RANCANG BANGUN APLIKASI PENCARIAN RUTE TERPENDEK UNTUK MENEMUKAN SPBU TERDEKAT DI KOTA MALANG DENGAN MENGGUNAKAN ALGORITMA GENETIK BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

VIKA NOVITASARI

NIM. 105060800111012

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI PENCARIAN RUTE TERPENDEK UNTUK MENEMUKAN SPBU TERDEKAT DI KOTA MALANG DENGAN MENGGUNAKAN ALGORITMA GENETIK BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

VIKA NOVITASARI

NIM. 105060800111012

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng Herman Tolle, ST., MT.	**Aryo Pinandito, ST., M.MT.**
NIP. 197408232000121 001	**NIK. 830519 16 1 1 0374**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PENCARIAN RUTE TERPENDEK UNTUK MENEMUKAN SPBU TERDEKAT DI KOTA MALANG DENGAN MENGGUNAKAN ALGORITMA GENETIK BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

VIKA NOVITASARI

NIM. 105060800111012

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 2 Juli 2014

Penguji I	Penguji II
Agi Putra Kharisma, ST., MT.	**Wibisono Sukmo Wardhono, ST., MT.**
-	**NIK. 820404 06 1 1 0091**

Penguji III

Aswin Suharsono, ST., MT.
NIK. 840919 06 1 1 0251

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, 9 Juni 2014

Mahasiswa,

Vika Novitasari
NIM. 105060800111012

KATA PENGANTAR

Puji dan syukur saya ucapkan kepada Allah SWT karena atas rahmat dan karunia-Nya, saya dapat menyelesaikan skripsi dengan judul "Rancang Bangun Aplikasi Pencarian Rute Terpendek untuk Menemukan SBPU Terdekat di Kota Malang dengan Menggunakan Algoritma Genetik Berbasis Android" sebagai salah satu persyaratan untuk menyelesaikan studi di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

Saya ingin menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi, diantaranya:

1. Bapak, Ibu dan seluruh keluarga atas segala nasehat, kasih sayang, perhatian dan kesabarannya dalam mendidik penulis serta memberikan doa dan semangat secara terus-menerus demi terselesaikannya skripsi ini.
2. Bapak Ir. Sutrisno, M.T., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Himawat Aryadita, S.T., M.Sc., dan Bapak Eddy Santoso, S.Kom. selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T. dan Bapak Issa Arwani, S.Kom., M.Sc. selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
4. Bapak Dr. Eng Herman Tolle, ST., MT. selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk menyelesaikan skripsi ini.
5. Bapak Aryo Pinandito, S.T., M.MT. selaku dosen pembimbing II yang juga memberikan ilmu dan saran untuk menyelesaikan skripsi ini.
6. Bapak Ir. Sutrisno, M.T. selaku dosen pembimbing akademik yang telah memberikan pengarahan selama penulis menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
7. Seluruh dosen dan karyawan Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

8. Sahabat-sahabat TIF 2010 yang telah memberikan doa, bantuan serta semangat kepada penulis selama menempuh studi di Teknik Informatika Universitas Brawijaya.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Semoga Tugas Akhir ini dapat memberikan manfaat bagi pembaca sekaligus dapat menjadi bahan acuan untuk penelitian selanjutnya.

Malang, 9 Juni 2014

Penulis

ABSTRAK

Vika Novitasari. 2014. Rancang Bangun Aplikasi Pencarian Rute Terpendek untuk Menemukan SBPU Terdekat di Kota Malang dengan Menggunakan Algoritma Genetik Berbasis Android. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing : Dr. Eng Herman Tolle, ST., MT. dan Aryo Pinandito, ST., M.MT.

Malang merupakan kota yang sangat berpotensi untuk terus mengembangkan diri menjadi daerah wisata yang sangat menjajikan. Banyak objek wisata yang berpotensial untuk menarik wisatawan. Seiring dengan perkembangan wisata Kota Malang, semakin banyak pula pendatang dan wisatawan di Kota Malang. Namun, tidak semua pendatang mengenal kondisi Kota Malang dengan baik, salah satunya tidak dapat mengetahui rute terpendek untuk menuju ke SPBU terdekat hanya dengan memanfaatkan fasilitas peta dan GPS apabila membutuhkannya sewaktu-waktu dan dalam keadaan mendesak.

Aplikasi dibuat berbasis *mobile* dengan *platform* Android dan dikembangkan dengan bahasa pemrograman Java. Hasil yang diperoleh merupakan sebuah aplikasi yang dapat menampilkan rute terpendek menuju SPBU terdekat pada peta (*Google Maps*).

Dalam penelitian ini digunakan Algoritma Genetik dalam pemilihan rute menuju ke SPBU terdekat. Algoritma Genetik dapat digunakan untuk memecahkan masalah pencarian rute terpendek termasuk dalam hal pencarian lokasi SPBU terdekat. Berdasarkan hasil pengujian yang dilakukan Algoritma Genetik memperoleh hasil optimal dan mencapai nilai yang konvergen pada jumlah generasi 35 dan 55 serta semakin besar nilai probabilitas *crossover* dan probabilitas mutasi maka nilai *fitness* yang dihasilkan juga semakin tinggi.

Kata Kunci: Algoritma Genetik, Android, *Google Maps*, Perangkat Bergerak, SPBU, Rute Terdekat

ABSTRACT

Vika Novitasari. 2014. ***Design and Implementation of Mobile Application for Finding the Shortest Route to The Gas Station in Malang by Using Genetic Algorithm- Android Based.*** *Information Technology and Computer Science Program, Brawijaya University,* Malang. *Advisors* : Dr. Eng Herman Tolle, ST., MT. dan Aryo Pinandito, ST., M.MT.

Malang is a city that has the potential to continue developed itself into a very promising tourist areas. Many attractions are potential to attract tourists. Along with the development of Malang, more the immigrants and tourists came to Malang. However, not all entrants know the conditions of Malang, one of them is entrants can not know the shortest route to get the nearest gas station only using maps and GPS when needed at any time and in urgent circumstances.

Mobile-based application created and developed with the Android platform with the Java programming language. The result is an application that can display the shortest route to the nearest gas station on the map (Google Maps).

In this research, the Genetic Algorithm use to select route to the nearest gas station. Genetic Algorithms can be used to solve problem of the shortest route, include searching for location of the nearest gas station. Based on the results of the tests, Genetic Algorithm obtain optimal results and achieve converges value on the number of generations 35 and 55, and the greater of crossover probability and mutation probability give the fitness value also higher.

Keywords: *Genetic Algorithm*, Android, *Google Maps, Mobile, Gas Station, Sortest Path*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang, rumusan masalah, tujuan, manfaat dan sistematika penulisan skripsi.

1.1 Latar Belakang

Malang merupakan kota yang sangat berpotensi untuk terus mengembangkan diri menjadi daerah wisata yang sangat menjajikan. Banyak objek wisata yang berpotensial untuk menarik wisatawan. Seiring dengan perkembangan wisata Kota Malang, semakin banyak pula pendatang di Kota Malang.

Smartphone merupakan suatu perangkat dengan kemampuan tinggi dan canggih. *Smartphone* dirancang untuk membantu penggunanya dengan berbagai macam fasilitas yang tersedia di dalamnya. Fasilitas yang dimiliki *smartphone* antara lain GPS dan peta (*Google Maps*). GPS atau *Global Positioning System* merupakan sebuah fasilitas yang dapat mendeteksi keberadaan perangkat *smartphone* dengan bantuan satelit [ROM-12]. Sedangkan *Google Maps* merupakan layanan pemetaan berbasis *web service* yang disediakan oleh Google dan bersifat gratis [ELI-12].

Namun, tidak semua pendatang mengenal kondisi Kota Malang dengan baik, salah satunya tidak dapat mengetahui rute terpendek untuk menuju ke SPBU terdekat hanya dengan memanfaatkan fasilitas peta dan GPS apabila membutuhkannya sewaktu-waktu dan dalam keadaan mendesak.

Tujuan dari penelitian ini adalah untuk membantu pengguna *smartphone* yang ingin mencari lokasi SPBU terdekat dari tempatnya berada sedangkan pengendara tersebut kurang mengetahui daerah. Aplikasi ini berbasis *mobile* dengan sistem operasi Android. Pemilihan rute terpendek untuk menuju SPBU terdekat dilakukan dengan menerapkan Algoritma Genetik serta bantuan *Google Maps* dan GPS untuk menampilkan hasil dari perhitungan dengan menggunakan algoritma tersebut.

Genetic Algorithm (GA) atau Algoritma Genetik adalah algoritma yang didasarkan pada proses genetik pada makhluk hidup. Algoritma Genetik merupakan cabang dari Algoritma Evolusi yang dapat digunakan untuk memecahkan suatu permasalahan yang berkaitan dengan masalah optimasi [YUD-12]. Oleh karena itu, Algoritma Genetik dapat digunakan untuk memecahkan masalah pencarian rute terpendek termasuk dalam hal pencarian lokasi SPBU terdekat. Namun, harus ditentukan nilai kriteria optimasi yang diinginkan dalam bentuk probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm) [MAR-13]. Selain itu, Algoritma Genetik juga dipengaruhi oleh banyaknya jumlah iterasi atau generasi [HID-07].

Keunggulan Algoritma Genetik antara lain tidak memerlukan persyaratan matematika yang terlalu banyak dalam penyelesaian proses optimasi, Algoritma Genetika memiliki operasi evolusi yang sangat efektif untuk mengobservasi posisi global secara acak serta lebih mudah dan fleksibel untuk diimplementasikan pada masalah-masalah tertentu [YUS-12].

1.2 Rumusan Masalah

Rumusan masalah berdasarkan latar belakang yang telah diuraikan sebelumnya antara lain:

1. Bagaimana merancang dan mengimplementasikan suatu aplikasi berbasis Android yang dapat membantu pengguna dalam menemukan lokasi SPBU terdekat?
2. Bagaimana cara menampilkan rute menuju SPBU pada *Google Maps*?
3. Bagaimana pengaruh probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm) dalam Algoritma Genetik dapat menentukan rute terpendek menuju SPBU terdekat?
4. Bagaimana pengaruh banyak iterasi atau generasi dalam Algoritma Genetik dapat menentukan rute terpendek menuju SPBU terdekat?

1.3 Batasan Masalah

Batasan masalah berdasarkan latar belakang dan rumusan masalah yang telah diuraikan sebelumnya yaitu sebagai berikut:

1. Penelitian menggunakan Algoritma Genetik untuk menyelesaikan masalah penentuan rute terpendek menuju SPBU terdekat.
2. Data penelitian yang digunakan berasal dari peta Kota Malang dan *Google Maps*.
3. Hanya data jalan-jalan besar di Kota Malang yang dimasukkan ke dalam sistem.
4. Data jalan mengikuti alur jalan pada pukul 07.00 WIB hingga 18.00 WIB.
5. Aplikasi ini dikembangkan untuk platform Android.
6. Aplikasi ini membutuhkan koneksi internet untuk membuka *Google Maps* dan menampilkan rute hasil.
7. Aktor yang berinteraksi pada aplikasi hanya pengguna aplikasi.
8. Pengujian Algoritma Genetik dilakukan dengan menggunakan rute-rute (populasi awal) yang telah ditentukan.

1.4 Tujuan

Tujuan dari dilakukannya penelitian ini yaitu:

1. Menghasilkan aplikasi yang bisa membantu pengguna menemukan lokasi SPBU terdekat di Kota Malang.
2. Menampilkan rute hasil dari perhitungan Algoritma Genetik pada *Google Maps*.
3. Melakukan pengujian terhadap pengaruh probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm) dalam Algoritma Genetik.
4. Melakukan pengujian terhadap pengaruh banyak iterasi atau generasi dalam Algoritma Genetik.

1.5 Manfaat

Penelitian ini diharapkan mempunyai manfaat yang berguna bagi pembaca, penulis maupun bagi pengguna aplikasi nantinya. Manfaat yang diharapkan dari penilitian ini yaitu sebagai berikut:

a) Bagi Pembaca
    1. Aplikasi ini diharapkan dapat menambah referensi terhadap penelitian baru dengan bidang studi terkait.
    2. Mendapatkan wawasan akan pengimplementasian Algoritma Genetik.

b) Bagi Penulis
    1. Diharapkan dapat menjadi pembelajaran dan menambah pengalaman di bidang studi keilmuan yang terkait.
    2. Sebagai media pengimplementasian ilmu pengetahuan teknologi khususnya pemrograman Android dan penerapan Algoritma Genetik.

c) Bagi pengguna aplikasi
    1. Pengguna yang kurang mengetahui lokasi SPBU di Kota Malang mampu menemukan lokasi SPBU terdekat.
    2. Pengguna dapat melakukan pencarian SPBU dengan menggunakan aplikasi ini sewaktu-waktu karena aplikasi ini merupakan aplikasi Android yang perangkatnya dapat dengan mudah dibawa.

1.6 Sistematika Penulisan

Sistematika isi dan penulisan skripsi yaitu sebagai berikut:

Bab I : Pendahuluan

Berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, dan manfaat dari penelitian.

Bab II : Kajian Pustaka dan Dasar Teori

Berisi tentang penjelasan mengenai Android beserta perkembangannya, arsitektur yang ada pada Android serta fitur yang digunakan, *Google Maps* dan SQLite. Penggunaan GPS dan LBS, serta pembahasan tentang Algoritma Genetik.

Bab III : Metodologi Penelitian

Bab ini berisi tentang gambaran umum metode penelitian yang digunakan dalam pembuatan aplikasi.

Bab IV : Perancangan

Bab ini membahas tentang perancangan terhadap sistem yang akan dibangun sebagai dasar untuk tahap selanjutnya yaitu tahap implementasi.

Bab V : Implementasi dan Pengujian

Bab ini membahas tentang implementasi berdasarkan metodologi dan perancangan yang telah diuraikan sebelumnya serta diuraikan tentang pengujian terhadap hasil yang diperoleh.

Bab VI: Penutup

Bab ini berisi tentang kesimpulan dan saran terhadap penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1 Algoritma Genetik

Algoritma Genetik atau biasa disebut GA (*Genetic Algorithm*) merupakan cabang dari Algoritma Evolusi yang biasa digunakan untuk memecahkan permasalahan optimasi yaitu permasalahan-permasalahan yang tak linear. GA pertama kali diperkenalkan oleh John Holland pada tahun 1975 sebagai *whitepaper* di Universitas Michigan. Algoritma ini didasarkan pada proses genetik pada makhluk hidup, John mengatakan bahwa setiap masalah yang berbentuk adaptasi baik adaptasi secara alami ataupun adaptasi buatan dapat diformulasikan dalam terminology genetika [YUD-12].

Konsep dasar dari GA dirancang untuk menirukan proses di dalam sistem alami yang penting bagi evolusi makhluk hidup untuk dapat terus bertahan hidup, teori ini secara rinci dicetuskan oleh Charles Darwin yaitu "*Survival of the Fittest*" [AFA-11].

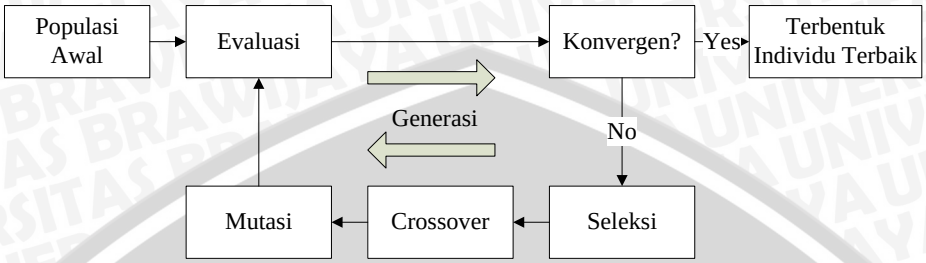
Variasi dari kromosom antar individu suatu organisme menjadikan keberagaman pada evolusi biologis. GA dapat memberikan solusi alternatif dari suatu masalah yang akan diselesaikan dengan memanfaatkan metode seleksi, *crossover* dan mutasi. GA banyak diaplikasikan dalam penyelesaian masalah dan pemodelan dalam bidang teknologi seperti optimasi, pemrograman otomatis dan *mechine learning*. Pada implementasi program salah satunya GA dapat digunakan untuk mencari jalan atau rute terpendek. Tahap awal pencarian pada GA dimulai dari himpunan penyelesaian acak yang disebut populasi.

Pada umumnya penerapan GA terdiri dari 3 bagian [AFA-11], yaitu:

1. Memilih populasi awal (inisial populasi).
2. Evaluasi nilai *fitness* dari setiap individu di dalam populasi.
3. Ulangi sampai proses berhenti (nilai *fitness* terbaik terpenuhi).
    a. Pilih individu terbaik berdasar ranking untuk reproduksi.
    b. Bentuk generasi baru melalui pindah silang (*crossover*) dan mutasi untuk menghasilkan keturunan baru (*child*).
    c. Evaluasi nilai *fitness* keturunan yang dihasilkan.

d. Gantikan individu dengan keturunan yang dihasilkan.

Ilustrasi proses GA pada Gambar 2.1.



Gambar 2.1 Siklus Algoritma Genetika
(Sumber : AFA-11)

Populasi terdiri dari kromosom-kromosom. Setiap kromosom merupakan gambaran solusi atas pemecahan masalah. Sebelum algoritma genetika dijalankan didefinisikan suatu fungsi *fitness* yang menyatakan tingkat keberhasilan sebuah populasi. Fungsi *fitness* dideskripsikan sebagai tingkat kebugaran dari tiap kromosom untuk menentukan yang akan direproduksi dan bertahan ke dalam generasi berikutnya [AFA-11]. Dengan melakukan perhitungan berdasarkan fungsi fitness, akan dapat ditentukan populasi yang akan dipertahankan untuk menghasilkan generasi selanjutnya. Proses ini biasa disebut sebagai proses seleksi [BAS-12].

Salah satu metode seleksi yaitu *Roulette Wheel*, *Roulette Wheel* digunakan untuk menentukan individu orang tua yang akan dikenai operasi genetik. Sesuai dengan namanya, metode ini menirukan permainan *roulette-wheel* dengan masing-masing kromosom menempati potongan lingkaran pada roda *roulette* secara proporsional sesuai dengan nilai *fitness*-nya [MIT-99]. Langkah-langkah seleksi *Roulette Wheel* adalah sebagai berikut [MAR-13]:

1. Hitung *fitness* (*Fk*) setiap individu pada suatu populasi (k=1,2,3,…,n)
2. Hitung jumlah total *fitness* dari semua individu pada populasi (k=1,2,3,…,n)

$$totFitness = \sum Fk \qquad (2.1)$$

3. Hitung peluang dari setiap individu yang ada (k=1,2,3,…n)

$$pk = \frac{Fk}{totFitness} \quad (2.2)$$

4. Hitung peluang kumulatif dari setiap individu (k=2,3,4,…n)

   $q_1 = p_1$

   $q_k = q_{k-1} + p_k$ (2.3)

5. Dibangkitkan bilangan *random (r)* agar diketahui individu mana yang terpilih dalam proses seleksi. Nilai bilangan *random* antara 0 sampai 1.
6. Jika $q_k \leq r$ dan $q_{k+1} > r$, maka pilih individu ke (k+1) sebagai kandidat induk.

   Keterangan:

   k = individu ke-k

   F = nilai *fitness*

   totFitness = total *fitness*

   pk = probabilitas individu ke-k

   qk = probabilitas kumulatif individu ke-k

Proses lain yang yaitu *crossover*. Pada proses ini, dilakukan persilangan atau perkawinan antar kromosom yang berada dalam satu generasi. Dengan demikian, kromosom yang terdapat pada populasi selanjutnya mewarisi sifat kedua induknya. Kromosom ini diharapkan bersifat lebih baik dibanding dengan generasi sebelumnya. Tidak semua kromosom pada suatu populasi akan mengalami proses rekombinasi. Kemungkinan suatu kromosom mengalami proses rekombinasi didasarkan pada probabilitas *crossover* (PC) yang telah ditentukan terlebih dahulu. Probabilitas *crossover* menyatakan peluang suatu kromosom akan mengalami crossover [MAR-13].

Terdapat bermacam-macam jenis metode *crossover* dalam algoritma genetika. Metode *crossover* yg digunakan pada skripsi ini adalah modifikasi dari metode *crossover* yang digunakan oleh Karas dan Atila (2011). Langkah-langkah dari metode *crossover* tersebut yaitu sebagai berikut [KAR-11]:

1. Membangkitkan bilangan *random* 0-1

2. Jika bilangan *random* < probabilitas *crossover*, lanjutkan langkah pada nomor 4.
3. Jika bilangan *random* > probabilitas *crossover*, maka tidak terjadi *crossover* pada induk tersebut.
4. Lakukan pencarian gen yang sama dimulai dari index 2 pada kromosom 1 dan 2 pilih sebagai titik *crossover*.
5. Dimulai dari titik *crossover*, tukar gen-gen antara 2 kromosom.
6. Jika terdapat rute yang berulang maka hapus rute yang berulang tersebut.

Sedangkan proses mutasi merupakan proses diubahnya satu atau lebih nilai gen kromosom dalam satu populasi. Nilai gen tersebut akan digantikan dengan suatu nilai yang dipilih secara acak [BAS-12]. Mutasi ditentukan oleh probabilitas mutasi (PM), apabila nilai *random* yang dibangkitkan kurang dari pM, maka offspring akan mengalami proses mutasi. Jika bilangan *random* yang di bangkitkan lebih besar dari pM, maka tidak akan terjadi proses mutasi [MAR-13].

Metode mutasi yang digunakan pada skripsi ini adalah modifikasi dari metode yang digunakan oleh Karas dan Utila (2011). Langkah-langkah dari metode mutasi tersebut adalah sebagai berikut [KAR-11].

1. Bangkitkan bilangan *random* 0-1.
2. Jika bilangan *random* < probabilitas mutasi, lanjutkan langkah 4.
3. Jika bilangan *random* > probabilitas mutasi, maka tidak terjadi mutasi pada induk tersebut.
4. Pilih *random gen* untuk dijadikan titik mutasi, kecuali *gen* awal dan akhir.
5. *Generate* kromosom baru dengan menggunakan langkah-langkah pada inisial populasi dengan menggunakan *gen* pada titik mutasi sebagai *gen* awal.
6. Tukar *gen-gen* setelah titik mutasi dengan kromosom yang baru dibentuk. Ganti *gen* yang bernilai 0 dengan *gen* pada *node* berikutnya yang memiliki nilai lebih besar.
7. Hapus *loop* rute yang mungkin terjadi pada proses mutasi.

Ada tiga keunggulan dari aplikasi GA dalam proses optimasi [YUS - 12], yaitu:

1. GA tidak terlalu banyak memerlukan persyaratan matematika dalam penyelesaian proses optimasi.
2. Operasi evolusi dari GA sangat efektif untuk mengobservasi posisi global secara acak.
3. GA mempunyai fleksibilitas untuk diimplementasikan secara efisien pada problematika tertentu.

2.2 Android

Android merupakan sistem operasi yang biasa digunakan pada telepon seluler seperti telepon pintar (*smartphone*) dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri. Awalnya, Google Inc. membeli Android Inc., yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel atau telepon pintar [HUM-12]. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [SAF-12].

Android dikatakan sebagai *platform mobile* pertama yang lengkap, terbuka, dan bebas, berikut merupakan penjelasannya [SAF-12].

1. Lengkap (*Complete Platform*)

   Android merupakan system operasi yang aman dan banyak menyediakan tools dalam membangun software dan memungkinkan untuk melakukan pengembangan aplikasi.

2. Terbuka (*Open Source Platform*)

   Platform Android disediakan melalui lisensi *Open Source*. Jadi pengembang bebas dalam membangun aplikasi.

3. Bebas (*Free Platform*)

   Android merupakan aplikasi yang bebas untuk develop. Tidak ada lisensi atau biaya royalty untuk di kembangkan pada *platform* Android. Dan aplikasi android dapat didistribusikan dan di perdagangkan dalam bentuk apapun.

Android merupakan generasi baru *Platform Mobile*, *platform* yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang di harapkan. System operasi android di berlisensi di bawah GNU, General Public Lisensi Versi 2 (GPLv2) atau di kenal sebagai *copyleft*. Aplikasi Android dapat dikembangkan pada sistem operasi berikut [SAF-12]:

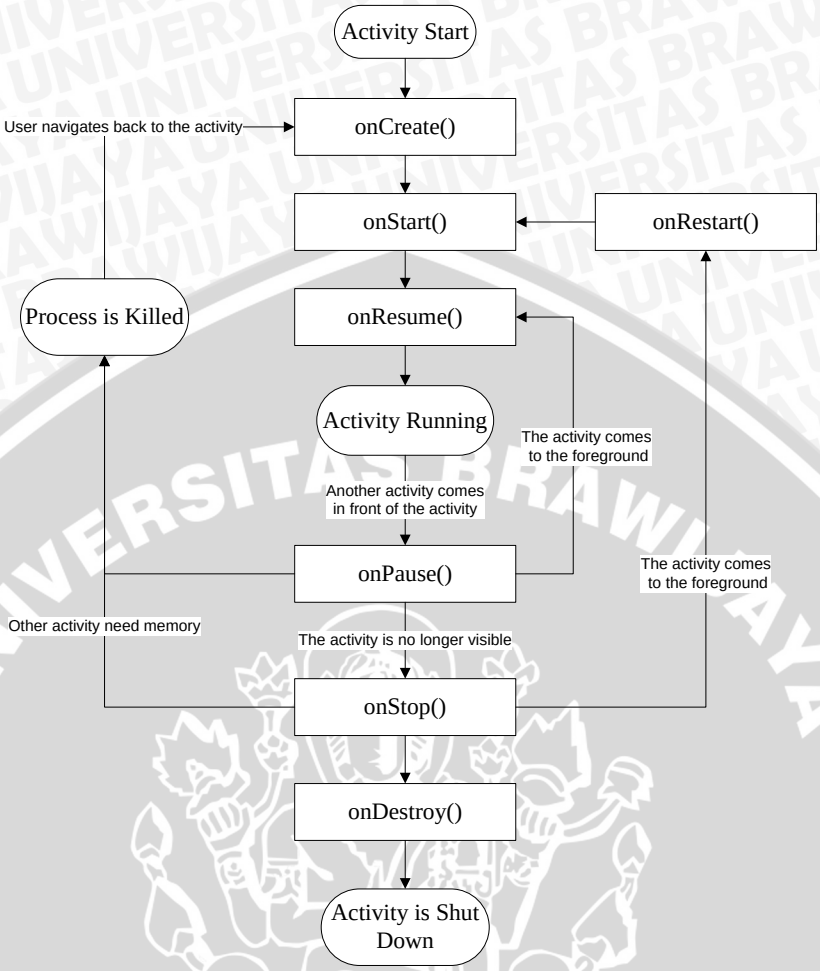
1. Windows XP Vista/Seven
2. Mac OS X (Mac OS X 10.4.8 atau lebih baru
3. Linux

Android memberikan sebuah lingkungan yang berbeda untuk pengembang. Android tidak membedakan antara aplikasi inti dengan aplikasi lain atau aplikasi pihak ketiga. Application Programming Interface (API) yang disediakan menawarkan akses ke perangkat keras maupun data-data pada ponsel atau data dari sistem sendiri. Bahkan pengguna dapat mengubah aplikasi inti ke applikasi pihak ketiga [ROM-12].

Sebagian besar para develop mengembangkan Android menggunakan *Eclipse* yang tersedia secara bebas untuk merancang dan mengembangkan aplikasi Android. *Eclipse* adalah *Integrated Development Environtment* (IDE) yang paling popular untuk pengembangan Android karena memilik Android Plug-in untuk pengembangan Android. Dan *Eclipse* pun mendapat lisensi langsung dari Google untuk menjadi IDE pengambangan aplikasi android [SAF-12].

Aplikasi pada android biasanya terdiri dari beberapa *activity* yang saling terkait. *Activity* merupakan suatu komponen aplikasi yang menyediakan user antarmuka pada layar sehingga pengguna dapat melakukan interaksi dengan aplikasi tersebut. *Life cycle activity* ditunjukkan Gambar 2.2.

Activity terdiri dari 4 state atau keadaan, yaitu *active*, *paused*, *stopped* dan *dead*. *Activity* dikatakan *active* yaitu pada saat berjalan atau *running* dan tampil pada antarmuka (*foreground*). *Paused* yaitu pada saat *activity* sedang berjalan namun ada sebuah dialog atau interupsi yang keluar. Sedangkan *stopped* yaitu keadaan dimana aplikasi tetap berjalan namun tersembunyikan oleh sesuatu yang lain. Dan yang terakhir *dead* yaitu keadaan dimana *activity* dihentikan, kemungkinan terjadi karena kurangnya *resource* atau *memory*.

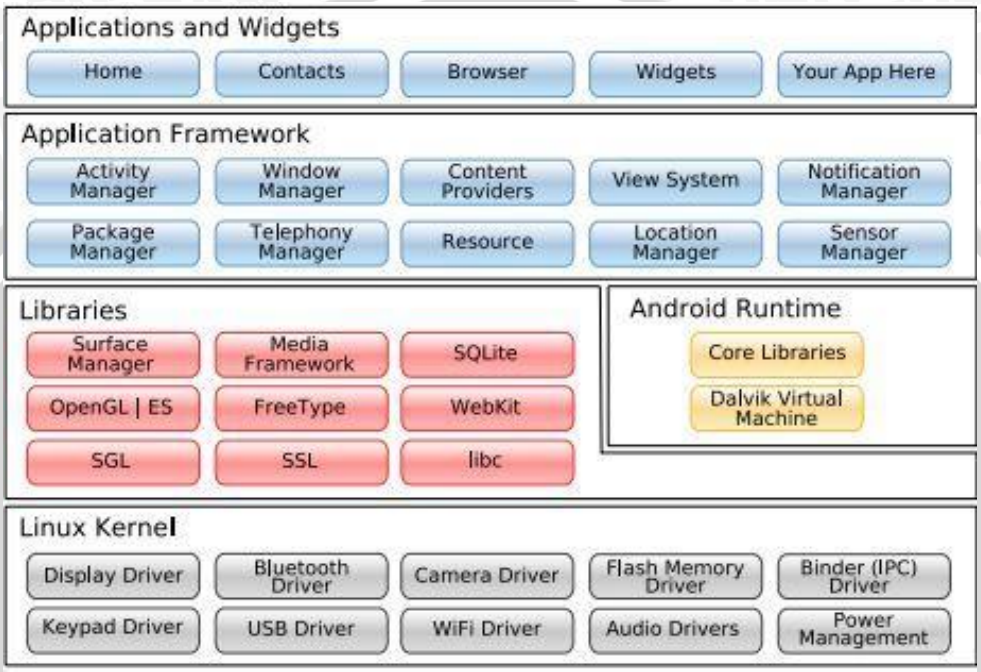


Gambar 2.2 Siklus *Activity* pada Android
(Sumber : Dr. Sean Ho, Trinity Western University)

Siklus hidup *activity* sesuai Gambar 2.1 yaitu:

a. *Activity* hidup (*activity exist*) yaitu keadaan diantara onCreate() dan onDestroy().
b. *Activity visible* yaitu keadaan diantara onStart() dan onStop().
c. *Activity* pada *foreground* yaitu keadaan diantara onResume() dan onPause(). Maksud dari berada pada *foreground* adalah *activity* menerima input dari pengguna.
d. Pada saat *activity* ditunda atau berada dalam keadaan *paused* ada kemungkinan dimusnahkan (*destroyed*) sebelum *activity* tersebut dilanjutkan kembali (onResume()).

Android memiliki arsitektur yang terdiri dari 5 bagian besar yaitu *application, application framework, libraries, android runtime* dan linux kernel. Arsitektur tersebut menggambarkan bahwa Android merupakan sebuah sistem yang memiliki desain arsitektur yang berbeda dengan arsitektur sistem yang lain. Gambar 2.3 merupakan gambar dari arsitektur Android [DYA-12].



Gambar 2.3 Arsitektur Android

Sumber: [ROM-12]

a. *Aplication* dan *Widgets*

Application dan *Widgets* ini adalah *layer* yang berhubungan dengan aplikasi. Pada layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Hampir semua aplikasi ditulis menggunakan bahasa pemrograman Java [SAF-12].

b. *Application Framework*

Application Frameworks adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *content providers* yang berupa sms dan panggilan telepon [ROM-12].

Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut [SAF-12]:

1. *Views*
2. *Content Provider*
3. *Resource Manager*
4. *Notification Manager*
5. *Activity Manager*

c. *Libraries*

Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat atau pengembang aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas Kernel, layer ini meliputi berbagai library C/C++ inti seperti Libc SSL [SAF -12], serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk dukungan *database.*
5. *Libraries* SSL dan WebKit terintegrasi dengan *web browser* dan *security.*
6. *Libraries* LiveWebcore mencakup modern *web browser* dengan *engine embedded web view.*
7. *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's.

d. Android *Run Time*

Android *Run Time* adalah *layer* yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. Android *Run Time* dibagi menjadi dua bagian yaitu *Core Libraries* dan *Dalvik Virtual Machine* (DVM) [ROM-12].

e. Linux Kernel

Linux Kernel adalah *layer* dimana inti dari sistem operasi Android itu berada. *Layer* ini berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya [SAF - 12].

Berikut merupakan beberapa API utama yang disediakan oleh Android API tersebut antara lain *Graphical User Interface* (GUI), akses *storage*, manipulasi grafik, manipulasi peta dan akses Location Based *Service* (LBS) [HAB-11].

a. *Graphical User Interface* (GUI)

Package android.view menyediakan berbagai kelas-kelas yang akan digunakan untuk menangani *screen, layout*, dan interaksinya dengan pengguna [HAB-11].

b. Akes *Storage*

Android menggunakan mekanisme *storage* yang berbeda dengan sistem operasi yang konvensional dimana setiap file dalam Android bersifat private terhadap aplikasi tersebut [HAB-11].

c. Manipulasi Grafik

Package android.graphics menyediakan manipulasi grafik *low-level* seperti kanvas, point, pewarnaan, dan manipulasi bentuk pada screen [HAB-11].

d. Manipulasi Peta

Package com.google.android.maps menyediakan API untuk mengakses Google Map [HAB-11].

e. Akses *Location Based Services*

Package android.location berisi kelas-kelas untuk mengakses berbagai layanan berbasis lokasi [HAB-11].

DVM dirancang khusus oleh para *engineer* Google sebagai bagian dari *platform mobile* Android. Dalvik adalah sebuah *virtual machine* (VM) berbasis register yang telah dioptimasi untuk berjalan pada perangkat *embedded* dengan memory minim. Dalvik juga dirancang untuk memungkinkan beberapa instans VM berjalan sekaligus secara efisien. Dalvik bergantung pada kernel linux untuk menyediakan fungsionalitas level rendah seperti isolasi proses, *threading* dan manajemen memori *low-level* [HAB-11].

Android SDK (*Software Development Kit*) adalah tools API (Application Programming Interface) yang diperlukan untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman Java. Saat ini Android SDK merupakan alat bantu dan API untuk pengembangan aplikasi Android yang menggunakan bahasa pemrograman Java. [SAF-12].

Aplikasi Android nantinya tidak akan berjalan langsung diatas kernel sistem operasi namun berjalan diatas Dalvik, sebuah *virtual machine* yang khusus dirancang untuk digunakan pada sistem *embedded* [HAB-11].

Android Development Tools (ADT) adalah plugin Eclipse IDE yang dirancang khusus untuk memberikan *integrated environment* yang kuat untuk membuat aplikasi Android. ADT memberikan kemampuan kepada Eclipse untuk membuat projek baru Android secara cepat, membuat aplikasi *user interface*, menambahkan komponen berdasarkan Android *framework* API, melakukan *debugging* aplikasi yang dibuat dengan menggunakan Android SDK *tools* dan serta melakukan distribusi aplikasi yang dibuat. Dengan disediakannya *project setup*, serta *tools* yang sudah terintegrasi di dalamnya, *custom* XML *editor*, dan *debugging* beserta *output* dalam *emulator* yang sudah disediakan Android SDK, mempermudah pengembang aplikasi Android dalam pembuatan aplikasinya [SAF-12].

2.3 SQLite

SQLite merupakan sistem manajemen basis data relasional yang ditulis dengan menggunakan bahasa C dan bersifat ACID-compliant serta memiliki ukuran library yang relative kecil. Protokol komunikasi utama yang digunakan

oleh SQLite adalah melalui pemanggilan API langsung melalui bahasa pemrograman sehingga dapat mengurangi *overhead* dan *latency time*.

Tipe data yang ada pada SQLite antara lain TEXT (mirip dengan String pada bahasa pemrograman Java) dan REAL (mirip dengan Double pada bahasa pemrograman Java). Sebelum disimpan pada basis data, semua tipe data harus dikonversi terlebih dahulu pada tipe ini. Namun, tidak ada validasi untuk tipe yang dituliskan pada kolom apakah sesuai dengan tipe yang telah didefinisikan. Misalnya, pengguna dapat menuliskan integer ke string atau sebaliknya.

Untuk mengakses basis data SQLite melibatkan pengaksesan sistem file. Oleh karena itu direkomendasikan untuk menggunakan operasi asinkron ketika melakukan operasi basis data agar performa tidak menjadi lebih lambat [HER-13].

2.4 *Google Maps*

Google Maps merupakan layanan pemetaan berbasis *web service* yang disediakan oleh Google dan bersifat gratis. *Google Maps* juga memiliki sifat *server side*, yaitu peta yang tersimpan pada server Google dapat dimanfaatkan oleh pengguna. *Google Maps* API adalah suatu *library* yang berbentuk javascript yang berguna untuk memodifikasi peta yang ada di *Google Maps* sesuai kebutuhan [ELI-12].

Google Maps API memiliki 4 jenis pilihan model peta yang disediakan oleh Google [ROM-12], yaitu:

1. ROADMAP, untuk menampilkan peta 2 dimensi.
2. SATELLITE, untuk menampilkan foto satelit.
3. TERRAIN, untuk menampilkan relief fisik permukaan bumi dan menunjukkan seberapa tingginya suatu lokasi, contohnya menunjukkan gunung dan sungai.
4. HYBRID, untuk menampilkan foto satelit yang diatasnya tergambar pula apa yang tampil pada ROADMAP seperti nama jalan dan nama kota.

2.5 *Location Based Service* (LBS)

Location Based Service adalah *service* yang berfungsi untuk mencari dengan teknologi *Global Positioning Service* (GPS) dan *Google's cell-based*

location. *Map* dan layanan berbasis lokasi menggunakan lintang dan bujur untuk menentukan lokasi geografis, namun *user* membutuhkan alamat atau posisi *realtime*, bukan nilai lintang dan bujur. Android menyediakan geocoder yang mendukung *forward* dan *reverse geocoding*. Dengan menggunakan *geocoder*, nilai lintang bujur menjadi alamat dunia nyata atau sebaliknya dapat dikonversi. Dua unsur utama LBS yaitu [ROM-12]:

1. *Location Manager* (API *Maps*)

   API *Maps* atau *Application Programming Interface Maps* menyediakan *tools/resource* untuk LBS. API *Maps* memiliki fasilitas untuk menampilkan, memanipulasi maps/peta beserta fitur lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya.

2. *Location Providers* (API Location)

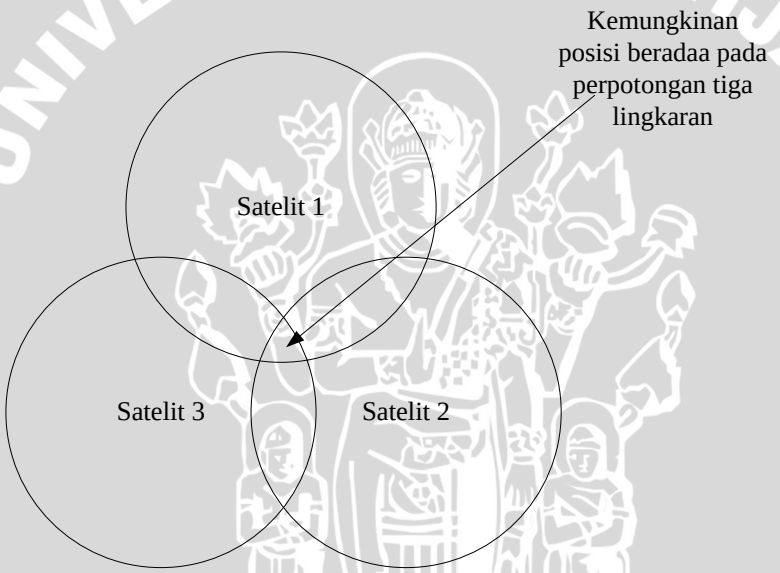
   Location Providers menyediakan teknologi pencarian lokasi yang digunakan oleh sebuah perangkat. API Location berhubungan dengan data GPS dan data lokasi *real-time*. API Location berada pada paket Android yaitu dalam paket android.location. Lokasi perangkat, *track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan dapat ditentukan dengan *Location Manager*.

2.6 *Global Positioning System* (GPS)

GPS atau *Global Positioning System* merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima (*receiver*) di permukaan, dimana GPS receiver ini akan mengumpulkan informasi dari satelit GPS [ROM-12], seperti:

1. Waktu. GPS receiver menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
2. Lokasi. GPS memberikan informasi lokasi dalam tiga dimensi:
   a. Latitude
   b. Longitude
   c. Elevasi

3. Kecepatan. Ketika berpindah tempat, GPS dapat menunjukkan informasi kecepatan berpindah tersebut.
4. Arah perjalanan. GPS dapat menunjukkan arah tujuan.
5. Simpan lokasi. Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS receiver.
6. Komulasi data. GPS receiver dapat menyimpan informasi track, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan, dan sebagainya.



Gambar 2.4 Trilaterasi pada GPS
(Sumber [HAB-11])

Untuk menentukan posisi. GPS receiver harus berada dalam *line-of-sight* (LoS) terhadap ketiga satelit untuk menentukan posisi, sehingga GPS hanya ideal untuk diguakan dalam *outdoor positioning*, seperti Gambar 2.4. Aplikasi yang berada di sisi target (server) setelah mendapatkan *request* dari pelacak (*client*) maka server akan meminta koordinat posisinya pada GPS, yang kemudian akan dikirimkan ke pelacak [HAB-11].

Posisi yang ditunjukkan oleh suatu GPS mempunyai faktor kesalahan atau juga disebut tingkat akurasi. Sebagai contoh suatu alat GPS menunjukkan titik

koordinat dengan tingkat akurasi 5 meter, itu berarti posisi pengguna bisa berada dalam range radius 5 meter dari titik yang ditunjukkan tersebut. Ada beberapa hal yang mempengaruhi tingkat akurasi tersebut [ROM-12], antara lain:

1. Kesalahan Ephemeris

   Terjadi jika satelit tidak dapat mentransmisikan posisinya di orbit dengan tepat.

2. Keadaan Ionosphere

   Ionosphere berada pada jarak sekitar 43-50 mil di atas permukaan bumi. Satelit yang melewati ionosphere akan menjadi lambat dikarenakan adanya plasma (gas dengan tingkat kepadatan rendah). Walaupun GPS receiver berusaha untuk mengkoreksi/memperbaiki faktor keterlambatan yang terjadi tetap saja aktivitas tertentu dari plasma bisa menyebabkan kesalahan perhitungan.

3. Keadaan Troposphere

   Troposphere adalah bagian terendah dari atmosfer sampai dengan ketinggian sekitar 11 mil dari permukaan tanah. Variasi pada temperatur, tekanan, dan kelembaban bisa menyebabkan perbedaan kecepatan penerimaan gelombang radio.

4. Kesalahan Waktu

   Karena penempatan jam atom pada setiap GPS receiver tidak berjalan sebagaimana mestinya. Kesalahan waktu dari GPS receiver yang tidak presisi dapat menimbulkan ketidakakurasian.

5. Kesalahan Multipath

   Terjadi karena sinyal satelit membentur permukaan keras (seperti bangunan atau tebing) sebelum mencapai GPS receiver. Hal tersebut bisa menyebabkan terjadinya delay sehingga perhitungan jarak menjadi tidak akurat.

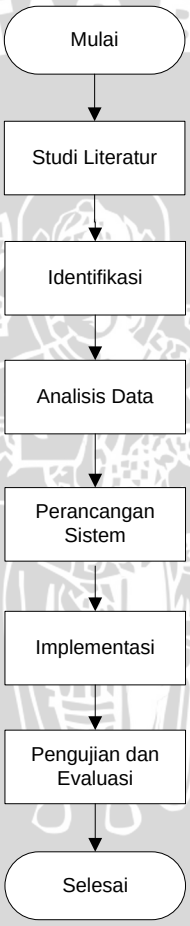
6. Buruknya Sinyal Satelit

   Keadaan langit yang terhalang akan menyebabkan GPS sulit menerima data satelit. Sebuah sinyal satelit yang pada hari tertentu diterima dengan sangat bagus belum tentu pada hari lain bisa diterima dengan kualitas yang sama walaupun user berdiri pada tempat yang sama. Hal tersebut

dikarenakan posisi dari satelit yang terus bergerak atau bisa juga disebabkan faktor penghalang lain seperti pohon, gedung bertingkat, dan sebagainya.

BAB III
METODOLOGI PENELITIAN

Metode penelitian yang digunakan dalam pembuatan aplikasi terdiri dari studi literatur, identifikasi, perancangan sistem, implementasi, pengujian dan evaluasi. Gambar 3.1 merupakan diagram alir dari metodologi penelitian yang dilakukan.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Pada tahap ini digunakan untuk memperdalam konsep dan teori yang akan diterapkan pada pembuatan aplikasi "Rancang Bangun Aplikasi Pencarian Rute Terpendek untuk Menemukan SPBU Terdekat di Kota Malang dengan Menggunakan Algoritma Genetik Berbasis Android" hal ini sangat penting dilakukan sebelum melakukan perancangan aplikasi. Metode yang dipelajari dalam studi literatur adalah:

1. Algoritma Genetik
2. Pemrograman Android
3. *SQLite*
4. *Google Maps*
5. *Location Based Service* (LBS)
6. *Global Positioning System* (GPS)

3.2 Identifikasi

Tahap ini merupakan tahap awal dalam pembuatan aplikasi. Tahapan ini merupakan tahapan untuk mengkaji dan membatasi masalah yang akan di implementasikan dalam aplikasi, mengumpulkan berbagai kebutuhan yang diperlukan sistem dan mengumpulkan data yang berkaitan dengan aplikasi yang akan dibuat. Identifikasi ini merupakan tahap awal yang akan menentukan keberhasilan dari tahap-tahap berikutnya.

3.3 Analisis Data

Analisis data merupakan tahap dimana dilakukan analisis terhadap data yang telah terkumpul. Pada tahap ini juga dilakukan pengujian terhadap data sampel dalam memecahkan suatu permasalahan dengan menggunakan Algotitma Genetik. Selain itu juga dilakukan analisis terhadap data jalan atau jalur yang akan dilalui karena tidak semua jalan diikutsertakan. Pemilihan jalan ini dilakukan berdasarkan fungsi dan lebar jalan. Jalan-jalan yang digunakan pada aplikasi ini yaitu jalan arteri, jalan kolektor, jalan lokal. Penjelasan masing-masing jalan berdasarkan UU No.38 Tahun 2004 yaitu sebagai berikut [NIZ-14].

Jalan Arteri Primer adalah ruas jalan yang menghubungkan antar kota yang berdampingan atau menghubungkan kota jenjang kesatu dengan kota jenjang kedua.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan Arteri Primer adalah :

a. Kecepatan rencana > 60 km/jam.
b. Lebar badan jalan > 8,0 m.
c. Kapasitas jalan lebih besar dari volume lalu lintas rata-rata.
d. Jalan masuk dibatasi secara efisien sehingga kecepatan rencana dan kapasitas jalan dapat tercapai.
e. Tidak boleh terganggu oleh kegiatan lokal, lalu lintas lokal.
f. Jalan primer tidak terputus walaupun memasuki kota.

Jalan Arteri Sekunder adalah ruas jalan yang menghubungkan kawasan primer dengan kawasan sekunder kesatu atau menghubungkan kawasan sekunder kesatu dengan kawasan sekunder lainnya.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan Arteri Sekunder adalah :

a. Kecepatan rencana > 30 km/jam.
b. Lebar jalan > 8,0 m.
c. Kapasitas jalan lebih besar atau sama dari volume lalu lintas rata-rata.
d. Tidak boleh diganggu oleh lalu lintas lambat.

Jalan Kolektor Primer adalah ruas jalan yang menghubungkan antar kota kedua dengan kota jenjang kedua, atau kota jenjang kesatu dengan kota jenjang ketiga.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan Kolektor Primer adalah :

a. Kecepatan rencana > 40 km/jam.
b. Lebar badan jalan > 7,0 m.
c. Kapasitas jalan lebih besar atau sama dengan volume lalu lintas rata-rata.

d. Jalan masuk dibatasi secara efisien sehingga kecepatan rencana dan kapasitas jalan tidak terganggu.
e. Tidak boleh terganggu oleh kegiatan lokal, lalu lintas lokal.
f. Jalan kolektor primer tidak terputus walaupun memasuki daerah kota.

Jalan Kolektor Sekunder adalah ruas jalan yang menghubungkan kawasan sekunder kedua dengan kawasan sekunder lainnya atau menghubungkan kawasan sekunder kedua dengan kawasan sekunder ketiga.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan Kolektor Sekunder adalah :

a. Kecepatan rencana > 20 km/jam.
b. Lebar jalan > 7,0 m.

Jalan Lokal Primer adalah ruas jalan yang menghubungkan kota jenjang kesatu dengan persil, kota jenjang kedua dengan persil, kota jenjang ketiga dengan kota jenjang ketiga lainnya, kota jenjang ketiga dengan kota jenjang di bawahnya.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan Lokal Primer adalah :

a. Kecepatan rencana > 20 km/jam.
b. Lebar badan jalan > 6,0 m.
c. Jalan lokal primer tidak terputus walaupun memasuki desa

Sedangkan untuk arah arus jalan ditentukan secara langsung dengan mendatangi jalan-jalan di Kota Malang dan dengan melihat arah arus pada *Google Maps*. Hal ini dilakukan karena ada beberapa ruas jalan yang memiliki arah arus yang berbeda pada jam-jam tertentu.

Data jalan ini akan disimpan dalam bentuk node pada database SQLite. Node ditentukan dari percabangan/persimpangan jalan atau perubahan nama jalan, bukan setiap nama jalan. Hal tersebut dilakukan agar perhitungan total jarak sesuai dari jalan yang dilalui. Jika node dibentuk dari setiap nama jalan, maka panjang jalan akan dihitung seluruhnya, sehingga total jarak menuju SPBU akan semakin panjang.

3.4 Perancangan Sistem

Perancangan sistem merupakan tahap dimana penulis mulai merancang alur kerja aplikasi yang mampu memenuhi semua kebutuhan fungsionalitas yang diperlukan oleh aplikasi yang akan dibuat. Teori-teori dan pengetahuan yang telah diperoleh akan digabungkan dan diimplementasikan dalam pembuatan Aplikasi Pencarian Rute Terpendek untuk Menemukan SPBU Terdekat.

Tahap ini lebih banyak membahas masalah bagaimana mengolah data dan bagaimana sistem bekerja, data-data yang telah dikumpulkan akan saling dikaitkan hingga terbentuk suatu relasi. Selain itu, juga membahas bagaimana alur kerja sistem dalam mengolah data yang diinputkan agar memperoleh hasil berupa jalur optimum untuk menuju suatu SPBU.

Peran Algoritma Genetik pada aplikasi yaitu untuk menghasilkan rute terpendek menuju SPBU dengan mengolah data yang telah tersimpan pada database SQLite. Data-data yang ada diolah sesuai dengan tahap-tahap yang ada pada Algoritma Genetik hingga terbentuk sebuah rute yang digambarkan pada *Google Maps*.

3.5 Implementasi

Apabila semua data dan pengetahuan telah diperoleh, serta cara dalam mengatasi suatu masalah atau kesulitan telah didapatkan, maka tahap selanjutnya yaitu tahap implementasi dapat dilaksanakan. Aplikasi diimplementasikan dengan mengacu pada tahap perancangan yang dilakukan sebelumnya. Pada tahap ini akan diperlukan beberapa hal antara lain input, proses, dan output yang dihasilkan oleh aplikasi.

Aplikasi akan dibuat dengan menggunakan software Eclipse dengan bahasa pemrograman Java dan menggunakan androin SDK untuk Windows. Android SDK merupakan *Software Development Kit* untuk pengembangan aplikasi berbasis android yang digunakan pada Eclipse. Selain itu juga diperlukan *Google Maps* API untuk menampilkan peta Kota Malang. Pada peta tersebut juga akan ditampilkan posisi user serta posisi SPBU yang ada di Kota Malang, sehingga pengguna dapat mengikuti rute jalan yang dihasilkan oleh aplikasi dengan menggunakan Algoritma Genetik.

3.6 Pengujian dan Evaluasi

Setelah semua tahap telah selesai dilakukan maka hasil yang diperoleh (berupa aplikasi) tentu harus diuji dan dievaluasi secara keseluruhan untuk menguji dan menemukan kesalahan yang mungkin terjadi guna memperoleh hasil yang maksimal. Pengujian dilakukan dengan menggunakan metode *black-box testing*, yaitu untuk memastikan bahwa aplikasi yang dibangun memenuhi kebutuhan-kebutuhan yang didefinisikan pada bab perancangan. Serta pengujian terhadap algoritma untuk menguji pengaruh probabilitas *crossover* dan probabilitas mutasi, pengaruh jumlah iterasi atau generasi serta pengujian validasi hasil untuk menguji apakah hasil perhitungan oleh aplikasi sama dengan hasil perhitungan manual.

BAB IV
PERANCANGAN

Pada bab ini akan dibahas mengenai analisis kebutuhan dan perancangan terhadap sistem. Sebelum masuk pada tahap perancangan terlebih dahulu dilakukan analisis kebutuhan yang terdiri dari aktor yang nantinya akan berinteraksi secara langsung pada aplikasi dan daftar kebutuhan yang diperlukan serta perhitungan manual Algoritma Genetik yang diimpementasikan pada aplikasi. Selanjutnya kebutuhan-kebutuhan tersebut akan dibentuk dalam suatu perancangan perangkat lunak yang terdiri dari *activity diagram, sequence diagram, class diagram,* perancangan basis data dan perancangan desain antarmuka. Diagram pohon perancangan sistem ditunjukkan oleh Gambar 4.1.

Gambar 4.1 Diagram Pohon Perancangan

4.1 Gambaran Umum Sistem

Gambaran umum sistem merupakan representasi desain arsitektur sistem yang dibuat secara umum. Tahap ini merupakan tahapan awal dari perancangan sistem yang akan dibuat. Gambaran umum sistem ditunjukkan pada Gambar 4.2.

Gambar 4.2 Gambaran Umum Sistem

Pengguna menjalankan aplikasi lalu memilih menu pencarian rute. Setelah itu pengguna melakukan set posisi. Set posisi dapat dilakukan secara otomatis oleh sistem dan secara manual oleh pengguna. Secara manual, set posisi dilakukan oleh pengguna dengan memilih marker jalan tempat pengguna berada atau marker jalan terdekat dengan posisi pengguna yang ditampilkan pada *Google Maps*. Marker jalan yang muncul pada aplikasi merupakan marker jalan yang telah tersimpan pada basis data SQLite dalam bentuk titik longitude dan latitude. Set posisi secara otomatis dilakukan oleh sistem dengan mendeteksi posisi pengguna dan mencari posisi marker terdekat dari posisi pengguna. Set posisi pengguna ini yang akan diproses lebih lanjut dengan menggunakan Algoritma Genetik. Hasil dari perhitungan akan ditampilkan pada layar *smartphone* berupa rute jalan menuju ke SPBU.

4.2 Analisis Kebutuhan Perangkat Lunak

Aplikasi yang dirancang adalah aplikasi perangkat bergerak yang dikembangkan pada *platform* Android. Kegunaan dari aplikasi adalah untuk membantu pengguna dalam mencari SPBU terdekat sekaligus mendapatkan rute terpendek untuk menuju ke SPBU.

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan. Sehingga kebutuhan-kebutuhan yang harus dipenuhi pada aplikasi didasarkan pada kebutuhan dari pengguna aplikasi.

4.2.1 Identifikasi Aktor

Aktor yang berinteraksi secara langsung pada perangkat lunak adalah pengguna aplikasi. Pengguna aplikasi memiliki hak akses pada seluruh fitur yang ada pada aplikasi.

4.2.2 Kebutuhan Fungsional

Daftar kebutuhan berisi semua kebutuhan yang diperlukan oleh aplikasi yang akan dibangun. Kebutuhan fungsional merupakan kebutuhan yang terdiri dari fitur-fitur yang disediakan oleh aplikasi untuk memenuhi kebutuhan pengguna.

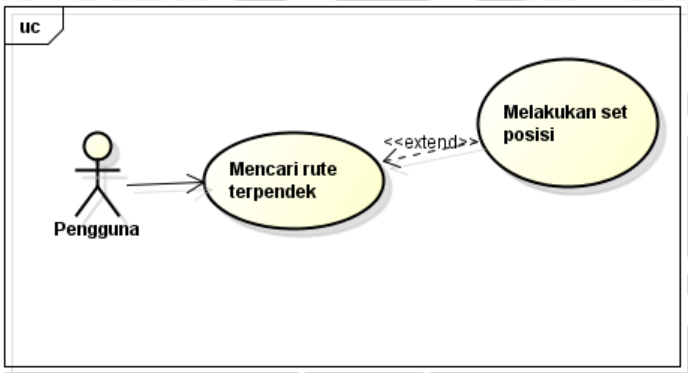
Kebutuhan fungsional ditunjukkan dengan penomoran SRS (*Software Requirement Specification*) serta digambarkan dengan menggunakan diagram *use case*. Daftar kebutuhan fungsional ditunjukkan pada Tabel 4.1.

Tabel 4.1 Spesifikasi Kebutuhan Fungsional

Nomor SRS	Kebutuhan	*Use Case*
SRS_001	Aplikasi harus dapat menampilkan posisi pengguna pada *Google Maps* serta pengguna dapat melakukan set posisi secara manual dengan memilih salah satu marker yang muncul pada setiap jalan di Kota Malang maupun dapat memilih untuk set posisi secara otomatis oleh sistem.	Melakukan set posisi
SRS_002	Aplikasi harus dapat menampilkan rute jalan menuju ke SPBU terdekat dari hasil perhitungan dengan menggunakan Algoritma Genetik.	Mencari rute terpendek

4.2.2.1 Diagram *Use Case*

Diagram *Use Case* merupakan diagram yang diunakan untuk menggambarkan perilaku sistem serta menggambarkan kebutuhan yang diperlukan oleh aktor dari suatu sistem atau aplikasi. Gambar 4.3 merupakan diagram *use case* dari daftar kebutuhan fungsional aplikasi.



Gambar 4.3 Diagram *Use Case*

4.2.2.2 Skenario *Use Case*

Skenario *use case* digunakan untuk menjelaskan secara lebih mendetail tentang masing-masing kebutuhan fungsional yang terdapat pada diagram *use case*.

Kebutuhan fungsional untuk melihat posisi pengguna pada *Google Maps* serta pengguna dapat melakukan set posisi dengan memilih salah satu marker jalan atau memilih set posisi yang secara otomatis dilakukan sistem untuk dikelola lebih lanjut dengan Algoritma Genetik ditunjukkan oleh *use case* melakukan set posisi. Setiap titik jalan yang terdaftar pada basis data akan ditampilkan pada *Google Maps* dengan menggunakan marker. Pengguna melakukan set posisi secara manual dengan memilih salah satu marker yang letaknya dekat dengan posisi pengguna sesuai posisi yang ditunjukan oleh *Google Maps*. Sedangkan set posisi otomatis dilakukan oleh sistem dengan mendeteksi titik terdekat dari posisi pengguna. Marker jalan yang dipilih pengguna maupun titik yang terdeteksi sebagai posisi atau jalan terdekat dari posisi pengguna akan menjadi lokasi atau titik awal untuk pencarian rute menuju SPBU dengan menggunakan Algoritma Genetik. Tabel 4.2 merupakan skenario *use case* melakukan set posisi.

Tabel 4.2 Skenario *Use Case* Melakukan Set Posisi

Skenario *Use Case* Melakukan Set Posisi	
Kode SRS	SRS_001
Nama Use Case	*Use Case* Melakukan Set Posisi
Tujuan	Untuk melakukan set posisi pengguna.
Deskripsi	*Use Case* ini digunakan untuk melakukan set posisi pengguna baik secara otomatis yang dilakukan oleh sistem maupun secara manual dengan memilih marker pada jalan yang ada pada *Google Maps*.
Aktor	Pengguna
Pemicu	Pengguna memilih menu pencarian rute terdekat.
Kondisi Awal	Sistem menampilkan halaman utama aplikasi.
Skenario Utama (*Basic Flow*)	
1. Pengguna memilih menu pencarian rute terdekat. 2. Sistem menampilkan Google Maps beserta marker jalan. 3. Sistem menampilkan pilihan set posisi.	
Skenario Bagian (*Sub Flow*)	
Set Posisi Otomatis	1. Pengguna memilih set posisi secara otomatis. 2. Sistem menampilkan rute terpendek hasil perhitungan Algoritma Genetik.
Set Posisi Manual	1. Pengguna memilih set posisi secara manual. 2. Pengguna memilih salah satu marker. 3. Sistem menampilkan rute terpendek hasil perhitungan Algoritma Genetik.
Kondisi Akhir	Sistem menampilkan rute terpendek hasil perhitungan Algoritma Genetik.

Hasil perhitungan dengan menggunakan Algoritma Genetik dapat ditampilkan dalam bentuk rute pada *Google Maps*. Kebutuhan fungsional untuk melihat rute hasil dari perhitungan dengan menggunakan Algoritma Genetik ditunjukkan oleh *use case* mencari rute terpendek. Tabel 4.3 merupakan skenario *use case* mencari rute terpendek.

Tabel 4.3 Skenario *Use Case* Mencari Rute Terpendek

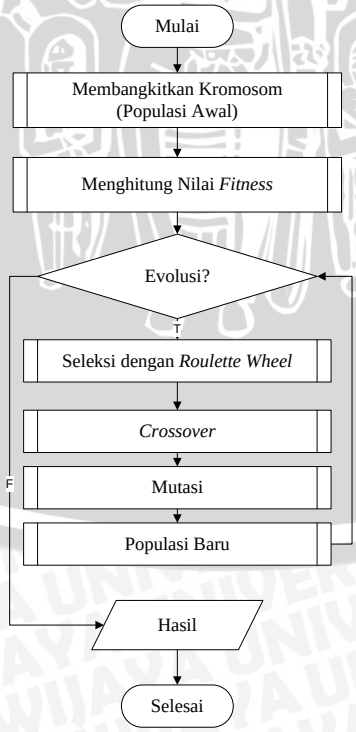
Skenario *Use Case* Mencari Rute Terpendek	
Kode SRS	SRS_002
Nama Use Case	*Use Case* Mencari Rute Terpendek
Tujuan	Untuk melihat rute hasil perhitungan Algoritma Genetik.
Deskripsi	*Use Case* ini digunakan untuk melihat rute hasil perhitungan Algoritma Genetik yang ditampilkan dalam bentuk rute pada *Google Maps*.
Aktor	Pengguna
Pemicu	Pengguna menginputkan posisi.
Kondisi Awal	Sistem menampilkan halaman set posisi.
Skenario Utama (*Basic Flow*)	
1. Sistem menampilkan halaman set posisi. 2. Pengguna melakukan set posisi. 3. Sistem menampilkan rute terpendek hasil perhitungan Algoritma Genetik.	
Kondisi Akhir	Sistem menampilkan rute terpendek hasil perhitungan Algoritma Genetik.

4.3 Perancangan Algoritma Genetik

Algoritma Genetik pada aplikasi berfungsi untuk menentukan rute menuju ke SPBU terdekat. Proses Algoritma Genetik dimulai ketika pengguna melakukan set posisi. Set posisi ini berupa id dari sebuah node yang akan menjadi titik awal dari perhitungan Algoritma Genetik yaitu untuk menentukan gen-gen selanjutnya di setiap kromosom pada populasi awal.

Algoritma Genetik terdiri dari beberapa tahap yaitu membangkitkan kromosom (populasi awal), menghitung nilai *fitness* setiap kromosom, melakukan seleksi terhadap kromosom, melakukan *crossover* atau kawin silang, dan mutasi gen. Setelah semua tahap dilalui maka akan terbentuk populasi baru. Populasi baru ini didapat dari membandingkan nilai *fitness offspring* hasil mutasi dengan nilai *fitness* setiap kromosom pada populasi sebelumnya. Kromosom dengan nilai *fitness* yang besar akan digunakan pada proses evolusi selanjutnya [AFA-11]. Proses evolusi dilakukan hingga mencapai nilai konvergen atau sesuai dengan jumlah proses evolusi yang telah ditentukan.

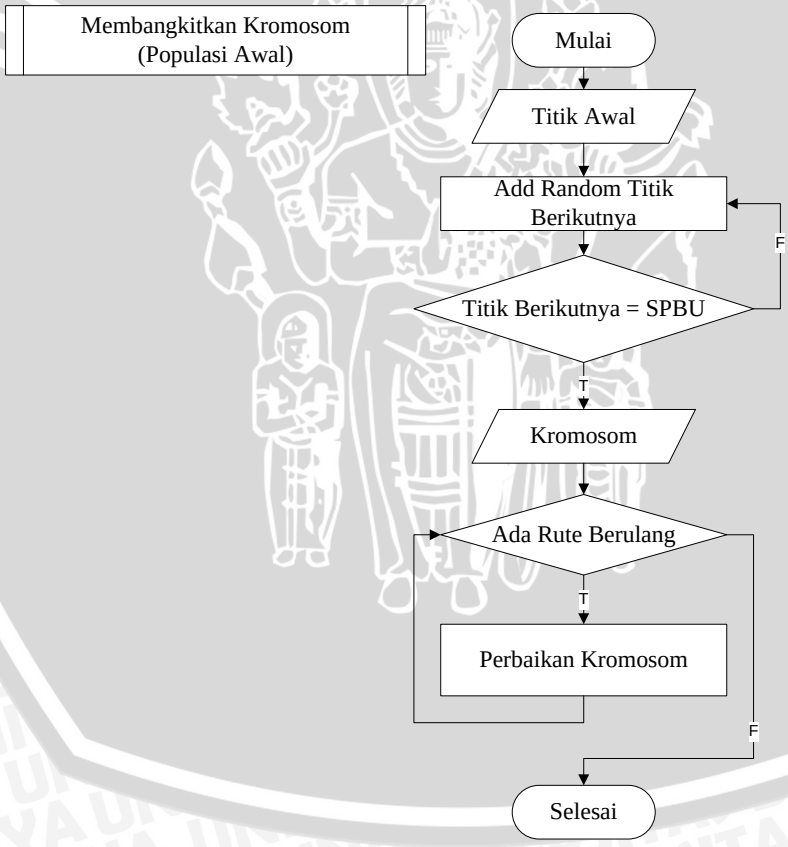
Langkah-langkah perhitungan menggunakan Algoritma Genetik ditunjukkan pada diagram alir Gambar 4.4.



Gambar 4.4 Diagram Alir Algoritma Genetik

4.3.1 Membangkitkan Kromosom (Populasi Awal)

Populasi awal terdiri dari 5 kromosom. Kromosom dibangkitkan secara *random* oleh sistem dengan gen (jalan) awal merupakan posisi yang diinputkan oleh pengguna dan gen akhir merupakan gen dimana terdapat SPBU. Gen pertama merupakan id atau node yang diperoleh dari set posisi. Gen selanjutnya merupakan mengambilan secara random salah satu node yang terhubung dengan node sebelumnya dengan menggunakan *query* ke database SQLite. Pencarian gen akan berhenti ketika nilai is_pom pada tabel data_jalan bernilai 1, artinya kromosom tersebut telah menemukan SPBU. Jika terdapat gen yang berulang maka akan mengalami proses perbaikan kromosom. Diagram alir membangkitkan kromosom ditunjukkan pada Gambar 4.5.



Gambar 4.5 Diagram Alir Membangkitkan Kromosom

Dimisalkan posisi pengguna berada pada jalan Soekarno-Hatta 1. Posisi pengguna berada didekat titik nomor 2. Sehingga populasi awal yang terbentuk yaitu:

2	26	28	29	33	32	31	27	1	2	26	24	28

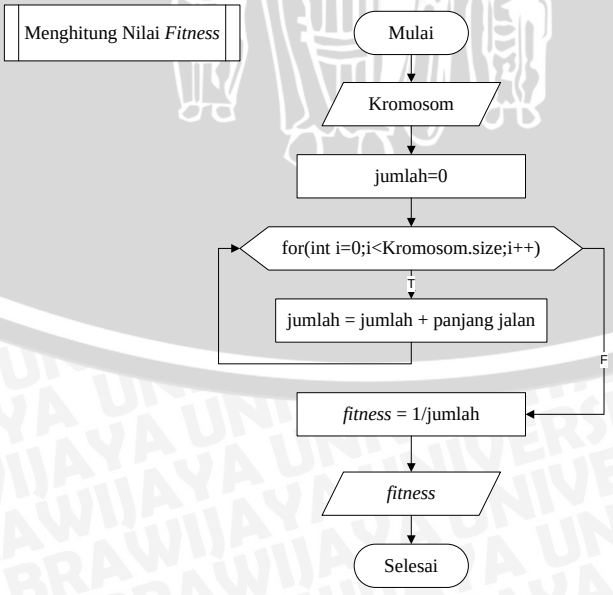
Kromosom tersebut memiliki rute yang berulang yaitu kolom dengan warna abu-abu karena terdapat titik dengan nomor 26 yang berulang. Rute yang berulang mengalami perbaikan kromosom untuk memotong gen yang berulang sehingga diperoleh:

2	26	24	28

Populasi awal terdiri dari 5 buah kromosom yang dibangkitkan secara *random*.

4.3.2 Perhitungan Nilai *Fitness*

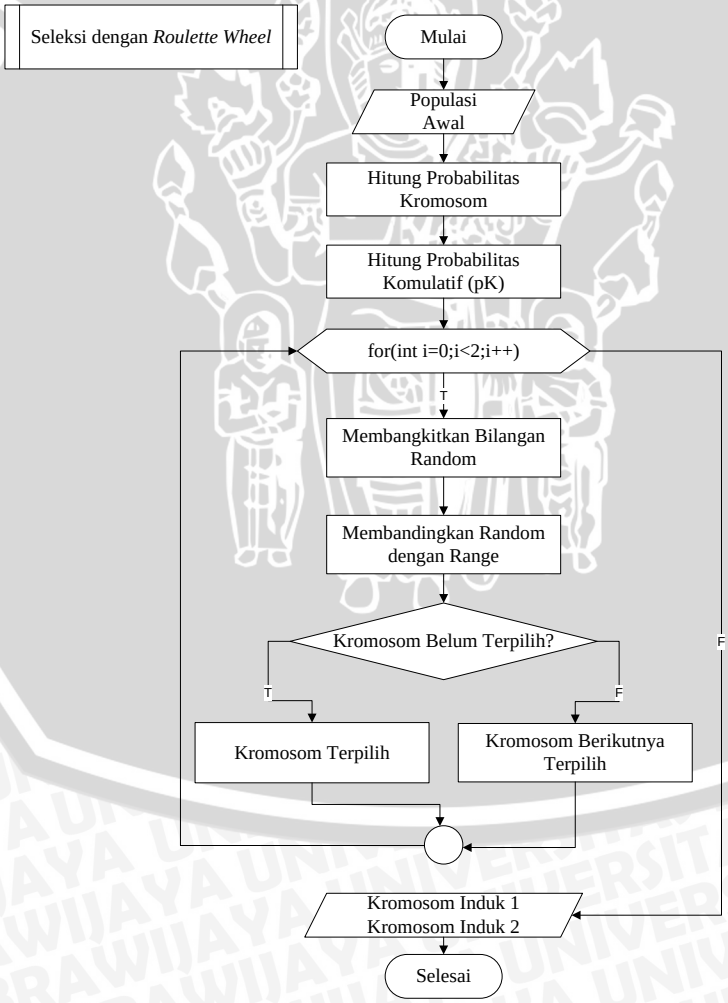
Nilai *fitness* dari setiap kromosom dihitung dengan inversi dari total jarak dari jalur yang diperoleh. Inversi dari total jarak dapat diperoleh menggunakan rumus 1/x, dimana x merupakan total jarak dari jalur yang diperoleh [FIR-13]. Jarak atau panjang setiap jalan diambil dari tabel data_jalan yaitu pada kolom panjang. Diagram alir perhitungan nilai *fitness* ditunjukkan Gambar 4.6.



Gambar 4.6 Diagram Alir Perhitungan Nilai *Fitness*

4.3.3 Seleksi dengan *Roulette Wheel*

Seleksi penentuan kromosom induk dilakukan dengan menggunakan *Roulette Wheel* karena dengan menggunakan metode ini setiap kromosom memiliki kemungkinan terpilih sesuai dengan besar probabilitas dari masing-masing kromosom. Serta tidak menutup kemungkinan kromosom dengan probabilitas kecil akan terpilih menjadi kromosom induk. Sehingga akan lebih banyak variasi kromosom induk yang dihasilkan (tidak harus dua kromosom dengan *fitness* terbaik yang menjadi kromosom induk) dan lebih banyak variasi *offspring* (anak) yang dihasilkan. Karena *offspring* dengan *fitness* terbaik belum tentu dihasilkan oleh dua kromosom induk dengan *fitness* terbaik walaupun kemungkinannya kecil. Urutan untuk melakukan seleksi dengan menggunakan Roulette Wheel digambarkan oleh diagram alir Gambar 4.7.



Gambar 4.7 Diagram Alir Seleksi dengan *Roulette Wheel*

Langkah-langkah seleksi dengan menggunakan Roulette Wheel sesuai dengan langkah-langkah yang ada pada bab 2. Seleksi dengan menggunakan *Roulette Wheel* dimulai dengan menghitung probabilitas setiap kromosom. Probabilitas dapat diperoleh menggunakan persamaan 2.2.

Setiap probabilitas akan dihitung komulatif probabilitasnya untuk menentukan range. Range yang dihasilkan ada 5 range karena terdapat 5 kromosom. Lalu dibangkitkan dua bilangan *random* pecahan antara 0 sampai dengan 1. Bilangan *random* yang dihasilkan akan dibandingkan dengan 5 range yang ada untuk menentukan nilai bilangan *random* tersebut termasuk pada range pertama, range kedua, range ketiga, range keempat atau range kelima.

Posisi bilangan *random* pada range akan menentukan kromosom mana yang akan mengalami proses selanjutnya. Misalnya, jika bilangan *random* pertama termasuk pada range ketiga dan bilangan *random* kedua termasuk pada range kedua, maka kromosom 3 dan kromosom 2 akan mengalami proses evolusi selanjutnya yaitu *crossover*.

4.3.4 *Crossover*

Crossover dipengaruhi oleh probabilitas *crossover* (pC). Jika dua bilangan *random* yang dibangkitkan < pC maka akan terjadi *crossover* pada dua kromosom induk. Sedangkan jika bilangan *random* yang dibangkitkan > pC maka tidak terjadi *crossover* [MAR-13].

Crossover dilakukan ketika pada dua kromosom terdapat gen yang sama. Hal ini dilakukan untuk menghindari putusnya rute jalan. Contoh *crossover* yaitu:
Kromosom 1 :

2	26	24	28

Kromosom 2 :

2	4	26	31	24	28

Gen yang sama adalah gen dengan warna abu-abu yaitu gen dengan node 26, maka hasil dari *crossover* dua kromosom tersebut yaitu:

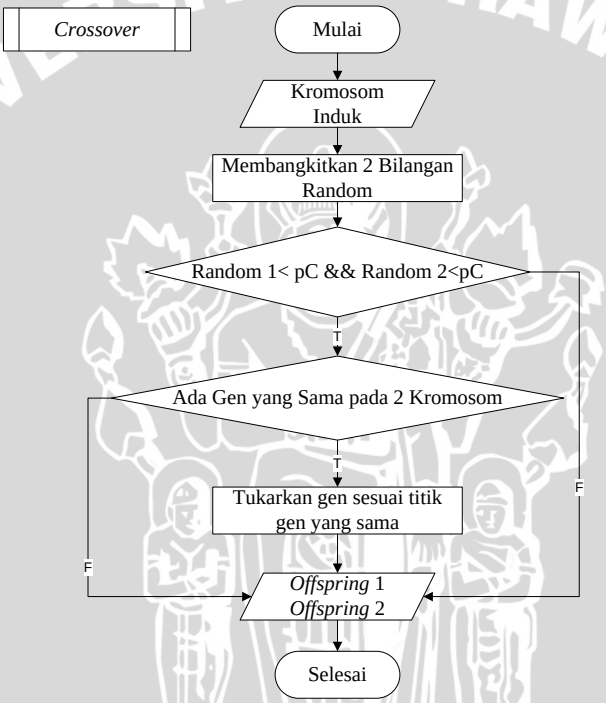
Offspring 1 :

2	4	26	24	28

Offspring 2 :

2	26	31	24	28

Urutan untuk melakukan *crossover* antara dua buah kromosom ditunjukkan oleh diagram alir Gambar 4.8.

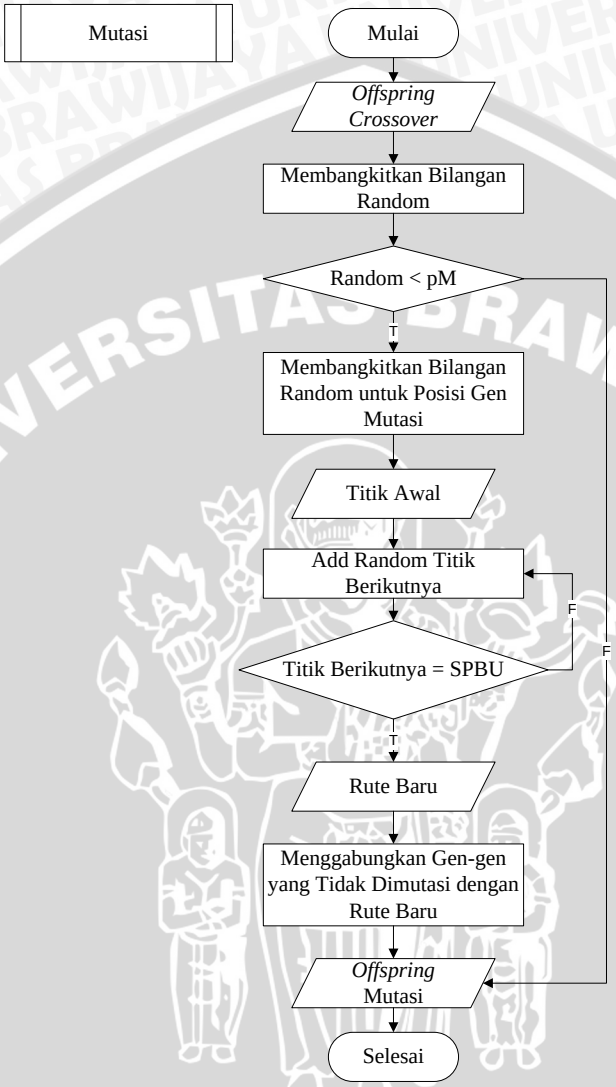


Gambar 4.8 Diagram Alir *Crossover*

4.3.5 Mutasi

Mutasi dilakukan untuk menghindari *local optimum*. *Local optimum* merupakan solusi terbaik yang dihasilkan pada suatu set solusi dan tidak bersifat global atau masih ada kemungkinan solusi yang lebih baik [KAR-11]. Mutasi dipengaruhi oleh probabilitas mutasi (pM). Bilangan *random* dibangkitkan untuk masing-masing kromosom hasil *crossover*. Jika bilangan *random* yang dibangkitkan < pM maka akan terjadi mutasi pada kromosom. Sedangkan jika bilangan *random* yang dibangkitkan > pM maka tidak terjadi mutasi [MAR-13].

Mutasi dilakukan dengan langkah-langkah yang digambarkan oleh diagram alir Gambar 4.9.



Gambar 4.9 Diagram Alir Mutasi

Penjelasan dari diagram alir mutasi Gambar 4.9 yaitu:

1. Pertama dibangkitkan terlebih dahulu sebuah bilangan *random* untuk menentukan suatu *offspring* mengalami mutasi atau tidak. Jika bilangan *random* yang dibangkitkan < pM maka akan dilakukan mutasi.
2. Setelah itu, membangkitkan bilangan *random* untuk menentukan posisi gen yang akan dimutasi.

3. Proses mutasi dilakukan dengan membangkitkan rute baru dengan titik awal yaitu posisi gen yang dimutasi, proses membangkitkan rute baru sama dengan proses dalam menentukan kromosom awal.
4. Jika posisi gen berada ditengah kromosom, maka kromosom baru yang dihasilkan adalah gen sebelum titik mutasi ditambah dengan rute baru yang dihasilkan dari langkah ketiga.

Contoh:

2	26	31	24	28

Gen yang menjadi titik mutasi adalah gen dengan warna abu-abu yaitu gen dengan node 31, rute baru yang dihasilkan dengan titik awal 31 yaitu:

31	27	24	28

Sehingga kromosom hasil (*offspring*) mutasi yaitu:

2	26	31	27	24	28

4.3.6 Populasi Baru

Setelah semua proses evolusi dilakukan maka proses selanjutnya adalah proses evaluasi untuk menentukan populasi baru yang akan mengalami proses evolusi berikutnya. Sebelumnya terlebih dahulu dilakukan proses perbaikan kromosom pada *offspring* hasil mutasi untuk memotong gen yang berulang.

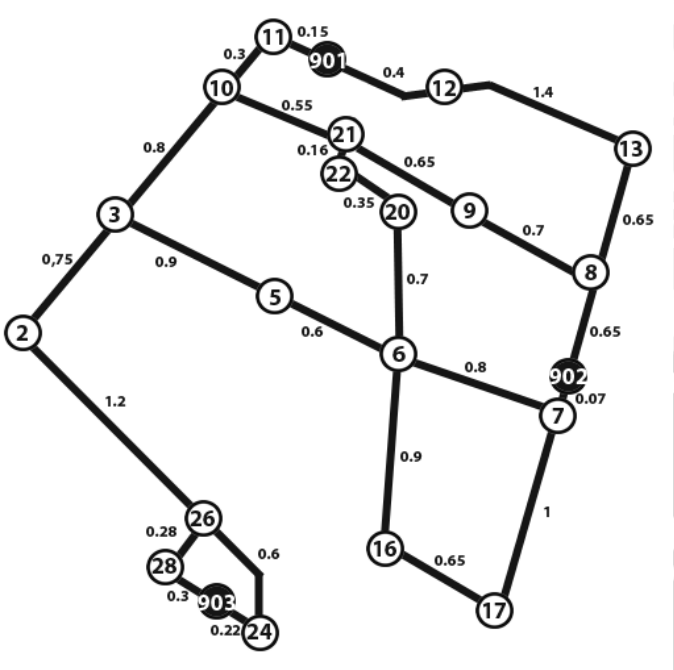
Untuk menentukan populasi baru, hasil *offspring* digabungkan dengan populasi awal atau populasi sebelumnya dan diurutkan berdasarkan nilai *fitness* tertinggi. Urutan lima teratas akan dijadikan populasi baru untuk menjalani proses evolusi selanjutnya [AFA-11]. Diagram alir populasi baru ditunjukkan Gambar 4.10.

Gambar 4.10 Diagram Alir Populasi Baru

Hasil dari Algoritma Genetik ketika telah mencapai jumlah iterasi yang telah ditentukan berupa sebuah rute dengan nilai *fitness* terbesar. Rute tersebut terdiri dari gen-gen yang berupa node dari kolom id di tabel data_koordinat. Untuk menampilkan rute tersebut dilakukan dengan cara menggambar garis dari satu node ke node yang lain hingga node terakhir. Penggambaran ini sesuai dengan titik latitude dan latitude yang juga telah tersimpan dalam data_koordinat. Aplikasi akan memilih titik latitude dan latitude sesuai dari id atau node dari rute hasil untuk menggambarkannya pada *Google Maps*. Rute hasil tersebut juga akan disimpan pada tabel data_hasil, sehingga aplikasi tidak perlu melakukan perhitungan kembali ketika set posisi dilakukan pada titik atau node yang sama.

4.4 Perhitungan Manual

Pada bagian ini akan dijelaskan mengenai perhitungan manual dari Agoritma Genetika dengan menggunakan data sampel. Data sampel ini diperoleh dari sebagian data yang nantinya akan dijadikan sebagai data pada aplikasi yang dibuat.



Gambar 4.11 Path Jalan

Gambar 4.11 merupakan peta dari suatu daerah di Kota Malang yang telah dijadikan path jalan. Setiap cabang jalan dan lokasi SPBU dijadikan sebuah titik. Dibuat sedemikian rupa karena jika titik dibuat berdasarkan satu nama jalan dapat menyebabkan seluruh panjang jalan akan ikut dihitung dalam algoritma. Sehingga panjang jarak tempuh yang dihasilkan dapat menjadi lebih panjang dari pada jarak sesungguhnya. Hal tersebut dapat mempengaruhi hasil dari perhitungan.

Nomor berwarna putih dengan gambar bulat berwarna hitam menandakan SPBU. Berikut akan dijelaskan perhitungan manual sesuai perancangan Algoritma Genetik dengan menggunakan data sesuai Gambar 4.11.

4.4.1 Membangkitkan Kromosom (Populasi Awal)

Populasi awal terdiri dari 5 kromosom. Dimisalkan posisi berada di dekat titik nomor 2, maka titik 2 menjadi gen pertama pada setiap kromosom. Kromosom-kromosom yang diperoleh berdasarkan data Gambar 4.11 yaitu sebagai berikut.

Kromosom 1	2	3	5	6	7	902
Kromosom 2	2	26	28	903		
Kromosom 3	2	3	10	11	901	
Kromosom 4	2	3	10	21	22	20	6	7	902
Kromosom 5	2	3	10	21	9	8	902		

4.4.2 Perhitungan Nilai *Fitness*

Nilai *fitness* yang diperoleh dari masing-masing kromosom yang dihasilkan yaitu:

Fitness kromosom 1 : 1/(0.75 + 0.9 + 0.6 + 0.8 + 0.07) = 0.32

Fitness kromosom 2 : 1/(1.2 + 0.28 + 0.3) = 0.56

Fitness kromosom 3 : 1/(0.75 + 0.8 + 0.3 + 0.15) = 0.5

Fitness kromosom 4 : 1/(0.75 + 0.8 + 0.55 + 0.16 + 0.35 + 0.7 + 0.8 + 0.07)= 0.24

Fitness kromosom 5 : 1/(0.75 + 0.8 + 0.55 + 0.65 + 0.7 + 0.65) = 0.24

4.4.3 Seleksi dengan *Roulette Wheel*

Untuk melakukan seleksi dengan menggunaka *Roulette Wheel* terlebih dahulu dihitung probabilitas dari setiap kromosom. Probabilitas kromosom dapat diperoleh dari *fitness* setiap kromosom dibagi dengan total *fitness*.

Total *fitness* : 0.32 + 0.56 + 0.5 + 0.24 + 0.24 = 1.86

Probabilitas kromosom 1 : 0.32/1. 86 = 0.17

Probabilitas kromosom 2 : 0.56/1. 86 = 0.3

Probabilitas kromosom 3 : 0.5/1. 86 = 0.27

Probabilitas kromosom 4 : 0.24/1. 86 = 0.13

Probabilitas kromosom 5 : 0.24/1. 86 = 0.13

Dari probabilitas di atas maka dapat diperoleh range seperti di bawah ini:

Range 1 : 0 - 0.17
Range 2 : 0.17 - 0.47
Range 3 : 0.47 - 0.74
Range 4 : 0.74 - 0.87
Range 5 : 0.87 - 1

Dimisalkan bilangan *random* yang diperoleh yaitu 0.75 dan 0.89, maka kromosom yang menjadi kromosom induk adalah kromosom 4 dan kromosom 5. Kromosom induk akan mengalami proses selanjutnya yaitu *crossover*.

4.4.4 *Crossover*

Kromosom induk yang akan mengalami proses *crossover* yaitu:

Kromosom 4	2	3	10	21	22	20	6	7	902
Kromosom 5	2	3	10	21	9	8	902		

Crossover terjadi jika pada kromosom terdapat gen yang sama kecuali gen pertama karena gen tersebut merupakan posisi awal serta bukan gen terakhir karena gen tersebut merupakan gen tujuan. Jika melakukan *crossover* pada titik awal dan titik tujuan maka kemungkinan besar *crossover* yang dilakukan tidak berpengaruh pada perubahan susunan gen pada kromosom induk.

Crossover dipengaruhi oleh pC atau probabilitas *crossover*. Dimisalkan pC=0.7 sedangkan bilangan *random* yang muncul adalah 0.76. Maka terjadi *crossover*. Gen yang sama pada kromosom induk yaitu gen dengan warna abu-abu pada gambar kromosom, gen tersebut yaitu 21. Jika ada lebih dari satu gen yang sama, maka gen terakhir yang sama yang dipilih. *Crossover* dilakukan pada gen dengan nomor 21 hingga gen terakhir (gen dengan warna abu-abu). Maka *offspring* atau kromosom anak yang dihasilkan yaitu:

Offspring 1	2	3	10	21	9	8	902		
Offspring 2	2	3	10	21	22	20	6	7	902

Gen dengan warna abu-abu merupakan gen yang mengalami *crossover*.

4.4.5 Mutasi

Mutasi dilakukan pada *offspring* hasil dari *crossover*. Mutasi dipengaruhi oleh probabilitas mutasi (pM). Dimisalkan pM=0.6, sedangkan bilangan *random* yang dihasilkan yaitu 0.5 dan 0.7. Maka tidak terjadi mutasi pada *offspring* 1 dan terjadi mutasi pada *offspring* 2.

Mutasi dilakukan dengan cara membangkitkan rute baru dengan titik awal adalah gen yang mengalami mutasi. Posisi gen yang mengalami mutasi merupakan posisi yang diperoleh secara *random*. Dimisalkan posisi gen yang mengalami mutasi pada *offspring* 2 adalah posisi ke-3 yaitu posisi gen dengan warna abu-abu.

Offspring 1	2	3	10	21	9	8	902		
Offspring 2	2	3	10	21	22	20	6	7	902

Hasil dari mutasi yaitu sebagai berikut:

Offspring 1	2	3	10	21	9	8	902
Offspring 2	2	3	10	21	9	8	902

Gen dengan warna abu-abu pada *Offspring* 2 merupakan rute baru yang dihasilkan dari proses mutasi.

4.4.6 Populasi Baru

Untuk menentukan populasi baru, hasil *offspring* digabungkan dengan populasi awal atau populasi sebelumnya dan diurutkan berdasarkan nilai *fitness* tertinggi. Urutan lima teratas akan dijadikan populasi baru untuk menjalani proses evolusi selanjutnya. Berikut merupakan lima kromosom dengan *fitness* tertinggi yang akan mengalami proses evolusi selanjutnya.

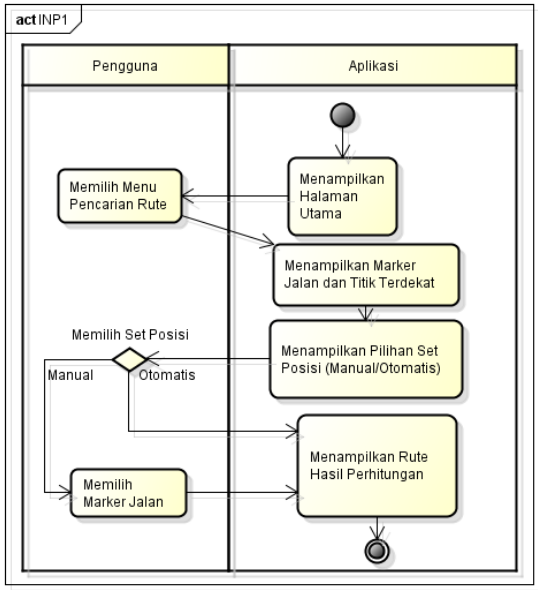
Kromosom 2	2	26	28	903			*Fitness*=0.56
Kromosom 3	2	3	10	11	901		*Fitness*=0.5
Kromosom 1	2	3	5	6	7	902	*Fitness*=0.32
Kromosom 5	2	3	10	21	9	8	902 *Fitness*=0.24
Offspring 1	2	3	10	21	9	8	902 *Fitness*=0.24

4.5 Perancangan Perangkat Lunak

Setelah dilakukan analisa kebutuhan, selanjutnya kebutuhan-kebutuhan tersebut dibentuk dalam suatu perancangan perangkat lunak yang terdiri dari diagram aktifitas (*activity diagram*), diagram sekuen (*sequence diagram*), diagram kelas (*class diagram*), perancangan basis data dan perancangan antarmuka.

4.5.1 *Activity Diagram*

Aktifitas antara pengguna dengan sistem yang berjalan dimodelkan menggunakan diagram aktifitas atau *activity diagram* sesuai dengan skenario *use case*. *Activity Diagram* pencarian rute terpendek yang diperoleh dari perhitungan menggunakan Algoritma Genetik ditunjukkan pada Gambar 4.12.

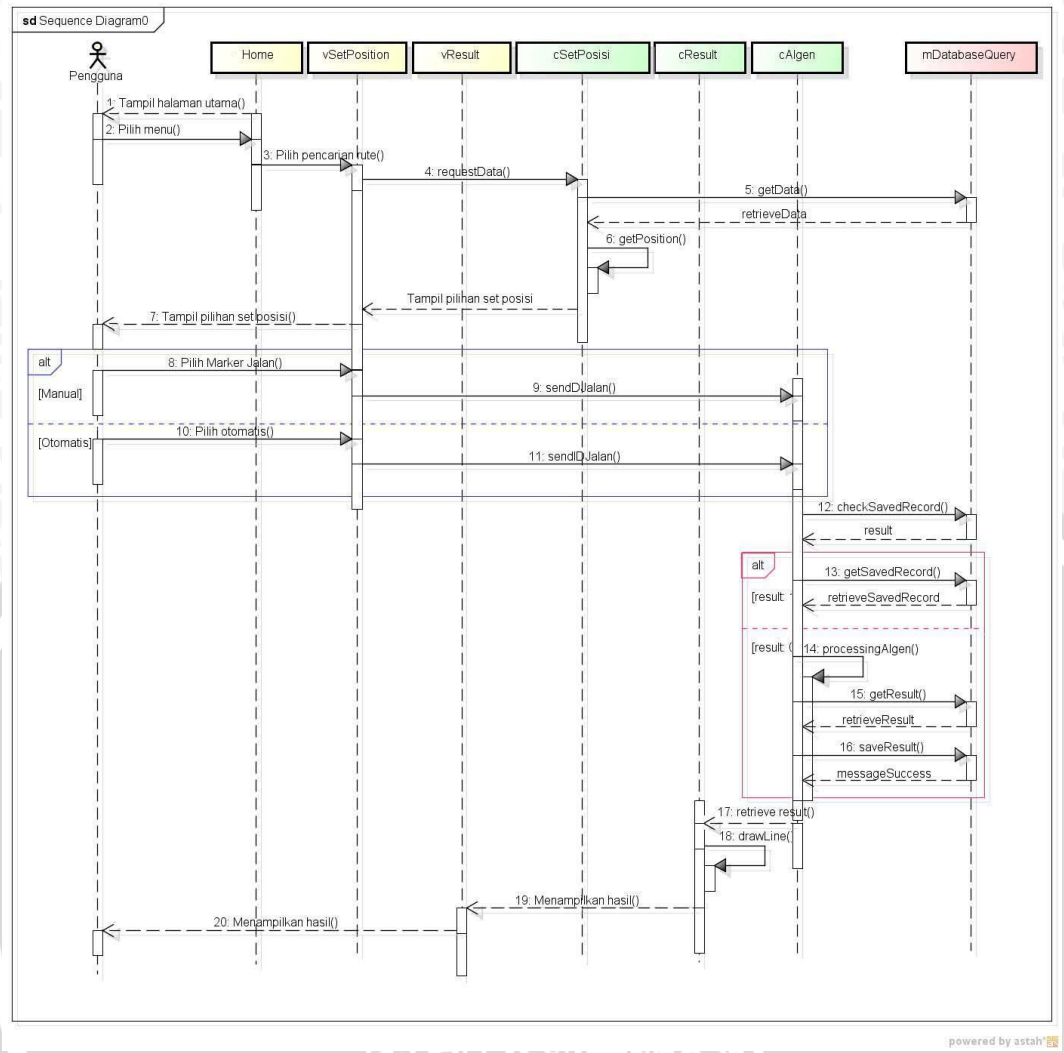


Gambar 4.12 *Activity Diagram* Aplikasi

Activity diagram Gambar 4.12 menggambarkan aktifitas antara pengguna dan aplikasi untuk memperoleh hasil berupa rute terpendek yang diperoleh dari perhitungan menggunakan Algoritma Genetik sesuai dengan skenario *use case* pada Tabel 4.2, dan Tabel 4.3. Skenario *use case* pada Tabel 4.2 digambarkan mulai *initial node* pada halaman utama aplikasi hingga set posisi yaitu memilih set posisi secara manual atau otomatis pada *activity diagram* Gambar 4.12. Sedangkan Skenario *use case* pada Tabel 4.3 digambarkan mulai set posisi hingga akhir (*activity final*) dari *activity diagram* Gambar 4.12.

4.5.2 *Sequence Diagram*

Interaksi antara objek satu dengan yang lainnya digambarkan melalui *sequence diagram. Sequence diagram* digambarkan sesuai dengan urutan waktu dari pesan yang digunakan pada saat terjadi interaksi. *Sequence diagram* pencarian rute dengan perhitungan menggunakan Algoritma Genetik ditunjukkan pada Gambar 4.13.

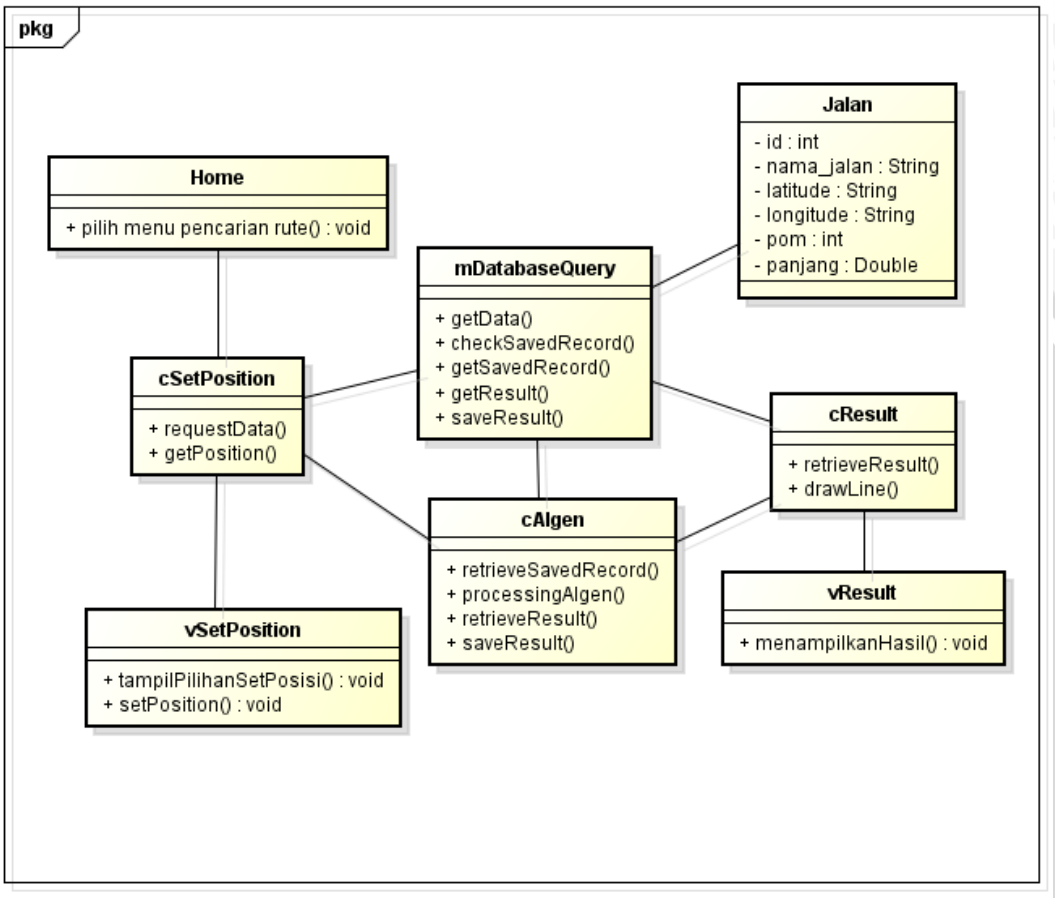


Gambar 4.13 *Sequence Diagram* Pencarian Rute

Pencarian rute terpendek dipicu oleh set posisi. Set posisi secara manual dilakukan dengan memilih salah satu marker jalan yang ditampilkan pada *Google Maps.* Sedangkan set posisi secara otomatis dilakukan oleh sistem dengan mendeteksi posisi atau titik terdekat dari posisi pengguna. Marker jalan yang dipilih oleh pengguna maupun posisi yang dideteksi oleh sistem akan menjadi kunci dari perhitungan dengan Algoritma Genetik.

4.5.3 *Class Diagram*

Class diagram merupakan diagram yang menggambarkan kelas-kelas yang ada pada suatu sistem dan hubungan antara satu kelas dengan kelas yang lainnya. *Class diagram* aplikasi ditunjukkan oleh Gambar 4.14.

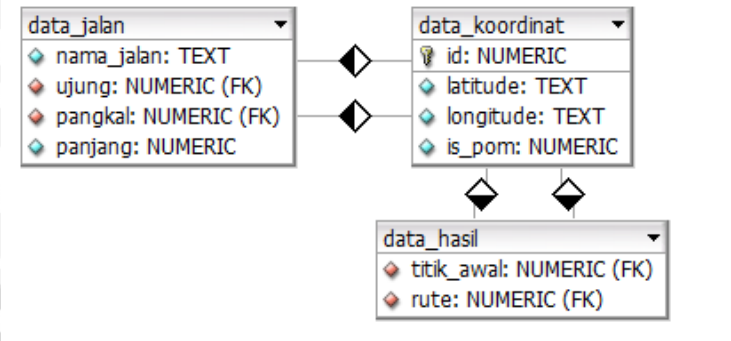


Gambar 4.14 *Class Diagram* Aplikasi

4.5.4 Perancangan Basis Data

Basis data berfungsi untuk menyimpan informasi atau data-data yang diperlukan oleh sistem. Perancangan basis data digambarkan dalam bentuk ERD (*Entity Relationship Diagram*). Sistem ini menggunakan tiga tabel yaitu untuk menyimpan data titik setiap percabangan jalan dan setiap SPBU, menyimpan data jalan yang ada di Kota Malang serta tabel untuk menyimpan rute hasil

perhitungan. Gambar 4.15 menunjukkan perancangan basis data yang akan digunakan untuk menyimpan data-data yang diperlukan.



Gambar 4.15 Perancangan Basis Data

Struktur table serta keterangan dari masing-masing field ditunjukkan pada Tabel 4.4, Tabel 4.5 dan Tabel 4.6.

Tabel 4.4 Struktur Tabel Data_jalan

No.	Nama Field	Tipe Data	Keterangan	Contoh Data
1.	nama_jalan	Text	Nama jalan.	Bendungan Sutami
2.	pangkal	Numeric	Titik pangkal jalan.	19
3.	ujung	Numeric	Titik ujung jalan.	29
4.	panjang	Text	Panjang jalan.	1.45

Data jalan disimpan dengan menggunakan titik pangkal dan titik ujung untuk membentuk suatu *path* jalan jika tiap titik dihubungkan. Titik-titik pada tabel data_jalan merupakan titik yang tersimpan pada tabel data_koordinat yang memiliki struktur seperti pada Tabel 4.5. Koordinat setiap jalan dibutuhkan untuk membuat marker atau tanda pada *Google Maps*.

Tabel 4.5 Struktur Tabel Data_koordinat

No.	Nama Field	Tipe Data	Keterangan	Contoh Data
1.	id	Numeric	Id jalan.	19
2.	latitude	Text	Koordinat latitude jalan.	-7.943153

3.	longitude	Text	Koordinat longitude jalan.	112.610319
4.	is_pom	Numeric	Berisi 0 untuk titik yang bukan SPBU dan 1 untuk titik yang merupakan SPBU.	1

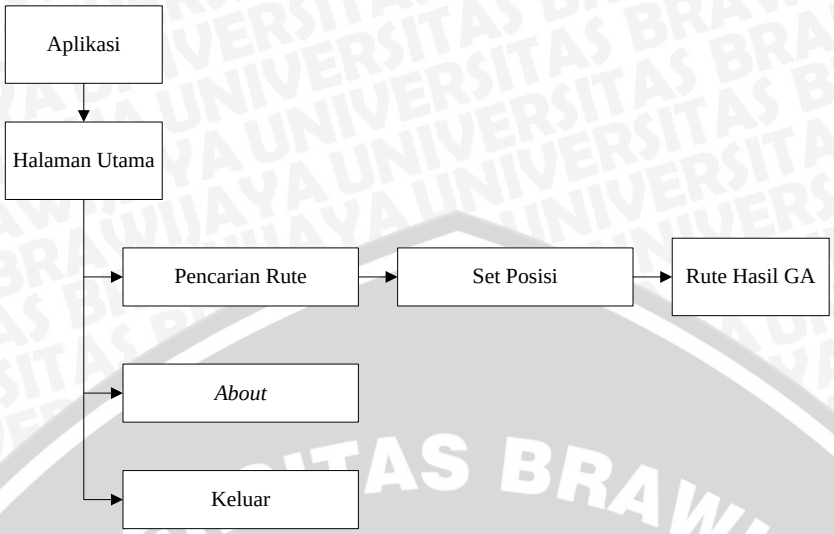
Tabel hasil digunakan untuk menyimpan rute hasil perhitungan. Jika pengguna menginputkan posisi pada jalan yang sama dengan input yang pernah dilakukan sebelumnya maka sistem tidak perlu melakukan perhitungan dengan Algoritma Genetik kembali karena rute telah tersimpan pada tabel hasil dan sistem hanya menampilkan kembali rute yang telah tersimpan. Struktur tabel hasil ditunjukkan oleh Tabel 4.6.

Tabel 4.6 Struktur Tabel Data_hasil

No.	Nama Field	Tipe Data	Keterangan	Contoh Data
1.	titik_awal	Numeric	Id jalan yang merupakan titik awal pengguna.	19
2.	rute	Numeric	Berisi id jalan yang merupakan rute ke SPBU.	124

4.5.5 Perancangan Antarmuka

Perancangan antarmuka berisi rancangan tampilan aplikasi yang dibuat. Perancangan ini berfungsi agar aplikasi yang dibuat mudah digunakan oleh pengguna. Untuk merepresentasikan keseluruhan antarmuka sistem digunakan *Site Map. Site Map* aplikasi ditunjukkan pada gambar 4.16.



Gambar 4.16 *Site Map* Aplikasi

4.5.5.1 Desain Antarmuka Halaman Utama Aplikasi

Halaman utama merupakan halaman pertama yang muncul ketika aplikasi dibuka. Pada halaman utama terdapat tiga menu yaitu menu untuk pencarian rute terpendek, menu penjelasan aplikasi dan petunjuk aplikasi serta tombol keluar aplikasi. Rancangan halaman utama aplikasi ditunjukkan Gambar 4.17.

Gambar 4.17 Desain Antarmuka Halaman Utama Aplikasi

Untuk melakukan pencarian rute pengguna memilih menu pencarian rute yaitu nomor 1 seperti yang ditunjukkan Gambar 4.17. Keterangan Gambar 4.17 ditunjukkan pada Tabel 4.7.

Tabel 4.7 Keterangan Desain Antarmuka Halaman Utama Aplikasi

Nama	Keterangan
Pencarian Rute	Menu ini merupakan menu pencarian Rute. Saat pengguna memilih menu ini maka akan masuk ke halaman input posisi pengguna. Input tersebut akan dijadikan sebagai acuan untuk perhitungan dengan Algoritma Genetik dan hasilnya akan kembali ditampilkan oleh aplikasi.
Tentang Aplikasi	Menu untuk masuk ke halaman *about* dan halaman petunjuk penggunaan.
Keluar	Menu untuk keluar aplikasi.

4.5.5.2 Desain Antarmuka Halaman Set Posisi

Halaman set menu merupakan halaman setelah menu pertama (menu pencarian rute) pada halaman utama dipilih oleh pengguna. Rancangan antarmuka halaman set posisi ditunjukkan Gambar 4.18.

Gambar 4.18 Desain Antarmuka Halaman Set Posisi

Pada halaman ini akan muncul sebuah dialog yang berisi dua pilihan yaitu pengguna ingin melakukan set posisi secara manual atau memilih untuk langsung melanjutkan ke halaman selanjutnya yaitu halaman hasil karena sistem dapat

mendeteksi titik terdekat dari posisi pengguna secara otomatis. Jika posisi yang terdeteksi tidak sesuai dengan posisi pengguna, pengguna dapat memilih set posisi secara manual dengan memilih tanda atau marker jalan. Keterangan Gambar 4.18 ditunjukkan pada Tabel 4.8.

Tabel 4.8 Keterangan Desain Antarmuka Halaman Set Posisi

Nama	Keterangan
Penjelasan	Menampilkan penjelasan singkat penggunaan aplikasi.
Maps	Menampilkan *Google Maps* beserta marker jalan untuk melakukan set posisi.

4.5.5.3 Desain Antarmuka Halaman Hasil

Halaman hasil berfungsi untuk menampilkan hasil dari perhitungan dengan menggunakan Algoritma Genetik yang berupa rute terpendek menuju SPBU. Halaman ini muncul setelah pengguna melakukan set posisi pada halaman set posisi. Rancangan halaman hasil ditunjukkan Gamar 4.19.

Gambar 4.19 Desain Antarmuka Halaman Hasil

Rute hasil ditampilkan oleh aplikasi dalam bentuk rute yang digambarkan pada *Google Maps*. Jalan-jalan yang harus dilalui oleh pengguna untuk sampai ke

SPBU ditandai dengan memberi warna pada jalan mulai dari posisi pengguna hingga SPBU yang dituju. Keterangan Gambar 4.19 ditunjukkan pada Tabel 4.9.

Tabel 4.9 Keterangan Desain Antarmuka Halaman Hasil

Nama	Keterangan
Penjelasan	Menampilkan penjelasan singkat penggunaan aplikasi.
Maps	Menampilkan rute hasil pada *Google Maps*.

BAB V
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dibahas hasil implementasi berdasarkan perancangan yang telah dibahas pada bab sebelumnya.

5.1 Lingkungan Implementasi

Lingkungan implementasi yang dijelaskan pada sub bab ini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak yang digunakan untuk pengembangan maupun untuk mengimplementasi aplikasi pencarian rute terpendek untuk menemukan SPBU terdekat di Kota Malang.

5.1.1 Lingkungan Perangkat Keras

Lingkungan perangkat keras yang digunakan untuk pengembangan, instalasi aplikasi dan pengujian terdiri dari dua, yaitu komputer/laptop untuk pengembangan aplikasi dan perangkat bergerak (*smartphone*) untuk instalasi dan pengujian aplikasi. Perangkat keras yang digunakan untuk pengembangan aplikasi ditunjukkan pada Tabel 5.1. Sedangkan lingkungan perangkat keras dalam melakukan instalasi dan pengujian ditunjukkan pada Tabel 5.2.

Tabel 5.1 Lingkungan Perangkat Keras Pengembangan Aplikasi

Nama Komponen	Spesifikasi
System Model	Toshiba Satellite L635
Processor	Intel(R) Core(TM) i5 CPU M450 @2.40GHz
Memory (RAM)	4.00 GB
Hardisk	500 GB
Display	ATI Mobility Radeon

Tabel 5.2 Lingkungan Perangkat Keras Instalasi dan Pengujian Aplikasi

Nama Komponen	Spesifikasi
System Model	Lenovo IdeaTab A3000-H
Processor	Quad-core 1.2 GHz Cortex-A7
Memory	16 GB 1 GB RAM
Display	IPS LCD capacitive touchscreen, 16M colors 600x1024 pixels, 7.0 inches.

5.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan untuk pengembangan aplikasi ditunjukkan pada Tabel 5.3. Sedangkan lingkungan perangkat lunak dalam melakukan instalasi dan pengujian ditunjukkan pada Tabel 5.4.

Tabel 5.3 Lingkungan Perangkat Lunak Pengembangan Aplikasi

Nama Komponen	Spesifikasi
Operating System	Windows 7 Home Premium 32-bit *Operating System*
IDE (*Integrated Development Environment*)	Eclipse Juno dengan ADT (*Android Development Tools*) plug in.
Perangkat lunak lain	Adobe Photoshop CS5 SQLite Database Browser

Tabel 5.4 Lingkungan Perangkat Lunak Instalasi dan Pengujian Aplikasi

Nama Komponen	Spesifikasi
Platform	Android 4.2.2 (Jelly Bean)

5.2 Batasan Implementasi

Dalam implementasi aplikasi pencarian rute terpendek menuju SPBU terdekat terdapat beberapa batasan. Batasan tersebut antara lain:

1. Implementasi *Google Map* untuk melakukan set posisi maupun menampilkan rute hasil menggunakan *Google Map* versi 2.
2. Untuk menampilkan rute pada *Google Map* diperlukan koneksi internet sehingga kecepatan akses untuk menampilkan rute hasil bergantung pada kecepatan koneksi internet pada masisng-masing perangkat yang digunakan.
3. Aplikasi hanya menampilkan satu rute yang merupakan rute terbaik yang dihasilkan oleh perhitungan Algoritma Genetik.
4. Set posisi secara otomatis dilakukan dengan mendeteksi marker terdekat dari posisi pengguna yang ditunjukan oleh GPS.
5. Penggambaran rute pada *Google Maps* dilakukan dengan menarik garis antara satu titik ke titik yang lain sesuai hasil Algoritma dan bukan penarikan garis dari titik awal langsung menuju titik tujuan.

5.3 Implementasi Basis Data

Aplikasi pencarian rute terpendek menggunakan basis data untuk menyimpan data-data yang diperlukan. Data-data tersebut antara lain data jalan, data koordinat dan data hasil. Implementasi basis data sesuai dengan perancangan basis data pada bab perancangan. Tabel 5.5 merupakan implementasi dari Tabel 4.4, Tabel 5.6 merupakan implementasi Tabel 4.5 dan Tabel 5.7 merupakan implementasi dari Tabel 4.6.

Tabel 5.5 Tabel Data Jalan

No.	Kolom	Tipe Data	Deskripsi	Contoh Data
1.	nama_jalan	Text	Nama jalan.	Bendungan Sutami
2.	pangkal	Numeric	Titik pangkal jalan.	19
3.	ujung	Numeric	Titik ujung jalan.	29
4.	panjang	Text	Panjang jalan.	1.45

Tabel 5.5 menjelaskan mengenai kolom-kolom dari tabel yang diimplementasikan. Contoh data merupakan contoh dari isi pada kolom tersebut.

Tabel 5.6 Tabel Data Koordinat

No.	Kolom	Tipe Data	Deskripsi	Contoh Data
1.	id	Numeric	Id jalan.	19
2.	latitude	Text	Koordinat latitude jalan.	-7.943153
3.	longitude	Text	Koordinat longitude jalan.	112.610319
4.	is_pom	Numeric	Berisi 0 untuk titik yang bukan SPBU dan 1 untuk titik yang merupakan SPBU.	1

Tabel 5.6 menjelaskan mengenai kolom-kolom dari tabel yang diimplementasikan. Contoh data merupakan contoh dari isi pada kolom tersebut.

Tabel 5.7 Tabel Data Hasil

No.	Kolom	Tipe Data	Deskripsi	Contoh Data
1.	titik_awal	Numeric	Id jalan yang merupakan titik awal pengguna.	19
2.	rute	Numeric	Berisi id jalan yang merupakan rute ke SPBU.	124

Tabel 5.7 menjelaskan mengenai kolom-kolom dari tabel yang diimplementasikan. Contoh data merupakan contoh dari isi pada kolom tersebut.

Berikut ini akan dipaparkan data riil dari masing-masing tabel. Data riil pada bab ini hanya ditampilkan 10 data pada data jalan dan data koordinat. Untuk seluruh data yang digunakan aplikasi dilampirkan pada halaman lampiran.

Data riil dari Tabel 5.5 ditunjukkan pada Tabel 5.8. Tabel 5.8 berisi nama jalan, node awal dan node akhir dari jalan serta panjang jalan.

Tabel 5.8 Data Riil Tabel Data Jalan

nama_jalan	pangkal	ujung	panjang
Mayjend MT Haryono	27	1	0.65
Mayjend MT Haryono	1	2	0.95
Mayjen Panjaitan	2	26	1.2
Mayjen Panjaitan	26	24	0.6
Brigjend Slamet Riadi	24	23	1.5
Soekarno Hatta	2	3	0.75
Soekarno Hatta	3	2	0.75
Soekarno Hatta	3	10	0.8
Soekarno Hatta	10	3	0.8

Data riil dari Tabel 5.6 ditunjukkan pada Tabel 5.9. Tabel 5.9 merupakan tabel yang menyimpan data node yang berupa id, titik longitude dan latitude serta is_pom yang menyatakan node tersebut merupakan SPBU atau tidak. is_pom bernilai 0 ketika node bukan merupakan SPBU dan bernial 1 ketika node tersebut merupakan SPBU.

Tabel 5.9 Data Riil Tabel Data Koordinat

id	latitude	longitude	is_pom
1	-7.94315	112.6103	0
2	-7.94983	112.6154	0
3	-7.94456	112.6194	0
5	-7.94783	112.6252	0
6	-7.95085	112.6316	0
7	-7.95293	112.6388	0
8	-7.94732	112.64	0
9	-7.94434	112.6345	0
10	-7.93946	112.6248	0

Data riil dari Tabel 5.7 ditunjukkan pada Tabel 5.10. Tabel 5.10 berisi node-node rute yang dihasilkan oleh perhitungan dengan Algoritma Genetik. Tabel data hasil berisi node awal serta node-node rute menuju SPBU. Tabel ini akan terisi setelah aplikasi dijalankan, sehingga pada awal penginstalan aplikasi tabel ini berada dalam keadaan kosong.

Tabel 5.10 Data Riil Tabel Data Hasil

titik_awal	rute
3	3
3	10
3	11
3	901

5.4 Implementasi *Class*

Class-class yang telah dirancang pada perancangan Gambar 4.14 direalisasikan pada *file-file* dengan ekstensi .java dan tampilan dari *class-class* tersebut direalisasikan pada *file-file* dengan ekstensi .xml. Implementasi Gambar 4.14 ditunjukkan pada Tabel 5.11.

Tabel 5.11 Implementasi Perancangan *Class*

No	Pakage	Objek	Nama *Class*	Nama File Program	Nama File Layout
1.	com.vire.spbu	*View* dan *controller*	SetPosition	SetPosition .java	set_position .xml
2.	com.vire.spbu	*View* dan *controller*	Result	ResultRoute .java	result_route .xml
3.	com.vire.spbu	*Controller*	Algen	Algoritma Genetik.java	-
4.	com.vire.database	Model	Database Query	DatabaseQuery .java	-

5.5 Implementasi Kode Program

Pada implementasi kode program akan dijelaskan hasil implementasi aplikasi sesuai dari perancangan yang telah dijelaskan pada bab sebelumnya. Aplikasi pencarian rute terpendek dengan menggunakan Algoritma Genetik terdiri dari beberapa fungsi atau *method*. Pada penulisan skripsi ini akan dijelaskan beberapa *method* utama yang membangun aplikasi, sehingga tidak semua *method* dijelaskan pada bagian ini. *Method-method* tersebut antara lain beberapa *method*

pada implementasi Algoritma Genetik dan beberapa *method* yang berfungsi untuk menampilkan hasil algoritma maupun *method* untuk melakukan set posisi.

5.5.1 Kode Program Set Posisi

Bagian ini menjelaskan potongan kode program dari implementasi menampilkan marker jalan yang digunakan untuk melakukan set posisi. Set posisi terdiri dari beberapa *method* sesuai *class* diagram Gambar 4.14. *Method-method* tersebut antara lain requestData(), getPosition(), tampilPilihanSetPosisi(), dan setPosisi(). *Method-method* disajikan dalam bentuk potongan kode program yaitu pada Gambar 5.1, Gambar 5.2, Gambar 5.3 dan Gambar 5.4.

```
1   private List<Jalan> getData(){
2     DatabaseQuery query = new DatabaseQuery(this);
3     query.createDatabase();
4     query.open();
5     Cursor curData = query.queryGetKoordinatAll();
6
7     List<Jalan> list = new ArrayList<Jalan>();
8     Jalan jl = null;
9     if (curData .moveToFirst()) {
10      while (curData.isAfterLast() == false) {
11        int id = curData.getInt(curData.getColumnIndex("id"));
12        String lat = curData.getString(curData
13                     .getColumnIndex("koordinat_x"));
14        String lang = curData.getString(curData
15                      .getColumnIndex("koordinat_y"));
16        int pom = curData.getInt(curData.getColumnIndex("is_pom"));
17        jl = new Jalan();
18        jl.setId(id);
19        jl.setLat(lat);
20        jl.setLang(lang);
21        jl.setPom(pom);
22        list.add(jl);
23        curData.moveToNext();
24      }
25    }
26    query.close();
27    return list;
28  }
```

Gambar 5.1 Implementasi *Method* requestData()

Gambar 5.1 adalah *method* yang digunakan untuk memperoleh data titik-titik koordinat yang digunakan untuk membuat marker jalan.

```
1   float[] distance = new float[1];
2   Location.distanceBetween(a,b, Double.parseDouble(jl.getLat())
3   ,Double.parseDouble(jl.getLang()), distance);
4
5   if(i==0) mindist=distance[0];
6   else if(mindist>distance[0]){
7     mindist=distance[0];
8     id=jl.getId();
9     lat=Double.parseDouble(jl.getLat());
10    lang=Double.parseDouble(jl.getLang());
```

```
11  }
12  i++;
```

Gambar 5.2 Implementasi *Method* getPosition()

Gambar 5.2 adalah *method* yang digunakan untuk memperoleh titik terdekat yang berfungsi sebagai titik awal perhitungan algoritma.

```
1  for(Jalan jl: list){
2      marker = map.addMarker(new MarkerOptions()
3                  .position(new LatLng(Double.parseDouble(jl.getLat())
4                  ,Double.parseDouble(jl.getLang())))
5                  .title("Set posisi!").icon(BitmapDescriptorFactory
6                  .fromResource(R.drawable.street)));
7      hmap.put(marker.getId(), jl.getId());
8  }
```

Gambar 5.3 Implementasi *Method* tampilPilihanSetPosisi()

Gambar 5.3 adalah *method* yang digunakan untuk menampilkan marker jalan yang berfungsi untuk melakukan set posisi secara manual yaitu dengan memilih marker terdekat.

```
1  map.setOnInfoWindowClickListener(new OnInfoWindowClickListener() {
2      public void onInfoWindowClick(Marker marker) {
3        Intent in = new Intent(getApplicationContext(),
4                    AlgoritmaGenetik.class);
5        in.putExtra("id", hmap.get(marker.getId()));
6        startActivity(in);
7      }
8  });
```

Gambar 5.4 Implementasi *Method* setPosisi()

Gambar 5.4 adalah *method* yang digunakan untuk melakukan set posisi yaitu dengan mengirimkan id marker yang dipilih untuk selanjutnya dilakukan perhitungan menggunakan algoritma.

5.5.2 Kode Program Menampilkan Rute

Bagian ini menjelaskan potongan kode program dari implementasi menampilkan rute hasil pada *Google Maps*. Menampilkan rute hasil (*result*) terdiri dari beberapa *method* sesuai *class* diagram Gambar 4.14. *Method-method* tersebut antara lain retrieveResult(), drawLine(), menampilkanHasil(). *Method-method* disajikan dalam bentuk potongan kode program yaitu pada Gambar 5.5, Gambar 5.6 dan Gambar 5.7.

```
1  Intent i = getIntent();
2  getPosition((ArrayList<Jalan>) i.getSerializableExtra("hasil"), a, b);
```

Gambar 5.5 Implementasi *Method* retrieveResult()

Gambar 5.5 adalah *method* yang digunakan untuk memperoleh hasil dari perhitungan menggunakan Algoritma Genetik.

```
ArrayList<LatLng> directionPoint = md.getDirection(doc);
PolylineOptions rectLine = new PolylineOptions()
                                .width(6).color(Color.BLUE);

for (int i = 0; i < directionPoint.size(); i++) {
  rectLine.add(directionPoint.get(i));
}
map_google.addPolyline(rectLine);
```

Gambar 5.6 Implementasi *Method* drawLine()

Gambar 5.6 adalah *method* yang digunakan untuk menggambar garis pada rute hasil pada *Google Maps*.

```
private void getPosition(ArrayList<Jalan> list, double a, double b){
  int count = list.size();
  LatLng MYPOSISI1 = new LatLng(a,b);
  LatLng MYPOSISI2 = new LatLng(Double.parseDouble(list.get(0).getLat()),
  Double.parseDouble(list.get(0).getLang()));

  CameraUpdate hasil=CameraUpdateFactory.newLatLng(MYPOSISI1);
  CameraUpdate zoom=CameraUpdateFactory.zoomTo(14.0f);
  map_google.addMarker(new MarkerOptions().position(MYPOSISI1)
  .title("Posisi Saya").snippet("").icon(BitmapDescriptorFactory
  .fromResource(R.drawable.current)));
  map_google.moveCamera(hasil);
  map_google.animateCamera(zoom,2000,null);

  getDirectionMap(MYPOSISI1, MYPOSISI2);

  for(int i=0;i<count;i++){
    LatLng POSISI = new LatLng(Double.parseDouble(list.get(i)
    .getLat()),Double.parseDouble(list.get(i).getLang()));
    if (i==0 && count>1){
      map_google.addMarker(new MarkerOptions().position(POSISI)
      .title("Start").snippet("Jl. "+list.get(i).getNamaJalan())
      .icon(BitmapDescriptorFactory.fromResource(R.drawable.start)));
    }
    else if (i==count-1){
      map_google.addMarker(new MarkerOptions().position(POSISI)
      .title("Finish").snippet("")
      .icon(BitmapDescriptorFactory.fromResource(R.drawable.finish)));
    }
    else {
      map_google.addMarker(new MarkerOptions().position(POSISI)
      .title("Jl. "+list.get(i).getNamaJalan()).snippet("")
      .icon(BitmapDescriptorFactory.fromResource(R.drawable.car1)));
    }

    if(i<count-1){
      LatLng POSISI2 = new LatLng(Double.parseDouble(list.get(i+1)
      .getLat()),Double.parseDouble(list.get(i+1).getLang()));
      getDirectionMap(POSISI, POSISI2);
    }
  }
}
```

Gambar 5.7 Implementasi *Method* menampilkanHasil()

Gambar 5.7 adalah *method* yang digunakan untuk menampilkan hasil dari perhitungan menggunakan Algoritma Genetik.

5.5.3 Kode Program Model

Bagian ini menjelaskan potongan kode program dari implementasi model *database query*. Model *database query* terdiri dari beberapa *method* sesuai *class* diagram Gambar 4.14. *Method-method* tersebut antara lain getData(), checkSavedRecord(), getSavedRecord(), getResult(), saveResult(). *Method-method* disajikan dalam bentuk potongan kode program yaitu pada Gambar 5.8, Gambar 5.9, Gambar 5.10, Gambar 5.11 dan Gambar 5.12.

```
1   public Cursor queryGetTableKoordinat(){
2     try{
3       String sql ="SELECT k.id as id, k.koordinat_x as koordinat_x,
4                    k.koordinat_y as koordinat_y, d.nama_jalan as nama_jalan
5                    from data_koordinat k join data_jalan d on k.id = d.id";
6       Cursor mCur = SQLiteDB.rawQuery(sql, null);
7       if (mCur!=null){
8         mCur.moveToNext();
9       }
10      return mCur;
11    }
12    catch (SQLException mSQLException){
13      Log.e(TAG, "getTestData >>"+ mSQLException.toString());
14      throw mSQLException;
15    }
16  }
```

Gambar 5.8 Implementasi *Method* getData()

Gambar 5.8 adalah *method* yang digunakan untuk memperoleh data jalan.

```
1   public Cursor queryCheck(int id){
2     try{
3       String sql ="select titik_awal from data_hasil where titik_awal ="+id;
4       Cursor mCur = SQLiteDB.rawQuery(sql, null);
5       if (mCur!=null){
6         mCur.moveToNext();
7       }
8       return mCur;
9     }
10    catch (SQLException mSQLException){
11      Log.e(TAG, "getTestData >>"+ mSQLException.toString());
12      throw mSQLException;
13    }
14  }
```

Gambar 5.9 Implementasi *Method* checkSavedRecord()

Gambar 5.9 adalah *method* yang digunakan untuk melakukan pengecekan pada record yang telah tersimpan.

```
1   public Cursor queryCheck(int id){
2     try{
3       String sql ="select titik_awal from data_hasil where titik_awal ="+id;
4       Cursor mCur = SQLiteDB.rawQuery(sql, null);
5       if (mCur!=null){
6         mCur.moveToNext();
7       }
8       return mCur;
9     }
10    catch (SQLException mSQLException){
11      Log.e(TAG, "getTestData >>"+ mSQLException.toString());
12      throw mSQLException;
13    }
14  }
```

Gambar 5.10 Implementasi *Method* getSavedRecord()

Gambar 5.10 adalah *method* yang digunakan untuk memperoleh data rute yang telah tersimpan.

```
1   public Cursor queryGetKoordinat(int id){
2     try{
3       String sql ="SELECT koordinat_x, koordinat_y from data_koordinat
4                    where id ="+id+" LIMIT 1";
5       Cursor mCur = SQLiteDB.rawQuery(sql, null);
6       if (mCur!=null){
7         mCur.moveToNext();
8       }
9       return mCur;
10    }
11    catch (SQLException mSQLException){
12      Log.e(TAG, "getTestData >>"+ mSQLException.toString());
13      throw mSQLException;
14    }
    }
```

Gambar 5.11 Implementasi *Method* getResult()

Gambar 5.11 adalah *method* yang digunakan untuk memperoleh data jalan sesuai id hasil yang diperoleh dari perhitungan.

```
1   public boolean SaveResult(int titik_awal, int rute){
2     try{
3       ContentValues cv = new ContentValues();
4       cv.put("titik_awal", titik_awal);
5       cv.put("rute", rute);
6       SQLiteDB.insert("data_hasil", null, cv);
7       return true;
8     }
9     catch(Exception ex){
10      return false;
11    }
12  }
```

Gambar 5.12 Implementasi *Method* saveResult()

Gambar 5.12 adalah *method* yang digunakan untuk meyimpan rute hasil perhitungan.

5.5.4 Kode Program Implementasi Algoritma Genetik

Bagian ini menjelaskan potongan kode program dari implementasi Algoritma Genetik. Algoritma Genetik terdiri dari beberapa *method* sesuai *class* diagram Gambar 4.14. *Method-method* tersebut antara lain retrieveSavedRecord(), processingAlgen(), retrieveResult(), saveResult(). *Method-method* disajikan dalam bentuk potongan kode program yaitu pada Gambar 5.13, Gambar 5.14, Gambar 5.15 dan Gambar 5.16.

```
1   private int [] getSavedRoute(int id){
2     route = new int [50];
3       query = new DatabaseQuery(this);
4       query.open();
5       cursor = query.queryGetSavedRoute(id);
6       if (cursor.moveToFirst()) {
7         int i=0;
8         while (cursor.isAfterLast() == false) {
9          int rute = cursor.getInt(cursor.getColumnIndex("rute"));
10         route[i]=rute;
11         cursor.moveToNext();
12         i++;
13        }
14      }
15      query.close();
16    return hasil1;
17  }
```

Gambar 5.13 Implementasi *Method* retrieveSavedRecord()

Gambar 5.13 adalah *method* yang digunakan untuk memperoleh data rute yang telah tersimpan.

```
1   route = Algen(getIntent().getIntExtra("id",1));
```

Gambar 5.14 Implementasi *Method* processingAlgen()

Gambar 5.14 adalah *method* yang digunakan untuk melakukan perhitungan menggunakan Algoritma Genetik.

```
1   list = new ArrayList<Jalan>();
2   jl = null;
3   for(int i=0;i<route.length;i++){
4     jl = new Jalan();
5     cursor = query.queryGetKoordinat(route[i]);
6     if (cursor.moveToFirst()) {
7       while (cursor.isAfterLast() == false) {
8         String lat = cursor.getString(curData
9                      .getColumnIndex("koordinat_x"));
10        String lang = cursor.getString(curData
11                     .getColumnIndex("koordinat_y"));
12        jl.setId(route[i]);
13        jl.setLat(lat);
14        jl.setLang(lang);
15        cursor.moveToNext();
16      }
17    }
```

```
18    if(i<route.length-1 && route.length>1){
19      cursor = query.queryGetNamaJalan(route[i], route[i+1]);
20      if (cursor.moveToFirst()) {
21        while (cursor.isAfterLast() == false) {
22          String nama = cursor.getString(cursor
23                        .getColumnIndex("nama_jalan"));
24          jl.setNamaJalan(nama);
25          cursor.moveToNext();
26        }
27      }
28    }
29    else if(i<route.length || route.length==1){
30      jl.setNamaJalan("");
31    }
32    list.add(jl);
33  }
```

Gambar 5.15 Implementasi *Method* retrieveResult()

Gambar 5.15 adalah *method* yang digunakan untuk memperoleh data jalan dan koordinat jalan sesuai id hasil yang diperoleh dari perhitungan.

```
1   public void saveResult(int id, int []route){
2     query = new DatabaseQuery(this);
3     query.open();
4     for(int i=0;i<route.size();i++) {
5       if(query.SaveResult(id, route[i]))
6           Log.d(TAG, " SAVED DATA");
7       }
8       else{
9           Log.d(TAG, "NOT SAVED DATA");
10      }
11    }
12    query.open();
13  }
```

Gambar 5.16 Implementasi *Method* saveResult()

Gambar 5.16 adalah *method* yang digunakan untuk menyimpan rute hasil perhitungan. *Method* processingAlgen() terdiri dari beberapa kode program sesuai perancangan Algoritma Genetik. Kode program tersebut antara lain potongan kode program membangkitkan kromosom, potongan kode program perhitungan nilai *fitness*, potongan kode program seleksi dengan menggunakan *Roulette Wheel*, potongan kode program *crossover* dan potongan kode program mutasi.

5.5.4.1 Kode Program Membangkitkan Kromosom

Implementasi algoritma membangkitkan kromosom ditunjukkan pada Gambar 5.17.

```
1   public int [] getKromosom(int id){
2     kromosom = new int[150];
3     kromosom[0]=id;
4     query = new DatabaseQuery(this);
5     query.open();
6
7     int pom=0, gen=0;
```

```
8     do{
9       if (gen==0){
10        cursor = query.queryGetRandomCabang(kromosom[gen]);
11        if (cursor.moveToFirst()) {
12          while (cursor.isAfterLast() == false) {
13            int cab = cursor.getInt(cursor.getColumnIndex("ujung"));
14            kromosom[gen+1]=cab;
15            cursor.moveToNext();
16          }
17        }
18      }
19      else {
20        cursor = query.queryGetRandomCabang2(kromosom[gen-1],kromosom[gen]);
21        if (cursor.moveToFirst()) {
22          while (cursor.isAfterLast() == false) {
23            int cab = cursor.getInt(cursor.getColumnIndex("ujung"));
24            kromosom[gen+1]=cab;
25            cursor.moveToNext();
26          }
27        }
28      }
29      cursor = query.queryGetCekPom(kromosom[gen+1]);
30      if (cursor.moveToFirst()) {
31        while (cursor.isAfterLast() == false) {
32          pom = cursor.getInt(cursor.getColumnIndex("is_pom"));
33          cursor.moveToNext();
34        }
35      }
36      gen++;
37    }
38    while ( pom==0);
39    query.close();
40    return hasil;
41  }
```

Gambar 5.17 Implementasi Membangkitkan Kromosom

Setiap kromosom terdiri dari gen-gen yang merupakan titik-titik yang dilalui hingga sampai ke SPBU. Gen-gen tersebut diperoleh secara *random* oleh sistem sesuai dari data yang disimpan pada basis data SQLite. Sistem akan berhenti melakukan pencarian ketika data yang diperoleh dari tabel data_koordinat yaitu pada kolom is_pom bernilai 1 yang artinya titik tersebut merupakan SPBU.

5.5.4.2 Kode Program Perhitungan Nilai *Fitness*

Implementasi algoritma perhitungan nilai *fitness* ditunjukkan pada Gambar 5.18.

```
1   public double getFitness(int [] kromosom1){
2      double hasil = 0;
3      int count = kromosom1.length;
4      query = new DatabaseQuery(this);
5      query.open();
6      for(int i=0;i<count-1;i++){
7         cursor = query.queryGetNamaJalan(kromosom1[i],kromosom1[i+1]);
8         if (cursor.moveToFirst()) {
9            while (cursor.isAfterLast() == false) {
10              double panjang = Double.parseDouble(cursor.getString
11              (cursor.getColumnIndex("panjang")));
```

```
12                  hasil=hasil+panjang;
13                  cursor.moveToNext();
14              }
15          }
16      }
17      query.close();
18      hasil=1/hasil;
19      return hasil;
20  }
```

Gambar 5.18 Implementasi Perhitungan Nilai *Fitness*

Perhitungan nilai *fitness* dilakukan dengan menjumlahkan data panjang yang tersimpan pada basis data SQLite sesuai dari gen-gen pada setiap kromosom yang diperoleh dari proses sebelumnya. Setelah itu, hasil yang diperoleh diinversi sehingga diperoleh nilai *fitness*.

5.5.4.3 Kode Program Seleksi dengan *Roulette Wheel*

Implementasi algoritma seleksi dengan menggunakan *Roulette Wheel* ditunjukkan pada Gambar 5.19.

```
1   double totalFitness = f1+f2+f3+f4+f5;
2   double p1 = f1/totalFitness;
3   double p2 = f2/totalFitness;
4   double p3 = f3/totalFitness;
5   double p4 = f4/totalFitness;
6   double p5 = f5/totalFitness;
7   double c1 = 0+p1;
8   double c2 = c1+p2;
9   double c3 = c2+p3;
10  double c4 = c3+p4;
11  double c5 = c4+p5;
12  double random1 = Math.random()*1;
13  double random2 = Math.random()*1;
14
15  if (random1 < c1) induk1 = kromosom1;
16  else if(random1 >= c1 && random1 < c2) induk1 = kromosom2;
17  else if(random1 >= c2 && random1 < c3) induk1 = kromosom3;
18  else if(random1 >= c3 && random1 < c4) induk1 = kromosom4;
19  else if(random1 >= c4 && random1 <= c5) induk1 = kromosom5;
20
21  if (random2 < c1){
22     if (random1 < c1) induk2 = kromosom2;
23     else induk2 = kromosom1;
24  }
25  else if(random2 >= c1 && random2 < c2){
26     if (random1 >= c1 && random1 < c2) induk2 = kromosom3;
27     else induk2 = kromosom2;
28  }
29  else if(random2 >= c2 && random2 < c3){
30     if (random1 >= c2 && random1 < c3) induk2 = kromosom4;
31     else induk2 = kromosom3;
32  }
33  else if(random2 >= c3 && random2 < c4){
34     if (random1 >= c3 && random1 < c4) induk2 = kromosom5;
35     else induk2 = kromosom4;
36  }
37  else if(random2 >= c4 && random2 <= c5){
38     if (random1 >= c4 && random1 <= c5) induk2 = kromosom1;
39     else induk2 = kromosom5;}
```

Gambar 5.19 Implementasi Seleksi dengan *Roulette Wheel*

Implementasi algoritma seleksi dengan menggunakan *Roulette Wheel* sesuai dengan perancangan pada diagram alir Gambar 4.6.

5.5.4.4 Kode Program *Crossover*

Implementasi algoritma *crossover* ditunjukkan pada Gambar 5.20.

```
1   public MyCrossover getCrossover(int [] kromosom1, int [] kromosom2){
2     int countK1 = kromosom1.length;
3     int countK2 = kromosom2.length;
4     int tempK1=0,tempK2=0;
5     for(int i=1;i<countK1-1;i++){
6       for(int j=1;j<countK2-1;j++){
7         if(kromosom1[i]==kromosom2[j]){
8           tempK1 = i;
9           tempK2 = j;
10        }
11      }
12    }
13    if(tempK1>0 && tempK2>0){
14      A1=new int[tempK1+1];
15      A2=new int[countK1-(tempK1+1)];
16      B1=new int[tempK2+1];
17      B2=new int[countK2-(tempK2+1)];
18      int loop1=0, loop2=0;
19      for(int i=0;i<countK1;i++){
20        if(i<=tempK1){
21          A1[i]=kromosom1[i];
22        }
23        else {
24          A2[loop1]=kromosom1[i];
25          loop1++;
26        }
27      }
28      for(int j=0;j<countK2;j++){
29        if(j<=tempK2){
30          B1[j]=kromosom2[j];
31        }
32        else {
33           B2[loop2]=kromosom2[j];
34          loop2++;
35        }
36      }
37      A = combine(A1, B2);
38      B = combine(B1, A2);
39      return new MyCrossover(A,B);
40    }
41    else return new MyCrossover(kromosom1,kromosom2);
42  }
```

Gambar 5.20 Implementasi *Crossover*

Implementasi *crossover* diawali dengan mendeteksi gen yang sama pada dua kromosom yang terpilih sesuai hasil seleksi. Ketika ditemukan gen yang sama pada kedua kromosom, maka *crossover* dilakukan dengan memotong gen yang sama tersebut menjadi dua bagian lalu menukarkannya (pindah silang), sehingga terbentuklah kromosom baru. Namun jika tidak ada gen yang sama, maka fungsi akan mengembalikan kromosom tanpa melakukan proses apapun.

5.5.4.5 Kode Program Mutasi

Implementasi algoritma mutasi ditunjukkan pada Gambar 5.21.

```
1  public int [] getMutasi (int [] kromosom1){
2  p = kromosom1.length;
3    if(p>2){
4      int rMut = 1 + (int)(Math.random() * (((p-2) - 1) + 1));
5      ArrayList<Integer> mutasi = new ArrayList<Integer>();
6      ArrayList<Integer> noMutasi = new ArrayList<Integer>();
7
8      mutasi=new int[150];
9      mutasi[0]=kromosom1[rMut];
10
11     query = new DatabaseQuery(this);
12     query.open();
13
14     int pom=0, gen=0;
15   do{
16     if (gen==0){
17       cursor = query.queryGetRandomCabang(mutasi[gen]);
18       if (cursor.moveToFirst()) {
19         while (cursor.isAfterLast() == false) {
20           int cab = cursor.getInt(cursor.getColumnIndex("ujung"));
21           mutasi[gen+1]=cab;
22           cursor.moveToNext();
23         }
24       }
25     }
26     else {
27       cursor = query.queryGetRandomCabang2(mutasi[gen-1], mutasi[gen]);
28       if (cursor.moveToFirst()) {
29         while (cursor.isAfterLast() == false) {
30           int cab = cursor.getInt(cursor.getColumnIndex("ujung"));
31           mutasi[gen+1]=cab;
32           cursor.moveToNext();
33         }
34       }
35     }
36     cursor = query.queryGetCekPom(mutasi[gen+1]);
37     if (cursor.moveToFirst()) {
38       while (cursor.isAfterLast() == false) {
39         pom = cursor.getInt(cursor.getColumnIndex("is_pom"));
40         cursor.moveToNext();
41       }
42     }
43     gen++;
44   }
45   while ( pom==0);
46   query.close();
47
48   int counter = 0;
49   for (int i = 0; i < mutasi.length; i ++)
50     if (mutasi[i] > 0) counter ++;
51   hasil2 = new int[counter];
52   for (int l = 0; l < counter; l++){
53     hasil2[l]=mutasi[l];
54   }
55   hasil = combine(hasil1, hasil2);
56   }
57   else hasil=kromosom1;
58   return hasil;
59 }
```

Gambar 5.21 Implementasi Mutasi

Mutasi diawali dengan membangkitkan bilangan *random*. Bilangan ini menentukan posisi gen yang menjadi titik awal untuk membangkitkan rute baru. Dalam membangkitkan rute baru, algoritma yang digunakan sama dengan algoritma yang digunakan untuk membangkitkan kromosom pada awal proses.

5.6 Implementasi Antarmuka

Bagian ini menampilkan hasil implementasi antarmuka aplikasi pencarian rute terpendek untuk menemukan SPBU terdekat.

5.6.1 Implementasi Antarmuka Halaman Utama Aplikasi

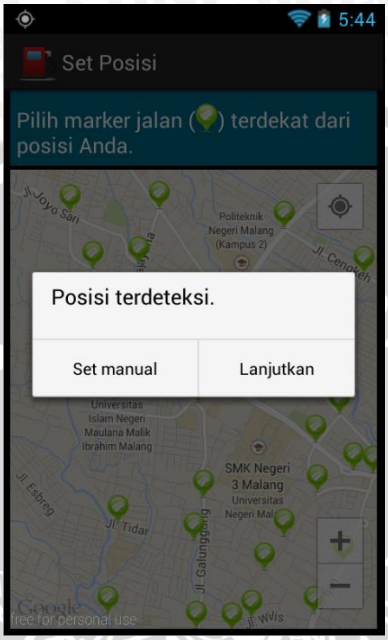
Halaman utama aplikasi merupakan halaman pertama yang muncul ketika aplikasi dibuka. Halaman ini terdiri dari menu pencarian rute, menu penjelasan aplikasi dan menu untuk keluar dari aplikasi. Implementasi halaman utama aplikasi ditunjukkan pada Gambar 5.22.

Gambar 5.22 Implementasi Antarmuka Halaman Utama Aplikasi

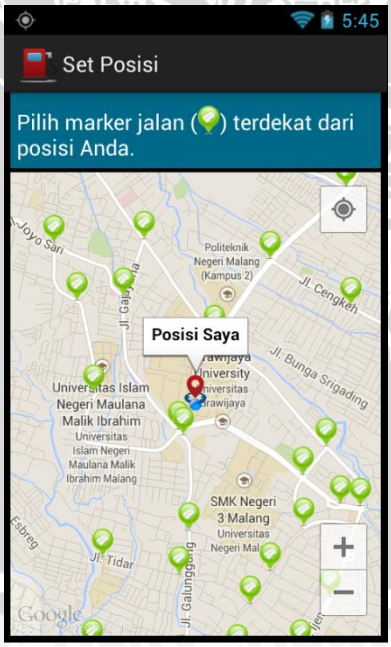
5.6.2 Implementasi Antarmuka Halaman Set Posisi

Halaman set posisi muncul ketika menu pencarian rute dipilih oleh pengguna aplikasi. Terdapat dialog yang muncul ketika masuk pada halaman ini. Dialog tersebut berisi pilihan untuk melakukan set posisi secara manual atau

langsung melanjutkan ke halaman berikutnya yaitu halaman hasil dengan posisi yang terdeteksi oleh sistem. Implementasi halaman set posisi ditunjukkan pada Gambar 5.23 dan 5.24.



Gambar 5.23 Implementasi Antarmuka Halaman Set Posisi

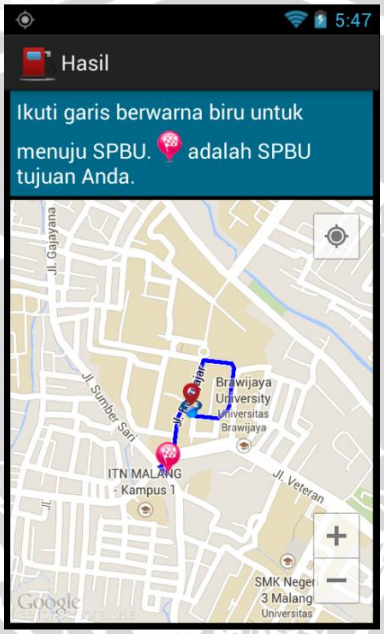


Gambar 5.24 Implementasi Antarmuka Halaman Memilih Marker

Gambar 5.24 menunjukkan halaman set posisi yang menampilkan marker jalan untuk melakukan set posisi secara manual.

5.6.3 Implementasi Antarmuka Halaman Hasil

Halaman hasil merupakan halaman yang muncul setelah pengguna melakukan set posisi baik secara manual ataupun otomatis. Halaman ini menampilkan rute hasil perhitungan dengan menggunakan Algoritma Genetik pada *Google Maps*. Implementasi halaman hasil ditunjukkan pada Gambar 5.25.



Gambar 5.25 Implementasi Antarmuka Halaman Hasil

5.7 Pengujian

Bagian ini menjelaskan tentang pengujian dan analisis yang akan dilakukan pada aplikasi yang telah dibangun. Pengujian dilakukan dengan metode *Black Box Testing*. Pengujian yang dilakukan antara lain pengujian validasi dan pengujian terhadap Algoritma Genetik. Setelah melalui tahap pengujian, selanjutnya dilakukan analisis terhadap hasil dari pengujian yang telah dilakukan. Analisis dilakukan untuk memperoleh kesimpulan dari hasil setiap pengujian yang dilakukan.

5.7.1 Pengujian Validasi Fitur

Pengujian validasi dilakukan untuk memastikan bahwa aplikasi yang dibangun telah memenuhi kebutuhan fungsional sesuai pada bab perancangan. Pengujian validasi merupakan pengujian yang tidak memperhatikan alur algoritma

program atau biasa disebut dengan metode *Black Box Testing*, tetapi lebih ditekankan pada kesesuaian antara sitem yang telah dibangun dengan daftar kebutuhan fungsional.

5.7.1.1 Kasus Uji Validasi Fitur

Untuk melakukan pengujian validasi sebelumnya ditentukan kasus uji sesuai dengan daftar kebutuhan pada bab perancangan. Berikut ini merupakan kasus uji sesuai dengan daftar kebutuhan fungsional. Kasus uji melakukan set posisi ditunjukkan pada Tabel 5.12 dan kasus uji mencari rute terpendek ditunjukkan pada Tabel 5.13.

Tabel 5.12 Kasus Uji Melakukan Set Posisi

Nomor Kasus Uji	UVL_01
Nama Kasus Uji	Kasus Uji Melakukan Set Posisi
Objek Uji	Kebutuhan Fungsional SRS_001
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa set posisi dapat dilakukan baik secara otomatis maupun manual dengan memilih marker jalan.
Data	Otomatis: posisi pengguna di dekat titik dengan id=29. Manual: memilih marker dengan id=29.
Prosedur Uji	1. Memilih menu pencarian rute. 2. Masuk pada halaman set posisi. 3. Muncul dialog yang menampilkan pilihan untuk langsung melanjutkan pada halaman hasil dengan posisi yang terdekteksi oleh sistem atau melakukan set posisi secara manual. 4. Melakukan set posisi (otomatis/manual). 5. Sistem melakukan perhitungan dengan id sesuai posisi yang diset.
Hasil yang Diharapkan	Pengguna dapat melakukan set posisi.

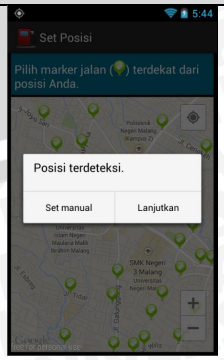
Tabel 5.13 Kasus Uji Mencari Rute Terpendek

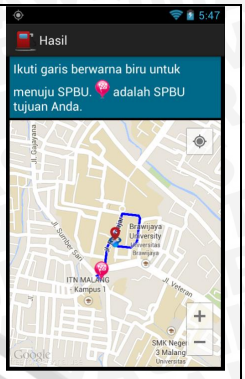
Nomor Kasus Uji	UVL_02
Nama Kasus Uji	Kasus Uji Mencari Rute Terpendek
Objek Uji	Kebutuhan Fungsional SRS_002
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat menampilkan rute hasil perhitungan dengan Algoritma Genetik pada *Google Maps*.
Data	Otomatis: posisi pengguna di dekat titik dengan id=29. Manual: memilih marker dengan id=29.
Prosedur Uji	1. Pengguna melakukan set posisi. 2. Sistem menampilkan rute hasil perhitungan dengan Algoritma Genetik pada *Google Maps*.
Hasil yang Diharapkan	Sistem dapat menampilkan rute hasil perhitungan dengan Algoritma Genetik pada *Google Maps*.

5.7.1.2 Hasil Pengujian Validasi Fitur

Tabel 5.14 merupakan hasil pengujian validasi fitur berdasarkan kasus uji yang telah dibuat sebelumnya.

Tabel 5.14 Hasil Pengujian Validasi Fitur

No.	Nomor Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validasi	Screenshot
1.	UVL_01	Pengguna dapat melakukan set posisi pada halaman set posisi.	Pengguna dapat melakukan set posisi pada halaman set posisi.	VALID	

2.	UVL_02	Sistem dapat menampilkan rute hasil perhitungan dengan Algoritma Genetik pada *Google Maps*.	Sistem dapat menampilkan rute hasil perhitungan dengan Algoritma Genetik pada *Google Maps*.	VALID	

5.7.1.3 Analisis Hasil Pengujian Validasi Fitur

Analisis hasil pengujian validasi dilakukan dengan membandingkan kesesuaian hasil kinerja sistem yang telah dibangun dengan daftar kebutuhan yang telah didefinisikan pada perancangan. Berdasarkan hasil pengujian yang telah dilakukan dapat diketahui bahwa hasil implementasi sistem telah sesuai dengan fitur-fitur pada daftar kebutuhan yang telah didefinisikan pada bab perancangan.

5.7.2 Pengujian Algoritma Genetik

Pengujian Algoritma Genetik dilakukan untuk menguji pengaruh probabilitas *crossover* dan probabilitas mutasi, pengaruh jumlah iterasi atau generasi serta pengujian validasi hasil untuk menguji apakah hasil perhitungan oleh aplikasi sama dengan hasil perhitungan manual dalam memecahkan permasalahan pencarian rute terpendek.

5.7.2.1 Sistematika Pengujian Algoritma Genetik

Pengujian dilakukan dengan menggunakan populasi awal yang sama dan telah ditentukan sebelumnya. Setiap kromosom dibuat bukan merupakan solusi optimum untuk menguji kinerja algoritma apakah dapat menemukan solusi optimum dari pencarian rute terpendek.

Pengujian terhadap probabilitas *crossover* dan probabilitas mutasi dilakukan untuk mengetahui pengaruh probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* yang dihasilkan. Proses pengujian dilakukan dengan mengombinasikan nilai probabilitas baik probabilitas *crossover* maupun

probabilitas mutasi. Nilai probabilitas yang digunakan yaitu 10%, 30%, 50%, 70% dan 90%. Pengujian dilakukan dengan jumlah iterasi atau generasi sebanyak 10 dan jumlah pengujian sebanyak 5 kali untuk setiap kombinasi.

Pengujian terhadap jumlah iterasi atau generasi dilakukan untuk mengetahui pengaruh jumlah iterasi atau generasi terhadap hasil perhitungan algoritma genetik. Pengujian dilakukan dengan mengubah jumlah generasi, jumlah generasi yang digunakan yaitu 1, 5, 10, 20, 35 dan 55 generasi. Pengujian generasi dilakukan sebanyak 10 kali dengan menggunakan probabilitas *crossover* maupun probabilitas mutasi sebesar 90%. Probabilitas *crossover* maupun probabilitas mutasi yang digunakan tinggi agar kemungkinan terjadi *crossover* maupun mutasi menjadi besar. Hasil pengujian ini berupa rata-rata nilai *fitness*.

Pengujian validasi hasil dilakukan dengan menggunakan 5 titik awal serta rute awal (populas awal) yang sama antara perhitungan manual dan perhitungan menggunakan aplikasi. Hasil yang diperoleh berupa rute hasil dari perhitungan manual maupun perhitungan menggunakan aplikasi. Hasil yang diperoleh akan dibandingkan apakah hasil dari perhitungan oleh aplikasi sama dengan perhitungan manual.

5.7.2.2 Hasil dan Analisis Pengujian Probabilitas *Crossover* dan Probabilitas Mutasi Terhadap Rata-rata Nilai *Fitness*

Hasil pengujian terhadap probabilitas *crossover* dan probabilitas mutasi yang dilakukan dengan mengombinasikan nilai probabilitas baik probabilitas *crossover* maupun probabilitas mutasi ditunjukkan Tabel 5.15.

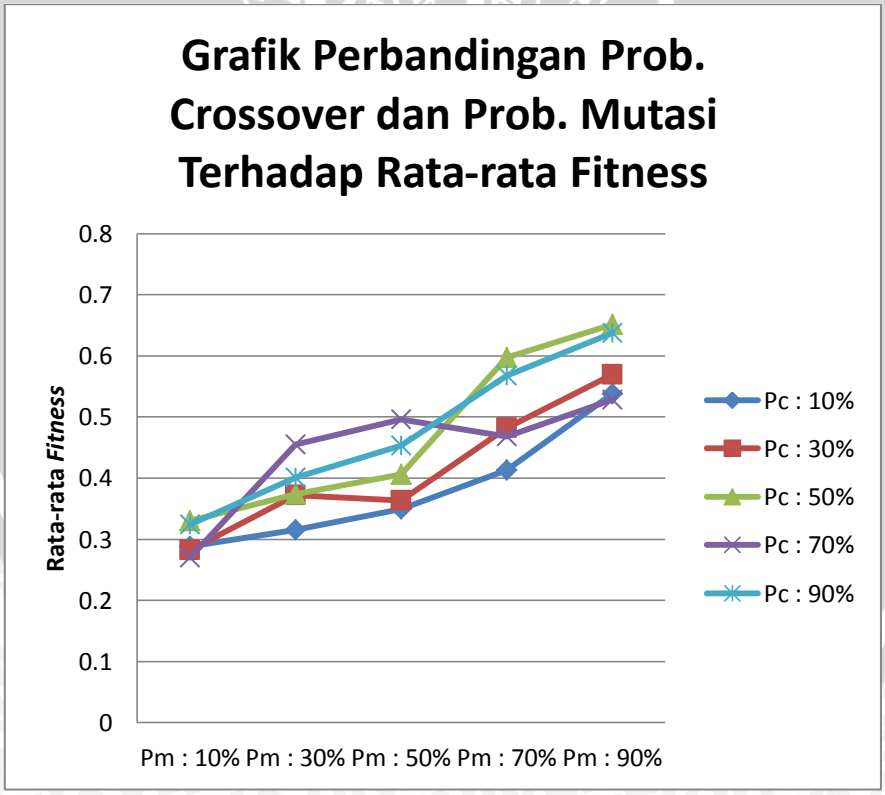
Berdasarkan Tabel 5.15 dapat diketahui bahwa rata-rata nilai *fitness* tertinggi diperoleh pada saat nilai probabilitas *crossover* 50% dan nilai probabilitas mutasi 90%.

Tabel 5.15 Hasil Pengujian Probabilitas *Crossover* dan Probabilitas Mutasi

Rata-rata Nilai *Fitness* **Populasi : 5 - Generasi : 10**										
Pm (%)	**Pc(%)**									
	10		**30**		**50**		**70**		**90**	
10	1	0.26954	1	0.26954	1	0.42194	1	0.26954	1	0.42194
	2	0.26954	2	0.26954	2	0.26954	2	0. 26954	2	0.36098
	3	0.36098	3	0.26954	3	0.26954	3	0. 26954	3	0.30002
	4	0.26954	4	0.26954	4	0.26954	4	0. 26954	4	0.26954
	5	0.26954	5	0.33051	5	0.42194	5	0. 42194	5	0.26954
Rata-rata	**0.287828**		**0.281734**		**0.3305**		**0.26954**		**0.324404**	
30	1	0.27733	1	0.40975	1	0.58796	1	0.37718	1	0.28783
	2	0.26954	2	0.26954	2	0.26954	2	0.26954	2	0.37563
	3	0.39756	3	0.61317	3	0.32295	3	0.26954	3	0.26954
	4	0.36101	4	0.30002	4	0.30002	4	0.80000	4	0.64877
	5	0.26954	5	0.26954	5	0.38840	5	0.55715	5	0.42194
Rata-rata	**0.314996**		**0.372404**		**0.373774**		**0.454682**		**0.400742**	
50	1	0.26968	1	0.26954	1	0.26954	1	0.42194	1	0.57316
	2	0.49755	2	0.32846	2	0.42194	2	0.31771	2	0.26954
	3	0.40930	3	0.36786	3	0.26954	3	0.72438	3	0.42194
	4	0.30002	4	0.48172	4	0.42194	4	0.64877	4	0.29468
	5	0.26954	5	0.36786	5	0.64877	5	0.36485	5	0.70535
Rata-rata	**0.349218**		**0.363088**		**0.406346**		**0.49553**		**0.452934**	
70	1	0.38538	1	0.8	1	0.47260	1	0.38525	1	0.8
	2	0.31148	2	0.41660	2	0.8	2	0.32516	2	0.34048
	3	0.8	3	0.46346	3	0.41660	3	0.8	3	0.62536
	4	0.26954	4	0.33051	4	0.64877	4	0.40625	4	0.42194

	5	0.30002	5	0.39880	5	0.64877	5	0.42194	5	0.64877
Rata -rata	**0.413284**		**0.481874**		**0.597348**		**0.46772**		**0.56731**	
90	1	0.8	1	0.42194	1	0.8	1	0.8	1	0.8
	2	0.72438	2	0.42194	2	0.38004	2	0.29206	2	0.71905
	3	0.29526	3	0.8	3	0.8	3	0.28112	3	0.50165
	4	0.40059	4	0.8	4	0.55715	4	0.8	4	0.36786
	5	0.46935	5	0.40441	5	0.71905	5	0.46935	5	0.8
Rata -rata	**0.537916**		**0.569658**		**0.651248**		**0.528506**		**0.637712**	

Grafik pengaruh probabilitas *crossover* dan probabilitas mutasi ditunjukkan pada Gambar 5.26.



Gambar 5.26 Grafik Perbandingan Probabilitas *Crossover* dan Probabilitas Mutasi Terhadap Rata-rata *Fitness*

Berdasarkan Gambar 5.26 rata-rata nilai *fitness* mengalami kenaikan sesuai dengan semakin besarnya probabilitas mutasi dan stabil ketika probabilitas *crossover* bernilai 90%. Hal ini disebabkan karena semakin besar probabilitas mutasi maka semakin besar pula kemungkinan perubahan susunan gen pada kromosom induk dengan cara membentuk rute baru dengan titik awal yaitu gen yang terpilih untuk dimutasi, sehingga jika populasi awal memiliki nilai rata-rata *fitness* yang rendah maka dapat diperbaiki dengan membentuk rute baru melalui mutasi. Namun, rute baru yang terbentuk dengan titik awal gen yang terpilih untuk dimutasi tidak selalu menghasilkan rute yang lebih pendek dari pada rute sebelum terjadinya mutasi, sehingga ada beberapa rata-rata nilai *fitness* yang mengalami penurunan meskipun terjadi mutasi.

Sedangkan pengaruh probabilitas *crossover* pada rata-rata *fitness* tidak semuanya mengakibatkan grafik naik. Ada beberapa titik yang menyebabkan grafik menjadi turun. Hal ini disebabkan karena terjadinya *crossover* juga dipengaruhi oleh adanya gen yang sama pada kedua kromosom induk. Jika tidak ada gen yang sama maka *crossover* tidak dapat dilakukan meskipun probabilitas *crossover* bernilai tinggi dan pada kedua kromosom induk memungkinkan terjadi *crossover* berdasarkan probabilitas *crossover* (Pc), namun jika tidak ada gen yang sama maka crossover tidak dapat dilakukan. Hal ini menyebabkan kemungkinan terjadinya *crossover* menjadi kecil meskipun probabilitas *crossover* (Pc) bernilai tinggi.

Secara keseluruhan pengaruh probabilitas *crossover* dan probabilitas mutasi menyebabkan grafik cenderung naik meskipun ada beberapa titik yang turun. Artinya, probabilitas *crossover* dan probabilitas mutasi berpengaruh pada besarnya nilai *fitness*. Semaikin besar probabilitas *crossover* dan probabilitas mutasi maka nilai *fitness* yang dihasilkan juga semakin besar.

5.7.2.3 Hasil dan Analisis Pengujian Jumlah Generasi

Pengujian jumlah iterasi atau generasi dilakukan dengan jumlah populasi sebanyak 5, probabilitas *crossover* sebesar 90% dan probabilitas mutasi sebesar 90%. Pengujian dilakukan dengan nilai probabilitas tinggi dikarenakan untuk memperbesar kemungkinan perubahan gen-gen pada kromosom melalui proses

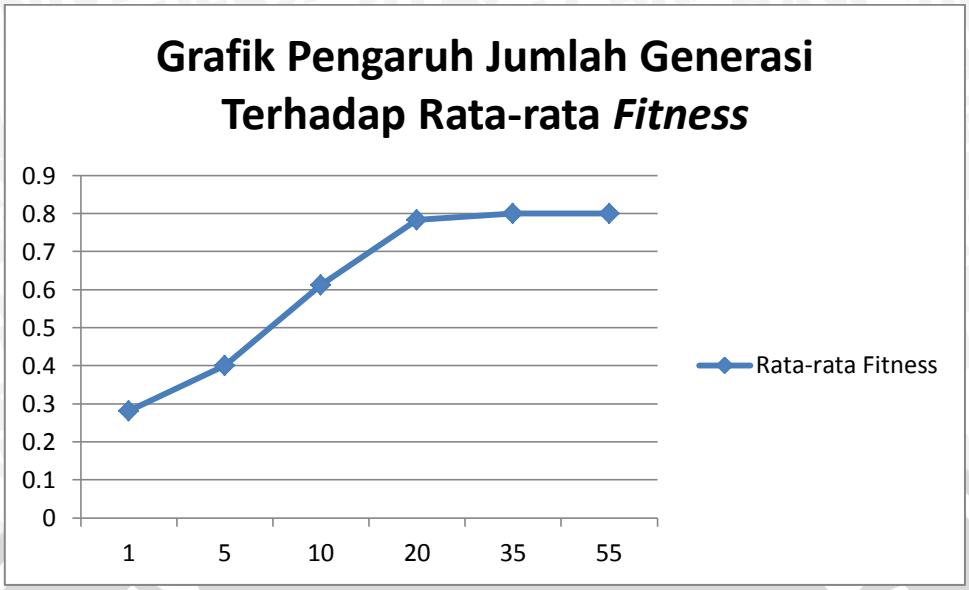
crossover dan mutasi, sehingga individu baru yang dihasilkan menjadi lebih bervariasi. Hasil pengujian terhadap jumlah iterasi atau generasi yang dilakukan untuk mengetahui pengaruh jumlah iterasi atau pengaruh jumlah generasi terhadap hasil perhitungan algoritma genetik ditunjukkan Tabel 5.16.

Tabel 5.16 Hasil Pengujian Jumlah Generasi

Percobaan ke-	**Rata-rata Nilai *Fitness* Populasi : 5 - Pc : 90% - Pm : 90%**					
	Generasi					
	1	**5**	**10**	**20**	**35**	**55**
1	0.34838	0.41126	0.39374	0.8	0.8	0.8
2	0.34838	0.40059	0.54879	0.8	0.8	0.8
3	0.27277	0.37733	0.64877	0.8	0.8	0.8
4	0.24243	0.28247	0.48154	0.8	0.8	0.8
5	0.25916	0.63812	0.8	0.63810	0.8	0.8
6	0.23213	0.27859	0.71905	0.8	0.8	0.8
7	0.23418	0.35076	0.8	0.8	0.8	0.8
8	0.38994	0.49317	0.50165	0.8	0.8	0.8
9	0.23720	0.34343	0.71220	0.8	0.8	0.8
10	0.24808	0.42830	0.51831	0.8	0.8	0.8
Waktu (detik)	± 1-2	±2- 3	± 3- 5	± 5-7	± 7-9	± 9-11
Rata-rata	**0.281265**	**0.400402**	**0.612405**	**0.78381**	**0.8**	**0.8**

Berdasarkan Tabel 5.16 dapat diketahui bahwa rata-rata nilai *fitness* tertinggi diperoleh ketika iterasi atau generasi berjumlah banyak yaitu 35 dan 55 generasi dengan rata-rata *fitness* sebesar 0.8. Namun semakin banyak iterasi atau generasi, semakin lama pula waktu eksekusi yang dibutuhkan.

Grafik pengaruh pengaruh jumlah generasi ditunjukkan pada Gambar 5.27.



Gambar 5.27 Grafik Pengaruh Jumlah Generasi Terhadap Rata-rata *Fitness*

Berdasarkan grafik Gambar 5.27 dapat diketahui bahwa terjadi kenaikan sesuai semakin banyaknya jumlah generasi. Grafik atau nilai *fitness* cenderung naik dikarenakan semakin banyak iterasi atau generasi maka akan mengakibatkan proses evolusi semakin sering terjadi, sehingga individu-individu baru yang dihasilkan juga akan semakin bervariasi dan memungkinkan variasi pada nilai *fitness* yang dihasilkan. Dengan demikian kemungkinan memperoleh *fitness* yang tinggi akan semakin besar. Tetapi, semakin banyak iterasi dapat mempengaruhi waktu eksekusi dari Algoritma Genetik. Sesuai Tabel 5.17, semakin banyak iterasi yang digunakan, semakin lama waktu eksekusi yang diperlukan.

Pada jumlah iterasi atau generasi 35 dan 55 memiliki rata-rata nilai *fitness* yang sama yaitu 0.8. Hal tersebut menandakan bahwa mulai jumlah iterasi atau generasi 35 nilai *fitness* yang dihasilkan tinggi dan sudah mencapai hasil yang optimal pada seluruh kromosom (konvergen) pada 10 kali percobaan yang dilakukan.

Dapat disimpulkan bahwa semakin besar jumlah iterasi atau generasi, maka semakin besar kemungkinan memperoleh *fitness* yang tinggi serta waktu eksekusi yang diperlukan juga semakin lama.

5.7.2.4 Hasil dan Analisis Pengujian Validasi Hasil

Pengujian validasi hasil dilakukan dengan menggunakan 5 titik awal yang sama antara perhitungan manual dan perhitungan menggunakan aplikasi. Titik awal yang digunakan yaitu 3, 2, 40, 35 dan 70. Hasil pengujian ditunjukkan Tabel 5.17.

Tabel 5.17 Hasil Pengujian Validasi Hasil

No	Perhitungan Manual		Aplikasi		
	Rute Hasil	***Fitness***	**Rute Hasil**	***Screenshot***	***Fitness***
1.	3-10-11-901	0.80000	3-10-11-901		0.80000
2.	2-26-28-903	0.56179	2-26-28-903		0.56179
3.	40-39-41-42-35-34-24-903	0.61276	40-39-41-42-35-34-24-903		0.61276

4.	35-34-24-903	1.58730	35-34-24-903		1.58730
5.	70-79-906	1.81818	70-79-906		1.81818

Tabel 5.17 pada nomor 1 titik awal berada pada titik nomor 3, nomor 2 titik awal berada pada titik nomor 2, nomor 3 titik awal berada pada titik nomor 40, nomor 4 titik awal berada pada titik nomor 35 dan nomor 5 titik awal berada pada titik nomor 70. Rute hasil merupakan titik-titik yang harus dilewati untuk menuju tujuan yaitu SPBU terdekat. Sedangkan kolom *fitness* merupakan nilai *fitness* dari rute hasil. Berdasarkan pengujian validasi hasil pada Tabel 5.17 dapat diketahui bahwa hasil dari aplikasi telah sesuai dengan hasil perhitungan manual.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Perancangan dan implementasi aplikasi berbasis yang dapat membantu pengguna dalam menemukan lokasi SPBU terdekat dapat dilakukan dengan pemrograman berorientasi objek dan memanfaatkan *Google Maps* untuk menampilkan rute hasil.
2. Hasil yang berupa rute terpendek menuju SPBU terdekat ditampilkan dengan cara menarik garis antara satu titik ke titik yang lain sesuai hasil Algoritma menggunakan fungsi yang ada pada *Google Maps* agar rute yang digambarkan sesuai dengan bentuk jalan pada peta.
3. Semakin besar probabilitas *crossover* dan probabilitas mutasi maka nilai *fitness* yang dihasilkan juga semakin besar artinya hasil semakin baik.
4. Semakin banyak iterasi atau generasi dapat mengakibatkan proses evolusi semakin sering terjadi. Individu-individu baru yang dihasilkan akan semakin bervariasi dan memungkinkan nilai *fitness* yang dihasilkan lebih bervariasi. Dengan demikian, kemungkinan memperoleh *fitness* yang tinggi akan semakin besar. Namun, semakin banyak iterasi waktu eksekusi yang diperlukan juga semakin lama. Berdasarkan hasil pengujian, rata-rata nilai *fitness* tertinggi dperoleh pada saat jumlah generasi 35 dan 55.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan aplikasi pencarian rute menuju SPBU adalah sebagai berikut:

1. Perlu dilakukan pengembangan pada sistem operasi mobile yang lainnya seperti Windows Phone dan iOS.
2. Aplikasi perlu dikembangkan dengan menambahkan faktor lain yang mempengaruhi pemilihan rute, misalnya waktu tempuh perjalanan, kepadatan lalu lintas dan pengaruh lampu lalu lintas pada persimpangan jalan.

DAFTAR PUSTAKA

[AFA-11] Afandi, Fachrudin, ER.,Mahendrawati S.T, M.Sc, Ph.D, Mahananto, Faizal, S.Kom, *Penerapan Algoritma Genetika untuk Masalah Penjadwalan Job Shop pada Lingkungan Industri Pakaian*, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS) Kampus Keputih, Sukolilo, Surabaya.

[BAS-12] Baskara J, Adi dan Nurcahyawati, Vivin,. 2012, "Penentuan Jarak Terpendek Pada Jalur Distribusi Barang Di Pulau Jawa Dengan Menggunakan Algoritma Genetika", *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, Vol. 1, No. 3, hal.244-258.

[DYA-12] Dyah P, Ayu, Martiana, Entin, K,S.Kom,M.Kom, Setiowati, Yuliana, S.Kom,M.Kom., *Rancang Bangun dan Penerapan Algoritma Backtrack pada Labirin Matematika Berbasis Android*, Institut Teknologi Sepuluh Nopember, Surabaya.

[ELI-12] Elian, Alqod, Mazharuddin S, Ary, Studiawan, Hildan, 2012, "Layanan Informasi Kereta Api Menggunakan GPS, Google Maps, dan Android", *Jurnal Teknik Pomits*, Vol. 1, No. 1, hal.1-6.

[FIR-13] Anitya, Firar S, Sugiharti, Endang, Dwijanto, 2013, "Implementasi Algoritma Genetika untuk Menyelesaikan Travelling Salesman Problem", *UNNES Jurnal of Mathematics*, hal.117-120.

[HAB-11] Habibi, Wildan, Mazharuddin S, Ary, S.Kom, M.Comp.Sc, 2011, *Pembangunan Sistem Pelacakan dan Penelusuran Device Mobile Berbasis Global Positioning System (GPS) pada Platform Mobile Google*, Institut Teknologi Sepuluh Nopember, Surabaya.

[HER-13] Hernita, P., 2013, *Android Programming with Eclipse*, Wahana Komputer dan Penerbit ANDI, Yogyakarta.

[HID-07] Hidayanto, Achmad, *Penerapan Algoritma Genetika pada Perencanaan Lintasan Kendaraan*, Uiversitas Diponegoro, Semarang.

[HUM-12] Humala, Arief, S.Kom. *Pembuatan Aplikasi Pencarian Halte Transjakarta Terdekat Berbasis Android*, Universitas Gunadarma, Pondok Aren Tangerang Selatan.

[KAR-11] Karas, I.R. dan U. Atila. 2011. *A Genetic Algorithm approach for finding the shortest driving time on mobile devices*, vol.6(2), 394-405.

[MAR-13] Martin, Rudi dan Hamsi, Alfian. 2013. *Optimasi Parameter Las Busur Listrik Arus Searah Tipe Legs 225 dengan Menggunakan Metode Optimasi ALgoritma Genetika.*. Universitas Sumatera Utara, Medan.

[MIT-99] Mitchell, Melanie. 1999. *An Introduction to Genetic Algorithms.* Massachusetts Institute of Technology. A Bradford Book The MIT Press. England.

[NIZ-14] Nizar, Chairil. *Klasifikasi Jalan Menurut Fungsinya*. http://www.ilmusipil.com/klasifikasi-jalan-menurut-fungsi (diakses tanggal 3 Juli 2014).

[ROM-12] Rompas, B.R, 2012, *Aplikasi Location-Based Service Pencarian Tempat di Kota Manado Berbasis Android*. UNSRAT, Manado.

[SAF-12] Safaat, Nazruddin, 2012, *Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Informatika, Bandung.

[YUD-12] Yudho S, Bhakti dan Sariman, 2012, "Metode Algoritma Genetika dengan Sistem Fuzzy Logic untuk Penentuan Parameter Pengendali PID", *Jurnal Rekayasa Elektrika,* Vol. 10, No. 2, hal.32-38.

[YUS-12] Yusuf, Akhmad dan Soesanto, Oni, 2012, "Algoritma Genetika pada Penyelesaian Akar Persamaan Sebuah Fungsi", *Jurnal Matematika Murni dan Terapan*, Vol. 6, No. 2, hal.47-56.

LAMPIRAN

Lampiran 1. Perhitungan Manual

Perhitungan Manual 1

Populasi Awal

Kromosom 1	3	5	6	20	22	21	10	11	901	
Kromosom 2	3	5	6	16	17	7	902			
Kromosom 3	3	5	6	20	22	21	9	8	902	
Kromosom 4	3	5	6	20	22	21	9	8	13	12	901
Kromosom 5	3	5	6	16	18	19	17	7	902	

Fitness

Kromosom 1	0.9	0.6	0.7	0.35	0.16	0.55	0.3	0.15		=	0.27
Kromosom 2	0.9	0.6	0.9	0.65	1	0.07				=	0.24
Kromosom 3	0.9	0.6	0.7	0.35	0.16	0.65	0.7	0.58		=	0.22
Kromosom 4	0.9	0.6	0.7	0.35	0.16	0.65	0.7	0.65	1.4	0.4	=	0.15
Kromosom 5	0.9	0.6	0.9	0.3	0.55	0.35	1	0.07			=	0.21
Total:											=	1.1

Probabilitas

Kromosom 1	:	0.246040021
Kromosom 2	:	0.221555456
Kromosom 3	:	0.196725965
Kromosom 4	:	0.140216356
Kromosom 5	:	0.195462201

Range

Range 1	:	0	-	0.246040021
Range 2	:	0.246040021	-	0.467595477
Range 3	:	0.467595477	-	0.664321443
Range 4	:	0.664321443	-	0.804537799
Range 5	:	0.804537799	-	1

Hasil Seleksi (Kromosom 1 dan Kromosom 4)

Offspring 1	:	3	5	6	20	22	21	10	11	901	
Offspring 2	:	3	5	6	20	22	21	9	8	13	12	901

Crossover

Offspring 1	:	3	5	6	20	22	21	10	11	901	
Offspring 2	:	3	5	6	20	22	21	9	8	13	12	901

Hasil Crossover

Offspring 1 :	3	5	6	20	22	21	9	8	13	12	901
Offspring 2 :	3	5	6	20	22	21	10	11	901		

Mutasi

Offspring 1 :	3	5	6	20	22	21	9	8	13	12	901
Offspring 2 :	3	5	6	20	22	21	10	11	901		

Hasil Mutasi

Offspring 1 :	3	5	6	20	22	21	9	8	902		
Offspring 2 :	3	10	11	901							

Hasil

Offspring 2 :	3	10	11	901							
Kromosom 1 :	3	5	6	20	22	21	10	11	901		
Kromosom 2 :	3	5	6	16	17	7	902				
Kromosom 3 :	3	5	6	20	22	21	9	8	902		
Offspring 1 :	3	5	6	20	22	21	9	8	902		
Kromosom 5 :	3	5	6	16	18	19	17	7	902		
Kromosom 4 :	3	5	6	20	22	21	9	8	13	12	901

Fitness

0.8
0.269542
0.242718
0.215517
0.215517
0.214133
0.15361

Perhitungan Manual 2

Populasi Awal

Kromosom 1	2	3	5	6	7	902			
Kromosom 2	2	26	28	903					
Kromosom 3	2	3	10	11	901				
Kromosom 4	2	3	10	21	22	20	6	7	902
Kromosom 5	2	3	10	21	9	8	902		

Fitness

Kromosom 1	0.75	0.9	0.6	0.8	0.07	= 0.32
Kromosom 2	1.2	0.28	0.3			= 0.56
Kromosom 3	0.75	0.8	0.3	0.15		= 0.5

Kromosom 4	0.75	0.8	0.55	0.16	0.35	0.7	0.8	0.07	=	0.24
Kromosom 5	0.75	0.8	0.55	0.65	0.7	0.65			=	0.24
Total:									=	1.87

Probabilitas

Kromosom 1	:	0.171815517
Kromosom 2	:	0.301159783
Kromosom 3	:	0.268032207
Kromosom 4	:	0.128245075
Kromosom 5	:	0.130747418

Range

Range 1	:	0	-	0.171815517
Range 2	:	0.171815517	-	0.4729753
Range 3	:	0.4729753	-	0.741007507
Range 4	:	0.741007507	-	0.869252582
Range 5	:	0.869252582	-	1

Hasil Seleksi (Kromosom 4 dan Kromosom 5)

Offspring 1	:	2	3	10	21	22	20	6	7	902
Offspring 2	:	2	3	10	21	9	8	902		

Crossover

Offspring 1	:	2	3	10	21	22	20	6	7	902
Offspring 2	:	2	3	10	21	9	8	902		

Hasil Crossover

Offspring 1	:	2	3	10	21	9	8	902		
Offspring 2	:	2	3	10	21	22	20	6	7	902

Mutasi

Offspring 1	:	2	3	10	21	9	8	902		
Offspring 2	:	2	3	10	21	22	20	6	7	902

Hasil Mutasi

Offspring 1	:	2	3	10	21	9	8	902		
Offspring 2	:	2	3	10	21	9	8	902		

Hasil

Kromosom 2	:	2	26	28	903		
Kromosom 3	:	2	3	10	11	901	
Kromosom 1	:	2	3	5	6	7	902
Kromosom 5	:	2	3	10	21	9	8	902
Offspring 1	:	2	3	10	21	9	8	902

Offspring 2	:	2	3	10	21	9	8	902	
Kromosom 4	:	2	3	10	21	22	20	6	7	902

Fitness

0.561798
0.5
0.320513
0.243902
0.243902
0.243902
0.239234

Perhitungan Manual 3

Populasi Awal

Kromosom 1	40	39	38	37	36	24	903	
Kromosom 2	40	43	100	904				
Kromosom 3	40	39	41	37	36	24	903	
Kromosom 4	40	43	100	35	36	24	903	
Kromosom 5	40	39	41	42	35	34	24	903

Fitness

Kromosom 1	0.22	0.55	0.45	0.45	0.65	0.22		=	0.39
Kromosom 2	0.65	0.35	0.78					=	0.56
Kromosom 3	0.22	0.16	0.45	0.45	0.65	0.22		=	0.47
Kromosom 4	0.65	0.35	0.7	0.4	0.65	0.22		=	0.34
Kromosom 5	0.22	0.16	0.35	0.26	0.2	0.21	0.22	=	0.62
Total:								=	2.37

Probabilitas

Kromosom 1	:	0.165796739
Kromosom 2	:	0.236586357
Kromosom 3	:	0.195871496
Kromosom 4	:	0.141792497
Kromosom 5	:	0.259952911

Range

Range 1	:	0	-	0.165796739
Range 2	:	0.165796739	-	0.402383096
Range 3	:	0.402383096	-	0.598254592
Range 4	:	0.598254592	-	0.740047089
Range 5	:	0.740047089	-	1

Hasil Seleksi (Kromosom 3 dan Kromosom 4)

Offspring 1	:	40	39	41	37	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Crossover

Offspring 1	:	40	39	41	37	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Hasil Crossover

Offspring 1	:	40	39	41	37	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Mutasi

Offspring 1	:	40	39	41	37	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Hasil Mutasi

Offspring 1	:	40	39	38	37	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Hasil

Kromosom 5	:	40	39	41	42	35	34	24	903
Kromosom 2	:	40	43	100	904				
Kromosom 3	:	40	39	41	37	36	24	903
Kromosom 1	:	40	39	38	37	36	24	903
Offspring 1	:	40	39	38	37	36	24	903
Kromosom 4	:	40	43	100	35	36	24	903
Offspring 2	:	40	43	100	35	36	24	903

Fitness

0.617284
0.561798
0.465116
0.393701
0.393701
0.3367
0.3367

Perhitungan Manual 4

Populasi Awal

Kromosom 1	35	34	24	903		
Kromosom 2	35	36	24	903		
Kromosom 3	35	100	904			
Kromosom 4	35	42	43	100	904	
Kromosom 5	35	42	37	36	24	903

Fitness

Kromosom 1	0.2	0.21	0.22			=	1.59
Kromosom 2	0.4	0.65	0.22			=	0.79
Kromosom 3	0.7	0.78				=	0.68
Kromosom 4	0.26	0.5	0.35	0.78		=	0.53
Kromosom 5	0.26	0.45	0.45	0.65	0.22	=	0.49
Total:						=	4.07

Probabilitas

Kromosom 1	:	0.389800203
Kromosom 2	:	0.193365455
Kromosom 3	:	0.165928465
Kromosom 4	:	0.129933401
Kromosom 5	:	0.120972477

Range

Range 1	:	0	-	0.389800203
Range 2	:	0.389800203	-	0.583165658
Range 3	:	0.583165658	-	0.749094122
Range 4	:	0.749094122	-	0.879027523
Range 5	:	0.879027523	-	1

Hasil Seleksi (Kromosom 3 dan Kromosom 4)

Offspring 1	:	35	100	904		
Offspring 2	:	35	42	43	100	904

Crossover

Offspring 1	:	35	100	904		
Offspring 2	:	35	42	43	100	904

Hasil Crossover

Offspring 1	:	35	100	904		
Offspring 2	:	35	42	43	100	904

Mutasi

Offspring 1	:	35	100	904		
Offspring 2	:	35	42	43	100	904

Hasil Mutasi

Offspring 1	:	35	100	904		
Offspring 2	:	35	42	43	100	904

Hasil

Kromosom 1	:	35	34	24	903		
Kromosom 2	:	35	36	24	903		
Kromosom 3	:	35	100	904			
Offspring 1	:	35	100	904			
Kromosom 4	:	35	42	43	100	904	
Offspring 2	:	35	42	43	100	904	
Kromosom 5	:	35	42	37	36	24	903

Fitness

1.587302
0.787402
0.675676
0.675676
0.529101
0.529101
0.492611

Perhitungan Manual 5

Populasi Awal

Kromosom 1	70	79	906			
Kromosom 2	70	97	909			
Kromosom 3	70	74	76	907		
Kromosom 4	70	69	907			
Kromosom 5	70	79	75	74	76	907

Fitness

Kromosom 1	0.45	0.1				=	1.82
Kromosom 2	0.65	0.3				=	1.05
Kromosom 3	0.17	0.5	0.35			=	0.98
Kromosom 4	0.8	0.05				=	1.18
Kromosom 5	0.45	0.2	0.45	0.5	0.35	=	0.51
Total:						=	5.54

Probabilitas

Kromosom 1	:	0.328162245
Kromosom 2	:	0.189988668
Kromosom 3	:	0.17695023
Kromosom 4	:	0.212340276
Kromosom 5	:	0.092558582

Range

Range 1	:	0	-	0.328162245
Range 2	:	0.328162245	-	0.518150912
Range 3	:	0.518150912	-	0.695101142
Range 4	:	0.695101142	-	0.907441418
Range 5	:	0.907441418	-	1

Hasil Seleksi (Kromosom 1 dan Kromosom 2)

Offspring 1	:	70	79	906
Offspring 2	:	70	97	909

Crossover

Offspring 1	:	70	79	906
Offspring 2	:	70	97	909

Hasil Crossover

Offspring 1	:	70	79	906
Offspring 2	:	70	97	909

Mutasi

Offspring 1	:	70	79	906
Offspring 2	:	70	97	909

Hasil Mutasi

Offspring 1	:	70	79	906	75	74	76	907
Offspring 2	:	70	97	909				

Hasil

Kromosom 1	:	70	79	906			
Kromosom 4	:	70	69	907			
Kromosom 2	:	70	97	909			
Offspring 2	:	70	97	909			
Kromosom 3	:	70	74	76	907		
Kromosom 5	:	70	79	75	74	76	907
Offspring 1	:	70	79	75	74	76	907

Fitness
1.818182
1.176471
1.052632
1.052632
0.980392
0.512821
0.512821

Lampiran 2. Data Riil

Data Jalan

nama_jalan	pangkal	ujung	panjang
Mayjend MT Haryono	27	1	0.65
Mayjend MT Haryono	1	2	0.95
Mayjen Panjaitan	2	26	1.2
Mayjen Panjaitan	26	24	0.6
Brigjend Slamet Riadi	24	23	1.5
Soekarno Hatta	2	3	0.75
Soekarno Hatta	3	2	0.75
Soekarno Hatta	3	10	0.8
Soekarno Hatta	10	3	0.8
Soekarno Hatta	10	11	0.3
Soekarno Hatta	11	10	0.3
Soekarno Hatta	11	901	0.15
Soekarno Hatta	901	11	0.15
Soekarno Hatta	901	12	0.4
Soekarno Hatta	12	901	0.4
Borobudur	12	13	1.4
Borobudur	13	12	1.4
Candi Mendut	10	21	0.55
Candi Mendut	21	10	0.55
Candi Mendut	21	9	0.65
Candi Mendut	9	21	0.65
Candi Telagawangi	9	8	0.7
Candi Telagawangi	8	9	0.7
Puncak Borobudur	14	11	0.95
Candi Panggung	15	10	1.4
Terusan Kendalsari	21	22	0.16
Terusan Kendalsari	22	21	0.16
Bukit Sari	22	20	0.35
Bukit Sari	20	22	0.35
Cengger Ayam	20	6	0.7
Cengger Ayam	6	20	0.7
Kedawung	6	7	0.8
Kedawung	7	6	0.8
Mawar	6	16	0.9
Mawar	16	6	0.9
Sarangan	16	17	0.65
Sarangan	17	16	0.65
Tawangmangu	16	18	0.3

Tawangmangu	18	16	0.3
Kaliurang	18	19	0.55
Kaliurang	19	18	0.55
Letjend Sutoyo	19	17	0.35
Letjend Sutoyo	17	19	0.35
S Parman	17	7	1
S Parman	7	17	1
S Parman	7	902	0.07
S Parman	902	7	0.07
S Parman	902	8	0.58
S Parman	8	902	0.58
S Parman	8	13	0.65
S Parman	13	8	0.65
Cengkeh	3	5	0.9
Cengkeh	5	3	0.9
Kalpataru	5	6	0.6
Kalpataru	6	5	0.6
Jaksa Agung Suprapto	78	19	0.35
Jaksa Agung Suprapto	19	78	0.35
Jaksa Agung Suprapto	77	78	0.4
Jaksa Agung Suprapto	78	77	0.4
Jaksa Agung Suprapto	77	23	0.4
Jaksa Agung Suprapto	23	77	0.4
Bandung	24	903	0.22
Bandung	903	24	0.22
Bandung	903	28	0.3
Bandung	28	903	0.3
Bogor	26	28	0.28
Bogor	28	34	0.4
Bogor	34	28	0.4
Jakarta	24	34	0.21
Jakarta	34	24	0.21
Jakarta	34	35	0.2
Jakarta	35	34	0.2
Veteran	28	29	1.1
Veteran	29	28	1.1
Sumbersari	29	30	1.1
Gajayana	30	1	0.45
Bendungan Sigura-gura	29	33	1.2
Bendungan Sigura-gura	33	29	1.2
Sunan Kalijaga	33	32	0.75
Sunan Kalijaga	32	33	0.75

Mertojoyo	32	31	0.5
Mertojoyo	31	32	0.5
Mertojoyo	31	27	0.7
Simpang Gajayana	30	31	0.75
Simpang Gajayana	31	30	0.75
Ijen	36	24	0.65
Ijen	24	36	0.65
Ijen	37	36	0.45
Ijen	36	37	0.45
Ijen	38	37	0.45
Ijen	37	38	0.45
Semeru	37	46	1
Semeru	46	37	1
Jend Basuki Rahmat	23	46	0.22
Jend Basuki Rahmat	46	23	0.22
Jend Basuki Rahmat	46	48	0.6
Jend Basuki Rahmat	48	46	0.6
Nasional	47	48	0.28
Kawi	38	47	0.95
Kawi	47	38	0.95
Kawi Atas	39	38	0.55
Kawi Atas	38	39	0.55
Wilis	39	41	0.16
Wilis	41	39	0.16
Wilis	41	37	0.45
Wilis	37	41	0.45
Raya Dieng	40	39	0.22
Raya Dieng	39	40	0.22
Simpang Wilis	41	42	0.35
Simpang Wilis	42	41	0.35
Retawu	42	37	0.45
Retawu	37	42	0.45
Gede	42	35	0.26
Gede	35	42	0.26
Bondowoso	43	42	0.5
Bondowoso	42	43	0.5
Surabaya	100	35	0.7
Surabaya	35	100	0.7
Galunggung	40	43	0.65
Galunggung	43	40	0.65
Galunggung	43	100	0.35
Galunggung	100	43	0.35

Bendungan Sutami	100	904	0.78
Bendungan Sutami	904	100	0.78
Bendungan Sutami	904	29	0.02
Bendungan Sutami	29	904	0.02
Tidar	45	43	0.65
Dieng	44	40	0.65
Raya Langsep	40	49	1.1
Raya Langsep	49	40	1.1
Ikhwan Ridwan Rais	49	50	0.8
Ikhwan Ridwan Rais	50	49	0.8
Brigjend Katamso	50	51	0.4
Brigjend Katamso	51	50	0.4
Kyai H. Hasyim Asyari	51	52	0.17
Kyai H. Hasyim Asyari	52	51	0.17
Kyai H. Hasyim Asyari	52	47	0.16
Kyai H. Hasyim Asyari	47	52	0.16
Kauman	53	52	0.3
Merdeka Barat	48	53	0.15
Ade Irma Suryani	113	54	0.24
Pasar Besar	54	55	0.55
Yulius Usman	56	912	0.2
Yulius Usman	912	56	0.2
Kapten Pierre Tandean	112	57	0.2
Kyai Tamin	58	59	0.5
Kyai Tamin	59	58	0.5
Martadinata	61	59	0.21
Martadinata	59	61	0.21
Martadinata	59	55	0.21
Martadinata	55	59	0.21
Arif Margono	62	56	0.35
Arif Margono	56	62	0.36
Arif Margono	56	51	0.12
Arif Margono	51	56	0.12
Sudanco Supriadi	63	905	0.7
Sudanco Supriadi	905	63	0.7
Sudanco Supriadi	905	60	0.05
Sudanco Supriadi	60	905	0.05
Sudanco Supriadi	60	62	0.75
Sudanco Supriadi	62	60	0.75
Mergan	49	60	1.4
Mergan	60	49	1.4
Janti Barat	63	64	1

Janti Barat	64	63	1
Sonokeling	64	65	0.35
Sonokeling	65	64	0.35
Niaga	65	66	0.55
Niaga	66	65	0.55
Kebalen	61	67	0.9
Kebalen	67	61	0.9
Juanda	68	67	0.75
Aris Munandar	72	68	0.55
Gatot Subroto	55	68	0.45
Gatot Subroto	68	55	0.45
Gatot Subroto	68	69	0.2
Gatot Subroto	69	68	0.2
Panglima Sudirman	69	70	0.8
Panglima Sudirman	70	69	0.8
Panglima Sudirman	70	79	0.45
Panglima Sudirman	79	70	0.45
Panglima Sudirman	79	906	0.1
Panglima Sudirman	906	79	0.1
Panglima Sudirman	906	71	0.05
Panglima Sudirman	71	906	0.05
WR Supratman	19	71	0.4
WR Supratman	71	19	0.4
Pahlawan Trip	35	36	0.4
Pahlawan Trip	36	35	0.4
Kahuripan	46	73	0.4
Kahuripan	73	46	0.4
Kertanegara	73	76	0.2
Kertanegara	76	73	0.2
Trunojoyo	69	907	0.05
Trunojoyo	907	69	0.05
Trunojoyo	907	76	0.35
Trunojoyo	76	907	0.35
Trunojoyo	76	74	0.5
Trunojoyo	74	76	0.5
Cokroaminoto	74	75	0.45
Cokroaminoto	75	74	0.45
Dr Cipto	75	78	0.35
Dr Cipto	78	75	0.35
Dr Cipto	75	79	0.2
Dr Cipto	79	75	0.2
Urip Sumoharjo	74	70	0.17

Urip Sumoharjo	70	74	0.17
Pattimura	74	77	0.65
Pattimura	77	74	0.65
Hamid Rusdi	71	98	0.9
Hamid Rusdi	98	71	0.9
Hamid Rusdi	98	908	0.05
Hamid Rusdi	908	98	0.05
Hamid Rusdi	908	85	0.35
Hamid Rusdi	85	908	0.35
Urip Sumoharjo	70	97	0.65
Urip Sumoharjo	97	70	0.65
Mayjend Wiyono	97	909	0.3
Mayjend Wiyono	909	97	0.3
Mayjend Wiyono	909	87	0.35
Mayjend Wiyono	87	909	0.35
Kesatrian	97	85	0.5
Kesatrian	85	97	0.5
Polehan	87	96	1.6
Polehan	96	87	1.6
Muharto	67	96	0.5
Muharto	96	67	0.5
Muharto	96	95	0.5
Muharto	95	96	0.5
Ki Ageng Gribig	95	910	1.5
Ki Ageng Gribig	910	95	1.5
Ki Ageng Gribig	910	92	0.7
Ki Ageng Gribig	92	910	0.7
Danau Toba	911	91	0.75
Danau Toba	91	911	0.75
Danau Toba	91	92	0.35
Danau Toba	92	91	0.35
Ranu Grati	87	911	0.4
Ranu Grati	911	87	0.4
Raya Sawojajar	911	89	1.8
Raya Sawojajar	89	911	1.8
Terusan Sulfat	84	89	0.65
Terusan Sulfat	89	84	0.65
Raya Sulfat	82	84	1.3
Raya Sulfat	84	82	1.3
Tumenggung Suryo	71	99	0.6
Tumenggung Suryo	99	71	0.6
Letjend Sunandar Priyo	99	82	0.5

Sudarmo			
Letjend Sunandar Priyo Sudarmo	82	99	0.5
Letjend Sunandar Priyo Sudarmo	82	25	0.28
Letjend Sunandar Priyo Sudarmo	25	82	0.28
Letjend Sunandar Priyo Sudarmo	25	81	1.2
Letjend Sunandar Priyo Sudarmo	81	25	1.2
Ciliwung	902	25	0.7
Ciliwung	25	902	0.7
Laksamana Adi Sucipto	80	81	0.6
Laksamana Adi Sucipto	81	80	0.6
Jendral Ahmad Yani	13	80	0.1
Jendral Ahmad Yani	80	13	0.1
Yulius Usman	912	57	0.15
Yulius Usman	57	912	0.15
Syarif Al Qodri	57	113	0.22
Kapten Pierre Tandean	112	58	0.2
Kapten Pierre Tandean	58	112	0.2
Sutan Syahrir	54	112	0.22
Ade Irma Suryani	51	113	0.23
Ade Irma Suryani	113	51	0.23
Sulawesi	114	912	0.35
Sulawesi	912	114	0.35
Tanimbar	114	115	0.4
Tanimbar	115	114	0.4
Halmahera	112	115	0.4
Halmahera	115	66	0.5
Halmahera	66	115	0.5

Data Koordinat

id	latitude	longitude	is_pom
1	-7.94315	112.6103	0
2	-7.94983	112.6154	0
3	-7.94456	112.6194	0
5	-7.94783	112.6252	0
6	-7.95085	112.6316	0
7	-7.95293	112.6388	0
8	-7.94732	112.64	0
9	-7.94434	112.6345	0

10	-7.93946	112.6248	0
11	-7.93699	112.6263	0
12	-7.93907	112.6303	0
13	-7.94175	112.6421	0
14	-7.93312	112.6188	0
15	-7.93546	112.6173	0
16	-7.95963	112.632	0
17	-7.96181	112.6362	0
18	-7.96198	112.6304	0
19	-7.96445	112.6349	0
20	-7.94444	112.6318	0
21	-7.94166	112.6294	0
22	-7.94282	112.6291	0
23	-7.97428	112.6298	0
24	-7.96214	112.6258	0
25	-7.9547	112.6451	0
26	-7.95789	112.6234	0
27	-7.93831	112.607	0
28	-7.95994	112.6217	0
29	-7.95671	112.6128	0
30	-7.94715	112.6089	0
31	-7.94341	112.6039	0
32	-7.94749	112.6056	0
33	-7.95408	112.6068	0
34	-7.96352	112.6218	0
35	-7.96669	112.6194	0
36	-7.96856	112.6237	0
37	-7.97224	112.6216	0
38	-7.97582	112.6206	0
39	-7.9734	112.6156	0
40	-7.9733	112.613	0
41	-7.97254	112.617	0
42	-7.9691	112.618	0
43	-7.96699	112.6129	0
44	-7.97224	112.6066	0
45	-7.96574	112.6074	0
46	-7.97641	112.6293	0
47	-7.98092	112.6275	0
48	-7.98159	112.6303	0
49	-7.98352	112.6143	0
50	-7.98319	112.622	0
51	-7.98408	112.6262	0

52	-7.98247	112.6269	0
53	-7.98319	112.6298	0
54	-7.98523	112.6309	0
55	-7.98709	112.6357	0
56	-7.98515	112.6248	0
57	-7.98663	112.6285	0
58	-7.98813	112.6322	0
59	-7.98926	112.6349	0
60	-7.99496	112.62	0
61	-7.99385	112.6335	0
62	-7.98794	112.6223	0
63	-8.00042	112.6181	0
64	-8.00272	112.6274	0
65	-7.99953	112.6265	0
66	-7.99834	112.6273	0
67	-7.98889	112.6388	0
68	-7.983	112.637	0
69	-7.98091	112.6379	0
70	-7.97361	112.6385	0
71	-7.96716	112.6384	0
72	-7.98108	112.6319	0
73	-7.97718	112.6341	0
74	-7.97346	112.6366	0
75	-7.96936	112.6363	0
76	-7.9776	112.6368	0
77	-7.97073	112.6319	0
78	-7.96714	112.6335	0
79	-7.96922	112.6383	0
80	-7.94065	112.6427	0
81	-7.94372	112.6482	0
82	-7.95711	112.6441	0
84	-7.9631	112.6548	0
85	-7.97313	112.6463	0
87	-7.97908	112.6493	0
89	-7.96626	112.6597	0
91	-7.97918	112.6597	0
92	-7.98082	112.6626	0
95	-7.99397	112.6478	0
96	-7.9921	112.6432	0
97	-7.97742	112.6434	0
98	-7.9713	112.6438	0
99	-7.95985	112.6426	0

100	-7.9639	112.6135	0
112	-7.98738	112.6304	0
113	-7.98474	112.6288	0
114	-7.98788	112.6259	0
115	-7.99112	112.6288	0
901	-7.93764	112.6276	1
902	-7.95186	112.6392	1
903	-7.96195	112.6244	1
904	-7.95708	112.6131	1
905	-7.99498	112.62	1
906	-7.96795	112.6383	1
907	-7.98045	112.6373	1
908	-7.97137	112.6439	0
909	-7.97826	112.6453	1
910	-7.98404	112.6599	1
911	-7.98123	112.6542	1
912	-7.98606	112.6268	1