OPTIMASI RUTE ANGKUTAN KOTA MALANG DENGAN PENERAPAN ALGORITMA GENETIKA

SKRIPSI

Oleh :

KHOIRON NISAA
NIM. 0810963050

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

OPTIMASI RUTE ANGKUTAN KOTA MALANG DENGAN PENERAPAN ALGORITMA GENETIKA

SKRIPSI

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Dalam bidang Ilmu Komputer

Oleh :

KHOIRON NISAA
NIM. 0810963050 – 96

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

LEMBAR PERSETUJUAN

OPTIMASI RUTE ANGKUTAN KOTA MALANG DENGAN PENERAPAN ALGORITMA GENETIKA

SKRIPSI

Oleh :

KHOIRON NISAA
NIM. 0810963050 – 96

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Drs. Muh. Arif Rahman M.Kom	**Drs. Achmad Ridok, M.Kom**
NIP. 19660423 199111 1 001	**NIP. 19680825 199403 1 002**

LEMBAR PENGESAHAN SKRIPSI

OPTIMASI RUTE ANGKUTAN KOTA MALANG DENGAN PENERAPAN ALGORTIMA GENETIKA

SKRIPSI

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Dalam Bidang Ilmu Komputer

Disusun Oleh :
KHOIRON NISAA
NIM. 0810963050-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 22 Januari 2013
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Penguji I,	**Penguji II,**
Dian Eka Ratnawati, S.Si., M.Kom	**Suprapto, S.T., M.T**
NIP. 19730619 200212 2 001	**NIP. 19710727 199603 1 001**

Penguji III,

Imam Cholissodin, S.Si., M.Kom

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2013
Penulis

Khoiron Nisaa
NIM. 0810963050

KATA PENGANTAR

Bismillahirrohmanirrohim

Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat, hidayah, anugerah serta bimbingan-Nya penulis dapat menyesaikan skripsi yang berjudul **" Optimasi Rute Angkutan Kota Malang dengan Penerapan Algoritma Genetika"**. Sholawat serta salam tak lupa penulis haturkan kepada Nabi Muhammad SAW, keluarga serta sahabatnya yang telah mengiringi penulis dalam menyelesaikan skripsi.

Selama penyelesaian skripsi, tak sedikit penulis mendapatkan bantuan petunjuk, ilmu, bimbingan, do'a, motivasi serta saran dari berbagai pihak hingga penulis dapat menyelesaikan skripsi ini. Oleh karena itu segala rasa terima kasih penulis sampaikan kepada yang terhormat :

1. Drs. Muh. Arif Rahman, M.Kom selaku dosen pembimbing utama serta pembimbing akademik, atas kesediaan menjadi pembimbing, ilmu yang berharga, segala masukan, kesabaran dalam membimbing, nasehat dan motivasi yang telah diberikan.
2. Drs. Achmad. Ridok, M.Kom selaku dosen pembimbing pendamping, atas segala kesabaran dalam membimbing, ilmu yang berharga, nasehat dan motivasi yang telah diberikan.
3. Kedua orang tua penulis. Umik tercinta atas segala kasih sayang, do'a yang tak henti, nasehat, pelajaran hidup yang berharga dan segala keikhlasan yang diberikan. Almarhum buya, atas segala pelajaran hidup yang cukup singkat dan berharga.
4. Drs. Marji, M.T, selaku Ketua Program Studi Ilmu Komputer. Terima kasih atas segala bantuan dan kemudahan yang telah diberikan.
5. Segenap bapak dan ibu dosen dengan ikhlas dan sabar telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
6. Kakak, adik dan semua keluarga besar. Terima kasih atas do'a yang tak henti-henti, motivasi dan kasih sayang yang selalu mengiringi penulis.

7. Dian C, Dafid Eko, Resthy, Tirana, Fahrur dan Nike terima kasih atas segala diskusi, motivasi, semangat, segala bantuan dan do'a yang selalu mengiringi penulis.
8. Teman – teman seperjuangan, ilkom 2k8, ilkom 2008 dan *all* ilkomers. Atas do'a, bantuan, motivasi dan pengalaman selama perkuliahan.
9. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala bantuan dalam penyelesaian skripsi.

Penulis berharap agar apa yang telah penulis buat dapat bermanfaat nantinya. Dan besar harapan penulis semoga skripsi ini dapat bermanfaat bagi penulis khususnya dan semua pihak yang berkepentingan.

Penulis menyadari bahwa masih ada ketidaksempurnaan selama penyelesaian skripsi ini. Pada kesempatan ini pula penulis memohon maaf atas segala ketidaksempurnaan yang ada. Dengan tangan terbuka penulis menerima kritik dan saran dari berbagai pihak bersifat membangun untuk bahan perbaikan dimasa yang akan datang,nisaelmuzammil@gmail.com.

Malang, Januari 2013

Penulis

ABSTRAK

Khoiron Nisaa. 2013 : Optimasi Rute Angkutan Kota Malang Dengan Penerapan Algoritma Genetika

Dosen Pembimbing : Drs. Muh. Arif Rahman, M.Kom dan Drs. Achmad Ridok, M.Kom

Angkutan kota sebagai angkutan umum yang murah dan menjangkau sebagian besar masyarakat kelas menengah kebawah masih banyak beroperasi dan diburu oleh masyarakat. Banyaknya angkutan kota yang beroperasi menimbulkan masalah tersendiri dalam hal pengaturan rute trayek angkutan kota dan pengoptimalan penggunaan bahan bakar. Pengoptimalan rute angkutan kota dapat dilakukan dengan optimasi rute angkutan kota, dalam studi penentuan rute trayek angkutan kota Malang dengan meminimalkan jarak tempuh dan meminimalkan penumpukan (*intersection*) jalur antar trayek dengan mempertahankan bahwa setiap trayek harus memenuhi titik awal dan titik tujuan yang telah ditetapkan. Salah satu cara untuk menyelesaikan permasalahan optimasi tersebut yaitu dengan penerapan algoritma genetika.

Pada algortima genetika terdapat 4 parameter yang mempengaruhi hasil *fitness* yang diperoleh yaitu jumlah populasi, probabilitas *crossover*, probabilitas mutasi dan jumlah generasi. Pada penelitian ini dilakukan analisa pengaruh parameter genetika terhadap nilai *fitness* rata – rata yang diperoleh. Hasil penelitian menunjukkan bahwa perubahan parameter genetika probabilitas *crossover* (pc) dan jumlah generasi mempunyai pengaruh yang cukup signifikan terhadap kenaikan nilai fitness yang diperoleh, yaitu semakin besar nilai probabilitas *crossover*, nilai *fitness* mengalami kenaikan (nilai *fitness* tinggi) , begitu juga untuk jumlah generasi. Dan dari nilai *fitness* terbaik yaitu pada pengujian dengan pc : 90%, pm :30%, jumlah generasi 30 dan jumlah populasi 100 diperoleh hasil bahwa sebanyak 19 dari 25 trayek, memenuhi jarak tempuh yang telah ditentukan.

Kata Kunci : Optimasi Rute, Angkutan Kota Malang, Algoritma Genetika

ABSTRACT

Khoiron Nisaa. 2013 : Route Optimization on Public Transportation of Malang With Genetic Algorithm Implementation

Advisor : Drs. Muh. Arif Rahman, M.Kom and Drs. Achmad Ridok, M.Kom

Currently, some people still choose to ride public transportation because the price is comparatively cheap, especially for middle-low class people. The high number of public transportation operating may cause the problem in term of routing optimization and fuel usage. Public transport route optimization can be done by route optimization with minimizing the mileage and the intersection between trajectories on public transportation of Malang, which each route has to fulfill the starting and destined point. One of the ways to overcome the optimization problem is using genetic algorithm.

In genetic algorithm, there are four parameters that affect the fitness of population, which are number of population, crossover probability, mutation probability, and number of generation. This research analyzes the correlation between four genetic parameters and the average value of fitness. The result showed that the genetic parameters; crossover probability and number of generation have significant influence to fitness, the higher the value of crossover probability, fitness value will also get higher, so does the number of generation. From testing, the best fitness value is 0.00099514 with pc: 90%, pm: 30%, number of generation: 30 and number of populations : 100, the testing showed that 19 out of 25 trajectories complete the specified mileage.

Key word **:** Route Optimization, Public Transportation of Malang, Genetics algorithm

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SOURCECODE

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kota Malang sebagai kota terbesar kedua di Jawa Timur memiliki tingkat pertumbuhan penduduk 3,9% per tahun [PUT-12]. Dalam basis data dinas kependudukan menyebutkan bahwa penduduk kota malang per tanggal 25 Maret 2011 berjumlah 894.653 jiwa [DIN-12]. Sebagai kota pendidikan, kota Malang banyak didatangi mahasiswa dari luar kota, luar pulau bahkan dari luar negeri. Laju peningkatan penduduk ini menjadi faktor yang berpengaruh terhadap peningkatan kendaraan bermotor dan tidak menutup kemungkinan meningkatnya pula jumlah transportasi umum. Meningkatnya jumlah kendaraan mengakibatkan semakin tingginya tingkat mobilitas sehingga semakin tinggi pula persaingan dalam hal penggunaan jalan. Peningkatan kegiatan transportasi memberikan banyak dampak negatif terhadap kota, antara lain : Kemacetan dan eksploitasi sumber energi khususnya BBM [HAN-09]. Hal ini dapat dikurangi dengan pengoptimalan sistem transportasi, salah satunya adalah angkutan kota.

Angkutan kota sebagai angkutan umum yang murah dan menjangkau sebagian besar masyarakat kelas menengah kebawah masih banyak beroperasi dan diburu oleh masyarakat. Banyaknya angkutan kota yang beroperasi menimbulkan masalah tersendiri dalam hal pengaturan rute trayek angkutan kota dan pengoptimalan penggunaan bahan bakar. Rute trayek angkutan kota diatur sedemikian rupa sehingga rute trayek tiap angkutan kota tidak ada yang sama. Meski demikian masih ada beberapa rute trayek angkutan kota yang mengalami penumpukan jalur dibeberapa titik. Berdasarkan data trayek angkutan kota yang diperoleh terlihat bahwa adanya penumpukan jalur dibeberapa titik, salah satunya pada trayek angkutan kota jenis AT, AG, ABB dan ADL mengalami penumpukan jalur di Jalan Jendral A.Yani [PEM-12]. Sedangkan cara untuk mengoptimalan penggunaan bahan bakar dapat dilakukan dengan pengoptimalan rute angkutan dari segi jarak tempuh. Hal ini dapat menekan jumlah penggunaan bahan bakar oleh angkutan kota.

Pengoptimalan rute angkutan kota Malang dapat dilakukan dengan optimasi rute angkutan kota, studi penentuan rute trayek angkutan kota Malang dengan meminimalkan jarak tempuh dan meminimalkan penumpukan (*intersection*) jalur antar trayek dengan mempertahankan titik awal dan titik tujuan yang telah ditetapkan Permasalahan optimasi mengacu pada sebuah metode yang dapat melaksanakan tugas secara optimal, dapat diselesaikan secara cepat dengan memilih solusi yang terbaik [SET-03]. Salah satu cara untuk menyelesaikan permasalahan optimasi yaitu dengan algoritma genetika, begitu halnya dalam permasalahan optimasi rute angkutan kota ini.

Algoritma genetika pertama kali dikembangkan oleh John Holland bersama rekan kerja dan murid – muridnya pada tahun 1975 di Universitas Michigan. Konsep dasar algoritma genetika diilhami berdasarkan teori evolusi alam yang dikemukakan oleh Charles Darwin. Dalam penelitiannya John Holland dan rekan melakukan uji coba untuk memanfaatkan prinsip proses evolusi dalam suatu perangkat lunak guna memecahkan suatu permasalahan optimasi [ROB-06]. Algoritma genetika merupakan salah satu algoritma yang terdapat diantara teknik – teknik yang memiliki kemampuan intelejen, dalam hal ini adalah algoritma stokastik dengan memanfaatkan fenomena alam [MIC-99]. Selain itu algoritma genetika sebagai cabang dari algoritma evolusi merupakan metode *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah permasalahan optimasi [ANO-12]

Sebelumnya telah ada penelitian yang dilakukan oleh [BUD-07] mengenai optimasi dengan menggunakan algoritma genetika, akan tetapi studi kasus yang digunakan adalah penentuan jalur kereta api. Dalam penelitiannya, peneliti menggunakan faktor antara lain jumlah penumpang dan jarak tempuh kereta. Penelitian terkait angkutan kota Malang juga pernah dilakukan oleh [ADH-09]. Peneliti melakukan penelitian mengenai sistem rekomendasi angkutan umum dengan menggunakan algoritma genetika. Sedangkan pada tahun 2011, terdapat penelitian yang dilakukan oleh [SAR-11] mengenai pencarian rute perjalanan tercepat menggunakan algoritma genetika. Akan tetapi dalam penelitiannya, peneliti menggunakan obyek secara umum yaitu kendaraan bermotor roda empat

(kendaraan pribadi). Dalam penelitiannya, peneliti menggunakan faktor kepadatan lalu lintas dan kecepatan maksimal yang diperbolehkan.

Dalam skripsi ini akan dilakukan penelitian mengenai optimasi rute angkutan kota Malang dengan penerapan algoritma genetika, studi penentuan rute trayek angkutan kota Malang dengan meminimalkan jarak tempuh dan meminimalkan penumpukan (*intersection*) jalur antar trayek.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka dibangun suatu rumusan masalah sebagai berikut :

1. Bagaimana menerapkan algoritma genetika untuk permasalahan optimasi rute angkutan kota Malang?
2. Bagaimana pengaruh parameter genetika (probabilitas crossover, probabilitas mutasi, jumlah populasi dan jumlah generasi) terhadap hasil dari optimasi rute angkutan kota?

1.3 Batasan Masalah

Penulisan skripsi ini perlu didefinisikan beberapa batasan masalah mengenai sejauh mana penelitian dan penulisan skripsi ini dikerjakan. Beberapa batasan masalah tersebut antara lain :

1. Objek yang digunakan dalam penelitian adalah angkutan kota Malang.
2. Parameter yang diproses adalah jarak tempuh dan jumlah *intersection* (tumpukan) jalur antar trayek dengan tidak mempertimbangkan rute yang dilalui dan jumlah penumpang.
3. Data yang digunakan dalam penelitian adalah
   - Data jaringan trayek angkutan kota beserta peta trayek angkutan kota Malang yang didapat dari Dinas Perhubungan kota Malang.
   - Data jalan kota Malang pada tahun 2009 yang didapat dari Dinas Perhubungan kota Malang ditambah dari hasil pengukuran berdasarkan skala peta yang digunakan.
4. Rute angkutan yang diproses sebatas rute keberangkatan dengan mengasumsikan bahwa rute berangkat dan pulang adalah sama.

5. Mengabaikan jalan dengan jalur yang searah (semua jalur diasumsikan dua arah) dan semua jalan diasumsikan dalam keadaan baik sehingga layak dilewati oleh angkutan kota.
6. Hasil yang diperoleh tidak dibandingkan dengan hasil dari metode yang lain.

1.4 Tujuan Penelitian

Adapun tujuan penelitian dan penulisan skripsi ini adalah sebagai berikut :

1. Menerapkan algoritma genetika untuk permasalahan optimasi rute angkutan kota Malang.
2. Mengukur dan mengetahui pengaruh parameter genetika (probabilitas crossover, probabilitas mutasi, jumlah populasi dan jumlah generasi) terhadap hasil yang diperoleh pada optimasi rute angkutan kota dikota Malang.

1.5 Manfaat

Manfaat yang diperoleh dari penulisan skripsi ini adalah mendapatkan suatu pemecahan masalah dalam optimasi rute angkutan kota Malang dengan tujuan meminimalkan jarak tempuh dan jumlah *intersection* (penumpukan) jalur antar trayek angkutan.

1.6 Metodologi Penelitian

Metodologi yang digunakan dalam penulisan skripsi ini adalah sebagai berikut :

1. Studi Literatur

   Studi literatur digunakan sebagai bahan acuan dan penunjang dalam penulisan skripsi. Studi literatur dilakukan dengan mempelajari teori yang berhubungan dengan topik skripsi yaitu rute trayek angkutan kota Malang dan algoritma genetika.

2. Pendefinisian dan Analisis Masalah

   Mendefiniskan dan menganalisis masalah yang diangkat dalam penelitian skripsi untuk memperoleh solusi yang tepat.

3. Perencanaan dan Implementasi Perangkat Lunak

   Pengolahan data mentah yang ada hingga menjadi data yang memungkinkan untuk diimplementasikan kedalam perangkat lunak, perancangan perangkat lunak dan mengimplementasikan hasilnya untuk pembuatan perangkat lunak.

4. Pengujian dan Analisa Hasil Implementasi Perangkat Lunak

   Menguji coba perangkat lunak yang telah dibuat dan menganalisa hasil dari implementasi perangkat lunak tersebut apakah telah sesuai dengan tujuan yang telah dirumuskan, untuk selanjutnya dilakukan evaluasi dan menyempurnaan.

5. Pengambilan Kesimpulan

   Mengambil kesimpulan dari output program yang dihasilkan.

1.7 Sistematika Penelitian

Sistematika penulisan skripsi ini dibagi menjadi lima bab dengan masing – masing bab memiliki pokok bahasan yang berbeda. Adapun penjabaran dari pembahasan masing – masing bab adalah sebagai berikut :

1. BAB I PENDAHULUAN

   Bab ini membahas mengenai latar belakang penulisan skripsi, rumusan masalah, batasan masalah, tujuan penuliasan, manfaat penulisan, metodologii penelitian yang digunakan serta sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

   Bab ini menjelaskan dasar teori penunjang yang digunakan dalam penulisan skripsi.

3. BAB III METODOLOGI DAN PERANCANGAN

   Bab ini membahas mengenai penggunaan teori – teori dalam perancangan perangkat lunak, metode dan konsep perhitungan manual serta membahas perencangan perangkat lunak.

4. BAB IV IMPLEMENTASI DAN PEMBAHASAN

   Bab ini membahas mengenai implementasi dari rancangan yang telah diuraikan dalam Bab III, disertai dengan *source code* pada bagian – bagian yang penting dalam perangkat lunak serta membahas hasil uji coba dari

perangkat lunak diserta evaluasi untuk mengetahui kemampuan perangkat lunak.

5. BAB V PENUTUP

   Bab ini berisi kesimpulan yang diperoleh dari pembahasan materi dan pengujian perangkat lunak serta saran yang diharapkan bermanfaat untuk pengembangan penelitian lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Angkutan Umum dan Angkutan Kota

Angkutan umum merupakan sarana angkutan untuk masyarakat kecil dan menengah supaya dapat melaksanakan kegiatannya sesuai dengan tugas dan fungsinya dalam masyarakat. Berdasarkan Peraturan Daerah Kota Malang Nomor 5 Tahun 2011 pengertian angkutan adalah perpindahan orang dan/atau barang dari satu tempat ke tempat lain dengan menggunakan kendaraan di ruang lalu lintas jalan. Sedangkan pengertian angkutan kota adalah angkutan dari suatu tempat ke tempat lain di daerah kota dengan menggunakan mobil bus umum atau mobil penumpang umum yang terkait dalam trayek [PER-11].

2.2 Trayek Angkutan Umum

Pengertian trayek menurut [KEP-03] tentang Penyelenggaraan Angkutan Orang di Jalan dengan kendaraan Umum, adalah lintasan kendaraan umum untuk pelayanan jasa angkutan orang dengan mobil bus, yang mempunyai asal dan tujuan perjalanan tetap, lintasan tetap dan jadwal tetap maupun tidak terjadwal. Sedangkan pengertian jaringan trayek yaitu kumpulan dari trayek – trayek yang menjadi satu kesatuan jaringan pelayanan angkutan orang.

Jaringan trayek memuat asal tujuan, rute yang dilalui, jenis, klasifikasi lalu lintas, dan kelas jalan. Penetapan jaringan trayek diatur dalam Keputusan Menteri Perhubungan Nomor KM 35 Tahun 2003 bahwa penetapan jaringan trayek dilakukan dengan mempertimbangkan [KEP-03] :

1. Bangkitan dan tarikan perjalanan pada daerah asal dan tujuan.
2. Jenis pelayanan angkutan.
3. Hirarki kelas jalan yang sama dan/atau lebih tinggi sesuai kebutuhan kelas jalan yang berlaku.

4. Tipe terminal yang sesuai dengan jenis pelayanannya dan simpul transportasi lainnya yang meliputi bandar udara, pelabuhan dan stasiun kereta api.
5. Tingkat pelayanan jalan yang berupa perbandingan antara kapasitas jalan dan volume lalu lintas.

2.3 Algoritma Genetika

Algortima genetika pertama kali dikembangkan pada tahun 1975 oleh John Holland dari Unversitas Michigan. Konsep yang mendasari timbulnya algoritma genetika yaitu teori genetika yang dikemukakan oleh Charles Darwin. Dalam teori tersebut dijelaskan bahwa pada proses genetika alami, setiap individu harus melakukan adaptasi terhadap lingkungan disekitarnya agar dapat bertahan hidup [SET-03].

Algoritma genetika juga merupakan suatu algoritma pencarian *heuristic* dengan berdasar pada mekanisme evolusi biologis. Bentuk keberagaman pada evolusi biologis berupa variasi dari kromosom antar individu organisme. Dimana variasi dari kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup. Pada dasarnya proses evolusi dipengaruhi oleh 4 hal yaitu [KUS-03] :

a. Kemampuan suatu organisme untuk melakukan reproduksi.
b. Keberadaan populasi organisme yang dapat melakukan reproduksi.
c. Keberagaman organisme dalam suatu populasi.
d. Adanya perbedaan kemampuan untuk bertahan hidup.

Individu yang lebih kuat akan memiliki tingkat ketahanan yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Sehingga populasi secara keseluruhannya akan lebih banyak memuat organisme yang fit [KUS-05]

2.3.1 Struktur Umum Algoritma Genetika

Algoritma genetika merupakan salah satu bentuk pemecahan masalah yang berasal dari perbaikan pendekatan dari metode yang sudah ada. Perbaikan

pendekatan tersebut dikenal dengan nama *generate and test*. Dengan strategi ini, akan dihasilkan suatu *state* yang kemudian akan dilakukan pengujian terhadap *state* tersebut. Solusi akan diterima ketika telah memenuhi kriteria yang diinginkan [COX-05].

Menurut [SUY-07] dalam bukunya menjelaskan bahwa suatu algoritma genetika dimulai dengan sekumpulan solusi yang disebut populasi, dimana solusi tersebut dinamakan sebagai individu. Satu hal yang tekankan oleh [SUY-07] bahwa satu individu adalah menyatakan satu solusi. Populasi awal akan melakukan evolusi menjadi populasi baru melalui serangkaian iterasi atau biasa disebut dengan generasi. Diakhir iterasi, algoritma genetika akan mengembalikan satu anggota populasi baru yang terbaik sebagai suatu solusi dari suatu permasalahan. Adapun proses evolusi yang dilakukan dalam setiap iterasi adalah :

1. Memilih dua individu sebagai orang tua (*parent*) berdasarkan mekanisme tertentu. Selanjutnya kedua inidividu tersebut dikawinkan melalui operator kawin silang (*crossover*) unuk menghasilkan inidividu anak (*offspring*)
2. Dua individu anak memungkinkan mengalami perubahan gen melalui operator *mutation,* dengan probabilitas tertentu.
3. Menerapkan suatu skema pergantian (*replacement scheme*) sehingga menghasilkan populasi baru.
4. Mengulang proses diatas hingga kondisi berhenti (*stopping condition*) tertentu. Kondisi berhenti dapat berupa jumlah iterasi tertentu, waktu tertentu atau ketika variansi individu – individu dalam populasi tersebut telah lebih kecil dari suatu nilai tertentu yang diinginkan.

Pada dasarnya terdapat dua variasi algoritma genetika, yaitu *Stedy State* dan *Genarational Replacement*. Pada algoritma genetika jenis *stedy state,* proses *replacement* dilakukan setiap kali dihasilkan dua *offspring* hasil *crossover*. *Offspring* ini akan menggantikan kromosom yang memiliki nilai *fitness* yang paling kecil, sehingga dapat dipastikan populasi baru akan memiliki individu – individu yang lebih baik dibandingkan dengan populasi lama. Sedangkan pada jenis *genarational replacement,* proses *replacement* dilakukan sekaligus ketika dihasilkan satu populasi baru [SUY-07].

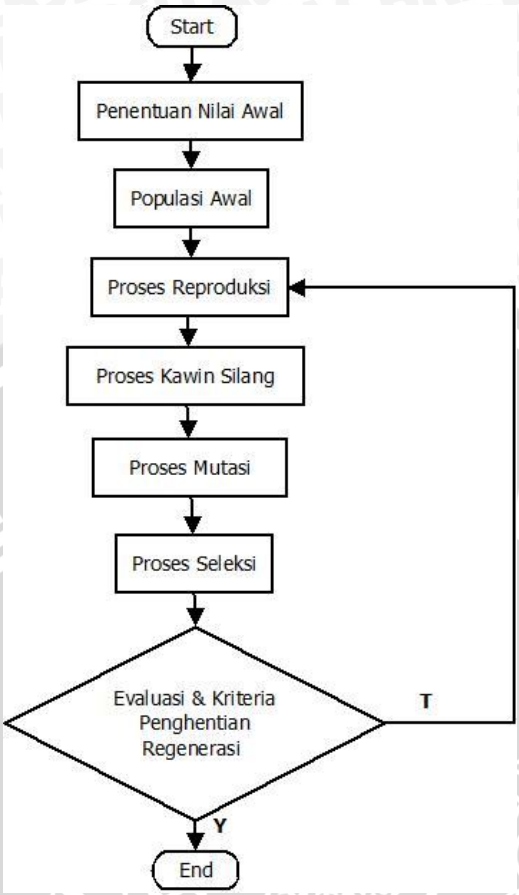
Adapun ciri – ciri bahwa suatu permasalahan dapat diselesikan dengan algoritma genetika adalah [BAS-03] :

- Memiliki fungsi tujuan optimalisasi non linear dengan banyak kendala yang juga non linear.
- Mempunyai kemungkinan solusi yang jumlahnya tak berhingga
- Membutuhkan solusi *real time*, maksudnya bahwa solusi bisa didapatkan secara cepat sehingga dapat diimplementasikan untuk permasalahan yang mempunyai perubahan cepat.
- Mempunyai *multi-objective* dan *multi-criteria*, sehingga diperlukan solusi yang dapat diterima oleh semua pihak.

Secara umum, algortima genetika memiliki lima komponen dasar, yaitu [GEN-00] :

1. Representasi kromosom untuk solusi permasalahan.
2. Bagaimana cara untuk membentuk populasi awal.
3. Evaluasi nilai *fitness*.
4. Operator genetika yang mengubah komposisi gen pada anak selama proses reproduksi.
5. Nilai parameter dari algortima genetika.

Secara umum, blok diagram dari mekanisme kerja algoritma genetika adalah sebagai berikut [HAN-02] :



Gambar 2.1 Mekanisme Kerja Algortima Genetika [HAN-02]

2.5 Pengkodean (Encoding)

Langkah awal dalam penerapan algortima genetika adalah menerjemahkan atau merepresentasikan permasalahan menjadi terminology biologi, atau biasa disebut kromosom. Dengan bahasa lain, cara untuk merepresentasikan masalah kedalam bentuk kromosom disebut pengkodean (*encoding*). Penentuan *encoding* bergantung pada bentuk permasalahan yang akan dipecahkan [SUY-05] .

Dalam bidang biologi satu individu(kromosom) memiliki dua jenis *entity* genetika, yaitu *phenotye* dan *genotype. Phenotype* merepresentasikan sifat nyata yang dapat dilihat secara langsung, sedangkan *genotype* merupakan bentuk representasi kode genetika suatu individu yang tidak dapat dilihat secara langsung. Hal ini mengilhami algoritma genetika sebagai *problem solver,* dimana dalam algoritma genetika pengkodean adalah merubah kemungkinan solusi dari suatu

permasalahan yang disebut *phenotypes* kedalam bentuk *genotypes* pada algoritma genetika yang diwakili oleh kode tertentu [EIB-03].

Menurut [OBI-98] yang dikutip dari [CIP-08], terdapat empat metode pengkodean yang biasa digunakan, yaitu :

2.4.1 Pengkodean Biner

Pengkodean biner merupakan metode pengkodean yang umum digunakan dalam merepresentasikan permasalahan dalam algortima genetika. Pada pengkodean biner, kromosom direpresentasikan dalam string bit 0 atau 1. Contoh pengkodean biner dtunjukkan pada Gambar 2.2.

Chromosome A :	0101101100010011
Chromosome B :	1011010110110101

Gambar 2.2 Contoh Representasi Kromosom Dengan Pengkodean Biner [CIP-08]

2.4.2 Pengkodean Permutasi

Pengkodean permutasi merupakan suatu pengkodean yang direpresentasikan dengan angka, dimana angka tersebut merepresentasikan sebuah urutan. Metode pengkodean ini biasa digunakan pada permasalahan kombinatorial, yang tidak dapat diselesaikan dengan representasi lain. Penentuan Rute merupakan salah satu contoh permasalahan yang dapat diseselesaikan dengan representasi ini, sehingga pengkodean digunakan kombinasi tempat dan urutan kunjungan. Salah satu contoh permasalahan adalah dalam penyelesaian *Traveling Salesman Problem* (TSP). Contoh pengkodean permutasi ditunjukkan pada Gambar 2.3.

Chromosome A :	8549102367
Chromosome B :	9102438576

Gambar 2.3 Contoh Representasi Kromosom Dengan Pengkodean Permutasi [CIP-08]

2.4.3 Pengkodean Nilai Langsung (*Real-value/ Floating*)

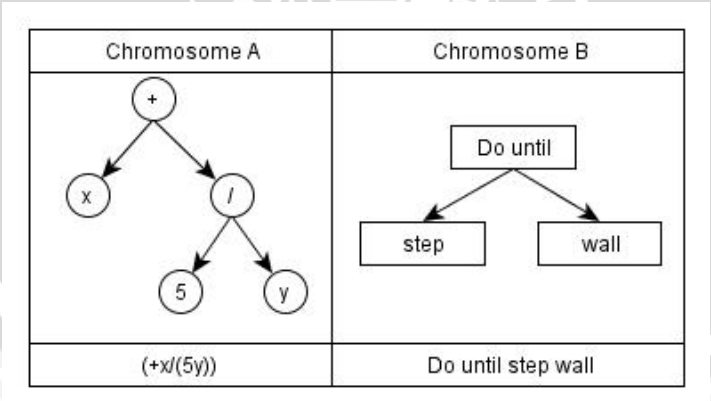
Pengkodean nilai merupakan suatu metode pengkodean dimana representasi kromosom dikodekan dengan berupa suatu nilai tertentu. Nilai dapat berupa apa saja seperti angka biasa, angka rill atau bentuk karakter dari suatu objek. Jenis metode ini dapat digunakan secara langsung dalam permsalahan dengan nilai yang rumit. Gambar 2.4 berikut mengilustrasikan contoh pengkodean nilai langsung.

Chromosome A :	[Red],[Yellow],[Black],[Red],[Blue],[Black]
Chromosome B :	1.8674, 3,4353, 9.7235, 1.3239, 2,1873, 4,8930
Chromosome C :	ABDWRNSALHWNWEAPMBF

Gambar 2.4 Contoh Representasi Kromosom Dengan Pengkodean Nilai [CIP-08]

2.4.4 Pengkodean Pohon (*tree*)

Pengkodean pohon merupakan jenis metode pengkodean yang biasa digunakan untuk menyusun program atau ekspresi didalam pemrograman genetika. Dalam pengkodean ini, setiap kromosom merupakan pohon dari beberapa objek, seperti fungsi atau perintah dalam bahasa pemrograman. Berikut adalah contoh pengkodean pohon yang diilustrasikan dalam Gambar 2.5.



Gambar 2.5 Contoh Representasi Kromosom Dengan Pengkodean Pohon [CIP-08]

Optimasi rute angkutan kota merupakan salah satu jenis permasalahan kombinatorial, dimana akan mencari suatu rute angkutan kota yang optimal. *Node* dalam representasi graf jalur angkutan akan dikodekan dalam *integer* angka sebagai *gen*. Angka – angka tersebut yang akan menjadi isi didalam kromosom, atau istilah yang biasa digunakan yaitu *gen*. Dimana angka – angka tersebut merupakan urutan rute – rute yang dilewati oleh angkutan kota. Sehingga digunakan metode pengkodean permutasi untuk menyusun rute- rute angkutan kota.

2.6 Fungsi *Fitness*

Fungsi fitness adalah fungsi yang dimiliki oleh masing – masing individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang ingin dicapai [KAR-11]. Suatu individu dievaluasi berdasarkan fungsi matematika tertentu sebagai ukuran performansinya. Didalam evolusi alam, individu yang bernilai *fitness* tinggi akan bertahan hidup, dan sebaliknya individu yang bernilai *fitness* rendah akan mati.

Pada permasalahan optimasi, fungsi *fitness* dapat berupa suatu solusi dengan memaksimalkan atau meminimalkan suatu fungsi *h*, kedua solusi tersebut adalah sama dan bersifat kebalikan. Namun dalam permasalahan dengan solusi meminimalkan fungsi *h*, fungsi *h* tidak dapat digunakan secara langsung sebagai fungsi *fitness*, sebab hal tersebut bertentangan dengan konsep evolusi. Dimana individu yang mampu bertahan hidup adalah individu yang memiliki nilai *fitness* yang tinggi. Sehingga digunakan suatu persamaan untuk mencari fungsi *fitness.*

$$f = \frac{1}{(h + a)} \tag{2.1}$$

Dimana a adalah bilangan yang sangat kecil dan bervariasi sesuai dengan permasalahan.

2.7 Seleksi

Proses seleksi merupakan proses pemilihan dua individu sebagai orang tua. Pada Proses seleksi, keanekaragaman populasi memegang peranan penting. Terdapat tiga hal penting yang harus diperhatikan dalam proses seleksi yaitu

daerah sampling, probabilitas seleksi dan mekanisme seleksi [SET-03]. Proses seleksi memiliki tujuan untuk memperoleh solusi terbaik dalam memperoleh individu yang paling fit. Untuk memperolehnya diperlukan dua langkah yaitu [WID-12] :

Seleksi populasi baru

Populasi baru merupakan kandidat solusi yang dipilih pada akhir setiap generasi, yang kemudian dijadikan sebagai populasi generasi berikutnya. Operator seleksi harus menjamin bahwa individu yang baik akan bertahan hidup pada generasi berikutnya.

a. Reproduksi

Offspring diciptakan lewat penerapan operator *crossover* (persilangan) atau mutasi. Dalam *crossover*, individu yang lebih unggul harus mempunyai peluang yang lebih besar unutk bereproduksi dan menjamin bahwa *offspring* berisi material genetika dari individu terbaik. Sedangkan dalam mutasi, mekanisme seleksi harus berfokus pada individu yang lemah, dengan harapan agar dapat menghasilkan sifat yang lebik baik sehingga menambah peluang untuk tetap bertahan.

Proses ini biasa dilakukan secara proporsional berdasarkan nilai – nilai *fitness*-nya. Terdapat berbagai macam metode seleksi antara lain : seleksi acak, seleksi proporsional, seleksi turnamen dan seleksi berbasis ranking [SUY-07]. Metode *rank base selection* digunakan dalam seleksi untuk menentukan popoulasi baru dalam generasi berikutnya.

2.6.1 Rank Base Selection

Rank base selection atau *rank base fitness* merupakan suatu metode seleksi dengan cara mengurutkan individu – individu berdasarkan nilai *fitness*-nya dari individu dengan nilai yang tinggi hingga individu dengan nilai *fitness* yang rendah. Individu yang menempati urutan atas maka akan dijadikan sebagai individu dalam populasi baru digenerasi berikutnya.

2.8 Operator Genetika

Setelah terbentuk individu sebagai induk atau orang tua maka individu tersebut berhak melakukan operasi genetika yang terdiri atas *crossover* dan *mutasi.*

2.7.1 Crossover

Proses operator genetika yang pertama yaitu *crossover* atau pindah silang. Proses *crossover* adalah proses memindah-silangkan dua buah kromosom. Proses *crossover* berfungsi untuk menghasilkan keturunan (anak) yang berasal dari dua induk yang telah terpilih, sehingga dapat dipastikan bahwa kromosom anak merupakan kombinasi gen-gen dari kedua induk.

Secara umum mekanisme *crossover* adalah sebagai berikut [SET-03]:

1. Menentukan induk dengan cara memilih dua buah kromosom.
2. Memilih posisi dalam kromosom yang akan dilakukan proses *crossover*, sehingga masing – masing kromosom induk terpecah menjadi dua segmen. Pemilihan posisi dilakukan secara acak, biasa disebut *crossover point.*
3. Melakukan penukaran antar segmen kromosom induk untuk menghasilkan kromosom anak.

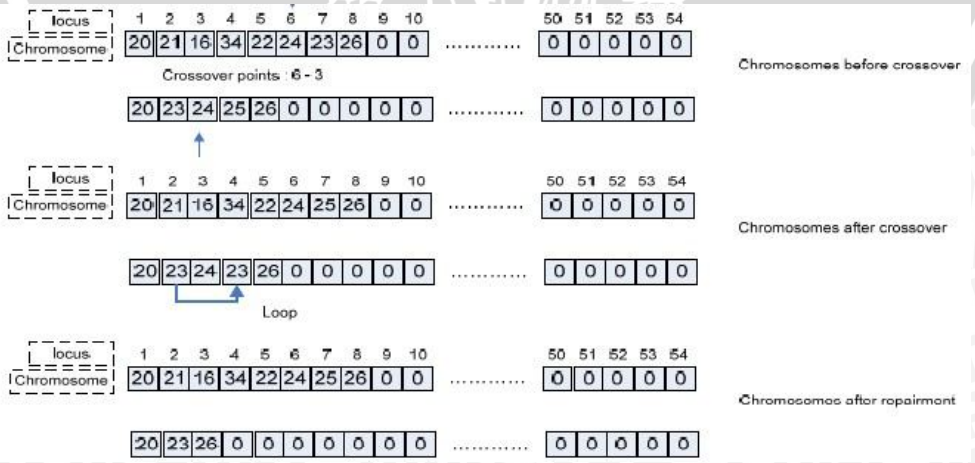
Dalam proses *crossover* jumlah populasi sangat mempengaruhi hasil dari proses *crossover*. Jumlah populasi yang sangat kecil akan berakibat buruk, hal itu akan menyebabkan suatu kromosom dengan gen – gen yang mengarah pada solusi akan sangat cepat menyebar ke kromosom – kromsom yang lain. Untuk mengatasi masalah tersebut digunakan suatu aturan bahwa proses *crossover* hanya dapat dilakukan dengan suatu probabilitas tertentu, artinya *crossover* hanya dapat dilakukan dengan membangkitkan bilangan *random* [0..1] dan nilai bangkitan *random* adalah kurang dari nilai p_c yang telah ditentukan. Pada umumnya, nilai p_c adalah mendekati 1 [SUY-05].

[HAN-02] mengartikan *crossover rate* atau probabilitas *crossover* sebagai suatu parameter penting, dimana *crossover rate* atau probabilitas *crossover* merupakan suatu nilai perbandingan jumlah kromosom dalam suatu populasi yang diharapkan akan mengalami *crossover*. *Crossover rate* yang tinggi akan

memungkinkan pencapaian alternatif solusi yang lebih bervariasi dan mengurangi kemungkinan menghasilkan nilai optimum yang tidak dikehendaki [HAN-02].

Terdapat banyak metode yang digunakan dalam proses *crossover* antara lain *single-point crossover, multi-point crossover, uniform crossover* dan *permutation crossover*. Metode *crossover* yang digunakan pada penelitian ini adalah modifikasi dari metode *crossover* yang digunakan dalam penelitian oleh [KAR-11] dan telah digunakan pula oleh [SAR-11] dalam penelitiannya. Ilustrasi metode *crossover* dapat dilihat pada gambar 2.6. Sedangkan langkah-langkah dari metode *crossover* tersebut adalah sebagai berikut [KAR-11] :

1. *Generate* bilangan *random* 0-1
2. Jika bilangan *random* < probabilitas *crossover*, lakukan langkah 4
3. Jika bilangan *random* > probabilitas *crossover*, lanjutkan ke pasangan induk berikutnya
4. Lakukan pencarian gen yang benilai (angka) sama dimulai dari index ke -2 pada pada masing – masing kromosom (1 dan 2), pilih sebagai titik *crossover*.
5. Tukar gen – gen antara kedua kromosom dimulai dari titik *crossover*.
6. Cek hasil *offspring*, jika ada yang tidak memenuhi aturan, ganti *gen* dengan 0.
7. Ganti *gen* yang bernilai 0 dengan *gen* pada *node* berikutnya yang memiliki nilai lebih besar.



Gambar 2.6 Ilustrasi Metode *Crossover* [KAR-11]

2.7.2 Mutasi

Proses operator genetika selanjutnya setelah *crossover* yaitu mutasi. Mutasi merupakan suatu proses untuk menghasilkan perubahan acak pada suatu kromosom. Proses mutasi berfungsi untuk menghasilkan individu baru yang memiliki sifat baru dengan cara melakukan perubahan pada sebuah gen atau lebih dari suatu individu. Proses mutasi memiliki tujuan agar individu – individu yang ada dalam populasi semakin bervariasi. Dengan begitu dapat mempertahankan keanekaragaman individu dalam suatu populasi [SET-03].

Cara mudah untuk melakukan proses mutasi yaitu dengan mengubah satu atau lebih gen dalam kromosom dengan bergantung pada *mutation rate*. *Mutation rate* merupakan suatu nilai yang menentukan probabilitas atau peluang jumlah gen dalam suatu populasi yang diharapkan akan mengalami mutasi. Tentunya nilai *mutation rate* berpengaruh terhadap banyaknya gen baru yang akan dimunculkan dalam evaluasi dan hal tersebut juga mempengaruhi hasil dari mutasi. Apabila nilai *mutation rate* terlalu kecil maka banyak gen – gen yang mungkin berguna tidak akan muncul untuk dievaluasi. Sedangkan bila nilai *mutation rate* terlalu besar tinggi dapat mengakibatkan gangguan acak, sehingga anak yang dihasilkan akan kehilangan sifat asli dari induk [HAN-02].

Proses mutasi dalam penerapan algortima genetika pada penelitian ini menggunakan metode mutasi *random* dengan memodifikasi dari penelitian yang dilakukan oleh [PRA-11]. Adapun langkah-langkah dari metode mutasi dalam penelitian sebelumnya adalah sebagai berikut :

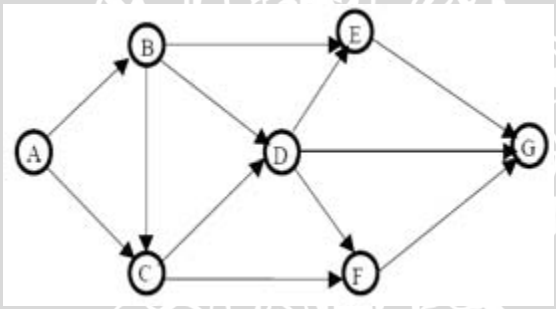
1. Membangkitkan bilangan *random r* [0..1] sebanyak jumlah individu.
2. Apabila bilangan *random* $r < pm$, maka individu tersebut terpilih untuk dikenai mutasi.
3. Memilih gen yang akan dimutasi (k) secara *random.*
4. Membangkitkan bilangan *random* antara 1 hingga jumlah bahan pakan sebanyak 1 bilangan (m) dengan ketentuan bilangan (m) merupakan bilangan yang belum pernah terpakai pada populasi awal.
5. Mengganti nilai pada gen k dengan nilai m.

2.9 Graf

Graf adalah kumpulan sejumlah berhingga obyek yang disebut simpul (*vertices, vertex*) yang dihubungkan dengan sisi atau busur (*edges*) [ADI-12]. Suatu graf G terdiri atas dua himpunan yaitu himpunan V sebagai simbol simpul dan himpunan E sebagai simbol busur [SAP-07].

Suatu busur dapat menunjukkan berbagai macam relasi antara lain jalan raya, rute penerbangan, ikatan kimia dan sebagainya. Notasi dari sebuah graf adalah G =(V,E) artinya bahwa sebuah graf G memiliki simpul V dan busur E. Dalam bahasa matematika notasi G =(V,E) berarti :

- V merupakan himpunan tak kosong dari simpul – simpul (*vertices*), misal $V = \{v_1, v_2, v_3, \dots, v_n\}$
- E merupakan himpunan sisi – sisi (*edges*) yang menghubungkan dua buah simpul, misal missal $E = \{e_1, e_2, e_3, \dots, e_n\}$



Gambar 2.7 Contoh Graf ABCDEFG
[SAP-07]

Menurut arah dan bobotnya suatu graf dibedakan menjadi 4 bagian, yaitu :

a. Graf berarah dan berbobot : suatu graf yang tiap sisi atau busurnya memiliki arah yang ditunjukkan dengan anak panah dan memiliki bobot.
b. Graf tidak berarah dan berbobot : suatu graf yang tiap sisi atau busurnya tidak memiliki arah (anak panah) tetapi memiliki bobot.
c. Graf berarah dan tidak berbobot : suatu graf yang tiap sisi atau busurnya memiliki arah yang ditunjukkan dengan anak panah tetapi tidak memiliki bobot.
d. Graf tidak berarah dan tidak berbobot : suatu graf yang tiap sisi atau busurnya tidak memiliki arah dan tidak memiliki bobot.

Representasi dari suatu graf dapat memiliki berbagai macam bentuk dimana bentuk tersebut adalah merepresentasikan dari suatu kasus. Tentunya dengan representasi kasus kedalam suatu graf memiliki bentuk yang berbeda – beda. Beberapa representasi garf yang biasa digunakan yaitu :

a. Matriks Kedekatan (*Adjacency Matrix*)

Suatu graf dengan jumlah simpul sebanyak n, maka matriks kedekatannya memiliki ukuran n x n (n baris dan n kolom). Matriks kedekatan untuk contoh graf ABCDEFG diatas (gambar 2.7) adalah :

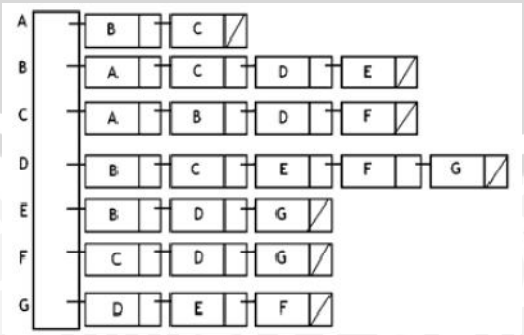
	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0	1	1	0	0	0	0
B	1	0	1	1	1	0	0
C	1	1	0	1	0	1	0
D	0	1	1	0	1	1	1
E	0	1	0	1	0	0	1
F	0	0	1	1	0	0	1
G	0	0	0	1	1	1	0

Gambar 2.8 Matriks Kedekatan Contoh Graf ABCDEFG

[SAP-07]

b. Senarai Kedekatan (*Adjacency List*)

Suatu graf yang mencari kedekatan dari suatu simpul x dengan simpul yang lain. Simpul x dapat dianggap sebagai senarai yang terdiri atas simpul pada graf. Berikut adalah senarai kedekatan dari contoh graf ABCDEFG (gambar 2.9).

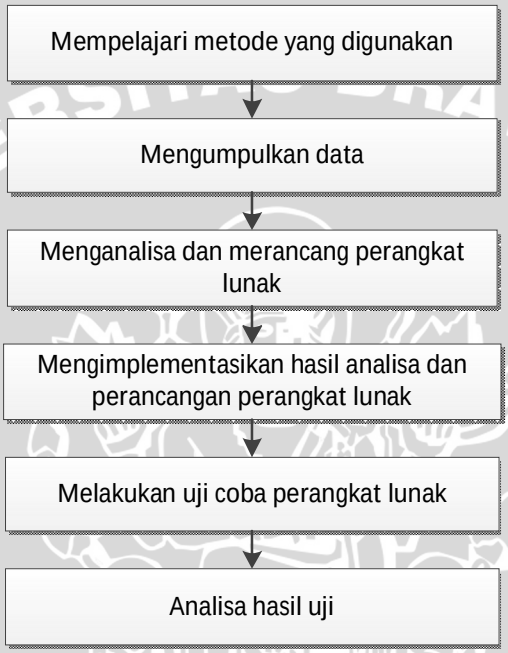


Gambar 2.9 Senarai Kedekatan Contoh Graf ABCDEFG

[SAP-07]

BAB III
METODOLOGI DAN PERANCANGAN

Dalam bab ini akan dijabarkan mengenai metodologi dan perancangan perangkat lunak optimasi rute angkutan kota dengan penerapan algortima genetika. Berikut adalah langkah - langkah yang dilakukan dalam penelitian skripsi ini, yang diilustrasikan pada gambar 3.1.



Gambar 3.1 Langkah – Langkah Penelitian

Berdasarkan gambar 3.1, penjabaran langkah - langkah yang dilakukan dalam penelitian skripsi ini adalah :

1. Mempelajari metode yang akan digunakan dalam proses penerapan algoritma genetika
2. Mengumpulkan data – data yang dibutuhkan yaitu data jalan dan data jaringan trayek angkutan kota Malang.
3. Menganalisis dan merancang perangkat lunak optimasi rute angkutan kota Malang dengan penerapan algoritma genetika.
4. Mengimplementasikan perangkat lunak berdasarkan analisa dan perancangan yang telah dilakukan.

5. Melakukan uji coba perangkat lunak yang telah dibuat
6. Menganalisa hasil uji dari implementasi perangkat lunak.

3.1 Analisa Perangkat Lunak

3.1.1 Deskripsi Umum

Secara umum perangkat lunak yang akan dibangun dalam penelitian ini adalah sebuah perangkat lunak yang mengimplementasikan algoritma genetika dalam permasalahan optimasi rute angkutan kota Malang. Perangkat lunak ini bertujuan untuk menentukan rute angkutan kota yang optimal dari segi jarak tempuh dan *intersection* jalur yaitu dengan meminimalkan jarak tempuh dan meminimalkan jumlah tumpukan (*intersection*) jalur yang lewati oleh trayek angkutan kota.

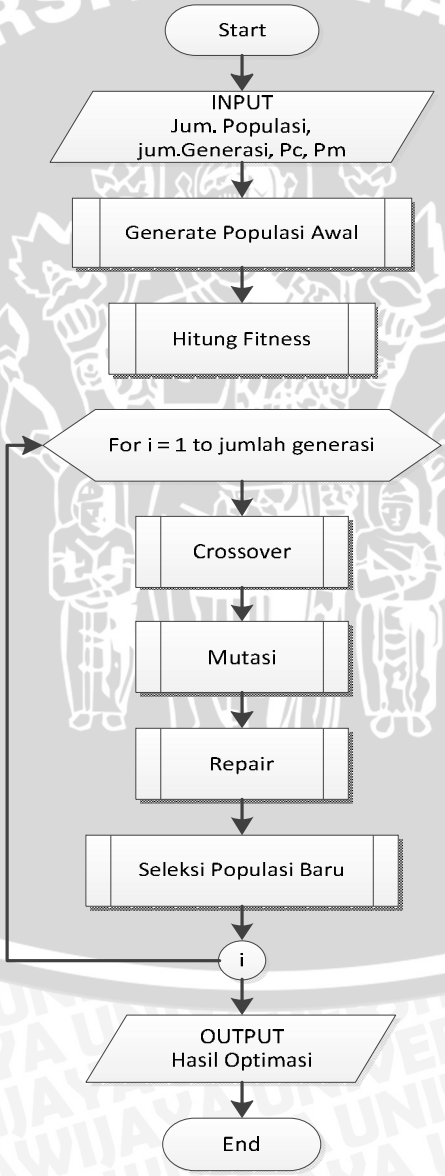
3.1.2 Deskripsi Data

Pada penelitian ini, data yang digunakan sebagai objek penelitian adalah data jalan dan peta jalan Kota Malang serta data jaringan beserta peta trayek angkutan kota Malang. Adapun data jaringan trayek angkutan kota Malang didapat dari Dinas Perhubungan Kota Malang. Sedangkan untuk peta jalan kota Malang menggunakan Peta Kota Malang dan Perkembangannya yang diterbitkan dan dicetak oleh PT.Karya Pembina Swajaya-Surabaya pada tahun 2011, serta didukung dengan peta yang diambil melalui *googlemaps*. Dan data jalan kota Malang menggunakan data jalan kota Malang tahun 2009 yang didapatkan dari dinas perhubungan kota Malang dan ditambah dari hasil pengukuran melalui skala peta tersebut diatas.

Data yang telah terkumpul selanjutnya akan dilakukan pengolahan sehingga data siap untuk diproses dengan menerapkan algoritma genetika. Untuk memperoleh data yang siap diolah maka data yang diperoleh terlebih dahulu dikodekan. Setelah semua data dikodekan maka data tersebut diimplementasikan menggunakan algoritma genetika.

3.2 Perancangan Perangkat Lunak

Berdasar analisis yang telah dilakukan, maka dalam subbab ini akan dibahas mengenai proses penerapan algoritma genetika dalam perangkat lunak. Dalam permasalahan ini perangkat lunak akan menerima masukan berupa parameter genetika yang terdiri atas jumlah populasi, jumlah generasi, probabilitas *crossover (pc)* dan probabilitas mutasi *(pm)*. Selanjutnya dilakukan tahapan – tahapan algortima genetika yang terdiri atas *generate* populasi awal, hitung nilai *fitness*, *crossover*, mutasi, *repair* dan seleksi populasi baru. *Flowchart* untuk proses algoritma genetika yang digunakan dapat dilihat pada gambar 3.2



Gambar 3.2 *Flowchart* Algortima Genetika Pada Optimasi Rute kota Malang

3.2.1 Proses Algoritma Genetika

Dari *flowchart* pada gambar 3.2, dapat dijelaskan langkah – langkah proses algoritma genetika dalam penerapannya kedalam optimasi rute angkutan kota Malang, adalah sebagai berikut :

1. *Inisialisasi* parameter awal, yaitu :
    1. Parameter algoritma genetika, antara lain:
        1. Jumlah populasi
        2. Jumlah generasi
        3. Probabilitas *crossover*
        4. Probabilitas mutasi
2. *Generate* populasi awal yang terdiri atas beberapa individu sejumlah yang telah ditentukan melalui *input.*
3. Menghitung nilai *fitness* dari setiap individu dalam populasi.
4. Melakukan proses *crossover* dengan metode yang telah ditentukan dan dengan berdasarkan pada nilai probabilitas *crossover* (pc) yang telah ditentukan pula.
5. Melakukan proses mutasi dengan metode mutasi *random* dan dengan berdasarkan pada probabilitas mutasi (pm) yang telah ditentukan.
6. Melakukan proses *repair* terhadap individu *invalid*.
7. Menentukan populasi baru untuk proses generasi selanjutnya dengan cara melakukan seleksi dengan metode *rank base selection* (seleksi rangking).

Secara lebih rinci, langkah – langkah proses algoritma genetika dalam penerapannya kedalam optimasi rute angkutan kota Malang dijabarkan pada sub bab 3.2.1.1 hingga sub bab 3.2.1.7, sebagai berikut :

3.2.1.1 Representasi Individu

Individu merupakan kromosom yang terdiri atas bagian tekecil yaitu *gen*. *Gen – gen* tersebut merupakan bentuk pengkodean yang merepresentasikan permasalahan optimasi rute angkutan kota Malang. Dalam hal ini individu merupakan jaringan trayek angkutan kota Malang.

Metode pengkodean yang digunakan dalam penelitian ini adalah metode pengkodean permutasi. *Gen* dalam individu merupakan representasi sebuah *node* dalam graf yang telah ditentukan. Setiap *gen* direpresentasikan dalam *type* data *integer*. Panjang dari trayek adalah dinamis, dengan berdasarkan pada jalur atau rute yang telah terpilih dari hasil *random*.

Pada penelitian ini, 1 individu merupakan satu solusi, dimana suatu solusi merupakan representasi dari keseluruhan trayek angkutan kota Malang, yang terdiri atas 25 trayek. Dalam individu mengandung *gen* – *gen* yang merepresentasikan rute setiap trayek angkutan kota.

Setiap individu akan direpresentasikan menggunakan *array* yang didalamnya memuat *arraylist*. Dimana dalam *arraylist* tersebut memuat *gen* - *gen* dengan panjang yang berbeda (dinamis), disesuaikan dengan rute yang telah terpilih dari hasil *random*. Ilustrasi representasi 1 individu digambarkan pada gambar 3.3.

K1	**S1**	**N_1**	**N_2**	**…**				**E1**	
K2	**S2**	**N_3**	**…**			**E2**			
K3	**S3**	**N_4**	**…**				**E3**		
K4	**S4**	**…**				**E4**			
…	**S…**	**…**					**E…**		
…	**S…**								**E…**
…	**S…**							**E…**	
K25	**S25**						**E25**		

Gambar 3.3 Ilustrasi Representasi 1 Individu

Keterangan Gambar:

S = *Start point* (Lokasi Awal)

E = *End point* (Lokasi Tujuan)

N = *Gen* / *Node* (Persimpangan Jalan)

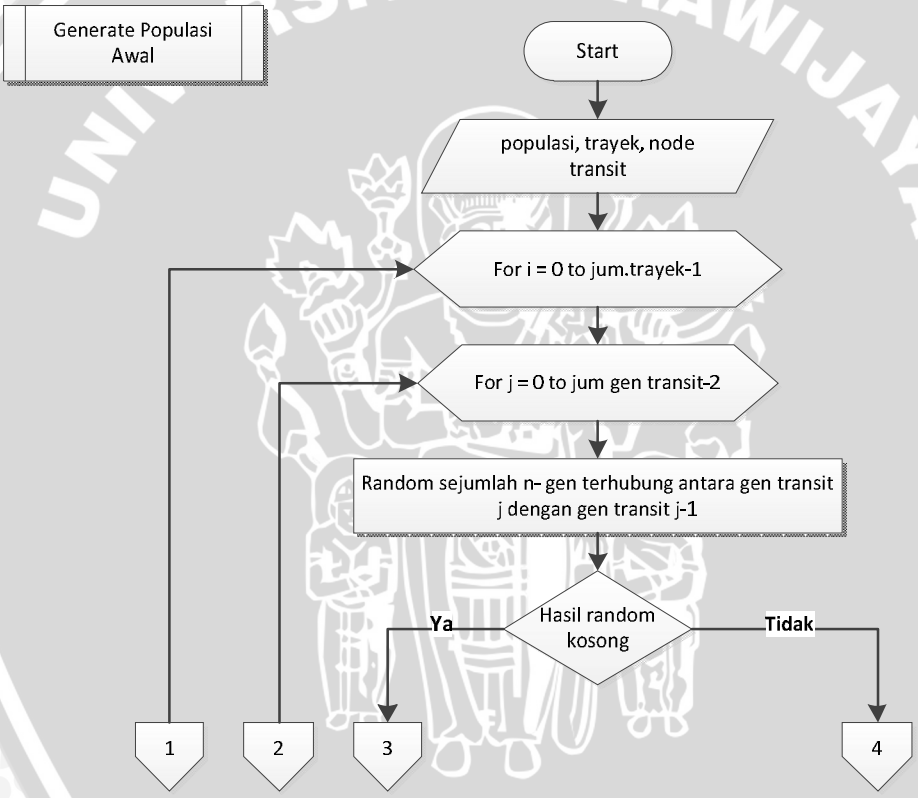
K = Trayek

3.2.1.2 Generate Populasi Awal

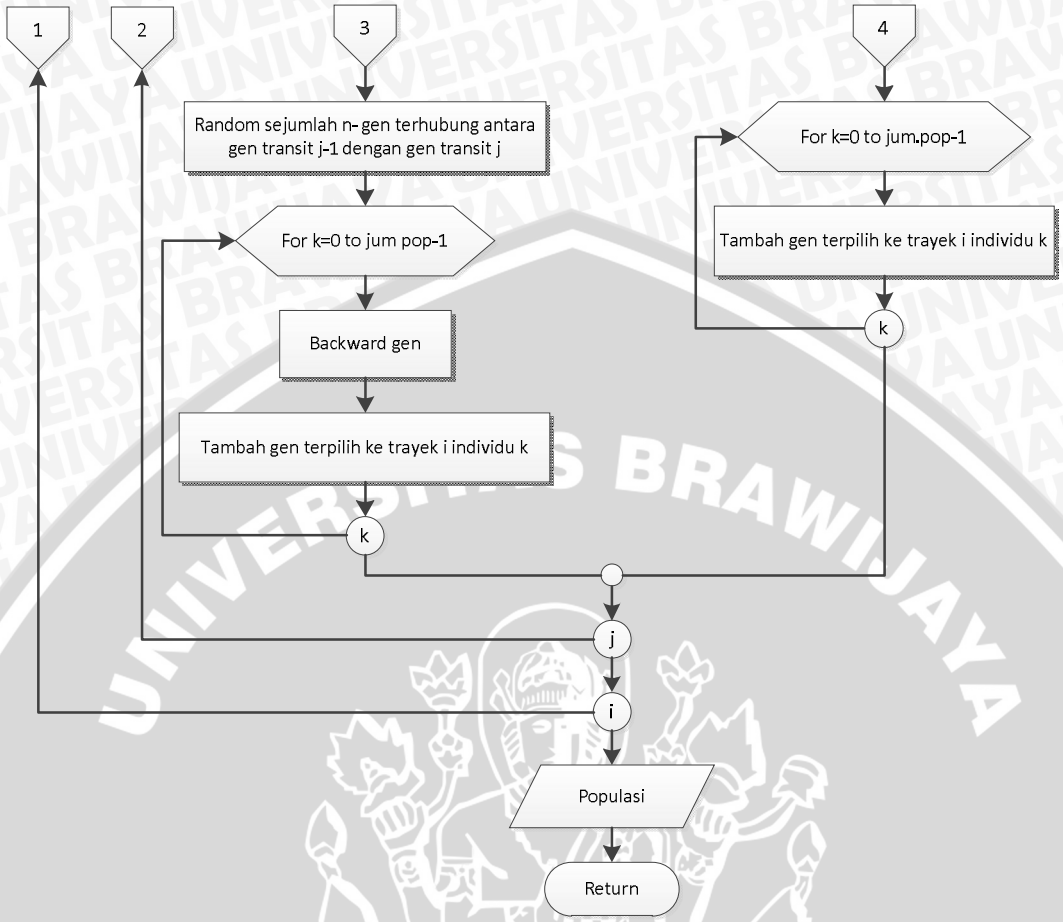
Proses *generate* populasi awal merupakan proses pembentukan individu sebanyak jumlah populasi awal yang telah ditentukan. Dalam penelitian ini 1 solusi adalah 1 individu, dimana 1 individu terdiri atas *gen* – *gen* yang

merepresentasikan rute trayek angkutan kota. Jumlah populasi individu akan ditentukan oleh *user* melalui *input* yang diinginkan.

Didalam proses *generate* populasi awal akan diikuti pula pembentukan individu. Individu yang terbentuk dalam proses g*enerate* populasi dilakukan melalui mekanisme *random* dengan terlebih dahulu ditetapkan beberapa *gen transit*. *Random* dilakukan pada setiap *gen transit* hingga membentuk suatu rute trayek yaitu telah memenuhi *node* tujuan. *Flowchart* untuk proses *generate* populasi awal ditampilkan dalam gambar 3.4 dan gambar 3.5.



Gambar 3.4 *Flowchart Generate* Populasi Awal (1)



Gambar 3.5 *Flowchart Generate* Populasi Awal (2)

3.2.1.3 Perhitungan Nilai Fitness

Fungsi *fitness* dalam penelitian skripsi ini adalah fungsi *fitness* untuk menentukan rute trayek angkutan kota Malang dengan meminimalkan jarak tempuh dan meminimalkan jumlah tumpukan jalur. Dalam permasalahan ini suatu nilai *fitness* yang baik dan dianggap sebagai solusi adalah nilai *fitness* yang besar.

Perhitungan nilai *fitness* dilakukan dengan memberikan nilai penalti untuk setiap aturan yang digunakan dalam optimasi rute angkutan kota Malang. Aturan yang memiliki prioritas tinggi akan diberikan nilai penalti pelanggaran yang besar, dan sebaliknya. Oleh karena itu, apabila suatu individu memiliki nilai penalti yang besar maka individu tersebut tidak layak ditetapkan sebagai solusi sebab individu tersebut memiliki nilai *fitness* yang kecil.

Persamaan fungsi *fitness* yang digunakan adalah modifikasi fungsi *fitness* dari persamaan 2.1 dengan menyesuaikan bentuk permasalahan skripsi. Adapun fungsi *fitness* yang dibentuk gunakan dinyatakan dalam rumus :

$$F = \frac{1}{(\sum_{i=1}^{3}(p_i \times f)) + 1} \quad (3.1)$$

Keterangan :

F = nilai *fitness*

p_i = penalti pelanggaran yang diberikan pada aturan ke-i

f = frekuensi pelanggaran pada aturan ke-i

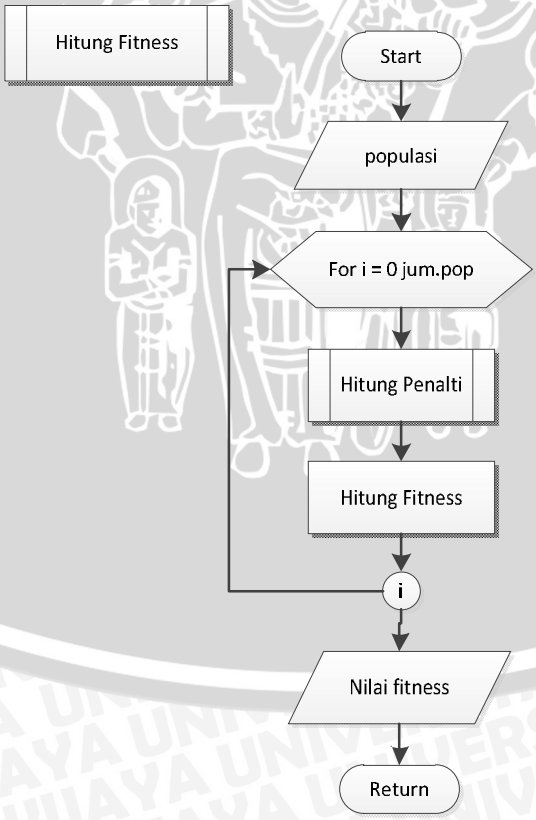
Adapun aturan dan nilai yang diberikan dalam permasalahan ini dijabarkan dalam tabel 3.1

Tabel 3.1 Aturan dan Nilai Penalti

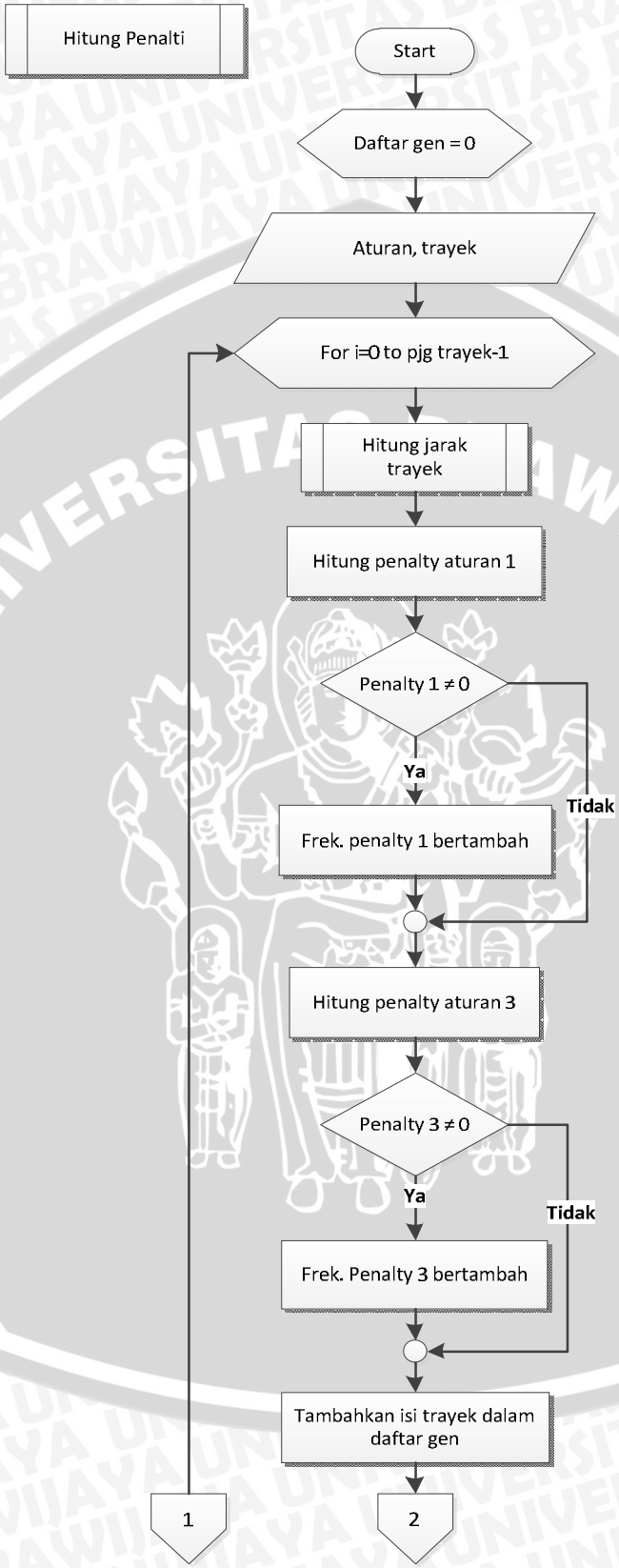
No.	Aturan	Penalti
1.	Jarak tempuh trayek AG/AH > 16 km	10
	Jarak tempuh trayek AL > 17.8 km	
	Jarak tempuh trayek AT > 18 km	
	Jarak tempuh trayek ABB > 16 km	
	Jarak tempuh trayek ABG/ABH > 26 km	
	Jarak tempuh trayek ADL > 14.8 km	
	Jarak tempuh trayek AJG/AJH > 18.2 km	
	Jarak tempuh trayek AMG/AMH > 16 km	
	Jarak tempuh trayek ASD > 11.5 km	
	Jarak tempuh trayek CKL > 22 km	
	Jarak tempuh trayek GA/HA > 16.3 km	
	Jarak tempuh trayek GL/HL > 15.8 km	
	Jarak tempuh trayek GM/HM > 9 km	
	Jarak tempuh trayek GML/HML > 18 km	
	Jarak tempuh trayek JDM > 13 km	
	Jarak tempuh trayek JPK > 10 km	
	Jarak tempuh trayek LG/LH > 16.8 km	
	Jarak tempuh trayek LDG/LDH > 15.2 km	
	Jarak tempuh trayek MK > 11.5 km	
	Jarak tempuh trayek MM > 15.2 km	

	Jarak tempuh trayek MT > 9 km	
	Jarak tempuh trayek MKS > 8 km	
	Jarak tempuh trayek TGT > 6.5 km	
	Jarak tempuh trayek TSG > 10 km	
	Jarak tempuh trayek TST > 16 km	
2.	Suatu jalur (*gen* terhubung) dilewati lebih dari 3 trayek	10
3.	Jarak tempuh tiap trayek angkutan < 5 km	5

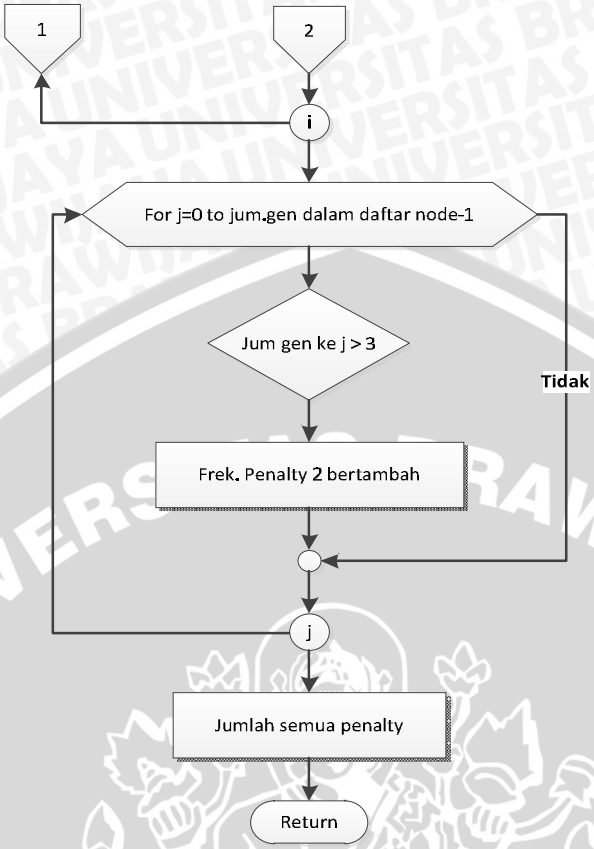
Proses perhitungan nilai *fitness* dimulai dengan melakukan perhitungan nilai penalti. Dimana perhitungan nilai penalti dilakukan sesuai jumlah aturan yaitu 3. Untuk perhitungan penalti aturan 1 dan 3 terlebih dahulu akan dilakukan perhitungan jarak tempuh pada setiap trayek. *Flowchart* proses hitung nilai *fitness*, hitung penalti dan hitung jarak trayek dapat dilihat pada gambar 3.6, gambar 3.7 – gambar 3.8 dan gambar 3.9.



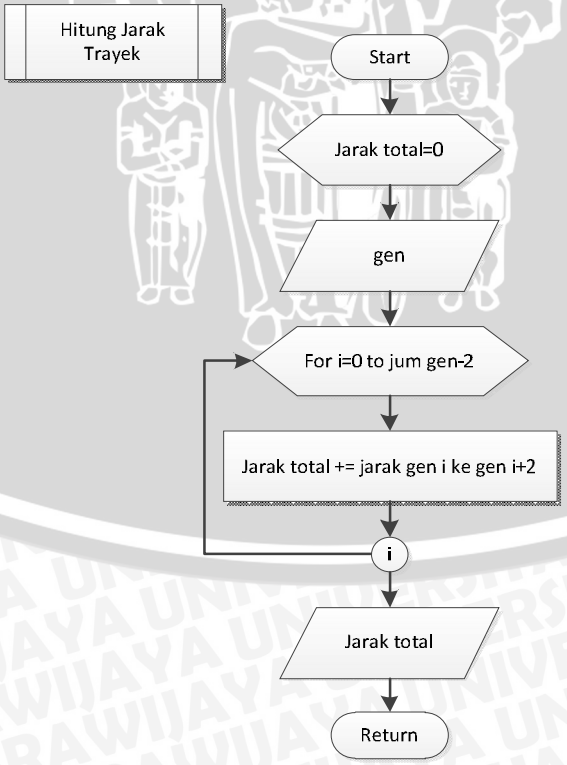
Gambar 3.6 *Flowchart* Hitung Fitness



Gambar 3.7 *Flowchart* Hitung Penalti (1)



Gambar 3.8 *Flowchart* Hitung Penalti (2)



Gambar 3.9 *Flowchart* Hitung Jarak Trayek

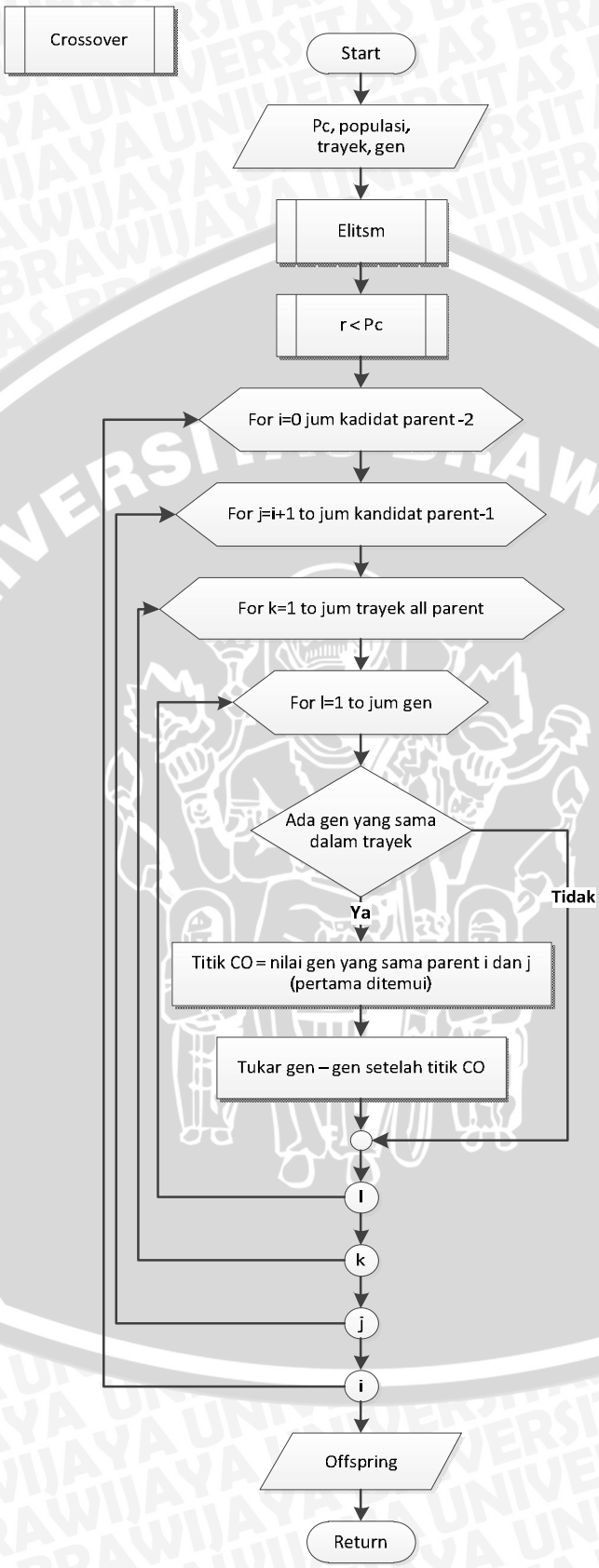
3.2.1.4 Proses Crossover

Proses berikutnya setelah menghitung nilai *fitness* adalah proses *crossover*. Proses *crossover* merupakan proses penyilangan 2 atau lebih individu (*parent*) untuk mendapatkan suatu keturunan yang disebut *child* atau *offspring*. Metode *crossover* yang digunakan adalah modifikasi dari metode yang telah dijelaskan dalam bab II.

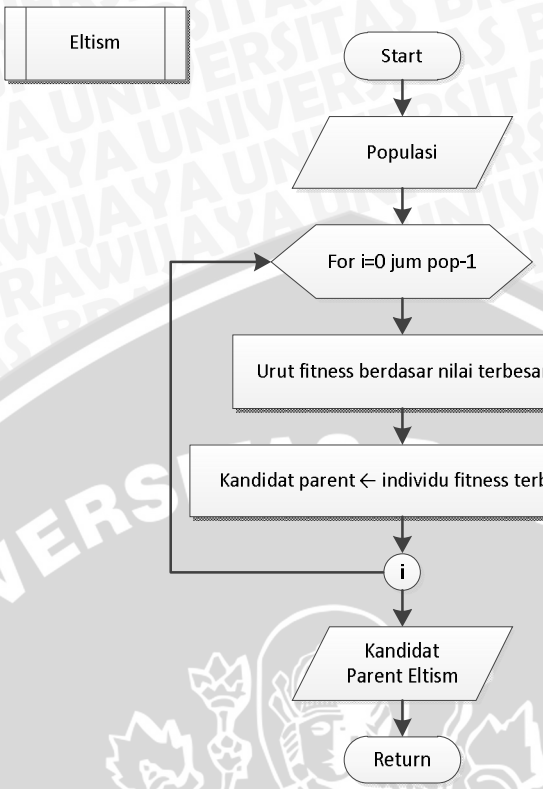
Pada penelitian ini, proses *crossover* dilakukan dengan mengkombinasikan persilangan antar kandidat *parent*. Sebelum dilakukan proses *crossver* terlebih dahulu ditentukan kandidat *parent* yang dilakukan dengan 2 teknik yaitu teknik *elitsm*, untuk tetap mempertahankan individu dengan nilai *fitness* yang baik dan berdasarkan nilai probabilitas *crossover* (pc). Kandidat *parent* dengan teknik *elitsm* adalah mengambil individu dengan nilai *fitness* terbesar. Sedangkan kandidat *parent* dengan berdasarkan nilai probabilitas *crossover* (pc) adalah mengambil individu – individu dengan nilai *random* r < probabilitas *crossover* (pc). Berikut adalah proses *crossover* dengan mengadopsi langkah – langkah *crossover* yang telah dijelaskan dalam bab 2 :

1. Melakukan pencarian *gen – gen* yang sama dalam trayek, mulai dari trayek ke-1.
2. Menentukan titik *crossover* dengan cara mengambil *gen* yang sama dan pertama kali ditemukan dalam trayek pada *parent* 1 dan 2 kecuali gen pertama dan terakhir.
3. Apabila tidak terdapat *gen* yang sama dalam trayek ke-1 pada *parent* 1 dan 2, lanjutkan pada trayek berikutnya dan ulangi mulai langkah 1.
4. Menukarkan *gen - gen* antara 2 trayek mulai dari titik *crossover*.
5. Mengulangi langkah 1 – 4 hingga memenuhi trayek ke-25.

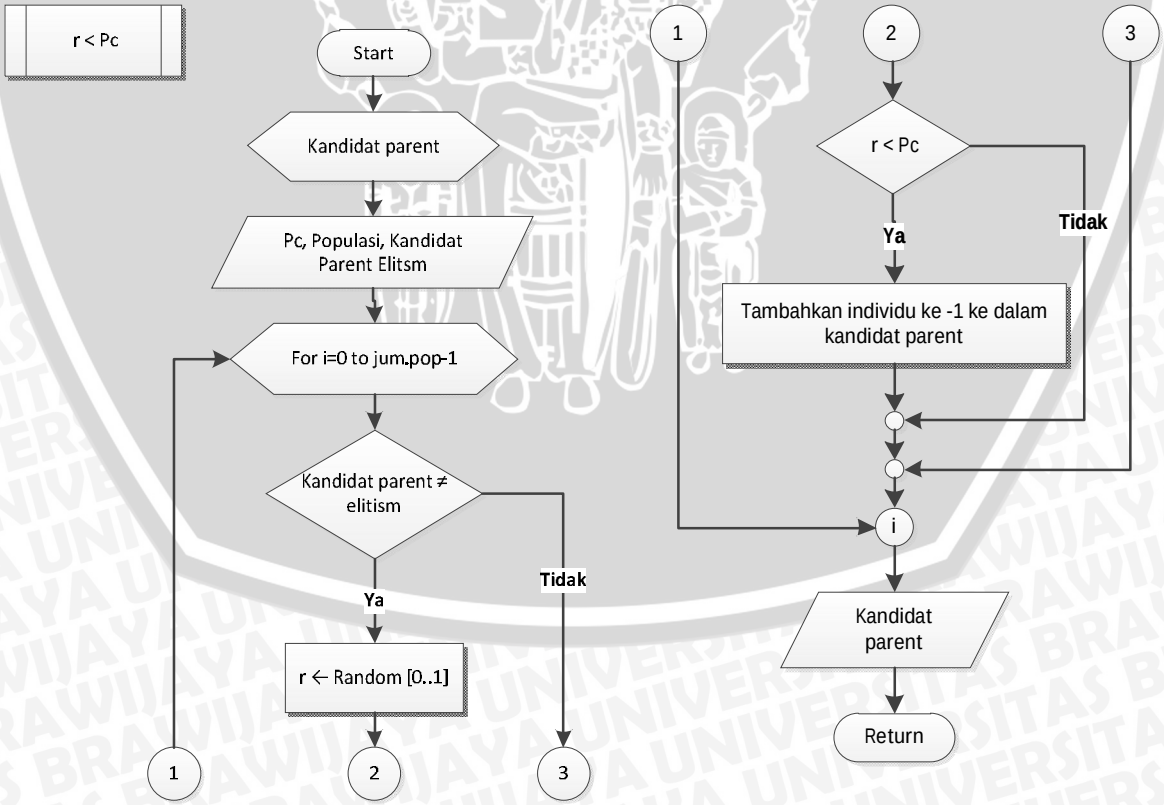
Flowchart proses *crossover* dan proses pemilihan kandidat *parent* dapat dilihat pada gambar 3.10 hingga gambar 3.12



Gambar 3.10 *Flowchart* Proses *Crossover*



Gambar 3.11 *Flowchart* Penentuan Kandidat *Parent (Elitsm)*



Gambar 3.12 *Flowchart* Penentuan Kandidat *Parent* (r < Pc)

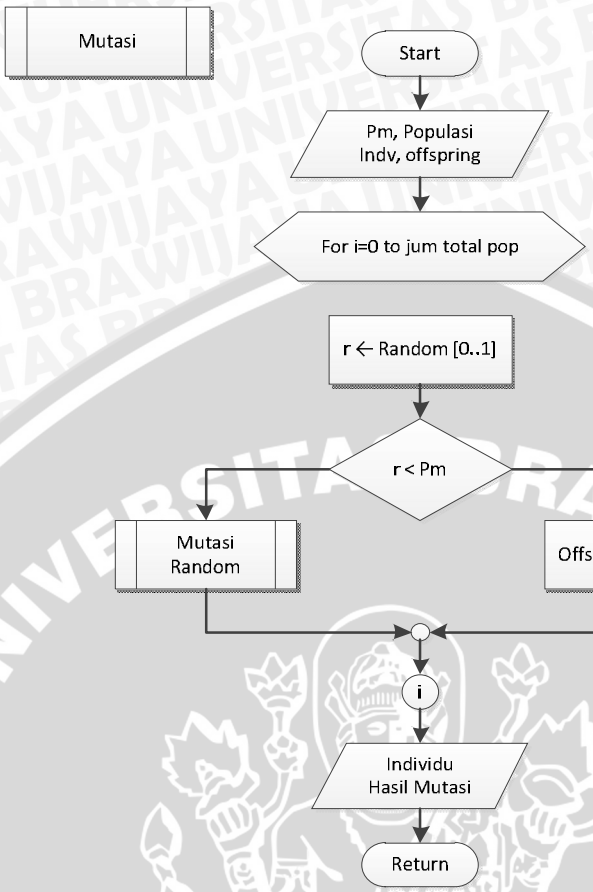
3.2.1.5 Proses Mutasi

Proses selanjutnya adalah proses mutasi. Proses dilakukan dengan tujuan untuk meningkatkan variasi individu dalam populasi atau mencegah konvergensi dini. Proses mutasi didasarkan pada nilai probabilitas mutasi (pm) sebagai nilai peluang besarnya individu yang akan mengalami mutasi.

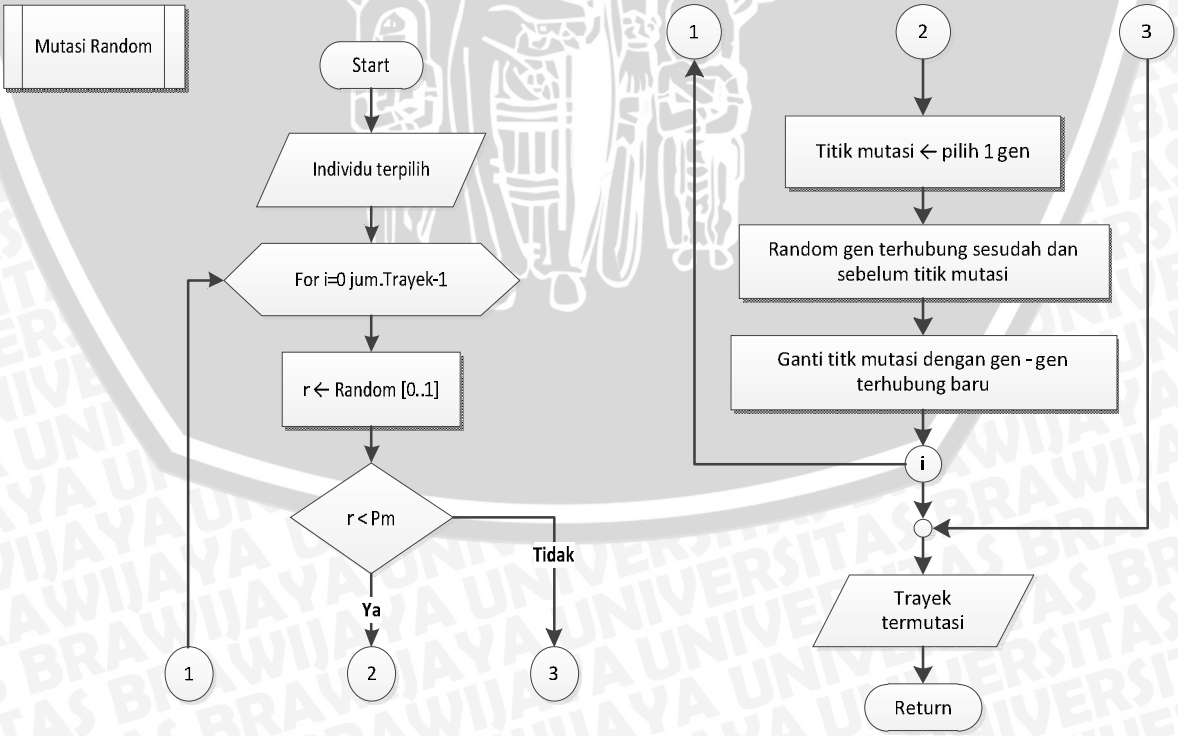
Proses mutasi yang digunakan adalah mutasi *random* yang merupakan hasil modifikasi dari proses mutasi yang dijelaskan dalam bab 2. Adapun langkah-langkah dari proses mutasi yang digunakan adalah sebagai berikut :

1. *Generate* bilangan *random* [0..1] sebanyak jumlah individu popolusi awal ditambah *child* atau *offspring* untuk menentukan individu yang akan dimutasi.
2. Apabila hasil *generate* bilangan *random* < probabilitas mutasi (pm), maka lakukan langkah 4.
3. Apabila hasil *generate* bilangan *random* > probabilitas mutasi (pm), maka tidak terjadi mutasi dan keturunan sama dengan induk.
4. *Generate* bilangan *random* [0..1] sebanyak jumlah total trayek dari individu terpilih untuk menentukan trayek yang akan dimutasi.
5. Menentukan titik mutasi secara *random* gen, pada trayek terpilih kecuali pada *gen* pertama dan terakhir.
6. Menentukan *gen* – *gen* yang terhubung dengan *gen* sebelum dan sesudah titik mutasi.
7. Memilih *gen* – *gen* yang terhubung.
8. Mengganti nilai *gen* pada titik mutasi dengan *gen* – *gen* baru yang terhubung

Flowchart proses mutasi dapat dilihat pada gambar 3.13 dan gambar 3.14.



Gambar 3.13 *Flowchart* Proses Mutasi



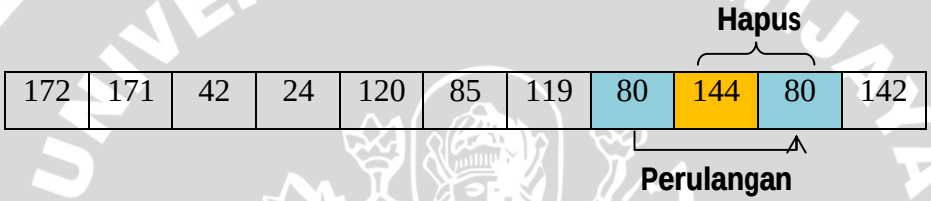
Gambar 3.14 *Flowchart* Proses Mutasi Random

3.2.1.6 Repair

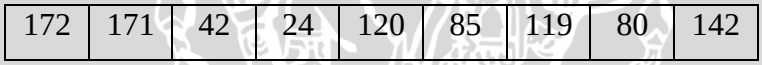
Proses *repair* merupakan proses untuk memperbaiki individu sehingga mendapatkan individu yang valid dan siap untuk diseleksi guna mendapatkan populasi generasi baru. Individu yang dianggap valid adalah apabila telah memenuhi kriteria bahwa tidak ada *gen* yang berulang dalam 1 trayek.

Untuk dapat memenuhi kriteria valid diatas maka dilakukan proses *repair*. Proses *repair* dilakukan dengan cara menghapus *gen* – *gen* yang berada pada perulangan. Berikut adalah ilustrasi gambar dari proses *repair* :

Sebelum *repair*



Sesudah *repair*

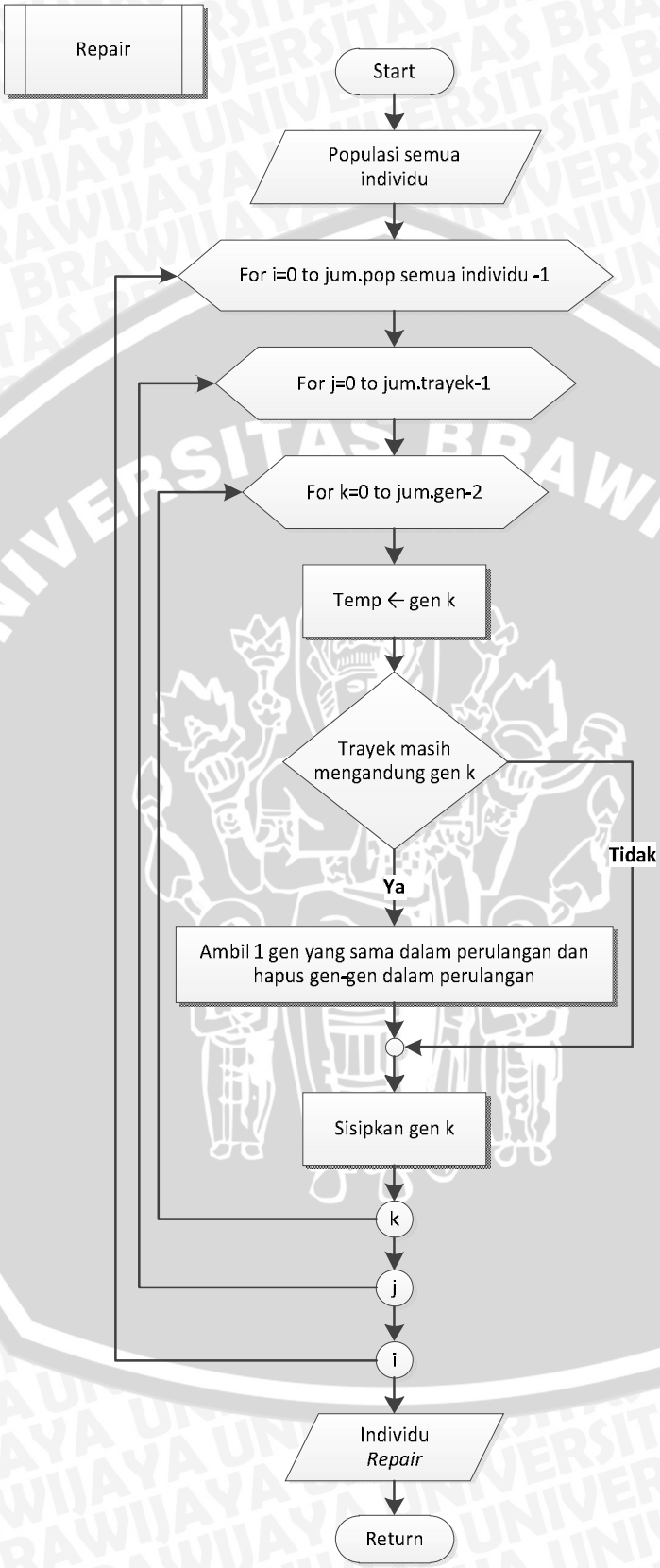


Gambar 3.15 Ilustrasi Proses *Repair*

Adapun langkah – langkah dari proses *repair* adalah sebagai berikut :

1. Mengambil 1 gen ke k dan menyimpannya sementara
2. Melakukan pengecekan terhadap trayek apakah mengandung *gen* k.
3. Apabila masih mengandung gen k maka hanya 1 *gen* yang diambil dan menghapus *gen* – *gen* dalam perulangan dan menyisipkan kembali *gen* k.
4. Apabila tidak mengandung *gen* k maka *gen* k langsung disisipkan kembali.

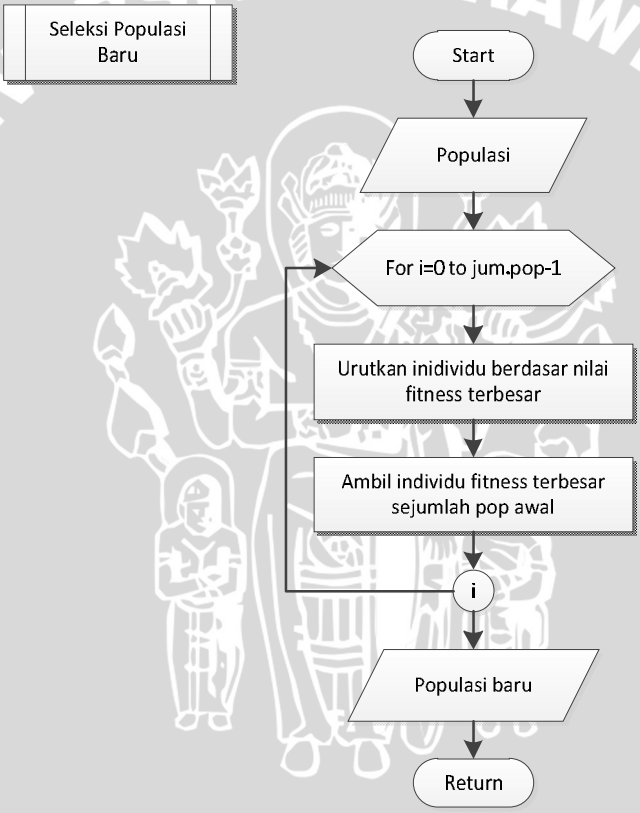
Flowchart proses *repair* dapat dilihat pada gambar 3.16



Gambar 3.16 *Flowchart* Proses *Repair*

3.2.1.7 Seleksi Populasi Baru

Proses terakhir dari algoritma genetika adalah penentuan populasi baru. Dalam penelitian ini penentuan populasi baru dilakukan dengan metode seleksi *rank base selection* (seleksi rangking) yaitu dengan mengurutkan semua individu yang telah melakukan proses rekombinasi berdasarkan nilai *fitness* terbesar, untuk selanjutnya diambil individu – individu dengan nilai *fitness* terbesar sejumlah populasi awal. *Flowchart* proses seleksi populasi baru dapat dilihat pada gambar 3.17.



Gambar 3.17 *Flowchart* Seleksi Populasi Baru

3.3 Perhitungan Manual

Dalam contoh perhitungan manual ini, dimisalkan terdapat 5 trayek angkutan kota yaitu ADL, GM, MKS, TGT dan TSG. Oleh karena 1 individu merupakan 1 solusi maka dalam contoh perhitungan manual ini 1 individu direpresentasikan dengan 5 trayek. Setiap trayek memiliki lokasi awal dan lokasi tujuan. Data lokasi

awal dan lokasi tujuan setiap trayek dapat dilihat pada tabel 3.2. Sedangkan data terminal/sub terminal/APK dapat dilihat pada tabel 3.3.

Tabel 3.2 Data Lokasi Awal dan Lokasi Tujuan Trayek

Trayek	Jenis Trayek	Lokasi Awal	Lokasi Tujuan
I	ADL	Term. Arjosari	Term. Landungsari
II	GM	Term. Gadang	Sub. Term. Mulyorejo
III	MKS	Sub. Term. Mulyorejo	APK Pasar Sukun
IV	TGT	APK Tlogowaru	APK Tirtosari
V	TSG	APK Tawangmangu	APK Gasek

Tabel 3.3 Data Terminal/ Subterminal/ APK

Nama	*Node*
Term. Arjosari	89
Term. Landungsari	1
Term. Gadang	74
Sub. Term. Mulyorejo	172
APK Pasar Sukun	142
APK Tlogowaru	132
APK Tirtosari	138
APK Tawangmangu	285
APK Gasek	354

Dari data yang telah didapat diperolah bahwa kota Malang memiliki 367 titik percabangan (*gen*/*node*) dan terbentuk 548 jalur terhubung (*arc*). Setiap *arc* memiliki parameter panjang jalan. Sebagian data jalan dapat dilihat pada tabel 3.4. Data lengkap dapat dilihat pada lampiran.

Tabel 3.4 Data Jalan

No	NamaJalan	PanjangJalan	Node1	Node2
1	Tlogomas 1	0.24	1	2
2	Tlogomas 2	0.38	2	280
3	MT Haryono 1	0.46	3	361
4	MT Haryono 2	0.16	361	362
5	Mayjen Panjaitan 2	1.32	5	199
6	Sukarno Hatta 5	0.46	5	300

7	Bandung 2	0.29	6	198
8	Jaksa Agung Suprapto 1	0.30	7	8
9	Jaksa Agung Suprapto 2	0.48	7	9
10	Brigjen Slamet Riadi 1	0.20	7	201
11	Basuki Rahmat 2	0.50	8	48
12	Kahuripan 1	0.22	8	165
13	Semeru 1	0.18	8	181
14	Jaksa Agung Suprapto 3	0.50	9	10
15	Pattimura 1	0.18	9	169
16	Jaksa Agung Suprapto 4	0.34	10	11
17	Dr. Cipto 2	0.14	10	170
18	Letjen Sutoyo 1	0.53	11	12
19	Kaliurang	0.57	11	203
20	WR Suprtaman 1	0.21	11	261
21	Letjen Sutoyo 2	0.98	12	13
22	Sarangan 1	0.41	12	90
23	Letjen Sutoyo 3	0.36	13	252
24	Letjen S.Parman 3	0.53	14	113
25	Jendral A. Yani 1	0.11	15	103
26	Jendral A. Yani 2	0.34	16	17
27	Cakalang	1.74	16	96
28	Raden Intan 1	0.38	17	18
29	Raden Intan 2	0.75	18	89
30	Laksda Adi Sucipto 3	0.82	19	108
30	Laksda Adi Sucipto 3	0.82	19	108
31	Laksda Adi Sucipto 4	0.09	19	250
32	Tumenggung Suryo 3	0.06	20	263
33	Sulfat 1	0.27	20	266
34	Panglima Sudirman 1	0.32	21	22
35	Hamid Rusdi	1.20	21	359

Dalam perhitungan manual ini akan ditetapkan parameter awal adalah sebagai berikut :

- Parameter algoritma genetika :
  a. Jumlah populasi = 3
  b. Jumlah generasi = 1
  c. Probabilitas *crossover* (pc) = 50% = 0.5
  d. Probabilitas mutasi (pm) = 30% = 0.3

3.3.1 *Generate* Populasi Awal

Misalkan telah terbentuk populasi awal sebanyak 3 individu, seperti berikut :

Individu 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	205	86	300
	5	365	4	212	213	214	352	354					

Individu 2

K1	89	18	17	16	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	364	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	153	65	56	64	63
	176	175	174	173	172								
K3	172	140	139	79	80	142							
K4	132	133	135	136	74	78	138						
K5	285	284	286	289	288	290	293	294	296	298	299	300	5
	365	4	212	213	214	352	354						

Individu 3

K1	89	18	93	94	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	299
	300	5	365	4	212	213	214	352	354				

3.3.2 Nilai Fitness

Setelah terbentuk individu dengan trayeknya, maka langkah selanjutnya adalah menghitung nilai *fitness.* Oleh karena nilai *fitness* yang digunakan adalah berdasarkan nilai penalti, maka sebelum menentukan nilai *fitness* terlebih dahulu menghitung penalti dari pelanggaran aturan.

Untuk mempermudah melakukan perhitungan jarak tempuh, diasumsikan terlebih dahulu dibentuk trayek tiap individu menjadi sebuah larik trayek yang mengandung *gen-gen* terhubung.

Individu 1

K1	89;92	92;94	94;95	95;107	107;103	103;15	15;245	245;240
	240;231	231;234	234;91	91;87	87;86	86;300	300;5	5;365
	365;4	4;362	362;361	361;3	3;280	280;2	2;1	
K2	74;367	367;73	73;71	71;75	75;76	76;151	151;148	148;143
	143;155	155;63	63;176	176;175	175;174	174;173	173;172	
K3	172;42	42;24	24;120	120;80	80;144	144;80	80;142	
K4	132;133	133;135	135;136	136;74	74;78	78;138		
K5	285;284	284;286	286;287	287;288	288;290	290;293	293;294	294;296
	296;297	297;205	205;86	86;300	300;5	5;365	365;4	4;212
	212;213	213;214	214;352	352;354				

Individu 2

K1	89;18	18;17	17;16	16;95	95;107	107;103	103;15	15;245
	245;240	240;231	231;234	234;88	88;87	87;86	86;300	300;5
	5;365	365;364	364;362	362;361	361;3	3;280	280;2	2;1
K2	74;367	367;73	73;71	71;75	75;76	76;151	151;152	152;153
	153;65	65;56	56;64	64;63	63;176	176;175	175;174	174;173
	173;172							
K3	172;140	140;139	139;79	79;80	80;142			
K4	132;133	133;135	135;136	136;74	74;78	78;138		
K5	285;284	284;286	286;289	289;288	288;290	290;293	293;294	294;296
	296;298	298;299	299;300	300;5	5;365	365;4	4;212	212;213
	213;214	214;352	352;354					

Individu 3

K1	89;18	18;93	93;94	94;95	95;107	107;103	103;15	15;245
	245;240	240;231	231;234	234;88	88;87	87;86	86;300	300;5
	5;365	365;4	4;362	362;361	361;3	3;280	280;2	2;1
K2	74;367	367;73	73;71	71;75	75;76	76;151	151;152	152;154
	154;156	156;155	155;63	63;176	176;175	175;174	174;173	173;172
K3	172;42	42;24	24;120	120;85	85;119	119;80	80;144	144;80
	80;142							
K4	132;133	133;135	135;136	136;74	74;78	78;138		
K5	285;90	90;283	283;284	284;286	286;289	289;288	288;290	290;293
	293;294	294;296	296;297	297;299	299;300	300;5	5;365	365;4
	4;212	212;213	213;214	214;352	352;354			

Selanjutnya menghitung jarak tempuh tiap trayek dari *gen* terhubung.

Individu 1

K1 = 89;92+92;94+94;95+95;107+107;103+103;15+15;245+245;240+ 240;231 +231;234+234;91+91;87+87;86+86;300+300;5+5;365+365;4+ 4;362+ 362;361+361;3+3;280+280;2+2;1

= 0.36+0.8+0.66+0.14+0.52+0.11+0.346+0.25+0.47+0.26+0.32+0.58+ 0.96+0.292+0.46+0.204+0.02+0.02+0.16+0.46+0.268+0.376+0.242

= 8.278

Individu 1	**Individu 2**	**Individu 3**
K1 = 8.278	K1 = 8.94	K1 = 9.088
K2 = 9.746	K2 = 9.036	K2 = 10.396
K3 = 3.63	K3 = 4.71	K3 = 5.084
K4 = 7.32	K4 = 7.32	K4 = 7.32
K5 = 7.202	K5 = 6.348	K5 = 7.254

Sedangkan untuk mempermudah melakukan perhitungan jumlah tumpukan (*intersection*) jalur maka keseluruhan *gen* dalam trayek dalam 1 individu diasumsikan dibentuk menjadi sebuah 1 larik individu dengan *gen* terhubung.

Individu 1

89;92	92;94	94;95	95;107	107;103	103;15	15;245	245;240
240;231	231;234	234;91	91;87	87;86	86;300	300;5	5;365
365;4	4;362	362;361	361;3	3;280	280;2	2;1	74;367
367;73	73;71	71;75	75;76	76;151	151;148	148;143	143;155
155;63	63;176	176;175	175;174	174;173	173;172	172;42	42;24
24;120	120;80	80;144	144;80	80;142	132;133	133;135	135;136
136;74	74;78	78;138	285;284	284;286	286;287	287;288	288;290
290;293	293;294	294;296	296;297	297;205	205;86	86;300	300;5
5;365	365;4	4;212	212;213	213;214	214;352	352;354	

Individu 2

89;18	18;17	17;16	16;95	95;107	107;103	103;15	15;245
245;240	240;231	231;234	234;88	88;87	87;86	86;300	300;5
5;365	365;364	364;362	362;361	361;3	3;280	280;2	2;1
74;367	367;73	73;71	71;75	75;76	76;151	151;152	152;153
153;65	65;56	56;64	64;63	63;176	176;175	175;174	174;173
173;172	172;140	140;139	139;79	79;80	80;142	132;133	133;135
135;136	136;74	74;78	78;138	285;284	284;286	286;289	289;288
288;290	290;293	293;294	294;296	296;298	298;299	299;300	300;5
5;365	365;4	4;212	212;213	213;214	214;352	352;354	

Individu 3

18;93	93;94	94;95	95;107	107;103	103;15	15;245	240;231
231;234	234;88	88;87	87;86	86;300	300;5	5;365	365;4
4;362	362;361	361;3	3;280	280;2	2;1	367;73	73;71
71;75	75;76	76;151	151;152	152;154	156;155	155;63	63;176
176;175	175;174	174;173	173;172	42;24	24;120	120;85	85;119
119;80	80;144	144;80	132;133	133;135	135;136	136;74	74;78
78;138	285;90	283;284	284;286	286;289	289;288	288;290	290;293
293;294	296;297	297;299	299;300	300;5	5;365	365;4	4;212
213;214	214;352	352;354					

Menghitung jumlah kemunculan setiap jenis *gen* terhubung. Hasil perhitungan jumlah kemunculan setiap jenis *gen* dapat dilihat pada tabel 3.5 hingga tabel 3.7.

Tabel 3.5 Jumlah Kemunculan Jenis *Gen* Pada Individu 1

Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah
89;92	1	92;94	1	94;95	1	95;107	1
240;231	1	231;234	1	234;91	1	91;87	1
365;4	2	4;362	1	362;361	1	361;3	1
367;73	1	73;71	1	71;75	1	75;76	1

155;63	1	63;176	1	176;175	1	175;174	1
24;120	1	120;80	1	80;144	1	144;80	1
136;74	1	74;78	1	78;138	1	285;284	1
290;293	1	293;294	1	294;296	1	296;297	1
5;365	2	245;240	1	4;212	1	212;213	1
103;15	1	74;367	1	15;245	1	132;133	1
86;300	2	143;155	1	300;5	2	286;287	1
280;2	1	42;24	1	2;1	1	205;86	1
151;148	1	135;136	1	148;143	1	214;352	1
173;172	1	288;290	1	172;42	1	133;135	1
352;354	1	287;288	1				

Tabel 3.6 Jumlah Kemunculan Jenis *Gen* Pada Individu 2

Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah
89;18	1	18;17	1	17;16	1	16;95	1
245;240	1	240;231	1	231;234	1	234;88	1
5;365	2	365;364	1	364;362	1	362;361	1
74;367	1	367;73	1	73;71	1	71;75	1
153;65	1	65;56	1	56;64	1	64;63	1
173;172	1	172;140	1	140;139	1	139;79	1
135;136	1	136;74	1	74;78	1	78;138	1
288;290	1	290;293	1	293;294	1	294;296	1
214;352	1	365;4	1	4;212	1	212;213	1
95;107	1	103;15	1	107;103	1	15;245	1
88;87	1	86;300	1	87;86	1	300;5	2
361;3	1	280;2	1	3;280	1	2;1	1
75;76	1	151;152	1	76;151	1	152;153	1
63;176	1	175;174	1	176;175	1	174;173	1
79;80	1	132;133	1	80;142	1	133;135	1
285;284	1	286;289	1	284;286	1	289;288	1
296;298	1	299;300	1	298;299	1	213;214	1
352;354	1						

Tabel 3.7 Jumlah Kemunculan Jenis *Gen* Pada Individu 3

Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah	Jenis	Jumlah
18;93	1	93;94	1	94;95	1	95;107	1
240;231	1	231;234	1	234;88	1	88;87	1
365;4	2	4;362	1	362;361	1	361;3	1
367;73	1	73;71	1	71;75	1	75;76	1

156;155	1	155;63	1	63;176	1	176;175	1
42;24	1	24;120	1	120;85	1	85;119	1
132;133	1	133;135	1	135;136	1	136;74	1
283;284	1	284;286	1	286;289	1	289;288	1
296;297	1	297;299	1	299;300	1	300;5	2
213;214	1	214;352	1	352;354	1	119;80	1
107;103	1	15;245	1	103;15	1	74;78	1
87;86	1	2;1	1	86;300	1	288;290	1
3;280	1	152;154	1	280;2	1	5;365	3
76;151	1	173;172	1	151;152	1	293;294	1
175;174	1	144;80	1	174;173	1	4;212	1
285;90	1	78;138	1	80;144	1	290;293	1

Melakukan pemberikan nilai penalty pada setiap individu berdasarkan aturan penalti. Hasil pemberian nilai penalti dapat dilihat pada tabel 3.8.

Tabel 3.8 Penalti Tiap Individu

Jenis	Aturan	f	$penalti\ x\ f$	$\sum penalti\ x\ f$
Individu 1	1	2	20	25
	2	0	0	
	3	1	5	
Individu 2	1	2	20	25
	2	0	0	
	3	1	5	
Individu 3	1	2	20	20
	2	0	0	
	3	0	0	

Menghitung nilai *fitness* dengan persamaan 3.1. Hasil perhitungan *fitness* dapat dilihat pada tabel 3.9

Tabel 3.9 Nilai *Fitness* Tiap Individu

Jenis	Nilai *fitness*
Individu 1	0.0385
Individu 2	0.0385
Individu 3	0.0476

3.3.3 Crossover

Proses berikutnya adalah *crossover*. Dalam proses *crossover* ini, proses penyilangan dilakukan secara kombinasi antar kandidat *parent*. Langkah pertama adalah menentukan kandidat *parent* yang akan dilakukan *crossover*. Kandidat p*arent* pertama didapatkan dengan teknik *elitsm* yaitu mengambil individu dengan nilai *fitness* terbaik. Oleh karena individu yang memiliki nilai *fitness* terbaik adalah individu ke-3 maka individu tersebut dijadikan kandidat *parent* pertama.

Kandidat *parent* kedua dan seterusnya ditentukan dengan membangkitkan bilangan *random* antara 0 dan 1 pada individu – individu selain yang telah terpilih menjadi kandidat *parent* 1. Kemudian dibandingkan dengan nilai pc-nya. Misal hasil nilai *random r* untuk individu 1 dan individu 2 masing – masing adalah 0.41 dan 0.63, pc=0.5, maka individu yang menjadi *parent* kedua adalah individu 1. Sehingga individu yang menjadi *parent* 1 dan 2, yang mengalami *crossover* adalah individu 3 dan 1.

***Parent* 1**

K1	89	18	93	94	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	299
	300	5	365	4	212	213	214	352	354				

***Parent* 2**

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	205	86	300
	5	365	4	212	213	214	352	354					

Selanjutnya menentukan titik *crossover* yaitu dengan cara mengambil *gen* yang sama dan pertama kali ditemukan dalam trayek pada *parent* 1 dan 2 kecuali gen pertama dan terakhir.

***Parent* 1**

K1	89	18	93	94	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	299
	300	5	365	4	212	213	214	352	354				

***Parent* 2**

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	205	86	300
	5	365	4	212	213	214	352	354					

Menukarkan *gen – gen* antar trayek pada *parent* 1 dan 2. Berikut adalah *offspring* hasil *crossover*

***Offspring* 1**

K1	89	18	93	94	95	107	103	15	245	240	231	234	91
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	205
	86	300	5	365	4	212	213	214	352	354			

***Offspring* 2**

K1	89	92	94	95	107	103	15	245	240	231	234	88	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	299	300	5
	365	4	212	213	214	352	354						

Menghitung nilai *fitness* dari masing – masing *offspring*. Hasil perhitungan nilai *fitness* dapat dilihat pada tabel 3.10

Tabel 3.10 Nilai *Fitness* Tiap *Offspring*

Jenis	Nilai *fitness*
Offspring 1	0.0385
Offspring 2	0.0476

3.3.4 Mutasi

Proses selanjutnya adalah mutasi. Mutasi dilakukan terhadap semua individu yang ada (individu populasi awal, *offspring* hasil *crossover*, individu hasil mutasi) berdasarkan nilai probabilitas mutasi (pm) yang telah ditentukan. Bangkitkan nilai *random r* sebanyak total individu yang ada untuk menentukan individu yang akan dimutasi. Bangkitkan nilai *random r* sebanyak total jumlah trayek dari individu terpilih untuk menentukan trayek – trayek yang akan dimutasi.

Misalkan nilai pobabilitas mutasi (pm) = 0.3 dan nilai *random r* yang muncul adalah 0.23 maka terjadi mutasi. Membangkitkan bilangan *random r* antara 0 dan 1 sebanyak jumlah individu yang ada untuk menentukan individu yang akan dimutasi.

* Misal pm = 0.3

a. Individu 1

r = 0.25, maka terjadi mutasi

b. Individu 2

r = 0.7, maka **tidak** terjadi mutasi

c. Individu 3

r = 0.32, maka **tidak** terjadi mutasi

d. *Offspring* 1

r = 0.85, maka **tidak** terjadi mutasi

e. *Offspring* 2

r = 0.19, maka terjadi mutasi

Selanjutnya membangkitkan bilangan *random r* antara 0 dan 1 sebanyak jumlah total trayek dari individu terpilih untuk menentukan trayek yang akan dimutasi. Nilai probabilitas mutasi yang digunakan adalah sama dengan nilai probabilitas mutasi (pm) yang digunakan dalam penentuan individu yang akan dimutasi. Dalam hal ini individu yang terpilih untuk dimutasi adalah individu 1 dan *child* 2.

Oleh karena jumlah total trayek dari individu terpilih adalah 10, maka dibangkitkan bilangan *random r* sebanyak 10. Nilai bangkitan *random r* pada setiap trayek dapat dilihat pada tabel 3.11

Tabel 3.11 Nilai *Random r* Pada Setiap Trayek Individu Terpilih

Trayek	***Random***
89 92 94 95 107 103 15 245 240 231 234 91 87 86 300 5 365 4 362 361 3 280 2 1	0.8
74 367 73 71 75 76 151 148 143 155 63 176 175 174 173 172	**0.28**
172 42 24 120 80 144 80 142	0.32
132 133 135 136 74 78 138	**0.14**
285 284 286 287 288 290 293 294 296 297 205 86 300 5 365 4 212 213 214 352 354	**0.04**

89 92 94 95 107 103 15 245 240 231 234 88 87 86 300 5 365 4 362 361 3 280 2 1	0.38
74 367 73 71 75 76 151 152 154 156 155 63 176 175 174 173 172	0.55
172 42 24 120 85 119 80 144 80 142	0.79
132 133 135 136 74 78 138	**0.07**
285 284 286 287 288 290 293 294 296 297 299 300 5 365 4 212 213 214 352 354	0.5

Dari tabel diatas maka trayek yang dikenai mutasi adalah trayek ke- 2, 4 dan 5 dari individu 1 serta trayek ke- 4 dari *offspring* 2. Selanjutnya menentukan titik mutasi secara *random gen* pada trayek – trayek terpilih kecuali pada gen pertama dan terakhir.

Trayek ke-2 individu 1

74	367	73	71	75	76	151	148	143	155	63	176
175	174	173	172								

Trayek ke-4 individu 1

132	133	135	136	74	78	138

Trayek ke-5 individu 1

285	284	286	287	288	290	293	294	296	297	205	86
300	5	365	4	212	213	214	352	354			

Trayek ke-4 *offspring* 2

132	133	135	136	74	78	138

Menentukan *gen* – *gen* yang terhubung dengan *gen* sebelum dan sesudah titik mutasi, pada masing – masing *gen* titik mutasi. *Gen-gen* terhubung dapat dilihat pada tabel 3.2. Selanjutnya mengganti *gen* titik mutasi yang terpilih dengan *gen* – *gen* baru yang terhubung.

Trayek ke-2 individu 1

74	367	73	71	75	76	151	152	154	156	143
155	63	176	175	174	173	172				

Trayek ke-4 individu 1

132	133	135	133	132	131	130	136	74	78	138

Trayek ke-5 individu 1

285	284	286	287	286	289	288	290	293	294	296	297
205	86	300	5	365	4	212	213	214	352	354	

Trayek ke-4 *child* 2

132	133	134	137	135	136	74	78	138

Menghitung nilai *fitness* dari individu yang mengalami mutasi. Tabel hasil perhitungan nilai *fitness* dapat dilihat pada tabel 3.12.Berikut adalah individu hasil mutasi

Individu-M 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	143	155	63
	176	175	174	173	172								
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	133	132	131	130	136	74	78	138		
K5	285	284	286	287	286	289	288	290	293	294	296	297	205
	86	300	5	365	4	212	213	214	352	354			

***Offspring*-M 2**

K1	89	92	94	95	107	103	15	245	240	231	234	88	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	134	137	135	136	74	78	138				
K5	285	284	286	287	288	290	293	294	296	297	299	300	5
	365	4	212	213	214	352	354						

Tabel 3.12 Hasil Perhitungan Nilai *Fitness* Individu Hasil Mutasi

Jenis	Nilai *fitness*
Individu-M 1	0.0385
Offspring-M 2	0.0476

3.3.5 Repair

Proses selanjutnya adalah *repair*. Proses ini berfungsi memperbaiki individu sehingga mendapatkan individu yang valid dan siap untuk diseleksi guna mendapatkan generasi baru. Individu yang dianggap valid adalah apabila telah memenuhi kriteria bahwa tidak ada *gen* yang berulang dalam 1 trayek.

Individu – Individu Sebelum *Repair*

Individu 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	205	86	300
	5	365	4	212	213	214	352	354					

Individu 2

K1	89	18	17	16	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	364	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	153	65	56	64	63
	176	175	174	173	172								
K3	172	140	139	79	80	142							
K4	132	133	135	136	74	78	138						
K5	285	284	286	289	288	290	293	294	296	298	299	300	5
	365	4	212	213	214	352	354						

Individu 3

K1	89	18	93	94	95	107	103	15	245	240	231	234	88
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	299
	300	5	365	4	212	213	214	352	354				

***Offspring* 1**

K1	89	18	93	94	95	107	103	15	245	240	231	234	91
	87	86	300	5	365	4	362	361	3	280	2	1	
K2	74	367	73	71	75	76	151	148	143	155	63	176	175
	174	173	172										
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	136	74	78	138						
K5	285	90	283	284	286	289	288	290	293	294	296	297	205
	86	300	5	365	4	212	213	214	352	354			

***Offspring* 2**

K1	89	92	94	95	107	103	15	245	240	231	234	88	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	135	136	74	78	138						
K5	285	284	286	287	288	290	293	294	296	297	299	300	5
	365	4	212	213	214	352	354						

Individu-M 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	143	155	63
	176	175	174	173	172								
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	133	132	131	130	136	74	78	138		
K5	285	284	286	287	286	289	288	290	293	294	296	297	205
	86	300	5	365	4	212	213	214	352	354			

***Offspring*-M 2**

K1	89	92	94	95	107	103	15	245	240	231	234	88	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	155	63	176
	175	174	173	172									
K3	172	42	24	120	85	119	80	144	80	142			
K4	132	133	134	137	135	136	74	78	138				
K5	285	284	286	287	288	290	293	294	296	297	299	300	5
	365	4	212	213	214	352	354						

Oleh karena hanya individu-M1 yang mengalami perulangan *gen*, maka akan dilakukan penghapusan *gen* yang mengalami perulangan.

Individu Yang Mengalami Perulangan

Individu-M 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	143	155	63
	176	175	174	173	172								
K3	172	42	24	120	80	144	80	142					
K4	132	133	135	133	132	131	130	136	74	78	138		
K5	285	284	286	287	286	289	288	290	293	294	296	297	205
	86	300	5	365	4	212	213	214	352	354			

Individu Setelah Proses *Repair*

Individu-R 1

K1	89	92	94	95	107	103	15	245	240	231	234	91	87
	86	300	5	365	4	362	361	3	280	2	1		
K2	74	367	73	71	75	76	151	152	154	156	143	155	63
	176	175	174	173	172								
K3	172	42	24	120	80	144	80	142					
K4	132	131	130	136	74	78	138						
K5	285	284	286	289	288	290	293	294	296	297	205	86	300
	5	365	4	212	213	214	352	354					

Oleh karena individu M-1 mengalami perubahan gen dalam trayek, maka dilakukan perhitungan nilai *fitness*. Sehingga nilai *fitness* individu M-1 menjadi 0.0385.

3.3.6 Seleksi

Setelah semua proses dilakukan langkah terakhir adalah menentukan individu populasi baru dengan cara seleksi. Metode seleksi yang digunakan adalah *rank base selection* (seleksi rangking). Langkah pertama yaitu mengumpulkan semua semua individu yang telah mengalami proses rekombinasi. Keseluruhan individu hasil rekombinasi dapat dilihat pada tabel 3.13

Tabel 3.13 Keseluruhan Individu Hasil Rekombinasi

Jenis	Nilai *fitness*
Individu-R 1	0.0385
Individu 2	0.0385
Individu 3	0.0476
Child 1	0.0385
Child-M 2	0.0476

Selanjutnya mengurutkan keseluruhan individu dengan hasil rekombinasi berdasarkan nilai *fitness* yang terbesar untuk selanjutnya diambil individu – individu dengan nilai terbaik sejumlah populasi awal. Hasil pengurutan dapat dilihat pada tabel 3.14. Sedangkan hasil populasi baru dapat dilihat pada tabel 3.15

Tabel 3.14 Hasil Pengurutan Nilai *Fitness*

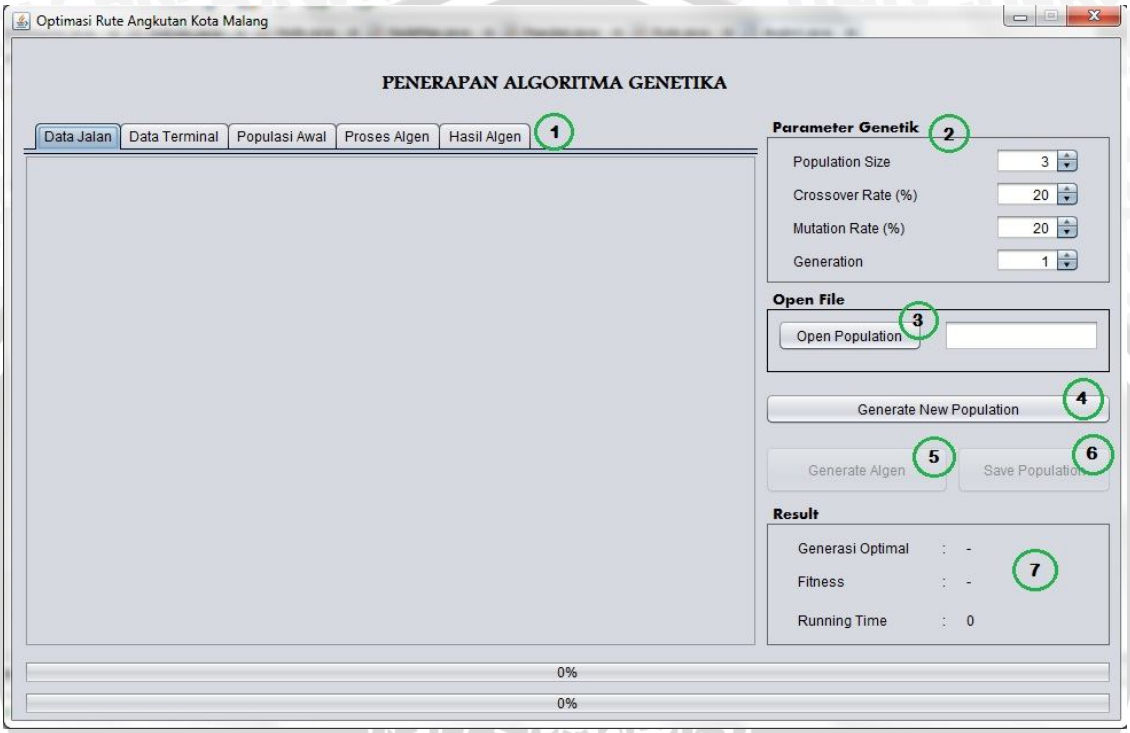
Jenis	Nilai *fitness*
Individu 3	0.0476
Child-M 2	0.0476
Individu-R 1	0.0385
Individu 2	0.0385
Child 1	0.0385

Tabel 3.15 Populasi Baru

Jenis	Nilai *fitness*
Individu 1	**0.0476**
Individu 2	**0.0476**
Individu 3	**0.0385**

3.4 Perancangan User Interface

Rancangan *user interface* dari perangkat lunak untuk permasalahan optimasi rute angkutan kota Malang dapat dilihat pada gambar 3.18 :



Gambar 3.18 Rancangan *User Interface*

Keterangan Gambar 3.18 :

1. Tab yang berisi data jalan, data terminal, populasi awal, proses algen dan hasil algen
2. Panel *input* paramater genetika
3. Panel *open file* populasi awal
4. Tombol *generate new population*
5. Tombol *generate* proses algoritma genetika
6. Tombol *save population*
7. Panel *result* dari proses algoritma genetika

3.5 Perancangan Uji Coba dan Evaluasi

Pengujian akan dilakukan terhadap perangkat lunak untuk mengetahui hasil optimasi rute angkutan kota dengan penerapan algoritma genetika. Pada proses pengujian, perangkat lunak akan diiuji dengan mengkombinasikan parameter – parameter genetika (probabilitas crossover, probabilitas mutasi, jumlah populasi dan jumlah generasi). Tujuan dari uji coba adalah untuk mengetahui pengaruh parameter genetika terhadap hasil optimasi rute angkutan kota, dalam hal ini adalah nilai *fitness*-nya.

Rancangan uji coba ke-1 yaitu untuk mengetahui pengaruh probabilitas *crossover* dan mutasi terhadap nilai *fitness* yang dihasilkan. Pengujian dilakukan dengan terhadap nilai probabilitas *crossover* dan probabilitas mutasi dengan jumlah populasi dan jumlah generasi yang digunakan sama. Probabilitas crossover dan probabilitas mutasi akan diujikan adalah 10% hingga 100%. Untuk setiap percobaan dilakukan pengulangan sebanyak 5 kali dan diambil rata – rata nilai *fitnes.* Tabel rancangan pengujian pengaruh probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.16

Tabel 3.16 Rancangan Uji Coba Pengaruh Probabilitas Crossover Dan Probabilitas Mutasi Terhadap Nilai Fitness Rata – Rata

Prob. Mutasi	Prob. *Crossover*									
	10	20	30	40	50	60	70	80	90	100
10										
20										
30										
40										
50										
60										
70										
80										
90										
100										

Rancangan uji coba selanjutnya yaitu untuk mengetahui pengaruh jumlah populasi dan jumlah generasi terhadap nilai *fitness* yang dihasilkan. Pengujian

dilakukan dengan memberikan nilai jumlah populasi atau jumlah generasi yang beragam dengan nilai probabilitas *crossover* dan probabilitas mutasi yang sama, yang diperoleh dari nilai probabilitas *crossover* dan probabilitas mutasi terbaik dalam pengujian ke-1. Untuk setiap percobaan dilakukan pengulangan sebanyak 5 kali dan diambil rata – rata nilai *fitnes.* Tabel rancangan pengujian pengaruh ukuran populasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.17. Sedangkan Tabel rancangan pengujian pengaruh jumlah generasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.18

Tabel 3.17 Rancangan Pengujian Pengaruh Ukuran Populasi Terhadap Nilai *Fitness* Rata – Rata

No.	**Jumlah Populasi**	**Pc : ... % ; Pm : ... %**					**Rata – Rata Nilai *Fitness***
		Percobaan Ke-					
		1	2	3	4	5	
1	10						
2	30						
3	50						
4	...						
5	N						

Tabel 3.18 Rancangan Pengujian Pengaruh Jumlah Generasi Terhadap Nilai *Fitness* Rata - Rata

No.	**Jumlah Generasi**	**Pc : ... % ; Pm : ... %**					**Rata – Rata Nilai *Fitness***
		Percobaan Ke-					
		1	2	3	4	5	
1	10						
2	30						
3	50						
4	...						
5	N						

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai penerapan rancangan perangkat lunak yang telah dipaparkan dalam bab III. Selain itu dalam bab ini juga menjelaskan hasil dari penelitian beserta analisanya. Secara umum proses yang dijelaskan dalam bab ini adalah implementasi program, proses pengujian perangkat lunak dan terakhir adalah analisa hasil pengujian.

4.1 Lingkungan Implementasi

Proses implementasi merupakan tahap perangkat lunak siap dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah perangkat lunak yang dibuat telah menghasilkan tujuan yang diinginkan. Lingkungan implementasi yang akan dijelaskan pada bab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan perangkat lunak optimasi rute angkutan kota Malang dengan penerapan algoritma genetika adalah sebagai berikut :

1. Prosesor Intel(R) Core(TM) I3 – 370M 2.40 GHz
2. Memori 2GB DDR3
3. *Harddisk* dengan kapasitas 320 GB
4. Monitor 14”
5. *Keyboard*
6. *Mouse*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan perangkat lunak optimasi rute angkutan kota Malang dengan penerapan algoritma genetika adalah sebagai berikut :

1. Sistem Operasi Microsoft Windows 7 Ultimate 32-bit sebagai lingkungan kerja dari sistem.
2. *Netbeans IDE 7.0* merupakan aplikasi *Integrated Development Environment* (IDE) yang berfungsi sebagai media untuk membuat dan *compile* program, dengan menggunakan bahasa Java.
3. JDK 1.7, sebagai *software development kit* untuk penulisan *code* program dalam java.
4. *XAMPP 1.7.4* sebagai perangkat lunak yang digunakan untuk mengembangkan *database* dengan menggunakan *database* mysql.

4.2 Implementasi Program

Berdasarkan perancangan perangkat lunak yang telah diuraikan dalan bab III, maka pada bagian ini akan dibahas mengenai implementasi program. Implementasi program terdiri dari beberapa kelas yang mempunyai fungsi masing – masing dengan sebuah kelas utama sebagai kelas yang menampung keseluruhan proses. Adapun kelas – kelas yang dibangun dijelaskan dalam tabel 4.1.

Tabel 4.1 Kelas – Kelas Yang Dibangun

No	Nama Kelas	Keterangan
1	`Chromosome`	Kelas yang mendeklarasikan proses pembentukan trayek dan berisi prosedur pengolahan trayek seperti fungsi *fitness*, *crossover*, mutasi dan *repair*
2	`Individu`	Kelas yang mendeklarasikan proses pembentukan individu dan berisi prosedur pengolahan individu
3	`Populasi`	Kelas yang merepresentasikan populasi atau sekumpulan individu
4	`Gen`	Kelas yang mendefinisikan dan menyimpan data *gen* yang saling terhubung
5	`GenMap`	Kelas yang mengelolah *gen-gen*, dengan *key* dan *value* masing – masing berupa *gen* dan jumlah *gen.*
6	`GenList`	Kelas yang berfungsi sebagai media penyimpanan data *node* jalan yang akan digunakan (sebagai *gen*)
7	`Rute`	Kelas yang merepresentasikan rute trayek angkutan kota
8	`Round`	Kelas yang berfungsi untuk mengubah nilai dengan cara pembulatan

9	`Angkot`	Kelas utama yang menyimpan dan memanggil proses dari kelas lain. Kelas ini juga berfungsi menampilkan *interface* dari program

Setiap kelas tersebut diatas memiliki fungsi dan proses masing – masing yang dijelaskan melalui *method*. Adapun penjelasan method dijabarkan dalam tabel 4.2 hingga tabel 4.9.

Tabel 4.2 *Method – Method* Dalam *Class* `Chromosome`

No.	Nama *Method*	Keterangan
1	`insert(int value)`	Menambahkan *gen* satu persatu
2	`insertAll(Collection c)`	Menambahkan keseluruhan *gen* dalam 1 collection
3	`crossover(Chromosome chrom1, Chromosome chrom2): chromosome []`	Menyilangkan 2 trayek, hasilnya adalah dua trayek baru
4	`mutate()`	Proses mutasi *gen* dalam trayek yaitu mengganti dengan *gen – gen yang* baru
5	`getFare():double`	Mengambil nilai jarak
6	`fare():double`	Membaca jarak dari *database*
7	`getGenMap():GenMap`	Mendapatkan jumlah *intersection* jalur
8	`repair():String`	Proses *repair* yang dilakukan terhadap *gen* yang dikenai
9	`toStreet():String`	*Method* untuk menampilkan format “Nama Jalan” dari *gen – gen*
10	`isFareChanged : boolean`	Menunjukkan status perubahan nilai *fare* dalam trayek

Tabel 4.3 *Method – Method* Dalam *Class* `Individu`

No.	Nama *Method*	Keterangan
1	`getSingleChromosome(int index):Chromosome`	Mengambil 1 trayek yang telah terbentuk
2	`setSingleChromosome(int index, Chromosome chromosome)`	Menetapkan trayek pada indeks tertentu dalam individu
3	`fitness():double`	*Method* persamaan *fitness*
4	`penalty():double`	Perhitungan total *penalty*
5	`crossover(Individu ind1,Individu ind2): Individu[]`	Menyilangkan 2 individu, hasilnya adalah dua individu baru
6	`mutate(double rate)`	Menentukan trayek dalam individu yang

		akan dikenai mutasi
7	`toString():String`	*Method override* yang berfungsi untuk menampilkan trayek dalam bentuk *string*
8	`toStreet():String`	Menampilkan trayek dalam bentuk nama jalan dan panjang jalan
9	`repair():String`	Proses *repair* dalam 1 individu yaitu menentukan individu yang dikenai *repair*

Tabel 4.4 *Method – Method* Dalam Class `Populasi`

No.	Nama *Method*	Keterangan
1	`toString():String`	*Method override* untuk menampilkan individu dalam bentuk string (angka)
2	`toStreet():String`	Menampilkan dalam bentuk nama jalan

Tabel 4.5 *Method – Method* Dalam *Class* `Gen`

No.	Nama Kelas	Keterangan
1	`hashCode():int`	Kode hash suatu *gen*
2	`equals(Object obj):boolean`	Pengecekan *gen*
3	`toString():String`	Menampilkan bentuk *string* (angka) dari *gen*

Tabel 4.6 *Method – Method* Dalam *Class* `GenMap`

No.	Nama *Method*	Keterangan
1	`add(Gen gen)`	Menambahkan *gen* satu per satu
2	`addAll(GenMap map)`	Menambahkan keseluruhan *gen* dalam *GenMap*

Tabel 4.7 *Method – Method* Dalam *Class* `GenList`

No.	Nama *Method*	Keterangan
1	`getValue(Index gen)`	Mengambil nilai dari *gen* berdasarkan *key index*
2	`keySet()`	*Method* untuk *set* nilai *key*
3	`Put(Index gen, Value value)`	Mengambil *key index gen* dan isi (*value*) dari *gen*

Tabel 4.8 *Method – Method* Dalam *Class* `Rute`

No.	Nama *Method*	Keterangan
1	`PenaltyMax(byte Track, double length):int`	*Method* untuk aturan *penalty* jarak maksimal dan pemberian nilai *penalty*
2	`PenaltyMin(double length):int`	*Method* untuk aturan *penalty* jarak minimal dan pemberian nilai *penalty*
3	`getTransGens(int index):int[]`	Mengambil *gen transite* berdasarkan *index*-nya

Tabel 4.9 *Method – Method* Dalam *Class* `Round`

No.	Nama *Method*	Keterangan
1	`Scale(double value): String`	Pembulatan berdasarkan kedekatan
2	`Ceiling(double value): double`	Pembulatan berdasarkan konsep matematika

Tabel 4.10 *Method – Method* Dalam *Class* `Angkot`

No.	Nama *Method*	Keterangan
1	`getJalan()`	Mengambil data jalan dari *database*
2	`getTerminal()`	Mengambil data terminal dari *database*
3	`initGenParam()`	Inisialisasi variabel kelas berdasarkan *input*-an jumlah generasi, pc dan pm
4	`initPopulation()`	Inisialisasi populasi (*random* populasi)
5	`initPopulationSize()`	Inisialisasi variabel kelas berdasarkan *input*-an jumlah populasi
6	`Backward(ArrayList <Integer>x): ArrayList <Integer>`	Membalikkan urutan elemen (*gen*) dalam *ArrayList*
7	`enable**Buttons(boolean able)**`	Pengaturan tombol
8	`hitungFitness():String`	Menghitung nilai *fitness* dalam tiap individu
9	`crossOver():String`	Proses *crossover*, pengecekan terhadap *elitism* dan *random* r berdasarkan Pc
10	`mutate():String`	Proses mutasi, menentukan individu yang akan dikenai mutasi berdasarkan Pm
11	`repair():String`	Proses *repair* (memperbaiki individu *invalid*)
12	`selection(int generation):String`	Proses seleksi populasi baru

13	`finishing()`	Menampilkan hasil proses algen

4.2.1 *Generate* Populasi Awal

Proses *generate* populasi awal merupakan proses pembentukan individu jumlah *input populiaton size.* Dalam proses *generate* populasi awal akan diikuti pula proses inisialiasasi individu. Setiap 1 individu akan di *generate* sebanyak 25 trayek. Jumlah trayek dalam individu adalah *dinamis*, sebagai representasi jumlah trayek angkutan kota Malang, yaitu 25 trayek.

Proses *generate setiap* trayek (rute) dilakukan secara *random* dengan ketentuan yang telah ditetapkan yaitu dengan memberikan *gen - gen transit* dalam setiap trayek (baris 11). Proses *random* rute akan dilakukan pada setiap antar *gen transit* dalam 1 trayek (baris 14-16). Proses *generate* populasi awal dapat dilihat pada *sourcecode* 4.1.

```
1  private void initPopulationSize(){
2          population_size = (Byte) popSpin.getValue();
3  
4          initPopulasi = new Populasi();
5          for(int i = 0; i < population_size; i++)
6              initPopulasi.add(new Individu());
7      }
8  
9  
10 for(int i = 0; i < total; i++){
11       int[] trans = Rute.getTransGens(i);
12       for(int j = 0; j < population_size; j++)
13 initPopulasi.get(j).setSingleChromosome(i, new
   Chromosome(trans[0], trans[trans.length-1]));
14       for(int j = 0; j < trans.length - 1; j++){
15       ConnDB dt = new ConnDB();
16       ArrayList<ArrayList<Integer>> asd =
   dt.randomStreet(trans[j], trans[j+1]);
17     if(asd.isEmpty()){
18       asd = dt.randomStreet(trans[j+1], trans[j]);
19       for(int k = 0; k < population_size; k++){
20 Chromosome c = initPopulasi.get(k).getSingleChromosome(i);
21       if(j != 0)
22       c.insert(trans[j]);
23 ArrayList<Integer> x = asd.get(ran.nextInt(asd.size()));
24       c.insertAll(backward(x));
25       }
```

```
26  }
27    else {
28         for(int k = 0; k < population_size; k++){
29  Chromosome c = initPopulasi.get(k).getSingleChromosome(i);
30         if(j != 0)
31         c.insert(trans[j]);
32  c.insertAll(asd.get(ran.nextInt(asd.size())));
33       }
34  }
```

***Sourcecode* 4.1** Generate Populasi Awal

4.2.2 Hitung *Fitness*

Perhitungan nilai *fitness* diperoleh dengan cara memberikan nilai penalti terhadap pelanggaran yang dilakukan untuk selanjutnya dihitung berdasarkan persamaan fungsi *fitness* yang telah dijabarkan dalam bab III.

Perhitungan nilai *fitness* dilakukan pada setiap individu (baris 1-4). Tahap pertama hitung nilai *fitness* adalah menghitung penalti jarak maksimal (baris 12-24) dan penalti jarak minimal (baris 26-28) dari setiap trayek serta *penalty* dari penumpukan (*intersection*) jalur (baris 45-47). Dari perhitungan tersebut akan dilakukan pula perhitungan jumlah total penalti berdasarkan jumlah *frekuensi* pelanggaran (baris 31-44). Tahap terakhir adalah menghitung nilai *fitness* berdasarkan persamaan *fitness* (baris 51-56). Proses hitung *fitness* dapat dilihat pada *sourcecode* 4.2.

```
1   private String hitungFitness(){
2           double max = Double.MIN_VALUE;
3           for(int i = 0; i < population_size; i++){
4              double p = population.get(i).fitness();
5              if(max < p){
6                   max = p;
7                   elitismIndex = i;
8               }
9           }
10     }
11
12  public static int PenaltyMax(byte track, double length){
13  if((track == AG && length > 15) || ( track == GA && length > 15.3)||
14         ( track == LG && length > 16.8) || (track == AL && length >  17.2)||
15         ((track == AT || track == GML ) && length > 18) ||
```

```
    (track == AJG && length > 18.2)||
16      ((track == ABB || track == AMG || track == TST) &&
length > 16) || (track == GL && length >15.8)||
17      (track == LDG || track == MM && length > 15.2) ||
(track == ADL && length > 14.8 )||
18      ((track == JPK || track == TSG)&& length > 10) ||
(track == ASD && length >10.5)||
19       (track == ABG && length > 26) ||(track == CKL &&
length > 22) ||(track == JDM && length > 13)||
20       (track == MK && length > 11.5) || (track == MT &&
length > 9) || (track == MKS && length > 8) ||
21      (track == GM && length > 8.5)||(track == TGT &&
length > 6.3))
22      return 10;
23    else return 0;
24   }
25
26  public static int PenaltyMin(double length){
27        if(length < 5) return 5;
28        else return 0;
29  }
30  public double penalty(){
31        int f1 = 0, f2 = 0, f3 = 0,
32            p1 = 0, p2 = 10, p3 = 0;
33        NodeMap map = new NodeMap();
34        for (int i = 0; i < chromosomes.length; i++) {
35            double fare = chromosomes[i].getFare();
36            double _p1 = Rute.PenaltyMax((byte)i, fare);
37            int _p3 = Rute.PenaltyMin(fare);
38            if(_p1 != 0) f1++;
39            if(_p3 != 0) f3++;
40            p1 += _p1;
41            p3 += _p3;
42            map.addAll(chromosomes[i].getNodeMap());
43        }
44        for(Node i : map.keySet())
45            if(map.get(i) > 3) f2++;
46        return p1*f1 + p2*f2 + p3*f3 ;
47   }
48  public double fitness(){
49        if(fitnessChanged) {
50            fitness = 1/ (penalty()+1);
51            fitnessChanged = false;
52        }
53        return fitness;
54  }
```

***Sourcecode* 4.2** Hitung Fitness

4.2.3 Proses *Crossover*

Proses *crossover* dimulai dengan memilih individu – individu sebagai kandidat *parent* yang akan mengalami *crossover*. Proses pemilihan *parent* dilakukan dengan 2 cara, yaitu dengan metode *elitism* dan nilai *random r* dengan nilai probabilitas *crossover*. Penentuan kandidat *parent* dengan cara *elitsm* yaitu dengan memilih individu yang memiliki nilai *fitness* terbaik (baris 4-8). Pemilihan kandidat *parent* selanjutnya yaitu dengan memilih individu – individu yang memiliki nilai *random r* < pc (baris 10-24). Proses penentuan kandidat *parent* dapat dilihat pada *sourcecode* 4.3.

```
1  private String crossOver(){
2          int i;
3          boolean found = false;
4          for(int i = 0; i < population_size; i++){
5          double p = round(population.get(i).fitness());
6              if(max < p){
7                  max = p;
8                  elitismIndex = i;
9              }
10
11 Populasi kandidatCO = new Populasi();
12 kandidatCO.add(population.get(elitismIndex));
13         boolean exist = false;
14         for(int i = 0; i < population.size(); i++)
15             if(i != elitismIndex)
16                 if(Math.random() < CORate){
17                     kandidatCO.add(population.get(i));
18                     exist=true;
19                 }
20
21         if (!exist){
22             int i=elitismIndex;
23             while(i==elitismIndex)
24                 i=new Random().nextInt(population.size());
25             kandidatCO.add(population.get(i));
```

***Sourcecode* 4.3** Kandidat *Parent*

Proses selanjutnya yaitu melakukan persilangan (*crossover*). Proses persilangan dilakukan dengan mengkombinasikan masing – masing 2 individu yang telah terpilih menjadi kandidat *parent* (baris 1-7). Metode persilangan menggunakan metode yang telah dijabarkan dalam bab III. Untuk memulai

persilangan terlebih dahulu ditentukan titik *crossover*, titik *crossover* didapat dengan mencari *gen* yang sama yang pertama kali ditemukan, antar trayek, antar individu. Selanjutnya dilakukan persilangan antar trayek dengan cara penukaran *gen – gen* antar trayek (baris 8-33), antar individu dimulai dari *gen* titik *crossover* (baris 38-49). Proses *crossover* dapat dilihat pada *sourecode* 4.4.

```
1  int total = kandidatCO.size(), p = 1
2      for(int i = 0; i < total - 1; i++){
3        for (int j = i + 1; j < total; j++) {
4  Individu[] co = Individu.crossOver(kandidatCO.get(i),
   kandidatCO.get(j));
5        population.add(co[0])
6                 population.add(co[1])
7         p++;
8     }
9  }
10
11  public static Chromosome[] co(Chromosome chrom1, Chromosome
   chrom2){
12          Chromosome newChromosome1 = new
   Chromosome(chrom1.start, chrom1.end),
13                      newChromosome2 = new
   Chromosome(chrom2.start, chrom2.end);
14          int index1 = 1, index2 = 1;
15          for(int j = 1; j < chrom1.size() - 1; j++)
16              if(chrom2.contains(chrom1.get(j))){
17                  index1 = j;
18                  index2 = chrom2.indexOf(chrom1.get(j));
19                  break;
20              }
21  if(index1 == 1 && index2 == 1){
22              for(int j = 1; j < index1; j++)
   newChromosome1.insert(chrom1.get(j));
23              for(int j = 1; j < index2; j++)
   newChromosome2.insert(chrom2.get(j));
24              for(int j = index1; j < chrom1.size() - 1; j++)
   newChromosome2.insert(chrom1.get(j));
25              for(int j = index2; j < chrom2.size() - 1; j++)
   newChromosome1.insert(chrom2.get(j));
26              return new Chromosome[]{newChromosome1,
   newChromosome2};
27          }
28          else return new Chromosome[]{chrom2, chrom1};
29      }
30
31  public static Individu[] crossOver(Individu individu1,
```

```
    Individu individu2){
32      Individu offspring1 = new Individu(), offspring2 =
new Individu();
33      for(int i = 0; i < Rute.TRACK; i++){
34          Chromosome[] crs =
Chromosome.co(individu1.getSingleChromosome(i),
35                  individu2.getSingleChromosome(i));
36          offspring1.setSingleChromosome(i, crs[0]);
37          offspring2.setSingleChromosome(i, crs[1]);
38      }
39      return new Individu[]{offspring1, offspring2};
40  }
```

***Sourcecode* 4.4** Proses *Crossover*

4.2.4 Proses Mutasi

Proses mutasi dimulai dengan menentukan individu yang akan dimutasi. Penentuan individu dilakukan secara *random* berdasarkan nilai probabilitas mutasi (baris 1-7). Selanjutnya menentukan trayek – trayek yang akan dimutasi dari individu mutasi terpilih. Penentuan trayek yang akan dimutasi dilakukan secara *random* berdasarkan nilai probabilitas mutasi (9-13). Tahap terakhir yaitu melakukan proses mutasi terhadap *gen* yang telah terpilih menjadi titik mutasi. Gen titik mutasi akan diganti dengan *gen* - *gen* baru yang terhubung dan telah terpilih (baris 16-28). Proses mutasi dapat dilihat pada *soucecode* 4.5.

```
1   private String mutate(){
2       for(int i = 0; i < population_size; i++){
3           if(Math.random() < MTRate){
4               population.get(i).mutate(MTRate);
5               mutation = true;
6           }
7   }
8
9   public void mutate(double rate){
10      for(int i = 0; i < Rute.TRACK; i++)
11          if(Math.random() < rate)
12              chromosomes[i].mutate();
13      fitnessChanged = true;
14  }
15
16  public void mutate(){
17      ConnDB db = new ConnDB();
18      int index = new Random().nextInt(size() - 2) + 1;
19      int temp = remove(index);
```

```
20          ArrayList<Integer> subRoute =
db.getStreetSubstitute(get(index-1), get(index));
21          if(subRoute.isEmpty()){
22              add(index,temp);
23          }
24          else{
25              addAll(index, subRoute);
26              fareChanged = true;
27          }
28          db.close();
29  }
```

***Sourcecode* 4.5** Proses Mutasi

4.2.5 Proses *Repair*

Proses *repair* merupakan proses memperbaiki individu yang tidak *valid* (individu yang mengalami perulangan *gen*). Proses *repair* dilakukan dengan menghapus *gen* – *gen* yang berulang dalam trayek. Langkah awal proses *repair* adalah mengambil semua trayek dalam individu (baris 1-5) untuk dilakukan pengecekan apakah terdapat perulangan *gen*. Apabila didalam trayek mengalami perulangan rute maka akan dilakukan proses *repair* (baris 10-28). Proses *repair* dapat dlihat pada *sourcecode* 4.6.

```
1   public String repair() {
2           String str = "";
3           for(int i = 0; i < Rute.TRACK; i++){
4               str += "<b><i>Kromosom " + i +"</i></b> : " +
chromosomes[i].repair() + "<br/>";
5               fitnessChanged = chromosomes[i].isFareChanged();
6           }
7           return str;
8       }
9
10  public String repair() {
11          String str  = "<font color = gray>";
12          for (int i = 0; i < size(); i++){
13              int removed = remove(i);
14              str += removed + " - ";
15              if(contains(removed)){
16                  int index = indexOf(removed), j = i;
17                  str += "</font><b><font color = red>";
18                  while(j <= index){
19                      str += remove(j) +" - ";
20                      index--;
```

```
21                  }
22                  add(i, removed);
23                  fareChanged = true;
24                  str += "</font></b><font color = gray>";
25              }
26              else add(i, removed);
27          }
28          return str.substring(0, str.length() - 2) +
"</font>";
29      }
```

***Sourcecode* 4.6** Proses *Repair*

4.2.6 Proses Seleksi

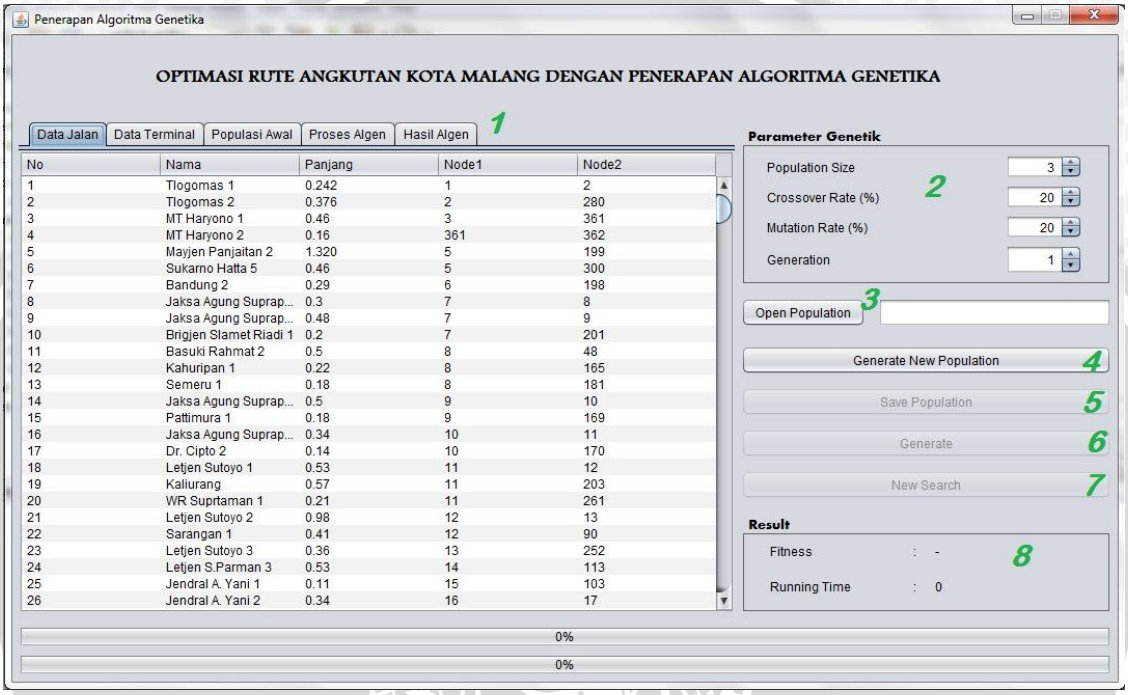
Proses seleksi merupakan tahapan akhir dari proses algortima genetika. Proses seleksi dilakukan dengan menggunakan metode *Rank Base Fitness*, dimana semakin besar nilai *fitness* suatu individu, maka semakin besar kesempatan terpilih untuk menjadi individu populasi baru dalam generasi berikutnya. Tahap ini dimulai dengan mengurutkan nilai *fitness* dari individu dalam populasi (baris 2-5). Proses seleksi akan mengambil individu dengan nilai *fitness* terbaik sejumlah populasi awal dan menyisihkan (*remove*) individu dengan nilai *fitness* terendah (baris 7-13). Proses seleksi dapat dilihat pada *sourcecode* 4.7.

```
1   private void  selection(int generation){
2           Compare bvc =  new Compare(map);
3           TreeMap<Individu, Double> sorted_map = new
4               TreeMap(bvc);
5           sorted_map.putAll(map);
6
7           for(Individu individu : sorted_map.keySet()){
8               db.addreport("Individu <font color =
9   blue>"+individu.parentIndex+"</font> : "+
10  Round.scale(individu.fitness()));
11              if(population.size() > population_size){
12                  population.remove(individu);
13                  db.addreport(" - telah dihapus<br/>");
14              }
15  }
```

***Sourcecode* 4.7** Proses Seleksi

4.3 Implementasi Antarmuka

Berdasarkan rancangan antarmuka yang telah dijabarkan dalam bab III, terdapat 8 komponen dalam tampilan utama dari program Optimasi Rute Angkutan Kota Dengan Penerapan Algoritma Genetika. Adapun tampilan utama dapat dilihat pada gambar 4.1



Gambar 4.1 Tampilan Utama Program

Keterangan Gambar 4.1 :

1. Panel tab terdiri atas 5 tab, terdiri atas :
   - Tab data jalan : berfungsi menampilkan data jalan
   - Tab data terminal : berfungsi menampilkan data terminal
   - Tab data populasi awal : berfungsi menampilkan *output* populasi awal
   - Tab proses algen : berfungsi menampilkan *output* proses algen
   - Tab hasil algen : berfungsi menampilkan *output* hasil algen
2. Panel *Input* Parameter Genetika berfungsi untuk meng-*input*-kan data parameter genetika
3. Tombol *Open Population* berfungsi untuk membuka *file* yang berisi populasi awal

4. Tombol *Generate* Populasi Awal digunakan untuk melakukan proses pembentukan (*generate*) populasi awal
5. Tombol *Save Population* berfungsi untuk menyimpan data populasi awal yang telah di-*generate*
6. Tombol *Generate* digunakan untuk melakukan proses algoritma genetika
7. Tombol *New Search* digunakan untuk melakukan proses baru
8. Panel *Result* berfungsi menampilkan hasil (solusi) dari proses algoritma genetika

4.4 Penerapan Aplikasi

Penerapan aplikasi dilakukan dengan memilih data – data yang digunakan. Data – data yang digunakan berupa data yang bertipe .sql. terdiri atas data jalan (*street.sql*) dan data terminal (*terminal.sql*). Tiap data memiliki fungsi masing – masing.

1. Data Jalan

   Data jalan kota Malang yang digunakan seluruhnya berjumlah 548 jalan yang tersusun dari 367 *node*. Data tersebut terdiri atas beberapa parameter yaitu nama jalan, panjang jalan dan *node*. Sebagian jalan dapat dilihat pada tabel 4.11.

Tabel 4.11 Data Jalan

No	NamaJalan	PanjangJalan	*Node*1	*Node*2
1	Tlogomas 1	0.24	1	2
2	Tlogomas 2	0.38	2	280
3	MT Haryono 1	0.46	3	361
4	MT Haryono 2	0.16	361	362
5	Mayjen Panjaitan 2	1.32	5	199
6	Sukarno Hatta 5	0.46	5	300
7	Bandung 2	0.29	6	198
8	Jaksa Agung Suprapto 1	0.30	7	8
9	Jaksa Agung Suprapto 2	0.48	7	9
10	Brigjen Slamet Riadi 1	0.20	7	201
11	Basuki Rahmat 2	0.50	8	48
12	Kahuripan 1	0.22	8	165
13	Semeru 1	0.18	8	181

14	Jaksa Agung Suprapto 3	0.50	9	10
15	Pattimura 1	0.18	9	169
16	Jaksa Agung Suprapto 4	0.34	10	11
17	Dr. Cipto 2	0.14	10	170
18	Letjen Sutoyo 1	0.53	11	12
19	Kaliurang	0.57	11	203
20	WR Suprtaman 1	0.21	11	261

2. Data Terminal

Data terminal adalah data yang berfungsi untuk menyimpan informasi terminal, subterminal dan APK. Dalam hal ini terminal, subterminal dan APK ditunjukkan dalam bentuk *node* dengan tipe data *integer*. Data tersebut terdiri atas nama pos dan node. Adapun data terminal dapat dilihat pada tabel 4.12.

Tabel 4.12 Data Terminal

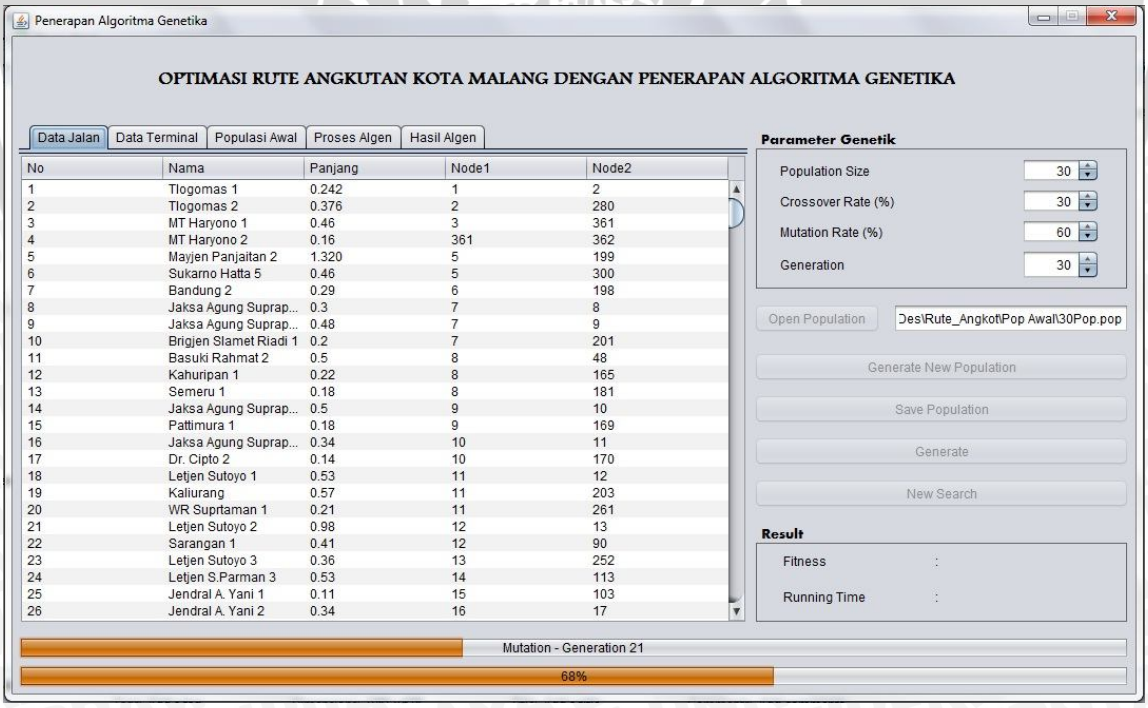
No	Node	NamaPos
1	89	Terminal Arjosari
2	1	Terminal Landungsari
3	74	Terminal Gadang
4	338	Sub.Terminal Madyopuro
5	172	Sub.Terminal Mulyorejo
6	132	Sub.Terminal Tlogowaru
7	360	Apk. Cemoro Kandang
8	138	Apk. Tirtosari
9	220	Apk. Joyogrand
10	41	Apk. Puncak Dieng
11	63	Apk. Mergan
12	285	Apk.Tawangmangu
13	355	Apk. Karang Besuki
14	353	Apk. Tidar
15	142	Apk. Pasar Sukun
16	354	Apk. Gasek
17	347	Apk. Tasikmadu
18	349	Apk. Karanglo Indah
19	358	Apk. Pasar Bunul

Hal yang pertama kali dilakukan saat program dijalankan adalah *input* parameter algoritma genetika yang terdiri atas *population size* (jumlah populasi),

crossover rate (prob. *Crossover)*, *mutation rate* (prob. Mutasi) dan *generation* (jumlah generasi). Antarmuka *input* parameter genetika dapat dilihat pada gambar 4.2

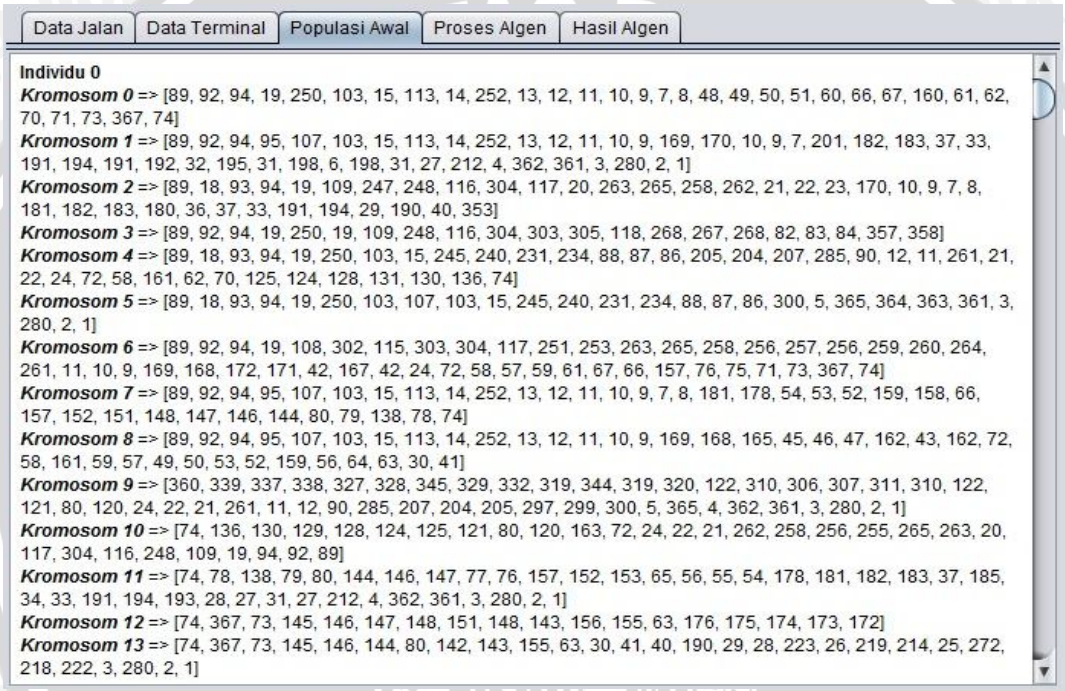
Gambar 4.2 *Input* Parameter Algoritma Genetika

Pada gambar 4.2 dapat dilihat bahwa *input* parameter algoritma genetika terdiri atas 30 *population size*, 30 % *crossover rate*, 60% *mutation rate* dan 30 *generation*. Dari *input* parameter genetik tersebut akan dilakukan proses algoritma genetika untuk sejumlah populasi individu sebanyak 30 dengan 30 kali generasi atau iterasi. Tampilan implementasi program dapat dilihat pada gambar 4.3.

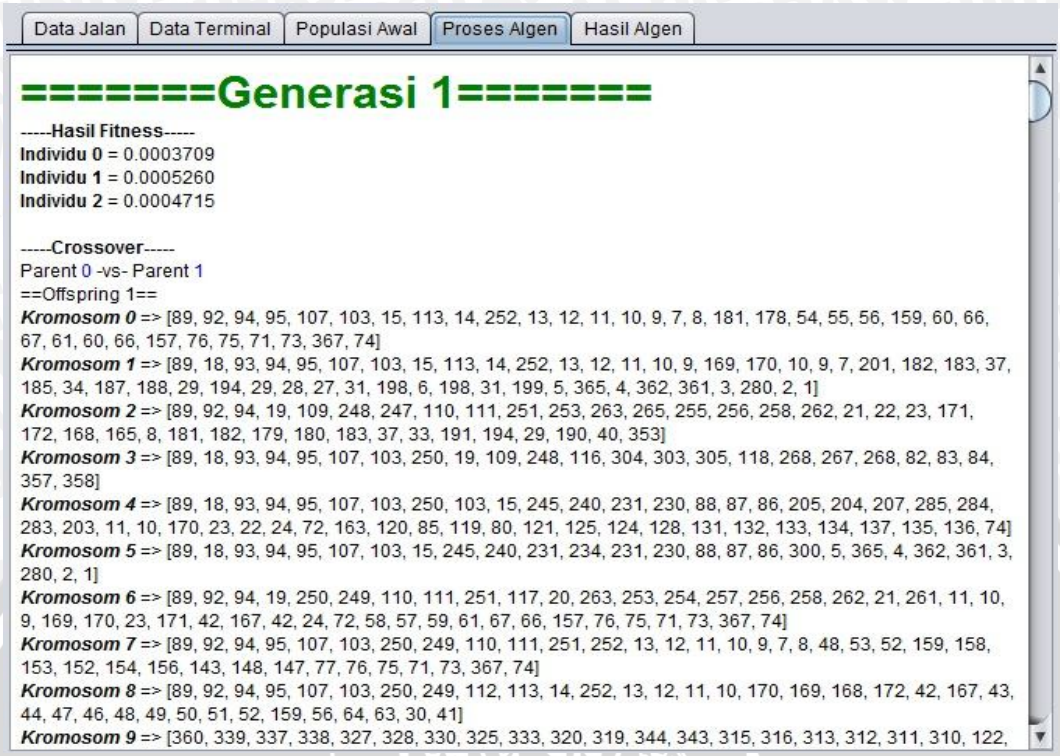


Gambar 4.3 Implementasi Program

Setelah *input* parameter genetika dilakukan, tahap selanjutnya yaitu *generate* populasi awal. Kemudian dilakukan proses *generate* algortima genetika untuk memperoleh solusi yang terbaik. Solusi terbaik diperoleh dari individu dalam popoulasi dengan nilai *fitness* terbaik dan merupakan individu yang paling optimal. Tampilan populasi awal, tampilan proses algoritma genetika dan tampilan *output* hasil dari optimasi rute angkutan kota Malang ditunjukkan masing – masing pada gambar 4.4, gambar 4.5 dan gambar 4.6.



Gambar 4.4 Tampilan Populasi Awal



Gambar 4.5 Tampilan Proses Algoritma Genetika

Gambar 4.6 Tampilan *Output* Hasil Optimasi Rute

4.5 Sistematika Pengujian

Subbab ini menjelaskan mengenai sistematika pengujian. Sesuai dengan rancangan pengujian yang telah dipaparkan dalam bab III, terdapat 3 macam pengujian, antara lain uji probabilitas *crossover* dan probabilitas mutasi, uji ukuran populasi dan uji jumlah generasi.

Rancangan uji coba ke-1 yaitu untuk mengetahui pengaruh probabilitas *crossover* dan mutasi terhadap nilai *fitness* yang dihasilkan. Pengujian dilakukan dengan terhadap nilai probabilitas *crossover* dan probabilitas mutasi dengan jumlah populasi dan jumlah generasi yang digunakan sama. Probabilitas crossover dan probabilitas mutasi akan diujikan adalah 10% hingga 100%. Untuk setiap percobaan dilakukan pengulangan sebanyak 5 kali dan diambil rata – rata nilai *fitnes.* Tabel rancangan pengujian pengaruh probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.16

Rancangan uji coba selanjutnya yaitu untuk mengetahui pengaruh jumlah populasi dan jumlah generasi terhadap nilai *fitness* yang dihasilkan. Pengujian dilakukan dengan memberikan nilai jumlah populasi atau jumlah generasi yang beragam dengan nilai probabilitas *crossover* dan probabilitas mutasi yang sama, yang diperoleh dari nilai probabilitas *crossover* dan probabilitas mutasi terbaik dalam pengujian ke-1. Untuk setiap percobaan dilakukan pengulangan sebanyak 5 kali dan diambil rata – rata nilai *fitnes.* Tabel rancangan pengujian pengaruh jumlah populasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.17. Sedangkan Tabel rancangan pengujian pengaruh jumlah generasi terhadap nilai *fitness* rata – rata dapat dilihat pada tabel 3.18

4.5.1 Sistematika Uji Probabilitas *Crossover* dan Probabilitas Mutasi

Pengujian pertama yaitu uji probabilitas *crossover* dan probabilitas mutasi. Pengujian ini dilakukan untuk mengetahui pengaruh perubahan nilai probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* rata – rata yang dihasilkan. Tujuan dari pengujian ini adalah untuk mengetahui apakah nilai probabilitas *crossover* dan probabilitas mutasi mempengaruhi nilai *fitness* dari proses algoritma genetika serta mengetahui berapa nilai probabilitas *crossover* dan probabilitas mutasi terbaik.

Adapun proses pengujiannya yaitu dengan mengkombinasikan 10 macam nilai probabilitas *crossover* dan 10 macam nilai probabilitas mutasi, dengan nilai antara 10% hingga 100%. Setiap satu kombinasi nilai probabilitas *crossover* dan probabilitas mutasi dilakukan 5 kali uji coba, dengan mengambil hasil nilai *fitness* rata – rata. Uji coba dilakukan dengan menggunakan ukuran populasi dan jumlah generasi tetap yaitu masing – masing 30 individu dan 100 generasi.

Hasil dari pengujian akan didapatkan nilai *fitness* rata – rata dari setiap satu kombinasi nilai probabilitas *crossover* dan probabilitas mutasi. Setiap nilai *fitness* rata – rata dari setiap kombinasi inilah yang akan dibandingkan, sehingga akan didapatkan analisa pengaruh dari perubahan nilai probabilitas *crossover* dan probabilitas mutasi. Analisa hasil pengujian dilakukan dengan tidak mempertimbangkan lama waktu pengujian.

4.5.2 Sistematika Uji Ukuran Populasi

Pengujian kedua yaitu uji ukuran populasi. Pengujian ini dilakukan untuk mengetahui pengaruh perubahan ukuran populasi terhadap nilai *fitness* rata – rata yang dihasilkan. Tujuan dari pengujian ini adalah untuk mengetahui apakah ukuran populasi mempengaruhi hasil dari proses algoritma genetika, dalam hal ini adalah nilai *fitness.*

Proses pengujian dilakukan dengan mengubah ukuran populasi dari 10 hingga 60 individu dalam satu populasi. Parameter genetika lain yang digunakan berturut – turut adalah pc : 90% , pm : 30% dan jumlah generasi : 100.

Setiap satu nilai perubahan ukuran populasi dilakukan pengulangan uji coba sebanyak 5 kali, dengan mengambil hasil nilai *fitness* rata – rata. Setiap nilai *fitness* rata – rata inilah yang akan dibandingkan untuk selanjutnya dianalisa. Analisa hasil pengujian dilakukan dengan tidak mempertimbangkan lama waktu pengujian.

4.5.3 Sistematika Uji Jumlah Generasi

Pengujian terakhir yaitu uji jumlah generasi. Pengujian ini dilakukan untuk mengetahui pengaruh perubahan jumlah generasi terhadap nilai *fitness* rata – rata yang dihasilkan. Tujuan dari pengujian ini adalah untuk mengetahui apakah

jumlah generasi mempengaruhi hasil dari proses algoritma genetika, dalam hal ini adalah nilai *fitness.*

Proses pengujian dilakukan dengan mengubah jumlah generasi dari 10, 30, 50, 80, 100 dan 150 generasi. Parameter genetika lain yang digunakan berturut – turut adalah pc : 90% , pm : 30% dan jumlah populasi : 30.

Setiap satu nilai perubahan jumlah generasi dilakukan pengulangan uji coba sebanyak 5 kali, dengan mengambil hasil nilai *fitness* rata – rata. Setiap nilai *fitness* rata – rata inilah yang akan dibandingkan untuk selanjutnya dianalisa. Analisa hasil pengujian dilakukan dengan tidak mempertimbangkan lama waktu pengujian.

4.6 Implementasi Pengujian

Pada subbab akan dijelaskan mengenai hasil uji coba beserta analisanya. Uji coba dilakukan sesuai dengan sistematika pengujian yang telah dijabarkan sebelumnya.

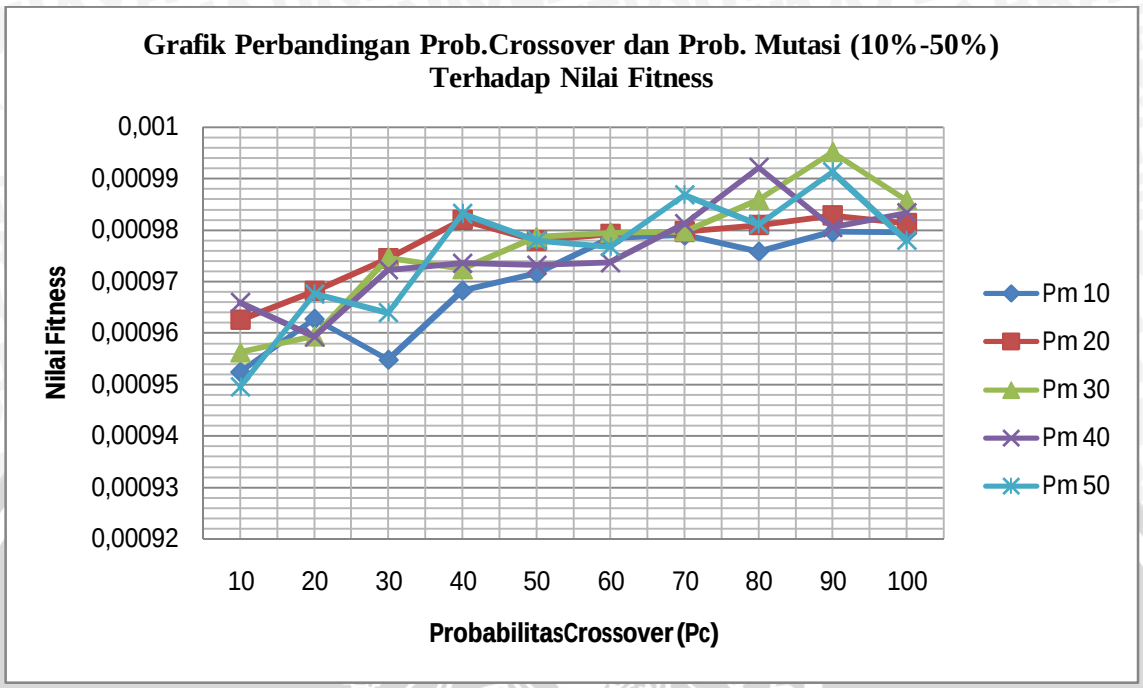
4.6.1 Hasil dan Analisa Uji Probabilitas *Crossover* dan Probabilitas Mutasi

Berdasarkan sistematika pengujian yang telah dijabarkan sebelumnya, pengujian terhadap pengaruh probabilitas crossover dan probabilitas mutasi dilakukan pada rentang nilai 10% - 100% untuk masing probabilitas crossover dan probabilitas mutasi. Setiap kombinasi nilai probabilitas crossover dan probabilitas mutasi dilakukan ujicoba sebanyak 5 kali dan diambil nilai rata – rata *fitness*. Hal ini dilakukan karena konsep dari algoritma genetika yang *random* menyebabkan rentang nilai yang dihasilkan beragam sehingga dilakukan beberapa kali uji coba untuk kemudian di rata – rata. Berikut adalah hasil uji coba probabilitas crossover dan probabilitas mutasi yang disajikan dalam tabel 4.13

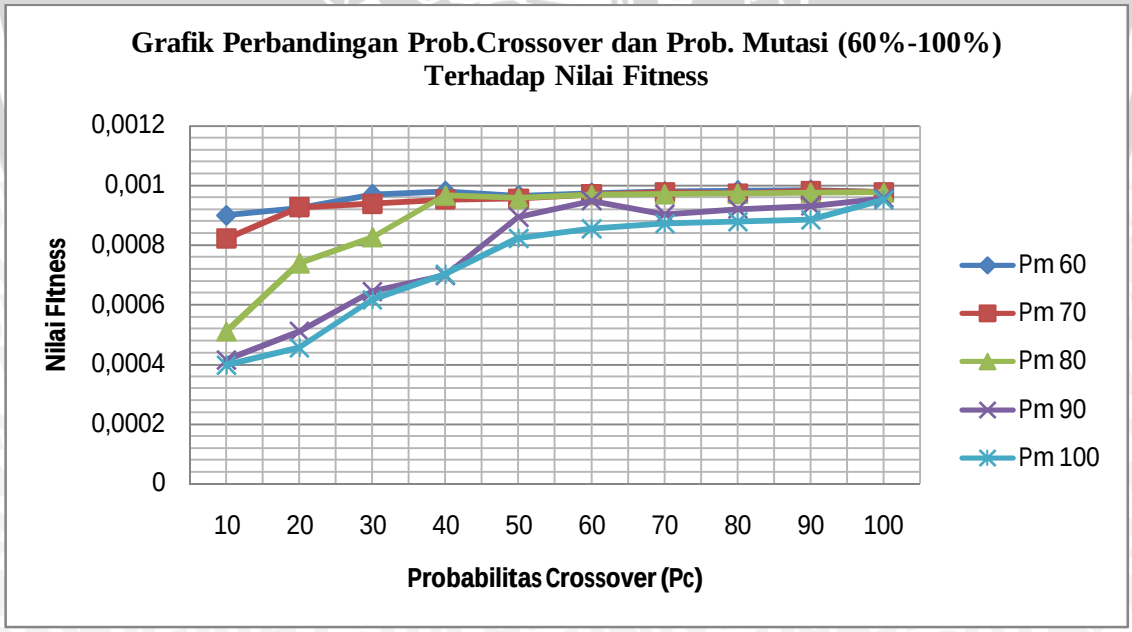
Tabel 4.13 Hasil Uji Coba Probabilitas Crossover dan Probabilitas Mutasi

Pop : 30 ; Generasi : 100										
Pm (%)	**Pc (%)**									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.0009524	0.00096272	0.00095478	0.00096824	0.00097158	0.00097844	0.00097896	0.00097584	0.00097964	0.00097946
20	0.00096256	0.0009681	0.00097444	0.00098198	0.00097792	0.00097924	0.00097966	0.00098094	0.00098282	0.00098138
30	0.00095626	0.00095932	0.00097458	0.0009724	0.00097864	0.00097946	0.00097972	0.00098586	0.00099514	0.0009858
40	0.00096592	0.00095926	0.00097226	0.00097358	0.0009732	0.00097368	0.00098118	0.00099216	0.00098054	0.00098332
50	0.0009496	0.0009676	0.00096396	0.00098318	0.00097798	0.00097676	0.0009868	0.00098112	0.0009913	0.00097804
60	0.0009004	0.00092514	0.00097078	0.0009805	0.00096554	0.00097452	0.00098	0.00098294	0.00098486	0.00097746
70	0.00082246	0.00092734	0.00093962	0.00095276	0.00095548	0.00096978	0.00097752	0.00097406	0.00098162	0.00097806
80	0.0005104	0.00073964	0.00082618	0.0009675	0.00095742	0.00096964	0.00097136	0.00097276	0.00097642	0.00097778
90	0.00041528	0.00051066	0.00064498	0.00069806	0.00089402	0.00094866	0.00090322	0.00092098	0.00093082	0.00095676
100	0.0003985	0.00045566	0.00061712	0.0007023	0.00082288	0.00085624	0.00087216	0.00087928	0.00088652	0.00095212

Berdasarkan tabel 4.11 dapat dilihat bahwa nilai *fitness* terbesar adalah 0.00099514. Nilai ini didapat pada saat probabilitas crossover 90% dan probabilitas mutasi 30%. Adapun grafik pengaruh probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* disajikan pada gambar 4.7 dan gambar 4.8.



Gambar 4.7 Grafik Perbandingan Probabilitas *Crossover* dan Probabilitas Mutasi (10%-50%) Terhadap Nilai *Fitness*



Gambar 4.8 Grafik Perbandingan Probabilitas Crossover dan Probabilitas Mutasi (60%-100%) Terhadap Nilai Fitness

Nilai *fitness* didapat dengan terlebih dahulu menghitung *frekuensi* perlanggaran yang dilakukan oleh individu dengan diberikan suatu penalti pelanggaran. Semakin banyak pelanggaran yang dilakukan maka semakin besar penalti yang diperoleh oleh suatu individu. Individu yang dianggap baik, yang akan menjadi solusi adalah individu yang sedikit melakukan pelanggaran. Oleh karena nilai *fitness* dan penalti pelanggaaran berbanding terbalik, maka berdasar persamaan fungsi fitness yang dijabarkan dalam bab III, individu yang baik adalah individu dengan nilai *fitness* yang tinggi.

Berdasarkan kedua grafik perbandingan probabilitas *crossover* dan probabilitas mutasi pada gambar 4.7 dan 4.8, secara keseluruhan dapat dilihat bahwa seiring bertambahnya nilai probabilitas *crossover* grafik yang dihasilkan cenderung naik. Hal ini berarti bahwa seiring bertambahnya nilai probabilitas *crossover* maka nilai *fitness* cenderung semakin besar. Perubahan nilai *fitness* berdasar probabilitas *crossover* juga dipengaruhi oleh probabilitas mutasi sehingga tidak semua nilai *fitness* naik untuk setiap kenaikan probabilitas *crossover*. Berdasarkan grafik pada gambar 4.7, menunjukkan hampir semua grafik menunjukkan kondisi yang *fluktuatif*. Akan tetapi kondisi fluktuatif jelas terlihat ketika probabilitas mutasi 50%, kenaikan nilai probabilitas *crossover* mengakibatkan nilai *fitness* yang dihasilkan mengalami penurunan dan kenaikan. Penurunan terjadi ketika nilai probabilitas *crossover* 30%, 60% dan 80%.

Perubahan nilai *fitness* berdasar kenaikan probabilitas *crosssover* dapat disebabkan karena semakin seringnya suatu individu mengalami proses persilangan sehingga variasi individu baru akan semakin banyak. Telah diketahui bahwa semakin besar nilai probabiltas *crossover* maka peluang individu yang mengalami proses *crossover* akan semakin besar, dan tentunya akan semakin banyak individu – individu baru yang dihasilkan. Dengan demikian nilai *fitness* juga semakin bervariasi, sehingga peluang untuk mendapatkan individu dengan nilai *fitness* yang besar juga akan semakin besar.

Berbeda halnya dengan probabilitas *crossover*, dari grafik diatas terlihat bahwa perubahan nilai probabilitas mutasi menghasilkan grafik tak tentu, dalam artian mengalami kenaikan dan penurunan dalam kondisi probabilitas *crossover* sama. Grafik *fluktuatif* terlihat jelas pada gambar 4.7, pada grafik ini seiring

bertambahnya nilai probabilitas mutasi tidak diiringi bertambahnya nilai *fitness*. Nilai *fitness* yang dihasilkan adalah tak tentu, dalam artian mengalami kenaikan dan penurunan. Hal ini terlihat jelas ketika probabilitas *crossover* 30%, pada keadaan ini nilai *fitness* mengalami kenaikan ketika probabilitas mutasi 10% hingga 20%. Kemudian mengalami penurunan ketika 30% hingga 50%. Penurunan yang cukup signifikan terjadi ketika probabilitas mutasi 50%. Penurunan nilai *fitness* terus terjadi hingga nilai probabilitas mutasi mencapai 100%. Pada grafik gambar 4.8 terlihat bahwa seiring bertambahnya nilai probabilitas mutasi menghasilkan nilai *fitness* yang cenderung turun. Jelas terlihat bahwa ketika dalam kondisi probabilitas *crossover* sama, nilai *fitness* cenderung mengalami penurunan. Sehingga dapat disimpulkan semakin besar nilai probabilitas mutasi maka nilai *fitness* cenderung mengalami penurunan. Hal ini dapat disebabkan karena metode mutasi yang *random* dan dengan menyisipkan gen – gen baru, sehingga memungkinkan individu memilki jarak tempuh yang semakin jauh dan memungkinkan pula naiknya jumlah tumpukan jalur. Tentunya hal ini berakibat semakin besarnya penalti pelanggaran, sehingga nilai *fitness* semakin kecil.

Berdasarkan hasil uji probabilitas *crossover* dan probabilitas mutasi diatas, didapatkan nilai *fitness* terbaik yaitu pada nilai probabilitas *crossover* 90% dan probabilitas mutasi 30%. Dari pengujian tersebut didapatkan optimasi rute angkutan kota Malang dengan hasil 6 dari 25 trayek tidak memenuhi kriteria aturan penalti berdasarkan jarak tempuh. Keenam jenis trayek tersebut adalah AL, ASD, JPK, LG, MK dan MT. Hasil uji probabilitas *crossover* dan probabilitas mutasi terhadap jarak tempuh dapat dilihat pada tabel 4.14.

Tabel 4.14 Hasil Uji Probabilitas *Crossover* Dan Probabilitas Mutasi Terhadap Jarak Tempuh

No.	Jenis Trayek	Jarak Tempuh Yang Diperoleh (Km)	Jarak Tempuh Aturan (Km)	Selisih Jarak (Km)
1.	AG / AH	14,474	16	1,526
2.	AL	**20,952**	17,8	**-3,152**
3.	AT	10,99	18	7,01

4.	ABB	9,202	16	6,98
5.	ABG / ABH	24, 834	26	1,166
6.	ADL	11,122	14,8	3,678
7.	AJG / AJH	18,032	18,2	0,168
8.	AMG / AMH	14,79	16	1,21
9.	ASD	**16,152**	11,5	**-4,652**
10.	CKL	17,268	22	4,732
11.	GA / HA	14,302	16,3	1,998
12.	GL / HA	14,7	15,8	1,1
13.	GM / HM	8,866	9	0,134
14.	GML / HML	14,814	18	3,186
15.	JDM	8,7	13	4,3
16.	JPK	**12,806**	10	**-2,806**
17.	LG / LH	**19,262**	16,8	**-2,462**
18.	LDG / LDH	14,078	15,2	1,122
19.	MK	**16,98**	11,5	**-5,48**
20.	MM	12,76	15,2	2,44
21.	MT	**14,034**	9	**-5,034**
22.	MKS	5,49	8	2,51
23.	TGT	6,26	6,5	0.24
24.	TSG	6,35	10	3,65
25.	TST	15,058	16	0,942

Hasil yang diperoleh dari uji probabilitas *crossover* dan probabilitas mutasi diatas berdasarkan jumlah tumpukan jalur (*intersection*) adalah terdapat 93 pelanggaran terhadap aturan penalti kedua. Hasil uji probabilitas *crossover* dan probabilitas mutasi terhadap tumpukan jalur dapat dilihat pada tabel 4.15

Tabel 4.15 Hasil Uji Probabilitas Crossover Dan Probabilitas Mutasi Terhadap Tumpukan Jalur

Total Tumpukan Jalur Yang Diperoleh	**Total Tumpukan Jalur (Keadaan nyata)**
95	176

Hasil tabel 4.14 dan 4.15 menunjukkan bahwa dalam penyelesaian permasalahan optimasi rute angkutan kota dengan penerapan algoritma genetika menunjukkan bahwa hasil yang diperoleh belum benar – benar optimal.

Ditunjukkan dengan masih banyak jenis trayek yang belum memenuhi jarak tempuh maksimal yang telah ditetapkan yaitu sejumlah 6 trayek. Hal ini dapat disebabkan ketika percobaan pengujian, parameter genetika sebagai parameter uji belum benar – benar mendapatkan hasil pada keadaan global optimum (masih dalam keadaan lokal optimum), dimana suatu nilai *fitness* telah konvergen akan tetapi belum dapat mencapai optimal. Tentunya hal ini mempengaruhi nilai *fitness* yang diperoleh yaitu nilai *fitness* tidak dapat mendekati nilai *fitness* terbaik yaitu 1.

Hasil tabel 4.14 ditunjukkan bahwa selisih jarak yang diperoleh dengan jarak tempuh aturan belum terlihat secara siginifikan. Begitu juga pada tabel 4.15 total tumpukan jalur yang diperoleh masih terbilang cukup banyak mengalami penumpukan. Dari hasil kedua tabel yaitu tabel 4.14 dan 4.15, belum terlihat parameter yang mempengaruhi hasil optimasi secara dominan. Hal ini dapat disebabkan pada pemberian skor pada aturan penalti untuk jarak tempuh maksimal (aturan 1) dan aturan tumupukan jalur (aturan 2).

Dalam aturan penalti tersebut skor yang diberikan adalah sama yaitu 10. Nilai 10 menyatakan bahwa aturan tersebut merupakan aturan yang wajib untuk dilakukan. Oleh karena parameter penelitian yang ditekankan adalah jarak tempuh maksimal dan jumlah tumpukan jalur maka diberikan skor penalti yang sama. Akan tetapi pada kenyataan hasil uji hal ini mengakibatkan tidak terlihat aturan yang lebih dominan dalam melakukan pelanggaran.

Pada penelitian ini nilai *fitness* diperoleh dengan memberikan skor penalti pada suatu individu yang melakukan pelanggaran dengan dikalikan total frekuensi pelanggaran. Pemberian skor didasarkan pada tingkat kewajiban suatu aturan untuk dilakukan. Oleh karena skor penalti menjadi dasar pembentukan nilai *fitness*, maka penentuan skor penalti juga berpengaruh terhadap baik buruknya suatu nilai *fitness*. Semakin besar skor penalti yang ditentukan maka akan semakin memperkecil nilai *fitness*, karena nilai *fitness* berbanding terbalik dengan skor penalti. Akan tetapi suatu nilai *fitness* yang baik dengan cara perubahan skor penalti tidak selalu menjamin hasil rute yang optimal, karena hasil optimasi rute juga dipengaruhi dengan parameter genetika yang lain.

4.6.2 Hasil dan Analisa Uji Ukuran Populasi

Berdasarkan sistematika pengujian yang telah dijabarkan sebelumnya, pengujian terhadap pengaruh ukuran populasi dilakukan dengan menggunakan 6 nilai uji yaitu 10, 20, 30, 40, 50 dan 60 . Untuk setiap nilai uji dilakukan 5 kali uji coba untuk kemudian diambil rata – rata. Berikut adalah hasil uji ukuran populasi yang disajikan dalam tabel 4.16.

Tabel 4.16 Hasil Uji Coba Ukuran Populasi

Populasi	Pc : 90% - Pm : 30% : Generasi : 100					
	Percobaan Ke -					
	1	2	3	4	5	Fitness Rata - Rata
10	0.0009258	0.0009258	0.0009232	0.0009302	0.0009212	0.00092524
20	0.0009513	0.0009567	0.0009545	0.0009545	0.0009545	0.0009543
30	0.0009878	0.0009857	0.0009905	0.0009003	0.0009003	0.00095292
40	0.0009979	0.0009898	0.0009898	0.0009788	0.0009788	0.00098702
50	0.0009923	0.001006	0.001006	0.0009989	0.0009989	0.00100042
60	0.0009998	0.0009998	0.0009882	0.0009986	0.0009986	0.000997

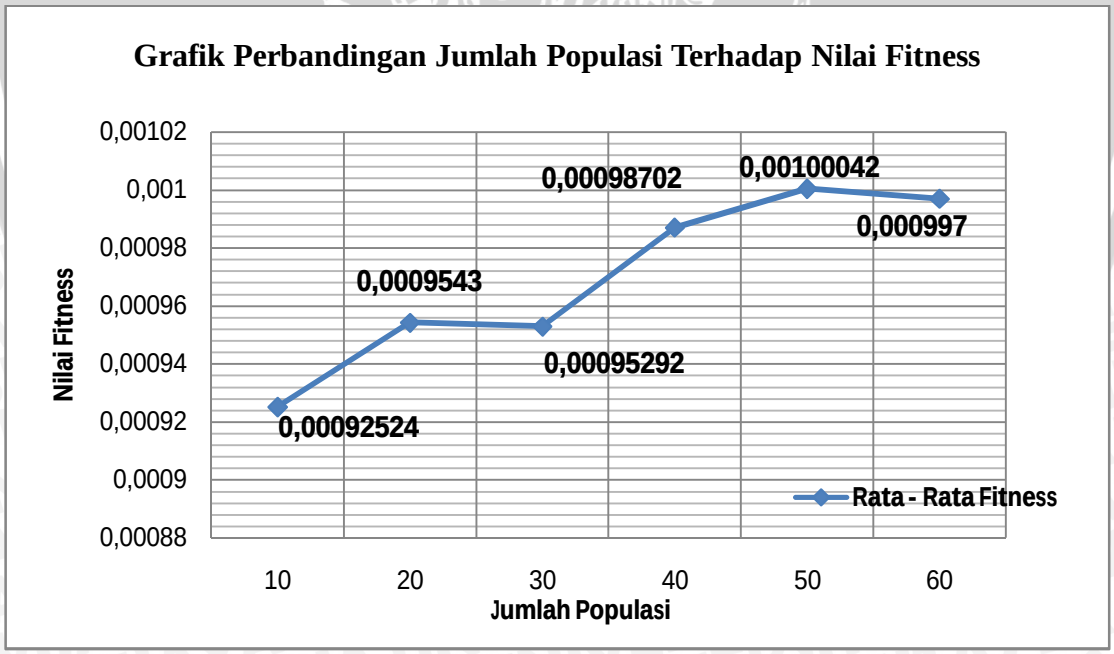
Pada tabel 4.16 terlihat bahwa nilai *fitness* terbesar dihasilkan oleh pengujian dengan ukuran populasi 50 yaitu 0.00100042. Sedangkan nilai *fitness* terkecil diperoleh pada saat ukuran populasi 10. Hasil uji coba ukuran populasi pada tabel 4.16, dengan 5 kali percobaan dan menggunakan populasi awal serta nilai parameter genetika (Pc, Pm dan Generasi) yang sama diperoleh nilai *fitness* yang tak tentu. Dalam artian nilai *fitness* yang diperoleh terkadang tidak mengalami perubahan (tetap) dan terkadang juga mengalami perubahan, baik perubahan kearah penurunan nilai *fitness* ataupun kenaikan nilai *fitness*.

Dengan menggunakan populasi awal dan nilai parameter genetika yang sama tidak dapat dipastikan akan memperoleh hasil (*nilai fitness*) yang sama. Hal ini dapat terjadi karena konsep algoritma genetika adalah *random*. Dimana akan dibangkitkan suatu bilangan *random* pada setiap individu, untuk selanjutnya dilakukan pengecekan terhadap nilai probabilitas *crossover* dan probabilitas mutasi. Ketika proses *generate* dilakukan maka akan menghasilkan nilai *random* yang baru. Hasil dari pengecekan terhadap nilai probabilitas *crossover* dan probabilitas mutasi akan mempengaruhi apakah suatu individu dalam populasi

mengalami persilangan dan/atau mutasi ataukah tidak. Tentunya hal tersebut akan berpengaruh terhadap nilai *fitness* yang dihasilkan. Nilai *fitness* dari satu individu dengan yang lain memungkinkan tidak sama.

Nilai *fitness* yang sama dalam 2 percobaan belum tentu menghasilkan jarak tempuh yang sama pada jenis trayek yang sama. Begitu juga dengan jumlah tumpukan (*intersection*) jalur dalam 1 individu, belum tentu menghasilkan jumlah tumpukan yang sama. Hal tersebut terlihat ketika percobaan ke-2 dan ke-3 pada pengujian populasi 50. Sesuai dengan rumus *fitness* yang telah dijabarkan dalam bab III, nilai *fitness* dalam permasalahan ini dipengaruhi oleh jumlah frekuensi pelanggaran yang dilakukan oleh suatu individu. Ketika jumlah frekuensi pelanggaran yang dilakukan adalah sama, maka akan menghasilkan nilai *fitness* yang sama pula. Begitu juga sebaliknya, ketika jumlah frekuensi pelanggaran yang dilakukan berbeda, maka akan menghasilkan nilai *fitness* yang berbeda pula.

Adapun grafik pengaruh probabilitas ukuran populasi terhadap nilai *fitness* disajikan pada gambar 4.9.



Gambar 4.9 Grafik Perbandingan Ukuran Populasi dengan Nilai *Fitness*

Grafik perbandingan ukuran populasi dengan nilai *fitness* pada gambar 4.9 cenderung naik. Hal ini menunjukkan bahwa seiring bertambahnya ukuran

populasi nilai *fitness* juga semakin besar. Dalam grafik diatas menunjukkan bahwa nilai *fitness* mengalami penurunan ketika jumlah populasi 30 dan naik kembali hingga jumlah populasi 50. Akan tetapi mengalami penurunan kembali ketika jumlah populasi 60.

Secara teori ukuran populasi yang besar memungkinkan terdapat variasi individu yang beragam dan ukuran populasi yang kecil memungkinkan variasi individu sedikit (tidak beragam). Akan tetapi dari grafik diatas tidak menjamin bahwa ukuran populasi yang besar menghasilkan nilai *fitness* yang besar dan ukuran populasi yang kecil menghasilkan nilai *fitness* yang kecil. Semakin besar ukuran populasi yang digunakan maka kemungkinan untuk menemukan solusi cenderung menjadi lambat. Sedangkan ukuran populasi yang terlalu kecil memungkinkan semakin kecilnya ruang pencarian yang dapat dieksplorasi ketika proses rekombinasi (*crossover* dan mutasi).

Hal ini dapat disebabkan pada saat pemilihan kandidat *parent* untuk proses *crossover*. Oleh karena pemilihan dilakukan secara *random* dengan berdasarkan porbabilitas *crossover*, maka tidak dapat dipastikan bahwa individu yang terpilih adalah individu dengan nilai *fitness* besar dari proporsi jumlah individu yang besar. Sehingga apabila kandidat *parent* yang terpilih adalah individu dengan nilai *fitness* kecil, anak yang dihasilkan dari proses rekombinasi juga memiliki kemungkinan bernilai *fitness* kecil. Hal yang sama pada ukuran populasi yang kecil dimana variasi individu-individu di dalamnya sedikit. Individu yang akan terpilih sebagai kandidat *parent* akan memiliki variasi terbatas, dan anak yang dihasilkan bisa jadi memiliki sifat yang mirip dengan nilai *fitness* yang hampir dekat (tidak mengalami improvisasi).

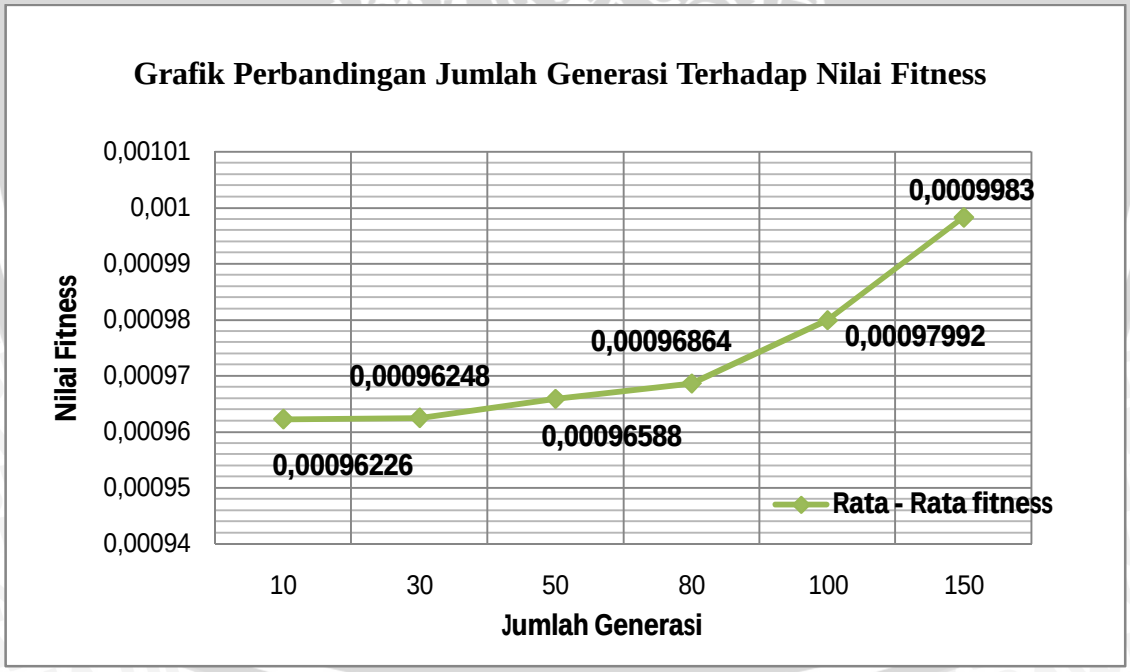
4.6.3 Hasil dan Analisa Uji Jumlah Generasi

Berdasarkan sistematika pengujian yang telah dijabarkan sebelumnya, pengujian terhadap pengaruh jumlah generasi dilakukan dengan menggunakan 6 nilai uji yaitu 10, 30, 50, 80, 100 dan 150 . Untuk setiap nilai uji dilakukan 5 kali uji coba untuk kemudian diambil rata – rata. Berikut adalah hasil uji ukuran populasi yang disajikan dalam tabel 4.17.

Tabel 4.17 Hasil Uji Coba Jumlah Generasi

Generasi	Populasi : 30 - Pc : 90% - Pm : 30%					
	Percobaan Ke -					
	1	2	3	4	5	Fitness Rata -Rata
10	0.000965	0.000965	0.000965	0.0009553	0.000961	0.00096226
30	0.000965	0.000965	0.000965	0.0009581	0.0009593	0.00096248
50	0.000965	0.000965	0.0009672	0.0009672	0.000965	0.00096588
80	0.000965	0.000968	0.000968	0.0009711	0.0009711	0.00096864
100	0.0009786	0.0009786	0.0009832	0.0009832	0.000976	0.00097992
150	0.0009983	0.0009983	0.000998	0.000998	0.001011	0.0009983

Pada tabel 4.17 terlihat bahwa nilai *fitness* terbesar dihasilkan oleh pengujian dengan jumlah generasi 150 yaitu 0.0009983. Sedangkan nilai *fitness* terkecil diperoleh pada saat jumlah generasi 10. Adapun grafik pengaruh jumlah generasi terhadap nilai *fitness* disajikan pada gambar 4.10.



Gambar 4.10 Grafik Perbandingan Jumlah Generasi dengan Nilai *Fitness*

Grafik pada gambar 4.10 dapat dilihat bahwa nilai *fitness* tidak mengalami perubahan yang signifikan ketika jumlah generasi 10 hingga 80. Selanjutnya mengalami kenaikan ketika jumlah generasi 100. Kenaikan mulai terlihat

signifikan ketika jumlah generasi 150. Sehingga secara keseluruhan dapat dikatakan bahwa nilai *fitness* cenderung naik seiring bertambahnya jumlah generasi.

Jumlah generasi yang tinggi akan mengakibatkan proses evolusi semakin sering dilakukan. Pada setiap satu generasi, akan dilakukan proses rekombinasi yang terdiri atas *crossover* dan mutasi. Sehingga semakin banyak jumlah generasi maka proses rekombinasi akan semakin sering dilakukan. Tentunya hal ini akan berpengaruh pula terhadap individu – individu baru yang dihasilkan. Semakin banyak melakukan proses *crossover* dan mutasi maka individu – individu baru yang dihasilkan akan semakin bervariasi dan memungkinkan pula bervariasinya nilai *fitness* yang dihasilkan. Dengan begitu akan memberikan peluang yang besar untuk mendapatkan nilai *fitness* yang baik.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa pembahsan diatas maka dapat disimpulkan bahwa :

1. Algoritma genetika dapat diterapkan dalam permasalahan optimasi rute angkutan kota, studi penentuan rute trayek angkutan kota Malang dengan berdasarkan pada jarak tempuh dan penumpukan (*intersection*) jalur antar trayek. Dengan metode yang digunakan adalah pengokodean permutasi, *crossover* modifikasi, mutasi *random* dan seleksi *rank base fitness.*
2. Penyelesaian permasalahan optimasi rute angkutan kota Malang dengan penerapan algoritma genetika dipengaruhi oleh beberapa parameter algoritma genetika, yaitu :
   - Pengaruh parameter probabilitas *crossover* menunjukkan bahwa semakin besar nilai probabilitas *crossover* maka nilai *fitness* cenderung semakin tinggi (baik)
   - Pengaruh parameter jumlah generasi menunjukkan bahwa semakin besar jumlah generasi yang digunakan, maka nilai *fitness* juga cenderung semakin tinggi (baik).
   - Pengaruh parameter probabilitas mutasi menunjukkan perubahan nilai probabilitas mutasi yang semakin tinggi mengakibatkan penurunan nilai *fitness.*
   - Sedangkan pengaruh jumlah populasi menunjukkan bahwa jumlah populasi antara 10 hingga 50 menghasilkan nilai *fitness* yang cenderung naik. Akan tetapi jumlah populasi diatas 50 mengakibatkan penurunan nilai *fitness.*
3. Pada permasalahan optimasi rute angkutan kota dengan menggunakan 548 data jalan terhubung yang terdiri atas 367 *node* percabangan, algoritma genetika mampu menyelesaikan permasalahan optimasi rute angkutan kota Malang dengan hasil pengujian probabilitas crossover dan probabilitas mutasi terbaik yang diperoleh yaitu jarak tempuh trayek yang memenuhi aturan

sebanyak 19 dari 25 trayek. Sehingga terdapat 6 trayek yang tidak memenuhi aturan. Dan terdapat 95 total tumpukan jalur dari 176 tumpukan jalur pada keadaan yang sebenarnya.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan diatas saran yang dapat diberikan untuk pengembangan lebih lanjut adalah sebagai berikut :

1. Proses pengujian parameter genetika dapat dilakukan pengembangan dengan melakukan beberapa kali uji coba dahulu untuk mendapatkan nilai *fitness* pada kondisi global optimal sehingga diharapkan dapat memperoleh hasil yang lebih baik lagi
2. Penelitian ini dapat dilakukan pengembangan untuk meyelesaikan permasalahan optimasi rute angkutan kota Malang dengan menggunakan kondisi yang sebenarnya yaitu dengan rute keberangkatan tidak sama dengan rute pulang dan menggunakan jalan 2 arah.
3. Penelitian selanjutnya juga dapat dilakukan penambahan parameter / kriteria penelitian terkait optimasi rute angkutan kota dengan melihat dari beberapa aspek, antara lain : aspek pendapatan dari pihak supir angkutan kota, mempertimbangkan rute – rute strategis yang harus dilewati, jumlah penumpang dan jumlah armada yang dibutuhkan dalam setiap trayek. Sehingga diharapkan dapat memperoleh hasil yang benar - benar optimal dan memenuhi kondisi yang sebenarnya.

DAFTAR PUSTAKA

[ADH-09] Adhiansyah, Fafan. 2009. *Sistem Rekomendasi Pemilihan Angkutan Umum Menggunakan Algoritma Genetika*. SKRIPSI. Fakultas MIPA. Universitas Brawijaya.

[ADI-12] Adiwijaya. 2012. *BAB IV Teori Graf*. http://deky.students.uii.ac.id/files/2012/01/bab-iv-teori-graf.pdf. Diakses Tanggal 3 Mei 2012.

[ANO-12] Anonim. 2012. *Bab 7 Algortima Genetika*. http://lecturer.eepis-its.edu/~entin/Kecerdasan%20Buatan/Buku/Bab%207%20Algoritma%20Genetika.pdf . Diakses tanggal 29 Februari 2012.

[BAS-03] Basuki, Achmad. 2003. *Strategi Menggunakan Algoritma Genetika*. http://lecturer.eepis-its.edu/~basuki/lecture/StrategiAlgoritmaGenetika.pdf. Politeknik Elektronika Negeri Surabaya PENS-ITS. Diakses tanggal 3 Mei 2012

[BUD-07] Budyanto. 2007. *Menentukan Jalur Kereta Api yang Optimal Menggunakan Algoritma Genetika*. SKRIPSI. Fakultas MIPA. Universitas Brawijaya.

[CIP-08] Citpayani, Indah. 2008. *Penerapan Algoritma Genetika untuk Kompresi Citra Fraktal*. SKSIPSI. Fakultas MIPA. Universitas Brawijaya.

[COX-05] Cox, Earl. 2005 . *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*. Elsevier Inc. San Fransisco

[DIN-12] Dinas Kependudukan Dan Pencatatan Sipil. 2012. www.pemkot-malang.go.id. Diakses tanggal 1 Maret 2012.

[EIB-03] Eiben, A., & Smith, J. 2003. *Introduction to Evolutionary Computing*. Springer. Amsterdam.

[GEN-00] Gen, Mitsuo and Cheng, Runwei. 2000. *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons. New York.

[HAN-02] Hannawati, Anies dan Thiang, Eleazer. 2002. *Pencarian Rute Optimum Menggunakan Algoritma Genetika*. Jurnal Teknik Elektro Vol 2, No.2, September 2002:78-83.

[HAN-09] Handajani, Mudjiastuti. 2009. *Analisis Gradien Kepadatan Penduduk dan Konsumsi BBM*. Jurnal Teknik Sipil & Perencanaan, Nomor 2 Volume 11 – Juli 2009, hal : 141 – 148.

[KAR-11] Karas, Ismail Rakip and Atila Umit. 2011. *A Genetic Algorithm Approach For Finding The Shortest Driving Time On Mobile Devices.* Scientific Research and Essays Vol 6(2), pp. 394-405.

[KEP-03] Keputusan Menteri Perhubungan Nomor KM 35 Tahun 2003. 2003. http://hukum.unsrat.ac.id/men/menhub_35_2003.pdf. Diakses tanggal 1 Maret 2012

[KUS-03] Kusumadewi. 2003. *Artifial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.

[KUS-05] Kusumadewi, Sri dan Purnomo, Hari. 2005. *Penyelesaian Masalah Optimasi dengan Teknik – Teknik Heuristik*. Graha Ilmu. Yogyakarta.

[MIC-99] Michalewicz, Z. 1999. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer Verlag Berlin Heidelberg. New York.

[OBI-98] Obitko, Marek. 1998. *Genetic Algorithms*. http://www.obitko.com/tutorials/genetic-algorithms/. Diakses tanggal 1 Maret 2012

[PEM-12] Pemerintah Kota Malang. 2012. *Jalur Angkutan di Kota Malang*. www.pemkot-malang.go.id. Diakses tanggal 1 Maret 2012.

[PER-11] Peraturan Daerah Kota Malang Nomor 5 Tahun 2011, Salinan Nomor 2/E. 2011. http://www.djpp.depkumham.go.id/files/ld/2011/KotaMalang-2011-5.pdf. Diakses tanggal 25 April 2012.

[PRA-11] Pranata, Lamhot. 2011. *Penerapan Algoritma Genetika Pada Penentuan Komposisi Pakan Ikan*. SKRIPSI. Fakultas MIPA. Universitas Brawijaya.

[PUT-12] Putra, Maifil Eka. 2012. *Malangnya Kota Malang*. http://pirac.org/2012/06/06/malangnya-kota-malang/. Diakses tanggal 12 Desember 2012.

[ROB-06] Robandi, Imam. 2006. *Desain Sistem Tenaga Modern : Optimasi, Logika Fuzzy*. Algortima Genetika. Penerbit Andi : Yogyakarta.

[SAP-07] Saptono, Fajar dan Hidayat, Taufiq. 2007. *Perancangan Algoritma Genetika untuk Menentukan Jalur Terpendek*. SNATI 2007, 1907-5022.

[SAR-11] Saranta, Yan Mahdi. 2011. *Pencarian Rute Tercepat Kendaraan Bermotor Roda Empat Menggunakan Algoritma Genetika*. SKSIPSI. Fakultas MIPA. Universitas Brawijaya.

[SET-03] Setiawan, Kuswara. 2003. *Paradigma Sistem Cerdas*. Bayumedia. Surabaya.

[SUY-05] Suyanto. 2005. *Algoritma Genetika dalam Matlab*. Penerbit Andi. Yogyakarta.

[SUY-07] Suyanto. 2007. *Artificial Intelligence (Searching, Reasoning, Planning and Learning)*. Penerbit Informatika. Bandung.

[WID-12] Widodo, Thomas Sri. 2012. *Komputasi Evolusioner (Algoritma Genetik, Pemrograman Genetik dan Pemrograman Evolusioner)*. Graha Ilmu. Yogyakarta.

LAMPIRAN 1

Tabel Data Jalan Kota Malang

No	NamaJalan	PanjangJalan (km)	Node1	Node2
1	Tlogomas 1	1.24	1	2
2	Tlogomas 2	0.38	2	280
3	MT Haryono 1	0.46	3	361
4	MT Haryono 2	0.16	361	362
5	Mayjen Panjaitan 2	1.32	5	199
6	Sukarno Hatta 5	0.46	5	300
7	Bandung 2	0.29	6	198
8	Jaksa Agung Suprapto 1	0.30	7	8
9	Jaksa Agung Suprapto 2	0.48	7	9
10	Brigjen Slamet Riadi 1	0.20	7	201
11	Basuki Rahmat 2	0.50	8	48
12	Kahuripan 1	0.22	8	165
13	Semeru 1	0.18	8	181
14	Jaksa Agung Suprapto 3	0.50	9	10
15	Pattimura 1	0.18	9	169
16	Jaksa Agung Suprapto 4	0.34	10	11
17	Dr. Cipto 2	0.14	10	170
18	Letjen Sutoyo 1	0.53	11	12
19	Kaliurang	0.57	11	203
20	WR Suprtaman 1	0.21	11	261
21	Letjen Sutoyo 2	0.98	12	13
22	Sarangan 1	0.41	12	90
23	Letjen Sutoyo 3	0.36	13	252
24	Letjen S.Parman 3	0.53	14	113
25	Jendral A. Yani 1	0.11	15	103
26	Jendral A. Yani 2	0.34	16	17
27	Cakalang	1.74	16	96
28	Raden Intan 1	0.38	17	18
29	Raden Intan 2	0.75	18	89
30	Laksda Adi Sucipto 3	0.82	19	108
31	Laksda Adi Sucipto 4	0.09	19	250
32	Tumenggung Suryo 3	0.06	20	263
33	Sulfat 1	0.27	20	266
34	Panglima Sudirman 1	0.32	21	22
35	Hamid Rusdi	1.20	21	359

No	NamaJalan	PanjangJalan (km)	Node1	Node2
36	Dr. Cipto 1	0.26	22	23
37	Panglima Sudirman 2	0.51	22	24
38	Cokroaminoto 1	0.16	23	171
39	Panglima Sudirman 3	0.88	24	72
40	Urip Sumoharjo	0.78	24	120
41	Joyo Utomo 2	0.34	25	215
42	Mertojoyo 1	0.24	25	272
43	Bendungan Sigura - gura 1	0.73	26	27
44	Bendungan Sigura - gura 2	0.56	26	219
45	Bendungan Sutami 1	0.84	27	28
46	Veteran	1.24	27	31
47	Sumbersari	1.30	27	212
48	Bendungan Sutami 2	0.41	28	29
49	Raya Candi 1	1.24	28	223
50	Raya Tidar 1	0.24	29	190
51	Raya Dieng 1	0.34	30	35
52	Terusan Dieng	0.84	30	41
53	Raya Langsep	0.26	30	63
54	Galunggung 1	0.35	30	188
55	Bogor 2	0.29	31	199
56	Gede	0.35	32	33
57	Pahlawan (TRIP)	0.54	32	38
58	Surabaya 2	0.22	32	192
59	Jakarta 1	0.49	32	195
60	Simpang Wilis 1	0.50	33	34
61	Retawu	0.56	33	37
62	Bondowoso 1	0.18	33	191
63	Raya Dieng 2	0.10	34	187
64	Ijen 3	0.44	36	37
65	Kawi 5	0.10	36	184
66	Ijen 2	0.50	37	38
67	Wilis 1	0.10	37	185
68	Ijen 1	0.41	38	270
69	Guntur	0.64	38	271
70	Simpang Mega Mendung	0.78	40	41
71	Pattimura 5	0.26	42	24
72	Kertanegara	0.36	43	44
73	Trunojoyo 3	0.19	43	167

No	NamaJalan	PanjangJalan (km)	Node1	Node2
74	Tugu 3	0.06	44	47
75	Tugu 1	0.10	45	164
76	Tugu 5	0.10	46	45
77	Mojopahit	0.50	46	48
78	Tugu 4	0.08	47	46
79	Gajahmada	0.38	47	162
80	MGR. Sugito 1	0.16	48	49
81	Basuki Rahmat 1	0.16	48	53
82	MGR. Sugito 2	0.14	49	50
83	Aris Munandar 1	0.34	49	57
84	Merdeka Timur	0.20	50	51
85	Merdeka Selatan	0.20	51	52
86	Agus Salim	0.30	51	59
87	Sukarjo WR	0.20	51	60
88	Merdeka Barat	0.18	52	53
89	Wahid Hasyim	0.16	52	159
90	Merdeka Utara	0.16	53	50
91	Arif Rahman Hakim	0.33	53	54
92	Hasyim Asyari 1	0.20	54	55
93	Kawi 1	0.16	54	178
94	Kauman	0.28	55	52
95	Hasyim Asyari 2	0.20	55	56
96	Brigjen Katamso	0.72	56	64
97	Arif Margono 1	0.24	56	65
98	Ade Irma Suryani 1	0.28	56	159
99	Aris Munandar 2	0.34	57	58
100	KH. Zainul Arifin 1	0.33	57	59
101	Juanda 1	0.53	58	69
102	Ahmad Dahlan	0.28	59	161
103	Pasar Besar 1	0.31	60	61
104	Sutan Sahrir	0.28	60	66
105	Pasar Besar 2	0.30	61	62
106	Sersan Harun	0.30	61	67
107	KP. Usman	0.28	61	160
108	Laks. Martadinata 1	0.28	62	68
109	Zainul Zackse	0.54	62	70
110	Gatot Subroto 1	0.25	62	161
111	IR Rais	0.13	63	64

No	NamaJalan	PanjangJalan (km)	Node1	Node2
112	Mergan Lori 1	0.97	63	155
113	Yulius Usman 1	0.33	65	153
114	Kapten Tendean 2	0.25	66	67
115	Kyai Tamin 1	0.08	67	160
116	Laks. Martadinata 2	0.62	68	71
117	Juanda 2	0.40	69	70
118	Kebalen	1.10	70	71
119	Muharto 2	0.68	70	125
120	Kolonel Soegiono 1	1.27	71	73
121	Sartono	0.26	71	75
122	Gatot Subroto 2	0.36	72	58
123	Untung Suropati Selatan	0.34	72	163
124	Kolonel Soegiono 2	2.24	73	367
125	Satsuit Tubun 1	1.12	74	78
126	Satsuit Tubun 2	0.78	74	136
127	Irian Jaya	0.48	75	76
128	Peltu Sudiono	0.44	75	150
129	Terusan Halmahera 1	0.55	76	77
130	Halmahera 2	0.33	76	157
131	Susanto 1	0.30	77	149
132	Sudanco Supriadi 1	1.17	78	138
133	Sudanco Supriadi 5	0.66	79	80
134	Klayatan III	0.63	79	139
135	Mayjen M. Wiyono 2	0.40	80	121
136	Sudanco Supriadi 4	0.30	80	142
137	Pucang 1	0.87	80	144
138	Simpang Laksda Adi Sucipto 1	0.83	81	301
139	Simpang Sulfat	1.00	82	83
140	Warinoi	0.47	83	84
141	Membrano 1	0.39	84	357
142	Terusan Membrano 1	0.38	85	119
143	Kesatrian terusan	0.66	85	120
144	Sukarno Hatta 2	0.96	86	87
145	Pisang Kipas	1.13	86	224
146	Sukarno Hatta 3	0.32	87	88
147	Terusan Candi Mendut	0.58	87	91
148	Sudimoro 1	0.19	88	230
149	Sukarno Hatta 1	0.57	88	234

No	NamaJalan	PanjangJalan (km)	Node1	Node2
150	Taman Intan	0.36	89	92
151	Candi Mendut 2	0.79	91	237
152	Simpang Panji Suroso	0.80	92	94
153	Raden Panji Suroso 2	0.39	93	18
154	Raden Panji Suroso 1	0.84	93	94
155	Raden Panji Suroso 3	0.56	94	19
156	Plaosan	0.66	94	95
157	Jendral A. Yani 3	0.60	95	16
158	Ikan Tombro 1	0.18	97	347
159	Ikan Tombro Barat	1.06	96	98
160	Ikan Kakap 2	0.61	97	101
161	KH. Yusuf	0.78	98	99
162	Akordion 1	0.66	99	100
163	Ikan Gurami 1	0.79	100	229
164	Simpang Piranha Atas	0.38	101	102
165	Ikan Piranha 1	0.74	101	104
166	Ikan Lodan 1	0.64	102	105
167	Jendral S.Parman 4	0.52	103	107
168	Laksda Adi Sucipto 5	0.62	103	250
169	Simpang Piranha	0.20	104	105
170	Ikan Piranha 2	0.26	104	106
171	Ikan Lodan 2	0.30	105	106
172	Ikan Piranha Atas	0.38	106	107
173	Letjen S.Parman 1	0.14	107	95
174	Laksda Adi Sucipto 2	0.62	108	114
175	Terusan Batubara 1	0.78	108	302
176	L. Sunandar Priyosudarmo 2	0.40	109	19
177	L. Sunandar Priyosudarmo 3	0.22	109	248
178	Tenaga Selatan 2	0.25	110	247
179	Tenaga Baru 1	0.21	110	249
180	Tenaga Selatan 1	0.27	111	110
181	Karya Timur 1	0.91	111	251
182	Tenaga Barat	0.10	112	111
183	Tenaga Utara	0.47	112	249
184	Letjen S.Parman 2	0.17	113	15
185	Tenaga 1	0.36	113	112
186	Laksda Adi Sucipto 1	0.14	114	81
187	Besi	0.15	115	303

No	NamaJalan	PanjangJalan (km)	Node1	Node2
188	Batubara 1	0.22	116	115
189	L. Sunandar Priyosudarmo 1	0.09	116	304
190	L. Sunandar Priyosudarmo 6	0.31	117	20
191	Sulfat 3	0.41	118	268
192	Indraprasta	0.74	119	80
193	Mayjen M. Wiyono 1	0.33	120	80
194	Kesatrian	0.66	120	163
195	Ranu Grati 1	0.26	121	122
196	Puntodewo	1.75	121	125
197	Raya Sawojajar 1	1.22	122	123
198	Ranu Grati 2	0.03	122	310
199	Raya Sawojajar 2	1.06	122	320
200	Ki Ageng Gribig 1	0.35	123	124
201	Ki Ageng Gribig 2	0.61	123	308
202	Muharto 1	0.66	124	125
203	Mayjen Sungkono 1	1.58	124	128
204	Danau Toba 1	0.34	126	127
205	Danau Sentani 1	0.14	127	342
206	Danau Sentani 2	0.13	127	342
207	Ki Ageng Gribig 5	1.09	127	346
208	Mayjen Sungkono 2	0.82	128	131
209	Kyai Parseh 4	0.64	129	128
210	Kyai Parseh 3	0.86	130	129
211	Kyai Parseh 2	0.87	130	131
212	Kyai Parseh 1	1.84	130	136
213	Mayjen Sungkono 3	1.48	131	132
214	Mayjen Sungkono 4	0.77	132	133
215	Raya Tlogowaru 1	1.10	133	134
216	Mayjen Sungkono 5	0.74	133	135
217	Raya Tlogowaru 2	0.59	134	137
218	Arjowinangun 2	2.74	135	136
219	Arjowinangun 1	0.96	137	135
220	Sudanco Supriadi 3	0.68	138	79
221	Klayatan Kemantren	1.84	139	140
222	Mulyorejo 3	0.60	140	141
223	Mulyorejo 2	1.28	140	172
224	Flamboyan	2.24	141	139
225	Sudanco Supriadi 2	0.36	142	143

No	NamaJalan	PanjangJalan (km)	Node1	Node2
226	Galang 2	0.45	143	148
227	Arif Margono 2	0.44	143	156
228	Pucang 2	0.30	144	145
229	Janti	0.44	144	146
230	Pucang 3	0.38	145	73
231	Sonokeling 1	0.34	145	146
232	Sonokeling 2	0.76	146	147
233	Niaga	0.54	146	149
234	Terusan Halmahera 2	0.29	147	77
235	Galang 1	0.24	147	148
236	Andalaste	0.85	148	151
237	Susanto 2	0.15	149	150
238	Tanimbar 2	0.38	151	76
239	Tanimbar 1	0.30	151	152
240	Sulawesi	0.20	152	153
241	Nusa Kambangan 2	0.48	152	157
242	Yulius Usman 2	0.34	153	158
243	Arif Margono 4	0.20	154	65
244	Nusa Kambangan 1	0.34	154	152
245	Mergan Lori 2	0.54	155	143
246	Rajawali	0.47	155	156
247	Arif Margono 3	1.30	156	154
248	Halmahera 1	0.23	157	66
249	Kapten Tendean 1	0.22	158	66
250	S. Al. Qodri	0.22	158	159
251	Ade Irma Suryani 2	0.24	159	60
252	Kyai Tamin 2	0.23	160	68
253	Gatot Subroto 3	0.24	161	58
254	Trunojoyo 2	0.22	162	43
255	Trunojoyo 1	0.21	162	72
256	Tugu 2	0.10	164	44
257	Untung Suropati	0.19	164	166
258	Kahuripan 2	0.23	165	45
259	Belakang RSU	0.50	165	168
260	Pajajaran	0.28	166	167
261	Trunojoyo 4	0.26	167	42
262	Pattimura 3	0.15	168	172
263	Pattimura 2	0.15	169	168

No	NamaJalan	PanjangJalan (km)	Node1	Node2
264	Diponegoro	0.43	169	170
265	Dr. Cipto 3	0.26	170	23
266	Cokroaminoto 2	0.29	171	42
267	Pattimura 4	0.22	172	42
268	Thamrin	0.33	172	171
269	Mulyorejo 1	0.32	172	173
270	Tebo Utara 1	0.30	173	174
271	Tebo Utara 2	0.28	174	175
272	Tebo Utara 3	0.59	175	176
273	Bandulan (Jupri)	0.71	176	63
274	Bandulan Barat 1	1.35	176	177
275	Bandulan Barat 2	1.44	177	41
276	Kawi 2	0.12	178	179
277	Arjuno	0.40	178	181
278	Kawi 3	0.15	179	180
279	Bromo 1	0.39	179	182
280	Kawi 4	0.48	180	36
281	Tangkuban Perahu	0.40	180	183
282	Semeru 2	0.07	181	182
283	Semeru 3	0.09	182	183
284	Bromo 2	0.40	182	201
285	Semeru 4	0.36	183	37
286	Panderman	0.32	184	185
287	Kawi 6	0.25	184	186
288	Wilis 2	0.34	185	34
289	Pulosari	0.23	186	34
290	Kawi 7	0.21	186	35
291	Raya Dieng 3	0.10	187	35
292	Gading	0.36	187	188
293	Galunggung 2	0.60	188	29
294	Bukit Barisan	0.30	188	189
295	Simpang Tambora	0.50	189	190
296	Raya Tidar 2	0.66	190	40
297	Gresik	0.20	191	192
298	Bondowoso 2	0.18	191	194
299	Surabaya 3	0.18	192	193
300	Surabaya 1	0.44	193	28
301	Bondowoso 3	0.20	194	29

No	NamaJalan	PanjangJalan (km)	Node1	Node2
302	Jombang	0.17	194	193
303	Bogor 1	0.40	195	31
304	Jakarta 2	0.22	195	196
305	Jakarta 3	0.21	196	197
306	Garut	0.15	196	198
307	Ijen 4	0.10	197	6
308	Raung	0.20	197	200
309	Bandung 1	0.27	198	31
310	Mayjen Panjaitan 1	0.60	199	6
311	Brigjen Slamet Riadi 4	0.31	200	6
312	Panggung 1	0.00	200	269
313	Brigjen Slamet Riadi 2	0.44	201	202
314	Brigjen Slamet Riadi 3	0.45	202	200
315	Guntur 1	0.00	202	271
316	Tawangmangu 2	0.15	203	283
317	Kalpataru 2	0.95	204	205
318	Kalpataru 1	0.44	204	206
319	Cengger Ayam 1	0.50	204	208
320	Coklat (Cengkeh)	0.70	205	86
321	Kedawung	0.32	206	13
322	Melati	0.30	207	204
323	Candi Mendut 1	0.26	208	209
324	Cengger Ayam 2	0.43	208	210
325	Bukitsari	0.39	209	211
326	Telaga Wangi 1	0.34	210	211
327	Telaga Wangi 2	0.13	211	91
328	Gajayana	0.50	212	4
329	Simpang Gajayana	0.34	212	213
330	Merjosari Selatan	0.36	213	214
331	Joyo Utomo 1 (Tambak sari)	0.26	213	221
332	Mertojoyo Selatan	0.56	214	25
333	Joyo Sari 1	0.34	215	216
334	Joyo Suryo 1	0.15	215	273
335	Joyo Sari 2	0.27	216	217
336	Perum Joyo Grand	0.14	217	220
337	Mertosari 1	0.20	218	222
338	Sunan Kalijogo	0.81	219	214
339	Joyo Utomo 3	0.12	221	25

No	NamaJalan	PanjangJalan (km)	Node1	Node2
340	Joyo Raharjo	0.58	221	222
341	Mertosari 2	0.13	222	3
342	Terusan Sigura - gura	0.44	223	26
343	Vinolia 1	0.44	224	225
344	Simpang Candi Panggung 1	0.57	225	226
345	Vinolia 2	0.31	225	228
346	Simpang Candi Panggung 2	0.09	226	227
347	Candi Panggung	0.80	227	87
348	Akordion 2	1.13	228	100
349	Akordion Timur	0.71	228	227
350	Ikan Gurami 2	0.43	229	232
351	Sudimoro 2	0.71	230	231
352	Sudimoro 3	1.06	230	232
353	Candi Sari Utara	0.17	231	233
354	Borobudur 1	0.47	231	240
355	Ikan Kakap 1	0.15	348	97
356	Candi Badut 1	0.28	233	235
357	Sukarno Hatta Indah	0.32	234	91
358	Sukarno Hatta 4	0.26	234	231
359	Candi Badut 2	0.23	235	236
360	Candi Sewu 1	0.20	235	238
361	Candi Bima	0.20	236	237
362	Candi Agung 1	0.24	236	243
363	Candi Tegowangi	0.71	237	14
364	Candi Trowulan 1	0.19	238	239
365	Candi Sewu 2	0.10	238	241
366	Candi Trowulan 2	0.04	239	242
367	Candi Trowulan 4	0.24	239	244
368	Borobudur 2	0.25	240	245
369	Candi Panataran 1	0.18	241	242
370	Candi Trowulan 3	0.07	242	240
371	Candi Panataran 2	0.23	242	246
372	Candi Waringin 3	0.25	243	244
373	Candi Waringin 1	0.03	244	246
374	Borobudur 3	0.35	245	15
375	Candi Waringin 2	0.07	246	245
376	Tenaga Selatan 3	0.13	247	109
377	Simpang Tenaga 2	0.26	247	248

No	NamaJalan	PanjangJalan (km)	Node1	Node2
378	L. Sunandar Priyosudarmo 5	0.43	248	116
379	Tenaga Baru 2	0.13	249	250
380	Ciliwung 2	0.59	251	117
381	Karya Timur 2	0.41	251	253
382	Letjen Sutoyo 4	0.52	252	14
383	Ciliwung 1	0.19	252	251
384	Industri Timur	0.26	253	254
385	Ciwulan	0.53	253	263
386	Indragiri 1	0.41	254	255
387	Asahan 1	0.13	254	257
388	Indragiri 2	0.11	255	256
389	Barito 2	0.39	256	257
390	Barito 1	0.07	256	258
391	Serayu	0.26	256	259
392	Asahan 2	0.20	257	260
393	Tumenggung Suryo 2	0.55	258	262
394	Musi	0.38	260	259
395	Mahakam 1	0.40	260	264
396	WR Suprtaman 2	0.27	261	21
397	Tumenggung Suryo 5	0.16	262	21
398	Kahayan	0.36	262	264
399	Tumenggung Suryo 4	0.24	263	265
400	Mahakam 2	0.14	264	261
401	Tumenggung Suryo 1	0.14	265	258
402	Sanan	0.76	265	255
403	Sulfat 2	0.23	266	118
404	Sulfat 4	0.25	268	82
405	Taman Sulfat	0.84	268	267
406	Panggung 2	0.33	269	270
407	Anjasmoro	0.39	269	271
408	Ijen 5	0.18	270	197
409	Mertojoyo 2	0.14	272	218
410	Merjosari	0.28	273	272
411	Joyo Pranoto	0.21	273	274
412	Joyo Suryo 2	0.16	273	275
413	Tlogo Sari	0.23	274	275
414	Tlogo Indah 1	0.08	275	276
415	Mertojoyo Barat	0.33	276	218

No	NamaJalan	PanjangJalan (km)	Node1	Node2
416	Tlogo Indah 2	0.10	276	277
417	Tlogo Agung	0.30	276	282
418	Tlogo Indah 3	0.04	277	278
419	Tlogomas Indah 1	0.15	278	279
420	Tlogomas Indah 2	0.27	279	280
421	Tlogo Joyo	0.11	279	281
422	Tlogomas 3	0.27	280	3
423	Joyo Mulyo	0.30	281	222
424	Tlogo Suryo	0.24	282	277
425	Tawangmangu 1	0.24	283	90
426	Parangtritis	0.10	283	284
427	Mawar	0.25	284	285
428	Gilimanuk 1	0.27	284	286
429	Sarangan 2	0.09	285	90
430	Bungur	0.58	285	207
431	Gilimanuk 2	0.11	286	287
432	Gilimanuk 3	0.50	286	289
433	Cempaka Putih	0.21	287	288
434	Anggrek Vanda	0.19	288	289
435	Simpang Flamboyan 1	0.06	288	290
436	Bougenvile	0.25	289	293
437	Simpang Flamboyan 2	0.04	290	291
438	Anggrek Garuda	0.37	290	293
439	Rasida	0.14	291	245
440	Simpang Flamboyan 3	0.04	291	292
441	Matahari	0.10	293	294
442	Srigading 1	0.34	294	296
443	Srigading 2	0.43	294	292
444	Kembang Dewandaru 1	0.35	296	297
445	Kumis Kucing	0.15	296	298
446	Kembang Dewandaru 2	0.24	297	205
447	Bunga Merak	0.52	297	299
448	Andong	0.36	298	299
449	Semanggi Timur	0.22	299	300
450	Sukarno Hatta 6	0.29	300	86
451	Simpang Laksda Adi Sucipto 2	0.90	301	82
452	Terusan Batubara 2	0.80	301	302
453	Batubara 2	0.61	302	115

No	NamaJalan	PanjangJalan (km)	Node1	Node2
454	Aluminium 2	0.14	303	305
455	L. Sunandar Priyosudarmo 4	0.11	304	117
456	Aluminium 1	0.27	304	303
457	Emas	0.50	305	118
458	Simpang Dirgantara 2	0.13	306	307
459	Simpang Dirgantara 3	0.10	307	308
460	Ki Ageng Gribig 3	0.93	308	309
461	Ki Ageng Gribik 4	0.33	309	127
462	Raya Dirgantara	0.58	310	306
463	Ranu Grati 3	0.07	310	311
464	Simpang Dirgantara 1	0.61	311	307
465	Danau Toba 2	0.15	311	312
466	Danau Toba 4	0.16	312	313
467	Selat Sunda	0.18	312	341
468	Danau Ranau 1	0.13	313	314
469	Danau Toba 3	0.19	313	316
470	Danau Ranau 2	0.19	314	315
471	Danau Maninjau 1	0.66	314	318
472	Sawojajar Anakan	0.14	315	316
473	Danau Ranau 3	0.16	315	343
474	Danau Toba 5	0.10	316	126
475	Danau Maninjau 3	0.19	317	321
476	Danau Maninjau 2	0.07	318	317
477	Danau Bedugul 1	0.45	318	319
478	Danau Kerinci 3	0.18	319	320
479	Danau Bedugul 2	0.37	319	332
480	Maninjau Barat	0.25	320	317
481	Danau Kerinci 4	0.13	320	333
482	Raya Sawojajar 3	0.54	320	340
483	Danau Montano	0.46	321	320
484	Danau Maninjau 4	0.16	321	322
485	Danau Maninjau 5	0.20	322	323
486	Danau Tondano 2	0.34	322	325
487	Danau Limboto 3	0.21	323	324
488	Danau Kerinci 6	0.16	325	324
489	Danau Paniai 1	0.38	325	330
490	Danau Tondano 1	0.24	326	322
491	Danau Limboto 2	0.20	326	323

No	NamaJalan	PanjangJalan (km)	Node1	Node2
492	Danau Jongge 1	0.28	327	338
493	Danau Sentani Tengah 3	0.15	328	327
494	Danau Paniai 3	0.29	328	334
495	Danau Bratan 2	0.45	329	336
496	Danau Sentani Tengah 1	0.26	329	345
497	Danau Paniai 2	0.12	330	328
498	Sentani Utara	0.12	331	330
499	Danau Kalimutu 2	0.13	331	345
500	Danau Bratan 1	0.12	332	329
501	Danau Sentani 4	0.26	332	331
502	Danau Kerinci 5	0.12	333	325
503	Danau Kalimutu 1	0.37	333	331
504	Danau Bratan Timur 3	0.10	334	338
505	Danau Bratan Timur 2	0.21	335	334
506	Terusan Danau Bratan	0.18	335	346
507	Danau Bratan Timur 1	0.12	336	335
508	Madyopuro 1	0.23	337	339
509	Danau Jongge 2	0.22	338	337
510	Madyopuro 2	1.09	339	360
511	Danau Limboto 1	0.30	340	326
512	Lesan Puro III	0.43	341	309
513	Danau Sentani 3	0.31	342	332
514	Danau Singkarak 1	0.26	343	344
515	Danau Kerinci 1	0.14	344	126
516	Danau Kerinci 2	0.31	344	319
517	Danau Singkarak 2	0.38	344	342
518	Danau Sentani Tengah 2	0.11	345	328
519	Ki Ageng Gribig 6	0.18	346	337
520	Simpang Candi Trowulan	0.24	239	236
521	Terusan Piranha 1	0.27	232	348
522	Ikan Tombro 2	0.49	347	96
523	Terusan Piranha 2	0.36	348	347
524	Ahmad Yani Utara	0.76	17	349
525	Raya Tidar 3	0.93	40	353
526	Joyo Suko Timur	0.12	214	352
527	Raya Candi 2	1.90	352	354
528	Candi VI 1	0.66	352	351
529	Candi VI 2	0.70	350	351

No	NamaJalan	PanjangJalan (km)	Node1	Node2
530	Candi V 1	0.36	350	356
531	Candi V 2	0.32	356	353
532	Karang Widoro	0.37	350	355
533	Terusan Membrano 2	0.07	85	359
534	Membrano 2	0.13	357	85
535	Sisingamangaraja	0.11	357	358
536	Raden Patah	0.15	358	359
537	MT Haryono 3	0.02	362	4
538	MT Haryono 4	0.02	4	365
539	MT Haryono 5	0.20	365	5
540	MT Haryono IX	0.31	365	364
541	MT Haryono XI 1	0.16	362	364
542	MT Haryono XI 2	0.01	364	363
543	MT Haryono XIII	0.25	361	363
544	MT Haryono XI 3	0.31	363	224
545	Kanjuruan 1	0.43	366	274
546	Kanjuruan 2	0.06	217	366
547	Kolonel Soegiono 3	0.11	367	74
548	KH. Zainul Arifin 2	0.16	59	61