TUGAS AKHIR – SM 141501

PENENTUAN POLA JARINGAN PERGERAKAN LOGISTIK YANG OPTIMAL PADA TRANSPORTASI LAUT MENGGUNAKAN *MINIMUM SPANNING TREE* BERBASIS ALGORITMA GENETIKA

RIFDY FACHRY
NRP 1211 100 013

DOSEN PEMBIMBING
Dr. Imam Mukhlash, S.Si, MT
Drs. Soetrisno, MI.Komp.

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya
2015

FINAL PROJECT – SM 141501

DETERMINING THE OPTIMAL LOGISTICS NETWORK MOVEMENT PATTERN ON SEA TRANSPORTATION USING MINIMUM SPANNING TREE BASED ON GENETIC ALGORITHM

RIFDY FACHRY
NRP 1211 100 013

SUPERVISOR
Dr. Imam Mukhlash, S.Si, MT
Drs. Soetrisno, MI.Komp.

Departement of Mathematics
Faculty of Mathematics and Science
Institut Teknologi Sepuluh Nopember
Surabaya
2015

LEMBAR PENGESAHAN

PENENTUAN POLA JARINGAN PERGERAKAN LOGISTIK YANG OPTIMAL PADA TRANSPORTASI LAUT MENGGUNAKAN *MINIMUM SPANNING TREE* BERBASIS ALGORITMA GENETIKA

DETERMINING THE OPTIMAL LOGISTICS NETWORK MOVEMENT PATTERN ON SEA TRANSPORTATION USING MINIMUM SPANNING TREE BASED ON GENETIC ALGORITHM

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk memperoleh Gelar Sarjana Sains
Pada Bidang Studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
RIFDY FACHRY
NRP. 1211 100 013

Menyetujui,

Dosen Pembimbing II,

Drs.Soetrisno,MI.Komp.
NIP. 19571103 198603 1003

Dosen Pembimbing I,

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Mengetahui,
Ketua Jurusan Matematika
FMIPA ITS

Prof. Dr. Erna Apriliani, M.Si
NIP. 19660414 199102 2 001

Surabaya, Agustus 2015

PENENTUAN POLA JARINGAN PERGERAKAN LOGISTIK YANG OPTIMAL PADA TRANSPORTASI LAUT MENGGUNAKAN *MINIMUM SPANNING TREE* BERBASIS ALGORITMA GENETIKA

Nama Mahasiswa : Rifdy Fachry
NRP : 1211 100 013
Jurusan : Matematika
Dosen Pembimbing : Dr. Imam Mukhlash, S.Si, MT
Drs. Soetrisno, MI.Komp.

Abstrak

Penentuan pola jaringan pergerakan logistik yang optimal berguna untuk mendukung perencanaan tol laut. Salah satu parameter yang dapat digunakan untuk menentukan pola jaringan pergerakan logistik yang optimal adalah dengan menentukan jalur-jalur yang mempunyai kepadatan dalam pergerakan kontainer. Penentuan pola jaringan pergerakan logistik dapat dilakukan dengan menggunakan *Minimum Spanning Tree* (MST) berbasis algoritma genetika. Algoritma genetika adalah sebuah algoritma yang dapat digunakan dalam menyelesaikan permasalahan MST. Adapun tahapan dari penentuan pola jaringan pergerakan logistik yang optimal pada Tugas Akhir ini adalah penentuan node, proses *crossover,* proses mutasi, proses evaluasi, dan proses seleksi. Dalam penentuan node terdapat 52 node yang merepresentasikan pelabuhan. Proses *crossover* menggunakan *crossover rate* sebesar 0,2. Proses mutasi menggunakan *mutation rate* sebesar 0,4. Berdasarkan hasil pengujian sistem ini diperoleh total jalur terpadat dengan jumlah kontainer pada tiga tahun, yaitu 2010, 2011, dan 2012 berturut-turut adalah 1647896 Teu's, 1825049 Teu's, dan 2027860 Teu's dengan inisialisasi populasi 100 dan generasi maksimum 2000.

Kata Kunci : Algoritma Genetika, Logistik, *Minimum Spanning Tree*, Optimasi.

”Halaman ini sengaja dikosongkan”

DETERMINING THE OPTIMAL LOGISTICS NETWORK MOVEMENT PATTERN ON SEA TRANSPORTATION USING MINIMUM SPANNING TREE BASED ON GENETIC ALGORITHM

Name	***: Rifdy Fachry***
NRP	***: 1211 100 013***
Department	***: Mathematics***
Supervisor	***: Dr. Imam Mukhlash, S.Si, MT***
	Drs. Soetrisno, MI.Komp.

Abstract

Determining the pattern of optimal movement logistics network is very useful to support the planning of toll marine. One of the parameters that can be used to determine the pattern of movement logistic network is to determine the optimal paths that have a density in the movement of containers. Determining the pattern of movement of the logistics network can be done by using the Minimum Spanning Tree (MST) based on genetic algorithm. Genetic algorithm is an algorithm that can be used to solve the problems of MST. The steps of determining the pattern of optimal movement of logistic network in this final project is the determination node, the process of crossover, mutation process, evaluation process, and the selection process. In determining the nodes, there are 52 nodes that represent the number of ports. The process of crossover is using crossover rate of 0.2. The process of mutation is using the mutation rate of 0.4. Based on test, results obtained by the system is the lines with the most number of containers in three years, namely 2010, 2011, and 2012 respectively TEU's 1,647,896, 1,825,049 TEU's, and 2.02786 million TEU's with a large initial population of 100 and a maximum of 2000 generation.

Keyword : Genetic Algorithm, Logistics, Minimum Spanning Tree, Optimization.

”Halaman ini sengaja dikosongkan”

KATA PENGANTAR

Alhamdulillahirobbil'alamin. Puji syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga naskah Tugas Akhir yang berjudul "**Penentuan Pola Jaringan Pergerakan Logistik yang Optimal Pada Transportasi Laut Menggunakan *Minimum Spanning Tree* Berbasis Algoritma Genetika**" dapat diselesaikan dengan baik. Tulisan ini tidak akan terwujud dengan baik tanpa bantuan, dukungan dan dorongan dari semua pihak. Untuk itu penulis sangat berterima kasih kepada:

1. Dr. Imam Mukhlash, S.Si, MT, dan Drs. Soetrisno, MI.Komp., selaku Dosen Pembimbing yang telah memberikan pengarahan dan bimbingan selama proses penyusunan naskah Tugas Akhir ini.
2. Prof. Dr. Erna Apriliani, M.Si., selaku Ketua Jurusan Matematika FMIPA ITS atas fasilitas yang telah diberikan hingga naskah Tugas Akhir ini dapat terselesaikan.
3. Dr. Budi Setiyono, S.Si, MT, Drs. Nurul Hidayat, M.Kom, dan Kistosil Fahim, M.Si, selaku dosen penguji yang telah membantu penyelesaian perbaikan naskah Tugas Akhir.
4. Dr. Imam Mukhlash, S.Si MT, selaku Kepala Laboratorium Komputasi yang telah membantu secara administrasi dalam penyusunan Tugas Akhir ini.
5. Kedua orang tua, Mas Faidh, Nanin yang selalu memberi semangat, dukungan dan doa.
6. Alfiyatur Rohmah yang selalu memberikan waktunya dalam pengerjaan Tugas Akhir ini.
7. Teman-teman mahasiswa Jurusan Matematika 2011 yang selalu membantu, memberikan semangat, doa dan dukungannya.

Penulis menyadari sepenuhnya bahwa penulisan naskah Tugas Akhir ini tidak lepas dari kekurangan, oleh karena itu penulis terbuka terhadap kritik dan saran yang membangun. Semoga Tugas Akhir ini memberikan manfaat bagi penulis dan pembaca.

Surabaya, 03 Agustus 2015

Penulis

”Halaman ini sengaja dikosongkan”

DAFTAR ISI

JUDUL i
PENGESAHAN iii
ABSTRAK v
ABSTRACT vii
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR GAMBAR xiii
DAFTAR TABEL xv

BAB I PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 4
1.5 Manfaat 4
1.6 Sistematika Penulisan Tugas Akhir 4

BAB II TINJAUAN PUSTAKA 7
2.1 Graf 7
2.2 *Minimum Spanning Tree* 8
2.3 Algoritma Genetika 11
2.3.1 Memilih variabel dan fungsi cost 12
2.3.2 Inisialisasi Populasi 13
2.3.3 *Crossover* 13
2.3.4 Mutasi 14
2.3.5 Penelitian Sebelumnya 15
2.4 Pelabuhan Indonesia 16

BAB III METODE PENELITIAN 21

BAB IV PERANCANGAN DAN IMPLEMENTASI 25
4.1 Perumusan Minimum Spanning Tree Berbasis Algoritma Genetika 25

4.2 Analisis Sistem 26
4.2.1 Kebutuhan User 26
4.2.2 Model Analisis 26
4.3 Perancangan Sistem 28
4.3.1 Arsitektur Sistem 29
4.3.2 Proses Perancangan Database 29
4.3.3 Gambaran Sistem Secara Umum 32
4.3.4 Tahap Inisialisasi Parameter 34
4.3.5 Tahap Membangkitkan Kromosom 34
4.3.6 Tahap *Crossover* 35
4.3.7 Tahap *Mutation* 36
4.3.8 Tahap Evaluasi 37
4.3.9 Tahap Seleksi 39
4.3.10 Tahap Pembentukan Populasi Baru 40
4.4 Implementasi 40
4.4.1 Implementasi Program 40
4.4.2 Implementasi Antarmuka 5

BAB V PENGUJIAN DAN PEMBAHASAN HASIL 53
5.1 Pengujian Sistem dengan Data pada Tahun 2010 53
5.2 Pengujian Sistem dengan Data pada Tahun 2011 55
5.3 Pengujian Sistem dengan Data pada Tahun 2012 57
5.4 Hasil dari Pengujian Sistem dengan Data pada Tahun 2010 59
5.5 Hasil dari Pengujian Sistem dengan Data pada Tahun 2011 60
5.6 Hasil dari Pengujian Sistem dengan Data pada Tahun 2010 62

BAB VI PENUTUP 63
6.1 Kesimpulan 63
5.1 Saran 64

DAFTAR PUSTAKA 65

LAMPIRAN 67
A. *Source code class* Home.java 67
B. *Source code class* prufer.java 95
C. *Source code class* Populasi.java 99
D. *Source code class* PQ.java 101
E. *Source code class* Graph.java 103

”Halaman ini sengaja dikosongkan”

DAFTAR TABEL

Tabel 5.1 Pengujian pada Tahun 2010 53
Tabel 5.2 Nilai Fitness dari Pengujian dengan Data pada Tahun 2010 54
Tabel 5.3 Pengujian pada Tahun 2011 56
Tabel 5.4 Nilai Fitness dari Pengujian dengan Data pada Tahun 2011 56
Tabel 5.5 Pengujian pada Tahun 2012 58
Tabel 5.6 Nilai Fitness dari Pengujian dengan Data pada Tahun 2012 58

“Halaman ini sengaja dikosongkan”

DAFTAR GAMBAR

Gambar 2.1 Sebuah Graf 7
Gambar 2.2 Sebuah *Path* dari 1 ke 6; 1,2,3,5,6 8
Gambar 2.3 Sebuah *Cycle*; 1, 2, 5, 1 8
Gambar 2.4 Graf Berbobot 9
Gambar 2.5 Beberapa Contoh *Minimum Spanning Tree* 10
Gambar 2.6 *Flowchart* dari Algoritma Genetika 12
Gambar 2.7 Pelabuhan di Indonesia 1
Gambar 2.8 Alur Pelayaran di Indonesia.....19
Gambar 3.1 Diagram Alir Metode Penelitian 23
Gambar 4.1 *Use Case Diagram* Sistem.....27
Gambar 4.2 *Activity Diagram* Sistem 28
Gambar 4.3 Arsitektur Sistem Penentuan Pola Pergerakan Kontainer dengan Arus terpadat.....29
Gambar 4.4 CDM (*Conceptual Data Model*) pada Sistem.....30
Gambar 4.5 PDM (*Physical Data Model*) pada Sistem 31
Gambar 4.6 Diagram Alir Tahapan Utama Sistem secara Umum 33
Gambar 4.7 Representasi Gen dalam Kromosom 35
Gambar 4.8 Diagram Alir one-cut-point Crossover 36
Gambar 4.9 Diagram Alir Reversing Mutation 37
Gambar 4.10Diagram Alir Decode Prufer Number 39
Gambar 4.11Antar Muka Program 51
Gambar 5.1 Grafik Perbandingan Antar Pengujian 55
Gambar 5.2 Grafik Perbandingan Antar Pengujian 57
Gambar 5.3 Grafik Perbandingan Antar Pengujian 59
Gambar 5.4 Model Graf Tree pada Data Tahun 2010 60
Gambar 5.5 Model Graf Tree pada Data Tahun 2011 61
Gambar 5.6 Model Graf Tree pada Data Tahun 2012 63

”Halaman ini sengaja dikosongkan”

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Indonesia merupakan negara maritim, dimana wilayahnya sebagian besar berupa laut. Aktifitas-aktifitas yang dilakukan antar pulau tidak dapat meninggalkan peran dari laut tersebut. Karena itu diperlukan transportasi laut yang mendukung agar aktifitas-aktifitas yang diperlukan oleh penduduk Indonesia dapat berkembang. Ada beberapa jenis kapal yang berlayar di perairan Indonesia sesuai dengan fungsinya. Salah satunya yaitu kapal pengangkut barang. Sistem pergerakan barang pada jalur laut masih buruk. Sebagai contoh, yaitu buruknya sistem transportasi laut menyebabkan biaya angkut dari Jakarta ke Padang 3,5 kali lipat lebih mahal daripada biaya angkut dari Jakarta ke Singapura. Saat ini, biaya Jakarta – Padang sebesar 600 US Dollar, sedangkan Jakarta – Singapura hanya sebesar 185 US Dollar [1].

Selain itu juga Indonesia merupakan negara dengan rata-rata biaya logistik yang lebih tinggi dibandingkan dengan beberapa negara lain di kawasan. Permasalahan tingginya biaya logistik di Indonesia, yang nilainya mencapai sekitar 24% dari PDB Indonesia, menunjukkan bahwa kinerja sistem logistik nasional yang sangat buruk [2]. Saat ini Indonesia merencanakan tentang konsep tol laut. Tol laut merupakan rute transportasi laut mulai dari Sumatera sampai Papua dengan melewati semua pelabuhan utama di Indonesia. Jalur laut mampu menjadi alternatif di tengah tingginya beban pengangkutan yang selama ini bertumpu pada jalur darat maupun jalur kereta api, contohnya dalam kasus untuk mengurangi beban jalur Pantai Utara (Pantura) [3]. Karena

banyaknya transaksi di wilayah laut, terdapat pola-pola rute yang bisa dilewati untuk distribusi logistik. Maka dari itu dibutuhkan suatu analisa untuk menentukan pola yang optimal sesuai dengan kriteria yang dibutuhkan. Dalam analisa tersebut terdapat beberapa metode untuk menentukan pola rute yang optimal untuk pergerakan logistik yang melewati jalur laut wilayah Indonesia Dari jalur-jalur tersebut dapat dibentuk suatu model graf dimana salah satu metodenya adalah dengan menggunakan metode *Minimum Spanning Tree* berbasis algoritma genetika.

Minimum Spanning Tree atau pohon rentang minimum adalah model graf dengan jumlah bobot yang terkecil. Dalam Tugas Akhir ini *Minimum Spanning Tree* diaplikasikan pada penentuan pola jalur pergerakan logistik. Terdapat beberapa pendekatan pada MST, yaitu dengan menggunakan algoritma Prim, algoritma Kruskal, dll [9]. Salah satu pendekatan pada MST adalah dengan menggunakan algoritma genetika. Dengan menggunakan algoritma genetika dalam menentukan jalur-jalur pergerakan logistik di wilayah Indonesia, akan diketahui pola jalur dengan arus pergerakan kontainer yang terpadat.

Dari penelitian-penelitian sebelumnya, yang dimuat dalam paper dengan judul "Nonlinear Fixed Charge Transportation Problem by Spanning Tree-based Genetic Algorithm" oleh Jung-Bok Jo, Yinzhen Li, dan Mitsuo Gen didapat bahwa solusi yang optimal untuk masalah biaya transportasi nonlinear telah didapat dengan menggunakan pendekatan Spanning Tree-based Genetic Algorithm [4]. Dan pada paper yang berjudul "Study on Multi-Stage Logistic Chain Network: A Spanning Tree-based Genetic Algorithm Approach" oleh Admi Syarif, YoungSu Yun, dan Mitsuo Gen didapat bahwa st-Genetic Algorithm dapat menemukan desain

produksi/ distribusi pada sistem pelevelan logistik. Hasil eksperimen yang telah dilakukan menunjukkan bahwa algoritma ini tidak hanya dapat memberikan solusi heuristik yang lebih baik, tetapi juga mempunyai performa yang lebih baik pada kecepatan waktu komputasi dan kebutuhan memori pada komputer daripada m-Genetic Algorithm. Algoritma ini efisien untuk menyelesaikan permasalahan seperti masalah desain rantai logistik bertingkat [5].

Berdasarkan permasalahan diatas, penulis mengangkat topik tersebut sebagai Tugas Akhir dengan judul "Penentuan Pola Jaringan Pergerakan Logistik pada Transportasi Laut menggunakan *Minimum Spanning Tree* Berbasis Algoritma Genetika".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka permasalahan yang akan diselesaikan dalam Tugas Akhir ini meliputi:

1. Bagaimana merancang simulasi jaringan pergerakan logistik pada transportasi laut yang optimal?
2. Bagaimana menentukan pola jaringan pergerakan logistik pada transportasi laut yang optimal?

1.3 Batasan Masalah

Agar tidak terjadi kesalahan persepsi dan tidak meluasnya pokok bahasan Tugas Akhir ini maka diberikan batasan-batasan sebagai berikut:

1. Pergerakan kontainer pada transportasi laut di wilayah Indonesia.
2. Tidak mmperhatikan kandungan kontainer.
3. Tidak memperhatikan arah pergerakan kontainer.

1.4 Tujuan

Tujuan dari Tugas Akhir yang diusulkan ini adalah sebagai berikut:

1. Merancang simulasi jaringan pergerakan logistik pada transportasi laut yang optimal menggunakan MST berbasis algoritma genetika
2. Mendapatkan pola jaringan pergerakan logistik pada transportasi laut yang optimal menggunakan MST berbasis algoritma genetika

1.5 Manfaat

Tugas akhir ini menghasilkan pola jaringan pergerakan logistik pada transportasi laut yang optimal. Dengan didapatkannya pola optimal ini diharapkan dapat menjadi rekomendasi dalam penentuan rute tol laut.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika penulisan didalam Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pembuatan Tugas Akhir, rumusan dan batasan permasalahan yang dihadapi dalam penelitian Tugas Akhir, tujuan dan manfaat pembuatan Tugas Akhir dan sistematika penulisan Tugas Akhir.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka dari referensi penunjang serta penjelasan permasalahan yang dibahas dalam Tugas Akhir ini, meliputi pengertian Graf, *Minimum Spanning Tree*, Algoritma Genetika, dan pelabuhan di Indonesia.

BAB IIIMETODOLOGI PENELITIAN

Bab ini berisi metodologi atau urutan pengerjaan yang dilakukan dalam menyelesaikan Tugas Akhir, meliputi studi literatur, pengumpulan data, analisa dan desain sistem, pembuatan program, uji coba dan evaluasi, hingga penulisan Tugas Akhir.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan mengenai perancangan dan implementasi sistem, proses pembuatan sistem secara utuh sehingga dapat digunakan untuk menentukan pola pergerakan logistik pada transportasi laut yang optimal.

BAB V PENGUJIAN DAN PEMBAHASAN HASIL

Bab ini akan menampilkan hasil uji coba serta pembahasan terkait sistem penentuan pola pergerakan logistik pada transportasi laut yang optimal yang telah dibuat.

BAB VI PENUTUP

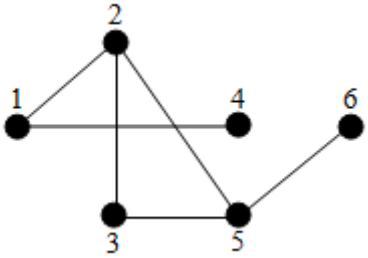
Bab ini merupakan penutup, berisi tentang kesimpulan yang dapat diambil berdasarkan data yang ada dan saran yang selayaknya dilakukan bila Tugas Akhir ini dilanjutkan.

“Halaman Sengaja dikosongkan”

BAB II
TINJAUAN PUSTAKA

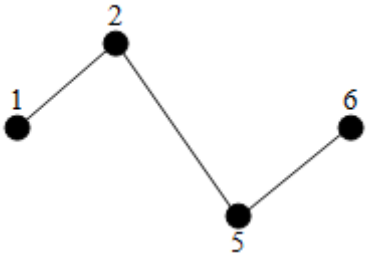
2.1 Graf

Sebuah graf mengandung sekumpulan elemen $\mathcal{V}$ yang disebut verteks (*nodes*) dan sekumpulan $\mathcal{A}$ yang merupakan sepasang verteks yang terdapat dalam $\mathcal{V}$ yang disebut *edges* (*arcs*). Graf digambarkan sebagai sistem grafik dengan gambar lingkaran untuk verteks dan gambar garis antara verteks i dan j yang mana dapat ditulis $\{i, j\}$ adalah sebuah *edges*[7]. Misalnya, graf dengan $\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$ dan $\mathcal{A} = \{\{1,2\}, \{1,4\}, \{2,5\}, \{2,3\}, \{3,5\}, \{5,6\}\}$ digambarkan pada Gambar 2.1.



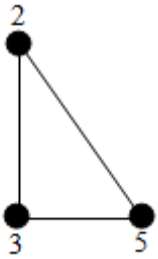
Gambar 2.1 Sebuah Graf

Sebuah urutan verteks $i, i_1, i_2, \ldots, i_k, j$ untuk $\{i, i_1\}, \{i_1, i_2\}, \ldots, \{i_{k-1}, i_k\}, \{i_k, j\}$ adalah semua *edges* disebut *path* dari verteks i ke verteks j [7]. Pada Gambar 2.2 ditunjukkan sebuah *path* dari verteks 1 ke verteks 6.



Gambar 2.2 Sebuah *path* dari 1 ke 6: 1, 2, 5, 6

Sebuah *path* $i, i_1, i_2, \dots, i_k, i$ dari sebuah verteks kembali ke dirinya sendiri pada setiap *edge* $\{i, i_1\}, \{i_1, i_2\}, \dots, \{i_{k-1}, i_k\}, \{i_k, i\}$ yang berbeda disebut *cycle* [7]. Pada Gambar 2.3 ditunjukkan sebuah *cycle*.

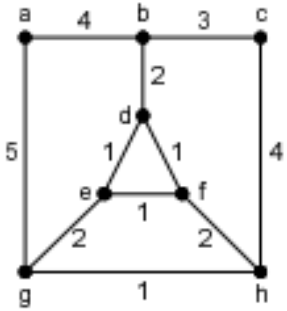


Gambar 2.3 Sebuah *cycle*: 2, 3, 5, 2

2.2 Minimum Spanning Tree

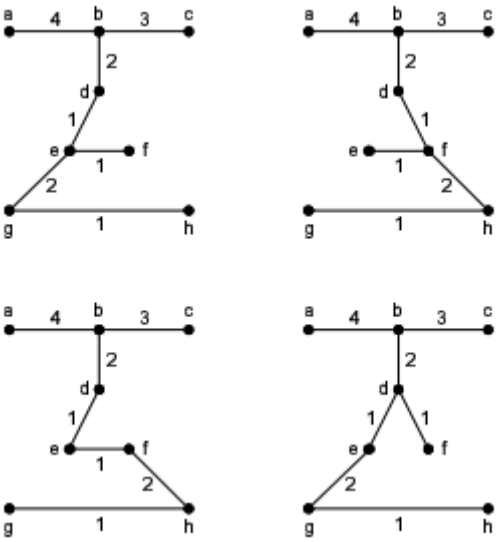
Minimum Spanning Tree dari graf G yang berbobot adalah *spanning tree* dari G yang mempunyai jumlah bobot yang minimal. Dengan kata lain, *minimum spanning tree* adalah sebuah bentuk *tree* pada graf dengan dua sifat: (1) merentang pada graf, artinya semua vertex dalam graf terhubung, dan (2) minimal, artinya jumlah bobot dari semua *edges* sekecil mungkin.

Minimum Spanning Tree dapat diaplikasikan di beberapa bidang. Dalam *Minimum Spanning Tree* terdapat algoritma *greedy* yang dapat menghasilkan solusi optimal, dan dibutuhkan struktur data yang baik untuk menjadikan *Minimum Spanning Tree* bekerja secara efisien [8]. Untuk graf berbobot dapat ditunjukkan pada Gambar 2.4.



Gambar 2.4 Graf Berbobot [8].

Dari Gambar 2.4 diketahui graf berbobot dengan verteks a, b, c, d, e, f, g, h dan edges {a, b}, {a, g}, {b, c}, {b, d}, {c, a}, {d, e}, {d, f}, {e, f}, {e, g}, {f, h}, {g, h}, dengan bobot yang ada pada Gambar 2.4. Kemudian dicari *edges* dengan bobot yang terkecil dan tidak membentuk *cycle*. Untuk hasil pada Gambar 2.4 dapat ditunjukkan pada Gambar 2.5.



Gambar 2.5 Beberapa contoh *minimum spanning tree* dari graf pada Gambar 2.4[8].

Pada *Minimum Spanning Tree* terdapat 2 algoritma yaitu Algoritma Prim, Algoritma Kruskal. Dalam Algoritma Prim dimisalkan T adalah pohon merentang yang sisi-sisinya diambil dari graf G. Algoritma Prim membentuk pohon merentang minimum langkah per langkah. Pada setiap langkah diambil sisi e dari graf G yang mempunyai bobot minimum dan bersisian dengan simpul-simpul di dalam T, tetapi e tidak membentuk sirkuit di dalam T. Sedangkan Algoritma Kruskal, sisi-sisi di dalam graf diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar. Sisi yang dimasukkan ke dalam himpunan T adalah sisi graf G sedemikian sehingga T adalah pohon. Pada keadaan awal, sisi-sisi sudah diurut berdasarkan

bobot membentuk hutan (*forest*), masing-masing pohon di hutan hanya berupa satu buah simpul. Hutan tersebut dinamakan hutan merentang (*spanning forest*). Sisi dari graf G ditambahkan ke T jika ia tidak membentuk *cycle* di T [9]. Dalam MST juga dapat menggunakan algoritma genetika, karena algoritma genetika dapat mencari nilai *fitness* yang terbaik dari kombinasi gen-gen. Sehingga MST dapat diselesaikan dengan menggunakan algoritma genetika.

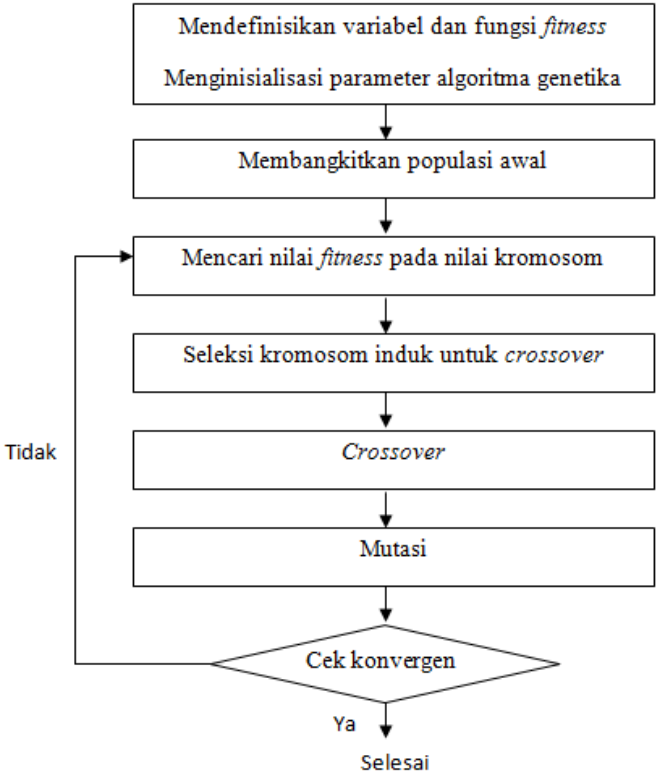
2.3 Algoritma Genetika

Algoritma genetika adalah teknik optimasi dan pencarian yang berdasarkan pada prinsip gen dan seleksi alam. Algoritma genetika memberikan susunan populasi dari banyak individu untuk mengembangkan aturan seleksi yang spesifik untuk sebuah pernyataan memaksimalkan "fitness".

Beberapa keuntungan dari algoritma genetika adalah:

- Mengoptimasi variabel kontinu atau diskrit.
- Tidak memerlukan informasi turunan.
- Simulasi pencarian dari sampel yang lebar pada lingkup biaya.
- Transaksi dengan variabel berjumlah besar.
- Sangat cocok untuk komputer paralel.
- Mengoptimasi variabel dengan lingkup biaya yang benar-benar kompleks.
- Menyediakan daftar dari variabel optimal, tidak hanya satu solusi.
- Dapat melakukan *encode* variabel-variabel sehingga optimasi dapat selesai dengan variabel *encode* .
- Bekerja dengan data yang diolah secara numerik, data eksperimen, atau fungsi analitik [10].

Algoritma genetika dimulai dengan mendefinisikan variabel optimasi, fungsi *cost*, dan *cost*, dan diakhiri dengan uji konvergensi. *Flowchart* algoritma genetika dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Flowchart* dari algoritma genetika.

2.3.1 Memilih variabel dan fungsi *cost*

Fungsi *cost* menghitung nilai *output* berdasarakan nilai dari sekumpulan variabel *input* (kromosom). Algoritma genetika dimulai dengan mendefinisikan sebuah kromosom

atau *array* dari nilai variabel untuk dioptimasi. Jika kromosom mempunyai N_{var} variabel (dimana N_{var} adalah dimensi masalah optimasi) dan diberikan $p_1, p_2, \dots, p_{N_{var}}$, maka kromosom dapat ditulis sebagai $1 \times N_{var}$ elemen vektor baris.

$$chromosome = [p_1, p_2, p_3 \dots, p_{N_{var}}]$$

Setiap kromosom mempunyai *cost* dengan mengevaluasi fungsi *cost* f, pada $p_1, p_2, \dots, p_{N_{var}}$:

$$cost = f(chromosome) = f(p_1, p_2, \dots, p_{N_{var}})$$

2.3.2 Inisialisasi populasi

Dalam memulai algoritma genetika, didefinisikan sebuah inisialisasi populasi pada N_{pop} kromosom. Sebuah matriks merepresentasikan populasi dengan setiap baris dalam matriks menjadi $1 \times N_{var}$ *array* (kromosom) pada nilai kontinu. Diberikan sebuah inisialisasi populasi pada N_{pop} kromosom, matriks penuh pada $N_{pop} \times N_{var}$.

2.3.3 *Crossover*

Banyak perbedaan pendekatan yang telah dicoba untuk melakukan *crossover* pada algorima genetika kontinu. Salah satunya yaitu dengan menyeleksi secara acak sebuah variabel dalam pasangan pertama pada *parent* untuk menjadi titik *crossover*.

$$\alpha = roundup\{random * N_{var}\}$$

Kemudian

$$parent_1 = [p_{m1} p_{m2} \dots p_{m\alpha} \dots p_{mN_{var}}]$$
$$parent_2 = [p_{d1} p_{d2} \dots p_{d\alpha} \dots p_{dN_{var}}]$$

dimana m dan d adalah pembeda antara *mom* dan *dad.* Kemudian variabel yang dipilih dikombinasi ke bentuk variabel baru yang akan muncul pada *children*:

$$p_{new1} = p_{d\alpha}$$
$$p_{new2} = p_{m\alpha}$$

Step terakhir adalah untuk melengkapi *crossover* dengan sisa dari kromosom sebelumnya:

$$offspring_1 = [p_{m1}p_{m2} \dots p_{new1} \dots p_{dN_{var}}]$$
$$offspring_2 = [p_{d1}p_{d2} \dots p_{new2} \dots p_{mN_{var}}]$$

Jika variabel pertama pada kromosom dipilih, maka hanya variabel yang kanan pada pemilihan variabel ditukar. Jika variabel terakhir pada kromosom dipilih, maka hanya variabel yang kiri pada pemilihan variabel ditukar [10].

2.3.4 Mutasi

Setelah melakukan *crossover* maka langkah selanjutnya adalah mutasi. Mutasi berfungsi untuk memulihkan gen yang hilang. Mutasi secara tradisional dianggap sebagai operator pencarian sederhana. Jika *crossover* seharusnya membuat solusi untuk menemukan hasil yang lebih baik, mutasi membantu untuk eksplorasi seluruh pencarian. Mutasi dipandang sebagai operator untuk mempertahankan keragaman genetik dalam populasi. Hal ini akan memperkenalkan struktur genetik baru dalam populasi secara acak dan memodifikasi beberapa blok. Mutasi dapat mempertahankan keragaman dalam populasi.

Ada berbagai bentuk mutasi di berbagai jenis representasi. Untuk representasi biner, mutasi sederhana terdiri dari pembalik nilai masing-masing gen dengan probabilitas kecil. Probabilitas biasanya diambil sekitar 1 / L, dimana L

adalah panjang kromosom. Operator tersebut dapat mempercepat pencarian [11].

2.3.5 Penelitian sebelumnya

Terdapat beberapa penelitian yang telah dilakukan tentang penggunaan *Minimum Spanning Tree* berbasis algoritma genetika.

1. Pada tahun 2002 oleh Admi Syarif, YongSu Yun, dan Mitsuo Gen dengan judul "Study on Multi-Stage Logistic Chain Network: A Spanning Tree-based Genetic Algorithm Approach" didapat bahwa st-Genetic Algorithm dapat menemukan desain produksi/distribusi pada sistem pelevelan logistik. Hasil eksperimen yang telah dilakukan menunjukkan bahwa algoritma ini tidak hanya dapat memberikan solusi heuristik yang lebih baik, tetapi juga mempunyai performa yang lebih baik pada waktu komputasi dan kebutuhan memori pada komputer daripada m-Genetic Algorithm. Algoritma ini efisien untuk menyelesaikan permasalahan seperti masalah desain rantai logistik bertingkat [5].
2. Pada tahun 2007 oleh Jung-Bok Jo, Yinzhen Li, dan Mitsuo Gen dengan judul "Nonlinear Fixed Charge Transportation Problem by Spanning Tree-based Genetic Algorithm" didapat bahwa solusi yang optimal untuk masalah biaya transportasi nonlinear telah didapat dengan menggunakan pendekatan *Spanning Tree-based Genetic Algorithm*. Selama algoritma tidak menyerap karakteristik dari struktur nonlinear, maka *Spanning Tree-based Genetic Algorithm* harus diimprovisasi selama digunakan untuk pemecahan masalah biaya transportasi nonlinear [4].

2.4 Pelabuhan Indonesia

Pelabuhan adalah tempat yang terdiri atas daratan dan/atau perairan dengan batas-batas tertentu sebagai tempat kegiatan pemerintahan dan kegiatan pengusahaan yang dipergunakan sebagai tempat kapal bersandar, naik turun penumpang, dan/atau bongkar muat barang, berupa terminal dan tempat berlabuh kapal yang dilengkapi dengan fasilitas keselamatan dan keamanan pelayaran dan kegiatan penunjang pelabuhan serta sebagai tempat perpindahan intra-dan antarmoda transportasi. Pada Gambar 2.7 menunjukkan tentang peta pelabuhan di Indonesia. Pada Gambar 2.8 menunjukkan alur pelayaran di Indonesia.

Di Indonesia terdapat 5 jenis pelabuhan, yaitu:

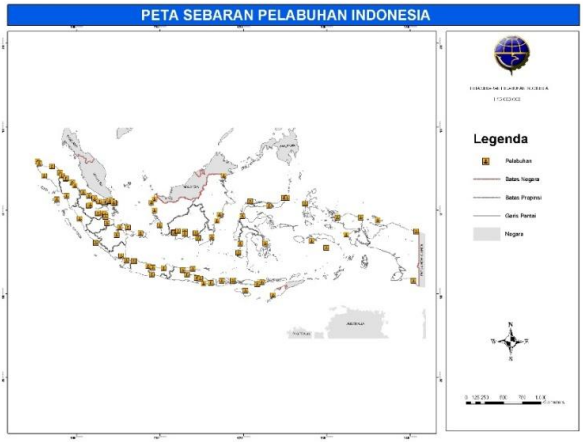
1. Pelabuhan Internasional HUB, adalah pelabuhan utama primer dengan memperhatikan: kedekatan dengan pasar internasional, kedekatan dengan jalur pelayaran internasional, kedekatan dengan jalur Alur Laut Kepulauan Indonesia, berperan sebagai tempat alih muat penumpang dan barang internasional, memiliki jarak tertentu dengan pelabuhan internasional hub lainnya, memiliki kondisi teknis pelabuhan yang terlindung dari gelombang dengan luas daratan dan perairan tertentu, volume kegiatan bongkar muat [12]. Di Indonesia terdapat 16 prasarana, seperti: Tg. Perak, Tg. Priok, Palembang, Samarinda, Balikpapan, dll [14].
2. Pelabuhan Internasional, adalah pelabuhan utama sekunder dengan memperhatikan: kedekatan dengan jalur pelayaran nasional dan internasional, sebagai tempat alih muat penumpang dan barang nasional, mempunyai jarak tertentu dengan pelabuhan

internasional lainnya, memiliki kondisi teknis pelabuhan yang terlindung dari gelombang dengan luas daratan dan perairan tertentu, volume kegiatan bongkar muat [12]. Di Indonesia terdapat 35 prasarana, seperti: Belawan, Sei Pakning, Teluk Bayur, Panjang, Benoa, dll [14].

3. Pelabuhan Nasional, adalah pelabuhan utama tersier dengan memperhatikan: kebijakan Pemerintah yang meliputi pemerataan pembangunan nasional dan meningkatkan pertumbuhan wilayah, sebagai tempat alih muat penumpang dan barang nasional dan bias menangani semi container, mempunyai jarak tertentu dengan pelabuhan nasional lainnya, mempunyai jarak tertentu terhadap jalur/rute lintas pelayaran nasioanal, memiliki kondisi teknis pelabuhan yang terlindung dari gelombang dengan luas daratan dan perairan tertentu, kedekatan dengan jalur/lalu lintas pelayaran antar pulau, berada (dekat) dengan pusat pertumbuhan wilayah ibu kota Kabupaten/Kota dan kawasan pertumbuhan nasional, volume kegiatan bongkar muat [12]. Di Indonesia terdapat 45 prasarana, seperti: Sinabang, Tual, Fak-fak, Sampit, Ambon, dll [14].
4. Pelabuhan Regional, adalah pelabuhan pengumpan primer dengan memperhatikan: kebijakan Pemerintah yang menunjang pusat pertumbuhan ekonomi, propinsi dan pemerataan pembangunan antar propinsi, berfungsi sebagai tempat pelayanan penumpang dan barang inter Kabupaten/Kota, memiliki jarak tertentu dengan pelabuhan regional lainnya, memiliki kondisi teknis pelabuhan yang

terlindung dari gelombang denga luas daratan dan perairan tertentu, volume kegiatan bongkar muat [12]. Di Indonesia terdapat 31 prasarana, seperti: Batang, Rembang, Siwa, Matui, Jailolo, dll [14].

5. Pelabuhan Lokal, pelabuhan pengumpan sekunder dengan memperhatikan kebijakan Pemerintah untuk menunjang pusat pertumbuhan ekonomi, Kabupaten/Kota dan pemerataan serta meningkatkan pembangu nan Kabupaten/Kota, berfungsi ntuk melayani penumpang dan barang antar Kecamatan dalam Kabupaten/Kota terhadap kebutuhan moda transportasi laut dan/atau perairan, memiliki kondisi teknis pelabuhan yang terlindung dari gelombang dengan luas daratan dan perairan tertentu, volume kegiatan bongkar muat [12]. Di Indonesia terdapat 132 prasarana, seperti: Sikakap, Pulau Kijag, Labuhan Alas, Pangkalan Bun, Cenrana, dll [14].



Gambar 2.7 Peta Pelabuhan di Indonesia [13]

Gambar 2.8 Alur Pelayaran di Indonesia [14]

”Halaman ini sengaja dikosongkan”

BAB III
METODE PENELITIAN

Penyusunan Tugas Akhir ini dilakukan dengan menggunakan beberapa metode yang dapat mendukung penulis dari awal hingga akhir. Adapun metode penelitian yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

a. Studi literatur

   Pada tahap ini dilakukan pengumpulan data dan informasi mengenai pelabuhan-pelabuhan di Indonesia, rute-rute yang dilewati kapal pengangkut kontainer, dan jumlah kontainer yang dibawa antar pelabuhan. Data dan informasi tersebut diambil dari Direktorat Jendral Perhubungan Laut dan *Google Map*. Pada tahap ini dilakukan penelitian mengenai algoritma genetika pada *Minimum Spanning Tree* yang diperlukan sebagai metode dalam menyelesaikan permasalahan pada Tugas Akhir ini dengan cara mempelajari literatur-literatur ilmiah yang memiliki hubungan dengan topik penelitian yang sedang penulis lakukan guna mendapat data teoritis sehubungan dengan pembahasan penelitian ini.

b. Perumusan *Minimum Spanning Tree* berbasis Algoritma Genetika

   Dalam tahap ini dilakukan perumusan *Minimum Spanning Tree* yang berbasis Algoritma Genetika. Dalam perumusan ini dilakukan penentuan *nodes* dan penentuan bobot tiap verteks. Bobot tersebut diperoleh dari jumlah kontainer yang dibawa dari pelabuhan asal ke pelabuhan tujuan dengan data pada tahun 2010, 2011, dan 2012. Bobot tersebut menggunakan satuan Teus.

c. Analisis data input

Pada tahap ini dilakukan analisis mengenai algoritma genetika pada *Minimum Spanning Tree* yang digunakan dalam penyelesaian masalah pada Tugas Akhir ini, dan juga analisis data yang diperoleh. Kemudian dibuat desain sistem dari program sesuai dengan hasil analisis. Sistem ini memiliki input berupa data pelabuhan, dan data jumlah kontainer yang dibawa antar pelabuhan .

d. Perancangan perangkat lunak

Pada tahap ini sistem yang telah dirancang diimplementasikan kedalam bentuk perangkat lunak menggunakan bahasa pemrograman Java dengan *compiler* Netbeans.

e. Pengujian dan evaluasi perangkat lunak

Pada tahap ini dilakukan pengujian perangkat lunak apakah berjalan dengan benar dan sesuai dengan perancangan. Dalam pengujian ini terdapat proses pengolahan data yang ada sehingga menghasilkan suatu nilai. Dari nilai tersebut akan digunakan dalam algoritma genetika untuk menentukan rute pergerakan kontainer terpadat. Selanjutnya dilakukan evaluasi perangkat lunak. Apabila masih terdapat error, maka dilakukan perbaikan pada perangkat lunak tersebut.

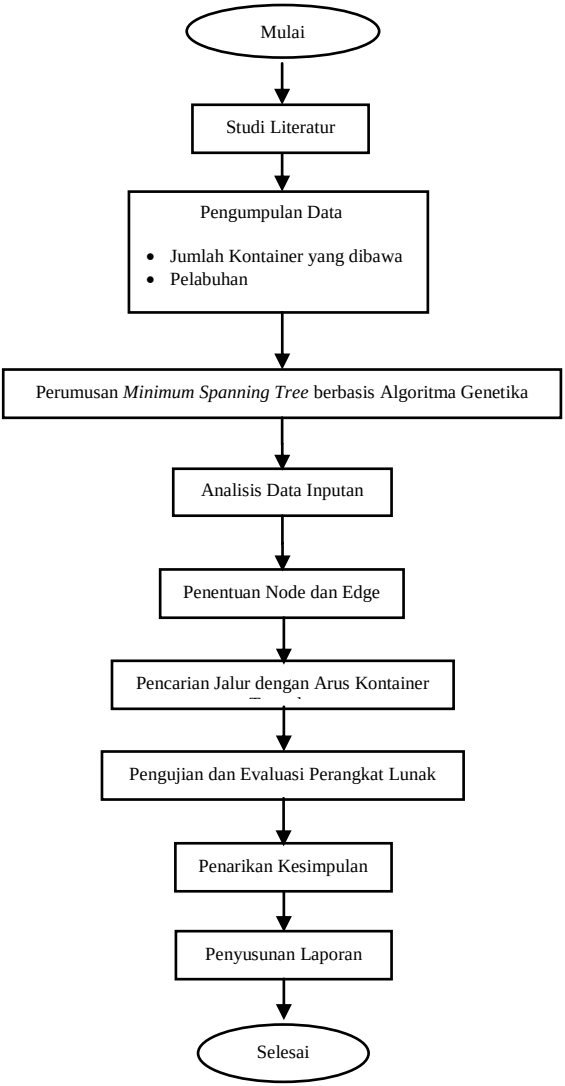
f. Penarikan kesimpulan

Pada tahap ini dilakukan penarikan kesimpulan terhadap dari hasil pengerjaan, uji coba, dan pembahasan dari pengerjaan Tugas Akhir ini serta disampaikan saran untuk pengembangan berikutnya.

g. Penyusunan laporan

Bagian terakhir dalam Tugas Akhir ini adalah membuat laporan seluruh tahapan/proses yang sudah dilakukan

Diagram alir metode penelitian dapat ditunjukkan pada Gambar 3.1



Gambar 3.1 Diagram alir metode penelitian

“Halaman Sengaja dikosongkan”

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Pada bab ini menjelaskan mengenai perancangan sistem dan hasil implementasi seluruh proses yang telah dirancang sebelumnya. Pembahasan perancangan sistem diawali dengan perumusan *Minimum Spanning Tree* berbasis algoritma genetika, analisis sistem dan perancangan sistem beserta proses-proses yang ada dalam Tugas Akhir ini. Selanjutnya membahas implementasi sistem yang dimulai dari implementasi program dan dilanjutkan dengan hasil implementasi antarmuka dan keseluruhan proses di dalam sistem.

4.1 Perumusan *Mininum Spanning Tree* Berbasis Algoritma Genetika

Dalam perumusan ini terdapat variabel yang digunakan untuk menentukan pengambilan hasil. Variabel tersebut adalah jumlah kontainer yang dibongkar muat antar pelabuhan pada tiap tahun. Dalam variabel ini juga terdapat indeks i yang merepresentasikan pelabuhan asal, dan j merepresentasikan pelabuhan tujuan, sehingga dapat dirumuskan sebagai berikut:

$$\mathbf{min}(\frac{\mathbf{1}}{\sum_i \sum_j \boldsymbol{x_{ij}}})$$

x: jumlah kontainer yang dibongkar muat
i: pelabuhan asal
j: pelabuhan tujuan

Dari rumus tersebut dapat dicari nilai yang minimal. Dalam penentuan jalur pergerakan kontainer terpadat dapat dicari dengan mendapatkan nilai dari rumus berikut:

$$\mathbf{max}(\ f) = \mathbf{min}\ (\frac{\mathbf{1}}{\sum_i \sum_j \boldsymbol{x_{ij}}})$$

f: total jumlah kontainer yang dibongkar rmuat
x: jumlah kontainer yang dibongkar muat
i: pelabuhan asal
j: pelabuhan tujuan

4.2 Analisis Sistem

Dalam sistem penentuan pola jaringan pergerakan logistik terdapat beberapa analisis, yaitu kebutuhan user, model analisis, seperti *use case diagram, activity diagram,* dan CDM (*Conceptual Data Model*).

4.2.1 Kebutuhan *User*

Dalam sistem yang dirancang terdapat kebutuhan *user* sehingga sistem yang dirancang sesuai dengan kebutuhan *user.* Dalam sistem ini kebutuhan *user* nya adalah:

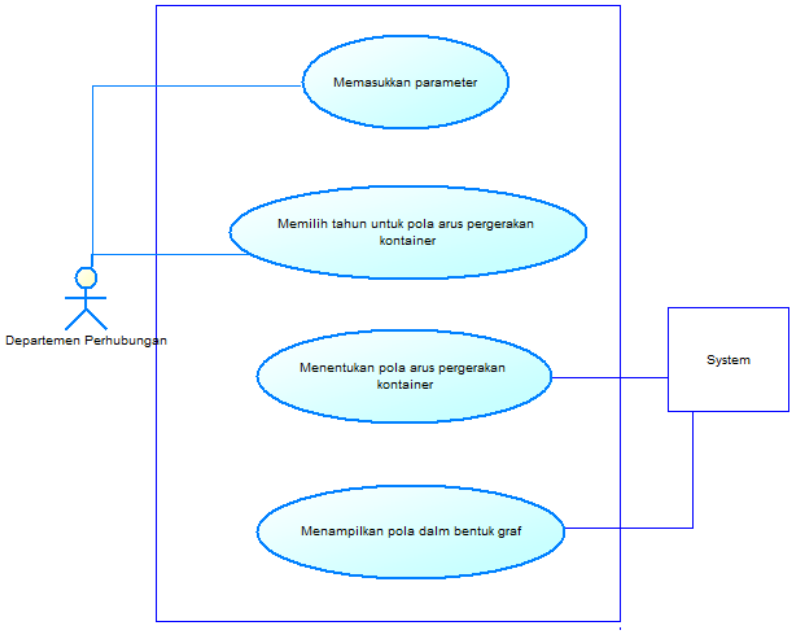
1. Dapat menentukan pola jaringan pergerakan kontainer dengan arus terpadat pada tahun 2010, 2011, dan 2012.
2. Dapat memberikan hasil jumlah arus terpadat pada tahun 2010, 2011,dan 2012.

4.2.2 Model Analisis

Dalam sistem ini diberikan suatu model análisis, sehingga dapat memberikan gambaran yang jelas terhadap sistem yang dirancang ini. Untuk model análisis sistem terdapat 2 model, yaitu *use case diagram* dan *activity diagram.*

4.2.2.1 *Use Case Diagram*

Use case diagram adalah diagram yang dibuat untuk mendeskripsikan sebuah interaksi antara satu atau lebih *user* dengan sistem yang dibuat. Dalam sistem ini *user* adalah Departemen Perhubungan. Untuk menggambarkan *use case diagram* dapat dilihat pada Gambar 4.1.

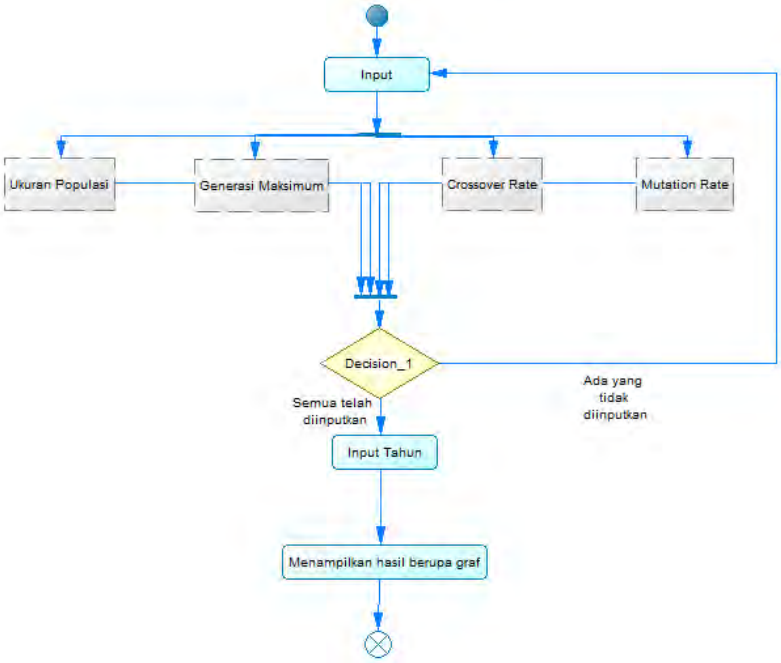


Gambar 4.1 *Use case diagram* sistem

Pada Gambar 4.1, Departemen Perhubungan sebagai *user* dapat melakukan inisialisasi parameter, dan memilih tahun untuk pola arus pergerakan kontainer. Sistem dapat menentukan pola pergerakan kontainer dengan arus terpadat dan menampilkannya dalam bentuk graf.

4.2.2.2 *Activity Diagram*

Activity diagram adalah diagram untuk menjelaskan alur aktifitas dalam sistem yang dirancang, termasuk awal dari sistem, *decisión* yang ada, dan akhir dari sistem ini. Untuk menggambarkan *activity diagram* pada sistem ini dapat dilihat pada Gambar 4.2.



Gambar 4.2 *Activity diagram* sistem

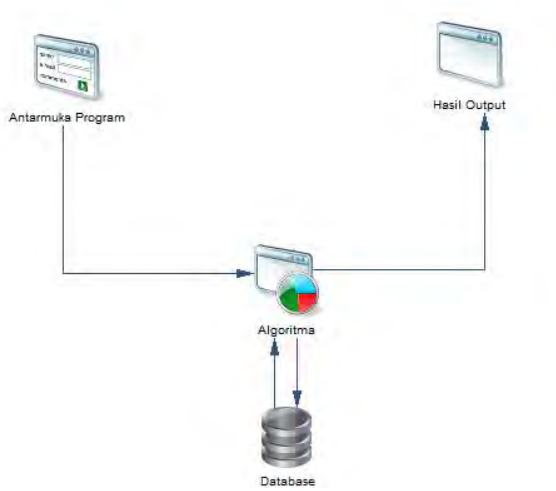
Pada Gambar 4.2, sistem mulai dengan memasukkan ukuran populasi, generasi maksimum, *crossover rate,* dan *mutation rate*. Apabila ada salah satu yang belum di-*input* kan maka akan kembali untuk memasukkan parameter-parameter tersebut. Apabila semua parameter telah di-*input* kan, maka sistem akan ke proses *input* tahun yang akan ditentukan pola pergerakan kontainer dengan arus terpadat. Setelah itu sistem akan menampilkan hasil dari pola pergerakan kontainer dengan arus terpadat dalam bentuk graf.

4.3 Perancangan Sistem

Tampilan dari sistem penentuan pola jaringan pergerakan logistik ini dibangun dengan tampilan yang sederhana. *Software* yang digunakan untuk membangun sistem penentuan pola jaringan pergerakan logistik adalah Netbeans 7.0.1.

4.3.1 Arsitektur Sistem

Arsitektur sistem adalah sekumpulan komponen-komponen yang terhubung dan menggambarkan sifat dasar dari sebuah sistem. Untuk menggambarkan arsitektur dari sistem ini dapat ditunjukkan pada Gambar 4.3.



Gambar 4.3 Aristektur sistem penentuan pola pergerakan kontainer dengan arus terpadat

Pada Gambar 4.3, terdapat 4 komponen utama, yaitu antar muka program, yang digunakan sebagai tempat interaksi kepada *user*. Dari *input* yang dilakukan oleh *user* kemudian dilanjutkan ke algoritma yang berhubungan langsung dengan *database*. Algoritma akan membutuhkan data yang ada dalam *database*. Setelah dilakukan proses dengan algoritma, maka akan dihasilkan *output* dari program tersebut, yaitu pola pergerakan kontainer dengan arus terpadat yang divisualisasikan dalam bentuk graf.

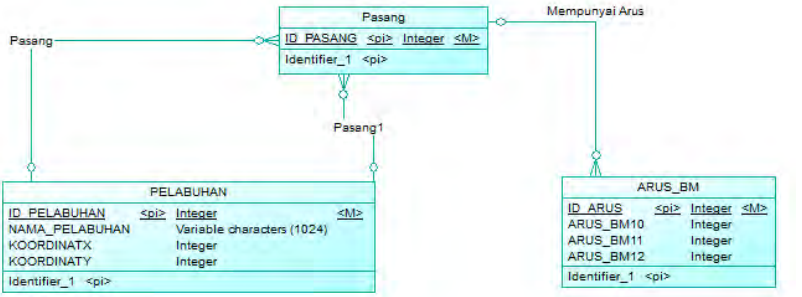
4.3.2 Proses Perancangan *Database*

Sistem penentuan pola jaringan pergerakan logistik dengan metode *Minimum Spanning Tree* berbasis algoritma genetika dalam Tugas Akhir ini menggunakan data arus pergerakan kontainer di

Indonesia pada tahun 2010, 2011, dan 2012. Setiap tahun terdapat transaksi distribusi kontainer dari pelabuhan asal ke pelabuhan tujuan. Dalam tiap tahun, transaksi dari pelabuhan asal ke pelabuhan tujuan tidak hanya melakukan satu kali transaksi, namun banyak transaksi. Sehingga data transaksi tersebut diolah menjadi total transaksi tiap pelabuhan asal dan pelabuhan tujuan. Kemudian data tersebut dirancang sebuah *database.*

4.3.2.1 CDM (*Conceptual Data Model*)

CDM (*Conceptual Data Model*) adalah tahapan untuk menganalisis struktur konsep dari suatu sistem informasi, mengidentifikasi entitas-entitas utama yang telah direpresentasikan, atribut-atribut, dan relasi antara keduanya. Untuk menggambarkan CDM pada sistem ini dapat ditunjukkan pada Gambar 4.4.



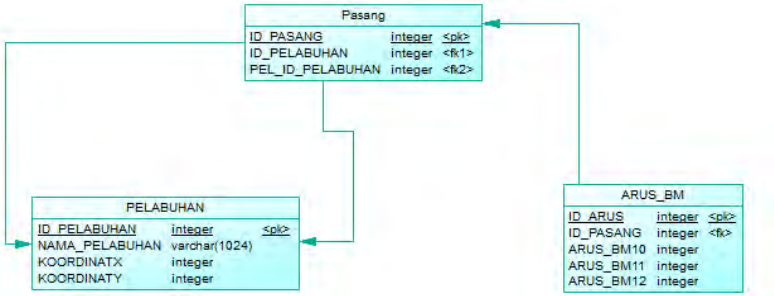
Gambar 4.4 CDM (*Conceptual DataModel*) pada sistem

Pada Gambar 4.4, terdapat 2 entitas yaitu PELABUHAN dan ARUS_BM, sedangkan entitas Pasang, digunakan untuk mengambil 2 Pelabuhan sebagai pelabuhan asal dan pelabuhan tujuan. Untuk entitas PELABUHAN digunakan untuk mendeskripsikan pelabuhan, yaitu terdapat 4 atribut, ID_PELABUHAN, NAMA_PELABUHAN, KOORDINATX, dan KOORDINATY. Pada tabel ARUS_BM digunakan untuk mencatat data arus pergerakan kontainer antar 2 pelabuhan pada tahun 2010,

2011, dan 2012. Dalam entitas ARUS_BM terdapat 4 atribut yaitu ID_ARUS, ARUS_BM10, ARUS_BM11, dan ARUS_BM12.

4.3.2.2 PDM (*Physical Data Model*)

PDM (*Physical Data Model*) adalah tahapan untuk menganalisis tabel, gambaran, dan objek lain dalam sebuah *database*, termasuk kebutuhan objek multidimensi untuk data *warehouse.* Untuk menggambarkan PDM pada sistem ini dapat ditunjukkan pada Gambar 4.5.



Gambar 4.5 PDM (*Physical Data Model*) pada sistem

Pada Gambar 4.5, terdapat 3 tabel yaitu PELABUHAN, Pasang, dan ARUS_BM. Pada tabel PELABUHAN terdapat kolom ID_PELABUHAN dan kolom NAMA_PELABUHAN. Kolom ID_PELABUHAN merepresentasikan label node-node yang digunakan dalam pemodelan graf. Terdapat 52 node dengan label yang sama dengan ID_PELABUHAN.

Untuk tabel ARUS_BM terdapat kolom ARUS_BM10, yang artinya jumlah kontainer yang dibawa pada tahun 2010, ARUS_BM11, yang artinya jumlah kontainer yang dibawa pada tahun 2011, ARUS_BM12, yang artinya jumlah kontainer yang dibawa pada tahun 2012, ID_ASAL, yang artinya ID_PELABUHAN dari pelabuhan asal, dan ID_TUJUAN, yang artinya ID_PELABUHAN dari pelabuhan tujuan. Dalam Tugas Akhir ini *database management* yang digunakan adalah JavaDB.

4.3.2.3 Data Luaran

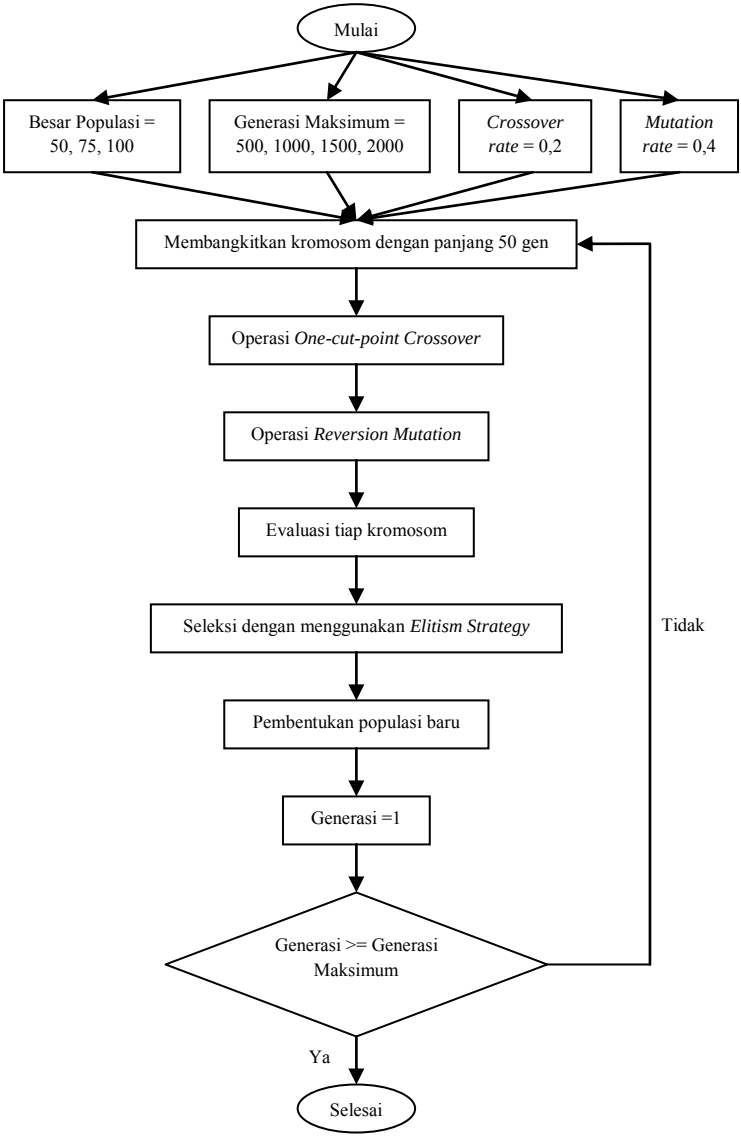
Data luaran pada sistem ini berupa jalur dengan node-node yang merupakan representasi dari pelabuhan. Dengan menentukan jalur tersebut yang dipengaruhi dengan besarnya jumlah kontainer yang dibawa dari pelabuhan asal ke pelabuhan tujuan, maka terbentuk model graf *tree*.

4.3.3 Gambaran Sistem Secara Umum

Gambaran sistem secara umum merupakan keseluruhan tahapan yang dilakukan sistem serta algoritma yang digunakan untuk mengerjakan tahapan tersebut. Sistem ini terdiri dari 7 tahapan utama, yaitu :

1. Tahap inisialisasi parameter, dalam sistem ini terdapat 4 parameter yaitu besar populasi, generasi maksimum, *crossover rate,* dan *mutation rate.*
2. Tahap pembangkitan kromosom, merupakan tahap untuk membentuk suatu kromosom dari sekumpulan gen, dengan gen merupakan node yang merepresentasikan pelabuhan.
3. Tahap *Crossover* / pindah silang, merupakan tahapan untuk memindahsilangkan gen-gen dari suatu kromosom ke kromosom lain, sehingga membentuk kromosom yang baru.
4. Tahap *Mutation* / mutasi, merupakan tahapan untuk melakukan mutasi gen di tiap kromosom.
5. Tahap evaluasi, merupakan tahap untuk membangkitkan nilai *fitness* pada setiap kromosom.
6. Tahap seleksi, merupakan tahap untuk menyeleksi kromosom dengan nilai *fitness* terbesar. Tahap seleksi ini dilakukan dengan menggunakan *elitism strategy.*
7. Tahap pembentukan populasi baru, merupakan pembentukan populasi dari hasil tahap genetik. Setelah tahap ini akan dilakukan proses pembentukan generasi selama generasi kurang dari generasi maksimum.

Gambaran sistem penentuan pola jaringan pergerakan logistik dengan *Minimum Spanning Tree* berbasis algoritma genetika dapat dilihat pada Gambar 4.6.



Gambar 4.6 Diagram Alir Tahapan utama Sistem Secara Umum

4.3.4 Tahap Inisialisasi Parameter

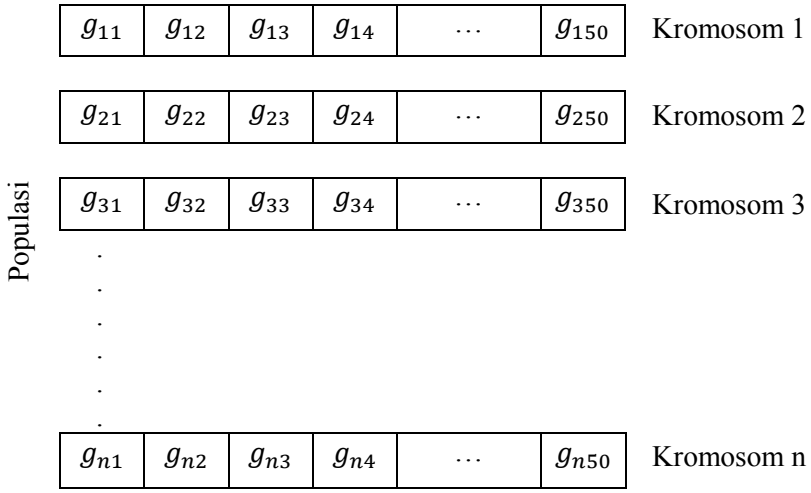
Tahap awal yang dilakukan sebelum melakukan tahapan yang lainnya adalah inisialisasi parameter. Dalam tahap ini terdapat 4 paremeter yang harus diinisialisasi, yaitu besar populasi, generasi maksimum, *crossover rate*, dan *mutation rate*.

1. Besar populasi adalah banyaknya populasi yang dibuat dalam satu generasi. Semakin besar populasi yang dibuat, maka semakin banyak kombinasi dari kromosom yang dibuat. Dalam Tugas Akhir ini besar populasi diinisialisasi dengan nilai 50, 75, dan 100.
2. Generasi maksimum adalah banyaknya generasi yang akan dibuat. Generasi maksimum berpengaruh pada pemberhentian iterasi pada sistem. Dalam Tugas Akhir ini generasi maksimum diinisialisasi dengan nilai 500, 1000, 1500, dan 2000.
3. *Crossover rate*, adalah peluang yang digunakan untuk menentukan peluang kromosom yang dipindahsilang. *Crossover rate* mempunyai *range* antara 0-1. Dalam Tugas Akhir ini *Crossover rate* diinisialisasi dengan nilai 0,2[5].
4. *Mutation rate*, adalah peluang yang digunakan untuk menentukan peluang gen yang dimutasi. *Mutation rate* mempunyai *range* antara 0-1. Dalam Tugas Akhir ini menginisialisasi *Mutation rate* diinisialisasi dengan nilai 0,4[5].

4.3.5 Tahap Membangkitkan Kromosom

Dalam tahap ini dilakukan dengan membangkitkan *prufer number*. *Prufer number* adalah bilangan acak antara 1-52 sebanyak 50, dengan n adalah banyaknya vertek dalam suatu graf. Dengan membuat bilangan acak antara 1-52 sebanyak 50 maka dapat dibentuk suatu kromosom dengan gen sebanyak 50. Sehingga gen merupakan hasil dari pembangkitan *prufer number*. Dalam permasalahan ini gen dalam kromosom merepresentasikan pelabuhan yang melakukan transaksi kontainer. Untuk

merepresentasikan gen-gen pada kromosom dapat dilihat pada Gambar 4.7.



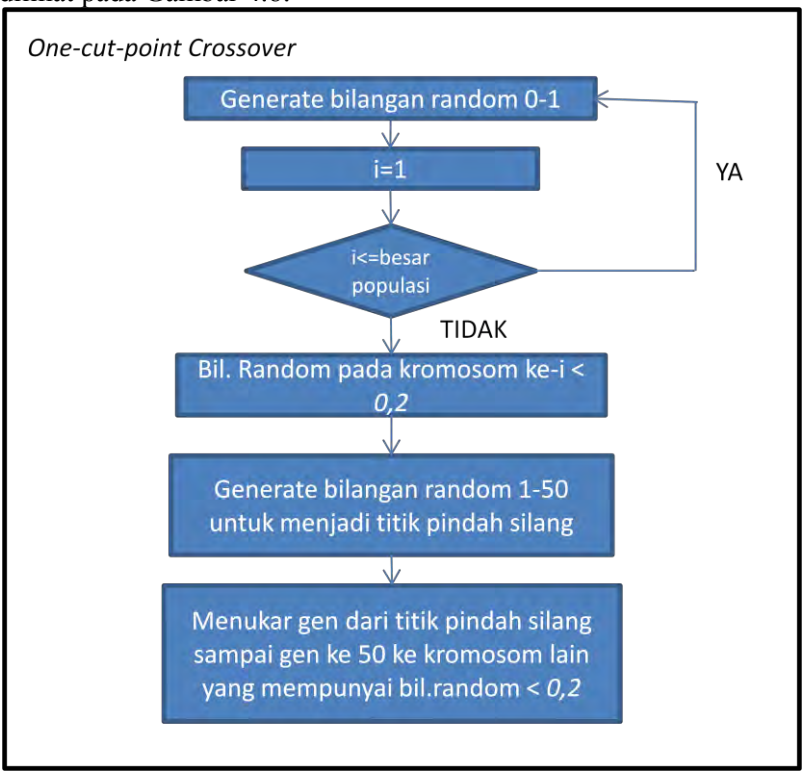
Gambar 4.7 Representasi gen dalam kromosom

4.3.6 Tahap *Crossover*

Metode *crossover* merupakan metode yang digunakan untuk memindahsilangkan gen-gen pada kromosom induk, sehingga menghasilkan kromosom anak dengan gen hasil pindahsilang. Terdapat beberapa metode pada *crossover*, yaitu *one-cut-point crossover*, *twopoint crossover*, *uniform crossover*, dan *arithmetic crossover*.

Metode *crossover* yang digunakan pada Tugas Akhir ini adalah *one-cut-point crossover*. Dalam tahap ini akan dilakukan pembangkitan bilangan acak di tiap kromosom. Untuk kromosom dengan nilai bilangan acak kurang dari *crossover rate* dijadikan sebagai kromosom induk, yang dapat dilakukan *crossover* ke induk lain. Setelah ditentukan kromosom-kromosom yang sebagai induk, maka dilakukan pembangkitan bilangan acak antara 1-50, untuk menentukan titik *crossover* di tiap kromosom induk. Setelah

ditentukan titik *crossover* tersebut, maka gen yang terdapat pada posisi di titik *crossover* tersebut sampai gen terakhir akan dipindahsilangkan dengan gen di kromosom induk yang lain pada posisi yang sama. Diagram alir *one-cut-point crossover* dapat dilihat pada Gambar 4.8.



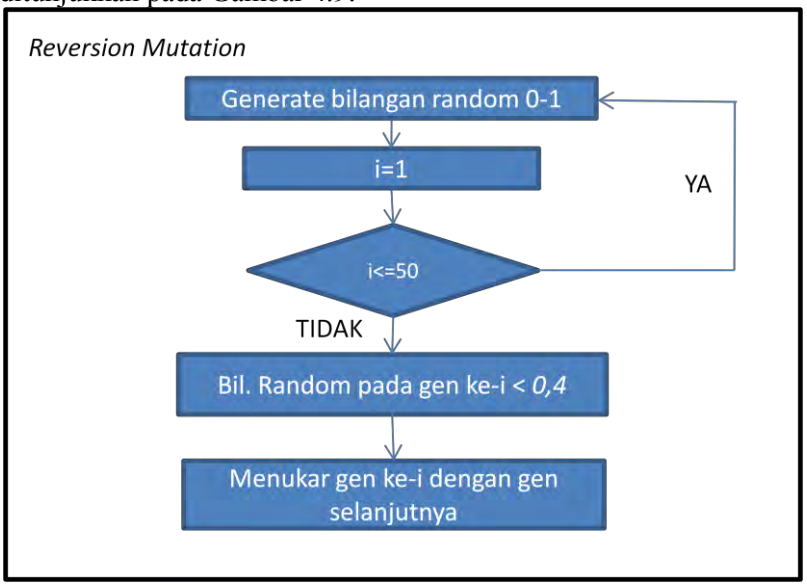
Gambar 4.8 Diagram alir *one-cut-point crossover.*

4.3.7 Tahap *Mutation*

Tahapan selanjutnya adalah tahap *mutation. Mutation* adalah metode untuk merekombinasi kromosom dengan cara memilih kromosom yang akan dimutasi, dan kemudian menentukan titik mutasi pada kromosom tersebut secara acak pula. Terdapat

beberapa metode dalam tahap *mutation,* yaitu *flipping*, *interchanging*, *reversing*, *inversion*, *insertion,* dan *reciprocal.*

Metode *mutation* yang digunakan pada Tugas Akhir ini adalah *reversing mutation.* Dalam tahap ini akan dilakukan pembangkitan bilangan acak di tiap kromosom anak hasil dari *crossover*. Kemudian dilakukan pembangkitan bilangan acak pada tiap gen. Untuk bilangan acak dengan nilai kurang dari *mutation rate,* maka akan dimutasi dengan gen dengan posisi selanjutnya. Dalam tahap *mutation* ini akan dihasilkan kromosom hasil rekombinasi dari tahap *crossover* dan *mutation*. Kemudian kromosom tersebut akan dievaluasi dengan cara me-*decode prufer number*. Untuk diagram alir dari *reversing mutation* dapat ditunjukkan pada Gambar 4.9.



Gambar 4.9 Diagram alir *reversing mutation.*

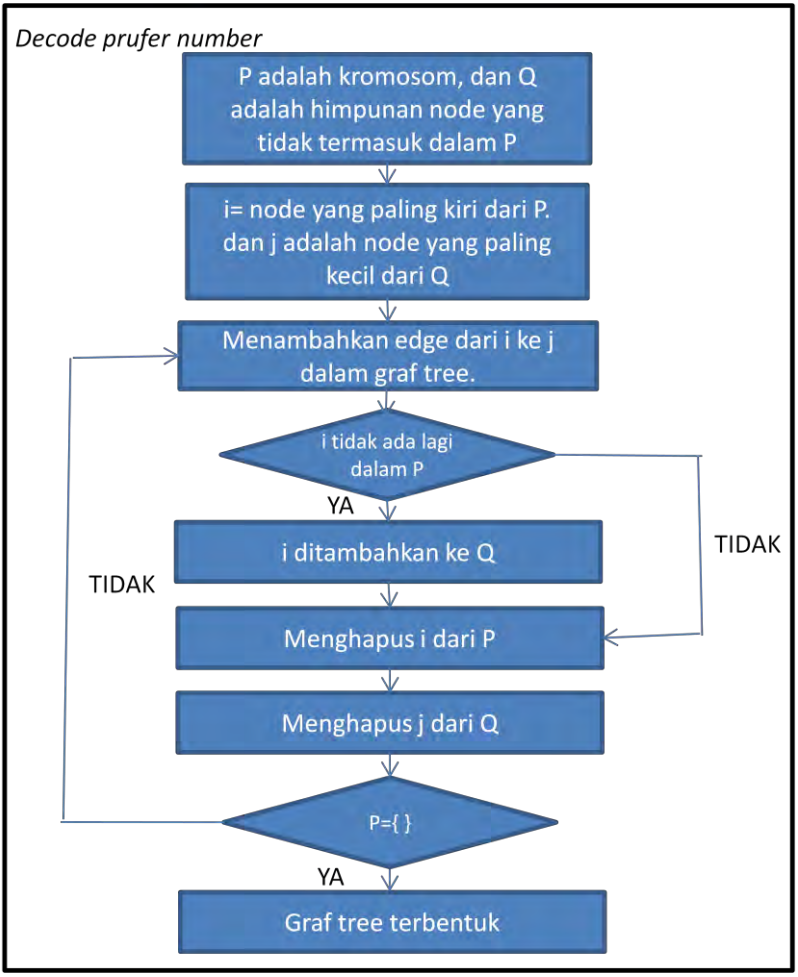
4.3.8 Tahap Evaluasi

Tahapan selanjutnya adalah tahap evaluasi. Evaluasi adalah metode untuk menghitung nilai *fitness* pada tiap kromosom. Tahap

evaluasi dilakukan dengan me-*decode prufer number*, sehingga menghasilkan bentuk *tree*. Pada metode ini tiap kromosom akan memiliki nilai *fitness* dari fungsi berikut

$$\min \frac{1}{\sum_i \sum_j x_{ij}}$$

Setelah mendapat kromosom dari hasil *crossover* dan *mutation*, maka kromosom yang berbentuk *prufer number* tersebut akan di-*decode* menjadi bentuk *tree*, sehingga dapat menghitung nilai *fitness* di tiap kromosom. Pertama adalah *prufer number* dinyatakan sebagai P, dan Q adalah himpunan dari 1-52 selain P. Terdapat i yang merupakan node paling kiri dari P, dan j adalah node paling kecil dari Q. Kemudian i dan j dihubungkan dengan *edge*. Apabila dalam P tidak mengandung lagi i, maka i dihapus dan ditambahkan ke Q. Apabila P masih mengandung i, maka i dihapus. Kemudian j dihapus dari Q, dan seterusnya sampai P merupakan himpunan kosong. Setelah itu akan terbentuk *tree*, yang kemudian dapat dicari suatu nilai *fitness*nya. Untuk diagram alir dari *decode prufer number* dapat ditunjukkan pada Gambar 4.10.



Gambar 4.10 Diagram alir *decode prufer number*.

4.3.9 Tahap Seleksi

Tahap seleksi adalah tahap untuk menyeleksi kromosom dengan nilai *fitness* terbesar. Dengan menggunakan *elitism strategy*, maka kromosom dengan nilai *fitness* terbesar akan dijaga,

sampai populasi baru terbentuk. Sehingga kromosom dengan nilai *fitness* terbesar akan tetap ada sampai ada kromosom baru dengan nilai *fitness*nya lebih besar dari kromosom tersebut. Strategi ini bertujuan untuk menjaga kromosom yang mempunyai nilai *fitness* terbesar agar tidak diganti oleh kromosom lain dengan nilai *fitness* lebih kecil.

4.3.10 Tahap Pembentukan Populasi Baru

Tahap pembentukan populasi baru adalah tahap untuk membentuk populasi baru dengan kromosom-kromosom hasil dari operator genetika. Kromosom-kromosom tersebut akan dikumpulkan menjadi populasi baru dengan jumlah yang sesuai dengan populasi awal.

Setelah tahap ini dilakukan, selanjutnya akan dilakukan iterasi lagi dengan membuat generasi baru yang dimulai dengan tahap *crossover*. Sehingga populasi awal telah diganti dengan populasi baru hasil darioperator genetika.

4.4 Implementasi

Perancangan sistem yang telah dibangun, selanjutnya diimplementasikan pada bahasa pemrograman java dengan menggunakan *software* Netbeans 7.0.1.

4.4.1 Implementasi Program

Dalam program yang telah dibuat, terdapat *class-class* untuk mengimplementasikan *Minimum Spanning Tree* berbasis algoritma genetika. *Class-class* tersebut adalah sebagai berikut:

a) Home.java

Class ini merupakan *class* utama yang digunakan untuk menjalankan program. Dalam *class* ini terdapat inisialisasi parameter yaitu besar populasi, generasi maksimum, *crossover rate,* dan *mutation rate.* Dalam *class* ini juga terdapat beberapa fungsi yang digunakan untuk melakukan operasi genetika yaitu, *one-cut-point crossover, reversión mutation,* dan *elitism strategy*.

Untuk keseluruhan *source code* dari Home.java dapat dilihat pada Lampiran A.

Nama Kelas : Home.java
Atribut: POP_SIZE : Integer MAX_ITER : Integer RASIO_MUTASI : Double RASIO_CROSSOVER : Double PC0 : String PM0 : String pop_size0 : String max_gen0 : String

Fungsi:
Home() // konstruktor untuk menginisialisasi nilai POP_SIZE, MAX_ITER, RASIO_MUTASI, RASIO_CROSSOVER
GA() // fungsi untuk melakukan operasi genetika

1) Inisialisasi parameter ini digunakan untuk menentukan nilai dari besar populasi, generasi maksimum, *crossover rate*, dan *mutation rate.* Inisialisasi ini dapat dilakukan secara otomatis, atau diinputkan sendiri oleh pengguna sesuai dengan kebutuhan. Tahap inisialisasi parameter diimplementasikan kedalam program dalam bentuk textfield. Dan dapat diinputkan sesuai dengan kebutuhan, dimana besar populasi dan generasi maksimum bertipe *Integer,* dan *crossover rate dan mutation rate* bertipe *Float.*
2) Pada tahap *crossover* digunakan metode *one-cut-point crossover*. Pertama yaitu membuat suatu *arraylist* sebagai orang tua yang akan dipindahsilangkan, dan *arraylist* sebagai anak yang merupakan hasil dari *crossover.* Untuk pembangkitan dua *arraylist* dapat diimplementasikan pada *source code* sebagai berikut :

```
ArrayList<Populasi> parent = new
ArrayList();
```

Selanjutnya dilakukan pembangkitan bilangan acak antara 0-1 untuk tiap kromosom, yang berfungsi untuk menjadikan kromosom tersebut menjadi kromosom induk/orang tua. Untuk pembangkitan bilangan acak dapat diimplementasikan pada *source code* sebagai berikut:

```
for (int i = 0; i < pop1.size(); i++) {
        if (Math.random() < RASIO_CROSSOVER) {
            parent.add(pop1.get(i));
        }
    }
```

Untuk perulangan *for* dilakukan sebanyak populasi yang sudah diinisialisasi di awal. Pada percabangan *if*, apabila kromosom dengan bilangan acak kurang dari *crossover rate*, maka kromosom tersebut merupakan kromosom induk. Jika tidak maka kromosom bukan merupakan kromosom induk. Kemudian menentukan titik potong untuk *crossover.* Dan dilanjutkan dengan memindahsilangkan antara gen-gen dari titik potong tersebut sampai gen terakhir dari suatu kromosom induk ke kromosom induk lain, yang kemudian kromosom hasil *crossover* disebut sebagai kromosom anak. Untuk tahap ini dapat diimplementasikan pada *source code* sebagai berikut:

```
for (int i = 0; i < parent.size(); i+=2) {
        int tipot =
rnd.nextInt(pop1.get(0).getP().size());
        ArrayList temp1 = (ArrayList)
parent.get(i).getP();
        ArrayList temp2 = (ArrayList)
parent.get(i+1).getP();
        ArrayList p1=new ArrayList();
        ArrayList p2=new ArrayList();
        for(int j=0;j<tipot;j++){
           p1.add(temp1.get(j));
           p2.add(temp2.get(j));
        }
        for(int
k=tipot;k<pop1.get(0).getP().size();k++){
           p1.add(temp2.get(k));
           p2.add(temp1.get(k));
        }
        Populasi a=new Populasi();
        a.setP(p1);
        Populasi b=new Populasi();
```

Perulangan *for* dilakukan sebanyak kromosom yang menjadi sebagai kromosom induk. Tipot adalah titik potong yang digunakan untuk *crossover*. Range dari tipot adalah 1-jumlah gen dari suatu kromosom induk. Untuk fungsi temp1 dan temp2 sebagai kromosom induk 1 dan kromosom induk 2. Untuk *arraylist* p1 dan p2 digunakan untuk membentuk suatu kromosom hasil dari pindah silang. Untuk perulangan *for* selanjutnya yaitu, gen pada indeks ke-0 sampai sebelum titik potong yang berada pada kromosom induk 1 akan dimasukkan ke dalam p1, begitu pula dengan p2. Untuk perulangan *for* berikutnya yaitu

memasukkan gen pada indeks ke-tipot sampai gen terkahir pada kromosom induk 2 akan dimasukkan kedalam p1, begitu pula dengan kromosom induk 1. Kemudian p1 dan p2 disimpan dalam populasi dan membentuk kromosom anak.

3) Tahap *Mutation* menggunakan metode *reversion mutation*, yaitu menukar gen dengan gen pada posisi selanjutnya. Pertama yaitu melakukan pembangkitan bilangan acak pada tiap kromosom induk dengan *range* 0-1. Untuk tahap *mutation* dapat diimplementasikan pada *source code* sebagai berikut:

```
ArrayList<Populasi> mutasi =new ArrayList();
      for(int j=0; j<child.size();j++){
         Populasi childtemp = new Populasi();
         ArrayList child2 = new ArrayList();
         Object child1 [];
         child1 = child.get(j).getP().toArray();
         for(int k=0;k<child.get(j).getP().size();k++){
            if (Math.random() < RASIO_MUTASI) {
               if(k!=child.get(j).getP().size()-1){
                  int temp = (int)child1[k];
                  child1[k] = child1[k+1];
                  child1[k+1]=temp; }
            }
            child2.add(k, child1[k]);}
         childtemp.setP(child2);
         mutasi.add(childtemp); }
```

Arraylist mutation digunakan untuk menyimpan hasil dari operator *mutation.* Untuk perulangan *for* dilakukan

sebanyak jumlah kromosom anak. Pertama yaitu menkonversi *arraylist* menjadi *array* kemudian dijadikan sebagai objek. Untuk perulangan *for* selanjutnya yaitu untuk beriterasi sampai gen terakhir dari kromosom anak. Jika bilangan random yang telah dibangkitkan sebelumnya kurang dari *mutation rate,* maka gen pada posisi tersebut akan dipindahkan ke posisi gen selanjutnya, dan untuk gen selanjutnya ditukar ke gen sebelumnya. Setelah itu kromosom anak dengan nama child1 yang berbentuk objek dikonversi lagi menjadi *arraylist* child2. Kemudian disimpan kedalam populasi. Dan diinisialisasi sebagai *arraylist* mutasi sebagai hasil dari operator *mutation*.

4) *Elitism strategy* digunakan untuk menjaga kualitas dari kromosom, sehingga pada tiap generasi kualitas kromosom tidak menurun. Tahap seleksi dapat diimplementasikan pada *source code* sebagai berikut:

```
int idbot1[][] = new int[pop1.size()][2];
      for (int i = 0; i < pop1.size(); i++) {
         ArrayList b = pop1.get(i).getP();
         int bobot = pop1.get(i).getTB();
         idbot1[i][0] = i;
         idbot1[i][1] = bobot;
      }
      System.out.println();
      Arrays.sort(idbot1, new Comparator<int[]>() {
         @Override
         public int compare(int[] t1, int[] t) {
            return Double.compare(t[1], t1[1]);
         }
      });
      int min = idbot[pop1.size()-1][0];
      prufer c = new prufer();
      c.arraygen(pop1.get(min).getP());
```

Idbot1[][] dibuat menjadi *array* 2 dimensi, yang berfungsi untuk satu *array*. Kemudian diurutkan dari nilai *fitness* yang terbesar sampai nilai *fitness* yang terkecil. Dengan demikian kromosom pada urutan terbawah merupakan kromosom dengan nilai *fitness* terkecil.

b) prufer.java

Class ini digunakan untuk membangkitkan kromosom dengan menggunakan *prufer number*. Kemudian terdapat fungsi untuk me-*decode prufer number*. Untuk keseluruhan *source code* pada *class* prufer.java dapat dilihat pada Lampiran B.

Nama Kelas : prufer.java
Atribut: n : Integer rand : Random p : Arraylist q : Arraylist pasangan : Arraylist

Fungsi:
arraygen() // menyimpan populasi baru yang berbentuk *arraylist* hasil operasi genetika kedalam *arraylist* p
gen() // mendapat nilai p
setgen() // menambahkan gen dan indeks gen pada *arraylist* p
prufer1() // membangkitkan kromosom
Q() // mendapatkan nilai q
setQ() // menambahkan gen dan indeks gen pada *arraylist* q
Qi() // cek bilangan 1-52yang tidak ada dalam p, dan menyimpan ke *arraylist* q
sortQi() // mengurutkan gen dari q mulai dari yang kecil sampai yang paling besar
pasang() // membuat kombinasi pasangan dari gen dalam *arraylist* p dan q
getPasangan() // mendapatkan *arraylist* pasangan
getP() // mendapatkan *arraylist* p

1) Tahap pembangkitan kromosom dilakukan dengan menggunakan *arraylist* sebagai kumpulan dari gen-gen yang disimpan. Tahap ini diimplementasikan pada *source code* sebagai berikut:

```
public void prufer1(){
      for(int i=0; i<n-2; ++i) {
         int r=rand.nextInt(n)+1;
         this.setgen(i, r);
      }
   }
```

perulangan *for* digunakan untuk membangkitkan *prufer number* sebanyak 50. Untuk r adalah bilangan acak yang dibangkitkan dengan atribut rand dengan tipe *integer*. Kemudian fungsi setgen digunakan untuk menyimpan hasil dari pembangkitan bilangan acak ke dalam *arraylist.*

2) *Decode prufer number* digunakan untuk membentuk suatu *tree*. Langkah pertama yaitu membuat himpunan Q yang berisi node yang tidak ada dalam *prufer number*. Tahap ini dapat diimplementasikan pada *source code* sebagai berikut:

```
public void Qi(){
      int y = 0;
      for(int i=1; i<=n;i++){
         if(p.contains(i)==false){
            this.setQ(y, i);
            y++;
         }
      }
```

Untuk perulangan *for* dilakukan sebanyak 52, yaitu jumlah dari node. Pada percabangan *if*, apabila tidak mengandung node pada *prufer number* maka disimpan kedalam Q. Jika mengandung node pada *prufer number*

maka tidak disimpan dalam Q. Langkah selanjutnya yaitu memasangkan antara *prufer number* dengan Q, yang diimplementasikan pada *source code* sebagai berikut:

```
public void pasang(){
      ArrayList p1=(ArrayList) p.clone();
      while(p1.size()!=0){
         int nP=(int)p1.get(0);
         int temp=1;
         for(int j=1;j<p1.size();j++){
            if(p1.get(j).equals(nP)){
               temp++;} }
         if(temp==1){
             int nQ=(int)q.get(0);
             pasangan.add(new PQ(nP,nQ));
             p1.remove(0);
             q.remove(0);
             q.add(nP);
             sortQi();}
         else{
            int nQ=(int)q.get(0);
             pasangan.add(new PQ(nP,nQ));
             p1.remove(0);
             q.remove(0);
             sortQi(); }}
      pasangan.add(new PQ((int)q.get(0),(int)q.get(1)));
}
```

Arraylist p1 merupakan hasil klon dari *prufer number*. Untuk perulangan *while* berjalan selama ukuran p1 tidak 0. Kemudian menginisialisasi nP sebagai gen paling kiri

dari p1. Perulangan *for* digunakan apabila ada gen pada kromosom mempunyai nilai yang sama dengan gen paling kiri, maka akan dilanjutkan ke fungsi if(temp==1), dimana fungsi tersebut memasangkan nP dan nQ, dan menghapus gen paling kiri dari kromosom p1, dan menghapus gen paling kecil dari kromosom q, serta menambah kromosom q dengan gen nP. Apabila temp tidak sama dengan 1, maka hanya menghapus gen paling kiri dari kromosom p1, dan menghapus gen paling kecil dari kromosom q. Tahap ini berulang sampai P berupa himpuan kosong.

c) Populasi.java

Class ini digunakan untuk mendeklarasikan populasi dalam bentuk *arraylist.* Nilai *fitness* pada tiap kromosom. Dalam *class* ini juga terdapat fungsi untuk menyimpan dan mendapat populasi dan nilai *fitness*. Untuk keseluruhan *source code* pada *class* Populasi.java dapat dilihat pada Lampiran C.

Nama Kelas : Populasi.java
Atribut: p : Arraylist totalbobot : double arus : double

Fungsi:
setP() // menyimpan kromosom ke arraylist p
setTB() // menyimpan nilai *fitness* kedalam totalbobot
getP() // mendapatkan *arraylist* p
getTB() // mendapatkan nilai totalbobot
setArus() //menyimpan jumlah pergerakan kontainer kedalam arus
getArus() // mendapatkan nilai arus

d) PQ.java

Class ini digunakan untuk mendeklarasikan dan menyimpan pasangan antara *prufer number* dan Q yang merupakan himpunan

yang tidak termasuk *prufer number*. Untuk keseluruhan *source code* pada *class* Populasi.java dapat dilihat pada Lampiran D.

Nama Kelas : PQ.java
Atribut: p : Integer q : Integer

Fungsi:
PQ() // menyimpan pasangan dari *prufer number* ke *arraylist* p dan q
setP() // menyimpan nilai *prufer number* ke *arraylist* p
setQ() // menyimpan nilai dari Q ke *arraylist* q
getP() // mendapatkan nilai dari p
getQ() // mendapatkan nilai dari q

4.4.2 Implementasi Antarmuka

Dalam implementasi program, terdapat antarmuka yang berfungsi untuk mempermudah *user* dalam penggunaan program. Implementasi antarmuka dapat ditunjukkan pada Gambar 4.11.

Gambar 4.11 Antarmuka program

Pada Gambar 4.11, terdapat 4 *textfield* yang berfungsi untuk memasukkan nilai. Yang pertama yaitu Ukuran Populasi, digunakan untuk memasukkan nilai parameter ukuran populasi. Yang kedua adalah *Max Generation*, digunakan untuk memasukkan nilai parameter generasi maksimum. Yang ketiga adalah *Crossover Rate*, digunakan untuk memasukkan nilai *crossover rate*. Yang keempat adalah *Mutation Rate*, digunakan untuk memasukkan nilai *mutation rate*. Untuk tombol Tahun 2010, digunakan untuk menentukan pola pergerakan kontainer dengan menggunakan data pada tahun 2010. Untuk tombol Tahun 2011, digunakan untuk menentukan pola pergerakan kontainer dengan menggunakan data pada tahun 2011. Dan Untuk tombol Tahun 2012, digunakan untuk menentukan pola pergerakan kontainer dengan menggunakan data pada tahun 2012.

BAB V
PENGUJIAN DAN PEMBAHASAN HASIL

Bab ini menjelaskan mengenai pengujian yang dilakukan terhadap sistem penentuan pola jaringan pergerakan logistik. Hasil pengujian kemudian dibahas untuk mengetahui unjuk kerja sistem secara keseluruhan dalam menjalankan fungsi yang diharapkan.

5.1 Pengujian Sistem dengan Data pada Tahun 2010

Pada pengujian sistem dengan data pada tahun 2010, dilakukan pengujian sebanyak 12 kali. Pengujian tersebut dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Pengujian tersebut dapat digambarkan pada Tabel 5.1.

Nama Pengujian	Ukuran Populasi	Generasi Maksimum
A1	50	500
A2	50	1000
A3	50	1500
A4	50	2000
B1	75	500
B2	75	1000
B3	75	1500
B4	75	2000
C1	100	500
C2	100	1000
C3	100	1500
C4	100	2000

Tabel 5.1 Data parameter pengujian pada tahun 2010

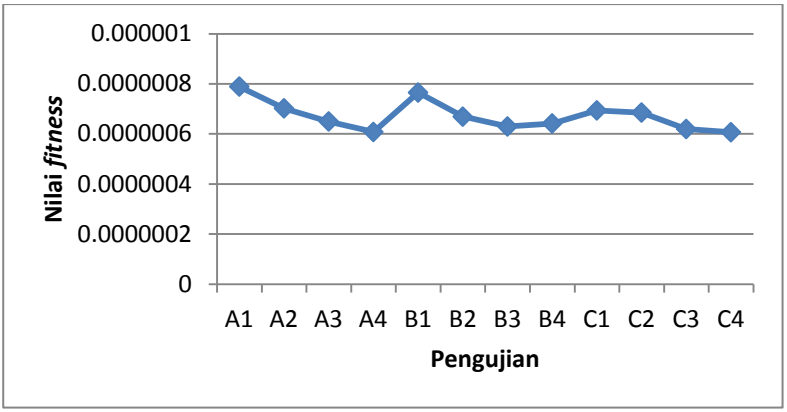
Tabel 5.1 menunjukkan bahwa pada pengujian A1, dilakukan dengan parameter ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500, maka algoritma genetika akan berhenti. Demikian juga pada

pengujian yang lain. Pada 12 pengujian ini, diperoleh hasil dari nilai *fitness*, yang dapat ditunjukkan pada Tabel 5.2.

Pengujian	Nilai *fitness*
A1	7,89125e-07
A2	7,02213e-07
A3	6,49793e-07
A4	6,07763e-07
B1	7,65185e-07
B2	6,69042e-07
B3	6,29661e-07
B4	6,41927e-07
C1	6,93922e-07
C2	6,85385e-07
C3	6,19716e-07
C4	6,06834e-07

Tabel 5.2 Nilai *fitness* dari pengujian dengan data pada tahun 2010

Pada Tabel 5.2, pengujian A1 mempunyai nilai *fitness* sebesar 7,89125e-07, dan juga pada pengujian-pengujian berikutnya. Untuk melihat grafik perbandingan antar pengujian dapat ditunjukkan pada Gambar 5.1.



Gambar 5.1 Grafik perbandingan antar pengujian

Pada Gambar 5.1, diketahui bahwa terjadi penurunan pada tiap jenis pengujian. Dari grafik tersebut terlihat bahwa pada pengujian C4 mempunyai nilai *fitness* yang paling kecil, sehingga dapat disimpulkan bahwa pengujian C4 merupakan pengujian yang paling baik diantara 11 pengujian yang lain.

5.2 Pengujian Sistem dengan Data pada Tahun 2011

Pada pengujian sistem dengan data pada tahun 2011, dilakukan pengujian sebanyak 12 kali. Pengujian tersebut dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Pengujian tersebut dapat digambarkan pada Tabel 5.3.

Nama Pengujian	Ukuran Populasi	Generasi Maksimum
A1	50	500
A2	50	1000
A3	50	1500
A4	50	2000
B1	75	500
B2	75	1000
B3	75	1500
B4	75	2000

Nama Pengujian	Ukuran Populasi	Generasi Maksimum
C1	100	500
C2	100	1000
C3	100	1500
C4	100	2000

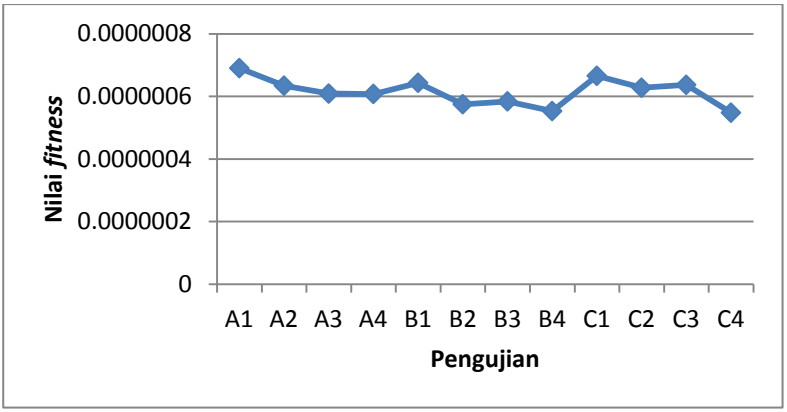
Tabel 5.3 Data parameter pengujian pada tahun 2011

Tabel 5.3, menunjukkan bahwa pada pengujian A1, dilakukan dengan parameter ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500, maka algoritma genetika akan berhenti. Demikian juga pada pengujian yang lain. Pada 12 pengujian ini, diperoleh hasil dari nilai *fitness*, yang dapat ditunjukkan pada Tabel 5.4.

Pengujian	Nilai *fitness*
A1	6,90179e-07
A2	6,34239e-07
A3	6,08923e-07
A4	6,07775e-07
B1	6,43445e-07
B2	5,75027e-07
B3	5,8409e-07
B4	5,53143e-07
C1	6,65664e-07
C2	6,27664e-07
C3	6,36857e-07
C4	5,4793e-07

Tabel 5.4 Nilai *fitness* dari pengujian dengan data pada tahun 2011

Pada Tabel 2., pengujian A1 mempunyai nilai *fitness* sebesar 6,90179e-07, dan juga pada pengujian-pengujian berikutnya. Untuk melihat grafik perbandingan antar pengujian dapat ditunjukkan pada Gambar 2.



Gambar 5.2 Grafik perbandingan antar pengujian

Pada Gambar 5.2, diketahui bahwa terjadi peningkatan pada pengujian B2 ke B3, serta C2 ke C3. Namun mengalami penurunan pada tiap jenis pengujian yang lain. Dari grafik tersebut terlihat bahwa pada pengujian C4 mempunyai nilai *fitness* yang paling kecil, sehingga dapat disimpulkan bahwa pengujian C4 merupakan pengujian yang paling baik diantara 11 pengujian yang lain.

5.3 Pengujian Sistem dengan Data pada Tahun 2012

Pada pengujian sistem dengan data pada tahun 2012, dilakukan pengujian sebanyak 12 kali. Pengujian tersebut dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Pengujian tersebut dapat digambarkan pada Tabel 5.5.

Nama Pengujian	Ukuran Populasi	Generasi Maksimum
A1	50	500
A2	50	1000
A3	50	1500
A4	50	2000
B1	75	500
B2	75	1000

Nama Pengujian	Ukuran Populasi	Generasi Maksimum
B3	75	1500
B4	75	2000
C1	100	500
C2	100	1000
C3	100	1500
C4	100	2000

Tabel 5.5. Data parameter pengujian pada tahun 2012

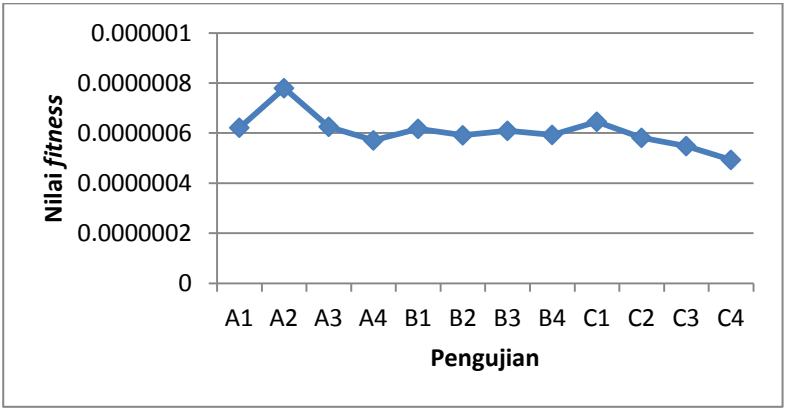
Tabel 5.5, menunjukkan bahwa pada pengujian A1 dilakukan dengan parameter ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500, maka algoritma genetika akan berhenti. Demikian juga pada pengujian yang lain. ada 12 pengujian ini, diperoleh hasil dari nilai *fitness*, yang dapat ditunjukkan pada Tabel 5.6.

Pengujian	Nilai *fitness*
A1	6,21591e-07
A2	7,79379e-07
A3	6,25297e-07
A4	5,71036e-07
B1	6,17149e-07
B2	5,9137e-07
B3	6,09336e-07
B4	5,92981e-07
C1	6,45476e-07
C2	5,81263e-07
C3	5,48337e-07
C4	4,93131e-07

Tabel 5.6 Nilai *fitness* dari pengujian dengan data pada tahun 2012

Pada Tabel 5.6, pengujian A1 mempunyai nilai *fitness* sebesar 6,90179e-07, dan juga pada pengujian-pengujian

berikutnya. Untuk melihat grafik perbandingan antar pengujian dapat ditunjukkan pada Gambar 5.3.



Gambar 5.3 Grafik perbandingan antar pengujian

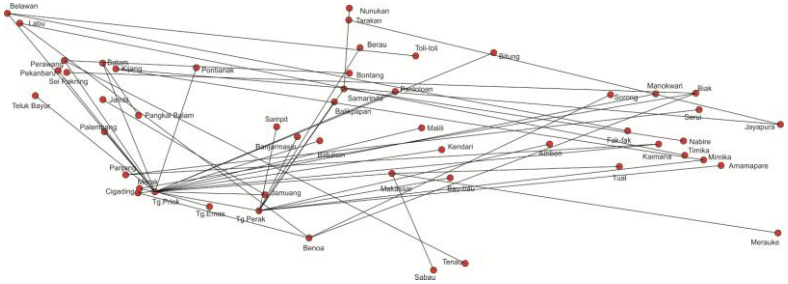
Pada Gambar 5.3, diketahui bahwa terjadi peingkatan pada pengujian A1 ke A2, serta B2 ke B3. Dan yang lain terjadi penurunan pada tiap jenis pengujian. Dari grafik tersebut terlihat bahwa pada pengujian C4 mempunyai nilai *fitness* yang paling kecil, sehingga dapat disimpulkan bahwa pengujian C4 merupakan pengujian yang paling baik diantara 11 pengujian yang lain.

5.4 Hasil dari Pengujian dengan Data pada Tahun 2010

Dari pengujian dengan data pada tahun 2010, didapat hasil bahwa jalur dengan arus pergerakan paling padat terdapat setelah pengujian C4, yaitu pada ukuran populasi ke 100 dan generasi maksimum 2000, dengan nilai *fitness* 6,06834e-07,dengan jumlah kontainer yang dibongkar muat sebesar 1647896 Teu's dan jalur sebagai berikut:

{Tg. Perak-Amamapare}, {Tg. Perak-Ambon}, {Tg. Perak-Balikpapan}, {Tg. Perak-Banjarmasin}, {Tg. Priok-Batu Licin}, {Tg. Perak-Bau-bau}, {Tg. Perak-Berau}, {Tg. Priok-Bitung},

{Perawang-Bontang}, {Belawan-Fak-fak}, {Jamuang-Jambi}, {Tg. Priok-Jamuang}, {Tg. Priok-Kendari}, {Perawang-Kupang}, {Tg. Priok-Malili}, {Tg. Priok-Manokwari}, {Makassar-Merauke}, {Samarinda-Nabire}, {Samarinda-Nunukan}, {Tg. Priok-Padang}, {Tg. Priok-Palembang}, {Tg. Priok-Palu}, {Batam-Pangkal Balam}, {Tg. Priok-Batam}, {Tg. Priok-Pekanbaru}, {Tg. Priok-Perawang}, {Tg. Priok-Pontianak}, {Tg. Perak-Samarinda}, {Tg. Perak-Sampit}, {Biak- Sei Pakning}, {Lampung-Serui}, {Kaimana-Lampung}, {Cigading-Kaimana}, {Benoa-Cigading}, {Benoa-Sorong}, {Jayapura-Tarakan}, {Kijang-Jayapura}, {Mimika-Kijang}, {Tg. Priok-Tg. Emas}, {Labu-Timika}, {Benoa-Labu}, {Biak-Benoa}, {Merak-Biak}, {Tg. Priok-Merak}, {Belawan-Toli-toli}, {Tg. Priok-Belawan}, {Makassar-Tual}, {Makassar-Sabau}, {Tg. Priok-Makassar}, {Tg. Perak-Tg. Priok}, {Tg. Perak-Mimika}. Dari jalur yang didapat, dimodelkan dalam bentuk graf yang dapat ditunjukkan pada Gambar 4.



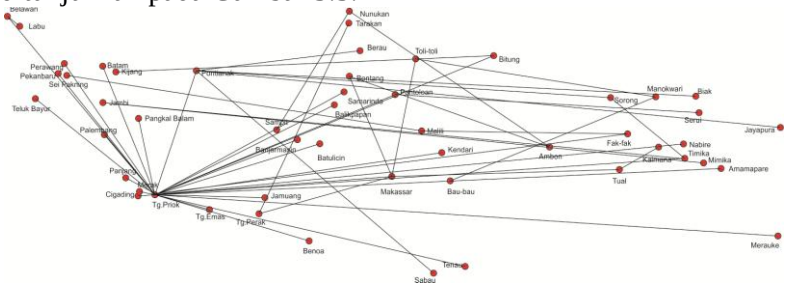
Gambar 5.4 Model graf *Tree* pada data tahun 2010

Dari Gambar 5.4, dapat ditunjukkan bahwa semua pelabuhan terhubung dan memiliki arus yang terpadat diantara kedua belas pengujian yang telah dilakukan.

5.5 Hasil dari Pengujian dengan Data pada Tahun 2011

Dari pengujian dengan data pada tahun 2011, didapat hasil bahwa jalur dengan arus pergerakan paling padat terdapat setelah

pengujian C4, yaitu pada ukuran populasi ke 100 dan generasi maksimum 2000, dengan nilai *fitness* 5,4793e-07,dengan jumlah kontainer yang dibongkar muat sebesar 1825049 Teu's dan jalur sebagai berikut:
{Tg. Priok-Amamapare}, {Tg. Priok-Balikpapan}, {Tg.Priok-Banjarmasin}, {Tg. Priok-Batam}, {Tg. Priok-Batu Licin}, {Manokwari-Bau-bau}, {Tg. Priok-Benoa}, {Pontianak-Berau}, {Pontianak-Biak}, {Timika-Cigading}, {Tg. Priok-Jamuang}, {Pontianak-Jayapura}, {Tual-Kaimana}, {Tg. Priok-Kendari}, {Bitung-Kijang}, {Tg. Priok-Bitung}, {Tg. Priok-Kupang}, {Belawan-Labu}, {Tg. Priok-Belawan}, {Tg. Priok-Lampung}, {Toli-toli-Manokwari}, {Tg. Priok-Merak}, {Tg. Priok-Merauke}, {Tg. Priok-Nabire}, {Tg. Priok-Padang}, {Tg. Priok-Palembang}, {Tg. Priok-Pangkal Balam}, {Tg. Priok-Pekanbaru}, {Tg. Priok-Perawang}, {Tg. Priok-Samarinda}, {Nunukan-Sampit}, {Ambon-Nunukan}, {Bontang, Ambon}, {Makassar-Bontang}, {Malili- Sei Pakning}, {Fak-fak-Malili}, {Tg. Priok-Fak-fak}, {Palu-Serui}, {Tg. PriokPalu}, {Tg. Perak-Tarakan}, {Tg. Priok-Tg. Emas}, {Makassar- Tg. Perak}, {Makassar-Toli-toli}, {Makassar-Tual}, {Tg. Priok-Makassar}, {Pontianak-Tg. Priok}, {Pontianak-Sabau}, {Sorong-Pontianak}, {Timika-Sorong}, {Jambi-Timika}, {Jambi-Mimika}. Dari jalur yang didapat, dimodelkan dalam bentuk graf yang dapat ditunjukkan pada Gambar 5.5.

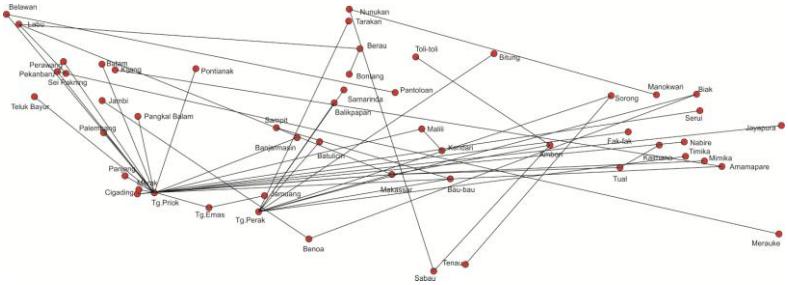


Gambar 5.5 Model graf *Tree* pada data tahun 2011

Dari Gambar 5.5, dapat ditunjukkan bahwa semua pelabuhan terhubung dan memiliki arus yang terpadat diantara kedua belas pengujian yang telah dilakukan.

5.6 Hasil dari Pengujian dengan Data pada Tahun 2012

Dari pengujian dengan data pada tahun 2012, didapat hasil bahwa jalur dengan arus pergerakan paling padat terdapat setelah pengujian C4, yaitu pada ukuran populasi ke 100 dan generasi maksimum 2000, dengan nilai *fitness* 4,93131e-07,dengan jumlah kontainer yang dibongkar muat sebesar 2027860 Teu's dan jalur sebagai berikut:
{Tg. Perak-Balikpapan}, {Tg. Priok-Banjarmasin}, {Tg. Priok-Batam}, {Tg. Perak-Batu Licin}, {Tg. Perak-Bitung}, {Berau-Bontang}, {Labu-Berau}, {Tg. Priok-Fak-fak}, {Benoa-Jambi}, {Biak-Benoa}, {Tg. Perak-Biak}, {Tg. Emas-Jamuang}, {Cigading-Jayapura}, {Makassar-Cigading}, {Tual-Kaimana}, {Malili-Kendari}, {Amamapare-Kijang}, {Tg. Priok-Amamapare}, {Sorong-Kupang}, {Tg. Priok-Lampung}, {Tg. Priok-Malili}, {Nunukan-Manokwari}, {Tg. Priok-Merak}, {Pekanbaru-Merauke}, {Tg. Priok-Nabire}, {Sabau-Nunukan}, {Tg. Priok-Padang}, {Tg. Priok-Palembang}, {Belawan-Palu}, {Tg. Priok-Belawan}, {Tg. Priok-Pangkal Balam}, {Tg. Priok-Pekanbaru}, {Tg. Priok-Perawang}, {Tg. Priok-Pontianak}, {Tg. Perak-Samarinda}, {Bau-bau-Sampit}, {Tg. Perak-Bau-bau}, {Labu-Sei Pakning}, {Makassar-Labu}, {Tg. Priok-Serui}, {Tg. Perak-Sorong}, {Tg. Perak-Tarakan}, {Tg. Priok-Tg. Emas}, {Makassar-Tg. Priok}, {Tg. Perak-Timika}, {Makassar-Tg. Perak}, {Ambon-Toli-toli}, {Makassar-Tual}, {Ambon-Sabau}, {Makassar-Ambon}, {Makassar-Mimika}. Dari jalur yang didapat, dimodelkan dalam bentuk graf yang dapat ditunjukkan pada Gambar 5.6.



Gambar 5.6 Model graf *Tree* pada data tahun 2012

Dari Gambar 5.6, dapat ditunjukkan bahwa semua pelabuhan terhubung dan memiliki arus yang terpadat diantara kedua belas pengujian yang telah dilakukan.

“Halaman Sengaja dikosongkan”

BAB VI
PENUTUP

Bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan. Disamping itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan eksperimen dan pembahasan terhadap hasil pengujian yang telah dilakukan terhadap sistem penentuan pola jaringan pergerakan logistik dengan menggunakan metode *Minimum Spanning Tree* berbasis algoritma genetika, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode *Minimum Spanning Tree* berbasis algoritma genetika menghasilkan pola pergerakan logistik dengan arus terpadat untuk keseluruhan pelabuhan yang direpresentasikan dengan node.
2. Untuk data pada tahun 2010 menghasilkan rute dengan nilai *fitness* 6,06834e-07 pada percobaan dengan besar populasi ke 100 dan jumlah generasi 2000. Untuk data pada tahun 2011 menghasilkan rute dengan nilai *fitness* 5,4793e-07 pada percobaan dengan besar populasi ke 100 dan jumlah generasi 2000. Untuk data pada tahun 2012 menghasilkan rute dengan nilai *fitness* 4,93131e-07 pada percobaan dengan besar populasi ke 100 dan jumlah generasi 2000.
3. Untuk data pada tahun 2010 menghasilkan rute yang optimal dengan arus pergerakan kontainer sebesar 1647896 Teu's. Untuk data pada tahun 2011 menghasilkan rute yang optimal dengan arus pergerakan kontainer sebesar 1825049 Teu's. Untuk data pada tahun 2012 menghasilkan rute yang optimal dengan arus pergerakan kontainer sebesar 2027860 Teu's.

6.2 Saran

Dengan melihat hasil yang dicapai pada penelitian ini, ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu:

1. Dalam pengujian ini hanya menggunakan data tahun 2010, 2011, dan 2012. Oleh karena itu diharapkan program ini dapat melakukan *input* data tahun berikutnya untuk analisis lebih lanjut mengenai jalur terpadat.
2. Dalam pengujian diperlukan adanya penambahan ukuran populasi dan ukuran generasi maksimum yang diharapkan dapat menambah keakuratan dalam menentukan hasil.

DAFTAR PUSTAKA

[1] Iqbal, A. (2014). ”*Menko Maritim: Tol Laut Bicara Masalah Pelabuhan, Pelayaran, dan Galangan Kapal Nasional*”. http://jurnalmaritim.com/2014/12/menko-maritim-tol-laut-bicara-masalah-pelabuhan-pelayaran-dan-galangan-kapal-nasional/. Diakses pada tanggal 16-02-2015.

[2] Prima, I. T. (2015). ”*Menurunkan Biaya Logistik*”. http://m.sindonews.com/read/959413/161/menurunkan-biaya-logistik-1422950158. Diakses pada tanggal 16-02-2015.

[3] Ariyanti, F. (2014). “*Tol Laut Sudah Direncanakan pada Era SBY?*”. http://bisnis.liputan6.com/read/2066576/tol-laut-sudah-direncanakan-pada-era-sby. Diakses pada tanggal 17-02-2015.

[4] Jo, J. Li, Y. Gen, M. (2007). “*Nonlinear Fixed Charge Transportation Problem by Spanning Tree-based Genetic Algorithm*”. Science Direct Computers & Industrial Engineering Vol. 53, Hal. 290-298.

[5] Syarif, A. Yun, Y. Gen, M. (2002). “*Study on Multi-Stage Logistic Chain Network: A Spanning Tree-based Genetic Algorithm Approach*”. ScienceDirect Computers & Industrial Engineering Vol. 43, Hal. 299-314.

[6] Ross, S. M. (2004). “*Topics in Finite and Dicrete Mathematics*”. United Kingdom: Cambridge University Press.

[7] Wu, B. Y. Chao, K. (2003). "*Spanning Trees and Optimization Problems*". Washington D.C. : CRC Press.

[8] Munir, R. (2010). "MATEMATIKA DISKRIT". Bandung: INFORMATIKA Bandung.

[9] Haupt, R. L. Haupt, S. E. (2004). "*Practical Genetic Algorithms Second Edition*". Canada: John Wiley & Sons, Inc.

[10] Sivanandam, S. N. Deepa, S. N. (2008). "Introduction to Genetic Algorithms". New York: Springer Science+Business Media.

[11] Peraturan Pemerintah No. 69 Tahun 2001 tentang Kepelabuhan.

[12] Indonesia (2015). "*Peta Sebaran Pelabuhan Indonesia*". http://gis.dephub.go.id/Metadata/images/metadata/pelabuhan_view.jpg. Diakses pada tanggal 16-02-2015.

[13] Indonesia (2015). "Data Prasarana Pelabuhan". http://gis.dephub.go.id/mappingf/Prasarana/Pelabuhan/PelabuhanList.aspx#. Diakses pada tanggal 16-02-2015.

[14] Sangian, R. (2014). "Pendulum Nusantara & Alur Laut Kepulauan Indonesia". http://supplychainindonesia.com/new/pendulum-nusantara-alur-laut-kepulauan-indonesia/. Diakses pada tanggal 20-02-2015

LAMPIRAN A

***Source Code class* Home.java**

```
package MSTGA;
import java.sql.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Random;
/**
 *
 * @author CakGan
 */
public class Home extends javax.swing.JFrame {
    int POP_SIZE = 5;             // population
size
    int MAX_ITER =5;              // max number
of iterations
    double RASIO_MUTASI = 0.4;    // probability
of mutation
    double RASIO_CROSSOVER = 0.2;
    int tahun = 2012;

    String PC0,PM0,pop_size0,max_gen0;
    Double PC,PM;
    Integer Pop_size,Max_gen;
    /**
     * Creates new form Home
     */

    public Home() {
        initComponents();

Pc.setText(String.valueOf(RASIO_CROSSOVER));
        System.out.println(RASIO_CROSSOVER);

Pm.setText(String.valueOf(RASIO_MUTASI));
        System.out.println(RASIO_MUTASI);

pop_size.setText(String.valueOf(POP_SIZE));
```

LAMPIRAN A (LANJUTAN)

```
        System.out.println(POP_SIZE);

max_gen.setText(String.valueOf(MAX_ITER));
        System.out.println(MAX_ITER);
    }

    public void tambah(){
    //SQL
        try{
            String host =
"jdbc:derby://localhost:1527/DB_port";
            String uName = "sa";
            String uPass = "sa";
            Connection con =
DriverManager.getConnection(host, uName, uPass);
            System.out.println("Berhasil");
            Statement
stmt=con.createStatement();
            Random rnd = new Random();

            //Populasi
            ArrayList<Populasi> pop1 = new
ArrayList<>();
            for (int a = 0; a < POP_SIZE; a++) {
                prufer c = new prufer();
                Populasi p = new Populasi();
                //generate p
                c.prufer1();
                //generate q
                c.Qi();
                //pasangkan
                c.pasang();
                ArrayList<PQ> pasangan;
                pasangan = c.getPasangan();
                double totalbobot = 0;
                double w = 0;
                for (int i = 0; i <
pasangan.size(); i++) {
```

LAMPIRAN A (LANJUTAN)

```
                    int baris =
pasangan.get(i).getP();
                    int kolom =
pasangan.get(i).getQ();
                    if(tahun==2010){
                    String sql1 = "SELECT
JUMLAH_BM10 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                    int id1 =
rs1.getInt("JUMLAH_BM10");
                    w += id1;
                    }
                    totalbobot = 1/w;
                    }
                    if(tahun==2011){
                    String sql1 = "SELECT
JUMLAH_BM11 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                    int id1 =
rs1.getInt("JUMLAH_BM11");
                    w += id1;
                    }
                    totalbobot = 1/w;
```

LAMPIRAN A (LANJUTAN)

```
                    }
                    if(tahun==2012){
                    String sql1 = "SELECT
JUMLAH_BM12 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                    int id1 =
rs1.getInt("JUMLAH_BM12");
                    w += id1;
                    }
                    totalbobot = 1/w;
                    }
                }
                p.setP(c.getP());
                p.setTB(totalbobot);
                p.setArus(w);
                pop1.add(p);
            }
        double totalfitness = 0;
        for (int i = 0; i < pop1.size(); i++) {
            ArrayList p = pop1.get(i).getP();
            double bobot = pop1.get(i).getTB();
            System.out.print("Pop ke-" + (i + 1)
+ " P: ");
            for (int j = 0; j < p.size(); j++) {
                System.out.print(p.get(j) + "
");
            }
            totalfitness += bobot;
            System.out.println("Bobot: " +
bobot);
        }
```

LAMPIRAN A (LANJUTAN)

```
        int z=1;
        while(z<=MAX_ITER){

        //crossover
        ArrayList<Populasi> parent = new
ArrayList();
        ArrayList<Populasi> child = new
ArrayList();

        for (int i = 0; i < pop1.size(); i++) {
            if (Math.random() < RASIO_CROSSOVER)
{
                parent.add(pop1.get(i));
            }
        }
        while (parent.size() % 2 != 0 ||
parent.isEmpty()) {
            parent.clear();
            for (int i = 0; i < pop1.size();
i++) {
                if (Math.random() <
RASIO_CROSSOVER) {
                    parent.add(pop1.get(i));
                }
            }

//System.out.println(pop1.get(0).getP().size());
        }
        for (int i = 0; i < parent.size(); i+=2)
{

System.out.println(parent.get(i).getP());

System.out.println(parent.get(i+1).getP());
            int tipot =
rnd.nextInt(pop1.get(0).getP().size());
```

LAMPIRAN A (LANJUTAN)

```
            System.out.println(tipot);
            ArrayList temp1 = (ArrayList)
parent.get(i).getP();
            ArrayList temp2 = (ArrayList)
parent.get(i+1).getP();

            ArrayList p1=new ArrayList();
            ArrayList p2=new ArrayList();
            for(int j=0;j<tipot;j++){
                p1.add(temp1.get(j));
                p2.add(temp2.get(j));
            }
            for(int
k=tipot;k<pop1.get(0).getP().size();k++){
                p1.add(temp2.get(k));
                p2.add(temp1.get(k));
            }
            Populasi a=new Populasi();
            a.setP(p1);
            Populasi b=new Populasi();
            b.setP(p2);
            child.add(a);
            child.add(b);

System.out.println(child.get(i).getP());

System.out.println(child.get(i+1).getP()+"\n");
        }
        //mutasi
        ArrayList<Populasi> mutasi =new
ArrayList();
        for(int j=0; j<child.size();j++){
            Populasi childtemp = new Populasi();
            ArrayList child2 = new ArrayList();
            Object child1 [];
            child1 =
child.get(j).getP().toArray();
```

LAMPIRAN A (LANJUTAN)

```
            for(int
k=0;k<child.get(j).getP().size();k++){
                if (Math.random() <
RASIO_MUTASI) {

if(k!=child.get(j).getP().size()-1){
                        int temp =
(int)child1[k];
                        child1[k] = child1[k+1];
                        child1[k+1]=temp;
                    }
                }
                child2.add(k, child1[k]);
            }
            childtemp.setP(child2);
            mutasi.add(childtemp);

System.out.println(mutasi.get(j).getP());
        }
        System.out.println(mutasi.size());
        //generate populasi baru
        double idbot[][] = new
double[pop1.size()][2];
        for (int i = 0; i < pop1.size(); i++) {
            ArrayList h = pop1.get(i).getP();
            double bobot = pop1.get(i).getTB();
            idbot[i][0] = i;
            idbot[i][1] = bobot;
        }
        /*for (int i = 0; i < pop1.size(); i++)
{
            System.out.println(idbot[i][0] + " "
+ idbot[i][1]);
        }*/
        //sort bobot terbesar
        System.out.println();
```

LAMPIRAN A (LANJUTAN)

```
        Arrays.sort(idbot, new
Comparator<double[]>() {
            @Override
            public int compare(double[] t1,
double[] t) {
                return Double.compare(t[1],
t1[1]);
            }
        });
        /*for (int i = 0; i < pop1.size(); i++)
{
        System.out.println(idbot[i][0] + " " +
idbot[i][1]);
        }*/
        for (int i = 0; i < pop1.size(); i++) {
            if(i<mutasi.size()){

pop1.remove((int)idbot[i][0]);
                    pop1.add((int)idbot[i][0],
mutasi.get(i));
            }
        }

            for(int i=0;i<pop1.size();i++){
            prufer c = new prufer();
            Populasi p = new Populasi();
            c.arraygen(pop1.get(i).getP());
            c.Qi();
            c.pasang();
            ArrayList<PQ> pasangan;
            pasangan = c.getPasangan();
            double totalbobot = 0;
            double w = 0;
            for (int j = 0; j < pasangan.size();
j++) {
                    int baris =
pasangan.get(j).getP();
```

LAMPIRAN A (LANJUTAN)

```
                    int kolom =
pasangan.get(j).getQ();
                    if(tahun==2010){
                    String sql1 = "SELECT
JUMLAH_BM10 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                        int id1 =
rs1.getInt("JUMLAH_BM10");
                        w += id1;
                    }
                    totalbobot = 1/w;
                    }
                    if(tahun==2011){
                    String sql1 = "SELECT
JUMLAH_BM11 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                        int id1 =
rs1.getInt("JUMLAH_BM11");
                        w += id1;
                    }
                    totalbobot = 1/w;
                    }
                    if(tahun==2012){
```

LAMPIRAN A (LANJUTAN)

```
                    String sql1 = "SELECT
JUMLAH_BM12 FROM ARUS_BM WHERE ID_ASAL = ? AND
ID_TUJUAN = ? ";
                    PreparedStatement pre1 =
con.prepareStatement(sql1);
                    pre1.setInt(1, baris);
                    pre1.setInt(2, kolom);
                    ResultSet rs1=
pre1.executeQuery();
                    while(rs1.next()){
                        int id1 =
rs1.getInt("JUMLAH_BM12");
                        w += id1;
                    }
                    totalbobot = 1/w;
                    }
                }
                p.setP(c.getP());
                p.setTB(totalbobot);
                p.setArus(w);
                pop1.set(i, p);
            }
        totalfitness = 0;
        for (int i = 0; i < pop1.size(); i++) {
            ArrayList m = pop1.get(i).getP();
            double bobot = pop1.get(i).getTB();
            double arus = pop1.get(i).getArus();
            System.out.print("Pop ke-" + (i + 1)
+ " P: ");
            for (int j = 0; j < m.size(); j++) {
                System.out.print(m.get(j) + "
");
            }
            totalfitness += bobot;
            System.out.println("Bobot: " +
bobot+" arus: "+arus);
        }
```

LAMPIRAN A (LANJUTAN)

```
        System.out.println("Total: " +
totalfitness);

        //sort bobot terbesar
        double idbot1[][] = new
double[pop1.size()][2];
        for (int i = 0; i < pop1.size(); i++) {
            ArrayList b = pop1.get(i).getP();
            double bobot = pop1.get(i).getTB();
            idbot1[i][0] = i;
            idbot1[i][1] = bobot;
        }
        /*for (int i = 0; i < pop1.size(); i++)
{
            System.out.println(idbot1[i][0] + "
" + idbot1[i][1]);
        }*/
        System.out.println();
        Arrays.sort(idbot1, new
Comparator<double[]>() {
            @Override
            public int compare(double[] t1,
double[] t) {
                return Double.compare(t[1],
t1[1]);
            }
        });
        /*for (int i = 0; i < pop1.size(); i++)
{
        System.out.println(idbot1[i][0] + " " +
idbot1[i][1]);
        }*/
        int min = (int)idbot1[pop1.size()-1][0];
        System.out.println("Kromosom Min
=\n"+"Pop ke-
"+(min+1)+pop1.get(min).getP()+"\nBobot:
"+pop1.get(min).getTB());
```

LAMPIRAN A (LANJUTAN)

```
        z++;
        prufer c = new prufer();
        c.arraygen(pop1.get(min).getP());
        c.Qi();
        c.pasang();
        ArrayList<PQ> pasangan;
        pasangan = c.getPasangan();
        if(z==MAX_ITER+1){
            int [][] koordinat=new int[2][102];
            int [] kx=new int[52];
            int [] ky=new int[52];
            for (int i=1;i<=52;i++){
               String sql1 = "SELECT * FROM
PELABUHAN WHERE ID_PELABUHAN = ?";
               PreparedStatement pre1 =
con.prepareStatement(sql1);
               pre1.setInt(1, i);
               ResultSet rs1=
pre1.executeQuery();
               while(rs1.next()){
                int x =
rs1.getInt("KOORDINATX");
                int y =
rs1.getInt("KOORDINATY");
                kx[i-1]=x;
                ky[i-1]=y;
               }
            }
            for(int i=0;i<pasangan.size();i++){
               System.out.println("Pasangan :
"+pasangan.get(i).getP()+" -
"+pasangan.get(i).getQ());
               String sql3 = "SELECT * FROM
PELABUHAN WHERE ID_PELABUHAN = ?";
               PreparedStatement pre1 =
con.prepareStatement(sql3);
               pre1.setInt(1,
pasangan.get(i).getP());
```

LAMPIRAN A (LANJUTAN)

```
                String sql4 = "SELECT * FROM
PELABUHAN WHERE ID_PELABUHAN = ?";
                PreparedStatement pre2 =
con.prepareStatement(sql4);
                pre2.setInt(1,
pasangan.get(i).getQ());
                ResultSet rs1=
pre1.executeQuery();
                ResultSet rs2=
pre2.executeQuery();
                while(rs1.next()&&rs2.next()){
                 String id1 =
rs1.getString("NAMA_PELABUHAN");
                 String id2 =
rs2.getString("NAMA_PELABUHAN");
                 int x1 =
rs1.getInt("KOORDINATX");
                 int x2 =
rs2.getInt("KOORDINATX");
                 int y1 =
rs1.getInt("KOORDINATY");
                 int y2 =
rs2.getInt("KOORDINATY");
                 System.out.println("Pasangan :
"+id1+" - "+id2);
                 koordinat[0][2*i]=x1;
                 koordinat[0][2*i+1]=x2;
                 koordinat[1][2*i]=y1;
                 koordinat[1][2*i+1]=y2;
                 }
              }
              System.out.println("Dengan Arus
sebanyak: "+pop1.get(min).getArus());
              Graph simulasi=new Graph(kx, ky, 52,
koordinat);
              simulasi.setVisible(true);
              }
```

LAMPIRAN A (LANJUTAN)

```
        }
        }
        catch (SQLException err){

System.out.println(err.getMessage());
        }
    }

    /**
     * This method is called from within the
constructor to initialize the form.
     * WARNING: Do NOT modify this code. The
content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        BM_2010 = new javax.swing.JButton();
        BM_2011 = new javax.swing.JButton();
        BM_2012 = new javax.swing.JButton();
        pop_size = new javax.swing.JTextField();
        max_gen = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        Pc = new javax.swing.JTextField();
        Pm = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
```

LAMPIRAN A (LANJUTAN)

```
        jLabel16 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConst
ants.EXIT_ON_CLOSE);

        jLabel1.setText("Ukuran Populasi");

        BM_2010.setText("Arus 2010");
        BM_2010.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt)
{
                BM_2010ActionPerformed(evt);
            }
        });

        BM_2011.setText("Arus 2011");
        BM_2011.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt)
{
                BM_2011ActionPerformed(evt);
            }
        });

        BM_2012.setText("Arus 2012");
        BM_2012.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt)
{
                BM_2012ActionPerformed(evt);
            }
```

LAMPIRAN A (LANJUTAN)

```
        });

        pop_size.setText("jTextField1");
        pop_size.setName("tfPopulasi"); //
NOI18N
        pop_size.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt)
{
                pop_sizeActionPerformed(evt);
            }
        });

        max_gen.setText("jTextField2");

        jLabel2.setText("Max Generation");

        jLabel3.setText("Crossover Rate");

        Pc.setText("jTextField3");
        Pc.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt)
{
                PcActionPerformed(evt);
            }
        });

        Pm.setText("jTextField4");

        jLabel4.setText("Mutation Rate");

        jLabel10.setText("PENENTUAN POLA
JARINGAN PERGERAKAN \n");
```

LAMPIRAN A (LANJUTAN)

```
        jLabel9.setText("LOGISTIK YANG OPTIMAL
PADA TRANSPORTASI LAUT");

        jLabel11.setText("MENGGUNAKAN MINIMUM
SPANNING TREE BERBASIS ALGORITMA GENETIKA");

        jLabel12.setText("OLEH: RIFDY FACHRY
(1211100013)");

        jLabel13.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/M
STGA/its.png"))); // NOI18N
        jLabel13.setText("jLabel13");

        jLabel14.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/M
STGA/mathits.png"))); // NOI18N
        jLabel14.setText("jLabel14");

        jLabel15.setText("DOSEN PEMBIMBING:");

        jLabel16.setText("DR. IMAM MUKHLASH,
MT.");

        jLabel17.setText("DRS. SOETRISNO,
MI.KOMP.");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
                .addGap(38, 38, 38)
```

LAMPIRAN A (LANJUTAN)

```
                .addComponent(jLabel13,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
                                .addGap(18, 18,
18)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addComponent(jLabel11)

.addGroup(layout.createSequentialGroup()

.addGap(31, 31, 31)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(BM_2010)

.addGap(35, 35, 35)

.addComponent(BM_2011)

.addGap(31, 31, 31)
```

LAMPIRAN A (LANJUTAN)

```
.addComponent(BM_2012))

.addGroup(layout.createSequentialGroup()

.addGap(55, 55, 55)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addComponent(pop_size)

.addComponent(max_gen,
javax.swing.GroupLayout.PREFERRED_SIZE, 76,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(64, 64, 64)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel4)

.addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

LAMPIRAN A (LANJUTAN)

```
.addComponent(Pc)

.addComponent(Pm,
javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)))))))

.addGroup(layout.createSequentialGroup()
                                    .addGap(126,
126, 126)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addComponent(jLabel12)

.addGroup(layout.createSequentialGroup()

.addGap(26, 26, 26)

.addComponent(jLabel15))

.addGroup(layout.createSequentialGroup()

.addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addComponent(jLabel17)

.addComponent(jLabel16))))))

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 18, Short.MAX_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, layout.createSequentialGroup()
```

LAMPIRAN A (LANJUTAN)

```
.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, layout.createSequentialGroup()

.addComponent(jLabel9)
                                    .addGap(71, 71,
71))

.addGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, layout.createSequentialGroup()

.addComponent(jLabel10)
                                    .addGap(93, 93,
93)))))
                .addComponent(jLabel14,
javax.swing.GroupLayout.PREFERRED_SIZE, 98,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(32, 32, 32))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
```

LAMPIRAN A (LANJUTAN)

```
                    .addComponent(jLabel13)
                    .addComponent(jLabel14)

.addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel10)

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(jLabel15)

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)

.addComponent(jLabel16)))

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                .addComponent(jLabel17)
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jLabel3))
```

LAMPIRAN A (LANJUTAN)

```
.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                    .addComponent(pop_size,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(Pc,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jLabel4))

.addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                    .addComponent(max_gen,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(Pm,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
```

LAMPIRAN A (LANJUTAN)

```
.addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                    .addComponent(BM_2010)
                    .addComponent(BM_2011)
                    .addComponent(BM_2012))
                .addContainerGap(17,
Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void
pop_sizeActionPerformed(java.awt.event.ActionEve
nt evt) {
        // TODO add your handling code here:
    }

    private void
PcActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:

    }

    private void
BM_2010ActionPerformed(java.awt.event.ActionEven
t evt) {
        // TODO add your handling code here:
        PC0=Pc.getText().toString();
        RASIO_CROSSOVER =
Double.parseDouble(PC0);
        System.out.println(RASIO_CROSSOVER);
        PM0=Pm.getText().toString();
        RASIO_MUTASI = Double.parseDouble(PM0);
        System.out.println(RASIO_MUTASI);
        pop_size0=pop_size.getText().toString();
```

LAMPIRAN A (LANJUTAN)

```
        POP_SIZE = Integer.parseInt(pop_size0);
        System.out.println(POP_SIZE);
        max_gen0=max_gen.getText().toString();
        MAX_ITER = Integer.parseInt(max_gen0);
        System.out.println(MAX_ITER);
        tahun=2010;
        tambah();

    }

    private void
BM_2011ActionPerformed(java.awt.event.ActionEven
t evt) {
        // TODO add your handling code here:
        PC0=Pc.getText().toString();
        RASIO_CROSSOVER =
Double.parseDouble(PC0);
        System.out.println(RASIO_CROSSOVER);
        PM0=Pm.getText().toString();
        RASIO_MUTASI = Double.parseDouble(PM0);
        System.out.println(RASIO_MUTASI);
        pop_size0=pop_size.getText().toString();
        POP_SIZE = Integer.parseInt(pop_size0);
        System.out.println(POP_SIZE);
        max_gen0=max_gen.getText().toString();
        MAX_ITER = Integer.parseInt(max_gen0);
        System.out.println(MAX_ITER);
        tahun=2011;
        tambah();
    }

    private void
BM_2012ActionPerformed(java.awt.event.ActionEven
t evt) {
        // TODO add your handling code here:

        PC0=Pc.getText().toString();
```

LAMPIRAN A (LANJUTAN)

```
        RASIO_CROSSOVER =
Double.parseDouble(PC0);
        System.out.println(RASIO_CROSSOVER);
        PM0=Pm.getText().toString();
        RASIO_MUTASI = Double.parseDouble(PM0);
        System.out.println(RASIO_MUTASI);
        pop_size0=pop_size.getText().toString();
        POP_SIZE = Integer.parseInt(pop_size0);
        System.out.println(POP_SIZE);
        max_gen0=max_gen.getText().toString();
        MAX_ITER = Integer.parseInt(max_gen0);
        System.out.println(MAX_ITER);
        tahun=2012;
        tambah();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed"
desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6)
is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswi
ng/lookandfeel/plaf.html
         */
        try {
            for
(javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()
) {
                if
("Nimbus".equals(info.getName())) {
```

LAMPIRAN A (LANJUTAN)

```
javax.swing.UIManager.setLookAndFeel(info.getCla
ssName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Home.class.ge
tName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Home.class.ge
tName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Home.class.ge
tName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch
(javax.swing.UnsupportedLookAndFeelException ex)
{

java.util.logging.Logger.getLogger(Home.class.ge
tName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                new Home().setVisible(true);
            }
```

LAMPIRAN A (LANJUTAN)

```
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton BM_2010;
    private javax.swing.JButton BM_2011;
    private javax.swing.JButton BM_2012;
    private javax.swing.JTextField Pc;
    private javax.swing.JTextField Pm;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JTextField max_gen;
    public javax.swing.JTextField pop_size;
    // End of variables declaration
}
```

LAMPIRAN B

***Source Code class* prufer.java**

```
package MSTGA;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

/**
 *
 * @author CakGan
 */
public class prufer {
    int n=52;
    Random rand = new Random();

    ArrayList p=new ArrayList();
    ArrayList q=new ArrayList();
    ArrayList<PQ> pasangan=new ArrayList<>();

    public prufer(){}
    public void arraygen(ArrayList w){
        this.p=w;
    }
    public int gen(int index){
        return (int)p.get(index);
    }
    public void setgen(int index, int gene){
        p.add(index, gene);
    }
    public void prufer1(){
        for(int i=0; i<n-2; ++i) {
            int r=rand.nextInt(n)+1;
            this.setgen(i, r);
        }
    }

    public int Q(int index){
        return (int)q.get(index);
```

```
 }
 public void setQ(int index, int gene){
     q.add(index, gene);
 }
 public void Qi(){
     int y = 0;
     for(int i=1; i<=n;i++){
         if(p.contains(i)==false){
             //System.out.print(i+" ");
             this.setQ(y, i);
             y++;
         }
     }
}
public void sortQi(){
    Object temp[];
    temp=q.toArray();
    Arrays.sort(temp);
    q.clear();
    for(int i=0;i<temp.length;i++){
        q.add(i, temp[i]);
        //System.out.print(Q(i)+" ");
    }
}


public void pasang(){
    ArrayList p1=(ArrayList) p.clone();
    while(p1.size()!=0){
        int nP=(int)p1.get(0);
        int temp=1;
        for(int j=1;j<p1.size();j++){
            if(p1.get(j).equals(nP)){
                temp++;
            }
        }
        if(temp==1){
             int nQ=(int)q.get(0);
```

LAMPIRAN B (LANJUTAN)

```
                pasangan.add(new PQ(nP,nQ));
                p1.remove(0);
                q.remove(0);
                q.add(nP);
                sortQi();
            }
            else{
                int nQ=(int)q.get(0);
                pasangan.add(new PQ(nP,nQ));
                p1.remove(0);
                q.remove(0);
                sortQi();
            }
        }
        //p habis
        pasangan.add(new
PQ((int)q.get(0),(int)q.get(1)));
    }

    public ArrayList<PQ> getPasangan(){
        return pasangan;
    }
    public ArrayList getP(){
        return p;
    }


}
```

“Halaman Sengaja dikosongkan”

LAMPIRAN C

***Source Code class* Populasi.java**

```
package MSTGA;

import java.util.ArrayList;

public class Populasi {
    private       ArrayList<Integer>       p=new
ArrayList();
    private double totalbobot;
    private double arus;

    public void setP(ArrayList p){
        this.p=p;
    }
    public void setTB(double val){
        totalbobot=val;
    }
    public ArrayList getP(){
        return p;
    }
    public double getTB(){
        return totalbobot;
    }
    public void setArus(double val){
        arus=val;
    }
    public double getArus(){
        return arus;
    }
}
```

"Halaman Sengaja dikosongkan"

LAMPIRAN D

***Source Code class* PQ.java**

```
package MSTGA;

public class PQ {
    private int p;
    private int q;

    public PQ(int a,int b){
        p=a;
        q=b;
    }
    public void setP(int val){
        p=val;
    }
    public void setQ(int val){
        q=val;
    }
    public int getP(){
        return p;
    }
    public int getQ(){
        return q;
    }
}
```

“Halaman Sengaja dikosongkan”

LAMPIRAN E

***Source Code class* Graph.java**

```
package MSTGA;
import java.awt.*;
import javax.swing.*;

class SurfaceAGB extends JPanel{

    private int[] X;
    private int[] Y;
    private int[][] XY;
    private int Jumlah;

    public SurfaceAGB (int[] x, int[] y, int
jumlah, int[][] xy){
        X = new int [jumlah];
        Y = new int [jumlah];
        XY = xy;
        X = x;
        Y = y;
        Jumlah = jumlah;
    }
    private void doDrawing(Graphics g){
        Graphics g2d = (Graphics2D) g;
        for (int i = 0 ; i < Jumlah ; i++){
            g2d.drawLine(X[i],Y[i],X[i],Y[i]);
            g2d.drawLine(X[i]-1,Y[i],X[i]-
1,Y[i]);

g2d.drawLine(X[i]+1,Y[i],X[i]+1,Y[i]);
            g2d.drawLine(X[i],Y[i]-1,X[i],Y[i]-
1);

g2d.drawLine(X[i],Y[i]+1,X[i],Y[i]+1);

            g2d.drawLine(X[i]-1,Y[i]-1,X[i]-
1,Y[i]-1);
            g2d.drawLine(X[i]-1,Y[i]+1,X[i]-
1,Y[i]+1);
```

LAMPIRAN E (LANJUTAN)

```
g2d.drawLine(X[i]+1,Y[i]+1,X[i]+1,Y[i]+1);
            g2d.drawLine(X[i]+1,Y[i]-
1,X[i]+1,Y[i]-1);
        }

        for (int i = 0 ; i < 51 ; i++){

g2d.drawLine(XY[0][2*i],XY[1][2*i],XY[0][2*i+1],
XY[1][2*i+1]);
        }
    }
    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        doDrawing(g);
    }
}
public class Graph extends JFrame{
    public Graph(int[] x, int[] y, int jumlah,
int[][] xy) {
        initUI(x, y, jumlah, xy);
    }

    private void initUI(int[] x, int[] y, int
jumlah, int[][] xy){
        add(new SurfaceAGB(x, y, jumlah, xy));

        setTitle("Hasil Minimum Spanning Tree
Berbasis Algoritma Genetika");
        setSize(1366, 500);
        setLocationRelativeTo(null);
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

BIODATA PENULIS

Penulis lahir di Gresik, 19 Mei 1993. Merupakan anak ke-2 dari pasangan M. Syariful Hari dan Zulfa. Penulis menjalani masa studi di TK Muslimat 29 Mahkota Gresik, kemudian melanjutkan ke jenjang sekolah dasar di MINU Salafiyyah Gresik. Tahun 2006 melanjutkan sekolah menengah di SMP N 1 Gresik. Tahun 2008 melanjutkan sekolah menengah atas di SMA NU 1 Gresik. Tahun 2011 penulis diterima di Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam ITS. Selama di bangku kuliah penulis aktif di kegiatan Organisasi intra kampus baik sebagai pengurus himpunan maupun kepanitiaan lomba berskala Nasional. Pada tahun kedua dan ketiga penulis menjadi anggota aktif HIMATIKA dibawah naungan Departemen Kesejahteraan Mahasiswa sekaligus menjadi Penanggung Jawab Official Website Olimpiade Matematika ITS serta penanggung jawab Publikasi dan Dokumentasi pada tahun berikutnya. Penulis dapat dihubungi melalui email: rifdyf@gmail.com.