentukan sebelumnya. Untuk itu perlu dideklarasikan variabel pop untuk mencatat jumlah individu yang telah dibangkitkan seperti pada baris 116 dalam Kode 5.11. Pada baris 117 menunjukkan bahwa dalam iterasi dilakukan dengan fungsi loop *do-while* dengan batasan bahwa individu yang dihasilkan pada pembangkitan populasi awal tidak boleh lebih dari jumlah populasi seperti pada Kode 5.12 baris 172.

Berdasarkan Kode 5.11 dan Kode 5.12, dimana kode tersebut merupakan implementasi dari Kode 4.1, dibuat sebuah

ArrayList dengan nama listRl untuk merepresentasikan individu/kromosom dimana gen awal dan akhirnya tetap. Selanjutnya isi dari individu tersebut (gen di antara gen awal dan akhir) disimpan ke dalam *ArrayList* dengan nama listRl_random dan isi dari gen tersebut diacak menggunakan fungsi *shuffle* seperti yang ditunjukkan pada baris 122 sampai 136.

```
115   //1. PEMBANGKITAN POPULASI AWAL DENGAN RANDOM
116   int loop = 0;
117   int maxLoop = 500000;
118   int pop = 0; //Menetapkan inisiasi jumlah individu dala
119   do {
120       //1.1. Menghasilkan nilai random, Rn , diantara inc
121       //digunakan untuk menentukan panjang solusi yang be
122       int Rn = new Random().nextInt(jumlahNode - 2) + 1;
123
124       //1.2. Menghasilkan list, Rl , diantara [1,N] secar
125       ArrayList<Integer> listRl = new ArrayList();
126       ArrayList<Integer> listRl_rand //Array yang digunal
127               = new ArrayList();     //node diantara node
128
129       //1.3. Menggenerate isi listRl_rand diantara node a
130       for (int i = 1; i < jumlahNode + 1; i++) {
131           if (i == nodeAwal || i == nodeAkhir) {
132               continue; //Memastikan tidak ada node awal
133           }             //yang muncul di tengah kromosom
134           listRl_rand.add(i);
135       }
136       Collections.shuffle(listRl_rand); //Mengacak node j
137
138       //1.4. Mengisi listRl
139       listRl.add(nodeAwal); //Merepresentasikan node awal
140       for (int i = 0; i < listRl_rand.size(); i++) {
141           listRl.add(listRl_rand.get(i)); //Representasi
142       }                                   //ke N-1 yang t
143       listRl.add(nodeAkhir); //Merepresentasikan node akh
```

Kode 5.11 Penggalan Iterasi Pembangkitan Populasi Awal

Solusi yang disimpan dalam *ArrayList* listRl merupakan solusi dengan panjang maksimal (pada kasus ini sejumlah 191 *node*). Oleh karena itu, dibuatlah ArrayList baru yaitu subListRs yang merupakan gabungan dari indeks 0 dari listRl, perpotongan dari listRl berdasarkan *insertion point* Rn yang merupakan bilangan

acak diantara 1 sampai indeks kedua terakhir, dan indeks terakhir listRl seperti Kode 5.12 baris 147 sampai 152. Hasil gabungan tersebut akan memiliki panjang yang bervariasi dan disimpan ke dalam subListRl.

```
145     //1.5. Menghasilkan sebuah sub list, Rs dengan ukuran
146     //yang bervariasi sesuai dengan Rn.
147     ArrayList<Integer> subListRs = new ArrayList();
148     subListRs.add(listRl.get(0));
149     for (int i = 1; i < (Rn + 1); i++) {
150         subListRs.add(listRl.get(i));
151     }
152     subListRs.add(listRl.get(listRl.size() - 1));
153
154     //1.6. Hanya memasukkan subList Rs (individu) yang uni
155     //dan tidak melebihi batasan waktu tMax ke dalam popul
156     for (int i = 0; i < subListRs.size(); i++) {
157         if (hitungWaktuTempuh(subListRs) <= tMax
158                 && !arrPopulation.contains(subListRs)) {
159             arrPopulation.add(subListRs);
160             pop++;
161         }
162     }
163
164     loop++;
165
166     if (loop == maxLoop) {
167         System.out.println("Tidak ditemukan solusi yang "
168                 + "layak untuk tMax = " + tMax);
169         System.exit(0);
170     }
171
172 } while (pop < jumlahPop);
```

Kode 5.12 Penggalan Iterasi Pembangkitan Populasi Awal

Dalam proses pembangkitan populasi awal, individu yang dihasilkan harus memenuhi batasan-batasan dari model OP, yaitu:

a. Batasan 1 yaitu individu/solusi yang dibangkitkan harus memiliki *node* awal dan *node* akhir yang tetap. Hal ini ditunjukkan pada Kode 5.11 baris 139 dan 143 dimana menggunakan fungsi *add* untuk

menambahkan *node* awal pada indeks pertama dan *node* akhir pada indeks terakhir.

b. Batasan 2 yaitu *node* saling terhubung dan hanya boleh dikunjungi maksimum satu kali. Hal ini menandakan dalam pembangkitan solusi tidak boleh ada gen yang sama dibangkitan dua kali. Ini ditunjukkan dari bagian lisRl_rand yang membangkitkan dan mengacak gen yang unik seperti pada Kode 5.11 baris 130 sampai 136. Baris 131 sampai 133 juga memastikan agar isi dari solusi yang dibangkitkan tidak akan memunculkan gen awal dan akhir dua kali atau lebih.

c. Batasan 3 yaitu solusi tidak boleh melebihi tMax. Ini ditunjukkan pada Kode 5.12 baris 156 sampai 162 dimana arrPopulation tidak akan menyimpan individu yang memiliki waktu tempuh lebih dari tMax dan tidak akan menyimpan individu yang sama dua kali.

d. Batasan 4 yaitu tidak boleh ada subtours, lintasan dimulai dan berakhir pada *node* yang sama. Batasan ini sudah tergambarkan dalam batasan 1 dan batasan 2. Sehingga secara tidak langsung, jika batasan 1 dan 2 terpenuhi, maka batasan 4 juga terpenuhi.

Solusi yang memenuhi 4 batasan di atas selanjutnya akan disimpan ke dalam arrPopulation. Untuk menghitung waktu tempuh setiap solusi yang dibangkitkan seperti pada penjesalan di atas, diperlukan sebuah *method* yang berfungsi menghitung total waktu tempuh setiap solusi yang ada seperti Kode 5.13.

```
567   //Fungsi untuk menghitung waktu tempuh setiap individu.
568   public static int
569           hitungWaktuTempuh(ArrayList<Integer> varList) {
570       int sum = 0; //Inisiasi nilai awal sum adalah 0
571       for (int i = 0; i < varList.size() - 1; i++) {
572           sum = sum
573                   + nilaiArc[varList.get(i) - 1][varList.get(i + 1) - 1];
574       } //Loop di atas utk menghitung waktu tempuh tiap node yg berdekatan
575       return sum;
576   }
```

Kode 5.13 Method untuk Menghitung Waktu Tempuh

Kode 5.13 menunjukkan sebuah *method* untuk menghitung waktu tempuh suatu solusi yang telah disimpan ke dalam bentuk *ArrayList*. Nilai dari total waktu tempuh tersebut disimpan dalam variabel sum. Sum akan terus menjumlahkan setiap waktu tempuh antar *node* (gen) yang terdapat dalam suatu solusi berdasarkan array 2 dimensi nilaiArc (telah dideklarasikan pada tahapan sebelumnya) seperti yang ditunjukkan pada baris 570 sampai 574. Ketika *method* ini dipanggil, *method* akan mengeluarkan total waktu tempuh dari *ArrayList* yang diinginkan.

Jika 4 batasan model OP tersebut tidak dapat dipenuhi maka pembangkitan populasi awal tidak akan menghasilkan jumlah individu yang sesuai dengan parameter jumlah populasi yang telah ditentukan sebelumnya. Maka dari itu ditentukan juga batas iterasi yang dilakukan saat proses ini dengan mendeklarasikan variable loop dan maxLoop seperti pada Kode 5.11 baris 116 dan 117. Ketika batas iterasi sudah tercapai (loop = maxLoop) namun pembangkitan populasi awal belum mencapai parameter jumlah populasi, maka proses Genetic Algorithm akan dibatalkan seperti pada Kode 5.12 baris 166 sampai 170.

```
174   //1.7. Mengisi array untuk waktu tempuh dan fitness se
175   //individu dari populasi awal.
176   for (int i = 0; i < arrPopulation.size(); i++) {
177       arrPopulation_WaktuTempuh.add(
178               hitungWaktuTempuh(arrPopulation.get(i)));
179       arrPopulation_Fitness.add(
180               hitungFitness(arrPopulation.get(i)));
181   }
```

Kode 5.14 Iterasi Menyimpan Waktu Tempuh dan Fitness Populasi Awal ke dalam ArrayList

Selanjutnya ketika sudah dibangkitkan sejumlah individu sesuai dengan parameter jumlah populasi yang telah ditentukan, maka nilai waktu tempuh dan nilai *fitness* dari setiap individu tersebut akan dimasukkan ke dalam arrPopulation_WaktuTempuh dan

arrPopulation_Fitness seperti pada Kode 5.14 baris 176 sampai 181.

```
183   //1.8. Mencari individu terbaik dari proses pembangkitan
184   //populasi awal.
185   int indSolusiAwal = 0;
186   int optFitnessAwal = arrPopulation_Fitness.get(0);
187   for (int i = 1; i < arrPopulation.size(); i++) {
188       if (arrPopulation_Fitness.get(i) >= optFitnessAwal) {
189           indSolusiAwal = i;
190           optFitnessAwal = arrPopulation_Fitness.get(i);
191       }
192   }
193
194   //1.9. Mencetak hasil individu terbaik.
195   pw.println("Generasi 0 | Rute: "
196           + arrPopulation.get(indSolusiAwal) + " | Waktu Tempuh: "
197           + arrPopulation_WaktuTempuh.get(indSolusiAwal) + " | Fitness: "
198           + arrPopulation_Fitness.get(indSolusiAwal));
```

Kode 5.15 Pemilihan Individu Terbaik dari Proses Pembangkitan Populasi Awal

Selain itu, pada proses pembangkitan populasi awal dicari individu/solusi awal terbaik dengan tujuan untuk mengetahui perubahan/perbaikan solusi yang terjadi selama awal sampai akhir dari penerapan *Genetic Algorithm* seperti yang ditunjukkan pada baris 185 sampai 192. Baris 195 sampai 198 digunakan untuk mencetak hasil tersebut.

5.2.3. Evaluasi Fitness

Pada tahap ini dilakukan perhitungan nilai *fitness* terbaik untuk mengevaluasi suatu populasi. Dalam perhitungan tersebut dibuat *method* baru dimana setiap individu dalam suatu populasi dihitung nilai *fitness*nya sehingga individu tersebut dapat saling dibandingkan satu sama lain dengan nilai *fitness* tersebut. Seperti penjelasan pada bagian 4.7.3, nilai *fitness* merupakan jumlah skor dimana kode dari *method* tersebut dijabarkan seperti Kode 5.16.

```
578   //Fungsi untuk menghitung fitness setiap individu.
579   public static int hitungFitness(ArrayList<Integer> varList) {
580       int sum = 0; //Inisiasi nilai awal sum adalah 0
581       //int fitness;
582       for (int i = 0; i < varList.size(); i++) {
583           sum = sum + nilaiSkor[varList.get(i) - 1];
584       } //Loop di atas utk menghitung skor tiap node
585       //fitness = (sum^3)/hitungWaktuTempuh(varList);
586       //return fitness;
587       return sum;
588   }
```

Kode 5.16 Method untuk Evaluasi Fitness

Berdasarkan Kode 5.16, *method* tersebut digunakan untuk menghitung nilai *fitness* pada setiap individu yang sebelumnya telah disimpan ke dalam bentuk *ArrayList*. Nilai dari total skor (*fitness*) tersebut disimpan dalam variabel sum. Sum akan terus menjumlahkan setiap skor *node* yang menjadi gen dalam suatu individu/solusi berdasarkan array 1 dimensi nilaiSkor (telah dideklarasikan pada tahapan sebelumnya) seperti yang ditunjukkan pada baris 580 sampai 586. Ketika *method* ini dipanggil, *method* akan mengeluarkan nilai *fitness* dari *ArrayList* yang diinginkan.

5.2.4. Seleksi

Tahapan selanjutnya adalah seleksi untuk memilih individu dari populasi yang akan dijadikan sebagai induk untuk proses perkawinan silang. Sebelumnya ditetapkan terlebih dahulu, jumlah individu yang akan lolos dalam seleksi. Sehingga jumlah induk akan selalu tetap untuk setiap proses perkawinan silang. Seleksi dilakukan dengan menggunakan *roulette wheel selection*. Seperti pada penjelasan 4.7.4, metode ini dipilih karena pemilihan individu tidak sepenuhnya dilakukan secara *random*, melainkan berdasarkan probabilitas dari masing-masing individu berdasarkan nilai *fitness*nya. Semakin tinggi nilai *fitness* maka semakin besar peluang individu tesebut terpilih.

Dalam tahapan ini, akan dipilih sejumlah individu dari populasi awal yang akan disimpan ke dalam ArrayList arrSelection,

beserta waktu tempuh masing-masing individu ke dalam ArrayList dengan nama arrSelection_WaktuTempuh dan nilai fitness dari masing-masing individu ke dalam ArrayList dengan nama arrSelection_Fitness seperti Kode 5.17.

```
48    static ArrayList<ArrayList<Integer>> arrSelection
49            = new ArrayList<ArrayList<Integer>>();
50    static ArrayList<Integer> arrSelection_WaktuTempuh
51            = new ArrayList<Integer>();
52    static ArrayList<Integer> arrSelection_Fitness
53            = new ArrayList<Integer>();
```

Kode 5.17 Deklarasi ArrayList sebagai Tempat Penyimpanan Hasil Seleksi beserta Waktu Tempuh dan Nilai Fitness

Nilai *fitness* dikatakan tinggi apabila nilai tersebut berada di atas rata-rata nilai *fitness* semua individu dalam satu populasi. Untuk itu, perlu dibuat *method* terlebih dahulu untuk menghitung total *fitness* keseluruhan individu dalam satu populasi seperti Kode 5.18.

```
590   //Fungsi untuk menghitung total fitness pada populasi.
591   public static int totalFitness() {
592       int sum = 0; //Inisiasi nilai awal sum adalah 0
593       for (int i = 0; i < arrPopulation_Fitness.size(); i++) {
594           sum = sum + arrPopulation_Fitness.get(i);
595       } //Loop di atas utk menghitung fitness tiap individu pd
596       return sum;
597   }
```

Kode 5.18 Method Menghitung Total Nilai Fitness dalam Satu Populasi

Kode 5.18 akan menghitung total keseluruhan dari nilai *fitness* setiap individu dalam satu populasi. Nilai dari total *fitness* tersebut disimpan dalam variabel sum. Sum akan terus menjumlahkan setiap nilai *fitness* dari seluruh individu dalam populasi seperti yang ditunjukkan pada baris 592 sampai 595. Setelah itu, hasil dari *method* di atas berupa total *fitness* akan digunakan oleh *method* lainnya yang menjalankan *roulette wheel selection* seperti Kode 5.19.

```
357  public static int rouletteWheelSelection() {
358      double r = new Random().nextDouble();
359      int totFitness = totalFitness();
360      int k = 0;
361      double sum = (double) arrPopulation_Fitness.get(k) / totFitness;
362      while (sum < r) {
363          k++;
364          sum = sum + (double) arrPopulation_Fitness.get(k) / totFitness;
365      }
366      return k;
367  }
```

Kode 5.19 Method dari Roulette Wheel Selection

Berdasarkan kode di atas, dibuat terlebih dahulu bilangan random antara 0 sampai 1 (*double*). Dideklarasikan variabel yang memanggil *method* totalFitness, variabel k untuk menyimpan indeks dari solusi yang mungkin terpilih, dan variabel sum yang digunakan untuk menyimpan nilai probabilitas dari individu sesuai indeks yang ada. Probabilitas didapatkan dengan membagi nilai *fitness* individu tersebut dengan total *fitness* dari populasi. Apabila probabilitas individu lebih besar dibandingkan bilangan random maka individu dengan indeks k akan terpilih, namun jika tidak, maka pencarian akan berlanjut ke indeks berikutnya.

```
208  //2. MELAKUKAN SELEKSI
209  //2.1. Mengambil individu dari populasi awal dengan jumla
210  //yang telah ditentukan sebelumnya
211  do {
212      int id = rouletteWheelSelection();
213      if (!arrSelection.contains(arrPopulation.get(id))) {
214          arrSelection.add(arrPopulation.get(id));
215      }
216  } while (arrSelection.size() < jumlahSelect);
217
218  //2.2. Mengisi array untuk waktu tempuh dan fitness setia
219  //individu dari hasil seleksi.
220  for (int i = 0; i < arrSelection.size(); i++) {
221      arrSelection_WaktuTempuh.add(
222              hitungWaktuTempuh(arrSelection.get(i)));
223      arrSelection_Fitness.add(
224              hitungFitness(arrSelection.get(i)));
225  }
```

Kode 5.20 Seleksi Menggunakan Roulette Wheel Selection

Setelah *method* dari *roulette wheel selection* dibentuk, seleksi dapat dilakukan seperti Kode 5.20. Kode tersebut berfungsi untuk melakukan seleksi dari *ArrayList* arrPopulation yang merupakan tempat untuk menyimpan individu dalam satu populasi. Individu yang terpilih pada arrPopulation akan disimpan juga pada *ArrayList* arrSelection dengan ketentuan bahwa arrSelection akan menyimpan individu yang *unique* (tidak sama) sebanyak jumlah populasi yang telah ditentukan seperti yang ditunjukkan pada baris 211 sampai 216.

Selanjutnya adalah memasukkan nilai waktu tempuh dan *fitness* dari setiap individu yang terpilih (yang disimpan ke dalam arrSelection) seperti yang ditunjukkan pada baris 220 sampai 225. Detailnya, baris 221 dan 222 merupakan kode untuk menyimpan total waktu tempuh pada individu terpilih (arrSelection) serta baris 223 dan 224 merupakan kode untuk menyimpan nilai *fitness* pada individu terpilih (arrSelection).

5.2.5. Perkawinan Silang

Dalam tahapan ini, akan dihasilkan sejumlah keturunan yang akan disimpan ke dalam ArrayList arrOffSpring, beserta waktu tempuh masing-masing keturunan ke dalam ArrayList dengan nama arrOffSpring_WaktuTempuh dan nilai *fitness* dari masing-masing keturunan ke dalam ArrayList dengan nama arrOffSpring_Fitness seperti Kode 5.21.

```
55  static ArrayList<ArrayList<Integer>> arrOffSpring
56          = new ArrayList<ArrayList<Integer>>();
57  static ArrayList<Integer> arrOffSpring_WaktuTempuh
58          = new ArrayList<Integer>();
59  static ArrayList<Integer> arrOffSpring_Fitness
60          = new ArrayList<Integer>();
```

Kode 5.21 Deklarasi ArrayList sebagai Tempat Penyimpanan Keturunan beserta Waktu Tempuh dan Nilai Fitness

Individu hasil seleksi kemudian dijadikan induk dalam tahap ini. Setiap induk secara berpasangan akan melakukan perkawinan silang dengan ketentuan sebagai berikut [17].

a. Indeks i merupakan individu dengan indeks nomor genap (dimulai dari indeks 0) dan indeks j merupakan individu dengan indeks nomor ganjil (dimulai dari indeks 1). Hal ini ditunjukkan dengan fungsi *loop for* pada Kode 5.22 baris 235 dan 236.
b. Setiap induk hanya melakukan perkawinan silang sebanyak satu kali. Hal ini ditunjukkan pada Kode 5.22 baris 242 dimana memastikan perkawinan silang yang dilakukan antara indeks i dan j dengan ketentuan indeks j merupakan indeks setelah i. Sehingga dicontohkan perkawinan silang dilakukan antara individu indeks 0 dengan 1, indeks 2 dengan 3, dan seterusnya.
c. Ukuran induk 1 tidak boleh melebihi induk 2. Hal ini dikarenakan penggunaan modifikasi *injection crossover* hanya menghasilkan 1 keturunan yang ukurannya disesuaikan dengan induk 1. Hal ini ditunjukkan pada Kode 5.23 fungsi *if* baris 251 dan 252 dimana jika individu indeks i memiliki ukuran lebih kecil atau sama dengan indeks j, maka individu indeks i akan diposisikan sebagai induk 1. Begitu juga sebaliknya (pada fungsi *if else* setelahnya), jika indeks i memiliki ukuran yang lebih besar dibandingkan indeks j, maka indeks j akan diposisikan sebagai induk 1 dan indeks i akan diposisikan sebagai induk 2.
d. Jika keturunan yang dihasilkan memiliki jumlah gen yang lebih sedikit dibandingkan induk 1, maka hasil keturunan tersebut akan digantikan dengan induk 1 seperti pada [20].
e. Perkawinan silang hanya dilakukan jika memenuhi probabilitas crossover (< 0.9). Hal ini ditunjukkan pada Kode 5.22 baris 244 sampai 246 dimana jika bilangan random yang dihasilkan lebih besar dari

probabilitas, maka perkawinan silang untuk indeks tersebut akan dibatalkan dan dilanjutkan pada indeks selanjutnya.

```
235  for (int i = 0; i < arrSelection.size(); i += 2) {
236
237      double rndProbCO = new Random().nextDouble();
238      List<Integer> offSpring = new ArrayList<Integer>();
239
240      //3.2. Memastikan perkawinan silang dilakukan satu
241      //pada setiap individu hasil seleksi.
242      int j = i + 1;
243
244      if (rndProbCO > probCrossOver) {
245          continue; //Crossover hanya dilakukan jika meme
246      }             //probabilitas crossover
```

Kode 5.22 Looping dan Deklarasi Pengecualian dalam Menjalankan CrossOver

Setelah ketentuan-ketentuan yang telah dijabarkan pada Kode 5.22, dimulai proses perkawinan silang dengan mengambil sebagian gen pada induk 1 dan induk 2 seperti Kode 5.23.

```
251  if (arrSelection.get(i).size()
252          <= arrSelection.get(j).size()) {
253
254      //Membuat insertion point untuk Induk 1.
255      int insertionPoint = new Random().nextInt(
256              arrSelection.get(i).size() - 1) + 1;
257      //Mengambil gen sejumlah insertion point
258      //pada Induk 1 dimulai dari indeks 0.
259      List<Integer> head = arrSelection.get(i).subList(0,
260              insertionPoint);
261      //Merupakan list yang nantinya akan digunakan jika
262      //proses perkawinan silang tidak berhasil.
263      List<Integer> cdg = arrSelection.get(i).subList(
264              insertionPoint,
265              arrSelection.get(i).size() - 1);
266      //Mengambil gen sejumlah insertion point pada
267      //Induk 2 dimulai dari indeks sebelum node akhir.
268      List<Integer> tail = arrSelection.get(j).subList(
269              arrSelection.get(j).size() - insertionPoint
270              - 1, arrSelection.get(j).size() - 1);
```

Kode 5.23 Pengambilan Gen pada Induk 1 dan Induk 2

Seperti yang terlihat pada Kode 5.23, dideklarasikan terlebih dahulu sebuah *insertion point* yang berguna untuk memisah kromosom/individu menjadi 2 bagian. Pada induk 1 diambil gen (sublist) mulai dari indeks 0 sampai indeks sebelum *insertion* point sehingga jumlah gen yang terambil akan sebanyak nilai dari *insertion point* seperti pada Kode 5.23 baris 259 dan 260 dimana sublist yang diambil pada induk 1 diberi nama *head*. Selanjutnya pada induk 2 akan diambil gen (sublist) sebanyak jumlah *insertion point* dari belakang tanpa ikut mengambil indeks terakhir dari induk 2 seperti pada baris 268 sampai 270 dimana sublist yang diambil pada induk 2 diberi nama *tail*. Selanjutnya membentuk keturunan dengan menggabungkan kedua sublist tersebut menjadi satu dan ditambah *node* akhir pada indeks terakhir. Pembentukan keturunan tersebut dapat dilihat pada Kode 5.24.

```
275  if (head.size() + tail.size() + 1
276          >= arrSelection.get(i).size()) {
277      //Memasukkan semua gen dari head ke keturuna
278      offSpring.addAll(head);
279      int index = 0;
280      //Memasukkan gen dari tail ke dalam keturuna
281      //dengan menyesuaikan ukuran Induk 1.
282      while (offSpring.size()
283              < arrSelection.get(i).size() - 1) {
284          if (index < tail.size()) {
285              if (!offSpring.contains(
286                      tail.get(index))) {
287                  offSpring.add(tail.get(index));
288              }
289              index++;
290          } else {
291              //Jika proses memasukkan tail gagal
292              //dalam offSpring, maka dimasukkan g
293              //sisa dari induk 1.
294              offSpring.addAll(cdg);
295          }
296      }
297      //Memasukkan gen akhir
298      offSpring.add(nodeAkhir);
299  } else {
300      offSpring.addAll(arrSelection.get(i));
301  }
```

Kode 5.24 Penyatuan SubList Induk 1 dan Induk 2

Pada Kode 5.22 baris 238, dibuat sebuah list yang akan menampung penggabungan SubList antara induk 1 dengan induk 2. Pada Kode 5.24 baris 275, dibuat sebuah kondisi dimana jika gabungan dari kedua SubList melebihi ukuran induk 1, maka perkawinan silang akan dilakukan. Pada Kode 5.24 baris 278, ditambahkan bagian gen dari induk 1 (SubList head). Selanjutnya pada baris 279 sampai 290 dilakukan penggabungan antara list (keturunan) yang telah berisi SubList induk 1 dengan SubList dari induk 2 jika ukuran keturunan lebih pendek dibandingkan induk 1 seperti yang ditunjukkan pada baris 282 dan 283. Sebelumnya, seperti pada baris 285 sampai 288, diperiksa terlebih dahulu apakah gen yang ada pada SubList induk 2 sudah ada pada list keturunan. Jika sudah ada maka gen yang sama tersebut tidak akan dimasukkan ke dalam list keturunan. Jika penggabungan subList induk 1 dan 2 belum mencukupi ukuran kromosom dari induk 1, maka seperti penjelasan sebelumnya, induk 1 akan menggantikan keturunan tersebut dengan menjalankan Kode 5.24 baris 278, 294, dan 298.

```
350     if (!arrOffSpring.contains(offSpring) && hitungWaktuTempuh(
351             (ArrayList<Integer>) offSpring) <= tMax) {
352         arrOffSpring.add((ArrayList<Integer>) offSpring);
353     }
```

Kode 5.25 Menyimpan Semua Keturunan ke dalam Satu ArrayList

Setelah iterasi untuk proses perkawinan silang selesai, maka keturunan hasil perkawinan silang akan dimasukkan ke dalam ArrayList arrOffSpring. Keturunan yang muncul lebih dari satu kali dan melebihi batasan waktu tempuh yaitu tMax (batasan dari model OP) tidak akan dimasukkan ke dalam ArrayList ini seperti yang ditunjukkan pada Kode 5.25.

```
357   //3.4. Mengisi array untuk waktu tempuh dan fitness
358   //individu dari hasil seleksi.
359   for (int i = 0; i < arrOffSpring.size(); i++) {
360       arrOffSpring_WaktuTempuh.add(
361               hitungWaktuTempuh(arrOffSpring.get(i)));
362       arrOffSpring_Fitness.add(
363               hitungFitness(arrOffSpring.get(i)));
364   }
```

Kode 5.26 Menyimpan Waktu Tempuh dan nilai Fitness Keturunan

Berdasarkan keturunan layak yang telah disimpan dalam ArrayList arrOffSpring, maka selanjutnya nilai waktu tempuh dan *fitness* akan dihitung dan disimpan pada ArrayListnya masing-masing yaitu arrOffSpring_WaktuTempuh dan arrOffSpring_Fitness seperti pada Kode 5.26.

5.2.6. Mutasi

Dari keturunan hasil perkawinan silang tersebut selanjutnya akan dimutasi menggunakan pencarian lokal dengan operator *add* dan *omit*. Seperti yang dijelaskan pada bagian 4.7.6, pencarian lokal akan dilakukan secara berurutan dan berulang-ulang hingga mencapai maksimum *loop*. Setiap hasil keturunan memiliki kesempatan untuk dimutasi asalkan probabilitas (nilai acak) yang dihasilkan saat mutasi lebih kecil dibandingkan probabilitas mutasi yang telah ditentukan sebelumnya.

```
372   //4. MELAKUKAN MUTASI (DENGAN LOCAL SEARCH)
373   for (int i = 0; i < arrOffSpring.size(); i++) {
374       double rndProbMutation = new Random().nextDouble();
375       if (rndProbMutation > probMutation) {
376           continue;
377       }
```

Kode 5.27 Looping dan Deklarasi Pengecualian Mutasi

```
379   int loopMutation = 0;
380   int maxLoopMutation = 1000;
381
382   //4.1. Operator Add.
383   do {
384
385       int rndInsert = new Random().nextInt(
386               arrOffSpring.get(i).size() - 2) + 1;
387       int rndNilaiInsert
388               = new Random().nextInt(jumlahNode) + 1;
389       if (arrOffSpring.get(i).contains(rndNilaiInsert)
390               || rndNilaiInsert == nodeAwal
391               || rndNilaiInsert == nodeAkhir) {
392           continue; //Memastikan node yang dimasukkan buk
393       }             //merupakan node awal dan akhir
394       arrOffSpring.get(i).add(rndInsert, rndNilaiInsert);
```

Kode 5.28 Looping Proses Mutasi dan Operasi Pencarian Lokal "Add"

Seperti yang ditunjukkan pada Kode 5.27 baris 375 sampai 377, mutasi akan dilakukan jika rndProbMutation yang merupakan bilangan double yang dihasilkan secara random lebih kecil dibandingkan probMutation, probabailitas mutasi yang telah ditentukan sebelumnya, yaitu 0.1. Jika lebih besar, maka mutasi pada keturunan indeks tersebut akan dibatalkan dan dilanjutkan dengan keturunan indeks setelahnya. Pada Kode 5.28 baris 379 dan 380 dapat dilihat bahwa *looping* dilakukan sebanyak 1000 iterasi. Maksud dari baris ini adalah iterasi akan berhenti jika tidak ditemukan lagi hasil mutasi yang lebih baik sampai iterasi ke-1000.

Berdasarkan Kode 5.28 ditunjukkan bahwa pada proses mutasi diawali dengan pencarian lokal add. Seperti pada penjelasan 4.7.6, pada pencarian lokal *add*, akan dimasukkan sebuah *node* (ditunjukkan pada baris 387 dan 388) yang tidak terdapat dalam kromosom keturunan pada indeks yang acak diantara indeks 1 sampai indeks sebelum gen terakhir (ditunjukkan pada baris 385 dan 386) agar batasan 1 model OP dapat terpenuhi (*node* awal dan akhir tidak berubah). Selain itu, dibuat juga ketentuan dimana tidak boleh ada gen yang sama muncul lebih dari satu kali seperti pada baris 389 sampai 393. Jika gen yang

dimasukkan sudah ada sebelumnya, sama dengan gen awal (indeks 0), atau sama dengan gen akhir (indeks terakhir), maka proses penambahan gen tersebut dibatalkan dan dilanjutkan dengan penambahan gen yang lain. Jika tidak melanggar ketentuan yang ada, maka proses *add* akan dilakukan pada keturunan tersebut.

```
396         //Menyimpan waktu tempuh dan fitness dari hasil proses add
397         int currentWaktuTempuh
398                 = hitungWaktuTempuh(arrOffSpring.get(i));
399         int currentFitness
400                 = hitungFitness(arrOffSpring.get(i));
401         //Jika proses add menghasilkan waktu tempuh dan fitness
402         //yang lebih baik atau sama dengan sebelumnya, maka
403         //hasil mutasi akan disimpan menggantikan yang lama.
404         if (currentWaktuTempuh <= arrOffSpring_WaktuTempuh.get(i)
405                 && currentFitness >= arrOffSpring_Fitness.get(i)) {
406             arrOffSpring_WaktuTempuh.set(i, currentWaktuTempuh);
407             arrOffSpring_Fitness.set(i, currentFitness);
408             loopMutation = 0; //looping dikembalikan ke awal
409         } else {
410             arrOffSpring.get(i).remove(rndInsert);
411         } //Jika tidak lebih baik, maka dikembalikan seperti semula
412
413         loopMutation++;
414
415     } while (loopMutation < maxLoopMutation);
```

Kode 5.29 Penggalan Proses Mutasi

Setelah dilakukan pencarian lokal dengan operator *add* pada keturunan, dihitung waktu tempuh dari keturunan tersebut seperti yang ada pada Kode 5.29 baris ke 397 dan 398 serta dihitung nilai *fitness*nya seperti yang ada pada baris 399 dan 400. Apabila setelah adanya penambahan gen, waktu tempuh setidaknya sama dengan atau lebih kecil dari waktu tempuh sebelumnya dan nilai *fitness* lebih besar dari nilai *fitness* sebelumnya, maka nilai waktu tempuh dan *fitness* baru akan disimpan ke dalam ArrayList arrOffSpring_WaktuTempuh dan arrOffSpring_ Fitness (seperti yang ditunjukkan pada baris 404 sampai 407. Selain itu jika pencarian lokal berhasil melakukan operator *add*, maka iterasi akan kembali ke iterasi awal (seperti yang ditunjukkan pada baris 408). Iterasi akan terus berlanjut/bertambah jika operator *add* gagal menghasilkan waktu tempuh dan nilai *fitness* yang lebih baik dari keturunan

sebelum mutasi (seperti pada baris 413) . Operator *add* gagal menambahkan gen sehingga gen yang ditambahkan pada kromosom keturunan akan dihapus seperti yang ditunjukkan baris 410. Iterasi pada operator *add* akan selesai jika sudah memenuhi jumlah maksimal iterasi yaitu 1000 (seperti yang ditunjukkan Kode 5.28 baris 380) yang mana artinya operator *add* tidak dapat menambahkan gen yang lebih baik sampai 1000 iterasi berturut turut. Setelah itu dilakukan pencarian lokal dengan operator omit dengan Kode 5.30.

```
418  do {
419      int rndDelete = new Random().nextInt(
420              arrOffSpring.get(i).size() - 2) + 1;
421      //Proses omit dilakukan jika ukuran keturunan lebih dari 3
422      if (arrOffSpring.get(i).size() > 3) {
423          //Temp menyimpan nilai dari gen/node yang dihapus
424          int temp = arrOffSpring.get(i).get(rndDelete);
425          arrOffSpring.get(i).remove(rndDelete);
426          //Menyimpan waktu tempuh & fitness hasil proses omit
427          int currentWaktuTempuh
428                  = hitungWaktuTempuh(arrOffSpring.get(i));
429          int currentFitness
430                  = hitungFitness(arrOffSpring.get(i));
431          if (currentWaktuTempuh
432                  < arrOffSpring_WaktuTempuh.get(i)) {
433              arrOffSpring_WaktuTempuh.set(i, currentWaktuTempuh);
434              arrOffSpring_Fitness.set(i, currentFitness);
435              loopMutation = 0; //looping dikembalikan ke awal
436          } else {
437              arrOffSpring.get(i).add(rndDelete, temp);
438          } //Jika tdk lebih baik, maka dikembalikan sprt semula
439
440      } else {
441          break;
442      }
443
444      loopMutation++;
445
446  } while (loopMutation < maxLoopMutation);
```

Kode 5.30 Pencarian Lokal dengan Operator Omit

Berdasarkan Kode 5.30, mirip seperti operator *add*, operator *omit* dimulai dengan mendeklarasikan variabel rndDelete yang merupakan bilangan random dengan rentang 1 sampai sebelum gen keturunan terakhir dengan fungsi menunjukkan indeks ke berapa dari gen keturunan yang akan dihapus. Sehingga

rndDelete tidak akan pernah menghapus gen awal dan akhir untuk menjaga batasan dari model OP (node awal dan akhir tetap). Penggunakan operator *omit* hanya dilakukan kepada keturunan yang memiliki panjang kromosom (jumlah gen) yang lebih besar dari 3 seperti yang ditunjukkan baris 422 sampai 442. Jika panjang kromosom kurang dari 3, maka operator *omit* akan dibatalkan dan dilanjutkan dengan keturunan setelahnya. Hal ini bertujuan untuk membatasi agar operator omit tidak melakukan penghapusan gen secara terus menerus.

Masuk ke dalam fungsi penghapusan gen, pada baris 424, dideklarasikan variabel *temp* yang menyimpan nilai dari gen yang terhapus. Pada baris 425 dijalankan penghapus gen berdasarkan indeks dari variabel rndDelete. Setelah itu dihitung waktu tempuh dan nilai *fitness*nya setelah gen terhapus seperti yang ditunjukkan pada baris 427 sampai 430. Apabila setelah penghapusan gen, waktu tempuh menjadi lebih kecil dari sebelumnya, maka waktu tempuh dan nilai *fitness* dari kromosom yang telah dihapus gennya tersebut akan disimpan serta iterasi akan kembali ke awal. Jika dalam penghapusan gen tidak menemukan waktu tempuh yang lebih baik, maka gen yang terhapus akan dikembalikan ke indeksnya semula. Iterasi pada operator omit akan selesai jika sudah memenuhi jumlah maksimal iterasi yaitu 1000 (seperti yang ditunjukkan baris 380) yang mana artinya operator *omit* tidak dapat menghapus gen untuk mengurangi waktu tempuh sampai 1000 iterasi berturut turut.

Pengoperasian pencarian lokal pada operator add dan omit disebut dengan 1 iterasi mutasi. Iterasi akan dijalankan sebanyak jumlah keturunan yang dihasilkan pada proses perkawinan silang. Setelah iterasi selesai, dengan begitu maka proses mutasi berakhir.

5.2.7. Pembentukan Populasi Baru

Populasi baru akan dipilih dari gabungan antara populasi awal dan keturunan hasil mutasi. Untuk itu gabungan antara populasi

awal dan keturunan hasil mutasi akan disimpan ke dalam ArrayList arrSimpanan, beserta waktu tempuh masing-masing gabungan individu ke dalam ArrayList dengan nama arrSimpanan_WaktuTempuh dan nilai *fitness* dari masing-masing gabungan individu ke dalam ArrayList dengan nama arrSimpanan_Fitness seperti Kode 5.31.

```
62    static ArrayList<ArrayList<Integer>> arrSimpanan
63            = new ArrayList<ArrayList<Integer>>();
64    static ArrayList<Integer> arrSimpanan_WaktuTempuh
65            = new ArrayList<Integer>();
66    static ArrayList<Integer> arrSimpanan_Fitness
67            = new ArrayList<Integer>();
```

Kode 5.31 Deklarasi ArrayList sebagai Tempat Penyimpanan Populasi Sementara beserta Waktu Tempuh dan Nilai Fitness

Sebelum arrSimpanan menyimpan individu dari populasi awal dan keturunan hasil mutasi, dipastikan terlebih dahulu bahwa arrSimpanan beserta waktu tempuh dan *fitness*nya tidak berisi nilai (kosong). Maka dari itu dibuat fungsi untuk menghapus ketiga ArrayList tersebut seperti pada Kode 5.32.

```
457    //Memastikan array utk simpan in
458    arrSimpanan.clear();
459    arrSimpanan_WaktuTempuh.clear();
460    arrSimpanan_Fitness.clear();
```

Kode 5.32 Penghapusan Individu dari Tempat Penyimpanan

Populasi baru yang dibentuk merupakan gabungan dari individu populasi awal dan keturunan hasil mutasi. Semua individu tersebut akan dimasukkan ke dalam populasi baru seperti Kode 5.33. Setiap individu yang disimpan ke tempat penyimpanan sementara bersifat *unique* (tidak sama satu sama yang lain).

```
462   //Memasukkan populasi ke dalam array simpanan.
463   for (int i = 0; i < arrPopulation.size(); i++) {
464       if (!arrSimpanan.contains(arrPopulation.get(i))) {
465           arrSimpanan.add(arrPopulation.get(i));
466       }
467   }
468
469   //Memasukkan keturunan ke dalam array simpanan
470   for (int i = 0; i < arrOffSpring.size(); i++) {
471       if (arrOffSpring_WaktuTempuh.get(i) <= tMax
472               && !arrSimpanan.contains(arrOffSpring.get(i))) {
473           arrSimpanan.add(arrOffSpring.get(i));
474       }
475   }
```

Kode 5.33 Penyimpanan Individu Hasil Seleksi dan Keturunan Hasil Mutasi.

Berdasarkan gabungan individu yang telah disimpan dalam ArrayList arrSimpanan, maka selanjutnya nilai waktu tempuh dan *fitness* akan dihitung dan disimpan pada ArrayListnya masing-masing yaitu arrSimpanan_WaktuTempuh dan arrSimpanan_Fitness seperti pada Kode 5.34.

```
477   //Mengisi array untuk waktu tempuh dan fitness setia
478   //individu dari array simpanan.
479   for (int i = 0; i < arrSimpanan.size(); i++) {
480       arrSimpanan_WaktuTempuh.add(
481               hitungWaktuTempuh(arrSimpanan.get(i)));
482       arrSimpanan_Fitness.add(
483               hitungFitness(arrSimpanan.get(i)));
484   }
```

Kode 5.34 Iterasi Menyimpan Waktu Tempuh dan Fitness Populasi Sementara ke dalam ArrayList

Selanjutnya menghapus isi ArrayList dari populasi (arrPopulation), hasil seleksi (arrSelection), dan keturunan (arrOffSpring) seperti pada Kode 5.35. Untuk seleksi dan keturunan dihapus agar dapat digunakan kembali pada iterasi/generasi selanjutnya.

```
486     //Mengosongkan isi dari array yang
487     arrPopulation.clear();
488     arrPopulation_WaktuTempuh.clear();
489     arrPopulation_Fitness.clear();
490
491     //Mengosongkan isi dari array yang
492     arrSelection.clear();
493     arrSelection_WaktuTempuh.clear();
494     arrSelection_Fitness.clear();
495
496     //Mengosongkan isi dari array yang
497     arrOffSpring.clear();
498     arrOffSpring_WaktuTempuh.clear();
499     arrOffSpring_Fitness.clear();
```

Kode 5.35 Penghapusan isi Populasi, Seleksi, dan Keturunan

Berdasarkan Kode 5.36, isi dari populasi yang telah dikosongkan akan diisi kembali sebanyak jumlah populasi yang telah ditentukan yaitu 50 individu. Baris 508 sampai 515 berfungsi untuk mencari satu individu/solusi terbaik berdasarkan nilai *fitness* dari arrSimpanan. Lalu satu solusi terbaik tersebut akan dimasukkan ke dalam populasi baru (ArrayList arrPopulation) begitu juga dengan waktu tempuh dan nilai *fitness*nya seperti pada baris 516 sampai 520. Selanjutnya solusi terbaik tersebut akan dihapus dari arrSimpanan seperti pada baris 521 sampai 523. Lalu akan diulangi kembali iterasi tersebut, mulai dari pencarian solusi terbaik pada arrSimpanan yang tersisa untuk dipindahkan ke populasi baru (arrPopulasi) dan penghapusan solusi tersebut pada arrSimpanan sampai memenuhi jumlah populasi.

```
505  //Memilih individu terbaik dari populasi sebelumnya dan
506  //sejumlah populasi sebelumnya.
507  do {
508      int indSolusi = 0;
509      int optFitness = arrSimpanan_Fitness.get(0);
510      for (int i = 1; i < arrSimpanan.size(); i++) {
511          if (arrSimpanan_Fitness.get(i) >= optFitness) {
512              indSolusi = i;
513              optFitness = arrSimpanan_Fitness.get(i);
514          }
515      }
516      arrPopulation.add(arrSimpanan.get(indSolusi));
517      arrPopulation_WaktuTempuh.add(
518              arrSimpanan_WaktuTempuh.get(indSolusi));
519      arrPopulation_Fitness.add(
520              arrSimpanan_Fitness.get(indSolusi));
521      arrSimpanan.remove(indSolusi);
522      arrSimpanan_WaktuTempuh.remove(indSolusi);
523      arrSimpanan_Fitness.remove(indSolusi);
524  } while (arrPopulation.size() < jumlahPop);
```

Kode 5.36 Pemilihan Individu Terbaik untuk Populasi Baru Berdasarkan Populasi Sebelumnya dan Keturunan

5.2.8. Pengambilan Solusi Terbaik

Setelah jumlah generasi terpenuhi, maka dilakukan pengambilan solusi terbaik seperti Kode 5.37. Pengambilan solusi terbaik tersebut dilakukan dengan cara membandingkan dengan runtut indeks satu dengan indeks lainnya secara bergantian. Maka dari itu dideklarasikan terlebih dahulu variabel indSolusi yang digunakan untuk menyimpan indeks dari solusi terbaik pada ArrayList populasi dan variabel optFitness yang digunakan untuk menyimpan nilai *fitness* dari solusi terbaik pada ArrayList Populasi. Kedua variabel tersebut diinisiasi dengan nilai 0 sehingga dianggap solusi terbaik sementara adalah individu pada indeks ke 0 seperti yang ditunjukkan pada baris 539 dan 540.

```
538         //Mencari individu terbaik dari populasi baru yang terbentuk
539         int indSolusi = 0;
540         int optFitness = arrPopulation_Fitness.get(0);
541         for (int i = 1; i < arrPopulation.size(); i++) {
542             if (arrPopulation_Fitness.get(i) >= optFitness) {
543                 indSolusi = i;
544                 optFitness = arrPopulation_Fitness.get(i);
545             }
546         }
547 
548         //Mencetak hasil individu terbaik dari setiap generasi
549         pw.println("Generasi " + (generasi + 1) + " | Rute: "
550                 + arrPopulation.get(indSolusi) + " | Waktu Tempuh: "
551                 + arrPopulation_WaktuTempuh.get(indSolusi) + " | Fitness: "
552                 + arrPopulation_Fitness.get(indSolusi));
553 
554         generasi++;
555 
556     } while (generasi < jumlahGenerasi);
```

Kode 5.37 Pencarian Solusi Terbaik

Selanjutnya dilakukan iterasi untuk membandingkan satu per satu individu yang ada dalam populasi pada arrPopulation dimana populasi yang memiliki nilai *fitness* tertinggi akan terpilih sebagai solusi terbaik seperti yang ditunjukkan pada baris 542 sampai 545. Pada baris 549 sampai 552 digunakan untuk mencetak solusi terbaik yang telah ditemukan beserta waktu tempuh dan nilai *fitness* dari solusi tersebut.

“Halaman ini sengaja dikosongkan”

BAB VI
HASIL DAN PEMBAHASAN

Bab ini akan menjelaskan hasil yang didapatkan dari penelitian ini dan pembahasan secara keseluruhan yang didapatkan dari penelitian.

6.1. Lingkungan Uji Coba

Lingkungan uji coba merupakan kriteria perangkat pengujian yang digunakan untuk implementasi *Genetic Algorithm* (GA) pada model *Orienteering Problem* (OP) dalam penyelesaian permasalahan pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang digunakan. Spesifikasi dari perangkat keras yang digunakan dalam implementasi metode ditunjukkan pada Tabel 6.1.

Tabel 6.1 Lingkungan Uji Coba Perangkat Keras

PERANGKAT KERAS	SPESIFIKASI
Jenis	Laptop
Processor	Intel® Core™ i5 4200U
RAM	4096MB
Hard Disk Drive	500GB

Selain perangkat keras, terdapat lingkungan perangkat lunak yang digunakan dalam implementasi metode seperti yang ditunjukkan pada Tabel 6.2.

Tabel 6.2 Lingkungan Uji Coba Perangkat Lunak

PERANGKAT LUNAK	FUNGSI
Windows 10	Sistem Operasi
Corel Draw X8	Membuat *network model*
NetBeans IDE 8.1	Membuat model matematis
	Melakukan optimasi
Java Development Kit 1.8	Bahasa pemrograman
Microsoft Office Excel 2016	Merekap data

6.2. Penggambaran Network Model

Network model digunakan sebagai gambaran dari persebaran *node* yang telah ditentukan sebelumnya dengan tujuan untuk mempermudah pembuatan model *Orienteering Probelm* (OP). Dalam *network model* tersebut, *node* yang merepresentasikan tempat pemberhentian angkot dilambangkan dalam bentuk titik-titik hitam, sedangkan *arc* yang menghubungkan antar *node* dilambangkan dalam bentuk garis dengan anak panah beserta satuan waktunya. Berdasarkan penggambaran trayek pada Gambar G.1, secara keseluruhan *network model* dapat dilihat pada LAMPIRAN H.

6.3. Hasil Pembaruan Pencarian Waktu Tempuh dengan Algoritma Dijkstra

Setelah program berhasil dijalankan seperti pada penjelasan sub bab 3.1.6.1, program tersebut akan menghasilkan sebuah matriks yang berisikan setiap waktu tempuh antar *node* yang disimpan ke dalam format .csv. Berikut di bawah ini merupakan potongan matriks hasil keluaran algoritma *Dijkstra*.

Tabel 6.3 Penggalan Hasil Algoritma Dijkstra

node	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	9	4	5	6	7
2	1	0	1	2	3	8	3	4	5	6
3	2	1	0	1	2	7	2	3	4	5
4	3	2	1	0	3	6	1	2	3	4
5	11	10	9	8	0	7	2	3	4	5
6	4	3	2	1	4	0	2	3	4	5
7	9	8	7	6	9	5	0	1	2	3
8	8	7	6	5	8	4	6	0	1	2
9	7	6	5	4	7	3	5	6	0	1
10	6	5	4	3	6	2	4	5	6	0

Dari penggalan matriks di atas, dapat dilihat bahwa algoritma *Dijkstra* dapat menemukan waktu tempuh tercepat antar *node*. Sehingga semua *node* yang ada pada *network model* sudah terhubung secara langsung dan setiap *cell* pada matriks sudah terisi.

6.4. Hasil Pencarian Solusi dengan Genetic Algorithm

Sesuai dengan pembahasan pada bagian 4.4 dan 5.2, program akan dijalankan untuk mencari solusi dari permasalahan pencarian rute optimum dari node 1 (Jl. Kalimas Barat No.63, Pangkalan Petekan) menuju node 166 (Jl. Wonorejo No.18, Pangkalan Manukon). Dalam studi kasus diatur waktu tempuh maksimal adalah 70 menit.

6.4.1. Validasi Solusi Layak Awal

Sebelum solusi optimal dapat ditemukan, program akan mencari solusi layak awal yang akan menjadi luaran (*output*) dari proses pembangkitan populasi awal dan menjadi input untuk proses selanjutnya, yaitu seleksi sampai mutasi. Pencarian solusi layak awal dilakukan dengan iterasi sampai memenuhi jumlah populasi yang telah ditentukan sebelumnya, yaitu 50 individu. Setiap solusi/individu disimpan dalam bentuk *array*. Berikut Tabel 6.4 yang merupakan hasil dari pembangkitan populasi awal.

Tabel 6.4 Hasil Pembangkitan Populasi Awal

Individu	[Solusi] (Waktu Tempuh, Skor)
1	[1, 13, 166] (58, 496)
2	[1, 83, 166] (58, 512)
3	[1, 75, 166] (70, 428)
4	[1, 107, 17, 166] (63, 828)
5	[1, 12, 105, 166] (58, 699)
6	[1, 107, 111, 18, 166] (68, 1076)
7	[1, 135, 166] (64, 402)

Indi vidu	[Solusi] (Waktu Tempuh, Skor)
8	[1, 111, 110, 166] (66, 541)
9	[1, 124, 162, 166] (70, 727)
10	[1, 96, 166] (58, 367)
11	[1, 156, 166] (58, 367)
12	[1, 23, 166] (66, 398)
13	[1, 134, 166] (64, 402)
14	[1, 128, 166] (64, 395)
15	[1, 22, 81, 166] (66, 627)
16	[1, 77, 166] (62, 428)
17	[1, 84, 166] (58, 512)
18	[1, 2, 156, 166] (58, 566)
19	[1, 18, 166] (58, 615)
20	[1, 134, 78, 166] (66, 547)
21	[1, 105, 166] (58, 486)
22	[1, 99, 166] (58, 367)
23	[1, 152, 166] (64, 428)
24	[1, 12, 166] (58, 496)
25	[1, 96, 75, 166] (70, 512)
26	[1, 151, 166] (66, 428)
27	[1, 95, 166] (64, 395)
28	[1, 133, 166] (64, 402)
29	[1, 131, 166] (60, 395)
30	[1, 101, 166] (58, 367)
31	[1, 29, 88, 83, 166] (66, 711)
32	[1, 15, 166] (58, 496)
33	[1, 19, 105, 166] (62, 601)
34	[1, 161, 166] (58, 367)
35	[1, 111, 98, 166] (66, 496)
36	[1, 87, 166] (58, 367)

Individu	[Solusi] (Waktu Tempuh, Skor)
37	[1, 22, 166] (66, 398)
38	[1, 99, 91, 166] (58, 563)
39	[1, 79, 166] (58, 512)
40	[1, 153, 150, 166] (68, 573)
41	[1, 125, 166] (68, 514)
42	[1, 159, 166] (58, 367)
43	[1, 76, 166] (66, 428)
44	[1, 104, 87, 166] (58, 570)
45	[1, 160, 166] (58, 367)
46	[1, 78, 166] (60, 428)
47	[1, 5, 166] (60, 496)
48	[1, 138, 166] (64, 402)
49	[1, 6, 166] (65, 611)
50	[1, 25, 77, 166] (70, 543)

Solusi yang dihasilkan dikatakan layak dan valid apabila tidak melanggar batasan-batasan dari model OP yang telah ditentukan sebelumnya, yaitu:

1. Batasan 1 terpenuhi. Solusi memiliki *node* awal dan akhir yang tetap. Dapat dilihat dari 50 individu/solusi yang dihasilkan memiliki *node*/gen awal dan akhir yang tetap sesuai dengan studi kasus permasalahan yaitu *node* 1 dan *node* 166.
2. Batasan 2 terpenuhi. Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali. Dari 50 solusi awal yang dihasilkan, tidak ada solusi yang memiliki gen/*node* ganda.
3. Batasan 3 terpenuhi. Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan (tMax). Setiap solusi yang dihasilkan memiliki waktu

tempuh yang lebih sedikit atau sama dengan tMax yaitu 70 menit.

4. Batasan 4 dan 5 terpenuhi. Tidak boleh ada *subtours* yaitu lintasan dimulai dan berakhir pada *node* yang sama. Ini dibuktikan dari gen awal dan akhir berbeda dari solusi yang dihasilkan yaitu 1 dan 166. Selain itu gen awal dan akhir hanya muncul satu kali pada setiap solusi.

Dari empat pernyataan di atas, dapat disimpulkan bahwa solusi awal yang digunakan untuk proses pembangkitan populasi awal adalah layak dan telah tervalidasi.

6.4.2. Validasi Solusi Akhir

Setelah dilakukan pembangkitan populasi awal sebanyak 50 individu, selanjutnya dilakukan proses seleksi, perkawinan silang, dan mutasi yang dilakukan secara berurutan dalam beberapa generasi untuk mendapatkan solusi akhir. Untuk mendapatkan solusi akhir diperlukan proses yang panjang mulai dari seleksi dengan memilih 50 induk dari populasi awal, perkawinan silang yang menghasilkan keturunan cara dengan menyilangkan gen induk 1 dan induk 2 sesuai probabilitas *crossover* yaitu 0.9, serta mutasi yaitu memperkaya gen keturunan dengan menggunakan pencarian lokal *add* dan *omit* sesuai dengan probabilitas mutasi, yaitu 0.1. Setelah itu, 50 individu terbaik (berdasarkan nilai *fitness*) dari gabungan keturunan hasil mutasi ditambah dengan populasi sebelumnya akan menjadi populasi baru dan akan digunakan untuk iterasi selanjutnya. Berikut ini pada Tabel 6.5 merupakan populasi akhir yang berisikan 50 individu (diurutkan berdasakan nilai *fitness*) dengan jumlah generasi sebanyak 300 iterasi.

Tabel 6.5 Hasil Populasi Akhir

Indi vidu	[Solusi] (Waktu Tempuh, Skor)
1	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (62, 7095)

Indi vidu	[Solusi] (Waktu Tempuh, Skor)
2	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 7095)
3	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 96, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (66, 7011)
4	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 87, 96, 88, 86, 85, 84, 83, 81, 79, 82, 166] (70, 7011)
5	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 90, 96, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (70, 7011)
6	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 96, 91, 92, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (66, 7011)
7	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 88, 96, 87, 86, 85, 84, 83, 81, 79, 82, 166] (68, 7011)
8	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 7011)
9	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 87, 86, 85, 88, 84, 83, 81, 79, 82, 166] (66, 7011)
10	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (62, 7011)
11	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 156, 166] (58, 7011)
12	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 102, 97, 96, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (68, 7011)
13	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 7011)
14	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 101, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (66, 7011)
15	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 7011)

Indi vidu	[Solusi] (Waktu Tempuh, Skor)
16	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 96, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (60, 7011)
17	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (58, 7011)
18	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (64, 7011)
19	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (64, 7011)
20	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 102, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (68, 7011)
21	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 96, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (68, 7011)
22	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 91, 92, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (60, 7011)
23	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (62, 7011)
24	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 101, 97, 92, 91, 96, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (70, 7011)
25	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 87, 86, 88, 85, 84, 83, 81, 79, 82, 166] (62, 7011)
26	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 96, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (60, 7011)
27	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 101, 97, 92, 96, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (68, 7011)
28	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 102, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (68, 7011)
29	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 101, 97, 96, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (66, 7011)

Indi vidu	[Solusi] (Waktu Tempuh, Skor)
30	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 96, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (68, 7011)
31	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 96, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 166] (64, 6927)
32	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 89, 87, 86, 88, 85, 84, 83, 81, 79, 166] (66, 6927)
33	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 97, 92, 91, 90, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 156, 166] (58, 6927)
34	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166] (58, 6927)
35	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 6927)
36	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 96, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166] (68, 6927)
37	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 88, 87, 86, 89, 85, 84, 83, 81, 79, 82, 166] (64, 6927)
38	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 88, 87, 86, 89, 85, 84, 83, 81, 79, 166] (68, 6927)
39	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166] (64, 6927)
40	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 102, 97, 96, 91, 92, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166] (70, 6927)
41	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 98, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (58, 6927)
42	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 91, 96, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 166] (68, 6927)
43	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 87, 86, 85, 88, 84, 83, 81, 79, 82, 166] (66, 6927)

Individu	[Solusi] (Waktu Tempuh, Skor)
44	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 96, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 166] (66, 6927)
45	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 97, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 156, 166] (58, 6927)
46	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 98, 102, 97, 92, 91, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 155, 166] (68, 6927)
47	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 101, 100, 99, 102, 98, 97, 92, 96, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166] (66, 6927)
48	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 91, 92, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 166] (60, 6927)
49	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 96, 90, 89, 87, 86, 85, 84, 83, 81, 79, 82, 166] (62, 6927)
50	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 96, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166] (60, 6927)

Solusi yang dihasilkan dikatakan layak apabila tidak melanggar batasan-batasan dari model OP yang telah ditentukan sebelumnya, yaitu:

1. Batasan 1 terpenuhi. Rute memiliki *node* awal dan akhir yang tetap. Dapat dilihat dari individu/solusi iterasi akhir yang dihasilkan memiliki *node* awal dan akhir yang tetap sesuai dengan studi kasus permasalahan yaitu *node* 1 dan *node* 166.
2. Batasan 2 terpenuhi. Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali. Dari solusi akhir yang dihasilkan, tidak ada solusi yang memiliki gen/*node* ganda.
3. Batasan 3 terpenuhi. Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan (tMax). Setiap solusi akhir yang dihasilkan memiliki waktu tempuh yang lebih sedikit atau sama dengan tMax.

4. Batasan 4 dan 5 terpenuhi. Tidak boleh ada subtours, lintasan dimulai dan berakhir pada *node* yang sama. Ini dibuktikan dari gen awal dan akhir dari setiap individu/solusi berbeda. Selain itu gen awal dan akhir hanya muncul satu kali pada setiap solusi.

Seperti yang ditunjukkan pada Tabel 6.5, semua solusi yang dihasilkan masih layak dan valid karena memenuhi batasan model OP dan selain itu semua solusi mengalami peningkatan skor. Dari solusi-solusi tersebut diambil solusi terbaik dengan melihat jumlah skor yang paling tinggi. Karena diurutkan berdasarkan nilai *fitness*, maka didapatkan solusi dengan nilai *fitness* tertinggi adalah solusi pada baris pertama dengan nilai *fitness* sebesar 7095.

Solusi dengan nilai *fitness* tertinggi tersebut dianggap sebagai rekomendasi rute yang paling optimum untuk studi kasus permasalahan ini dengan lintasan mulai dari *node* 1 (Jl. Kalimas Barat No.63) → 2 (Jl. Kalimas Barat No.45) → 3 (Jl. Kalimas Barat No.3) → 4 (Jl. Kasuari No.12) → 7 (Jl. Jembatan Merah No.3) → 8 (Jl. Veteran No.6-8) → 9 (Jl. Kebon Rojo) → 12 (Jl. Indrapura No.17) → 13 (Jl. Indrapura No.43) → 15 (Jl. Rajawali No.90) → 18 (Jl. Gresik Gadukan Timur No.20) → 107 (Jl. Gresik Gadukan Timur No.47) → 106 (Jl. Demak No.449) → 105 (Jl. Demak No.377) → 104 (Jl. Demak 301-303) → 103 (Jl. Demak 233-235) → 102 (Jl. Demak No.171) → 101 (Jl. Demak No.97) → 100 (Jl. Kalibutuh No.26) → 99 (Jl. Tidar No.314) → 98 (Jl. Pacuan Kuda No.42) → 97 (Jl.Petemon II No.148) → 92 (Jl. Banyu Urip No.320) → 91 (Jl. Banyu Urip No.228) → 96 (Jl. Simo Kwagean No.41) → 90 (Jl. Raya Simo Gunung 66-68) → 89 (Jl. Raya Simo Gunung No.6 B) → 88 (Jl. Kupang Jaya No.25) → 87 (Jl. Kupang Jaya No.112B) → 86 (Jl. Pattimura No.20) → 85 (Jl. Darmo Baru Barat XII No.63) → 84 (Jl. Raya Darmo Permai III Blok A No.12) → 83 (Jl. Raya Darmo Permai II No.14) → 81 (Jl. Mayjen Yono Suwoyo No.46) → 79 (Jl. Mayjen HR. Moh. No.368) → 82 (Jl. Raya Darmo Permai Selatan 17-19) → 155 (Jl. Simpang Darmo Permai Selatan 28-40) → 166 (Jl. Wonorejo No.18) dimana waktu tempuhnya adalah 62 menit.

6.5. Hasil dan Pembahasan Uji Coba 1: Mengubah Node Awal dan Akhir

Setelah menyelesaikan studi kasus permasalahan dari *node* 1 menuju *node* 166, pada bagian ini dilakukan uji coba dengan mencari rute optimum dari *node* 188 (Jl. Kalimas Baru No.5, Pangkalan Ujung Baru) ke *node* 42 (Jl. Ngagel Jaya Selatan No.20) dengan jumlah populasi, Tmax, probabilitas perkawinan silang, dan probabilitas mutasi yang sama dengan studi kasus sebelumnya, yaitu masing-masing 50 individu, 70 menit, 0.9, dan 0.1. Tujuan dari uji coba ini adalah memastikan bahwa program yang dibuat dapat digunakan untuk studi kasus dengan *node* awal dan akhir yang berbeda dimana masih dapat menghasilkan solusi yang layak dan memenuhi batasan model OP.

6.5.1. Validasi Solusi Layak Awal

Sebelum solusi optimal dapat ditemukan, program akan mencari solusi layak awal yang akan menjadi luaran (*output*) dari proses pembangkitan populasi awal dan menjadi input untuk proses selanjutnya, yaitu seleksi sampai mutasi. Pencarian solusi layak awal dilakukan dengan iterasi sampai memenuhi jumlah populasi yang telah ditentukan sebelumnya, yaitu 50 individu. Setiap solusi/individu disimpan dalam bentuk *array*. Berikut Tabel 6.6 yang merupakan hasil dari pembangkitan populasi awal.

Tabel 6.6 Hasil Pembangkitan Populasi Awal Uji Coba 1

Individu	[Solusi] (Waktu Tempuh, Skor)
1	[188, 26, 42] (66, 349)
2	[188, 97, 42] (56, 318)
3	[188, 137, 98, 42] (66, 437)
4	[188, 31, 42] (56, 461)
5	[188, 189, 42] (56, 353)
6	[188, 140, 42] (60, 353)
7	[188, 138, 90, 42] (66, 437)

Individu	[Solusi] (Waktu Tempuh, Skor)
8	[188, 2, 3, 9, 42] (70, 1089)
9	[188, 112, 42] (68, 363)
10	[188, 113, 42] (70, 363)
11	[188, 88, 42] (64, 318)
12	[188, 138, 37, 42] (64, 468)
13	[188, 132, 130, 42] (64, 465)
14	[188, 127, 42] (64, 465)
15	[188, 101, 42] (56, 318)
16	[188, 2, 42] (70, 433)
17	[188, 7, 42] (66, 562)
18	[188, 184, 42] (56, 353)
19	[188, 28, 42] (66, 349)
20	[188, 129, 31, 42] (64, 573)
21	[188, 110, 34, 42] (62, 478)
22	[188, 110, 42] (62, 363)
23	[188, 139, 42] (62, 353)
24	[188, 92, 42] (56, 430)
25	[188, 31, 95, 42] (60, 573)
26	[188, 185, 42] (56, 353)
27	[188, 20, 42] (70, 349)
28	[188, 154, 31, 42] (70, 606)
29	[188, 33, 42] (56, 349)
30	[188, 8, 42] (66, 562)
31	[188, 41, 42] (56, 349)
32	[188, 184, 29, 42] (62, 468)
33	[188, 21, 42] (66, 349)
34	[188, 90, 42] (60, 318)
35	[188, 22, 24, 42] (66, 464)
36	[188, 135, 42] (64, 353)

Individu	[Solusi] (Waktu Tempuh, Skor)
37	[188, 34, 42] (56, 349)
38	[188, 37, 42] (56, 349)
39	[188, 130, 42] (62, 346)
40	[188, 35, 42] (56, 349)
41	[188, 22, 42] (66, 349)
42	[188, 100, 42] (56, 318)
43	[188, 93, 42] (56, 346)
44	[188, 5, 13, 33, 42] (70, 775)
45	[188, 31, 43, 42] (62, 576)
46	[188, 95, 31, 42] (56, 573)
47	[188, 136, 42] (64, 353)
48	[188, 39, 42] (66, 349)
49	[188, 137, 42] (64, 353)
50	[188, 44, 42] (64, 349)

Solusi yang dihasilkan dikatakan layak dan valid apabila tidak melanggar batasan-batasan dari model OP yang telah ditentukan sebelumnya, yaitu:

1. Batasan 1 terpenuhi. Solusi memiliki *node* awal dan akhir yang tetap. Dapat dilihat dari 50 individu/solusi yang dihasilkan memiliki *node*/gen awal dan akhir yang tetap sesuai dengan studi kasus permasalahan yaitu *node* 188 dan *node* 42.
2. Batasan 2 terpenuhi. Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali. Dari 50 solusi awal yang dihasilkan, tidak ada solusi yang memiliki gen/*node* ganda.
3. Batasan 3 terpenuhi. Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan (tMax). Setiap solusi yang dihasilkan memiliki waktu tempuh yang lebih sedikit atau sama dengan tMax yaitu 70 menit.

4. Batasan 4 dan 5 terpenuhi. Tidak boleh ada *subtours* yaitu lintasan dimulai dan berakhir pada *node* yang sama. Ini dibuktikan dari gen awal dan akhir berbeda dari solusi yang dihasilkan yaitu 188 dan 42. Selain itu gen awal dan akhir hanya muncul satu kali pada setiap solusi.

Dari empat pernyataan di atas, dapat disimpulkan bahwa solusi awal yang digunakan untuk proses pembangkitan populasi awal pada uji coba ini adalah layak dan telah tervalidasi.

6.5.2. Validasi Solusi Akhir

Setelah dilakukan pembangkitan populasi awal sebanyak 50 individu, selanjutnya dilakukan proses seleksi, perkawinan silang, dan mutasi yang dilakukan secara berurutan dalam beberapa generasi untuk mendapatkan solusi akhir. Untuk mendapatkan solusi akhir diperlukan proses yang panjang mulai dari seleksi dengan memilih 50 induk dari populasi awal, perkawinan silang yang menghasilkan keturunan cara dengan menyilangkan gen induk 1 dan induk 2 sesuai probabilitas *crossover* yaitu 0.9, serta mutasi yaitu memperkaya gen keturunan dengan menggunakan pencarian lokal *add* dan *omit* sesuai dengan probabilitas mutasi, yaitu 0.1. Setelah itu, 50 individu terbaik (berdasarkan nilai *fitness*) dari gabungan keturunan hasil mutasi ditambah dengan populasi sebelumnya akan menjadi populasi baru dan akan digunakan untuk iterasi selanjutnya. Berikut ini pada Tabel 6.7 merupakan populasi akhir yang berisikan 50 individu (diurutkan berdasakan nilai *fitness*) dengan jumlah generasi sebanyak 300 iterasi.

Tabel 6.7 Hasil Populasi Akhir Uji Coba 1

Individu	[Solusi] (Waktu Tempuh, Skor)
1	[188, 187, 186, 185, 184, 183, 182, 189, 190, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 105, 104, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 35, 36, 37, 42] (70, 7983)
2	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7929)

Individu	[Solusi] (Waktu Tempuh, Skor)
3	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7849)
4	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7849)
5	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7845)
6	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7845)
7	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7814)
8	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7765)
9	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 93, 94, 95, 31, 32, 33, 34, 35, 42] (70, 7736)
10	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7730)
11	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7730)
12	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7730)
13	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7706)
14	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 35, 42] (66, 7667)
15	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7646)
16	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7646)

Individu	[Solusi] (Waktu Tempuh, Skor)
17	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7642)
18	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 42] (66, 7636)
19	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7632)
20	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 93, 94, 95, 31, 32, 33, 42] (70, 7625)
21	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7622)
22	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7622)
23	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7622)
24	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7611)
25	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 13, 18, 107, 106, 105, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 35, 36, 37, 40, 42] (66, 7596)
26	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7587)
27	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 42] (66, 7552)
28	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 42] (66, 7552)
29	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 100, 98, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 42] (68, 7552)
30	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 103, 102, 101, 100, 99, 98, 97, 96, 93, 94, 95, 31, 32, 33, 34, 35, 42] (70, 7533)

Individu	[Solusi] (Waktu Tempuh, Skor)
31	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 105, 104, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7523)
32	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (66, 7521)
33	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 93, 94, 31, 32, 33, 42] (70, 7513)
34	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7513)
35	[188, 187, 186, 185, 184, 183, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 18, 107, 106, 105, 104, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7513)
36	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 93, 94, 95, 31, 32, 42] (70, 7510)
37	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7503)
38	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 42] (70, 7503)
39	[188, 187, 186, 185, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 42] (70, 7486)
40	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 2, 3, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 42] (70, 7479)
41	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 34, 42] (66, 7468)
42	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 104, 105, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (68, 7437)
43	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (66, 7437)
44	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 105, 101, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7437)

Individu	[Solusi] (Waktu Tempuh, Skor)
45	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (66, 7437)
46	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (66, 7437)
47	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (66, 7437)
48	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 105, 102, 101, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7437)
49	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 104, 103, 105, 102, 100, 99, 98, 97, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (70, 7437)
50	[188, 187, 186, 185, 184, 183, 182, 189, 190, 191, 16, 15, 14, 6, 4, 7, 8, 9, 12, 13, 18, 107, 106, 104, 105, 103, 102, 101, 100, 99, 98, 96, 92, 93, 94, 95, 30, 31, 32, 33, 42] (68, 7437)

Solusi yang dihasilkan dikatakan layak apabila tidak melanggar batasan-batasan dari model OP yang telah ditentukan sebelumnya, yaitu:

1. Batasan 1 terpenuhi. Rute memiliki *node* awal dan akhir yang tetap. Dapat dilihat dari individu/solusi iterasi akhir yang dihasilkan memiliki *node* awal dan akhir yang tetap sesuai dengan studi kasus permasalahan yaitu *node* 188 dan *node* 42.
2. Batasan 2 terpenuhi. Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali. Dari solusi akhir yang dihasilkan, tidak ada solusi yang memiliki gen/*node* ganda.
3. Batasan 3 terpenuhi. Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan (tMax). Setiap solusi akhir yang dihasilkan memiliki waktu tempuh yang lebih sedikit atau sama dengan tMax.
4. Batasan 4 dan 5 terpenuhi. Tidak boleh ada subtours, lintasan dimulai dan berakhir pada *node* yang sama. Ini dibuktikan dari gen awal dan akhir dari setiap

individu/solusi berbeda. Selain itu gen awal dan akhir hanya muncul satu kali pada setiap solusi.

Dapat dilihat pada Tabel 6.7, semua solusi yang dihasilkan masih layak dan valid karena memenuhi batasan model OP dan selain itu semua solusi mengalami peningkatan skor. Dari solusi-solusi tersebut diambil solusi terbaik dengan melihat jumlah skor yang paling tinggi. Karena diurutkan berdasarkan nilai *fitness*, maka didapatkan solusi dengan nilai *fitness* tertinggi adalah solusi pada baris pertama dengan nilai *fitness* sebesar 7983. Solusi dengan nilai *fitness* tertinggi tersebut dianggap sebagai rekomendasi rute yang paling optimum untuk studi kasus permasalahan ini.

Dari uji coba ini dapat dilihat bahwa program yang dibuat dapat digunakan untuk studi kasus dengan *node* awal dan akhir yang berbeda dimana masih dapat menghasilkan solusi yang layak dan valid yaitu memenuhi batasan model OP.

6.6. Hasil dan Pembahasan Uji Coba 2: Mengubah Nilai Tmax

Seperti permasalahan pada studi kasus ini, pada uji coba ini dicari rute optimum dari *node* 1 (Jl. Kalimas Barat No.63, Pangkalan Petekan) menuju *node* 166 (Jl. Wonorejo No.18, Pangkalan Manukon) dengan jumlah populasi, probabilitas perkawinan silang, dan probabilitas mutasi yang sama dengan studi kasus sebelumnya, yaitu masing-masing 50 individu, 0.9, dan 0.1. Khususnya pada uji coba ini, batasan waktu (Tmax) dirubah dimana Tmax dibuat bervariasi mulai dari 5 sampai 150 menit dengan rentang waktu setiap 5 menit. Tujuan dari uji coba ini adalah memastikan bahwa program yang dibuat dapat digunakan untuk menghasilkan solusi yang layak dan memenuhi batasan model OP dengan ukuran Tmax yang bervariasi/beragam. Berikut ditampilkan solusi terbaik untuk setiap batasan waktu tempuh (Tmax) dalam Tabel 6.8.

Tabel 6.8 Solusi Terbaik untuk Setiap Tmax

Tmax	Skor	Waktu Tempuh	Solusi
5	0	0	Tidak ditemukan solusi yang layak
10	0	0	Tidak ditemukan solusi yang layak
15	0	0	Tidak ditemukan solusi yang layak
20	0	0	Tidak ditemukan solusi yang layak
25	0	0	Tidak ditemukan solusi yang layak
30	0	0	Tidak ditemukan solusi yang layak
35	0	0	Tidak ditemukan solusi yang layak
40	0	0	Tidak ditemukan solusi yang layak
45	0	0	Tidak ditemukan solusi yang layak
50	0	0	Tidak ditemukan solusi yang layak
55	0	0	Tidak ditemukan solusi yang layak
60	7118	58	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 79, 82, 155, 156, 157, 158, 159, 166]
65	7683	64	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 93, 95, 94, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 155, 156, 157, 158, 166]
70	7011	60	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 99, 100, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 82, 166]
75	7798	67	[1, 2, 3, 8, 9, 10, 11, 6, 4, 7, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 154, 85, 84, 83, 81, 79, 82, 155, 166]
80	8811	79	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127,

Tmax	Skor	Waktu Tempuh	Solusi
			126, 125, 124, 141, 142, 143, 167, 168, 169, 166]
85	8795	75	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 17, 16, 107, 109, 110, 111, 113, 114, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 129, 130, 131, 91, 90, 89, 88, 86, 85, 84, 83, 81, 79, 82, 155, 166]
90	9891	87	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127, 126, 125, 124, 141, 142, 143, 167, 168, 169, 170, 171, 172, 166]
95	10347	89	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 102, 101, 99, 98, 96, 92, 91, 129, 128, 132, 127, 126, 125, 124, 141, 142, 143, 167, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 86, 85, 166]
100	11126	92	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 140, 138, 139, 136, 135, 133, 132, 124, 169, 171, 170, 168, 167, 143, 142, 141, 126, 127, 128, 129, 130, 131, 91, 90, 89, 88, 87, 85, 84, 83, 81, 166]
105	14678	105	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 134, 135, 133, 132, 127, 126, 125, 124, 141, 142, 143, 167, 168, 169, 141, 142, 143, 167, 168, 169, 167, 126, 125, 124, 141, 142, 143, 167, 168, 166]
110	11691	97	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127, 126, 125, 124, 141, 142, 143, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 166]

Tmax	Skor	Waktu Tempuh	Solusi
115	12283	114	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127, 125, 124, 142, 143, 167, 168, 169, 170, 171, 172, 176, 175, 174, 173, 141, 126, 129, 130, 131, 96, 97, 99, 98, 91, 90, 166]
120	13535	117	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 105, 104, 103, 140, 137, 136, 134, 132, 127, 124, 143, 142, 167, 168, 141, 125, 126, 128, 129, 130, 131, 91, 92, 93, 94, 95, 30, 126, 124, 141, 142, 143, 168, 167, 144, 145, 146, 148, 149, 150, 151, 152, 153, 154, 86, 166]
125	17138	122	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127, 126, 125, 124, 142, 143, 167, 168, 169, 170, 141, 128, 127, 126, 125, 124, 142, 143, 167, 168, 169, 170, 142, 143, 167, 168, 169, 170, 143, 167, 141, 129, 128, 130, 91, 166]
130	13479	127	[1, 2, 3, 4, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 133, 132, 127, 142, 167, 169, 170, 172, 178, 177, 176, 175, 174, 173, 171, 168, 143, 126, 134, 125, 141, 123, 124, 143, 144, 145, 146, 147, 148, 149, 151, 152, 153, 154, 166]
135	14115	129	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 141, 124, 127, 132, 125, 128, 129, 130, 131, 91, 90, 89, 88, 87, 86, 154, 153, 152, 151, 150, 149, 147, 146, 132, 127, 126, 125, 142, 167, 143, 141, 124,

Tmax	Skor	Waktu Tempuh	Solusi
			128, 129, 130, 131, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 166]
140	18377	127	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 132, 127, 126, 125, 124, 141, 170, 172, 176, 177, 175, 174, 173, 171, 169, 168, 167, 143, 144, 124, 141, 142, 143, 167, 171, 172, 173, 175, 174, 170, 169, 168, 145, 147, 148, 150, 151, 152, 154, 86, 85, 84, 83, 166]
145	14452	140	[1, 2, 3, 12, 15, 108, 179, 180, 181, 182, 189, 190, 191, 16, 14, 6, 4, 7, 8, 9, 13, 18, 107, 106, 105, 104, 103, 140, 139, 138, 137, 136, 135, 134, 133, 125, 145, 146, 144, 167, 169, 168, 143, 142, 141, 124, 126, 127, 128, 129, 130, 131, 92, 93, 94, 95, 31, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 166]
150	16113	148	[1, 2, 3, 4, 7, 8, 9, 12, 13, 15, 18, 107, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 141, 142, 143, 144, 145, 150, 151, 152, 87, 88, 91, 131, 130, 128, 127, 126, 125, 167, 170, 171, 173, 174, 175, 176, 172, 169, 168, 147, 148, 149, 153, 154, 86, 85, 84, 83, 81, 79, 77, 76, 75, 58, 59, 60, 166]

Dapat dilihat pada Tabel 6.8 bahwa solusi terbaik pada tiap Tmax tidak ada yang melanggar batasan dari model OP (tidak termasuk untuk tMax 5 sampai 55 menit karena tidak memiliki solusi yang layak) dengan penjelasan sebagai berkut.

1. Batasan 1 terpenuhi. Rute memiliki *node* awal dan akhir yang tetap. Dapat dilihat dari solusi terbaik pada setiap Tmax yang dihasilkan memiliki *node* awal dan akhir yang

tetap sesuai dengan studi kasus permasalahan yaitu *node* 1 dan *node* 166.

2. Batasan 2 terpenuhi. Semua *node* harus saling terhubung dan setiap *node* hanya dapat dikunjungi maksimum satu kali. Dari solusi terbaik pada setiap Tmax yang dihasilkan, tidak ada solusi yang memiliki gen/*node* ganda.
3. Batasan 3 terpenuhi. Jumlah total waktu tempuh tidak boleh melebihi batasan total waktu yang dialokasikan (Tmax). Setiap solusi terbaik pada setiap tMax memiliki waktu tempuh yang lebih sedikit atau sama dengan Tmax.
4. Batasan 4 dan 5 terpenuhi. Tidak boleh ada *subtours*, lintasan dimulai dan berakhir pada *node* yang sama. Ini dibuktikan dari gen awal dan akhir dari setiap individu/solusi berbeda. Selain itu gen awal dan akhir hanya muncul satu kali pada setiap solusi terbaik.

Dari uji coba ini dapat dilihat bahwa semakin besar tMax maka semakin banyak *node* yang dapat dikunjungi sehingga nilai *fitness* juga semakin besar. Untuk tMax 5 sampai 55 menit tidak ditemukan solusi karena memang tidak ada rute/lintasan dari *node* 1 ke *node* 166 yang memiliki waktu tempuh kurang dari tMax tersebut.

6.7. Hasil dan Pembahasan Uji Coba 3: Mengubah Parameter GA

Pada uji coba ini akan dirubah parameter-parameter yang dapat mempengaruhi implementasi dari Genetic Algorithm (GA) meliputi probabilitas perkawinan silang, probabilitas mutasi, dan jumlah populasi. Perubahaan penggunaan parameter tentunya akan mempengaruhi solusi optimum yang dihasilkan dan dari penelitian ini dapat diketahui kombinasi-kombinasi dari parameter yang dapat menghasilkan solusi dengan nilai *fitness* tertinggi. Berdasarkan nilai fitness dari solusi optimum yang dihasilkan, parameter-parameter yang digunakan sebagai perbandingan hasil *Genetic Algorithm* akan digunakan untuk mengetahui beberapa poin penting sebagai berikut.

a. Mengetahui perbandingan pengaruh banyaknya jumlah individu dalam populasi pada nilai *fitness* yang dihasilkan.
b. Mengetahui perbandingan pengaruh probabilitas perkawinan silang pada nilai *fitness* yang dihasilkan.
c. Mengetahui perbandingan pengaruh probabilitas mutasi pada nilai *fitness* yang dihasilkan.

Selanjutnya dalam uji coba ini, dibuat beberapa skenario uji coba untuk mengetahui hasi *Genetic Algorithm* (GA) berdasarkan permasalahan yang ada pada studi kasus, yaitu pencarian rute optimum dari node 1 (Jl. Kalimas Barat No.63, Pangkalan Petekan) menuju node 166 (Jl. Wonorejo No.18, Pangkalan Manukon) dan digunakan jumlah generasi serta Tmax yang sama seperti studi kasus yaitu 300 dan 70 menit. Pada studi kasus ini, dilakukan beberapa skenario dengan melibatkan perbandingan dalam poin-poin sebagai berikut.

a. Perbandingan nilai *fitness* berdasarkan parameter jumlah populasi sebanyak 50, 100, dan 150 individu.
b. Perbandingan nilai *fitness* berdasarkan parameter perkawinan silang (*cross over*) yaitu 0.8 dan 0.9 sesuai dengan rekomendasi yang dijelaskan pada bagian 4.5.1 [29].
c. Perbandingan nilai *fitness* berdasarkan parameter mutasi yaitu 0.05 dan 0.1 sesuai dengan rekomendasi yang dijelaskan pada bagian 4.5.2 [29].

Berdasarkan poin-poin tersebut, maka dilakukan beberapa skenario dengan hasil sebagai berikut.

6.7.1. Skenario 1

Pada skenario ini dijalankan program *Genetic Algorithm* (GA) yang telah dibuat berdasarkan parameter probabilitas perkawinan silang (Prob CO) 0.9, probabilitas mutasi (Prob Mut) 0.1, dan jumlah populasi (Jum Pop) 50, 100, 150. Program akan dijalankan sebanyak sepuluh kali dimana diperoleh rata-rata solusi optimum ketika menggunakan parameter Prob CO

0.9, Prob Mut 0.1, dan Jum Pop 150 seperti pada Tabel 6.9. Dipergunakan rata-rata karena solusi optimum yang dihasilkan dapat berbeda setiap program dijalankan. Secara keseluruhan hasil dari uji coba 3 skenario 1 ada pada

Tabel 6.9 Rata-Rata Hasil Uji Coba 3 Skenario 1

Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
0.9	0.1	50	7484	66.2
0.9	0.1	100	7880.9	66.4
0.9	0.1	150	8045.1	67

6.7.2. Skenario 2

Pada skenario ini dijalankan program *Genetic Algorithm* (GA) yang telah dibuat berdasarkan parameter probabilitas perkawinan silang (Prob CO) 0.9, probabilitas mutasi (Prob Mut) 0.05, dan jumlah populasi (Jum Pop) 50, 100, 150. Program akan dijalankan sebanyak sepuluh kali dimana diperoleh rata-rata solusi optimum ketika menggunakan parameter Prob CO 0.9, Prob Mut 0.05, dan Jum Pop 150 seperti pada Tabel 6.10. Dipergunakan rata-rata karena solusi optimum yang dihasilkan dapat berbeda setiap program dijalankan.

Tabel 6.10 Rata-Rata Hasil Uji Coba 3 Skenario 2

Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
0.9	0.05	50	6963.5	64.7
0.9	0.05	100	7614.1	66.9
0.9	0.05	150	7991.7	67.6

6.7.3. Skenario 3

Pada skenario ini dijalankan program *Genetic Algorithm* (GA) yang telah dibuat berdasarkan parameter probabilitas perkawinan silang (Prob CO) 0.8, probabilitas mutasi (Prob Mut) 0.1, dan jumlah populasi (Jum Pop) 50, 100, 150. Program akan dijalankan sebanyak sepuluh kali dimana diperoleh rata-rata solusi optimum ketika menggunakan parameter Prob CO 0.8, Prob Mut 0.1, dan Jum Pop 150 seperti pada Tabel 6.11. Dipergunakan rata-rata karena solusi optimum yang dihasilkan dapat berbeda setiap program dijalankan.

Tabel 6.11 Rata-Rata Hasil Uji Coba 3 Skenario 3

Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
0.8	0.1	50	7350.8	62.5
0.8	0.1	100	7747.3	66.5
0.8	0.1	150	7912.2	66.9

6.7.4. Skenario 4

Pada skenario ini dijalankan program *Genetic Algorithm* (GA) yang telah dibuat berdasarkan parameter probabilitas perkawinan silang (Prob CO) 0.8, probabilitas mutasi (Prob Mut) 0.05, dan jumlah populasi (Jum Pop) 50, 100, 150. Program akan dijalankan sebanyak sepuluh kali dimana diperoleh rata-rata solusi optimum ketika menggunakan parameter Prob CO 0.8, Prob Mut 0.05, dan Jum Pop 150 seperti pada Tabel 6.12. Dipergunakan rata-rata karena solusi optimum yang dihasilkan dapat berbeda setiap program dijalankan.

Tabel 6.12 Rata-Rata Hasil Uji Coba 3 Skenario 4

Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
0.8	0.05	50	7049.5	64.8
0.8	0.05	100	7658.3	65.1
0.8	0.05	150	7894.4	66.6

6.7.5. Perbandingan Tiap Skenario

Berdasarkan empat skenario yang telah dijalankan tersebut, maka dipilih hasil terbaik dari masing-masing skenario dan ditampilkan pada Tabel 6.13.

Tabel 6.13 Hasil Uji Coba 3 Terbaik tiap Skenario

Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
0.9	0.1	150	8045.1	67
0.9	0.05	150	7991.7	67.6
0.8	0.1	150	7912.2	66.9
0.8	0.05	150	7894.4	66.6

Pada uji coba ini dapat disimpulkan dari Tabel 6.13 bahwa hasil terbaik untuk permasalahan pada studi kasus ini (pencarian solusi optimum) berdasarkan rata-rata nilai *fitness* dari empat skenario yang telah dijalankan di atas didapatkan menggunakan parameter probabilitas perkawinan silang sebesar 0.9, probabilitas mutasi sebesar 0.1, dan jumlah populasi sebanyak 150. Hasil tersebut merupakan rata-rata dari sepuluh percobaan dengan nilai *fitness* sebesar 8045.1. Dari masing-masing skenario tersebut juga didapatkan jumlah populasi yang terbaik adalah 150 individu.

Walaupun setiap program dijalankan dapat menghasilkan solusi dengan nilai *fitness* yang berbeda, ketika menggunakan parameter probabilitas perkawinan silang 0.9, probabilitas mutasi 0.1, dan jumlah populasi 150, maka hasil yang

didapatkan cenderung lebih optimum dibandingkan menggunakan nilai parameter lainnya yang digunakan pada uji coba ini.

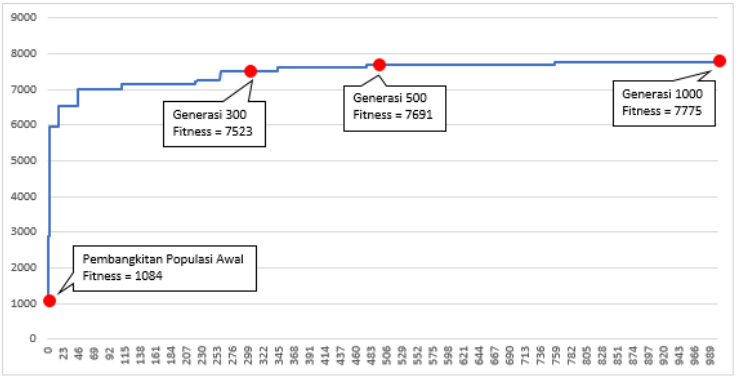
6.8. Hasil dan Pembahasan Uji Coba 4: Mengubah Jumlah Generasi

Selanjutnya dengan menggunakan studi kasus permasalahan yaitu mencari rute optimum dari node 1 menuju node 166, dilakukan uji coba dengan menggunakan jumlah generasi yang bervariasi yaitu 300, 500, dan 1000 iterasi. Tujuan dari uji coba ini adalah menentukan jumlah generasi yang menghasilkan solusi yang paling optimum (berdasarkan nilai *fitness*) dari ketiga jumlah generasi yang dibandingkan. Dalam studi kasus ini digunakan jumlah populasi, Tmax, probabilitas perkawinan silang, dan probabilitas mutasi yang sama dengan studi kasus sebelumnya, yaitu masing-masing 50 individu, 70 menit, 0.9, dan 0.1. Hasil dari perbandingan tersebut dapat dilihat pada Tabel 6.14.

Tabel 6.14 Hasil dari Perbandingan Jumlah Generasi

Generasi	Solusi Optimum	Waktu Tempuh	Fitness
0	[1, 6, 3, 78, 166]	69	1084
300	[1, 2, 3, 7, 8, 9, 10, 11, 6, 4, 12, 13, 15, 18, 107, 106, 105, 104, 103, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 86, 85, 84, 83, 81, 79, 78, 77, 166]	69	7523
500	[1, 2, 3, 7, 8, 9, 10, 11, 6, 4, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 86, 85, 84, 83, 81, 79, 78, 77, 166]	69	7691
1000	[1, 2, 3, 7, 8, 9, 10, 11, 6, 4, 12, 13, 15, 18, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 81, 79, 78, 77, 166]	69	7775

Generasi 0 merupakan hasil dari pembangkitan populasi awal. Hal ini menunjukkan bahwa terjadi perubahan nilai *fitness* yang signifikan dari proses pembangkitan populasi awal sampai dihasilkannya solusi optimum melalui proses seleksi, perkawinan silang, dan mutasi. Dari Tabel 6.14 juga dapat dilihat perbaikan nilai fitness pada setiap generasi sampai pada generasi ke 1000 masih ditemukan solusi yang lebih optimal. Perubahan nilai *fitness* pada tiap generasinya dapat dilihat pada Gambar 6.1.



Gambar 6.1 Perubahan Nilai Fitness tiap Generasi

Pada uji coba ini dapat dilihat bahwa perbaikan nilai *fitness* terus terjadi di setiap generasinya. Hal ini menunjukkan jumlah generasi menjadi salah satu faktor penting dalam menemukan solusi yang optimum. Selain itu, pada uji coba ini dapat dilihat bahwa sebenarnya perbaikan nilai *fitness* sudah berhenti pada generasi ke-758 dengan nilai *fitness* sebesar 7775. Nilai *fitness* tersebut tidak berubah sampai generasi ke-1000 (maksimum jumlah generasi yang ditentukan pada uji coba ini). Untuk studi kasus lain, jumlah generasi yang terbaik pun berbeda-beda untuk menghasilkan solusi yang optimum.

“Halaman ini sengaja dikosongkan”

BAB VII
KESIMPULAN DAN SARAN

Bab ini akan menjelaskan kesimpulan dari penelitian, beserta saran yang dapat bermanfaat untuk perbaikan di penelitian selanjutnya.

7.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Metode *Orienteering Problem* (OP) dapat digunakan untuk memodelkan penentuan rute perjalanan dengan angkot di Surabaya yang optimum.
2. Algoritma Dijkstra dapat digunakan untuk membantu pencarian waktu tempuh tercepat antar *node* sehingga membantu dalam pencarian solusi layak pada proses pembangkitan populasi awal. Hasil dari algoritma ini dapat dijadikan sebagai data set untuk penelitian selanjutnya.
3. *Genetic Algorithm* (GA) dapat digunakan untuk mencari rute optimum berdasarkan model OP yang telah ditentukan sebelumnya. Terbukti dalam penerapannya, GA mampu menghasilkan solusi akhir yang lebih baik dari solusi saat proses pembangkitan populasi awal. Genetic Algorithm dapat digunakan untuk mencari solusi yang layak untuk semua *node* tanpa melanggar batasan dari model OP.
4. Pemilihan jumlah generasi, jumlah populasi, probabilitas perkawinan silang, dan probabilitas mutasi memiliki peran penting dalam melakukan optimasi menggunakan *Genetic Algorithm* untuk pencarian solusi yang optimum.

7.2. Saran

Saran yang diberikan berdasarkan hasil penelitian yang dilakukan untuk penelitian selanjutnya antara lain:

1. Pada penelitian ini dibatasi untuk enam jalur mikrolet di Surabaya dengan kode trayek yaitu DP, I, Q, TV2, Z, Z1. Pada penelitian selanjutnya, dapat ditambahkan jalur mikrolet lain untuk memperluas cakupan optimasi angkot di Kota Surabaya.
2. Dalam mutasi menggunakan pencarian lokal pada penelitian ini digunakan operator *add* dan *omit*. Untuk penelitian selanjutnya dapat ditambahkan operator lainnya seperti *swap* dan *replace* untuk memungkinkan munculnya kembali gen yang tidak muncul pada proses pembangkitan populasi awal.
3. Representasi skor dari model Orienteering Problem pada studi kasus penelitian ini adalah jumlah angkot yang lewat di setiap *node* (tempat pemberhentian angkot). Representasi skor dapat disempurnakan dengan mempertimbangkan beberapa faktor lain seperti tingkat kemacetan di setiap *node*, jumlah pengunjung yang melewati setiap *node*, atau tingkat popularitas dari *node* atau tempat sekitar *node*.
4. Waktu tempuh yang digunakan pada penelitian ini bersifat statis (tetap) yang diambil melalui aplikasi *Google Maps*. Untuk penelitian selanjutnya dapat dicoba menggunakan waktu tempuh yang lebih dinamis, misalnya berdasarkan jam keberangkatan pagi, siang, sore, atau malam menggunakan teknik *machine learning* dari rekaman historis data *real-time* di lapangan.
5. Objek penelitian dari tugas akhir ini adalah angkot. Ke depannya diharapkan penelitian ini dapat diterapkan pada studi kasus lainnya, misalnya untuk transportasi umum bus.
6. Luaran dari penelitian ini hanya berupa rekomendasi rute yang menampilkan *node*/tempat mana saja yang dikunjungi dengan skor dan waktu tempuhnya. Ke depannya dapat dikembangkan lagi dengan menambahkan rekomendasi angkot dengan kode mana saja yang harus diambil untuk melalui rute tersebut.

DAFTAR PUSTAKA

[1] Liputan 6, “10 Kota dengan Lalu Lintas Terburuk di Dunia,” 23 March 2016. [Online]. Available: http://bisnis.liputan6.com/read/2465078/10-kota-dengan-lalu-lintas-terburuk-di-dunia.

[2] Kompas, “Ahok: Tiap Hari Ada 1.200 Kendaraan Baru di Jakarta,” 23 June 2016. [Online]. Available: http://megapolitan.kompas.com/read/2016/06/23/15142971/ahok.tiap.hari.ada.1.200.kendaraan.baru.di.jakarta.

[3] Kompas, “Enam Hal Ini Sebabkan Kemacetan Masih "Awet" di Jakarta,” 13 February 2016. [Online]. Available: http://megapolitan.kompas.com/read/2016/02/13/17182371/Enam.Hal.Ini.Sebabkan.Kemacetan.Masih.Awet.di.Jakarta.

[4] J. Sukma, “Tidak Ada Jalan Macet Lagi di Surabaya,” 19 July 2016. [Online]. Available: http://www.cowasjp.com/read/1496/20160719/163047/tidak-ada-jalan-macet-lagi-di-surabaya/.

[5] Jakarta Raya, “Kemacetan Jakarta: Nilai Kerugian Masyarakat Rp150 Triliun Per Tahun,” 25 April 2016. [Online]. Available: http://jakarta.bisnis.com/read/20160425/77/541368/kemacetan-jakarta-nilai-kerugian-masyarakat-rp150-triliun-per-tahun.

[6] detikFinance, “Ini Dampak Kemacetan Terhadap Perekonomian Jakarta,” 27 June 2016. [Online]. Available: http://finance.detik.com/ekonomi-bisnis/3243447/ini-dampak-kemacetan-terhadap-perekonomian-jakarta.

[7] E. Y. Kristanti, “Surabaya Macet, Rp1 Triliun Menguap,” viva.co.id, 23 September 2010. [Online]. Available: http://nasional.news.viva.co.id/news/read/179159-surabaya-macet-rp1-triliun-per-hari-menguap.

[8] Tribunnews.com, “Ini Cara Tri Rismaharini Antisipasi Kemacetan di Surabaya,” 29 November 2013. [Online]. Available:

http://www.tribunnews.com/regional/2013/11/29/ini-cara-tri-rismaharini-antisipasi-kemacetan-di-surabaya.

[9] Dinas Perhubungan Kota Surabaya, "Dibalik Berdirinya SITS," [Online]. Available: http://sits.dishub.surabaya.go.id/ver2/tentang-sits.

[10] D. Kurniasari, "Siapkah Indonesia Implementasikan ITS?," Kompasiana, 23 June 2015. [Online]. Available: http://www.kompasiana.com/dian_kur/siapkah-indonesia-implementasikan-its_54f7c3d3a33311181d8b4980.

[11] A. A. Rozak, "Perihal Angkot dan Tips Aman Menaikinya," [Online]. Available: http://www.jurnalrozak.web.id/2016/12/perihal-angkot-dan-tips-aman-menaikinya.html.

[12] P. Vansteenwegen, W. Souffriau dan D. V. Oudheusden, "The Orienteering Problem: A Survey," *European Journal of Operational Research,* pp. 1-10, 2011.

[13] T. Tsiligirides, "Heuristic Methods Applied to Orienteering," *Journal of the Operational Research Society,* pp. 797-809, 1984.

[14] W. e. a. Souffriau, "A Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides," *Applied Artificial Intelligence,* vol. 22, pp. 964-985, 2008.

[15] X. Wang, B. L. Golden dan E. A. Wasil, "Using a Genetic Algorithm to Solve the Generalized Orienteering Problem," *The Vehicle Routing Problem: Latest Advances and New Challenges,* vol. 43, pp. 263-274, 2008.

[16] M. Schilde, K. Doerner, R. Hartl dan G. Kiechle, "Metaheuristics for the Bi-Objective Orienteering Problem," *Swarm Intelligence,* vol. 3, pp. 179-201, 2009.

[17] M. F. Tasgetiren, "A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem," *Journal of Economic and Social Research,* pp. 1-26, 2001.

[18] M. F. Tasgetiren dan A. E. Smith, "A Genetic Algorithm for the Orienteering Problem," *Evolutionary Computation,* vol. 2, pp. 910-915, 2000.

[19] J. Ferreira, A. Quintas, J. A. Oliveira, G. A. Pereira dan L. & Dias, "Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach," *Soft Computing in Industrial Applications*, pp. 365-375, 2014.

[20] J. Karbowska-Chilinska, J. Koszelew, K. Ostrowski dan P. Zabielski, "Genetic Algorithm Solving Orienteering Problem in Large Networks," *KES*, pp. 28-38, 2012.

[21] M. Aftabuzzaman, "Measuring Traffic Congestion - A Critical Review," dalam *Australasian Transport Research Forum (ATRF), 30TH*, 2007.

[22] E. K. P. Chong dan S. H. Zak, An Introduction to Optimization 2nd Edition, Canada: John Wiley & Sons, Inc., 2001.

[23] E. Martiana, "Data Preprocessing," [Online]. Available: http://entin.lecturer.pens.ac.id/Data%20Mining/Minggu%202%20Data%20Preprocessing.pdf.

[24] B. W. Taylor, Introduction to Management Science Ed. 11, Pearson, 2013.

[25] T. Abiy, H. Pang dan J. Khim, "Dijkstra's Shortest Path Algorithm," [Online]. Available: https://brilliant.org/wiki/dijkstras-short-path-finder/.

[26] "Bab 7 Algoritma Genetika," [Online]. Available: http://entin.lecturer.pens.ac.id/Kecerdasan%20Buatan/Buku/Bab%207%20Algoritma%20Genetika.pdf.

[27] F. Saptono dan T. Hidayat, "Perancangan Algoritma Genetika untuk Menentukan Jalur Terpendek," 2007.

[28] Dinas Perhubungan Kota Surabaya, "Jumlah dan Rute (Trayek) Angkutan Umum," 2015. [Online]. Available: http://dishub.surabaya.go.id/index.php/post/id/1840.

[29] M. Obitko, "Parameters of GA," [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/recommendations.php. [Diakses 2 June 2017].

“Halaman ini sengaja dikosongkan”

BIODATA PENULIS

Penulis bernama lengkap I Wayan Angga Kusuma Yoga, dilahirkan di Denpasar pada tanggal 14 Januari 1995. Penulis merupakan anak pertama dari tiga bersaudara. Pada tahun 2007, penulis lulus dari Sekolah Dasar Negeri 13 Sanur. Pada tahun 2010, penulis lulus dari SMP Negeri 9 Denpasar. Pada tahun 2013, penulis lulus dari SMA Negeri 1 Denpasar. Dan masih pada tahun yang sama, penulis diterima di Jurusan Sistem Informasi Institut Teknologi Sepuuh Nopember (ITS) melalui jalur SNMPTN.

Sebagai mahasiswa, penulis aktif dalam kegiatan organisasi. Tercatat penulis aktif berkontribusi melalui keanggotaan di Tim Pembina Kerohanian Hindu sebagai Staf Departemen Hubungan Masyarakat pada tahun kepengurusan 2014-2015 dan sebagai Staf Ahli Departemen Hubungan Internal pada tahun kepengurusan 2015-2016. Penulis juga aktif dalam kepanitiaan acara seperti Seminar Nasional Sistem Informasi Indonesia (SESINDO) pada tahun 2014 dan Information Systems Expo (ISE) pada tahun 2015. Penulis pernah melakukan kerja praktik di PT Pertamina sebagai Systems Analyst pada bulan Juni sampai Agustus tahun 2016.

Di Jurusan Sistem Informasi, penulis mengambil laboratorium Rekayasa Data dan Inteligensi Bisnis dengan topik tugas akhir adalah optimasi menggunakan Orienteering Problem dan Genetic Algorithm khususnya pada bidang transportasi. Untuk keperluan penelitian, penulis dapat dihubungi melalui e-mail: aka.yogaa@gmail.com.

“Halaman ini sengaja dikosongkan”

LAMPIRAN A

Lampiran ini merupakan deskripsi berupa alamat dan keterangan tambahan setiap *node* yang ada pada *network model*. Keterangan tambahan merupakan tempat-tempat (seperti pusat perbelanjaan, kantor, objek wisata, dan lainnya) yang posisinya dekat dengan *node* (tempat pemberhentian angkot). Keterangan tambahan digunakan untuk mempermudah mengetahui letak *node* tersebut.

Tabel A.1 Deskripsi Node

Node	Deskripsi	Keterangan Tambahan
1	Jl. Kalimas Barat No.63	Pangkalan Petekan, dekat AML (Asia Mandiri Lines)
2	Jl. Kalimas Barat No.45	Dekat PT Rajawali Nusindo
3	Jl. Kalimas Barat No.3	Pangkalan JMP, dekat pertigaan Jl. Kalimas Barat dan Jl. Kasuari
4	Jl. Kasuari No.12	Dekat Jembatan Merah Plaza (JMP)
5	Jl. Garuda No.2	Dekat Jembatan Merah Plaza (JMP)
6	Jl. Rajawali 41-43	Pertigaan Krembangan Timur-Rajawali
7	Jl. Jembatan Merah No.3	Dekat Banyak Syariah Mandiri
8	Jl. Veteran No.6-8	Dekat BCA KCU Veteran
9	Jl. Kebon Rojo	Dekat Halte, Masjid Raudhatul Musyaawarah
10	Jl. Krembangan Barat No.24	Dekat BCA KCP Krembangan
11	Jl. Krembangan Barat 70-112	Dekat Ajenrem 084 Dam V BRW
12	Jl. Indrapura No.17	Sebelum Parang Kusumo, dekat halte

A-2

Node	Deskripsi	Keterangan Tambahan
13	Jl. Indrapura No.43	Dekat Bank Mandiri KCP Surabaya Indrapura
14	Jl. Rajawali No.53	Dekat Bank Rakyat Indonesia
15	Jl. Rajawali No.90/113	Dekat U Turn Rajawali
16	Jl. Perak Timur No.44/49	Dekat U Turn Perak Timur, halte
17	Jl. Ikan Belanak	Dekat Pertigaan Ikan Belanak-Bandeng
18	Jl. Gresik Gadukan Timur No.20	Dekat Puskopal Armatim Surabaya
19	Jl. Pahlawan	Dekat Bappeda Provinsi Jawa Timur
20	Jl. Bubutan 123-125	Dekat SMP Katolik Stella Maris, halte
21	Jl. Tembaan No.97	Dekat Pasar Turi Poultry Shop
22	Jl. Semarang 118-119	Dekat BRI, halte
23	Jl. Semarang No.53	Dekat PT Karya Energi Indonesia, Indomaret
24	Jl. Raya Arjuno No.7	Dekat Perempatan Semarang, Kranggan, Raya Arjuno
25	Jl. Tidar 30-32	Dekat RM Ayam Goreng Asli Pemuda, Joyo Optikal
26	Jl. Kedung Doro No.73	Dekat BNI, Depot Slamet, Shinhan Bank
27	Jl. Kedung Doro No.161	Dekat U Turn, Bank Mestika, PT Angkasa Mulya
28	Jl. Kedung Doro No.265	Dekat Martabak Holland

Node	Deskripsi	Keterangan Tambahan
29	Jl. Pasar Kembang No.95	Dekat Yayasan Mardi Sunu, CIMB Niaga Cabang, Hotel Hasanah Jaya
30	Jl. Raya Diponegoro No.225	Dekat Pasar Burung Kupang, halte
31	Jl. Raya Diponegoro No.174	Dekat U Turn, dekat Baruna, Western Union
32	Jl. Raya Diponegoro 110-112	Dekat Bank Mandiri KCP Diponegoro, BCA KCU Diponegoro, halte
33	Jl. Dr. Soetomo No.73/100	Dekat Artotel Surabaya
34	Jl. Dr. Soetomo No.58/39	Dekat PropNex Indonesia, Bank Permata
35	Jl. Polisi Istimewa No.18/7	Dekat Chubb, Pipe and Barrel, halte
36	Jl. Raya Dinoyo No.63	Dekat Telkom Indonesia, Suara Muslim, halte
37	Jl. Raya Dinoyo 125-127	Dekat Kantor Pelayanan Pajak Pratama, Pom Bensin Dinoyo
38	Jl. Darmokali No.12	Dekat U Turn
39	Jl. Ratna No.9	Dekat Central Point Mall, Carrefour Kalimas
40	Jl. Bung Tomo	Dekat Marvel City Mall
41	Jl. Bung Tomo No.8/39	Dekat STMJ Bu Nunuk
42	Jl. Ngagel Jaya Selatan No.20/39	Dekat Bank Bumi Arta Cab. Ngagel, halte
43	Jl. Ngagel Jaya Selatan No.103	Dekat Ngagel Jaya Kimia, CFC, halte

A-4

Node	Deskripsi	Keterangan Tambahan
44	Jl. Ngagel Jaya Selatan No.145	Dekat PGN Solution, MORIN Bakery Ngagel
45	Jl. Bratang Binangun No.41	Dekat BNI Syariah Cabang Ngagel, Bank Mandiri KCP Bratang, halte
46	Jl. Bratang Jaya Gg. Bengkok I No.5	Terminal Bratang (dekat Sangkar Burung Jaya, halte)
47	Jl. Raya Manyar No.51	Dekat halte, Manyar Auto Service, Sapi Milk
48	Jl. Raya Manyar No.79	Dekat halte, Pulung, Depot Bubur Ayam Jakarta Manyar
49	Jl. Kembang Kuning No.75	Dekat Bujoko, Chizkek Lumer Surabaya
50	Jl. Kembang Kuning	Dekat Bundaran Pakis Sidokumpul, Makam Kembang Kuning
51	Jl. Pakis Sidokumpul 8-10	Dekat Warung Pojok, Warkop 69
52	Jl. Dukuh Kupang Tim. X 59-67	Dekat Warkop Koncoku, Apotek Pelita Jaya, JNE T Dukuh Kupang
53	Jl. Raya Dukuh Kupang No.83A	Pangkalan Kupang, dekat Kecamatan Sawah
54	Jl. Dukuh Kupang XXV No.63C	Dekat Puskesmas Dukuh Kupang Surabaya
55	Jl. Dukuh Kupang Timur No.303	Dekat Pasar Dukuh Kupang
56	Jl. Raya Dukuh Kupang Barat No.41	Dekat Bank Danamon

Node	Deskripsi	Keterangan Tambahan
57	Jl. Raya Dukuh Kupang Barat No.155	Dekat Ikan Bakar Cianjur
58	Jl. Mayjen Sungkono No.190	Dekat The Sages Institute International, JNE
59	Jl. Mayjen Sungkono No.7	Dekat Pengadilan Tinggi Agama Surabaya
60	Jl. Mayjen Sungkono No.91A	Dekat halte, BCA, Bank Permata Darmo Park
61	Jl. Mayjen Sungkono No.45	Dekat Gedung Juang 45 Jawa Timur, halte
62	Jl. Mayjen Sungkono No.68	Dekat Garuda Sport, Bank Internasional Indonesia
63	Jl. Adityawarman No.110	Dekat Dinas Kebudayaan dan Pariwisata, halte
64	Jl. Indragiri No.26	Dekat Ikan Bakar Cianjur, Cocari
65	Jl. Indragiri No.2A	Dekat Warung Kuline Surabaya, Warkop 777
66	Jl. Adityawarman No.41	Dekat Warung Kopi Adityawarman, halte
67	Jl. Hayam Wuruk No.8	Dekat The Cushy Kitchen, Taxi Services Bluebird
68	Jl. Hayam Wuruk No.24	Dekat Warkop Braga
69	Jl. Brawijaya No.1A	Dekat Triple "A" Reload, Toko Jago
70	Jl. Gunung Sari No.22	Warung Kopi Cak Kacong, Dinas Perhubungan Kota Surabaya
71	Jl. Gunung Sari No.1	Terminal Joyoboyo, dekat halte, Depot Laksana Jaya

A-6

Node	Deskripsi	Keterangan Tambahan
72	Jl. Raya Darmo 147-149	Dekat Kebun Binatang Surabaya
73	Jl. Raya Diponegoro No.25	Dekat Bank Danamon Cabang Syariah, BTN Syariah, BRI Syariah Darmo
74	Jl. Ciliwung 46-48	Dekat Bank OCBC NISP Syariah, Gaya Sepeda Indonesia, halte
75	Jl. Mayjen HR. Moh. No.1	Dekat Bundaran Satelit, PT Trisakti Makmur Persada
76	Jl. Mayjen HR. Moh. No.41a	Dekat Bank Syariah Bukopin, Warung Tekko, Ming Garden
77	Jl. Mayjen HR. Moh. No.75	Dekat Auto2000 Hr Muhammad, Artomoro Sea Food
78	Jl. Mayjen HR. Moh. No.123	Ruko Golden Palace, dekat Rodalink HR Muhammad
79	Jl. Mayjen HR. Moh. No.368	Dekat U Turn, Bank Artha Graha
80	Jl. Mayjen Yono Suwoyo No.10 CC	Dekat U Turn, Hartono
81	Jl. Mayjen Yono Suwoyo No.46	Dekat Warung Tekos, Sentra Digital
82	Jl. Raya Darmo Permai Selatan 17-19	Dekat halte, Fu Man Lou
83	Jl. Raya Darmo Permai II No.14	Dekat BKB Pty, Impressions
84	Jl. Raya Darmo Permai III Blok A No.12	Dekat Tahu Tek Pak Jayen, GSJA Maranatha
85	Jl. Darmo Baru Barat XII No.63	Dekat Radio Cakra Awigra, Mulia Pigora

Node	Deskripsi	Keterangan Tambahan
86	Jl. Pattimura No.20	Dekat STO Telkom Tandes, Depot Rhenata, Esa Farma Apotik
87	Jl. Kupang Jaya No.112B	Dekat Galeri Smartfren, Pink Bakery, 88 Depot
88	Jl. Kupang Jaya No.25	Dekat Fuma, Kantor Pos Surabaya Selatan
89	Jl. Raya Simo Gunung No.6 B	Dekat halte, Bank Tabungan Negara
90	Jl. Raya Simo Gunung 66-68	Dekat Indomaret, Restu Bunda
91	Jl. Banyu Urip No.228	Dekat U Turn, Bank Mandiri Banyu Urip, Warung Ketan
92	Jl. Banyu Urip No.320	Dekat Ampera Rumah Makan, BPR Intan Nasional
93	Jl. Banyu Urip No.230A	Dekat Depot 99, Depot Nasi Goreng 149
94	Jl. Banyu Urip Kidul II Molin 3 7-19	Dekat Rolas Tea, Boneka Kue
95	Jl. Banyu Urip Kidul II Molin 3 No.34	Dekat halte, Mebel Surya
96	Jl. Simo Kwagean No.41	Dekat Bakso Buntut Sapi, halte
97	Jl.Petemon II No.148	Dekat Warung Lim Jaya
98	Jl. Pacuan Kuda No.42	Dekat Alfamart, Toko Sumber Jaya
99	Jl. Tidar No.314	Dekat Warung Barokah, Ajiib Kebab n Burger
100	Jl. Kalibutuh No.26	Dekat PT Spindo Tbk, Wika Karya Abadi
101	Jl. Demak No.97	Dekat Alfamart, Tiga Putra

Node	Deskripsi	Keterangan Tambahan
102	Jl. Demak No.171	Dekat Panin Bank, Jupee Dupee Bread
103	Jl. Demak 233-235	Dekat halte, Warung Regeng, Sahabat Java
104	Jl. Demak 301-303	Dekat Kantor Kelurahan Jepara, Bank Rakyat Indonesia, Kedai Tretan
105	Jl. Demak No.377	Dekat halte, Bengkel Las "Nandas Laskar"
106	Jl. Demak No.449	Dekat Hotel Antariksa, Dua Putri Jaya, Kikil Sapi Demak
107	Jl. Gresik Gadukan Timur No.47	Dekat Masjid Al-Muttaqin, UD Berkah Sihatsu
108	Jl. Tanjung Sadari No.150	Dekat B.Net, Warung Fajar
109	Jl. Gresik Gadukan Timur No.1	Dekat Masjid Taqwirayul, Warung Pojok, Herbastore NASA Surabaya
110	Jl. Gresik Gadukan Timur No.270	Dekat Toko Mekarsari, Warung GIRAS KOPLER
111	Jl. Kalianak Timur No.352	Dekat halte, Warung Endik
112	Jalan Kalianak Barat	Dekat PT Sinar Sosro, PT Indra Jaya Swastika
113	Jalan Kalianak Barat No.57	Dekat PT Sumiati, Warung B Ida
114	Jl. Greges Timur No.63	Dekat Warung Panadia, Warung Samudra
115	Jl. Greges Timur	Dekat Warung Barokah, Warung Nasi Berkat
116	Jl. Greges Tim. No.17	Dekat Masjid Al Anshor, Depot Ayam, Gule Krengsengan

Node	Deskripsi	Keterangan Tambahan
117	Jl. Gerges Bar. No.6	Dekat Kantor Kelurahan Greges, Perum. Citra Graha Mandiri, Greges Collection
118	Jl. Margomulyo No.66	Dekat PT Energi Nusantara Prima, UD Sinar Jaya
119	Jl. Margomulyo Blok A No.40	Dekat UD Makmur Jaya, PT Hacaca Setio Abadi, CV Trisari Kumpul
120	Jl. Margomulyo No.39	Dekat Bank OCBC NISP Syariah, PT Tungya Perkasa Freight Forwarding
121	Jl. Margomulyo No.28	Dekat PT Jasa Ekspedisi Hasil Lasem Indah, PT Gold Coin Indonesia
122	Jl. Margomulyo No.16	Dekat PT Harma Presis Meka Indonesia, PT Sinarindo Megantara
123	Jl. Margomulyo No.4	Dekat PT Subaindo Cahaya Polintroco, Kedai Kopi Aurora
124	Jl. Raya Balongsari No.18	Dekat Warkop 87, Warung Mulyo
125	Jl. Tanjungsari 17-18	Dekat Abadi Motor, Mega Jaya
126	Jl. Tanjungsari 23-24	Dekat halte, Toko Cahaya Nusantara, Zulvi Cell
127	Jl. Tanjungsari No.82	Dekat Kelurahan Tanjungsari, Bakso Cak Adi, Warung Kopi Cak Emon
128	Jl. Sukomanunggal No.184	Dekat Soto Ayam Raja Rasa Lamongan, Bakso Arjuno
129	Jl. Sukomanunggal No.82	Dekat Bank Permata Capem Sukomanunggal, Warung Sederhana

Node	Deskripsi	Keterangan Tambahan
130	Jl. Sukomanunggal No.74	Dekat UD HA Rachman Toko Bangunan, halte
131	Jl. Simo Pomahan III No.47	Dekat Green Nitrogen Simo Pomahan SPBU, Warung Ijo
132	Jl. Tanjungsari No.5	Dekat PT Tanjungsari Megah, UD Sumber Rejeki, CV Mekar Buana
133	Jl. Tanjungsari No.19	Dekat PT Ardiles Ciptawijaya, UD Sumber Jaya Plastik, halte
134	Jl. Tanjungsari 38-40	Dekat PT Ricky Jaya Sakti, Warkop Ibu Ika
135	Jl. Tambak Mayor Sel. 31-33	Dekat Jawa Indah, TK Al Hidayah Tanjung Sari, Sinar Mas
136	Jl. Tambak Mayor No.4A	Dekat halte, Bengkel Honda Tunggal Dewi Motor
137	Jalan Tambak Mayor Utara No.180,	Dekat Pasar Buah, PT SCG Readymix Indonesia
138	Jl. Dupak Rukun No.97	Dekat PT Thomas Pratama Agung Diesel, Depot 88 Alina, Katebet Warung Pojok
139	Jl. Dupak Rukun No.79	Dekat Surya Inti, Attala Surabaya
140	Jl. Dupak Rukun No.24	Dekat halte, Toko Sabar Subur, Lik kopi
141	Jl. Raya Bibis No.12	Dekat Warung Mak Pin, Indomaret, Planet Ban
142	Jl. Raya Manukan Kulon No.68	Dekat Bank Danamon, BCA, Unggul Makmur

Node	Deskripsi	Keterangan Tambahan
143	Jl. Raya Manukan Kulon No.80	Dekat I. G. A. R. S, Solusi Rumah Holcim Surabaya Barat
144	Jl. Kyai Amir No.25	Dekat Toko Abadi, Alex Sandy, Mie Ayam Barokah
145	Jl. Manukan Tama No.75	Dekat Bank Tabungan Pensiunan Nasional, Depot Top Rasa
146	Jl. Manukan Dalam No.14	Dekat Karunia Sport, Roti Mandala
147	Jl. Bumi Indah No.51	Dekat halte, UD Putra Bali, AHASS Din Jaya Motor
148	Jl. Balongsari Tama Tengah No.12B,	Dekat Maxima Card, Panti Asuhan Baitul Yatim, Toko Meubel Novi Shinta Jaya
149	Jl. Raya Darmo Indah Sel. Blok LL No.44	Dekat Mitra Lapis Kukus Surabaya, ASM Air Sehat Mineral
150	Jl. Darmo Indah Tim. Blok G No.52	Dekat halte, KSP Putra Mandiri, Luxsera
151	Jl. Raya Satelit Utara No.17	Dekat Super Indo Satelit Utara
152	Jl. Raya Satelite Indah No.40	Dekat Betania Depot, CV. Surabaya Genset Engineering, Tahu Campur & Tahu Telor Surabaya
153	Jl. Raya Satelite Indah Blok FF No.8	Dekat Bank BCA KCP Darmo Indah Timur, Tahu Campur Satelit Indah, Holland Bakery
154	Jl. Raya Sukomanunggal Jaya No.86	Dekat Alfamidi, CV Duta Utama, Soto ayam Lamongan
155	Jl. Simpang Darmo Permai Selatan 28-40	Dekat Apotik Indah Farma 2, Depot Mamiku

Node	Deskripsi	Keterangan Tambahan
156	Jl. Raya Pradah Indah No.42	Dekat halte, Nasi & Mie Goreng Anglo Khas Kediri, Chepa & Chepi
157	Jl. Raya Pradah Indah 100-102	Dekat Warkop Ashokarso, Warung Pak Yanto
158	Jl. Raya Lontar No.16	Dekat halte, Warkop Rania
159	Jl. Raya Lontar 119-121	Dekat halte, Warkop Jula - Juli, Warkop Giras
160	Jl. Raya Lontar No.229	Dekat Makam Islam, PT Centro Media Indonesia
161	Jl. Raya Lontar No.51	Dekat Indomaret, Liek Caffe
162	Jl. Raya Sambi kerep No.20	Dekat halte, Burjo 234, Toko Juariah
163	Jl. Sambikerep Gg. III No.1	Dekat Warung Ande Masakan Padang, Toko Mega Hardware
164	Jl. Raya Jelidro II No.47	Dekat Warung Bang Tojib
165	Jl. Raya Jelidro II 21-27	Dekat halte, Pangsit Mie Ayam Ronggolawe, Panasonic Mandiri Teknik
166	Jl. Wonorejo No.18	Pangkalan Manukan, dekat halte, Toko Rwb Sticker
167	Jl. Raya Banjarsugihan No.15	Terminal Manukon Kulon, dekat halte, Limun Factory, Bandoeng Helm II
168	Jl. Raya Banjarsugihan No.87	Dekat Ayu Bakery, Masjid At Taubah, Toko Kaca Langgeng Jaya
169	Jl. Raya Kandangan No.21	Dekat Western Union, Lontong Balap Sate Kerang Cak Mat

Node	Deskripsi	Keterangan Tambahan
170	Jl. Klakah Rejo Surabaya No.1	Dekat Depot Rejo Pak Jo, Toko Sido Muncul, Jakarta Sport Kupang
171	Jl. Raya Sememi No.38	Dekat halte, Depot Sedap Malam, Warung Kopi Nikmat
172	Jl. Raya Sememi No.9a	Dekat halte, Polsek Benowo, UD Anugerah Jaya, Rumah Makan Rajo Bungsu
173	Jl. Raya Sememi No.8	Dekat SMK Wachid Hasyim, Koperasi Podo Tresno Jawa Timur
174	Jl. Raya Sememi No.53	Dekat Kantor Kelurahan Babat Jerawat, PT Kasogi International Tbk
175	Jl. Raya Babat Jerawat No.44	Dekat PT Reska Multi Usaha Area Surabaya, Mini Market Lima-Lima
176	Jl. Raya Pakal Surabaya 148-150	Dekat halte, Citra Abadi Sentosa
177	Jl. Pakal No.106	Dekat SPBU, Warung Kopi Bung Tomo
178	Jl. Benowo No.37 A	Pangkalan Benowo, dekat M2M Benowo, Benowo Farma Apotek
179	Jl. Tanjung Sadari No.109	Dekat Graha Alken, Warung Bu Wiwin
180	Jl. Tanjung Sadari No.78B	Dekat Gereja Keristen Jawi Wetan, Rumah Sakit TNI AL Dr. Oepomo
181	Jl. Tanjung Sadari No.3	Dekat Boby Corner Coffe, PT Orela Bahari, Toko Muncul
182	Jl. Tanjung Priuk No.11 S	Dekat PT Pelayaran Fivica Indonesia, Graha Barunawati

Node	Deskripsi	Keterangan Tambahan
183	Jl. Laksda Moh. Nazir No.30	Dekat Depo JAPFA, Burda Bread And Case, PT Hersindo Anugerah Multitrans
184	Jl. Laksda Moh. Nazir No.56	Dekat RSAL Tanjung Perak
185	Jl. Prapat Kurung Utara No.58	Dekat PT Pelindo Marine Service, Kapal Wisata Artama Harbour Cruise
186	Jl. Prapat Kurung Utara No.12	Dekat PT Sumber Arta, PT Chandra Karya (Adhi Karya Group), PT Jetty Benoa Kade Pertamina Surabaya
187	Jl. Prapat Kurung Utara No.4	Dekat BRI Cabang Surabaya Tanjung Perak
188	Jl. Kalimas Baru No.5	Pangkalan Ujung Baru, dekat CV Selaras Makmur
189	Jl. Perak Bar. No.199	Dekat PT Bank Bukopin KCP Perak Barat, PT Andalan Pacific Samudra
190	Jl. Perak Timur No.168	Dekat Taman Barunawati
191	Jl. Perak Timur No.92	Dekat Suzuki United Motors Centre Perak Timur, Bank OCBC NISP Surabaya - Perak

LAMPIRAN B

Lampiran ini merupakan keterangan dari setiap variabel keputusan yang digunakan untuk pada pemodelan *Orienteering Problem* (OP).

Tabel B.1 Variabel Keputusan

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

B-2

Variabel	Keterangan
x15.18	Kunjungan dari node 15 ke node 18
x16.18	Kunjungan dari node 16 ke node 18
x8.19	Kunjungan dari node 8 ke node 19
x20.9	Kunjungan dari node 20 ke node 9
x19.21	Kunjungan dari node 19 ke node 21
x21.20	Kunjungan dari node 21 ke node 20
x21.22	Kunjungan dari node 21 ke node 22
x22.21	Kunjungan dari node 22 ke node 21
x22.23	Kunjungan dari node 22 ke node 23
x23.22	Kunjungan dari node 23 ke node 22
x23.24	Kunjungan dari node 23 ke node 24
x24.23	Kunjungan dari node 24 ke node 23
x24.25	Kunjungan dari node 24 ke node 25
x25.24	Kunjungan dari node 25 ke node 24
x25.26	Kunjungan dari node 25 ke node 26
x26.25	Kunjungan dari node 26 ke node 25
x26.27	Kunjungan dari node 26 ke node 27
x27.26	Kunjungan dari node 27 ke node 26
x27.28	Kunjungan dari node 27 ke node 28
x28.27	Kunjungan dari node 28 ke node 27
x28.29	Kunjungan dari node 28 ke node 29
x29.28	Kunjungan dari node 29 ke node 28
x29.30	Kunjungan dari node 29 ke node 30
x30.29	Kunjungan dari node 30 ke node 29
x30.31	Kunjungan dari node 30 ke node 31
x31.30	Kunjungan dari node 31 ke node 30
x31.32	Kunjungan dari node 31 ke node 32
x32.31	Kunjungan dari node 32 ke node 31
x32.33	Kunjungan dari node 32 ke node 33

Variabel	Keterangan
x33.32	Kunjungan dari node 33 ke node 32
x33.34	Kunjungan dari node 33 ke node 34
x34.33	Kunjungan dari node 34 ke node 33
x34.35	Kunjungan dari node 34 ke node 35
x35.34	Kunjungan dari node 35 ke node 34
x35.36	Kunjungan dari node 35 ke node 36
x36.35	Kunjungan dari node 36 ke node 35
x36.37	Kunjungan dari node 36 ke node 37
x37.36	Kunjungan dari node 37 ke node 36
x38.37	Kunjungan dari node 38 ke node 37
x39.38	Kunjungan dari node 39 ke node 38
x37.40	Kunjungan dari node 37 ke node 40
x40.41	Kunjungan dari node 40 ke node 41
x41.39	Kunjungan dari node 41 ke node 39
x41.42	Kunjungan dari node 41 ke node 42
x42.41	Kunjungan dari node 42 ke node 41
x42.43	Kunjungan dari node 42 ke node 43
x43.42	Kunjungan dari node 43 ke node 42
x43.44	Kunjungan dari node 43 ke node 44
x44.43	Kunjungan dari node 44 ke node 43
x45.44	Kunjungan dari node 45 ke node 44
x46.45	Kunjungan dari node 46 ke node 45
x44.47	Kunjungan dari node 44 ke node 47
x47.48	Kunjungan dari node 47 ke node 48
x48.46	Kunjungan dari node 48 ke node 46
x31.49	Kunjungan dari node 31 ke node 49
x49.50	Kunjungan dari node 49 ke node 50
x50.51	Kunjungan dari node 50 ke node 51
x51.52	Kunjungan dari node 51 ke node 52

Variabel	Keterangan
x52.53	Kunjungan dari node 52 ke node 53
x53.54	Kunjungan dari node 53 ke node 54
x54.55	Kunjungan dari node 54 ke node 55
x55.56	Kunjungan dari node 55 ke node 56
x56.57	Kunjungan dari node 56 ke node 57
x57.58	Kunjungan dari node 57 ke node 58
x58.59	Kunjungan dari node 58 ke node 59
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
x61.60	Kunjungan dari node 61 ke node 60
x61.62	Kunjungan dari node 61 ke node 62
x62.61	Kunjungan dari node 62 ke node 61
x62.63	Kunjungan dari node 62 ke node 63
x63.62	Kunjungan dari node 63 ke node 62
x63.64	Kunjungan dari node 63 ke node 64
x64.65	Kunjungan dari node 64 ke node 65
x65.32	Kunjungan dari node 65 ke node 32
x66.63	Kunjungan dari node 66 ke node 63
x63.67	Kunjungan dari node 63 ke node 67
x67.68	Kunjungan dari node 67 ke node 68
x68.69	Kunjungan dari node 68 ke node 69
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
x72.73	Kunjungan dari node 72 ke node 73
x73.74	Kunjungan dari node 73 ke node 74
x74.66	Kunjungan dari node 74 ke node 66

Variabel	Keterangan
x58.75	Kunjungan dari node 58 ke node 75
x75.58	Kunjungan dari node 75 ke node 58
x75.76	Kunjungan dari node 75 ke node 76
x76.75	Kunjungan dari node 76 ke node 75
x76.77	Kunjungan dari node 76 ke node 77
x77.76	Kunjungan dari node 77 ke node 76
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
x80.81	Kunjungan dari node 80 ke node 81
x81.79	Kunjungan dari node 81 ke node 79
x79.82	Kunjungan dari node 79 ke node 82
x82.81	Kunjungan dari node 82 ke node 81
x81.83	Kunjungan dari node 81 ke node 83
x83.81	Kunjungan dari node 83 ke node 81
x83.84	Kunjungan dari node 83 ke node 84
x84.83	Kunjungan dari node 84 ke node 83
x84.85	Kunjungan dari node 84 ke node 85
x85.84	Kunjungan dari node 85 ke node 84
x85.86	Kunjungan dari node 85 ke node 86
x86.85	Kunjungan dari node 86 ke node 85
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
x88.87	Kunjungan dari node 88 ke node 87
x88.89	Kunjungan dari node 88 ke node 89
x89.88	Kunjungan dari node 89 ke node 88

Variabel	Keterangan
x89.90	Kunjungan dari node 89 ke node 90
x90.89	Kunjungan dari node 90 ke node 89
x90.91	Kunjungan dari node 90 ke node 91
x91.92	Kunjungan dari node 91 ke node 92
x92.91	Kunjungan dari node 92 ke node 91
x92.90	Kunjungan dari node 92 ke node 90
x92.93	Kunjungan dari node 92 ke node 93
x93.92	Kunjungan dari node 93 ke node 92
x93.94	Kunjungan dari node 93 ke node 94
x94.93	Kunjungan dari node 94 ke node 93
x94.95	Kunjungan dari node 94 ke node 95
x95.94	Kunjungan dari node 95 ke node 94
x95.30	Kunjungan dari node 95 ke node 30
x30.95	Kunjungan dari node 30 ke node 95
x92.96	Kunjungan dari node 92 ke node 96
x96.92	Kunjungan dari node 96 ke node 92
x96.97	Kunjungan dari node 96 ke node 97
x97.96	Kunjungan dari node 97 ke node 96
x97.98	Kunjungan dari node 97 ke node 98
x98.97	Kunjungan dari node 98 ke node 97
x98.99	Kunjungan dari node 98 ke node 99
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
x102.101	Kunjungan dari node 102 ke node 101
x102.103	Kunjungan dari node 102 ke node 103

Variabel	Keterangan
x103.102	Kunjungan dari node 103 ke node 102
x103.104	Kunjungan dari node 103 ke node 104
x104.103	Kunjungan dari node 104 ke node 103
x104.105	Kunjungan dari node 104 ke node 105
x105.104	Kunjungan dari node 105 ke node 104
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
x107.18	Kunjungan dari node 107 ke node 18
x18.107	Kunjungan dari node 18 ke node 107
x106.108	Kunjungan dari node 106 ke node 108
x107.109	Kunjungan dari node 107 ke node 109
x109.107	Kunjungan dari node 109 ke node 107
x109.110	Kunjungan dari node 109 ke node 110
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

Variabel	Keterangan
x117.116	Kunjungan dari node 117 ke node 116
x117.118	Kunjungan dari node 117 ke node 118
x118.117	Kunjungan dari node 118 ke node 117
x118.119	Kunjungan dari node 118 ke node 119
x119.118	Kunjungan dari node 119 ke node 118
x119.120	Kunjungan dari node 119 ke node 120
x120.119	Kunjungan dari node 120 ke node 119
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
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
x125.126	Kunjungan dari node 125 ke node 126
x126.125	Kunjungan dari node 126 ke node 125
x126.127	Kunjungan dari node 126 ke node 127
x127.126	Kunjungan dari node 127 ke node 126
x127.128	Kunjungan dari node 127 ke node 128
x128.127	Kunjungan dari node 128 ke node 127
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

Variabel	Keterangan
x131.91	Kunjungan dari node 131 ke node 91
x91.131	Kunjungan dari node 91 ke node 131
x127.132	Kunjungan dari node 127 ke node 132
x132.127	Kunjungan dari node 132 ke node 127
x132.133	Kunjungan dari node 132 ke node 133
x133.132	Kunjungan dari node 133 ke node 132
x133.134	Kunjungan dari node 133 ke node 134
x134.133	Kunjungan dari node 134 ke node 133
x134.135	Kunjungan dari node 134 ke node 135
x135.134	Kunjungan dari node 135 ke node 134
x135.136	Kunjungan dari node 135 ke node 136
x136.135	Kunjungan dari node 136 ke node 135
x136.137	Kunjungan dari node 136 ke node 137
x137.136	Kunjungan dari node 137 ke node 136
x137.138	Kunjungan dari node 137 ke node 138
x138.137	Kunjungan dari node 138 ke node 137
x138.139	Kunjungan dari node 138 ke node 139
x139.138	Kunjungan dari node 139 ke node 138
x139.140	Kunjungan dari node 139 ke node 140
x140.139	Kunjungan dari node 140 ke node 139
x140.103	Kunjungan dari node 140 ke node 103
x103.140	Kunjungan dari node 103 ke node 140
x124.141	Kunjungan dari node 124 ke node 141
x141.124	Kunjungan dari node 141 ke node 124
x141.142	Kunjungan dari node 141 ke node 142
x142.141	Kunjungan dari node 142 ke node 141
x142.143	Kunjungan dari node 142 ke node 143
x143.142	Kunjungan dari node 143 ke node 142
x143.144	Kunjungan dari node 143 ke node 144

Variabel	Keterangan
x144.143	Kunjungan dari node 144 ke node 143
x144.145	Kunjungan dari node 144 ke node 145
x145.144	Kunjungan dari node 145 ke node 144
x145.146	Kunjungan dari node 145 ke node 146
x146.145	Kunjungan dari node 146 ke node 145
x146.147	Kunjungan dari node 146 ke node 147
x147.146	Kunjungan dari node 147 ke node 146
x147.148	Kunjungan dari node 147 ke node 148
x148.147	Kunjungan dari node 148 ke node 147
x148.149	Kunjungan dari node 148 ke node 149
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
x151.152	Kunjungan dari node 151 ke node 152
x152.151	Kunjungan dari node 152 ke node 151
x152.153	Kunjungan dari node 152 ke node 153
x153.152	Kunjungan dari node 153 ke node 152
x153.154	Kunjungan dari node 153 ke node 154
x154.153	Kunjungan dari node 154 ke node 153
x154.86	Kunjungan dari node 154 ke node 86
x86.154	Kunjungan dari node 86 ke node 154
x82.155	Kunjungan dari node 82 ke node 155
x155.82	Kunjungan dari node 155 ke node 82
x155.156	Kunjungan dari node 155 ke node 156
x156.155	Kunjungan dari node 156 ke node 155
x156.157	Kunjungan dari node 156 ke node 157
x157.156	Kunjungan dari node 157 ke node 156

Variabel	Keterangan
x157.158	Kunjungan dari node 157 ke node 158
x158.157	Kunjungan dari node 158 ke node 157
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
x163.162	Kunjungan dari node 163 ke node 162
x163.164	Kunjungan dari node 163 ke node 164
x164.163	Kunjungan dari node 164 ke node 163
x164.165	Kunjungan dari node 164 ke node 165
x165.164	Kunjungan dari node 165 ke node 164
x165.166	Kunjungan dari node 165 ke node 166
x166.165	Kunjungan dari node 166 ke node 165
x143.167	Kunjungan dari node 143 ke node 167
x167.143	Kunjungan dari node 167 ke node 143
x167.168	Kunjungan dari node 167 ke node 168
x168.167	Kunjungan dari node 168 ke node 167
x168.169	Kunjungan dari node 168 ke node 169
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

Variabel	Keterangan
x172.171	Kunjungan dari node 172 ke node 171
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
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
x108.179	Kunjungan dari node 108 ke node 179
x179.108	Kunjungan dari node 179 ke node 108
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

Variabel	Keterangan
x186.187	Kunjungan dari node 186 ke node 187
x187.186	Kunjungan dari node 187 ke node 186
x187.188	Kunjungan dari node 187 ke node 188
x188.187	Kunjungan dari node 188 ke node 187
x182.189	Kunjungan dari node 182 ke node 189
x189.190	Kunjungan dari node 189 ke node 190
x190.191	Kunjungan dari node 190 ke node 191
x191.16	Kunjungan dari node 191 ke node 16

“Halaman ini sengaja dikosongkan”

LAMPIRAN C

Lampiran ini merupakan daftar seluruh waktu tempuh antar *node* yang saling terhubung langsung beserta kode angkot yang melewati *node* tersebut.

Table C.1 Daftar Nilai Arc antar Node

Node Awal	Node Akhir	Waktu (menit)	Trayek
1	2	1	DP, Q
2	1	1	DP, Q
2	3	1	DP, Q
3	2	1	DP, Q
3	4	1	Q
4	3	1	DP, Q, Z
3	5	2	DP, Z
6	4	1	DP, Q, Z
4	7	1	Q
5	7	2	DP, Z
7	8	1	DP, Q, Z
8	9	1	DP, Z
9	10	1	Q
10	11	1	Q
11	6	1	Q
9	12	1	DP, Z
12	13	1	DP, Z
14	6	1	DP, Z
15	14	1	DP, Z
13	15	1	DP, Z
16	15	1	DP, Z
17	16	1	DP, Z

C-2

Node Awal	Node Akhir	Waktu (menit)	Trayek
18	17	1	DP, Z
15	18	1	DP, Z
16	18	1	Z1
8	19	1	Q
20	9	1	Q
19	21	2	Q
21	20	1	Q
21	22	2	Q
22	21	2	Q
22	23	3	Q
23	22	3	Q
23	24	2	Q
24	23	2	Q
24	25	2	Q
25	24	2	Q
25	26	3	Q
26	25	3	Q
26	27	2	Q
27	26	2	Q
27	28	2	Q
28	27	2	Q
28	29	3	Q
29	28	3	Q
29	30	3	Q
30	29	3	Q
30	31	1	Q, I
31	30	1	Q, I
31	32	1	Q

Node Awal	Node Akhir	Waktu (menit)	Trayek
32	31	1	Q, I
32	33	3	Q
33	32	3	Q
33	34	2	Q
34	33	2	Q
34	35	4	Q
35	34	4	Q
35	36	2	Q
36	35	2	Q
36	37	1	Q
37	36	1	Q
38	37	2	Q
39	38	2	Q
37	40	2	Q
40	41	2	Q
41	39	2	Q
41	42	1	Q
42	41	1	Q
42	43	3	Q
43	42	3	Q
43	44	1	Q
44	43	1	Q
45	44	1	Q
46	45	2	Q
44	47	2	Q
47	48	1	Q
48	46	2	Q
31	49	2	I

C-4

Node Awal	Node Akhir	Waktu (menit)	Trayek
49	50	2	I
50	51	3	I
51	52	2	I
52	53	1	I
53	54	1	I
54	55	1	I
55	56	1	I
56	57	1	I
57	58	1	I
58	59	1	I, TV2
59	58	1	TV2
59	60	1	I, TV2
60	59	1	TV2
60	61	1	I, TV2
61	60	1	TV2
61	62	1	I, TV2
62	61	1	TV2
62	63	1	I, TV2
63	62	1	TV2
63	64	1	I
64	65	1	I
65	32	1	I
66	63	1	TV2
63	67	3	TV2
67	68	1	TV2
68	69	1	TV2
69	70	1	TV2
70	71	1	TV2

Node Awal	Node Akhir	Waktu (menit)	Trayek
71	72	1	TV2
72	73	1	TV2
73	74	1	TV2
74	66	1	TV2
58	75	2	TV2
75	58	2	TV2
75	76	2	TV2
76	75	2	TV2
76	77	2	TV2
77	76	2	TV2
77	78	1	TV2
78	77	1	TV2
78	79	1	TV2
79	78	1	TV2
79	80	2	TV2
80	81	2	TV2
81	79	2	DP, TV2
79	82	2	DP
82	81	3	DP
81	83	1	DP, TV2
83	81	1	DP, TV2
83	84	1	DP, TV2
84	83	1	DP, TV2
84	85	1	DP, TV2
85	84	1	DP, TV2
85	86	2	DP, TV2
86	85	2	DP, TV2
86	87	1	DP

C-6

Node Awal	Node Akhir	Waktu (menit)	Trayek
87	86	1	DP
87	88	1	DP
88	87	1	DP
88	89	1	DP
89	88	1	DP
89	90	1	DP
90	89	1	DP
90	91	1	DP
91	92	1	DP, I
92	91	1	I
92	90	2	DP
92	93	1	I
93	92	1	I
93	94	1	I
94	93	1	I
94	95	1	I
95	94	1	I
95	30	1	I
30	95	1	I
92	96	1	I
96	92	1	I
96	97	2	I
97	96	2	I
97	98	2	I
98	97	2	I
98	99	2	I
99	98	2	I
99	100	1	I

Node Awal	Node Akhir	Waktu (menit)	Trayek
100	99	1	I
100	101	1	I
101	100	1	I
101	102	1	I
102	101	1	I
102	103	1	I
103	102	1	I
103	104	1	I, Z1
104	103	1	I, Z1
104	105	1	I, Z1
105	104	1	I, Z1
105	106	1	I, Z1
106	105	1	I, Z1
106	107	1	DP
107	106	1	DP, Z1
107	18	1	DP, Z
18	107	1	DP, Z, Z1
106	108	2	Z1
107	109	2	Z
109	107	2	Z
109	110	1	Z
110	109	1	Z
110	111	1	Z
111	110	1	Z
111	112	2	Z
112	111	2	Z
112	113	1	Z
113	112	1	Z

C-8

Node Awal	Node Akhir	Waktu (menit)	Trayek
113	114	1	Z
114	113	1	Z
114	115	1	Z
115	114	1	Z
115	116	2	Z
116	115	2	Z
116	117	2	Z
117	116	2	Z
117	118	1	Z
118	117	1	Z
118	119	1	Z
119	118	1	Z
119	120	1	Z
120	119	1	Z
120	121	1	Z
121	120	1	Z
121	122	1	Z
122	121	1	Z
122	123	1	Z
123	122	1	Z
123	124	2	Z
124	123	2	Z
124	125	1	I, Z1
125	124	1	I, Z1
125	126	1	I, Z1
126	125	1	I, Z1
126	127	1	I, Z1
127	126	1	I, Z1

Node Awal	Node Akhir	Waktu (menit)	Trayek
127	128	2	I
128	127	2	I
128	129	1	I
129	128	1	I
129	130	1	I
130	129	1	I
130	131	1	I
131	130	1	I
131	91	1	I
91	131	1	I
127	132	2	Z1
132	127	2	Z1
132	133	1	Z1
133	132	1	Z1
133	134	1	Z1
134	133	1	Z1
134	135	1	Z1
135	134	1	Z1
135	136	1	Z1
136	135	1	Z1
136	137	1	Z1
137	136	1	Z1
137	138	1	Z1
138	137	1	Z1
138	139	1	Z1
139	138	1	Z1
139	140	1	Z1
140	139	1	Z1

C-10

Node Awal	Node Akhir	Waktu (menit)	Trayek
140	103	2	Z1
103	140	2	Z1
124	141	1	I, Z, Z1
141	124	1	I, Z, Z1
141	142	1	I, Z, Z1
142	141	1	I, Z, Z1
142	143	1	I, Z, Z1
143	142	1	I, Z, Z1
143	144	2	TV2
144	143	2	TV2
144	145	1	TV2
145	144	1	TV2
145	146	1	TV2
146	145	1	TV2
146	147	1	TV2
147	146	1	TV2
147	148	2	TV2
148	147	2	TV2
148	149	1	TV2
149	148	1	TV2
149	150	1	TV2
150	149	1	TV2
150	151	1	TV2
151	150	1	TV2
151	152	1	TV2
152	151	1	TV2
152	153	1	TV2
153	152	1	TV2

Node Awal	Node Akhir	Waktu (menit)	Trayek
153	154	1	TV2
154	153	1	TV2
154	86	1	TV2
86	154	1	TV2
82	155	1	DP
155	82	1	DP
155	156	3	DP
156	155	3	DP
156	157	2	DP
157	156	2	DP
157	158	2	DP
158	157	2	DP
158	159	1	DP
159	158	1	DP
159	160	1	DP
160	159	1	DP
160	161	1	DP
161	160	1	DP
161	162	1	DP
162	161	1	DP
162	163	1	DP
163	162	1	DP
163	164	1	DP
164	163	1	DP
164	165	1	DP
165	164	1	DP
165	166	2	DP
166	165	2	DP

Node Awal	Node Akhir	Waktu (menit)	Trayek
143	167	1	TV2, I, Z, Z1
167	143	1	TV2, I, Z, Z1
167	168	1	I, Z, Z1
168	167	1	I, Z, Z1
168	169	1	I, Z, Z1
169	168	1	I, Z, Z1
169	170	1	I, Z, Z1
170	169	1	I, Z, Z1
170	171	1	I, Z, Z1
171	170	1	I, Z, Z1
171	172	2	I, Z, Z1
172	171	2	I, Z, Z1
172	173	1	I, Z, Z1
173	172	1	I, Z, Z1
173	174	1	I, Z, Z1
174	173	1	I, Z, Z1
174	175	1	I, Z, Z1
175	174	1	I, Z, Z1
175	176	1	I, Z, Z1
176	175	1	I, Z, Z1
176	177	1	I, Z, Z1
177	176	1	I, Z, Z1
177	178	2	I, Z, Z1
178	177	2	I, Z, Z1
108	179	1	Z1
179	180	1	Z1
180	181	2	Z1

Node Awal	Node Akhir	Waktu (menit)	Trayek
181	182	1	Z1
182	183	2	Z1
183	182	2	Z1
183	184	2	Z1
184	183	2	Z1
184	185	2	Z1
185	184	2	Z1
185	186	2	Z1
186	185	2	Z1
186	187	1	Z1
187	186	1	Z1
187	188	2	Z1
188	187	2	Z1
182	189	1	Z1
189	190	1	Z1
190	191	2	Z1
191	16	1	Z1

"Halaman ini sengaja dikosongkan"

LAMPIRAN D

Lampiran ini merupakan hasil running dari uji coba 3 untuk skenario 1 sampai 4. Baris yang diberi warna biru merupakan parameter terbaik dari masing-masing skenario berdasarkan rata-rata nilai *fitness* dari sepuluh running yang telah dilakukan.

Tabel D.1 Hasil Running Uji Coba 3 Skenario 1

Running	Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
1	0.9	0.1	50	7282	65
	0.9	0.1	100	7347	58
	0.9	0.1	150	8592	67
2	0.9	0.1	50	7808	67
	0.9	0.1	100	7762	65
	0.9	0.1	150	7724	61
3	0.9	0.1	50	7469	70
	0.9	0.1	100	7717	68
	0.9	0.1	150	8125	70
4	0.9	0.1	50	8151	70
	0.9	0.1	100	7892	61
	0.9	0.1	150	7887	69
5	0.9	0.1	50	7919	66
	0.9	0.1	100	8209	67
	0.9	0.1	150	7808	64
6	0.9	0.1	50	6723	70
	0.9	0.1	100	8054	68
	0.9	0.1	150	8061	68
7	0.9	0.1	50	7942	70
	0.9	0.1	100	7808	68
	0.9	0.1	150	8321	69
8	0.9	0.1	50	6469	60

	0.9	0.1	100	7595	70
	0.9	0.1	150	7888	67
9	0.9	0.1	50	7898	66
	0.9	0.1	100	8484	70
	0.9	0.1	150	7823	69
10	0.9	0.1	50	7179	58
	0.9	0.1	100	7941	69
	0.9	0.1	150	8222	66
Rata-rata	0.9	0.1	50	7484	66.2
	0.9	0.1	100	7880.9	66.4
	0.9	0.1	150	8045.1	67

Tabel D.2 Hasil Running Uji Coba 3 Skenario 2

Running	Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
1	0.9	0.05	50	8067	70
	0.9	0.05	100	8075	64
	0.9	0.05	150	7810	67
2	0.9	0.05	50	7011	58
	0.9	0.05	100	7522	70
	0.9	0.05	150	8341	66
3	0.9	0.05	50	7110	66
	0.9	0.05	100	7630	67
	0.9	0.05	150	7808	69
4	0.9	0.05	50	5653	70
	0.9	0.05	100	8026	68
	0.9	0.05	150	8197	70
5	0.9	0.05	50	7576	60
	0.9	0.05	100	7472	66
	0.9	0.05	150	7640	67
6	0.9	0.05	50	7437	61

	0.9	0.05	100	7769	67
	0.9	0.05	150	8145	64
7	0.9	0.05	50	7545	64
	0.9	0.05	100	7942	68
	0.9	0.05	150	8200	67
8	0.9	0.05	50	5424	69
	0.9	0.05	100	7133	62
	0.9	0.05	150	8452	67
9	0.9	0.05	50	7114	65
	0.9	0.05	100	7714	67
	0.9	0.05	150	7410	69
10	0.9	0.05	50	6698	64
	0.9	0.05	100	6858	70
	0.9	0.05	150	7914	70
Rata-rata	0.9	0.05	50	6963.5	64.7
	0.9	0.05	100	7614.1	66.9
	0.9	0.05	150	7991.7	67.6

Tabel D.3 Hasil Running Uji Coba 3 Skenario 3

Runing	Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
1	0.8	0.1	50	7362	66
	0.8	0.1	100	7455	68
	0.8	0.1	150	7809	64
2	0.8	0.1	50	7217	62
	0.8	0.1	100	7472	61
	0.8	0.1	150	7808	61
3	0.8	0.1	50	7636	67
	0.8	0.1	100	8183	68
	0.8	0.1	150	7804	67

4	0.8	0.1	50	7599	58
	0.8	0.1	100	7473	68
	0.8	0.1	150	8271	66
5	0.8	0.1	50	7011	58
	0.8	0.1	100	7720	67
	0.8	0.1	150	7630	69
6	0.8	0.1	50	7640	61
	0.8	0.1	100	7046	62
	0.8	0.1	150	7507	70
7	0.8	0.1	50	7179	58
	0.8	0.1	100	7774	66
	0.8	0.1	150	7927	68
8	0.8	0.1	50	7347	62
	0.8	0.1	100	7720	67
	0.8	0.1	150	7907	66
9	0.8	0.1	50	7072	64
	0.8	0.1	100	8324	70
	0.8	0.1	150	8433	68
10	0.8	0.1	50	7445	69
	0.8	0.1	100	8306	68
	0.8	0.1	150	8026	70
Rata-rata	0.8	0.1	50	7350.8	62.5
	0.8	0.1	100	7747.3	66.5
	0.8	0.1	150	7912.2	66.9

Tabel D.4 Hasil Running Uji Coba 3 Skenario 4

Running	Prob CO	Prob Mut	Jum Pop	Fitness	Waktu Tempuh
1	0.8	0.05	50	6445	70
	0.8	0.05	100	7640	65
	0.8	0.05	150	7431	58

2	0.8	0.05	50	6798	65
	0.8	0.05	100	7748	70
	0.8	0.05	150	7556	63
3	0.8	0.05	50	7072	60
	0.8	0.05	100	7263	58
	0.8	0.05	150	8164	70
4	0.8	0.05	50	7362	68
	0.8	0.05	100	8025	70
	0.8	0.05	150	7773	61
5	0.8	0.05	50	7417	64
	0.8	0.05	100	7556	61
	0.8	0.05	150	7536	69
6	0.8	0.05	50	7036	67
	0.8	0.05	100	7724	67
	0.8	0.05	150	7855	69
7	0.8	0.05	50	7069	62
	0.8	0.05	100	8073	65
	0.8	0.05	150	7780	69
8	0.8	0.05	50	7263	58
	0.8	0.05	100	8026	70
	0.8	0.05	150	8403	69
9	0.8	0.05	50	7106	70
	0.8	0.05	100	7144	58
	0.8	0.05	150	7437	68
10	0.8	0.05	50	6927	64
	0.8	0.05	100	7384	67
	0.8	0.05	150	9009	70
Rata-rata	0.8	0.05	50	7049.5	64.8
	0.8	0.05	100	7658.3	65.1
	0.8	0.05	150	7894.4	66.6

"Halaman ini sengaja dikosongkan"

LAMPIRAN E

Lampiran ini merupakan keseluruhan kode dari penerapan algoritma Dijkstra untuk membuat matriks waktu tempuh.

```
/*
 * Kelas ini merupakan pengimplementasian dari algoritma Dijkstra untuk
 * mencari waktu tempuh tercepat antar node. Hasil dari algoritma ini akan
 * disimpan ke dalam file dengan nama Arc191x191.txt pada folder Data.
 */
package opt.op.ga;

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

/**
 *
 * @author 5213100006
 */
public class AlgoritmaDijkstra {

    // A utility function to find the vertex with minimum distance value,
    // from the set of vertices not yet included in shortest path tree
```

E-2

```
    static int jumlahNode = 191;

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

        for (int i = 0; i < jumlahNode; i++) {
            if (sptSet[i] == false && dist[i] <= min) {
                min = dist[i];
                min_index = i;
            }
        }
        return min_index;
    }

    // A utility function to print the constructed distance array
    void printSolution(int dist[], int n) {
        for (int i = 0; i < jumlahNode; 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[jumlahNode];

    Boolean sptSet[] = new Boolean[jumlahNode];

    for (int i = 0; i < jumlahNode; i++) {
        dist[i] = Integer.MAX_VALUE;
        sptSet[i] = false;
    }

    dist[src] = 0;

    for (int count = 0; count < jumlahNode - 1; count++) {

        int u = minDistance(dist, sptSet);
        sptSet[u] = true;

        for (int v = 0; v < jumlahNode; 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];
```

```
                }
            }
        }
        printSolution(dist, jumlahNode);
    }

    // Driver method
    public static void main(String[] args) throws
            FileNotFoundException, IOException {
        //KONVERSI ARRAY 2D UNTUK NILAI ARC DARI FILE .CSV
        String thisLine1;
        BufferedReader bufRdr2D = new BufferedReader(
                new FileReader("data/MatriksAwal.csv"));

        ArrayList<String[]> lines = new ArrayList<String[]>();
        while ((thisLine1 = bufRdr2D.readLine()) != null) {
            lines.add(thisLine1.split(","));
        }

        String[][] array = new String[lines.size()][0];
        lines.toArray(array);

        int[][] graph = new int[jumlahNode][jumlahNode];
        for (int i = 0; i < array.length; i++) {
```

```
            for (int j = 0; j < array.length; j++) {
                graph[i][j] = Integer.parseInt(array[i][j]);
            }
        }

        AlgoritmaDijkstra t = new AlgoritmaDijkstra();

        for (int i = 0; i < jumlahNode; i++) {
            t.dijkstra(graph, i);
            System.out.println();
        }
    }
}
```

“Halaman ini sengaja dikosongkan”

LAMPIRAN F

Lampiran ini merupakan keseluruhan kode dari penerapan *Genetic Algorithm* (GA) untuk menghasilkan solusi optimum dari model *Orienteering Problem* (OP).

```
/*
 * Kelas ini merupakan pengimplementasian Genetic Algorithm (GA) untuk
 * meyelesaikan model Orienteering Problem (OP). Pengguna dapat menentukan
 * sendiri node awal dan tujuan sesuai dengan network model yang telah
 * dibentuk. Hasil dari algoritma ini akan disimpan ke dalam file dengan nama
 * Hasil Running GA.txt pada folder Data.
 */
package opt.op.ga;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;
```

F-2

```
import java.util.List;
import java.util.Random;

/**
 *
 * @author 5213100006
 */
public class GeneticAlgorithm {

    static int nodeAwal = 1;
    static int nodeAkhir = 166;
    static int jumlahNode = 191;
    static int tMax = 70; //Menetapkan nilai batasan waktu (tMax) (dalam
menit)
    static int jumlahPop = 50; //Menetapkan jumlah populasi adalah 30
    static int jumlahSelect = 40; //Menetapkan jumlah hasil seleksi adalah 20
    static int jumlahGenerasi = 300; //Menetapkan jumlah generasi/iterasi
    static double probCrossOver = 0.9;
    static double probMutation = 0.1;

    static int[][] nilaiArc = new int[jumlahNode][jumlahNode];
    static int[] nilaiSkor = new int[jumlahNode];

    static ArrayList<ArrayList<Integer>> arrPopulation
```

```
            = new ArrayList<ArrayList<Integer>>();
    static ArrayList<Integer> arrPopulation_WaktuTempuh
            = new ArrayList<Integer>();
    static ArrayList<Integer> arrPopulation_Fitness
            = new ArrayList<Integer>();

    static ArrayList<ArrayList<Integer>> arrSelection
            = new ArrayList<ArrayList<Integer>>();
    static ArrayList<Integer> arrSelection_WaktuTempuh
            = new ArrayList<Integer>();
    static ArrayList<Integer> arrSelection_Fitness
            = new ArrayList<Integer>();

    static ArrayList<ArrayList<Integer>> arrOffSpring
            = new ArrayList<ArrayList<Integer>>();
    static ArrayList<Integer> arrOffSpring_WaktuTempuh
            = new ArrayList<Integer>();
    static ArrayList<Integer> arrOffSpring_Fitness
            = new ArrayList<Integer>();

    static ArrayList<ArrayList<Integer>> arrSimpanan
            = new ArrayList<ArrayList<Integer>>();
    static ArrayList<Integer> arrSimpanan_WaktuTempuh
            = new ArrayList<Integer>();
```

```
    static ArrayList<Integer> arrSimpanan_Fitness
            = new ArrayList<Integer>();

    /**
     * @param args the command line arguments
     * @throws java.io.FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException,
            IOException {

        //MENCETAK CETAK HASIL RUNNING KE DALAM FILE .txt
        File file = new File("data/Hasil Running GA.txt");
        FileWriter fw = new FileWriter(file);
        PrintWriter pw = new PrintWriter(fw);

        //A. KONVERSI ARRAY 2D UNTUK NILAI ARC DARI FILE .CSV
        String thisLine2D;
        BufferedReader bufRdr2D
                = new BufferedReader(new FileReader("data/Arc191x191.csv"));

        ArrayList<String[]> lines = new ArrayList<String[]>();
        while ((thisLine2D = bufRdr2D.readLine()) != null) {
            lines.add(thisLine2D.split(","));
        }
```

```
//A.1. Mengkonversi list ke dalam String array
String[][] array = new String[lines.size()][0];
lines.toArray(array);

//A.2. Mengkonversi String array ke Integer array
for (int i = 0; i < array.length; i++) {
    for (int j = 0; j < array.length; j++) {
        nilaiArc[i][j] = Integer.parseInt(array[i][j]);
    }
}

//B. KONVERSI ARRAY 1D UNTUK SCORE TIAP NODE DARI FILE .CSV
BufferedReader bufRdr1D
        = new BufferedReader(new FileReader("data/Score191.csv"));

String thisLine1D;
int o = 0;
//B.1. Membaca setiap baris yang ada pada file .csv dan memasukkannya
//ke dalam array integer
while ((thisLine1D = bufRdr1D.readLine()) != null) {
    nilaiSkor[o] = Integer.parseInt(thisLine1D);
    o++;
}
```

```
        //1. PEMBANGKITAN POPULASI AWAL DENGAN RANDOM
        int pop = 0; //Menetapkan inisiasi jumlah individu dalam populasi = 0
        int loop = 0;
        int maxLoop = 500000;
        do {
            //1.1. Menghasilkan nilai random, Rn , diantara indeks awal &
akhir,
            //digunakan untuk menentukan panjang solusi yang bervariasi.
            int Rn = new Random().nextInt(jumlahNode - 2) + 1;

            //1.2. Menghasilkan list, Rl , diantara [1,N] secara random
            ArrayList<Integer> listRl = new ArrayList();
            ArrayList<Integer> listRl_rand //Array yang digunakan untuk random
                    = new ArrayList();     //node diantara node awal dan akhir

            //1.3. Menggenerate isi listRl_rand diantara node awal dan akhir
            for (int i = 1; i < jumlahNode + 1; i++) {
                if (i == nodeAwal || i == nodeAkhir) {
                    continue; //Memastikan tidak ada node awal dan akhir
                }             //yang muncul di tengah kromosom
                listRl_rand.add(i);
            }
            Collections.shuffle(listRl_rand); //Mengacak node yang digenerate
```

```
            //1.4. Mengisi listRl
            listRl.add(nodeAwal); //Merepresentasikan node awal
            for (int i = 0; i < listRl_rand.size(); i++) {
                listRl.add(listRl_rand.get(i)); //Representasi node indeks 2
            }                                   //ke N-1 yang telah dibuat
acak
            listRl.add(nodeAkhir); //Merepresentasikan node akhir

            //1.5. Menghasilkan sebuah sub list, Rs dengan ukuran
            //yang bervariasi sesuai dengan Rn.
            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));

            //1.6. Hanya memasukkan subList Rs (individu) yang unique
            //dan tidak melebihi batasan waktu tMax ke dalam populasi.
            for (int i = 0; i < subListRs.size(); i++) {
                if (hitungWaktuTempuh(subListRs) <= tMax
                        && !arrPopulation.contains(subListRs)) {
                    arrPopulation.add(subListRs);
```

```
                    pop++;
                }
            }

            loop++;

            if (loop == maxLoop) {
                System.out.println("Tidak ditemukan solusi yang "
                        + "layak untuk tMax = " + tMax);
                System.exit(0);
            }

        } while (pop < jumlahPop);

        //1.7. Mengisi array untuk waktu tempuh dan fitness setiap
        //individu dari populasi awal.
        for (int i = 0; i < arrPopulation.size(); i++) {
            arrPopulation_WaktuTempuh.add(
                    hitungWaktuTempuh(arrPopulation.get(i)));
            arrPopulation_Fitness.add(
                    hitungFitness(arrPopulation.get(i)));
        }

        //1.8. Mencari individu terbaik dari proses pembangkitan
```

```
        //populasi awal.
        int indSolusiAwal = 0;
        int optFitnessAwal = arrPopulation_Fitness.get(0);
        for (int i = 1; i < arrPopulation.size(); i++) {
            if (arrPopulation_Fitness.get(i) >= optFitnessAwal) {
                indSolusiAwal = i;
                optFitnessAwal = arrPopulation_Fitness.get(i);
            }
        }

        //1.9. Mencetak hasil individu terbaik.
        pw.println("Generasi 0 | Rute: "
                + arrPopulation.get(indSolusiAwal) + " | Waktu Tempuh: "
                + arrPopulation_WaktuTempuh.get(indSolusiAwal) + " | Fitness:
"
                + arrPopulation_Fitness.get(indSolusiAwal));

        pw.println("---CETAK HASIL PEMBANGKITAN POPULASI AWAL---");
        for (int i = 0; i < arrPopulation.size(); i++) {
            pw.println(arrPopulation.get(i) + " ("
                    + arrPopulation_WaktuTempuh.get(i) + ", "
                    + arrPopulation_Fitness.get(i) + ")");
        }
        int generasi = 0;
```

```
        do {
            //2. MELAKUKAN SELEKSI
            //2.1. Mengambil individu dari populasi awal dengan jumlah
            //yang telah ditentukan sebelumnya
            do {
                int id = rouletteWheelSelection();
                if (!arrSelection.contains(arrPopulation.get(id))) {
                    arrSelection.add(arrPopulation.get(id));
                }
            } while (arrSelection.size() < jumlahSelect);

            //2.2. Mengisi array untuk waktu tempuh dan fitness setiap
            //individu dari hasil seleksi.
            for (int i = 0; i < arrSelection.size(); i++) {
                arrSelection_WaktuTempuh.add(
                        hitungWaktuTempuh(arrSelection.get(i)));
                arrSelection_Fitness.add(
                        hitungFitness(arrSelection.get(i)));
            }

//              pw.println("---CETAK HASIL SELEKSI---");
//              for (int i = 0; i < arrSelection.size(); i++) {
//                  pw.println(arrSelection.get(i) + "("
//                          + arrSelection_WaktuTempuh.get(i) + ", "
```

```
//                          + arrSelection_Fitness.get(i) + ")");
//            }
            //3. MELAKUKAN PERKAWINAN SILANG (CROSSOVER)
            //3.1. Looping untuk semua kemungkinan perkawinan silang.
            for (int i = 0; i < arrSelection.size(); i += 2) {

                double rndProbCO = new Random().nextDouble();
                List<Integer> offSpring = new ArrayList<Integer>();

                //3.2. Memastikan perkawinan silang dilakukan satu kali
                //pada setiap individu hasil seleksi.
                int j = i + 1;

                if (rndProbCO > probCrossOver) {
                    continue; //Crossover hanya dilakukan jika memenuhi
                }             //probabilitas crossover

                //Memastikan Induk 1 memiliki jumlah gen yang lebih
                //sedikit atau sama dengan Induk 2. Indeks i menjadi
                //Induk 1 dan indeks j menjadi Induk 2.
                if (arrSelection.get(i).size()
                        <= arrSelection.get(j).size()) {

                    //Membuat insertion point untuk Induk 1.
```

```
                int insertionPoint = new Random().nextInt(
                        arrSelection.get(i).size() - 1) + 1;
                //Mengambil gen sejumlah insertion point
                //pada Induk 1 dimulai dari indeks 0.
                List<Integer> head = arrSelection.get(i).subList(0,
                        insertionPoint);
                //Merupakan list yang nantinya akan digunakan jika
                //proses perkawinan silang tidak berhasil.
                List<Integer> cdg = arrSelection.get(i).subList(
                        insertionPoint,
                        arrSelection.get(i).size() - 1);
                //Mengambil gen sejumlah insertion point pada
                //Induk 2 dimulai dari indeks sebelum node akhir.
                List<Integer> tail = arrSelection.get(j).subList(
                        arrSelection.get(j).size() - insertionPoint
                        - 1, arrSelection.get(j).size() - 1);

                //Fungsi untuk menghasilkan keturunan. Jika proses
                //perkawinan silang gagal, maka dihasilkan keturunan
                //yang sama dengan induk 1.
                if (head.size() + tail.size() + 1
                        >= arrSelection.get(i).size()) {
                    //Memasukkan semua gen dari head ke keturunan.
                    offSpring.addAll(head);
```

```
                int index = 0;
                //Memasukkan gen dari tail ke dalam keturunan
                //dengan menyesuaikan ukuran Induk 1.
                while (offSpring.size()
                        < arrSelection.get(i).size() - 1) {
                    if (index < tail.size()) {
                        if (!offSpring.contains(
                                tail.get(index))) {
                            offSpring.add(tail.get(index));
                        }
                        index++;
                    } else {
                        //Jika proses memasukkan tail gagal ke
                        //dalam offSpring, maka dimasukkan gen
                        //sisa dari induk 1.
                        offSpring.addAll(cdg);
                    }
                }
                //Memasukkan gen akhir
                offSpring.add(nodeAkhir);
            } else {
                offSpring.addAll(arrSelection.get(i));
            }
            //Memastikan Induk 1 memiliki jumlah gen yang lebih
```

```
                    //sedikit atau sama dengan Induk 2. Indeks j menjadi
                    //Induk 1 dan indeks i menjadi Induk 2. Fungsi
                    //if di bawah ini memiliki isi yang sama dengan
                    //fungsi if sebelumnya.
                } else if (arrSelection.get(i).size()
                        > arrSelection.get(j).size()) {

                    int insertionPoint = new Random().nextInt(
                            arrSelection.get(j).size() - 1) + 1;
                    List<Integer> head
                            = arrSelection.get(j).subList(0,
                            insertionPoint);
                    List<Integer> cdg = arrSelection.get(j).subList(
                            insertionPoint,
                            arrSelection.get(j).size() - 1);
                    List<Integer> tail = arrSelection.get(i).subList(
                            arrSelection.get(i).size() - insertionPoint
                            - 1, arrSelection.get(i).size() - 1);

                    if (head.size() + tail.size() + 1
                            >= arrSelection.get(j).size()) {
                        offSpring.addAll(head);
                        int index = 0;
                        while (offSpring.size()
```

```
                                    < arrSelection.get(j).size() - 1) {
                                if (index < tail.size()) {
                                    if (!offSpring.contains(
                                            tail.get(index))) {
                                        offSpring.add(tail.get(index));
                                    }
                                    index++;
                                } else {
                                    offSpring.addAll(cdg);
                                }
                            }
                            offSpring.add(nodeAkhir);
                        } else {
                            offSpring.addAll(arrSelection.get(j));
                        }
                    }


//                  } else {
//                    continue;
//                  }
                    //3.3. Memasukkan keturunan hasil crossover ke dalam array
                    //yang unique dan memenuhi batasan tMax
                    if (!arrOffSpring.contains(offSpring) && hitungWaktuTempuh(
```

```
                        (ArrayList<Integer>) offSpring) <= tMax) {
                    arrOffSpring.add((ArrayList<Integer>) offSpring);
                }
                //}
            }

            //3.4. Mengisi array untuk waktu tempuh dan fitness setiap
            //individu dari hasil seleksi.
            for (int i = 0; i < arrOffSpring.size(); i++) {
                arrOffSpring_WaktuTempuh.add(
                        hitungWaktuTempuh(arrOffSpring.get(i)));
                arrOffSpring_Fitness.add(
                        hitungFitness(arrOffSpring.get(i)));
            }

//              pw.println("---CETAK HASIL CROSSOVER---");
//              for (int i = 0; i < arrOffSpring.size(); i++) {
//                  pw.println(arrOffSpring.get(i) + "("
//                          + arrOffSpring_WaktuTempuh.get(i)
//                          + ", " + arrOffSpring_Fitness.get(i) + ")");
//              }
            //4. MELAKUKAN MUTASI (DENGAN LOCAL SEARCH)
            for (int i = 0; i < arrOffSpring.size(); i++) {
                double rndProbMutation = new Random().nextDouble();
```

```
                if (rndProbMutation > probMutation) {
                    continue;
                }

                int loopMutation = 0;
                int maxLoopMutation = 1000;

                //4.1. Operator Add.
                do {

                    int rndInsert = new Random().nextInt(
                            arrOffSpring.get(i).size() - 2) + 1;
                    int rndNilaiInsert
                            = new Random().nextInt(jumlahNode) + 1;
                    if (arrOffSpring.get(i).contains(rndNilaiInsert)
                            || rndNilaiInsert == nodeAwal
                            || rndNilaiInsert == nodeAkhir) {
                        continue; //Memastikan node yang dimasukkan bukan
                    }             //merupakan node awal dan akhir
                    arrOffSpring.get(i).add(rndInsert, rndNilaiInsert);

                    //Menyimpan waktu tempuh dan fitness dari hasil proses add
                    int currentWaktuTempuh
                            = hitungWaktuTempuh(arrOffSpring.get(i));
```

```
                    int currentFitness
                            = hitungFitness(arrOffSpring.get(i));
                    //Jika proses add menghasilkan waktu tempuh dan fitness
                    //yang lebih baik atau sama dengan sebelumnya, maka
                    //hasil mutasi akan disimpan menggantikan yang lama.
                    if (currentWaktuTempuh <= arrOffSpring_WaktuTempuh.get(i)
                            && currentFitness >= arrOffSpring_Fitness.get(i))
{
                        arrOffSpring_WaktuTempuh.set(i, currentWaktuTempuh);
                        arrOffSpring_Fitness.set(i, currentFitness);
                        loopMutation = 0; //looping dikembalikan ke awal
                    } else {
                        arrOffSpring.get(i).remove(rndInsert);
                    } //Jika tidak lebih baik, maka dikembalikan seperti
semula

                    loopMutation++;

                } while (loopMutation < maxLoopMutation);

                //4.2. Operator Omit.
                do {
                    int rndDelete = new Random().nextInt(
                            arrOffSpring.get(i).size() - 2) + 1;
```

```
                    //Proses omit dilakukan jika ukuran keturunan lebih dari 3
                    if (arrOffSpring.get(i).size() > 3) {
                        //Temp menyimpan nilai dari gen/node yang dihapus
                        int temp = arrOffSpring.get(i).get(rndDelete);
                        arrOffSpring.get(i).remove(rndDelete);
                        //Menyimpan waktu tempuh & fitness hasil proses omit
                        int currentWaktuTempuh
                                = hitungWaktuTempuh(arrOffSpring.get(i));
                        int currentFitness
                                = hitungFitness(arrOffSpring.get(i));
                        if (currentWaktuTempuh
                                < arrOffSpring_WaktuTempuh.get(i)) {
                            arrOffSpring_WaktuTempuh.set(i,
currentWaktuTempuh);
                            arrOffSpring_Fitness.set(i, currentFitness);
                            loopMutation = 0; //looping dikembalikan ke awal
                        } else {
                            arrOffSpring.get(i).add(rndDelete, temp);
                        } //Jika tdk lebih baik, maka dikembalikan sprt semula

                    } else {
                        break;
                    }
```

```
                    loopMutation++;

                } while (loopMutation < maxLoopMutation);

            }

//              pw.println("---CETAK HASIL MUTASI---");
//              for (int i = 0; i < arrOffSpring.size(); i++) {
//                  pw.println(arrOffSpring.get(i)
//                          + "(" + arrOffSpring_WaktuTempuh.get(i)
//                          + ", " + arrOffSpring_Fitness.get(i) + ")");
//              }
            //5. MEMBUAT POPULASI BARU
            //Memastikan array utk simpan individu populasi & keturunan kosong
            arrSimpanan.clear();
            arrSimpanan_WaktuTempuh.clear();
            arrSimpanan_Fitness.clear();

            //Memasukkan populasi ke dalam array simpanan.
            for (int i = 0; i < arrPopulation.size(); i++) {
                if (!arrSimpanan.contains(arrPopulation.get(i))) {
                    arrSimpanan.add(arrPopulation.get(i));
                }
            }
```

```
//Memasukkan keturunan ke dalam array simpanan
for (int i = 0; i < arrOffSpring.size(); i++) {
    if (arrOffSpring_WaktuTempuh.get(i) <= tMax
            && !arrSimpanan.contains(arrOffSpring.get(i))) {
        arrSimpanan.add(arrOffSpring.get(i));
    }
}

//Mengisi array untuk waktu tempuh dan fitness setiap
//individu dari array simpanan.
for (int i = 0; i < arrSimpanan.size(); i++) {
    arrSimpanan_WaktuTempuh.add(
            hitungWaktuTempuh(arrSimpanan.get(i)));
    arrSimpanan_Fitness.add(
            hitungFitness(arrSimpanan.get(i)));
}

//Mengosongkan isi dari array yang menyimpan populasi
arrPopulation.clear();
arrPopulation_WaktuTempuh.clear();
arrPopulation_Fitness.clear();

//Mengosongkan isi dari array yang menyimpan hasil seleksi
```

```
            arrSelection.clear();
            arrSelection_WaktuTempuh.clear();
            arrSelection_Fitness.clear();

            //Mengosongkan isi dari array yang menyimpan keturunan
            arrOffSpring.clear();
            arrOffSpring_WaktuTempuh.clear();
            arrOffSpring_Fitness.clear();

            //Memilih individu terbaik dari populasi sebelumnya dan keturunan
            //sejumlah populasi sebelumnya.
            do {
                int indSolusi = 0;
                int optFitness = arrSimpanan_Fitness.get(0);
                for (int i = 1; i < arrSimpanan.size(); i++) {
                    if (arrSimpanan_Fitness.get(i) >= optFitness) {
                        indSolusi = i;
                        optFitness = arrSimpanan_Fitness.get(i);
                    }
                }
                arrPopulation.add(arrSimpanan.get(indSolusi));
                arrPopulation_WaktuTempuh.add(
                        arrSimpanan_WaktuTempuh.get(indSolusi));
                arrPopulation_Fitness.add(
```

```
                        arrSimpanan_Fitness.get(indSolusi));
                arrSimpanan.remove(indSolusi);
                arrSimpanan_WaktuTempuh.remove(indSolusi);
                arrSimpanan_Fitness.remove(indSolusi);
            } while (arrPopulation.size() < jumlahPop);

            arrSimpanan.clear();
            arrSimpanan_WaktuTempuh.clear();
            arrSimpanan_Fitness.clear();

            pw.println("---CETAK POPULASI AKHIR---");
            for (int i = 0; i < arrPopulation.size(); i++) {
                pw.println(arrPopulation.get(i)
                        + " (" + arrPopulation_WaktuTempuh.get(i)
                        + ", " + arrPopulation_Fitness.get(i) + ")");
            }
//              pw.println("---CETAK SISA---");
//              for (int i = 0; i < arrSimpanan.size(); i++) {
//                  pw.println(arrSimpanan.get(i)
//                          + "(" + arrSimpanan_WaktuTempuh.get(i)
//                          + ", " + arrSimpanan_Fitness.get(i) + ")");
//              }
            //Mencari individu terbaik dari populasi baru yang terbentuk
            int indSolusi = 0;
```

```
            int optFitness = arrPopulation_Fitness.get(0);
            for (int i = 1; i < arrPopulation.size(); i++) {
                if (arrPopulation_Fitness.get(i) >= optFitness) {
                    indSolusi = i;
                    optFitness = arrPopulation_Fitness.get(i);
                }
            }

            //Mencetak hasil individu terbaik dari setiap generasi
            pw.println("Generasi " + (generasi + 1) + " | Rute: "
                    + arrPopulation.get(indSolusi) + " | Waktu Tempuh: "
                    + arrPopulation_WaktuTempuh.get(indSolusi) + " | Fitness:
"
                    + arrPopulation_Fitness.get(indSolusi));

            generasi++;

        } while (generasi < jumlahGenerasi);

        pw.close();

    }

    //Fungsi untuk menghitung waktu tempuh setiap individu.
```

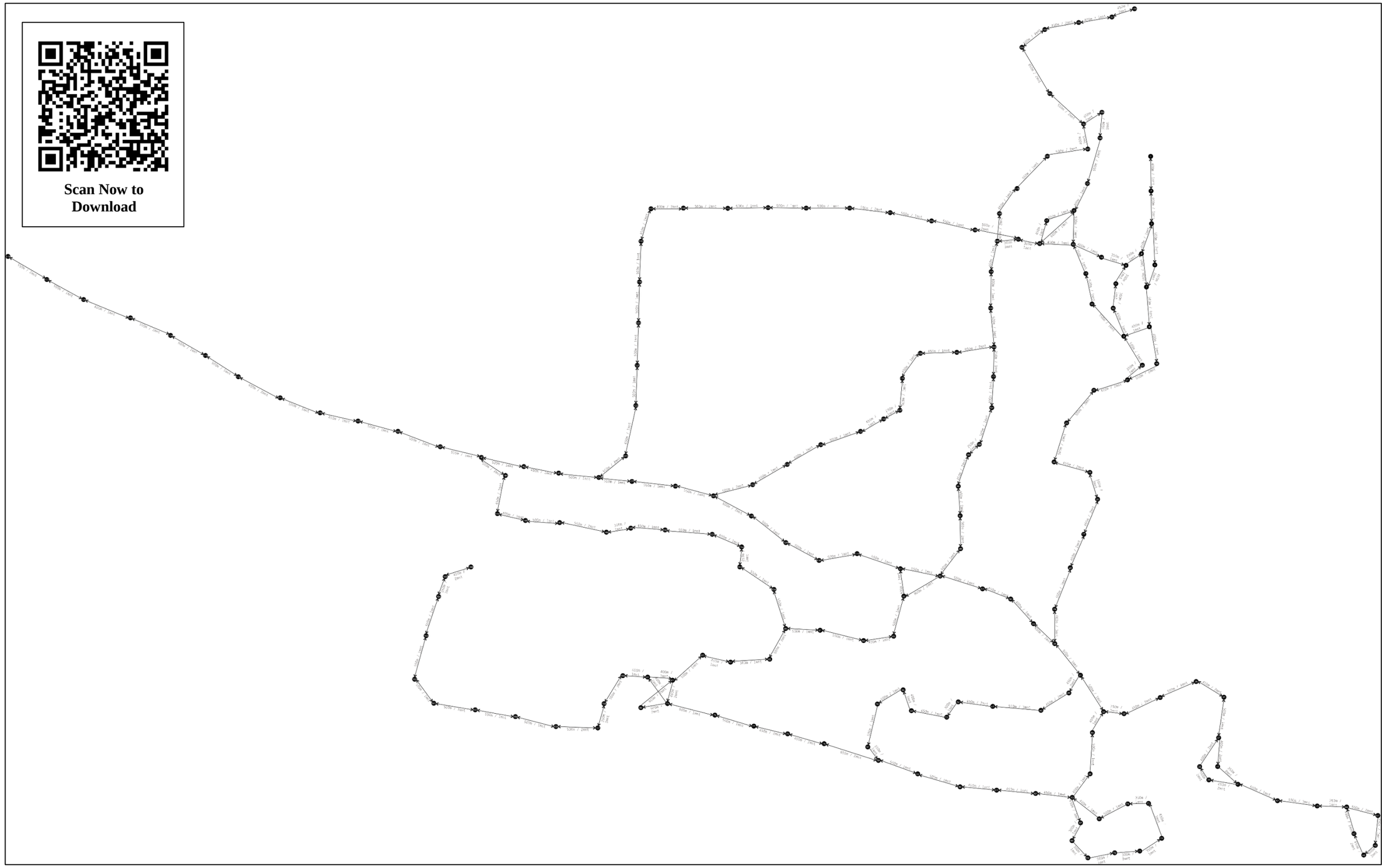
```
    public static int
            hitungWaktuTempuh(ArrayList<Integer> varList) {
        int sum = 0; //Inisiasi nilai awal sum adalah 0
        for (int i = 0; i < varList.size() - 1; i++) {
            sum = sum
                    + nilaiArc[varList.get(i) - 1][varList.get(i + 1) - 1];
        } //Loop di atas utk menghitung waktu tempuh tiap node yg berdekatan
        return sum;
    }

    //Fungsi untuk menghitung fitness setiap individu.
    public static int hitungFitness(ArrayList<Integer> varList) {
        int sum = 0; //Inisiasi nilai awal sum adalah 0
        //int fitness;
        for (int i = 0; i < varList.size(); i++) {
            sum = sum + nilaiSkor[varList.get(i) - 1];
        } //Loop di atas utk menghitung skor tiap node
        //fitness = (sum^3)/hitungWaktuTempuh(varList);
        //return fitness;
        return sum;
    }

    //Fungsi untuk menghitung total fitness pada populasi.
    public static int totalFitness() {
```

```
        int sum = 0; //Inisiasi nilai awal sum adalah 0
        for (int i = 0; i < arrPopulation_Fitness.size(); i++) {
            sum = sum + arrPopulation_Fitness.get(i);
        } //Loop di atas utk menghitung fitness tiap individu pd populasi
        return sum;
    }

    //Fungsi untuk untuk melakukan seleksi individu pada populasi.
    public static int rouletteWheelSelection() {
        double r = new Random().nextDouble();
        int totFitness = totalFitness();
        int k = 0;
        double sum = (double) arrPopulation_Fitness.get(k) / totFitness;
        while (sum < r) {
            k++;
            sum = sum + (double) arrPopulation_Fitness.get(k) / totFitness;
        }
        return k;
    }

}
```

LAMPIRAN G

Gambar G.1 Persebaran Node dalam Trayek

Scan Now to Download



Gambar H.1 Penggambaran Network Model