TESIS – SM 142501

PENENTUAN JARINGAN LOGISTIK PADA TRANSPORTASI LAUT MENGGUNAKAN *FUZZY C-MEANS* DAN *MINIMUM SPANNING TREE* BERBASIS *HYBRID GENETIC ALGORITHM*

Shinta Tri Kismanti
1214 201 032

DOSEN PEMBIMBING
Dr. Imam Mukhlash, S.Si., MT

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS – SM 142501

DETERMINING OF LOGISTIC NETWORK ON SEA TRANSPORTATION USING FUZZY C-MEANS AND MINIMUM SPANNING TREE BASED HYBRID GENETIC ALGORITHM

Shinta Tri Kismanti
1214 201 032

Supervisor
Dr. Imam Mukhlash, S.Si., MT

MASTER'S DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

PENENTUAN JARINGAN LOGISTIK PADA TRANSPORTASI LAUT MENGGUNAKAN *FUZZY C-MEANS* DAN *MINIMUM SPANNING TREE* BERBASIS *HYBRID GENETIC ALGORITHM*

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember

oleh:
SHINTA TRI KISMANTI
NRP. 1214 201 032

Tanggal Ujian : 19 Juli 2016
Periode Wisuda : September 2016

Disetujui oleh:

1. **Dr. Imam Mukhlash, S.Si., MT.** **(Pembimbing)**
   NIP. 19700831 199403 1 003

2. **Prof. Dr. Mohammad Isa Irawan, MT.** **(Penguji)**
   NIP. 19631225 198903 1 001

3. **Dr. Budi Setiyono, S.Si., MT.** **(Penguji)**
   NIP. 19720207 199702 1 001

4. **Dr. Dwi Ratna Sulistyaningrum, S.Si., MT.** **(Penguji)**
   NIP. 19690405 199403 2 003

Direktur Program Pascasarjana,

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 19601202 198701 1 001

PENENTUAN JARINGAN LOGISTIK PADA TRANSPORTASI LAUT MENGGUNAKAN *FUZZY C-MEANS* DAN *MINIMUM SPANNING TREE* BERBASIS *HYBRID GENETIC ALGORITHM*

Nama Mahasiswa	:	Shinta Tri Kismanti
Mahasiswa ID	:	1214201032
Pembimbing	:	Dr. Imam Mukhlash, S.Si., M.T

ABSTRAK

Indonesia sebagai negara kepulauan dengan lebih dari 17.000 pulau dengan wilayah perairan menjadi salah satu moda transportasinya. Dengan demikian sebagian besar aktivitas terjadi di wilayah perairan, diantaranya distribusi logistik. Pergerakan distribusi logistik tersebut akan menghasilkan pola rute suatu pergerakan logistik. Penentuan pola jaringan pergerakan logistik yang optimal dapat mendukung kelancaran dalam sistem pendistribusian. Pada penelitian ini penentuan pola jaringan logistik dilakukan dengan dua tahapan. Tahap pertama, akan dilakukan pengklasteran dengan menggunakan *Fuzzy C-means*, pengklasteran bertujuan untuk mendapatkan kelompok-kelompok pulau yang berada pada lokasi yang berdekatan. Tahap kedua setelah didapatkan hasil *cluster* yang optimal, dengan menggunakan *minimum spanning tree* berbasis *hybrid genetic algorithm* akan didapatkan pola jaringan yang optimal. Pola jaringan tersebut akan menghubungkan pulau yang terpilih sebagai titik pusat ke pulau-pulau disekitarnya. Hasil uji coba pada proses pengklasteran menggunakan FCM didapatkan jumlah *cluster* optimal sebanyak 3 *cluster*. Pada proses pembentukkan MST berbasis *hybrid* GA digunakan parameter *crossover rate* 0,2 dan *mutation rate* 0,4 dan diperoleh hasil terbaik ketika iterasi minimumnya konvergen pada solusi optimal. C*luster* 1 didapatkan hasil terbaik dengan ukuran populasi 100 dan generasi maksimum 2000 dengan nilai *fitness* yaitu 9.41, *cluster* 2 dengan ukuran populasi 100 dan generasi maksimum 1000 dengan nilai *fitness* yaitu 14.97, dan *cluster* 3 dengan ukuran populasi 100 dan generasi maksimum 1000 dengan nilai *fitness* yaitu 17.46.

Kata kunci: Logistik, *Minimum Spanning Tree*, *Fuzzy C-means*, *Hybrid Genetic Algorithm*

DETERMINING OF LOGISTIC NETWORK ON SEA TRANSPORTATION USING FUZZY C-MEANS AND MINIMUM SPANNING TREE BASED HYBRID GENETIC ALGORITHM

Name	:	Shinta Tri Kismanti
Student Identity Number	:	1214201032
Supervisor	:	Dr. Imam Mukhlash, S.Si., M.T

ABSTRACT

Indonesia as the island nation with more than 17.000 islands the the territorial waters as one of the routes of transportation. For this condition, most of activities are conducted in marine waters, such as logistic distribution. The movement of logistics distribution will result in a movement pattern of the logistics. Determining the pattern of movement of the logistics network that can support optimal smoothness in the distribution system. In this study, determining the pattern of the logistics network is done in two stages. The first stage, to be carried out clustering using Fuzzy C-Means, clustering aims to get the island groups that are at a nearby location. The second stage after the results obtained optimal cluster, using the minimum spanning tree genetic algorithm-based hybrid will get the optimal network pattern. The pattern of the network will connect the island was chosen as the central point to the nearby islands. The results of trials on the process of clustering using FCM obtain optimal cluster number as many as three clusters. In the process of formation of the MST-based hybrid GA used parameter crossover rate and mutation rate 0.2 0.4 and obtained the best results when the minimum iteration converges toward the optimal solution. Cluster 1 obtained the best results with a population size of 100 and a maximum generation in 2000 with a fitness value is 9.41, cluster 2 with a population size of 100 and a maximum generation in 1000 with a value of fitness is 14.97 and cluster 3 with a population size of 100 and a maximum generation in 1000 with a value of fitness, namely 17.46.

Keywords: Logistic, Minimum Spanning Tree, Fuzzy C-means, Hybrid Genetic Algorithm

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Berdasarkan kondisi geografis, Indonesia terdiri lebih dari 17.000 (tujuh belas ribu) pulau yang terbentang sepanjang 1/8 (satu per delapan) garis khatulistiwa. Salah satunya adalah propinsi Maluku yang merupakan daerah kepulauan dengan jumlah pulau yang diperkirakan sekitar ± 559 buah. Dengan rasio perairan wilayah yang dominan dibandingkan luasan daratannya (*inland*) menjadikan propinsi Maluku merupakan wilayah dengan sebaran kepulauan yang dominan. Sehingga dapat dinyatakan jika Maluku sebagai propinsi kepulauan terbesar di Indonesia dengan kekuatan maritim dan kelautan yang khas dan dominan bagi Indonesia secara umum dan di wilayah timur Indonesia secara khusus [2].

Pertumbuhan dan penyebaran aktivitas ekonomi di Propinsi Maluku saat ini terpusat di kota Ambon sebagai wilayah transit dan wilayah jasa potensial. Namun situasi ini memberikan konsekuensi jarak yang dalam perkembangannya menjadi item kelemahan bagi wilayah lain di Maluku khususnya dengan keberadaan wilayah-wilayah terbelakang dan terisolasi. Karenanya dengan kondisi wilayah yang relatif berjauhan ini membutuhkan sistem transportasi laut yang efektif dalam arti tingkat ketersediaan yang tinggi dan waktu tempuh yang relatif cepat menjadi kebutuhan bagi wilayah Maluku [6].

Banyaknya aktivitas yang dilakukan diwilayah laut, akan diperoleh pola-pola rute yang dapat dilalui untuk distribusi logistik. Oleh karena itu diperlukan suatu analisa untuk menentukan pola rute yang optimal. Dalam analisa tersebut terdapat beberapa metode untuk menentukan pola rute yang optimal untuk pergerakan logistik di wilayah Maluku. Berdasarkan letak geografis kepulauan Maluku akan dilakukan pengelompokan yang bertujuan untuk mendapatkan kelompok-kelompok pulau-pulau yang berada pada lokasi yang berdekatan.Oleh karena itu, dibutuhkan suatu penelitian dengan menerapkan suatu metode untuk

memudahkan pengelompokan atau *clustering* daerah tersebut, yaitu mengunakan *Fuzzy C-Means*.

Fuzzy C-Means menggunakan model pengelompokan *fuzzy* sehingga data dapat menjadi anggota dari semua kelas atau kelompok terbentuk dengan derajat atau tingkat keanggotaan yang berbeda antara 0 hingga 1. Tingkat keberadaan data dalam suatu kelas atau *cluster* ditentukan oleh derajat keanggotaannya [16]. Hasil *cluster* yang diperoleh dengan menggunakan *Fuzzy C-Means* akan dilanjutkan untuk menentukan jalur-jalur yang dapat dibuat model graf dengan menggunakan sebuah metode, salah satunya ialah metode *Minimum Spanning Tree* berbasis algortima hybrid genetika.

Minimum Spanning Tree (MST) adalah suatu keadaan dimana semua node dalam graf terhubung, namun tidak boleh terdapat *loop* didalamnya dan dihitung bobot tree yang terkecil. Masalah MST bertujuan untuk menghubungkan seluruh simpul dalam pola jaringan sehingga total panjang cabang tersebut dapat diminimumkan. Pola jaringan yang dihasilkan menghubungkan semua titik dalam jaringan tersebut dengan total jarak minimum.

Penyelesaian permasalahan *minimum spanning tree* sederhana mungkin dapat diselesaikan dengan melakukan perhitungan manual. Namun untuk kasus *spanning tree* yang besar dan kompleks, perhitungan manual akan sulit dilakukan karena akan memakan waktu yang lama. Oleh sebab itu dibutuhkan satu program aplikasi komputer yang dapat melakukan perhitungan nilai *minimum spanning tree* dengan cepat dan akurat. Salah satu algoritma yang dapat digunakan yaitu algoritma genetika.

Sejak diperkenalkan oleh Holland (1992), algoritma genetika saat ini telah dikenal luas sebagai salah satu metode *heuristic* yang banyak digunakan untuk mendapatkan solusi berbagai persoalan dunia nyata yang sulit diperoleh solusi eksaknya. Bererapa ahli yang mempopulerkan algoritma genetika diantaranya Michalewicz (1994) dan Gen & Cheng (1997, 2000). Beberapa tahun terakhir, banyak dilakukan pengembangan algoritma genetika untuk menyelesaikan berbagai persoalan logistik diantaranya: *travelling salesman*, transportasi, *supply chain*, dan sebagainnya.

Beberapa hasil eksperimen berbasis algoritma genetika tersebut, diperoleh informasi bahwa algoritma genetika mampu memberikan solusi pendekatan yang optimal untuk persoalan-persoalan tersebut dalam waktu yang relatif singkat. Meskipun tidak dapat dijamin bahwa algoritma genetika akan selalu memberikan solus optimal dari persoalan-persoalan optimisasi, setelah melalui proses evolusi pada beberapa generasi, algoritma genetika pada umumnya akan mampu memberikan sulusi yang baik. Oleh karena itu, algoritma genetika saat ini banyak dipakai pada berbagai aplikasi bisnis, teknik maupun pada bidang-bidang keilmuan lain.

Dalam perkembangannya algoritma genetik dapat dikombinasikan (*hybrid*) dengan berbagai jenis metode lain. Pada prinsipnya hibridisasi diharapkan mampu memberikan solusi lain yang lebih baik. Algoritma hybrid genetika merupakan kombinasi metode-metode heuristik lain ke dalam algoritma genetika dengan harapan mampu meningkatkan kinerja algoritma genetika. Algoritma genetika dapat dikombinasikan dengan berbagai jenis metode, diantaranya dengan metode *local search*. Algoritma *local search* digunakan untuk meningkatkan efesiensi dari algoritma genetika dengan mengevaluasi setiap solusi atau individu yang dihasilkan agar tidak terjebak pada solusi-solusi yang buruk.

Berdasarkan latar belakang tersebut, dalam penelitian ini akan dilakukan pengklasteran dengan menggunakan *Fuzzy C-means,* kemudian melakukan penyelesaian *minimum spanning tree* berbasis algoritma *hybrid* genetika. Penyelesaian *minimum spanning tree* berbasis algoritma *hybrid* genetika ini akan diimplementasikan untuk menentukan pola jaringan pergerakan logistik di kepulauan Maluku.

1.2 Rumusan Masalah

Berkaitan dengan latar belakang yang telah diuraikan di atas, disusun suatu rumusan masalah yang dibahas dalam penelitian ini antara lain:

1. Bagaimana mendapatkan variasi kelompok pergerakan logistik pada transportasi laut di wilayah kepulauan dengan menggunakan *Fuzzy C-Means*?

2. Bagaimana menentukan pola jaringan pergerakan logistik pada transportasi laut di wilayah kepulauan dengan menggunakan *Minimum Spanning Tree* berbasis *Hybrid Genetic Algorithm*?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam penelitian ini dibatasi sebagai berikut:

1. Pergerakan logistik pada transportasi laut di wilayah Kepulauan Maluku.
2. MST yang di modelkan merupakan graf lengkap tak berarah
3. Tidak memperhatikan isi atau muatan kapal dan arah pergerakan kapal.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian Tesis ini antara lain:

1. Mendapatkan pola jaringan pergerakan logistik pada transportasi laut wilayah kepulauan dengan menggunakan *Clustering Fuzzy C-Means* dan *Hybrid Genetic Algorithm*.
2. Merancang jaringan pergerakan logistik pada transportasi laut wilayah kepulauan dengan menggunakan *Clustering Fuzzy C-Means* dan *Hybrid Genetic Algorithm*.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini berdasarkan tujuan yang telah dipaparkan adalah sebagai berikut.

1. Mendapatkan pola jaringan pergerakan logistik pada transportasi laut optimal di daerah kepulauan.
2. Sebagai salah satu kontribusi untuk pengembangan ilmu pengetahuan Matematika di bidang Jaringan Transportasi.
3. Menambah pemahaman dan pengalaman dalam menggunakan metode *Clustering* dengan *Fuzzy C-Means* dalam memetakan atau mengelompokan suatu data.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini diuraikan kajian pustaka dan dasar teori yang digunakan dalam penelitian Tesis ini. Pertama, dibahas mengenai penelitian sebelumnya yang menjadi referensi dari penelitian. Selanjutnya, dijelaskan mengenai sistem logistik, graf dan *minimum spanning tree*, *fuzzy C-means*, serta algoritma genetika.

2.1 Penelitian Terdahulu

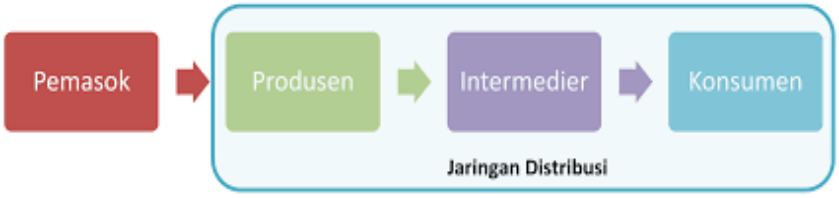
Berikut ini penelitian-penelitian yang berkaitan dengan proposal tesis ini, yaitu:

1. Jung-Bok Jo, Yinzhen Li, dan Mitsuo Gen (2007) dalam "*Nonlinear Fixed Charge Transportatation Problem by Spanning Tree-based Genetic Algorithm*" menjelaskan bahwa solusi yang optimal untuk masalah biaya transportasi nonlinear telah didapat dengan menggunakan pendekatan Spanning Tree-based Genetic Algorithm [9].
2. Zaverdehi, Kesthehi, dan Moghaddam (2011) dalam "*Solving Capacitated Fixed-charge Transportation Problem by Artificial Immune and Genetic Algorthm with a Prufer Number Representation*" menjelaskan bahwa solusi optimal untuk masalah biaya transportasi dapat diselesaikan dengan menggunakan pendekaan *Spanning Tree* [20].
3. Hesam Izakian dan Ajith Abraham (2011) dalam "*Fuzzy C-means and Fuzzy Swarm for Fuzzy Clustering Problem*" membahas bahwa algoritma FCM merupakan salah satu teknik *fuzzy clustering* yang efisien dan mudah diterapkan. Hasil eksperimen menunjukkan bahwa metode ini efektif dan mendapatkan hasil yang baik [8].
4. Shahab, Utomo, dan Irawan (2016) dalam "*Decomposing and Solving Capacitated Vehicle Routing Problem* (CVRP) *using Two-Step Genetic Algorithm* (TSGA)" membahas penggunaan dua algoritma genetika untuk menyelesaikan CVRP dengan cara yang berbeda. GA digunakan untuk

menyelesaikan CVRP secara langsung, sedangkan TSGA pertama akan membagi wilayah CVRP yang diselesaikan dengan TGSA1 dan kemudian mendapatkan rute terpendek untuk masing-masing daerah menggunakan TGSA2 [22].

2.2 Sistem Logistik

Logistik adalah bagian dari rantai pasok (*supply chain*) yang menangani arus barang, arus informasi dan arus uang melalui proses pengadaan (*procurement*), penyimpanan (*warehousing*), transportasi (*transportation*), distribusi (*distribution*), dan pelayanan pengantaran (*delivery services*) sesuai dengan jenis, kualitas, jumlah, waktu dan tempat yang dikehendaki konsumen, secara aman, efektif dan efisien, mulai dari titik asal (*point of origin*) sampai dengan titik tujuan (*point of destination*). Pada dasarnya obyek logistik tidak terbatas pada logistik barang, namun mencakup pula logistik penumpang, logistik bencana, dan logistik militer (pertahanan keamanan), sedangkan aktivitas pokok logistik meliputi pengadaan, produksi, pergudangan, distribusi, transportasi, dan pengantaran barang yang dilakukan oleh setiap pelaku bisnis dan industri baik pada sektor primer, sekunder maupun tersier dalam rangka menunjang kegiatan operasionalnya [11].



Gambar 2.1. Sistem logistik

Masalah transportasi sebagai perencanaan logistik merupakan tujuan utama dari logistik. Permasalahan transportasi mengenai bagaimana memindahkan dan menyimpan barang dari sumber (*source*) untuk sampai ke tujuan (*destination*) dengan tujuan meminimalkan transportasi dan biaya pengiriman. Penyelesaian permasalahan ini menggunakan model transportasi dua tahap.

Ciri khusus pada model permasalahan transportasi dua tahap adalah terdapat sumber dan sejumlah tujuan, kuantitas komoditas produk yang dipindahkan dari sumber sesuai dengan permintaan tujuan dan produk yang diminta harus sesuai dengan jumlah barang yang diproduksi oleh sumber. Pada permasalahan transportasi dua tahap digambarkan pada suatu model transportasi dengan sumber (I), distributor (J) dan konsumen (K) untuk memperjelas proses yang ada [5].

Untuk penyelesaian masalah transportasi variabel-variabel yang digunakan adalah :

I : Jumlah sumber $(i = 1,2, \dots, I)$

x_{ij} : Jumlah barang yang dikirimkan dari sumber (pabrik) menuju distributor.

a_i : Unit persediaan (kapasitas produksi) sumber untuk dipindahkan ke distributor.

J : Jumlah distributor $(j = 1,2, \dots, J)$

t_{ij} : Biaya perjalanan dari sumber (pabrik) menuju distributor.

y_{jk} : Jumlah barang yang dikirimkan dari distributor menuju konsumen.

b_j : Jumlah kapasitas ditributor.

c_{jk} : Biaya pengiriman dari distributor menuju konsumen.

K : Jumlah konsumen $(k = 1,2, \dots, K)$

d_k : Jumlah permintaan dari konsumen K.

Untuk meminimilakan biaya transportasi diberikan fungsi obyektif hubungan antara biaya dan variabelnya. Fungsi obyektif untuk permasalahan transportasi pada distribusi dua tahap adalah dengan miminimalkan Z yang dinyatakan secara matematis pada Persamaan (2.1).

$$Z = \sum_{i=1}^{I}\sum_{j=1}^{J} t_{ij}x_{ij} + \sum_{j=1}^{J}\sum_{k=1}^{K} c_{jk}y_{jk} \qquad (2.1)$$

Kendala (2.2) dan (2.3) untuk memastikan bahwa kapasitas dari pendistribusian cukup.

$$\sum_{j=1}^{J} x_{ij} \leq a_i, \quad \forall\, i \tag{2.2}$$

$$\sum_{k=1}^{K} y_{jk} \leq b_j z_j, \quad \forall\, j \tag{2.3}$$

Jumlah distributor yang dibuka tidak melebihi batas atasnya dan dinyatakan dalam kendala (2.4).

$$\sum_{j=1}^{J} z_j \leq W \tag{2.4}$$

Semua permintaan konsumen harus terpenuhi, dinyatakan dalam kendala (2.5).

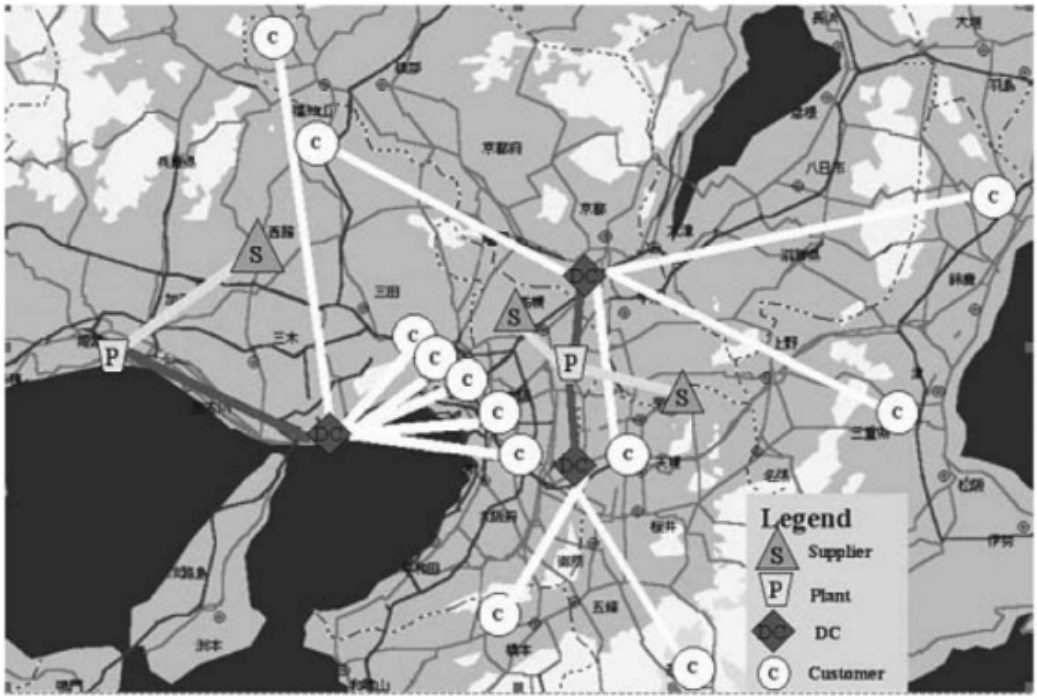
$$\sum_{j=1}^{J} y_{jk} \geq d_k \tag{2.5}$$

Total penawaran sama dengan total permintaan yang biasa disebut transportasi berimbang. Tidak boleh terdapat perhitungan negatif, dinyatakan dalam kendala (2.6) dan (2.7)

$$\sum_{i=1}^{I} x_{ij} \sum_{j=1}^{J} x_{ij} = \sum_{j=1}^{J} \sum_{k=1}^{K} y_{jk} \tag{2.6}$$

$$x_{ij}, y_{jk} \geq 0, \quad \forall\, i, j, k \tag{2.7}$$

Permasalahan jaringan transportasi sistem logistik dalam rantai pasok dapat diilustrasikan pada Gambar 2.2 [5]

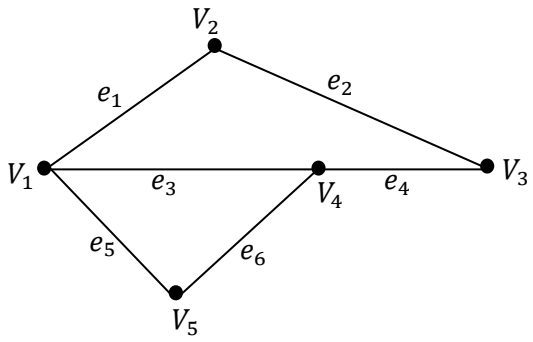


Gambar 2.2 Ilustrasi Jaringan Transportasi

2.3 Graf

Graf merupakan representasi dari suatu masalah yang digambarkan sebagai sekumpulan noktah atau simpul (*vertex*) yang dihubungkan dengan sekumpulan garis atau sisi (*edge*). Secara matematis, graf *G* didefinisikan sebagai pasangan himpunan *(V,E)* ditulis dengan notasi *G = (V,E)* yang dalam hal ini *V* adalah himpunan berhingga dan tidak kosong dari simpul-simpul (*vertex*) dengan notasi V = {v_1, v_2,…v_n}, sedangkan *E* adalah himpunan sisi yang menghubungkan sepasang simpul dengan notasi *E* = {e_1, e_2,…e_m} [21].

Simpul pada graph dapat dinomori dengan huruf, seperti *v,w,…*, dengan bilangan asli *1,2,3,…*, atau gabungan keduanya. Sisi (*edge*) yang menghubungkan dua simpul (*vertex*) v_i dan v_j dinyatakan dengan pasangan (v_i, v_j) atau dengan lambang e_1, e_2, dengan kata lain jika e adalah sisi yang menghubungkan simpul v_i dan v_j, maka *e* dapat ditulis sebagai *e* = (v_i, v_j).

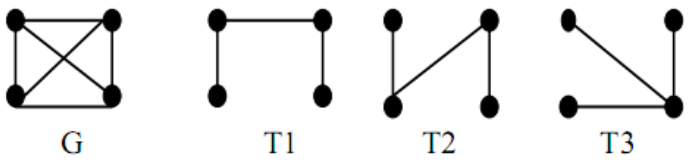


Gambar 2.3 Graf G

Diberikan sebuah graf *G* pada Gambar 2.2 yang terdiri dari empat simpul *V* = {v_1, v_2, v_3, v_4} dan lima sisi *E* = {e_1, e_2, e_3, e_4, e_5}dengan sisi-sisinya dinyatakan oleh e_1 = {v_1; v_2}, e_2 = {v_2; v_3}, e_3 = {v_1; v_4}, e_4 = {v_4; v_3}, e_5 = {v_1;v_5}, e_6 = {v_5; v_4}.

2.4 *Minimum Spanning Tree*

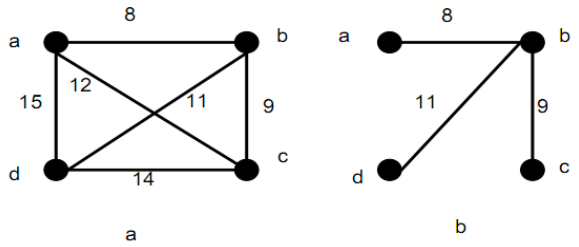
Spanning tree merupakan suatu subgraf dari suatu graf dimana setiap simpul pada *spanning tree* sama dengan semua simpul pada graf. Gambar 2.4 adalah graf lengkap dan tiga buah pohon rentangnya.



Gambar 2.4. Graph lengkap G dengan pohon rentangnya T1, T2, T3

Spanning tree menghubungkan semua simpul pada graf. Apabila graf tersebut adalah graf berbobot (*Weighted Graph*), kemudian dari *spanning tree* yang dimiliki oleh graf didefinisikan sebagai penjumlahan dari bobot-bobot seluruh cabang pada pohon rentang maka akan diperoleh pohon rentang yang

memiliki bobot. *Spanning tree* yang memiliki bobot terkecil pada suatu graph berbobot disebut pohon rentang minimum (*minimum spanning tree*) [13, 18].



Gambar 2.5. (a) Graph berbobot , (b) pohon rentang minimumnya

Secara umum penyelesaian masalah *minimum spanning tree* (MST) dapat dilakukan dengan menggunakan metode konvensial dan metode heuristik. Metode konvensional menggunakan perhitungan matematis biasa untuk menyelesaikan masalah MST sepeti algortima Prim dan Kruskal. Sementara metode *heuristic* menggunakan kecerdasan buatan untuk menentukan rute terpendek pada MST, salah satu metode yang dapat digunakan adalah algoritma genetika karena algoritma genetika dapat mencari nilai fitness yang terbaik dari kombinasi gen-gen. Oleh karena itu MST dapat diselesaikan dengan algoritma genetika.

2.5 *Clustering*

Clustering adalah suatu metode pengelompokkan berdasarkan ukuran kedekatan (kemiripan). *Cluster* merupakan pola yang terbentuk dalam suatu proses pembagian sekelompok data ke dalam sejumlah sub-kelas. *Clustering* dapat digunakan untuk memberikan label suatu data yang belum diketahui kelasnya. Prinsip dari *clustering* adalah memaksimalkan kesamaan anggota dalam setiap kelompok (*cluster*) dan meminimalkan jarak antara pusat *cluster* dengan *cluster* lain. Perhitungan jarak tersebut digunakan untuk mengukur kemiripan data [7].

2.5.1 *Fuzzy Clustering*

Fuzzy clustering adalah salah satu teknik untuk menentukan *cluster* optimal dalam suatu ruang vector yang didasarkan pada bentuk normal *Euclidian* untuk jarak antar vektor. *Fuzzy clustering* sangat berguna bagi pemodelan *fuzzy* terutama dalam mengidentifikasi aturan-aturan *fuzzy* [10, 12].

2.5.2 *Fuzzy C-Means*

Fuzzy C-Means adalah suatu teknik pengklusteran dimana keberadaan tiap-tiap titik data dalam *cluster* ditentukan oleh derajat keanggotaan. *Fuzzy C-Means* (FCM) merupakan salah satu algoritma *fuzzy clustering*. FCM merupakan teknik pengklusteran dimana tiap-tiap data ditentukan oleh derajat keanggotaannya. Tujuan dari algoritma FCM adalah untuk menemukan pusat *cluster* (*centroid*) dengan meminimumkan fungsi objektif [1].

Konsep dasar FCM, pertama kali adalah menentukan pusat *cluster*, yang akan menandai lokasi rata-rata untuk tiap-tiap *cluster*. Pada kondisi awal, pusat *cluster* ini masih belum akurat. Tiap-tiap titik data memiliki derajat keanggotaan untuk tiap-tiap *cluster*. Dengan cara memperbaiki pusat *cluster* dan derajat keanggotaan tiap-tiap titik data secara berulang, maka akan dapat dilihat bahwa pusat *cluster* akan bergerak menuju lokasi yang tepat. Perulangan ini didasarkan pada minimisasi fungsi objektif yang menggambarkan jarak dari titik data yang diberikan ke pusat *cluster* yang terbobot oleh derajat keanggotaan titik data tersebut [10].

Asumsikan ada sejumlah data dalam set data (X) yang berisi m data: $x_1, x_2, \dots, x_m$, dinotasikan $X = \{x_1, x_2, \dots, x_m\}$, dimana setiap data mempunyai n dimensi: $x_{i1}, x_{i2}, \dots, x_{im}$, dinotasikan $x_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$. Selanjutnya sejumlah kelompok C dengan sentroid: $c_1, c_2, \dots, c_k$, dimana k adalah jumlah kelompok. Setiap data mempunyai derajat keanggotaan pada setiap kelompok yang dinyatakan dengan u_{ij}, dengan nilai di antara 0 dan 1. i menyatakan data x_i, dan j menyatakan kelompok c_j[16].

Algoritma *clustering* dengan *Fuzzy C-Means* [1,14]

1. Masukan data yang akan di*cluster* kedalam sebuah matriks X, dimana matriks berukuran m x n, dengan m adalah jumlah data yang akan di*cluster* dan n adalah atribut setiap data. X_{ij} = data ke- i (i= 1,2,…m), atribut ke-j (j= 1,2,…,n).
2. Menentukan parameter.
   - Jumlah *cluster* = c
   - Bobot pangkat = w
   - Maksimum iterasi = MaxIter
   - Error terkecil yang diharapkan = ζ
   - Fungsi objektif awal = $P_0 = 0$
   - Iterasi awal = $t = 1$
2. Bangkitkan bilangan acak μ_{ik} (dengan i= 1,2,…m dan k= 1,2,…c) sebagai elemen matriks partisi awal U.

   μ_{ik} adalah derajat keanggotaan yang merujuk pada seberapa besar kemungkinan suatu data bisa menjadi anggota ke dalam suatu *cluster*. Posisi dan nilai matriks dibangun secara random. Dimana nilai keangotaan terletak pada interval 0 sampai dengan 1. Pada posisi awal matriks partisi U masih belum akurat begitu juga pusat *cluster*nya. Sehingga kecendrungan data untuk masuk suatu *cluster* juga belum akurat.

   Hitung jumlah setiap kolom (atribut):

$$Q_j = \sum_{k=1}^{c} \mu_{ik} \tag{2.8}$$

   dengan $j = 1,2,\dots,n$

   Q_j adalah jumlah nilai derajat keanggotaan perkolom = 1
3. Hitung pusat cluster ke-k: V_{kj}, dengan k=1,2,…c dan j= 1,2,…,n

$$V_{kj} = \frac{\sum_{i=1}^{n}\left((\mu_{ik})^w * X_{ij}\right)}{\sum_{k=1}^{n}(\mu_{ik})^w} \tag{2.9}$$

4. Hitung fungsi objektif pada iterasi ke- t, P_t :

   Fungsi obyektif digunakan sebagai syarat perulangan untuk mendapatkan pusat *cluster* yang tepat. Sehingga diperoleh kecendrungan data untuk masuk ke *cluster* mana pada step akhir.

$$P_t = \sum_{i=1}^{n}\sum_{k=1}^{c}\left(\sum_{j=1}^{m}\left[\left(X_{ij} - V_{kj}\right)^2\right](\mu_{ik})^w\right) \qquad (2.10)$$

5. Hitung perubahan derajat keanggotaan setiap data pada setiap cluster (memperbaiki matriks partisi U) dengan

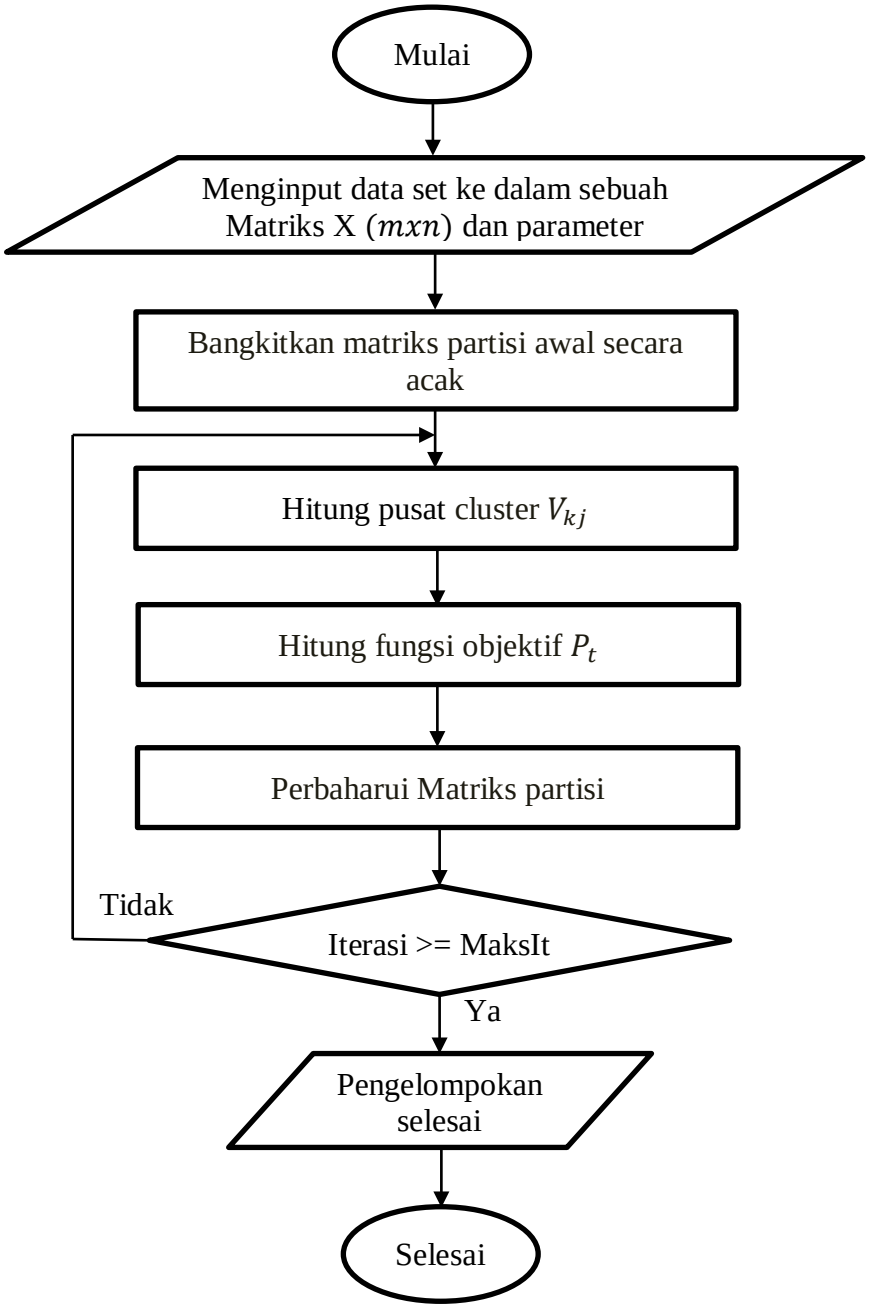
$$\mu_{ik} = \frac{\left[\sum_{j=1}^{m}\left(X_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}}{\sum_{k=1}^{c}\left[\sum_{j=1}^{m}\left(X_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}} \qquad (2.11)$$

   dengan : i = 1,2,…,n; dan k = 1,2,…,c.

6. Cek kondisi berhenti:

   a) Jika : (|Pt-Pt-1|< ξ) atau (t> MaxIter) maka berhenti;

   b) Jika tidak : t = t + 1, ulangi langkah ke 4



Gambar 2.6 *Flowchart Fuzzy C-Means Clustering*

2.5.3 Indeks Validitas

Indeks validitas adalah suatu ukuran yang digunakan untuk menentukan jumlah kelompok yang optimal. Beberapa indeks validitas yang dapat digunakan dalam diantaranya :

- ***Partition Coefficient (PC)***

Indeks ini mengukur jumlah *overlapping* antar kelompok. Indeks ini dirumuskan oleh Bezdek sebagai berikut [19]:

$$PC(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{k=1}^{N} {u_{ik}}^2 \qquad (2.12)$$

dimana *N* adalah banyak objek penelitian, *c* adalah banyak kelompok,u_{ik} adalah nilai keanggotaan objek ke-*k* dengan pusat kelompok ke-*i*. Indeks ini memiliki rentang $1/c$ sampai 1. Jumlah kelompok yang optimal ditunjukkan oleh nilai PC yang paling besar.

- ***Classification Entropy (CE)***

CE hanya mengukur kekaburan (*fuzziness*) dari partisi kelompok. Indeks ini dirumuskan sebagai berikut [19] :

$$CE(c) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}\ln(u_{ik}) \qquad (2.13)$$

dimana N adalah banyak objek penelitian, *c* adalah banyak kelompok, dan u_{ik} adalah nilai keanggotaan objek ke-k dengan pusat kelompok ke-*i*. Indeks ini memiliki rentang 0 sampai ln(c). Indeks CE yang semakin kecil menunjukkan pengelompokan yang lebih baik.

- ***Xie and Beni's index (XB)***

XB bertujuan untuk menghitung rasio total variasi di dalam kelompok dan pemisahan kelompok. Indeks ini dapat dirumuskan sebagai berikut [19] :

$$XB(c) = \frac{\sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^m \|x_k - v_i\|^2}{N min_{i,k}\|v_k - v_i\|^2} \qquad (2.14)$$

diman *N* adalah banyaknya objek penelitian, *c* banyaknya kelompok, u_{ik} adalah keanggotaan objek ke-*k* dengan pusat kelompok ke-*i*. *m* adalah fuzzifier, $\|x_k -$

$v_i\|$ merupakan jarak *euclidean* titik data (x_k) dengan pusat kelompok v_i, dan $\|v_k - v_i\|$ adalah jarak *Euclidean* antara pusat kelompok. Nilai *XB* yang terendah mengindikasikan partisi kelompok yang lebih baik.

2.6 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*" [17]. Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan pupulasi berikutnya merupakan evolusi kromosom-kromosom melalui iterasi yang disebut dengan generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan nilai *fitnes*. Algoritma genetika dimulai dengan mendefinisikan variabel optimasi, fungsi *cost*, dan *cost*, dan diakhiri dengan uji konvergensi.

2.6.1 Memilih variabel dan fungsi *cost*

Fungsi *cost* menghitung nilai output berdasarkan nilai dari sekumpulan variabel *input* (kromosom). Algoritma genetika dimulai dengan mendefinisikan sebuah kromosom atau *array* atau nilai variabel untuk dioptimalkan. 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 $p_1, p_2, \dots, p_{N_{var}}$:

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

2.6.2 Inisialisasli 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 $N_{pop} \times N_{var}$.

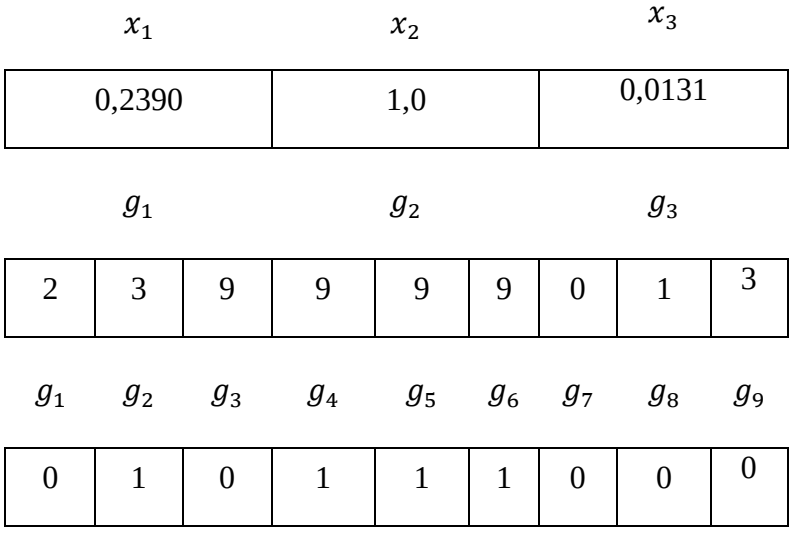
Populasi awal adalah sekumpulan kromosom yang telah dibangkitkan secara random, kromosom tersebut dijadikan sebagai kandidat solusi. Hal terpenting yang harus diperhatikan dalam inisialisasi populasi adalah jumlah populasi (*size population*) sebab jika jumlah populasi terlalu sedikit maka populasi akan terlalu cepat mencapai konvergen (konvergensi prematur), sehingga populasi terjebak pada *local optima*. Sedangkan jika jumlah populasi terlalu banyak maka perhitungan akan menjadi terlalu kompleks dan akan memakan waktu komputasi yang lebih lama dan mengakibatkan hasil perhitungan tidak cepat diperoleh. Sehingga perlu dipikirkan keseimbangan untuk kedua hal ini.

Tahap pengkodean kromosom (*decode chromosome*) ini bertujuan untuk membangkitkan kromosom (individu) secara acak sebagai kandidat awal solusi suatu masalah dengan cara mengkodekan kromosom kedalam bentuk gen. Kromosom tersebut merupakan representasi dari variabel keputusan, setiap kolom merupakan kromosom dan elemen dalam kolom merupakan gen. Kromosom yang dibangkitkan mempunyai batasan (interval), yaitu batas bawah dan atas dari variabel keputusan hal ini berfungsi agar kromosom yang dibangkitkan tidak keluar dari range solusi optimal.

Dalam algoritma genetika pengkodean (*decoding*) solusi kedalam suatu kromosom merupakan hal terpenting. Terdapat tiga skema yang paling umum digunakan dalam pengkodean algoritma genetika, yaitu:

- *Real-number encoding*. Pada skema ini, nilai gen berada dalam interval [0,R], dimana R adalah bilangan real positif dan biasanya R=1.
- *Discrete decimal encoding*. Setiap gen bisa bernilai salah satu bilangan bulat dalam interval [0,9].
- *Binary Encoding*. Setiap gen hanya bisa bernilai 0 atau 1.

Misalkan terdapat tiga variabel, yaitu x_1, x_2, x_3 yang dikodekan ke dalam sebuah kromosom, maka kromosom tersebut mempunyai tiga gen. Skema pengkodean untuk ketiga variabel tersebut dapat dilihat pada Gambar 2.7 berikut ini.



Gambar 2.7 Tiga jenis skema pengkodean

Pada Gambar 2.7 menunjukkan masing-masing *real-number encoding* (atas), *discrete decimal encoding* (tengah), dan *binary encoding* (bawah). Berdasarkan Gambar 2.7 tersebut terdapat tiga variabel, yaitu x_1, x_2, x_3, yang dikodekan ke dalam sebuah kromosom yang terdiri dari tiga gen (untuk *real-number encoding*). Sedangkan pada *discrete decimal encoding* maupun *binary encoding* ketiga variabel dikodekan ke dalam kromosom yang terdiri dari sembilan gen (masing-masing variabel dikodekan ke dalam tiga gen). Dalam Tesis ini, pengkodean kromosom menggunakan *Real-number encoding*.

2.6.3 *Crossover*

Banyak perbedaan pendekatan yang telah dicoba untuk melakukan *crossover* pada algoritma kontinu. Salah satunya yaitu dengan menyeleksi secara acak sebuah variabel dengan 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 dikombinasikan ke bentuk variabel baru yang akan muncul pada children.

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

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

Tahap terakhir adalah untuk melengkapi crossover dengan sisa dari kromosom sebelumnya :

$$of\ f\ spring_1 = [p_{m1}p_{m2} \dots p_{new1} \dots p_{dN_{var}}]$$

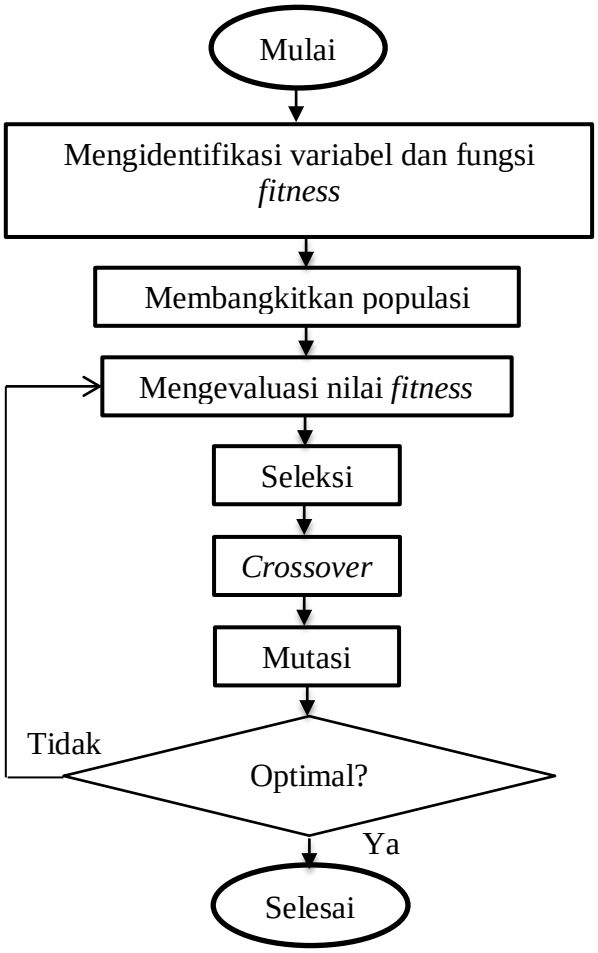
$$of\ f\ spring_2 = [p_{d1}p_{d2} \dots p_{new2} \dots p_{mN_{var}}]$$

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 [18].

2.6.4 Mutasi

Langkah selanjutnya setelah melakukan *crossover* yaitu 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 genetika baru dalam populasi secara acak dan memodifikasi beberapa blok. Mutasi dapat mempertahankan keragaman 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.



Gambar 2.8 *Flowchart* dari algoritma genetika [15]

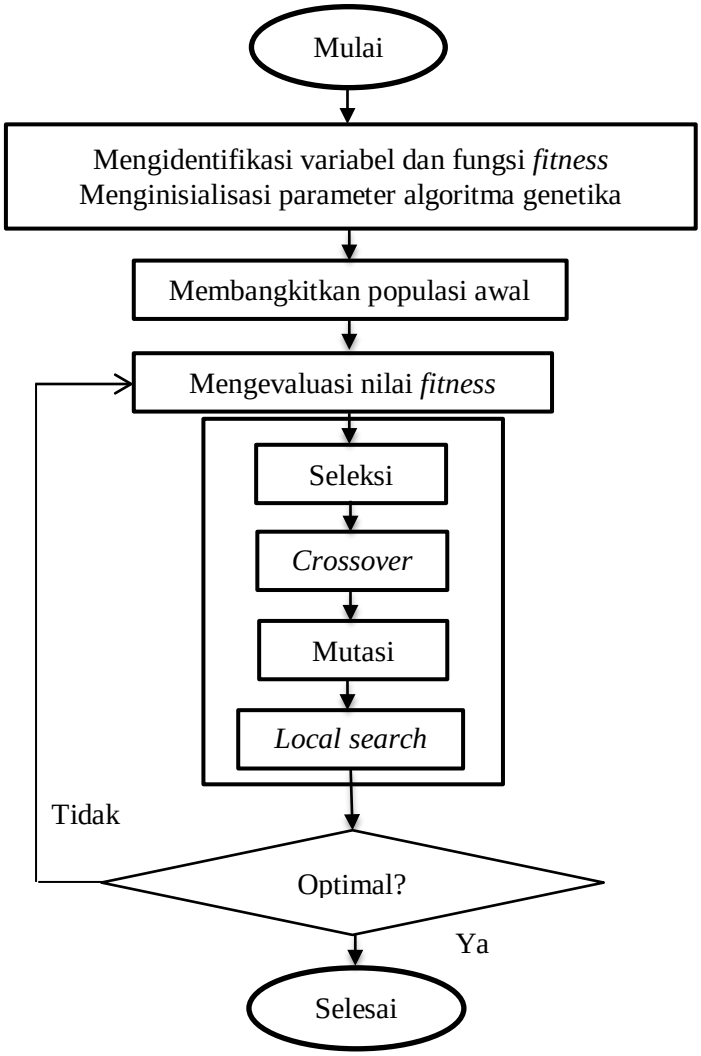
2.6.5 Algoritma Hybrid Genetika

Algoritma *hybrid* genetika merupakan kombinasi metode-metode heuristik lain ke dalam algoritma genetika dengan harapan mampu meningkatkan kinerja algoritma genetika [3]. Dalam perkembangannya algoritma genetik dapat dikombinasikan (*hybrid*) dengan berbagai jenis metode lain, diantaranya dengan metode *local search*. Pada prinsipnya hibridisasi ini diharapkan mampu memberikan solusi lain yang lebih baik disekitar lokal optimum yang diberikan oleh algoritma genetika atau dikenal dengan istilah *local search*. Algoritma *hybrid* genetika menggunakan fungsi random sehingga menyebabkan algoritma *hybrid*

genetika menjadi suatu algoritma berbasis komputer untuk menghasilkan solusi yang lebih optimal dengan waktu komputasi lebih singkat. [4]

2.6.6 *Local Search*

Local search pada *hybrid* algoritma genetika bertujuan untuk melakukan perbaikan lokal yang dapat diterapkan sebelum dan atau sesudah proses seleksi, crossover dan mutasi. Algoritma *local search* digunakan untuk meningkatkan efesiensi dari algoritma genetika dengan mengevaluasi setiap solusi atau individu yang dihasilkan agar tidak terjebak pada solusi-solusi yang buruk.



Gambar 2.9 *Flowchart* dari algoritma *hybrid* genetika

BAB 3
METODE PENELITIAN

Dalam bab ini diuraikan langkah-langkah sistematis yang akan dilakukan dalam proses pengerjaan penelitian Tesis. Terdapat enam tahap yang akan dilakukan dalam pengerjaan penelitian ini, antara lain studi literatur, pengumpulan data penelitian, analisa data, simulasi, analisa hasil dan pembahasan, dan terakhir adalah penyusunan hasil penelitian.

1. Studi Literatur

   Pada bagian ini peneliti akan melakukan studi literatur terhadap hal-hal yang berkaitan dalam penelitian, diantaranya mengenai sistem logistik, *clustering*, graf, *minimum spanning tree*, algoritma genetika. Pembelajaran lebih mendalam mengenai hal-hal tersebut dapat diperoleh baik melalui buku-buku literatur, jurnal, paper, maupun artikel dari internet. Selain itu dilakukan pengumpulan data mengenai kodisi di kepulauan Maluku yaitu titik-titik koordinatnya dan data diperoleh dari *Google Map*.

2. Proses *clustering* wilayah

   Proses *clustering* wilayah akan dilakukan berdasarkan data skunder. Data skunder yang didapatkan akan di-*cluster* menggunakan metode *Fuzzy C-means*. Proses integrasi data yang dilakukan adalah titik koordinat *dummy* (X dan Y). Integrasi titik koordinat pada data diperlukan untuk visualisasi persebaran *cluster* wilayah pada peta. Titik koordinat yang diberikan bersifat random pada satu daerah.

Tujuan dari proses cluster pada penelitian ini adalah untuk mendapatkan berbagai kelompok data titik koordinat yang dikelompokan berdasarkan karakteristik titik koordinatnya. Penggunaan data akan dilakukan proses cluster menggunakan *Fuzzy C-means* yang menghasilkan kelompok-kelompok data titik koordinat sebanyak jumlah cluster yang diberikan.

Uji coba running *Fuzzy C-means* dilakukan sebanyak 5 kali dalam pemilihan hasil *cluster* yang memiliki total jarak kuadrat di antara setiap titik data dengan representasi *cluster* terdekat yang nilainya paling kecil. Kelompok-

kelompok data wilayah ini masing-masing akan membentuk suatu jaringan dengan menggunakan *minimum spanning tree* berbasis algoritma genetika.

3. Proses *Minimum Spanning Tree* berbasis *Hybrid Genetic Algorithm*

Pada tahap ini dilakukan perumusan *Minimum Spanning Tree* berbasis Algoritma Genetika. Dalam perumusan ini dilakukan penentuan *nodes* dan bobot tiap *vertex*. Bobot tersebut diperoleh dari jarak antar koordinat asal dengan koordinat tujuan.

Penyelesaian *minimum spanning tree* berbasi Algoritma Genetika terdapat beberapa tahapan. Secara umum tahapan yang dilakukan dalam Algortima Genetika, yaitu :

a. Tahap inisialisasi parameter, dalam sistem ini terdapat 4 utama parameter yaitu besar populasi, generasi maksimum, *crossover rate*, dan *mutation rate*.
b. Tahap pembangkit kromosom, merupakan tahap untuk membentuk suatu kromosom dari sekumpulan gen, dengan gen merupakan node yang mempresentasikan titik-titik koordinat.
c. Tahap crossover/pindah silang, merupakan tahapan untuk memindah silangkan gen-gen dari suatu kromosom ke kromosom lain, sehingga membentuk kromosom yang baru.
d. Tahap mutation/mutasi, merupakan tahap untuk melakukan mutasi gen di tiap kromosom.
e. Tahap evaluasi, merupakan tahap untuk membangkitkan nilai fitness pada bagian kromosom.
f. Tahap seleksi, merupakan tahap untuk menyeleksi kromosom dengan nilai fitness terbesar.
g. Tahap pembentukan populasi baru, merupakan pembentukan populasi dari hasil tahap genetic. Setelah tahap ini akan dilakukan proses pembentukan generasi selama generasi kurang dari generasi maksimum.

4. Simulasi

Pada tahap ini akan dilakukan simulasi menggunakan *software* MATLAB terhadap penerapan *Fuzzy C-means clustering* dan algoritma genetika dalam permasalahan *minimum spanning tree*. Dalam simulasi ini terdapat proses

pengolahan data yang ada sehingga menghasilkan suatu nilai. Dari nilai tersebut akan digunakan dalam algoritma genetika untuk menentukan pergerakan logistik. Dari hasil simulasi tersebut, dapat dilihat kinerja *minimum spanning tree* berbasis *hybrid genetic algorithm* yang mampu mendapakan hasil yang optimal.

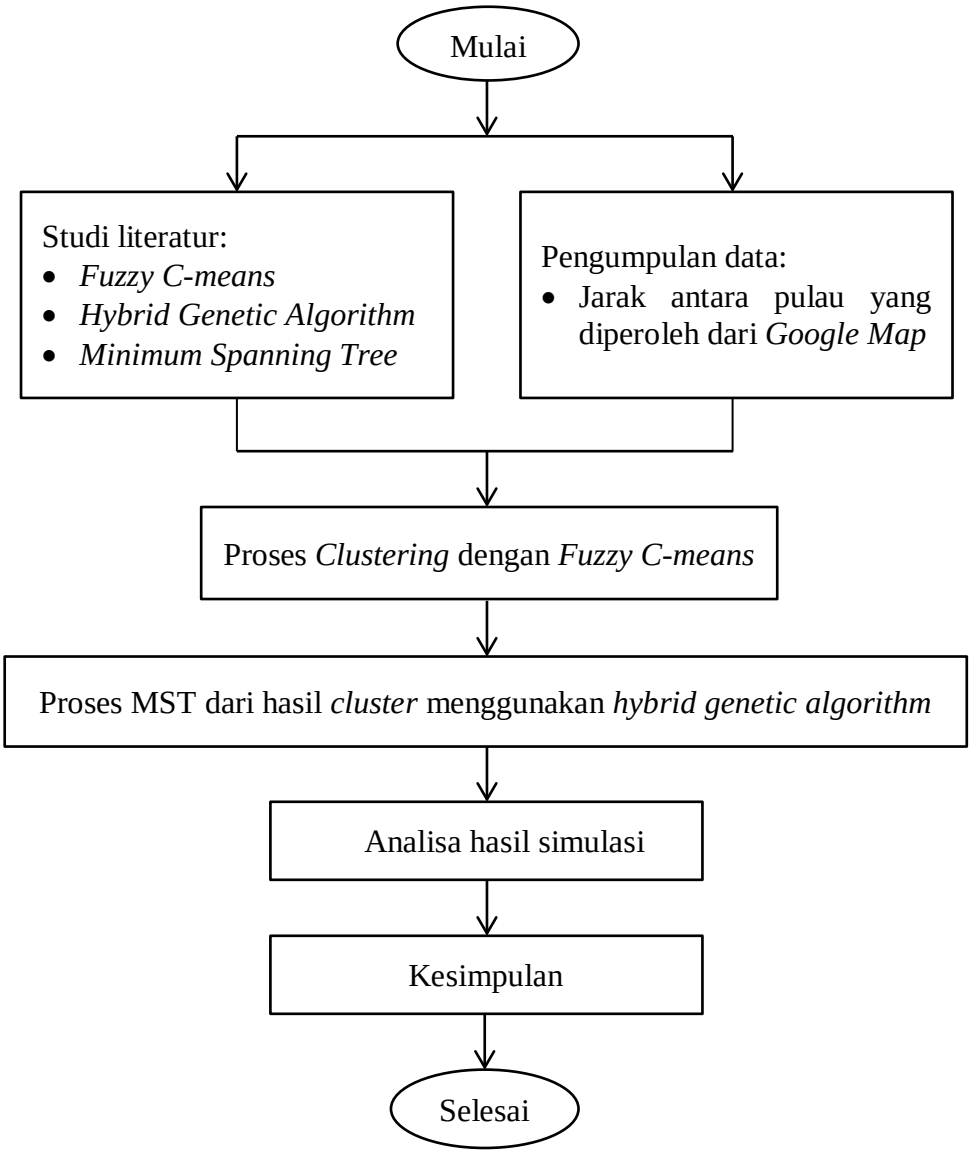
5. Analisa Hasil dan Pembahasan

Pada tahap ini akan dilakukan analisa dan pembahasan terhadap hasil simulasi yang telah didapatkan. Kemudian akan disusun kesimpulan-kesimpulan dari hasil yang diperoleh dalam penelitian ini.

6. Penyusunan Hasil Penelitian

Pada tahap ini akan dilakukan pembuatan laporan hasil penelitian yang dimulai dari Halaman Judul, Abstrak, Daftar Isi, Bab 1 sampai Bab 5, dan Daftar Pustaka.

Diagram alir tahap penelitian dapat ditunjukkan pada Gambar 3.1



Gambar 3.1 Diagram alir tahapan penelitian

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini dijabarkan hasil penelitian tentang penyelesaian pembentukan jaringan logistik menggunakan *Fuzzy C-Means* dan *Minimum Spanning Tree* berbasis *hybrid Genetik Algorithm* pada studi kasus di Kepulauan Maluku.

4.1 Implementasi Perangkat Lunak

Penelitian ini akan dibangun dalam lingkungan pengembangan dengan spesifikasi Intel Core I3 3110M 2.40GHz, RAM 2GB. Sistem operasi yang digunakan adalah Windows 7 32 bit dengan bahasa pemprograman Matlab R2010a.

4.2 Data Set

Data set yang digunakan pada penelitian ini merupakan data skunder yang diperoleh dari *Google Map* berupa titik koordinat. Pada data set terdapat dua fitur, yaitu garis lintang (*latitude*) dan garis bujur (*longitude*). Pada penelitian ini, data set tersebut tidak langsung digunakan untuk menentukan pola jaringan, akan tetapi dilakukan terlebih dahulu proses *clustering* menggunakan metode *Fuzzy C-means* untuk mendapatkan kelompok wilayah yang dikelompokkan berdasarkan kemiripan karakteristik koordinatnya. Proses pembentukan *minimum spanning tree* dilakukan setelah proses pengklusteran dengan membangun jalur *minimum spanning tree* masing-masing *cluster*.

Studi kasus yang digunakan dalam uji coba adalah studi kasus jaringan transportasi laut di Kepulauan Maluku pada 44 pulau. Persebarann pulau di Kepulauan Maluku dapat dilihat pada Gambar 4.1. Kemudian studi kasus jaringan logistik pada trasportasi laut dapat dimodelkan dalam bentuk graf.

Gambar 4.1 Persebaran Pulau di Kepulauan Maluku

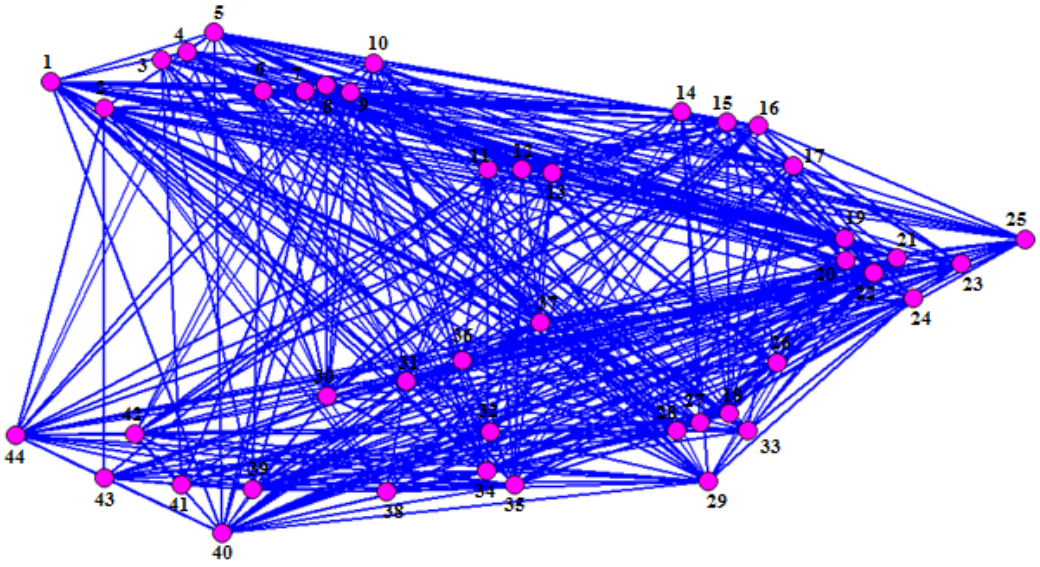
Untuk memudahkan penulisan rute, 44 data set berupa pulau (*node*) tersebut diberikan kode seperti Tabel 4.1 berikut.

Tabel 4.1 Kode 44 Pulau

Node	Nama Pulau	Node	Nama Pulau
1	Pulau Buru	16	Pulau Gorong
2	Pulau Ambelau	17	Pulau Kasiui
3	Pulau Manipa	18	Pulau Wotab
4	Pulau Kelang	19	Pulau Kur
5	Pulau Boano	20	Pulau Manggu
6	Pulau Ambon	21	Pulau Taam
7	Pulau Haruku	22	Pulau Tajondu
8	Pulau Saparua	23	Tual
9	Nusa Laut	24	Kai Tanimbar
10	Amahai	25	Pulau Kai Besar
11	Pulau Banda	26	Pulau Wolu
12	Pulau Run	27	Pulau Wuliaru
13	Pulau Rozengain	28	Pulau Selu
14	Pulau Ceram Laut	29	Adaut
15	Pulau Panjang	30	Pulau Serua
31	Pulau Nila	38	Regola

Node	Nama Pulau	Node	Nama Pulau
32	Lewa	39	Sera
33	Pulau Jamdena	40	Pulau Patti
34	Pulau Babar	41	Serwaru
35	Masela	42	Pulau Romang
36	Pulau Teun	43	Kisar
37	Pulau Damar	44	Pulau Wetar

Model graf dari studi kasus dapat digambarkan seperti Gambar 4.2 berikut.



Gambar 4.2 Model graf pada studi kasus

4.3 Proses *Clustering* menggunakan *Fuzzy C-means*

Penggunaan data set untuk proses *cluster* menghasilkan kelompok-kelompok wilayah sebanyak *cluster* yang diberikan. Uji coba *Fuzzy C-means* dilakukan untuk memilih hasil *cluster* yang memiliki total jarak kuadrat di antara setiap titik data dengan representasi *cluster* terdekat yang nilainya paling kecil.

Tabel 4.2 Titik Koordinat

Kode	Pulau	Garis Bujur (*Longitude*)	Garis Lintang (*Latitude*)
1	Pulau Buru	126.84477	-3.551656
2	Pulau Ambelau	127.187158	-3.855652

Kode	Pulau	Garis Bujur (*Longitude*)	Garis Lintang (L*atitude*)
3	Pulau Manipa	127.564931	-3.301472
4	Pulau Kelang	127.728244	-3.200381
5	Pulau Boano	127.904621	-2.985122
6	Pulau Ambon	128.216549	-3.656837
7	Pulau Haruku	128.489281	-3.652303
8	Pulau Saparua	128.628097	-3.593272
9	Nusa Laut	128.786025	-3.671457
10	Amahai	128.93482	-3.338037
11	Pulau Banda	129.89608	-4.545365
12	Pulau Run	129.681928	-4.555526
13	Pulau Rozengain	130.09337	-4.595815
14	Pulau Ceram Laut	130.932106	-3.886755
15	Pulau Panjang	131.228521	-4.005453
16	Pulau Gorong	131.43631	-4.046344
17	Pulau Kasiui	131.662864	-4.51643
18	Pulau Wotab	131.242643	-7.351233
19	Pulau Kur	131.990025	-5.339729
20	Pulau Manggu	132.002295	-5.586912
21	Pulau Taam	132.33542	-5.567877
22	Pulau Tajondu	132.184911	-5.7418
23	Tual	132.751852	-5.625556
24	Kai Tanimbar	132.440685	-6.024499
25	Pulau Kai Besar	133.167246	-5.348583
26	Pulau Wolu	131.552526	-6.762675
27	Pulau Wuliaru	131.052973	-7.448368
28	Pulau Selu	130.903374	-7.537538
29	Adaut	131.110482	-8.124175
30	Pulau Serua	130.016429	-6.310296
31	Pulau Nila	129.513599	-6.733204
32	Lewa	129.690278	-7.559742
33	Pulau Jamdena	131.361594	-7.538421
34	Pulau Babar	129.672503	-8.004632
35	Masela	129.854621	-8.156691
36	Pulau Teun	129.147385	-6.971403
37	Pulau Damar	128.635098	-7.152166
38	Regola	129.018682	-8.241491
39	Sera	128.154297	-8.214408
40	Pulau Patti	127.957913	-8.716017

Kode	Pulau	Garis Bujur (*Longitude*)	Garis Lintang (L*atitude*)
41	Serwaru	127.691305	-8.157831
42	Pulau Romang	127.386747	-7.587924
43	Kisar	127.18871	-8.077801
44	Pulau Wetar	126.617165	-7.590476

Mengukur jarak antara pulau yang disajikan pada Tabel 4.2 dapat menggunakan metode *Euclidean*, dengan persamaan berikut.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (4.1)$$

sehingga dari persamaan (4.1) dapat diimplementasikan menjadi persamaan berikut.

$$Jarak = \sqrt{(Lat_1 - Lat_2)^2 + (Long_1 - Long_2)^2} \qquad (4.2)$$

Hasil perhitungan (jarak) pada persamaan (4.2) masih dalam satuan *decimal degree* sehingga untuk menyesuaikan perlu dikalikan dengan 111.319 km (1 derajat bumi = 111.319km).

Pada studi kasus penelitian ini perhitungan jarak dapat implementasi sebagai berikut. Dimisalkan menghitung jarak dari Pulau Buru ke Pulau Ambelau, maka perhitungangannya sebagai berikut.

$$Jarak = \sqrt{\left(-3.551656 - (-3.855654)\right)^2 + (126.84477 - 127.187158)^2}$$
$$= 0.457868 \text{ x } 111.319\text{km}$$
$$= 50.96941\text{km}$$

Dari perhitungan yang telah dilakukan didapatkan jarak antarak Pulau Buru dengan Pulau Ambelau adalah 50.96941km. Dengan melakukan perhitungan yang sama akan diperoleh jarak antara pulau yang lainnya.

Beberapa langkah yang dilakukan dalam proses *clustering* dapat dijabarkan sebagai berikut.

1. Membentuk matriks X_{mxn}

Data ke-i	Atribut	
	Garis Bujur	Garis Lintang
1	126.84477	-3.551656
2	127.187158	-3.855652
3	127.564931	-3.301472
4	127.728244	-3.200381
5	127.904621	-2.985122
⋮		
44	126.617165	-7.590476

2. Menentukan parameter

Banyaknya cluster	c	3
Pembobotan	w	2
Maksimum Iterasi	MaksIter	30
Error	e	0.0001
Fungsi Objektif	PO	0
Iterasi Awal	itrer	1

3. Membangkitkan bilangan acak sebagai elemen matriks U_{ik}, $i =$ banyaknya data dan $k =$ banyaknya *cluster* (dengan nilai antara $0 - 1$).

I	k1	k2	k3
1	0.3	0.4	0.9
2	0.2	0.7	0.8
3	0.4	0.8	0.4
4	0.6	0.5	0.6
5	0.5	0.9	0.2
⋮			
44	0.3	0.2	0.4

4. Menghitung pusat *cluster*

I	Uik			Xij		Uik^w		
	1	2	3	1	2	1	2	3
1	0.3	0.4	0.9	126.8448	-3.551656	0.09	0.16	0.81

I	Uik			Xij		Uik^w		
2	0.2	0.7	0.8	127.1872	-3.855652	0.04	0.49	0.64
3	0.4	0.8	0.4	127.5649	-3.301472	0.16	0.64	0.16
4	0.6	0.5	0.6	127.7282	-3.200381	0.36	0.25	0.36
5	0.5	0.9	0.2	127.9046	-2.985122	0.25	0.81	0.04
⋮								
44	0.3	0.2	0.4	126.6172	-7.590476	0.09	0.04	0.16

Dengan menggunakan persamaan (2.9) maka diperoleh pusat *cluster* sebagai berikut.

Vkj	1	2
1	131.8542493	-5.64206874
2	128.78925	-7.73759083
3	128.2824091	-3.62902941

5. Menghitung fungsi objektif

I	k1		k2		k3	
	(Xi1-Vi1)^2	(Xi2-Vi1)^2	(Xi1-Vi2)^2	(Xi2-Vi2)^2	(Xi1-Vi3)^2	(Xi2-Vi3)^2
1	25.0948824	4.36982541	3.781002459	17.52205039	2.066806311	0.005986645
2	21.7817408	3.191284757	2.566698767	15.06944907	1.199575071	0.051357796
3	18.3982511	5.478393084	1.498957006	19.67915026	0.514774889	0.10729386
4	17.0239193	5.961839003	1.125733726	20.58627303	0.307099008	0.183739464
5	15.5995633	7.059365961	0.782568462	22.58595997	0.142723882	0.414616759
⋮						
44	27.4270515	3.796290864	4.717953234	0.021642773	2.773038062	15.69305905

Dengan menggunakan persamaan (2.10) maka akan diperoleh fungsi objektif untuk mendapatkan pusat *cluster* yang tepat.

6. Memperbaharui U

Dengan menggunakan persamaan (2.11) maka akan didapatkan perubahan derajat keanggotaan setiap data pada *cluster* (memperbarui matriks U)

I	k1	k2	k3
1	0.060247859	0.083330102	0.856422039
2	0.04468368	0.0632727	0.89204362

I	k1	k2	k3
3	0.024685222	0.02783064	0.947484138
4	0.020454815	0.021654813	0.957890372
5	0.023460369	0.022747981	0.953791649
⋮			
44	0.107774748	0.709994658	0.182230594

7. Mengecek kondisi berhenti

Cek kondisi berhenti Apakah iter > maxIter? Apakah |P1-P0| < e? Jika syarat berhenti belum terpenuhi maka ulangi langkah ke-4.

Data inputan yang diberikan pada Tabel 4.2 disajikan dalam bentuk *Ms. Excel*, jumlah *cluster* yang akan digunakan adalah 3, 4, dan 5. Setelah dilakukan pengujian pada masing-masing jumlah *cluster* dilakukan validasi untuk mengevaluasi hasil yang telah diperoleh. Proses evaluasi hasil *clustering* bertujuan untuk menentukan *cluster* terbaik yang dapat dilakukan dengan menggunakan indeks validitas *cluster.*

Pada tesis ini indeks validitas yang digunakan adalah *Partition Coefficient (PC)*, *Classification Entropy (CE)*, dan *Xie and Beni's index (XB).* Penggunaan indeks ini untuk pengelompokan, karena memiliki ketepatan dan kehandalan yang tinggi untuk digunakan sebagai kriteria dalam menentukan jumlah kelompok yang optimum.

Jumlah *cluster* optimal ditentukan dengan validitas indeks *cluster* melalui perbandingan nilai indeks. Perhitungan nilai indeks dilakukan dengan parameter yang telah ditentukan, yaitu m = 2, $\varepsilon = 10^{-3}$, dan $c = 3,4, dan\, 5$. Proses validasi dilakukan dengan bantuan MATLAB.

4.4 Proses pembentukan *Minimum Spanning Tree* berbasis *Hybrid Genetic Algorithm*

Hybrid Genetic Algorithm merupakan kombinasi metode heuristik lain ke dalam algoritma genetika dengan harapan mampu meningkatkan kinerja algoritma genetika. Hibridasi ini diharapkan mampu memberikan solusi lain yang lebih baik disekitar lokal optimum yang diberikan oleh algoritma genetika yang biasa

dikenal dengan istilah pencarian lokal *(local search*). *Hybrid Genetic Algorithm* menggabungkan algoritma genetika dengan *local search* untuk menghasilkan solusi yang optimal.

Dalam proses pembentukan *minimum spanning tree* berbasis *hybrid genetic algorithm* terdapat beberapa tahapan.

4.4.1 Inisialisasi Parameter

Tahap awal yang dilakukan sebelum melakukan tahapan yang lainnya adalah inisialisasi parameter. Inisialisasi parameter ini digunakan untuk menentukan 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 Tesis ini besar populasi diinisialisasikan dengan nilai 50, 75, 100.
2. Generasi maksimum adalah banyaknya generasi yang akan dibuat. Generasi maksimum berpengaruh pada pemberhentian iterasi pada sistem. Dalam Tesis ini generasi maksimum diinisialisasikan dengan nilai 500, 1000, dan 2000.
3. *Crossover rate* adalah peluang yang digunakan untuk menentukan peluang kromosom yang dipindah silang. *Crossover rate* mempunyai range antara 0-1. Dalam Tesis ini crossover rate diinisialisasikan dengan nilai 0,2 [9].
4. *Mutation rate* adalah peluang yang digunakan untuk menentukan peluang gen yang dimutasi. *Mutation rate* mempunyai range antara 0-1. Dalam tesis ini *mutation rate* diinisialisasikan dengan nilai 0,4 [9].

4.4.2 Membangkitkan Kromosom (Inisialisasi Populasi)

Tahap inisialisasi populasi ini bertujuan untuk membangkitkan sejumlah kromosom (solusi awal). Setiap populasi berisi sekumpulan kromosom. Kromosom yang telah dibangkitkan pada tahap *decode* kromosom dimasukkan kedalam tahap inisialisasi populasi.

Teknik pembangkitan populasi awal yang digunakan dalam Tesis ini yaitu *random generator*. *Random generator* melibatkan pembangkitan bilangan random untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.

Proses *generate* populasi awal dimulai dengan melakukan *generate* kromosom dengan membangkitkan gen-gen secara acak. Setiap kromosom berisi gen-gen yang merepresentasikan nomor pulau. Jumlah gen dalam setiap kromosom adalah sama dengan jumlah pulau yang dinotasikan dengan m. Jika gen-gen dalam suatu kromosom tidak ada yang rangkap, maka kromosom tersebut dipilih untuk dimasukkan dalam populasi awal. Proses ini dilakukan sebanyak n yang merupakan *popsize*.

Untuk merepresentasikan gen-gen pada kromosom dapat ditunjukkan pada Gambar 4.3 berikut.

g_{11}	g_{12}	g_{13}	$\cdots$	g_{1m}	Kromosom 1
g_{21}	g_{22}	g_{11}	$\cdots$	g_{2m}	Kromosom 2
g_{21}	g_{11}	g_{11}	$\cdots$	g_{3m}	Kromosom 3
.					
.					
.					
g_{n1}	g_{n2}	g_{n3}	$\cdots$	g_{nm}	Kromosom n

Gambar 4.3 Representasi gen dalam kromosom

Berdasarkan Gambar 4.3 representasi kromosom pada studi kasus dapat disajikan pada Gambar 4.4 berikut.

1	2	6	3	5	4	7	8	10	11	12	9	K 1
2	3	5	1	6	7	4	9	11	10	9	12	K 2
2	6	1	3	5	8	4	10	12	9	12	11	K 3
.												
.												
.												
g_{n1}	g_{n2}	g_{n3}	g_{n4}	g_{n5}	g_{n6}	g_{n7}	g_{n8}	g_{n9}	g_{n10}	g_{n11}	g_{n12}	K n

Gambar 4.4 Representasi Kromosom pada Cluster 1

4.4.3 Evaluasi

Evaluasi adalah metode untuk menghitung nilai *fitness* pada setiap kromosom yang telah dibangkitkan secara random pada tahap inisialisasi populasi. Nilai *fitness* dari setiap koromosom dihitung berdasarkan panjang jalur linier yang dihasilkan dari jumlah jarak keseluruhan dari urutan node-node yang dilalui. Dalam masalah optimasi pada Tesis ini individu (kromosom) yang bernilai *fitness* yang tinggi yang akan bertahan hidup atau yang akan terpilih dan kromosom yang bernilai rendah akan mati atau tidak terpilih pada tahap selanjutnya. Karena solusi yang dicari adalah meminimalkan sebuah fungsi *h*, maka nilai *fitness* yang dicari adalah kromosom yang memiliki panjang jalur yang pendek.

Oleh karena itu, rumus untuk mencari nilai *fitness* pada masalah minimasi ini adalah:

$$f = \frac{1}{h} \tag{4.6}$$

keterangan :

f = fungsi fitness

h = fungsi yang akan dimaksimasi / diminimasi (TJ= Total Jarak)

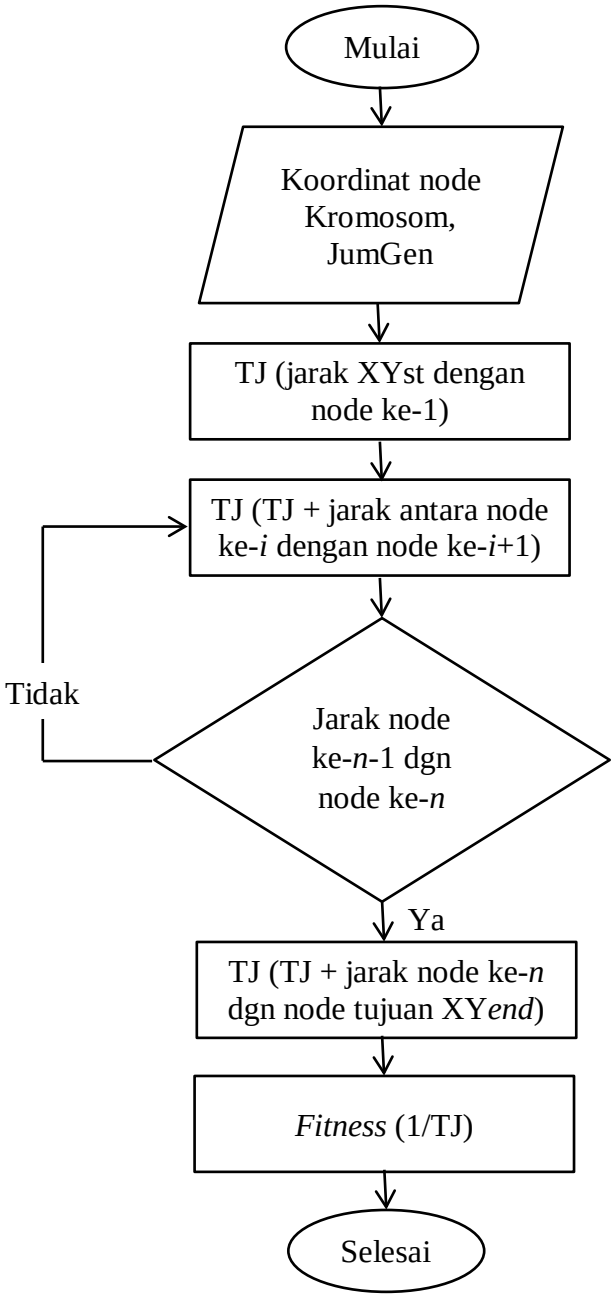
Pada tahap ini koordinat node *XYnode*, *Populasi* dan jumlah node *JumGen* merupakan input. Mula-mula hitung jarak antara node asal *XYst* dengan node pertama simpan hasilnya pada variabel *TJ* (total jarak), kemudian jumlahkan dengan jarak-jarak dari setiap node berikutnya yang dilalui yakni sebanyak *JumGen*. Setelah itu jumlahkan dengan jarak dari node terakhir ke node tujuan *XYend*. Total jarak disimpan pada variabel *TJ*.

Pada *minimum spanning tree*, permasalahannya adalah meminimalkan total biaya (masalah minimasi). Dalam hal ini yang dimaksud total biaya adalah total jarak kartesian antara satu node dengan node lainnya. Jarak kartesian antara node A dan node B dapat dihitung dengan rumus :

$$\|A - B\| = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2} \qquad (4.7)$$

(X_A, Y_A) menyatakan posisi koordinat node A dan (X_B, Y_B) adalah posisi koordinat node B. Setelah didapat total jarak atau panjang jalur satu kromosom, yang terakhir menghitung nilai *fitness*-nya dengan menggunakan rumus 4.6. Hasilnya disimpan pada variabel *Fitness* yang merupakan argumen output fungsi seperti halnya *Populasi*. Nilai *fitness* ini merupakan input bagi proses berikutnya pada program utama.

Algoritma program dari prosedur evaluasi individu dapat ditunjukkan pada Gambar 4.5 berikut.



Gambar 4.5 *Flowchart* evaluasi individu

Pada program utama, tahap evaluasi individu ini dilakukan (di*looping*) sebanyak *PopSize*. Sehingga didapat nilai *fitness* dari semua kromosom dalam satu populasi. Nilai *fitness* suatu kromosom ini kemudian akan dibandingkan dengan *fitness*-*fitness* kromosom yang lainnya yang ada pada semua generasi. Dimana nilai *fitness* paling tinggi yang akan terpilih.

4.4.4 Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit*. Kemampuan algoritma genetik untuk memproduksi kromosom yang lebih baik secara progresif tergantung pada penekanan selektif (*selective pressure*) yang diterapkan ke populasi. Penerapan penekanan selektif adalah memilih orang tua yang lebih baik ketika membuat keturunan baru. Dengan metode ini, hanya kromosom sebanyak yang dipelihara dalam populasi yang perlu dibuat bagi generasi berikutnya. Walaupun penekanan selektif tidak diterapkan ke level keturunan, metode ini akan terus menghasilkan kromosom yang lebih baik, karena adanya penekanan selektif yang diterapkan ke orangtua.

Ada beberapa metode untuk memilih kromosom yang dapat digunakan antara lain adalah seleksi roda rolet (*roulette wheel selection*), seleksi ranking (*rank selection*) dan seleksi turnamen (*tournament selection*). Dalam Tesis ini, metode yang digunakan adalah *roulette wheel selection*. Pada seleksi ini, orang tua dipilih berdasarkan *fitness*-nya. Lebih baik kualitas suatu kromosom, lebih besar peluangnya untuk terpilih. Sebuah bilangan random akan dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan random tersebut akan diseleksi. Proses ini diulang hingga diperoleh sejumlah individu yang diharapkan.

Pada proses *roulette wheel* ini akan dihitung nilai kumulatif dari probabilitas *fitness* masing-masing kromosom.

$$P[i] = \frac{fitness[i]}{jumlah\ fitness} \tag{4.8}$$

$$C[i] = \sum_{k=1}^{i} P[k] \tag{4.9}$$

Keterangan :

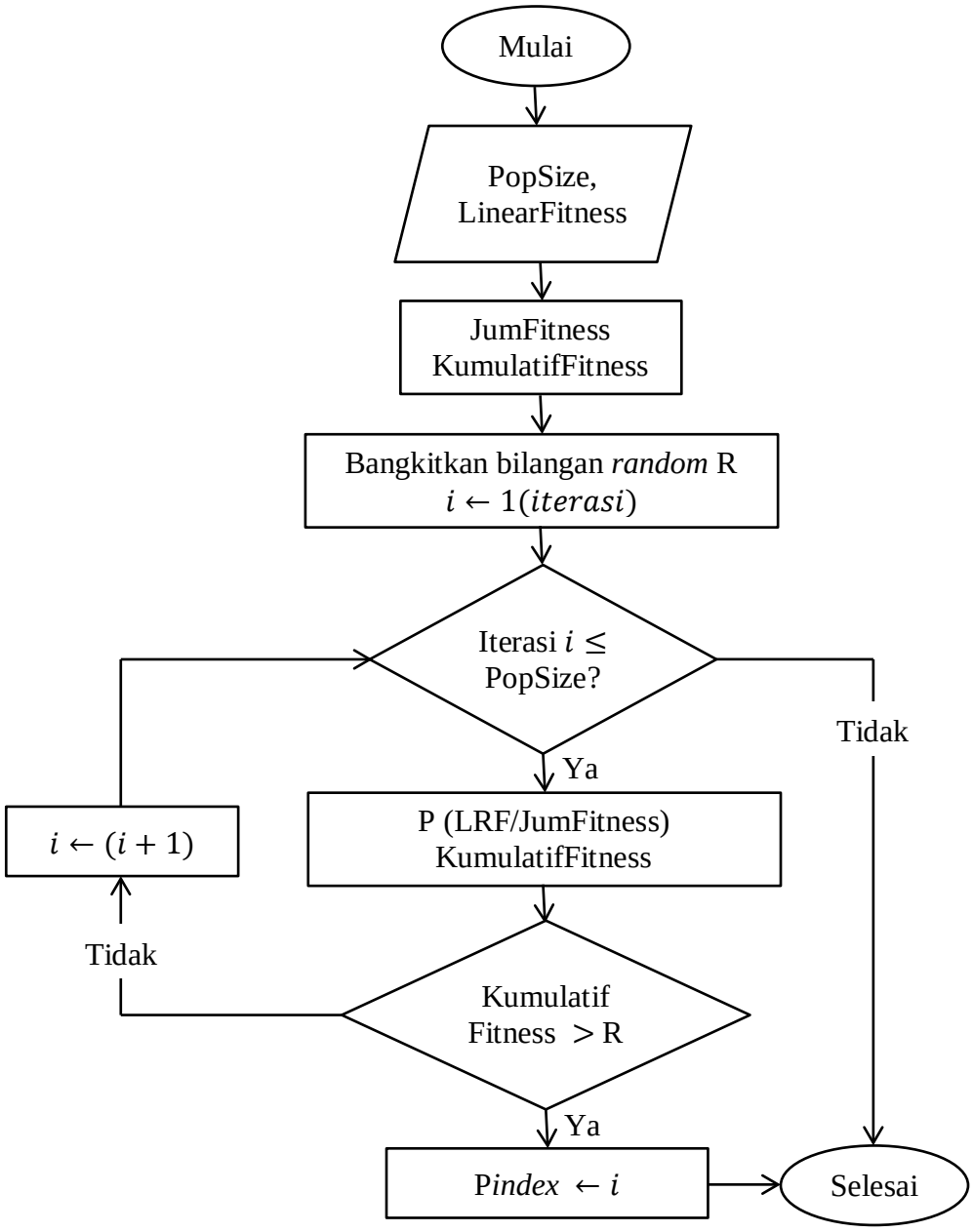
$P[i]$ = probabilitas *fitness[i]*

$C[i]$ = nilai komulatif indeks ke-i

i = indeks kromosom $(1,2,3,\dots,n)$

k = counter $(1,2,3,\dots,n)$

Algoritma program dari prosedur *roulette wheel* dapat ditunjukkan pada Gambar 4.6 berikut.



Gambar 4.6 Diagram Alir *roulette wheel selection*

Proses *roulette wheel* ini dikendalikan oleh sebuah bilangan random (acak) *R* yang dibangkitkan oleh program pada interval [0,1). Apabila nilai kumulatif

kurang dari bilangan *random* yang dibangkitkan (C[i] > *R*), maka kromosom dengan indeks-*i* akan terpilih sebagai induk atau individu generasi berikutnya. Indeks dari kromosom yang terpilih ini disimpan pada sebuah variabel *Pindex* yang merupakan nama fungsinya. *Pindex* ini merupakan input untuk proses-proses berikutnya. Proses *roulette wheel* diputar sebanyak ukuran populalsi (*PopSize*).

4.4.5 Pindah Silang *(Crossover)*

Metode *crossover* merupakan metode yang digunakan untuk memindahsilangkan gen-gen pada kromosom *parent* sehingga menghasilkan kromosom *offspring* dengan gen hasil pindahsilang. Tahap pindah silang bertujuan untuk mengubah nilai gen induk secara random, hal ini berlaku untuk kromosom real dan integer, sedangkan untuk kromosom biner pindah silang (*crossover*) dilakukan untuk menukarkan bagian gen dari induk (kromosom *parent*) secara random, sehingga hasil dari pindah silang (*crossover*) tersebut berupa kromosom anak (offspring) yang mempunyai sifat sebagian atau mirip dengan induknya. Terdapat beberapa metode pada *crossover*, yaitu *one point crossover*, *two point crossover*, *uniform crossover*, dan *arithmetic crossover*. Metode yang digunakan pada Tesisi ini adalah *arithmetic crossover*.

Aritmatic Crossover digunakan untuk kondisi bilangan real pada kedua *parent* tersebut. Pada proses crossover terdapat satu parameter yang sangat penting yaitu probabilitas crossover (*Pc*). Probabilitas crossover ini digunakan untuk menentukan kromosom yang akan mengalami crossover. Crossover ini bertujuan untuk memindahsilangkan bagian kromosom parent, sehingga dihasilkan dua buah kromosom anak.

Nilai variabel *offspring* dipilih di sekitar dan antara nilai-nilai variabel *parent*. Crossover dilakukan dengan menggunakan nilai *alpha* sebagai bilangan random lebih dari 0 dan kurang dari 1. Selain itu ditentukan posisi dari gen yang dilakukan menggunakan bilangan random. *Offspring* dihasilkan menurut aturan sebagai berikut:

$$x_1'(k) = alpha \cdot x_1(k) + (1 - alpha) \cdot x_2(k) \qquad (4.4)$$

$$x_2'(k) = alpha \cdot x_2(k) + (1 - alpha) \cdot x_1(k) \quad (4.5)$$

Hasil akhir dari prosedur pindah silang ini adalah dua buah kromosom anak hasil persilangan atau perkawinan dua induk. Anak ini merupakan input untuk proses berikutnya yakni mutasi.

Pada program utama kromosom yang akan dipindahsilangkan diplilih secara acak dengan membangkitkan nilai acak *R* pada interval [0,1). Jumlah kromosom yang akan dipindahsilangkan juga dipengaruhi probabilitas pindah silang (*Pc*) yang besarnya telah ditentukan pada tahap inisialisai populasi. Pindah silang dapat terjadi apabila nilai random yang dibangkitkan *R* lebih kecil dari probabilitas pindah silang *Pc* (*R* < *Pc*). Sehingga banyaknya pindah silang yang akan terjadi pada setiap generasinya adalah *Pc x PopSize*.

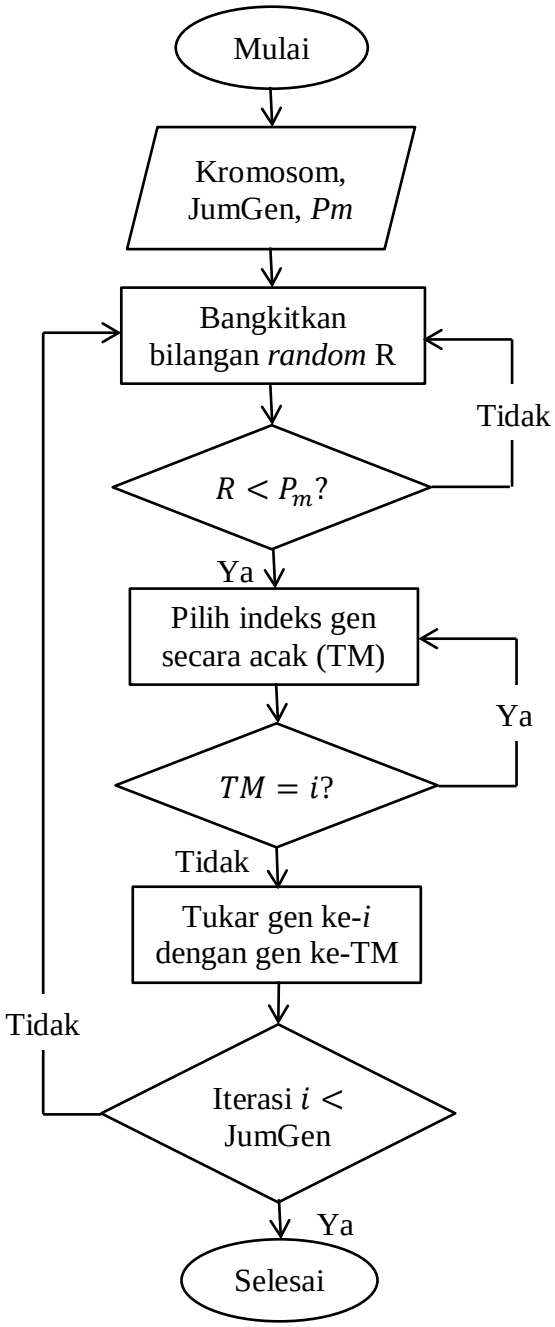
4.4.6 Mutation

Mutation adalah metode rekombinasi kromosom dengan cara memilih kromosom yang akan dimutasi, dan kemudian menemukan titik mutasi pada kromosom tersebut secara acak pula. Mutasi bertujuan untuk mengubah nilai gen secara acak. Jika kromosom berbentuk real maka akan mengubah nilai gen dengan mengurangkan dan menambahkan nilai gen tersebut. Pada mutasi ini sangat dimungkinkan munculnya kromosom baru yang semula belum muncul dalam populasi awal.

Pada mutasi ada satu parameter yang sangat penting yaitu probabilitas mutasi (*Pm*). Probabilitas mutasi ini akan digunakan untuk menentukan kromosom yang akan dimutasi. Langkah pertama yang harus dilakukan adalah membangkitkan bilangan random, jika bilangan random yang dibangkitkan kurang dari probabilitas mutasi maka kromosom tersebut mengalami mutasi, dan sebaliknya jika bilangan random lebih besar dari probabilitas mutasi maka kromosom tersebut tidak mengalami mutasi.

Dalam tahapan *mutation* ini terdapat beberapa metode yang dapat digunakan, diantaranya mutasi penukaran (*Swap Mutation*), mutasi penyisipan (*Insert Mutation*), mutasi pembalikan (*Invertion Mutation*), dan mutasi

penggantian (*Flip mutation*). Metode mutation yang digunakan dalam Tesis ini adalah *Swap Mutation*.



Gambar 4.7 Diagram Alir *Swap Mutation*

Gambar 4.7 adalah algoritma program dari prosedur mutasi. Dengan skema *swap mutation* ini muatasi dilakukan dengan cara menukarkan gen-gen

yang dipilih secara acak dengan gen yang dipilih secarak acak juga. Jumlah kromosom yang mengalami mutasi dalam satu populasi ditentukan oleh parameter probabilitas mutasi *Pm*. Diperkirakan total gen yang mengalami mutasi pada seluruh generasi adalah *Pm x PopSize x MaxG*.

Metode *swapping* dilakukan dengan memilih dua buah gen yang ada pada kromosom, jika bilangan random [0,1) yang dibangkitkan kurang dari probabilitas mutasi maka nilai gen tersebut akan ditukarkan dengan nilai gen lainnya yang dipilih secara random. Penukaran terarah seperti yang dilakukan metode ini dinilai lebih efisien dan mempercepat proses perbedaan di antara individu.

Input pada prosedur ini adalah kromosom-kromosom baru hasil pindah silang, jumlah gen *JumGen* dan probabilitas mutasi *Pm*. Bangkitkan bilangan acak sebuah bilangan *R*, apabila nilai *R* lebih kecil dari probabilitas mutasi *Pm* maka pilih gen secara acak, indeks dari gen yang terpilih ini disimpan pada pvariabel *TM*. Apabila *TM* sama dengan nilai iterasi *i* saat itu maka cari lagi nilai *TM* secara random sampai didapat nilai indeks yang tidak sama dengan nilai itersi *i* pada saat itu. Terakhir tukar posisi gen yang ada pada indeks ke-*TM* dengan gen pada indeks ke-*i*. Pada program utama utama proses mutasi ini dieksekusi sebanyak jumlah populasi *PopSize*. Setelah proses muatasi selesai, maka akan didapatkan populasi baru.

4.4.7 Pencarian Lokal (*Local Search*)

Pencarian lokal (*local search*) dapat dilakukan dengan menukar dua gen, atau permutasi beberapa gen tanpa mengurangi kualitas kromosom sebelumnya. Pada *hybrid genetic algorithm*, tahapan *local search* diterapkan pada proses evolusi, yaitu sesudah proses mutasi namun tidak diterapkan untuk semua kromosom dalam populasi.

4.4.8 Pembentukan Populasi Baru

Tahap pembentukan populasi baru merupakan 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. Kromosom-kromosom anak yang

dihasilkan oleh *crossover* dan mutasi, langsung menggantikan kromosom-kromosom terjelek dalam populasi.

4.5 Implementasi dengan MATLAB

4.5.1 Implementasi *Fuzzy C-Means* dengan MATLAB

Langkah pertama dalam implementasi proses *cluster* yaitu dengan melakukan proses pengklusteran dengan menginputkankan data set yang disajikan dalam bentuk *Ms. Excel* dan banyaknya *cluster* yang diinginkan. Dalam tesis ini jumlah *cluster* yang akan diujikan yaitu 3, 4, dan 5. Berikut implementasi proses FCM dalam MATLAB.

```
clc, clear, close all;

data=xlsread('datamaluku.xlsx','empat');
data=data(:, :);

cluster=3;      %Banyaknya kluster (3,4,5)
CM=jet(cluster);
[center,U,obj_fcn]=fcm(data,cluster);

[~,y]=max(U);   %Mengambil nilai keanggotaan terbesar sebagai
                 cluster yang diikuti

U=U';
y=y';
f1=figure(1);clf
plot(data(:,1),data(:,2),'+k')
hold on
for i=1:cluster
    plot(data(y==i,1),data(y==i,2),'o','color',CM(i,:))
    plot(center(i,1),center(i,2),'xr','MarkerSize',12,...
        'LineWidth',5)
end
hold off
```

Langkah selanjutnya adalah proses validasi untuk menetukan jumlah *cluster* terbaik yang akan digunakan dalam pembentukan jaringan logistik. Penentuan *cluster* terbaik dapat dilakukan dengan menggunakan indeks validitas *cluster*, dalam tesis ini indeks validitas yang digunakan adalah *Partition Coefficient (PC)*, *Classification Entropy (CE)*, dan *Xie and Beni's index (XB)*. Proses validasi diimplementasikan dalam fungsi `validity` berikut.

```
function result = validity(result)
%validation of the clustering

N = size(result.data.f,1);

m = 2;

     %partition coefficient (PC)
      fm = (result.data.f).^m;
      PC = (1/N)*sum(sum(fm));

     %classification entropy (CE)
      fm = (result.data.f).*log(result.data.f);
      CE = (-1/N)*sum(sum(fm));

     %Xie and Beni's index (XB)
      XB =(sum(sum(result.data.d.*result.data.f.^2))./...
            (N*min(result.data.d)));

     %results
      result.validity.PC = PC;
      result.validity.CE = CE;
      result.validity.XB = XB;

 end
```

Sebagai program utama `FCMcall.m`, memanggil fungsi `validity`. Proses ini diimplementasikan pada program berikut.

```
clear all
close all
path(path,'..\..\FUZZCLUST')

% data
load luku.txt
data.X = luku(:,:);

%normalization
data=clust_normalize(data,'range');

%parameters
param.c=3;
param.m=2;
param.e=1e-3;

%FCM clustering
result = FCMclust(data,param);

%validation
result = validity(result);

%evaluation
result.validity
```

4.5.2 Implementasi *Hybrid Genetic Algorithm* dengan MATLAB

Dalam Tesis ini digunakan suatu fungsi random yang disediakan oleh MATLAB untuk membangkitkan sebuah bilang real. Kode program untuk membangkitkan populasi awal ini terdapat dalam program utama dengan perintah sebagai berikut :

```
pop=repmat(empty_individual,nPop,1);

for i=1:nPop

    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);

end
```

Inputan untuk perintah ini adalah `nPop` (ukuran populasi atau jumlah kromosom dalam populasi). `repmat` menyatakan pembangkit matriks dalam interval `nPop`x1, dengan komponen `empty_individual`. Kemudian perintah `unifrnd` menyatakan pembangkikan matriks sebelumnya yang berisi bilangan random dalam dengan inputan *VarMin, VarMax,* dan *VarSize*. Dimana *VarMin* merupakan batas bawah, *VarMax* adalah batas atas, dan *VarSize* adalah ukuran matriks.

Selanjutnya dilakukan perhitungan jarak kartesian yang diimplementasikan menggunakan fungsi `sqrt`, yang sudah tersedia di dalam MATLAB. Sedangkan perhitungan nilai *fitness* diimplementasikan dalam fungsi `CreateModel` yang merupakan total biaya, fungsi ini akan dipanggil pada program utama untuk menghitung nilai *fitness*-nya. Variabel X dan Y berisi koordinat-koordinat dari semua *node* (pulau).

```
function model=CreateModel()

    X=xlsread('MST.xlsx','GAC2');
    X=X(:, 1);
    Y=xlsread('MST.xlsx','GAC2');
    Y=Y(:, 2);

    n=numel(X);

    d=zeros(n,n);
```

```
    for i=1:n
        for j=i+1:n
            d(i,j)=sqrt((X(i)-X(j))^2+(Y(i)-Y(j))^2);
            d(j,i)=d(i,j);
        end
    end

    model.n=n;
    model.X=X;
    model.Y=Y;
    model.d=d;

end
```

Pada program utama, tahap evaluasi individu ini dilakukan (di*looping*) sebanyak *PopSize*. Sehingga didapat nilai *fitness* dari semua kromosom dalam satu populasi. Nilai *fitness* suatu kromosom ini kemudian akan dibandingkan dengan *fitness-fitness* kromosom yang lainnya yang ada pada semua generasi. Dimana nilai *fitness* paling tinggi yang akan terpilih.

Proses seleksi yang digunakan adalah *roulette wheel selection*. Fungsi *roulette wheel* dapat diimplementasikan dengan kode pemprograman berikut ini.

```
function i=RouletteWheelSelection(P)

    r=rand;

    C=cumsum(P);

    i=find(r<=C,1,'first');

end
```

Proses *roulette wheel* ini dikendalikan oleh sebuah bilangan random (acak) *r* yang dibangkitkan oleh program pada interval [0,1). Apabila nilai kumulatif lebih besar dari bilangan *random* yang dibangkitkan $(r \leq C)$, maka kromosom dengan indeks-*i* akan terpilih sebagai induk atau individu generasi berikutnya.

Crossover untuk MST dapat diimplementasikan dengan skema *Aritmatic Crossover*.

```
function [y1, y2]=Crossover(x1,x2,gamma,VarMin,VarMax)

    alpha=rand(size(x1));

    if alpha < gamma

    else
    y1=alpha.*x1+(1-alpha).*x2;
    y2=alpha.*x2+(1-alpha).*x1;

    y1=max(y1,VarMin);
    y1=min(y1,VarMax);

    y2=max(y2,VarMin);
    y2=min(y2,VarMax);

    end
end
```

Pada program utama kromosom yang akan dipindahsilangkan diplilih secara acak dengan membangkitkan nilai acak *alpha* pada interval [0,1). Jumlah kromosom yang akan dipindahsilangkan juga dipengaruhi probabilitas *crossover Pc* (*gamma*) yang besarnya telah ditentukan pada tahap inisialisai populasi. *Crossover* dapat terjadi apabila nilai random yang dibangkitkan (*alpha*) lebih kecil dari probabilitas *crossover Pc* ($alpha < gamma$). Sehingga banyaknya *crossover* yang akan terjadi pada setiap generasinya adalah *Pc x PopSize*.

Setelah dilakukan proses *crossover*, langkah selanjutnya adalah mutasi. Metode *mutation* yang digunakan dalam Tesis ini adalah *Swap Mutation*. Metode *swapping* dilakukan dengan memilih dua buah gen yang ada pada kromosom, jika bilangan random [0,1) yang dibangkitkan kurang dari probabilitas mutasi maka nilai gen tersebut akan ditukarkan dengan nilai gen lainnya yang dipilih secara random.

```
function y=Mutate(x,mu,VarMin,VarMax)

    y=x;
    if rand<0.5
        r = rand(size(x));
        A = (r<=mu);
        y(A) = 1 - x(A);
```

```
    else
        A = find(x>=0.5);
        B = find(x<0.5);
        i1 = randsample(A, 1);
        i2 = randsample(B, 1);
        y([i1 i2]) = x([i2 i1]);

        y([i1 i2])=max(y([i1 i2]),VarMin);
        y([i1 i2])=min(y([i1 i2]),VarMax);
    end

end
```

Fungsi `Mutate` merupakan implementasi *swapping mutation*. Masukan untuk fungsi ini adalah `x,mu,VarMin,` dan `VarMax`. Untuk semua gen dalam kromosom, jika bilangan random yang dibangkitkan kurang dari `mu`, maka nilai gen tersebut akan ditukarkan dengan nilai gen lain yang dipilih secara random. Keluaran dari fungsin ini adalah `y`, yaitu kromosom yang sudah termutasi.

Selanjutnya melakukan tahap *local search*, *local search* dilakukan setelah proses mutasi, tetapi tidak diterapkan untuk semua kromosom dalam populasi. *Local search* dilakukan dengan menukar dua gen tanpa mengurangi kualitas kromosom sebelumnya. Implementasi *local search* ditunjukkan pada kode program berikut.

```
% Local Search based on Mutation
for k=1:nm/2
        NewIndividual=BestSol;
        NewIndividual.Position=Mutate(BestSol.Position, mu,
VarMin,VarMax);
        [NewIndividual.Cost, NewIndividual.Sol]=CostFunction...
            (NewIndividual.Position);
        if NewIndividual.Cost<=BestSol.Cost
            BestSol=NewIndividual;
        end
end
```

4.6 Pengujian dan Analisa Hasil

Uji coba dilakukan dengan menggunakan Matlab. Pada penelitian ini disajikan dua proses pengujian yaitu proses pengklusteran dengan metode FCM dan proses MST berbasis *hybrid algorithm genetic*. Pada proses *clustering* tahapan yang dilakukan adalah menentukan jumlah *cluster* dan validasi jumlah *cluster* yang diujikan. Sedangkan proses pengujian untuk MST berbasis *hybrid algorithm genetic* dilakukan penentuan PopSize dan iterasi. Tahap selanjutnya menentukan jaringan logistik dari pengujian dari proses *clustering* dan MST berbasis *hybrid algorithm genetic*.

4.6.1 Pengujian Proses *Cluster* dengan *Fuzzy C-Means*

Untuk menentukan banyaknya *cluster* yang diberikan pengujian dilakukan dengan berbagai variasi *cluster*. Pada proses ini banyaknya *cluster* yang digunakan pada data set adalah K=3, K=4 dan K=5. Hasil dari masing-masing jumlah *cluster* akan di validasi menggunakan *Partition Coefficient (PC)*, *Classification Entropy (CE)*, dan *Xie and Beni's index (XB)* untuk mendapatkan jumlah *cluster* terbaik. Hasil validasi jumlah *cluster* dapat disajikan pada Tabel 4.3 berikut.

Tabel 4.3 Nilai Validitas Jumlah *Cluster*

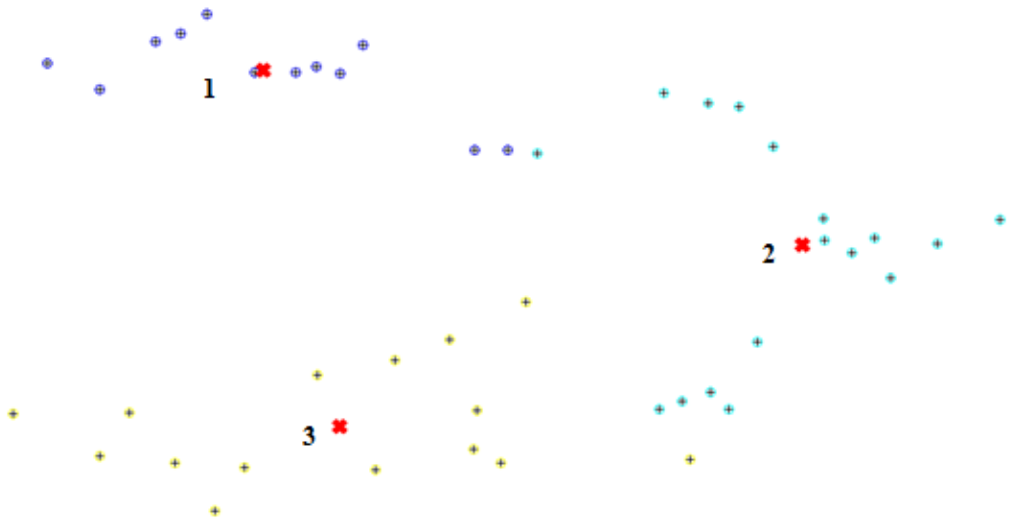
Jumlah *Cluster*	*XB*	*PC*	*CE*
3	0.0829	0.7339	0.4857
4	0.1011	0.7132	0.5696
5	0.0985	0.7095	0.6162

Berdasarkan Tabel 4.3 menunjukkan hasil indeks XB pada jumlah *cluster* 3 lebih rendah yaitu 0.0829 dibandingkan dengan jumlah *cluster* 4 dan 5 dengan nilai masing-masing 0.1011 dan 0.0985. Pada indeks validasi PC didapatkan nilai validasi untuk jumlah cluster 3 sebesar 0.7339, jumlah cluster 4 sebesar 0.7132, sedangkan jumlah cluster 5 sebesar 0.7095. Nilai validasi untuk menggunakan CE pada masing-masing jumlah cluster 3, 4, dan 5 yaitu 0.4857, 0.5696, dan 0.6162.

Dari hasil pengujian validasi jumlah *cluster* dapat dilihat bahwa jumlah *cluster* 3 mempunyai tingkat validitas yang lebih baik dibandingkan dengan kelompok lainnya. Hal ini dikarenakan nilai XB yang terendah mengindikasikan partisi kelompok yang lebih baik, nilai PC yang paling besar menunjukkan jumlah kelompok yang optimal, dan indeks CE yang semakin kecil menandakan bahwa jumlah kelompok yang diberikan lebih baik.

Berdasarkan kriteria masing-masing indeks validasi yaitu nilai *XB* yang rendah, nilai PC yang paling besar, dan indeks CE yang terkecil jumlah *cluster* 3 menunjukkan hasil yang lebih baik dari jumlah kelompok 4 dan 5. Oleh karena itu dalam penelitian ini jumlah *cluster* yang akan digunakan adalah 3 *cluster*.

Setelah didapatkan jumlah cluster yang valid, yaitu 3 *cluster* langkah selanjutnya yaitu dilakukan pengklusteran terhadap data set. Pengujian data set sebanyak 44 data dengan jumlah *cluster* 3 diperoleh pembentukan *cluster* yang ditunjukkan pada Gambar 4.8.



Gambar 4.8 *Cluster* dengan FCM

Pada *cluster* 1 terdiri dari 12 titik yaitu P Buru, P Ambelau, P. Manipa, P. Kelang, P. Boano, P. Ambon, P. Haruku, P. Saparua, Nusa Laut, Amahai, P. Banda, P. Run. *Cluster* 2 dengan 12 titik yang terdiri dari P. Rozengain, P.

Panjang, P. Gorong, P. Kasiui, P. Wotab, P. Kur, P. Manggu, P. Taam, P. Tajondu, Tual, Kai Tanimbar, P. Kai Besar, P. Wolu, P. Wuliaru, P. Selu, P. Jamdena, P. Ceram Laut. *Cluster* 3 terdiri dari 15 Adaut, P. Serua, P. Nila, Lewa, P. Babar, Masela, P. Teun, P. Damar, Regola, Sera, P. Patti, Serwaru, P. Romang, Kisar, P. Wetar

Berdasarkan hasil *cluster* yang terbentuk akan dilakukan pembentukan jaringan pergerakan logistik menggunakan *minimum spanning tree* berbasis algoritma genetika. Pembentukan jaringan ini akan dilakukan dengan masing-masing *cluster*.

4.5.1 Pengujian Proses *Minimum Spanning Tree* berbasis *Hybrid Algorithm*

Proses *clustering* merupakan tahap awal dalam pembentukan jaringan pergerakan logistik di Kepulauan Maluku. Jaringan pergerakan ini akan direpresentasikan dimasing-masing *cluster* dengan menggunakan *minimum spanning tree* berbasis algoritma genetika. Penyelesaian *minimum spanning tree* diselesaikan dengan menggunakan algoritma genetika.

4.5.2.1 Pengujian *Cluster* 1

Pengujian *cluster* pertama, terdiri dari 12 node dengan uji coba yang dilakukan sebanyak 9 kali percobaan. Data set berupa node yang akan diujikan disajikan pada Tabel 4.4 berikut.

Tabel 4.4 Data set *cluster* 1

Node	Nama Pulau	Node	Nama Pulau
1	P Buru	7	P. Haruku
2	P Ambelau	8	P. Saparua
3	P. Manipa	9	Nusa Laut
4	P. Kelang	10	Amahai
5	P. Boano	11	P. Banda
6	P. Ambon	12	P. Run

Pengujian yang akan dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Adapun parameter yang digunakan untuk melakukan pengujian tersebut sebagaimana ditunjukkan pada Tabel 4.5.

Tabel 4.5 Data Parameter Pengujian *Cluster* 1

Nama Pengujian	Generasi Maksimum	Pop Size
A	500	50
B	500	75
C	500	100
D	1000	50
E	1000	75
F	1000	100
G	2000	50
H	2000	75
I	2000	100

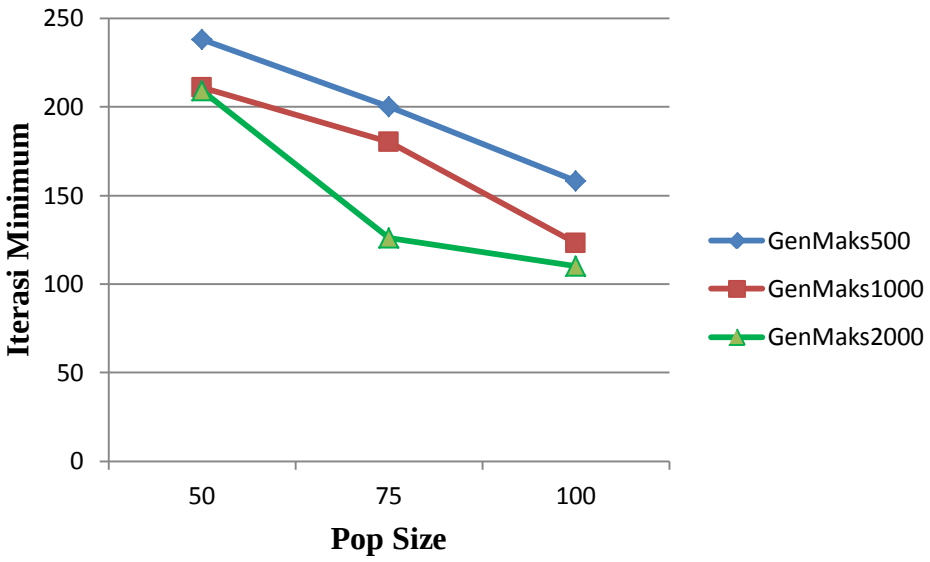
Tabel 4.5 menunjukkan bahwa pada pengujian A akan dilakukan dengan parameter pada ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500 maka algoritma genetika akan berhenti. Demikian juga pada pengujian B,C,D,E,F,G,H, dan I yang akan berhenti ketika mencapai generasi maksimumnya. Pada 9 pengujian ini, diperoleh hasil yang ditunjukkan pada Tabel 4.6.

Table 4.6 Pengujian Cluster 1

Pengujian	Iterasi	Waktu
A	238	15.778621
B	200	18.937184
C	158	23.637111
D	211	31.41883
E	180	34.103827
F	123	46.865622

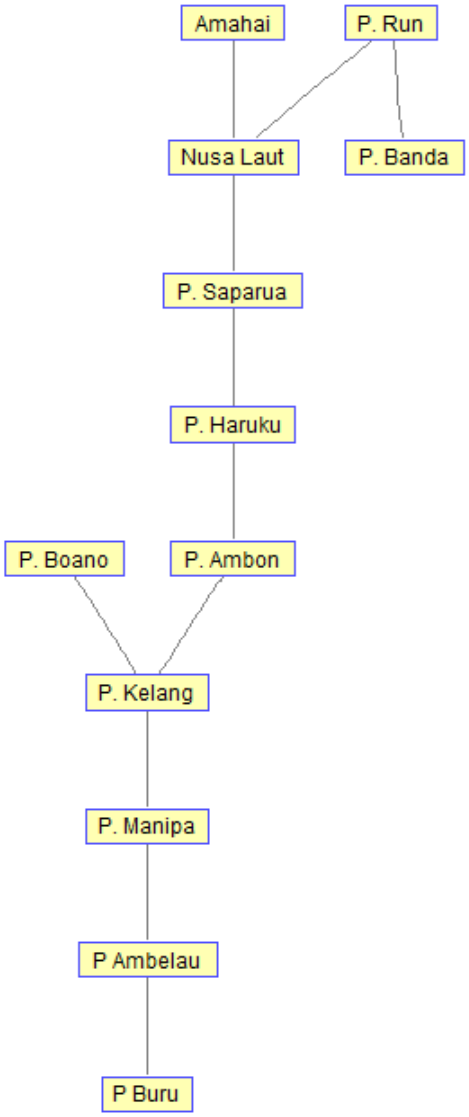
Pengujian	Iterasi	Waktu
G	209	62.878905
H	126	74.519633
I	110	93.672811

Berdasarkan hasil uji coba 9 pengujian untuk setiap generasi maksimum dengan masing-masing ukuran populasi didapatkan solusi optimal yang sama yaitu 9.41069801. Hal ini menunjukkan bahwa hasil pengujian ini konvergen pada solusi optimal yang sama. Dalam mencapai solusi optimalnya, algoritma genetika telah konvergen pada iterasi minimumnya. Dari Tabel 4.6 dapat dilihat pada pengujian A dengan generasi maksimum 500 dan ukuran populasi sebesar 50 dihasilkan waktu komputasi 15.778621 detik dan pada iterasi ke-238 telah menunjukkan konvergensi kesolusi optimal yaitu 9.41069801. Pengujian B menunjukkan konvergensi pada iterasi ke-200 dengan waktu komputasi 18.937184. Dengan melakukan hal yang sama pada pengujian C, D, E, F, G, H, dan I masing-masing konvergen pada iterasi ke-158,211,180,123,209,126, dan 110. Sedangkan waktu komputasinya masing-masing adalah 23.637111, 31.41883, 34.103827, 46.865622, 62.878905, 74.519633, dan 93.672811detik.



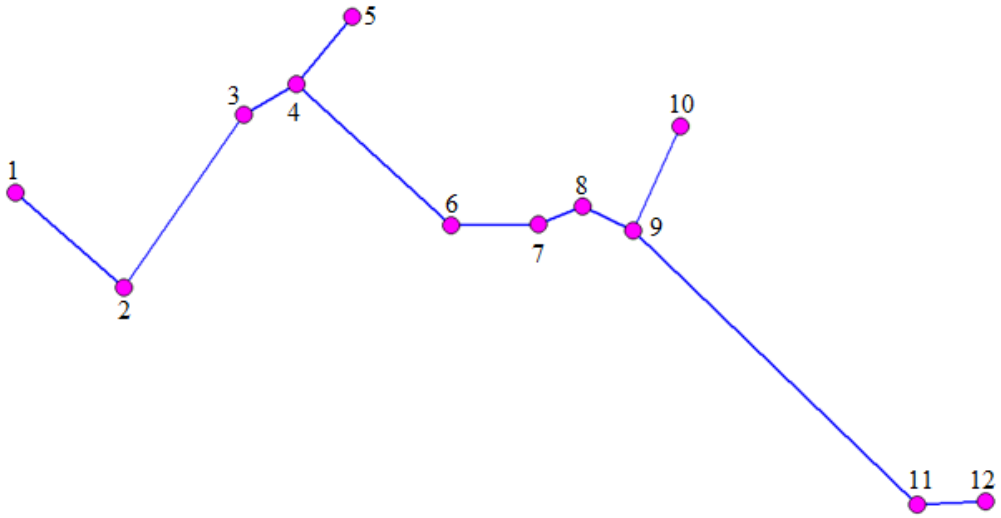
Gambar 4.9 Grafik Perbandingan antara Pengujian *Cluster* 1

Berdasarkan Gambar 4.9 terlihat bahwa pada penambahan ukuran populasi pada setiap pengujian mempengaruhi iterasi minimalnya. Semakin besar ukuran populasi yang diberikan iterasi minimum yang konvergen kesolusi optimal semakin kecil. Dari hasil pengujian *Cluster* 1 iterasi minimum yang paling kecil yang akan dipilih sebagai solusi paling baik, dan pengujian yang dipilih yaitu pada pengujian I. oleh karena itu, pengujian I akan digunakan dalam pembentukan jalur logistik di *cluster* 1. Pada pengujian I dengan ukuran populasi 100 dan generasi maksimum 2000 dengan 12 node diperoleh jalur {P.Buru-P. Ambelau}, {Ambelau - P. Manipa}, {P. Manipa – P. Kelang}, {P. Kelang – P. Boano}, {P. Kelang - P. Ambon}, {Ambon -P. Haruku}, {P. Haruku - P. Saparua}, P. Saparua - Nusa Laut}, {Nusa Laut – Amahai}, {Nusa Laut – P. Run}, {P. Run – P. Banda}. Dari jalur yang didapatkan, dimodelkan dalam bentuk graf yang ditunjukkan pada Gambar 4.10.



Gambar 4. 10 Model *Minimum Spanning Tree Cluster* 1

Dari model graf pada Gambar 4.10 dapat di implemtasikan kedalam studi kasus yang ditunjukkan pada Gambar 4.11.



Gambar 4.11 Implementasi *Minimum Spanning Tree* pada *Cluster* 1

4.5.2.2 Pengujian *Cluster* 2

Pengujian *cluster* 2, terdiri dari 17 node dengan uji coba yang dilakukan sebanyak 9 kali percobaan. Data set berupa node yang akan diujikan disajikan pada Tabel 4.7 berikut.

Tabel 4.7 Data set *cluster* 2

Node	Nama Pulau	Node	Nama Pulau
1	P. Rozengain	10	Tual
2	P. Panjang	11	Kai Tanimbar
3	P. Gorong	12	P. Kai Besar
4	P. Kasiui	13	P. Wolu
5	P. Wotab	14	P. Wuliaru
6	P. Kur	15	P. Selu
7	P. Manggu	16	P. Jamdena
8	P. Taam	17	P. Ceram Laut
9	P. Tajondu		

Pengujian yang akan dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Adapun parameter yang digunakan untuk melakukan pengujian tersebut sebagaimana ditunjukkan pada Tabel 4.8.

Tabel 4.8 Data Parameter Pengujian *Cluster* 2

Nama Pengujian	Generasi Maksimum	Pop Size
A	500	50
B	500	75
C	500	100
D	1000	50
E	1000	75
F	1000	100
G	2000	50
H	2000	75
I	2000	100

Tabel 4.8 menunjukkan bahwa pada pengujian A akan dilakukan dengan parameter pada ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500 maka algoritma genetika akan berhenti. Demikian juga pada pengujian B,C,D,E,F,G,H, dan I. Pada 9 pengujian ini, diperoleh hasil yang ditunjukkan pada Tabel 4.9.

Table 4.9 Pengujian Cluster 2

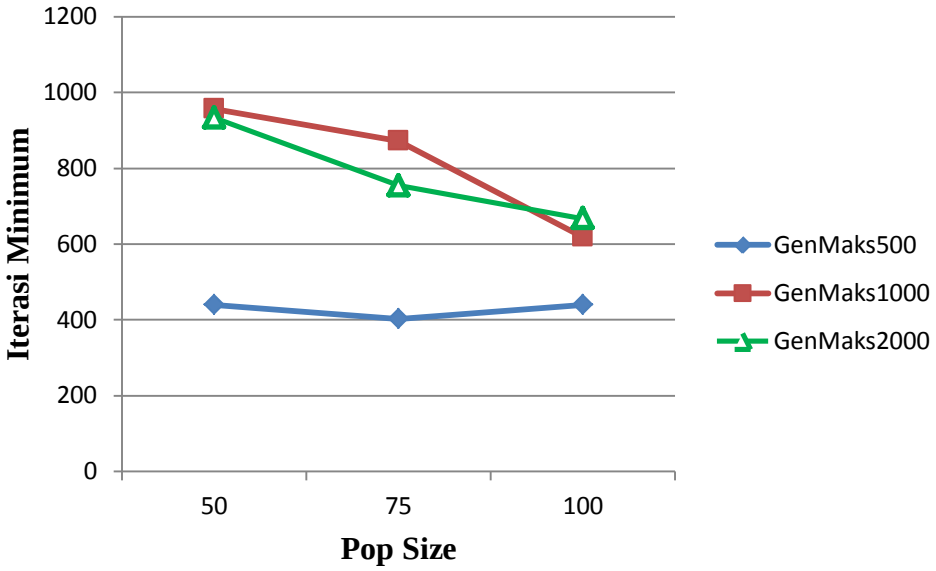
Pengujian	Iterasi	Waktu
A	439	23.759132
B	403	23.718635
C	440	23.913201
D	957	37.247211
E	872	47.252283
F	619	48.686291
G	933	81.994267
H	754	94.508243
I	667	97.367564

Berdasarkan Tabel 4.9 dapat dilihat bahwa pada pengujian A dengan generasi maksimum 500 dan ukuran populasi sebesar 50 dihasilkan waktu komputasi 23.759132 detik pada iterasi ke-439. Pada pengujian B dengan generasi maksimum 500 dan ukuran populasi sebesar 75 dihasilkan waktu komputasi 23.718635 detik pada iterasi ke-403. Pada pengujian C dengan generasi maksimum 500 dan ukuran populasi sebesar 100 dihasilkan waktu komputasi 23.913201 detik pada iterasi ke-440.

Pada pengujian D,E, dan F dengan generasi maksimun 1000 diperoleh hasil untuk pengujian D dengan ukuran populasi sebesar 50 dihasilkan waktu komputasi 37.247211 detik pada iterasi ke-957, pengujian E untuk ukuran populasi sebesar 75 dihasilkan waktu komputasi 47.252283 detik pada iterasi ke-872, dan pengujian F dengan ukuran populasi sebesar 100 dihasilkan waktu komputasi 48.686291 detik pada iterasi ke-619.

Sedangkan pengujian G dengan generasi maksimum 2000 dan ukuran populasi sebesar 50 dihasilkan waktu komputasi 81.994267 detik pada iterasi ke-933. Pada pengujian H dengan generasi maksimum 2000 dan ukuran populasi sebesar 75 dihasilkan waktu komputasi 94.508243 detik pada iterasi ke-754. Pada pengujian I dengan generasi maksimum 2000 dan ukuran populasi sebesar 100 dihasilkan waktu komputasi 97.367564 detik pada iterasi ke-667.

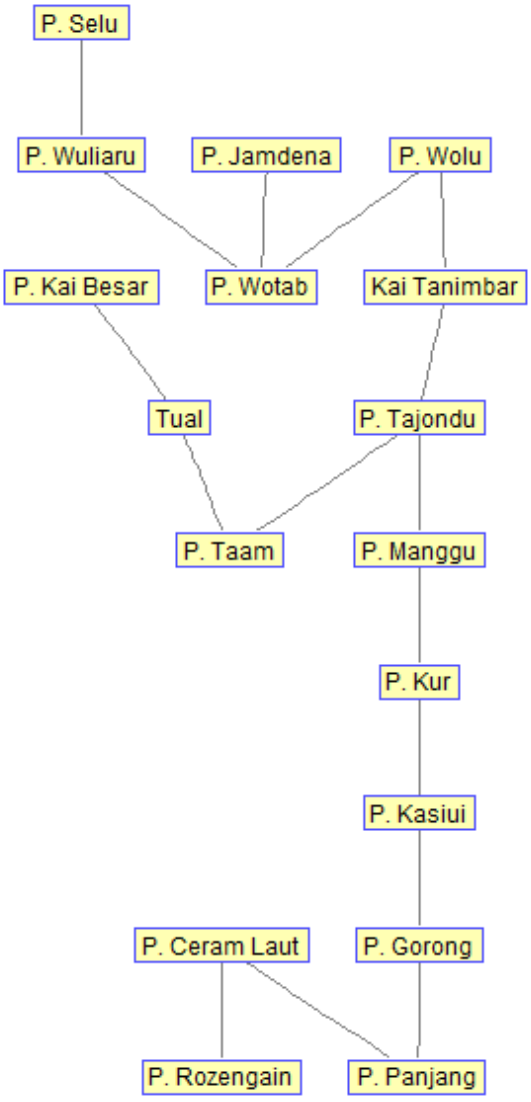
Dari 9 pengujian tersebut diperoleh hasil bahwa setiap generasi maksimum dengan masing-masing ukuran populasi didapatkan solusi optimal yang sama yaitu 14.96806833. Hal ini menunjukkan bahwa hasil pengujian ini konvergen pada solusi optimal yang sama. Untuk melihat perbandingan terhadap pengujian ini, dapat ditunjukkan oleh Gambar 4.12.



Gambar 4. 12 Grafik Perbandingan antara Pengujian *Cluster* 2

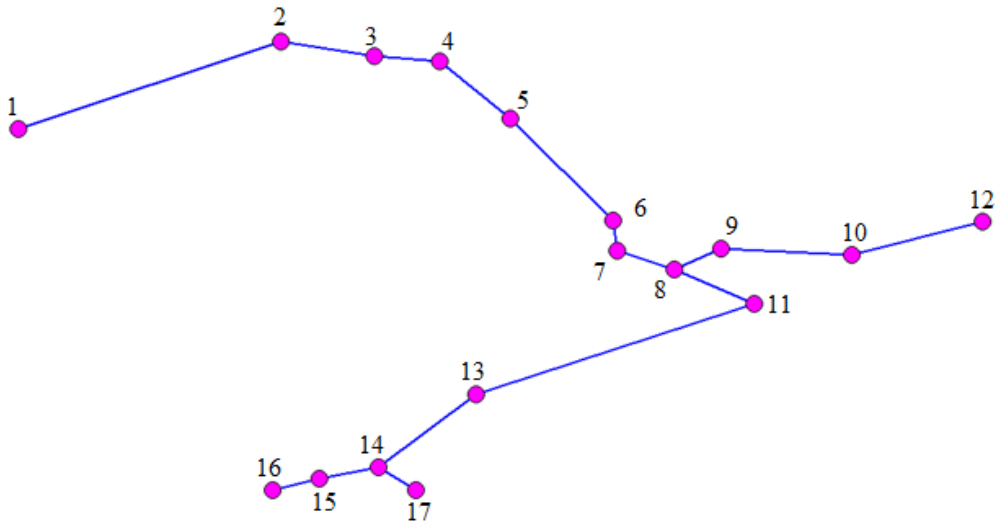
Gambar 4.12 menunjukkan untuk pengujian A, B, dan C dengan generasi maksimumnya adalah 500, pada pengujian B mengalami penurunan pada iterasi mimimumnya, setelah ukuran populasinya ditambah pada pengujian C nilai iterasi minimumnya menjadi lebih besar. Sedangkan untuk pengujian lainnya semakin besar generasi maksimum dan semakin banyak ukuran populasi yang diberikan maka akan semakin kecil iterasi. Oleh karena itu, pengujian yang akan dipilih adalah pengujian dengan maksimum generasi 1000 atau 2000. Dengan demikian akan dipilih berdasarkan iterasi minimum yang paling kecil yaitu pada pengujian F yang akan digunakan dalam pembentukan jalur logistik di *cluster* 2.

Pada pengujian F dengan ukuran populasi 100 dan generasi maksimum 1000 dengan 17 node diperoleh jalur. {P. Selu – P. Wuliaru}, {P. Wuliaru – P. Wotab}, {{P. Wotab – P. Jamdena}, {P. Wotab – P. Wolu}, {P. Wolu – Kai Tanimbar}, {Kali Tanimbar – P. Tajondu}, {P. Tajondu – P. Taam}, {P. Taam – Tual}, {Tual – P. Kai Tanimbar}, {P. Tajondu – P. Manggu}, {P. Manggu – P. Kur}, {P. Kur – P. Kasiuri}, {P. Kasiuri – P. Gorong}, {P. Gorong – P. Panjang}, {P. Panjang – P. Ceram Laut}, dan {P. Ceram Laut – P. Rozengain}. Dari jalur yang didapatkan, dimodelkan dalam bentuk graf yang ditunjukkan pada Gambar 4.12.



Gambar 4. 13 Model *Minimum Spanning Tree Cluster* 2

Dari model graf pada Gambar 4.13 dapat di implemtasikan kedalam studi kasus yang ditunjukkan pada Gambar 4.14.



Gambar 4.14 Implementasi *Minimum Spanning Tree* pada *Cluster* 2

4.5.2.3 Pengujian Cluster 3

Pengujian *cluster* 3, terdiri dari 15 node dan uji coba dilakukan sebanyak 9 kali percobaan. Data set berupa node yang akan diujikan disajikan pada Tabel 4.10 berikut.

Tabel 4.10 Data set *cluster* 3

Node	Nama Pulau	Node	Nama Pulau
1	Adaut	9	Regola
2	P. Serua	10	Sera
3	P. Nila	11	P. Patti
4	Lewa	12	Serwaru
5	P. Babar	13	P. Romang
6	Masela	14	Kisar
7	P. Teun	15	P. Wetar
8	P. Damar		

Pengujian ini dilakukan untuk melihat perbandingan hasil dari tiap pengujian. Pengujian tersebut dapat ditunjukkan pada Tabel 4.11.

Tabel 4.11 Data Parameter Pengujian *Cluster* 3

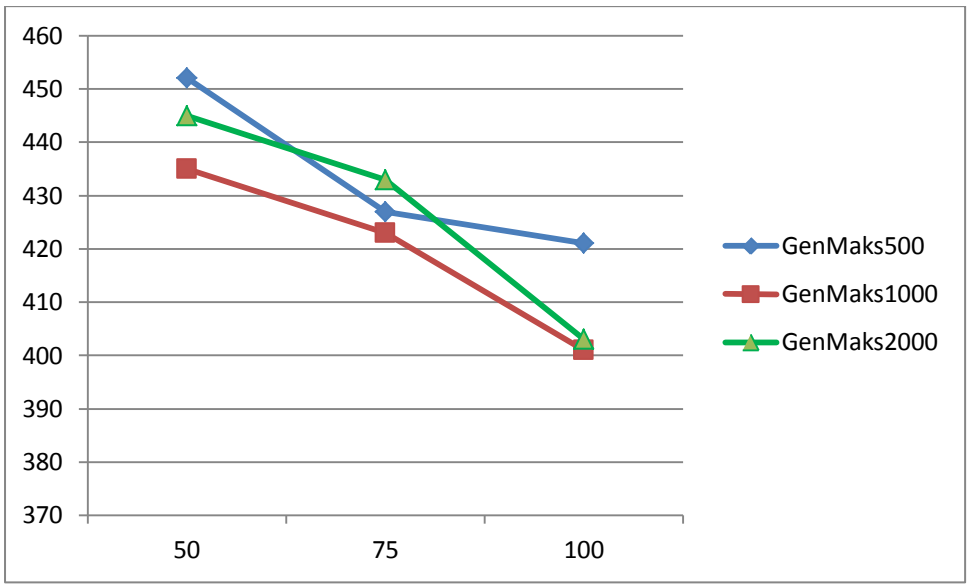
Nama Pengujian	Generasi Maksimum	Pop Size
A	500	50
B	500	75
C	500	100
D	1000	50
E	1000	75
F	1000	100
G	2000	50
H	2000	75
I	2000	100

Tabel 4.11 menunjukkan bahwa pada pengujian A akan dilakukan dengan parameter pada ukuran populasi sebesar 50 dan generasi maksimum sebesar 500. Ketika generasi mencapai 500 maka algoritma genetika akan berhenti. Demikian juga pada pengujian lainnya, ketika mencapai generasi maksimum maka iterasinya akan berhenti. Pada 9 pengujian ini, diperoleh hasil yang ditunjukkan pada Tabel 4.12.

Table 4.12 Pengujian Cluster 3

Pengujian	Iterasi	Waktu
A	452	16.624348
B	427	23.489966
C	421	23.757217
D	435	32.873568
E	423	46.962563
F	401	48.0905
G	445	64.704674
H	433	93.474252
I	403	93.937217

Dari hasil pengujian menunjukkan bahwa setiap generasi maksimum dengan masing-masing ukuran populasi didapatkan solusi optimal yang sama yaitu 17.46042583. Hal ini menunjukkan bahwa hasil pengujian ini konvergen pada solusi optimal yang sama. Berdasarkan Tabel 4.9 pada pengujian A dengan generasi maksimum 500 dan ukuran populasi sebesar 50 dihasilkan waktu komputasi 16.624348 detik pada iterasi ke-452, dan iterasi minimal tersebut menunjukkan konvergensi kesolusi optimal yaitu 17.46042583. Begitu pula dengan pengujian yang lainnya setelah mencapai iterasi maksimum maka akan konvergen ke solusi optimal. Untuk melihat perbandingan terhadap pengujian ini, dapat ditunjukkan oleh Gambar 4.15.

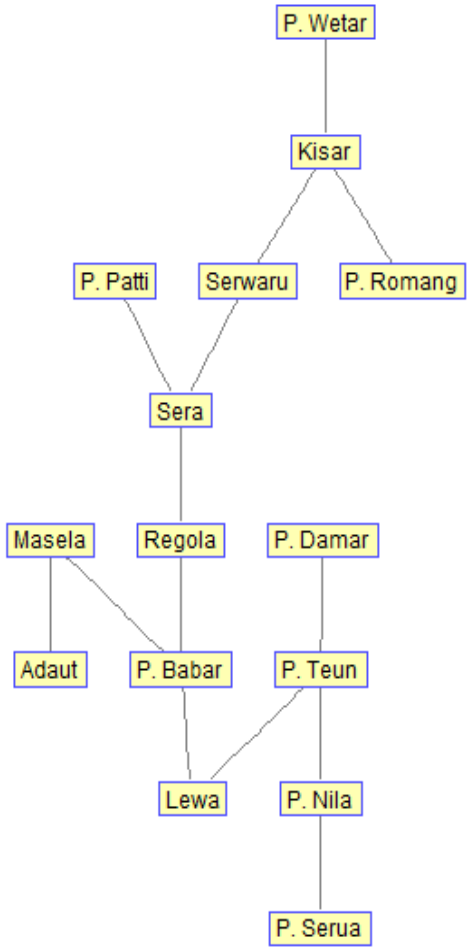


Gambar 4.15 Grafik Perbandingan antara Pengujian *Cluster* 3

Berdasarkan Gambar 4.15 terlihat bahwa untuk membentuk MST dengan *hybrid* GA ketika ukuran populasi bertambah iterasi minimum yang menuju solusi optimal semakin kecil. Sehingga semakin besar generasi maksimum dan semakin banyak populasi yang diberikan maka akan semakin kecil iterasi minimumnya. Dengan demikian iterasi minimum yang paling kecil yaitu pada pengujian F yang akan digunakan dalam pembentukan jalur logistik di *cluster* 3.

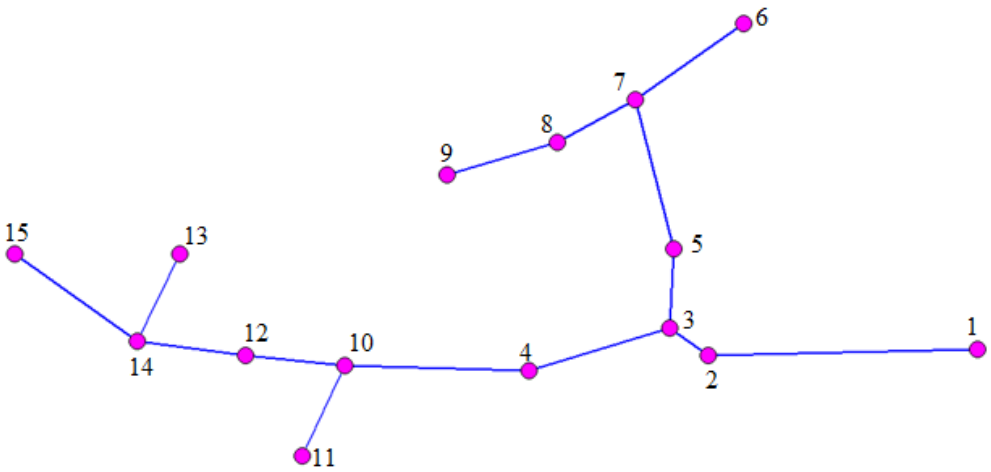
Pada pengujian F dengan ukuran populasi 100 dan generasi maksimum 1000 dengan 15 node diperoleh jalur {P.Wetar – Kisar}, {Kisar – P. Romang},

{Kisar – Serwaru}, {Serwaru – Sera}, {Sera – P. Patti}, {Sera – Regola}, {Regola – P. Babar}, {P. Babar – Masela}, {Masela – Adaut}, {P. Babar – Lewa}, {Lewa – P. Teun}, {P. Teun – P. Damar}, {P. Teun – P. Nila}, {P. Nila – P. Serua}. Dari jalur yang didapatkan, dimodelkan dalam bentuk graf yang ditunjukkan pada Gambar 4.15.



Gambar 4.16 Model *Minimum Spanning Tree Cluster* 3

Dari model graf pada Gambar 4.16 dapat di implemtasikan kedalam studi kasus yang ditunjukkan pada Gambar 4.17.

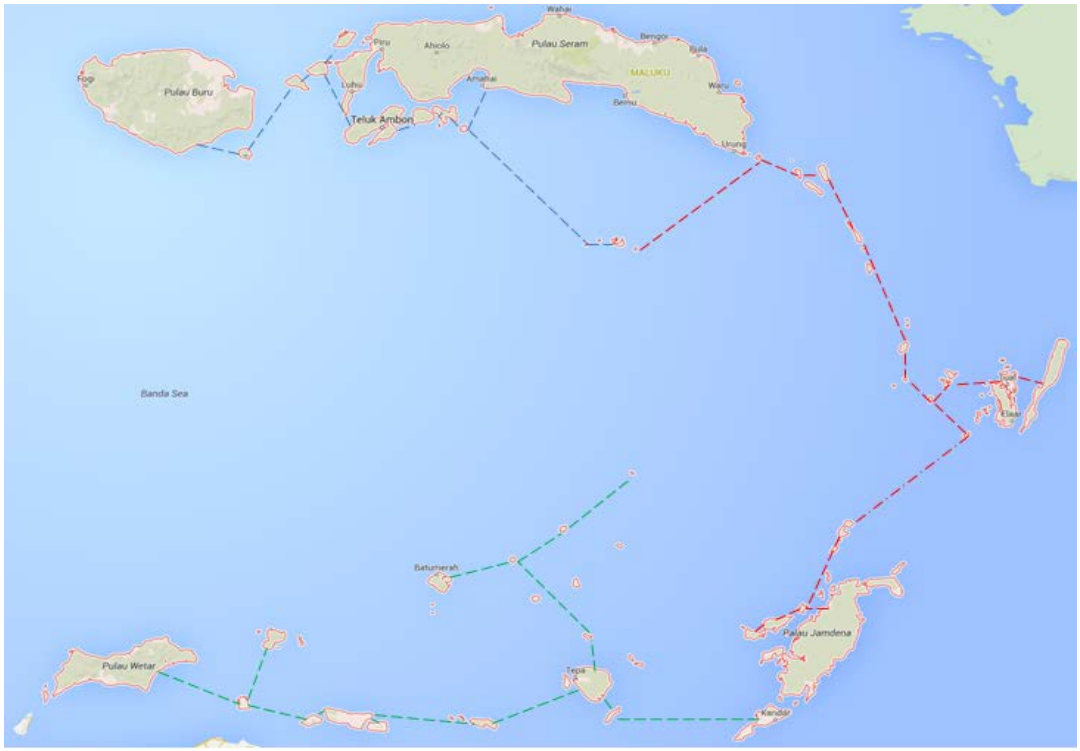


Gambar 4.17 Implementasi *Minimum Spanning Tree* pada *Cluster* 3

Dengan menggunakan metode *hybrid* GA yang telah disesuaikan, diperoleh graf studi kasus seperti pada Gambar 4.18.

Gambar. 4.18 Graf Studi Kasus

Dari graf tersebut, rute studi kasus pada tesis ini dapat diilustrasikan oleh Gambar 4.19 berikut.



Gambar 4.19 Rute Studi Kasus

Dengan demikian, didapatkan tiga rute dari empat puluh empat node atau pulau yang diusulkan oleh hybrid GA dengan rute tiap cluster sebagai berikut:

- *Cluster* 1 memiliki rute {P.Buru-P. Ambelau}, {Ambelau - P. Manipa}, {P. Manipa – P. Kelang}, {P. Kelang – P. Boano}, {P. Kelang - P. Ambon}, {Ambon -P. Haruku}, {P. Haruku - P. Saparua}, P. Saparua - Nusa Laut}, {Nusa Laut – Amahai}, {Nusa Laut – P. Run}, {P. Run – P. Banda}, dimana pusat *cluster* terletak pada pulau Ambon.
- *Cluster* 2 memiliki rute {P. Selu – P. Wuliaru}, {P. Wuliaru – P. Wotab}, {{P. Wotab – P. Jamdena}, {P. Wotab – P. Wolu}, {P. Wolu – Kai Tanimbar}, {Kali Tanimbar – P. Tajondu}, {P. Tajondu – P. Taam}, {P. Taam – Tual}, {Tual – P. Kai Tanimbar}, {P. Tajondu – P. Manggu}, {P. Manggu – P. Kur}, {P. Kur – P. Kasiuri}, {P. Kasiuri – P. Gorong}, {P. Gorong – P. Panjang}, {P.

Panjang – P. Ceram Laut}, dan {P. Ceram Laut – P. Rozengain}, dengan pusat *cluster* pada pulau Taam.

- *Cluster* 3 memiliki rute {P.Wetar – Kisar}, {Kisar – P. Romang}, {Kisar – Serwaru}, {Serwaru – Sera}, {Sera – P. Patti}, {Sera – Regola}, {Regola – P. Babar}, {P. Babar – Masela}, {Masela – Adaut}, {P. Babar – Lewa}, {Lewa – P. Teun}, {P. Teun – P. Damar}, {P. Teun – P. Nila}, {P. Nila – P. Serua} yang pusat *cluster*nya terletak di pulau Regola.

LAMPIRAN 1

Source Code hybrid Genetic Algorithm

```
==========================================================

clc;
clear;
close all;
-----------------------------------------------------------------

%% Problem Definition

model=CreateModel();

CostFunction=@(xhat) MyCost(xhat,model);     % Cost Function

nVar=model.n*(model.n-1)/2;     % Number of Decision Variables

VarSize=[1 nVar];  % Decision Variables Matrix Size

VarMin=0;         % Lower Bound of Variables
VarMax=1;         % Upper Bound of Variables
-----------------------------------------------------------------

%% GA Parameters

MaxIt=500;      % Maximum Number of Iterations (500,1000,2000)

nPop=50;        % Population Size (50,75,100)

pc=0.8;                 % Crossover Percentage
nc=2*round(pc*nPop/2);  % Number of Offsprings (Parnets)

pm=0.3;                 % Mutation Percentage
nm=round(pm*nPop);      % Number of Mutants

gamma=0.2;      %Crossover Rate

mu=0.4;         % Mutation Rate

beta=8;         % Selection Pressure
-----------------------------------------------------------

%% Initialization

empty_individual.Position=[];
empty_individual.Cost=[];
empty_individual.Sol=[];

pop=repmat(empty_individual,nPop,1);

for i=1:nPop
```

```
    % Initialize Position
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);

    % Evaluation
    [pop(i).Cost, pop(i).Sol]=CostFunction(pop(i).Position);

end

% Sort Population
  Costs=[pop.Cost];
  [Costs, SortOrder]=sort(Costs);
  pop=pop(SortOrder);

% Store Best Solution
  BestSol=pop(1);

% Array to Hold Best Cost Values
  BestCost=zeros(MaxIt,1);

% Store Cost
  WorstCost=pop(end).Cost;
---------------------------------------------------------------------

%% GA Main Loop

tic
for it=1:MaxIt

    P=exp(-beta*Costs/WorstCost);
    P=P/sum(P);

    % Crossover
     popc=repmat(empty_individual,nc/2,2);
        for k=1:nc/2

        % Select Parents Indices
        i1=RouletteWheelSelection(P);
        i2=RouletteWheelSelection(P);

        % Select Parents
        p1=pop(i1);
        p2=pop(i2);

        % Apply Crossover
        [popc(k,1).Position, popc(k,2).Position]=...

Crossover(p1.Position,p2.Position,gamma,VarMin,VarMax);

        % Evaluate Offsprings
          [popc(k,1).Cost,
popc(k,1).Sol]=CostFunction(popc(k,1).Position);
          [popc(k,2).Cost,
popc(k,2).Sol]=CostFunction(popc(k,2).Position);

        end
```

```
    popc=popc(:);

  % Mutation
    popm=repmat(empty_individual,nm,1);
    for k=1:nm

        % Select Parent
        i=randi([1 nPop]);
        NewIndividual=pop(i);

        % Apply Mutation
        popm(k).Position=Mutate(NewIndividual.Position,...
            mu,VarMin,VarMax);

        % Evaluate Mutant
        [popm(k).Cost,popm(k).Sol]=...
        CostFunction(popm(k).Position);

        if popm(k).Cost<=NewIndividual.Cost || rand < 0.1
            NewIndividual=popm(k);
        end
    end

% Create Merged Population
    pop=[pop
        popc
        popm]; %#ok

 % Sort Population
 Costs=[pop.Cost];
 [Costs, SortOrder]=sort(Costs);
 pop=pop(SortOrder);

 % Update Worst Cost
 WorstCost=max(WorstCost,pop(end).Cost);

 % Local Search based on Mutation
 for k=1:nm/2
     NewIndividual=BestSol;
     NewIndividual.Position=Mutate(BestSol.Position,...
         mu, VarMin,VarMax);
     [NewIndividual.Cost, NewIndividual.Sol]=...
         CostFunction(NewIndividual.Position);
     if NewIndividual.Cost<=BestSol.Cost
         BestSol=NewIndividual;
     end
 end

 % Truncation
 pop=pop(1:nPop);
 Costs=Costs(1:nPop);

 % Store Best Solution Ever Found
 BestSol=pop(1);
 % Store Best Cost Ever Found
 BestCost(it)=BestSol.Cost;
```

```
    % Show Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = '...
        num2str(BestCost(it))]);

    % Plot Solution
    figure(1);
    PlotSolution(BestSol.Sol,model);
    pause(0.01);

end
toc
-------------------------------------------------------------

%% Results

figure;
plot(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;


=============================================================
```

LAMPIRAN 2

Jarak antar node (pulau) pada *cluster* 1

Dari\Ke	1	2	3	4	5	6	7	8	9	10	11	12
1	0	50.9694	84.8674	105.836	133.78	153.153	183.408	198.572	216.51	233.874	357.227	335.017
2	50.9694	0	74.6608	94.5985	125.577	116.708	146.708	163.041	179.161	202.902	311.175	288.437
3	84.8674	74.6608	0	21.3809	51.6725	82.6232	110.06	122.727	142.034	152.549	294.133	273.906
4	105.836	94.5985	21.3809	0	30.979	74.4086	98.529	109.302	128.9	135.186	283.994	264.679
5	133.78	125.577	51.6725	30.979	0	82.4437	98.7518	105.211	124.355	121.223	281.623	264.016
6	153.153	116.708	82.6232	74.4086	82.4437	0	30.3644	46.3563	63.4144	87.4791	211.515	191.358
7	183.408	146.708	110.06	98.529	98.7518	30.3644	0	16.792	33.102	60.6937	185.494	166.541
8	198.572	163.041	122.727	109.302	105.211	46.3563	16.792	0	19.6168	44.4195	176.512	158.859
9	216.51	179.161	142.034	128.9	124.355	63.4144	33.102	19.6168	0	40.6442	157.269	140.113
10	233.874	202.902	152.549	135.186	121.223	87.4791	60.6937	44.4195	40.6442	0	171.795	159.013
11	357.227	311.175	294.133	283.994	281.623	211.515	185.494	176.512	157.269	171.795	0	23.866
12	335.017	288.437	273.906	264.679	264.016	191.358	166.541	158.859	140.113	159.013	23.866	0

Keterangan :

1	P. Buru	5	P. Boano	9	Nusa Laut
2	P Ambelau	6	P. Ambon	10	Amahai
3	P. Manipa	7	P. Haruku	11	P. Banda
4	P. Kelang	8	P. Saparua	12	P. Run

“Halaman ini sengaja dikosongkan”

LAMPIRAN 3

Jarak antar node (pulau) pada *cluster* 2

Dari\Ke	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	142.4316	161.5241	174.9378	332.3419	226.7934	239.4334	272.0308	265.4865	317.3645	305.8949	352.2921	290.8052	335.0294	339.6569	356.6957	122.2607
2	142.4316	0	23.5745	74.65442	372.4522	171.018	195.9891	213.1517	220.6696	247.5508	262.153	262.5489	309.0431	383.7597	394.8506	393.5654	35.54391
3	161.5241	23.5745	0	58.0897	368.5281	156.6178	182.7019	196.7376	206.3151	228.8022	246.9643	241.1275	302.6559	381.1065	393.1382	388.8235	58.8719
4	174.9378	74.65442	58.0897	0	319.0157	98.61981	125.012	138.9425	148.2703	173.0305	188.8909	191.3795	250.3512	333.367	346.7711	338.0726	107.3809
5	332.3419	372.4522	368.5281	319.0157	0	238.8753	213.8339	232.8276	207.6075	255.2016	198.9943	309.1925	74.04412	23.72164	43.08678	24.68891	387.2084
6	226.7934	171.018	156.6178	98.61981	238.8753	0	27.55004	46.07978	49.73875	90.5782	91.25472	131.0508	165.7188	256.8654	272.9286	254.5574	200.0748
7	239.4334	195.9891	182.7019	125.012	213.8339	27.55004	0	37.14363	26.65594	83.55075	68.95205	132.3672	140.1343	232.6069	249.2295	228.6484	223.6331
8	272.0308	213.1517	196.7376	138.9425	232.8276	46.07978	37.14363	0	25.60389	46.79934	52.16389	95.7618	159.0135	253.3802	271.0868	244.6836	243.7724
9	265.4865	220.6696	206.3151	148.2703	207.6075	49.73875	26.65594	25.60389	0	64.42426	42.43854	117.788	133.68	227.9638	245.5842	219.9979	249.1834
10	317.3645	247.5508	228.8022	173.0305	255.2016	90.5782	83.55075	46.79934	64.42426	0	56.3213	55.57776	183.9771	277.3795	296.0446	263.2375	280.1814
11	305.8949	262.153	246.9643	188.8909	198.9943	91.25472	68.95205	52.16389	42.43854	56.3213	0	110.4671	128.5592	221.3303	240.1141	206.9575	291.2595
12	352.2921	262.5489	241.1275	191.3795	309.1925	131.0508	132.3672	95.7618	117.788	55.57776	110.4671	0	238.9336	331.7091	350.5513	315.9532	297.3029
13	290.8052	309.0431	302.6559	250.3512	74.04412	165.7188	140.1343	159.0135	133.68	183.9771	128.5592	238.9336	0	94.43947	112.5264	88.93245	327.5094
14	335.0294	383.7597	381.1065	333.367	23.72164	256.8654	232.6069	253.3802	227.9638	277.3795	221.3303	331.7091	94.43947	0	19.38714	35.78806	396.7034
15	339.6569	394.8506	393.1382	346.7711	43.08678	272.9286	249.2295	271.0868	245.5842	296.0446	240.1141	350.5513	112.5264	19.38714	0	51.00869	406.4141
16	356.6957	393.5654	388.8235	338.0726	24.68891	254.5574	228.6484	244.6836	219.9979	263.2375	206.9575	315.9532	88.93245	35.78806	51.00869	0	409.3017
17	122.2607	35.54391	58.8719	107.3809	387.2084	200.0748	223.6331	243.7724	249.1834	280.1814	291.2595	297.3029	327.5094	396.7034	406.4141	409.3017	0

1 P. Rozengain
2 P. Panjang
3 P. Gorong
4 P. Kasiui
5 P. Wotab
6 P. Kur
7 P. Manggu
8 P. Taam
9 P. Tajondu
10 Tual
11 Kai Tanimbar
12 P. Kai Besar
13 P. Wolu
14 P. Wuliaru
15 P. Selu
16 P. Jamdena
17 P. Ceram Laut

“Halaman ini sengaja dikosongkan”

LAMPIRAN 4

Jarak antar node (pulau) pada *cluster* 3

Dari\Ke	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	235.805	235.745	170.124	160.627	139.848	253.422	296.04	233.223	329.233	357.072	380.638	418.799	436.598	503.707
2	235.805	0	73.14	143.748	192.458	206.327	121.552	180.076	241.975	296.476	352.461	330.593	325.455	371.213	404.348
3	235.745	73.14	0	94.088	142.635	162.945	48.6314	108.346	176.709	223.795	280.553	257.489	255.162	298.971	336.254
4	170.124	143.748	94.088	0	49.5642	68.9241	89.1161	125.92	106.531	185.867	231.855	232.27	256.446	284.381	342.113
5	160.627	192.458	142.635	49.5642	0	26.4107	129.02	149.471	77.4115	170.611	206.643	221.203	258.642	276.613	343.228
6	139.848	206.327	162.945	68.9241	26.4107	0	153.648	175.881	93.5335	189.387	220.129	240.818	281.923	296.896	365.861
7	253.422	121.552	48.6314	89.1161	129.02	153.648	0	60.4733	142.109	177.109	235.052	209.084	207.661	250.419	289.97
8	296.04	180.076	108.346	125.92	149.471	175.881	60.4733	0	128.561	129.797	189.707	153.528	147.188	191.159	229.872
9	233.223	241.975	176.709	106.531	77.4115	93.5335	142.109	128.561	0	96.2697	129.36	148.055	195.692	204.524	276.983
10	329.233	296.476	223.795	185.867	170.611	189.387	177.109	129.797	96.2697	0	59.9655	51.9232	110.291	108.559	184.671
11	357.072	352.461	280.553	231.855	206.643	220.129	235.052	189.707	129.36	59.9655	0	68.8606	140.757	111.263	194.87
12	380.638	330.593	257.489	232.27	221.203	240.818	209.084	153.528	148.055	51.9232	68.8606	0	71.9322	56.6532	135.227
13	418.799	325.455	255.162	256.446	258.642	281.923	207.661	147.188	195.692	110.291	140.757	71.9322	0	58.8201	85.6696
14	436.598	371.213	298.971	284.381	276.613	296.896	250.419	191.159	204.524	108.559	111.263	56.6532	58.8201	0	83.6116
15	503.707	404.348	336.254	342.113	343.228	365.861	289.97	229.872	276.983	184.671	194.87	135.227	85.6696	83.6116	0

Keterangan :

1	Adaut	5	P. Babar	8	P. Damar	12	Serwaru
2	P. Serua	6	Masela	9	Regola	13	P. Romang
3	P. Nila	7	P. Teun	10	Sera	14	Kisar
4	Lewa			11	P. Patti	15	P. Wetar

“Halaman ini sengaja dikosongkan”

BAB 5
PENUTUP

Pada bab ini akan diuraikan beberapa kesimpulan dari pembahasan dan analisis hasil yang telah dikerjakan pada Bab 4 serta saran untuk perbaikan pada penelitian selanjutnya.

5.1 Kesimpulan

Berdasarkan eksperimen dan pembahasan terhadap hasil pengujian yang telah dilakukan, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Proses *clustering* menggunakan Fuzzy C-Means diperoleh bahwa jumlah *cluster* sebanyak 3 *cluster* menunjukkan hasil yang lebih baik dibandingkan dengan jumlah *cluster* 4 dan 5. Dengan demikian, proses pembentukan *cluster* dengan FCM diperoleh 3 *cluster* untuk proses pembentukan pola jaringan logistik, dengan *cluster* 1 berpusat di Ambon, *cluster* 2 berpusat di Taam, dan *cluster* 3 berpusat di Regola.
2. Metode *Minimum Spanning Tree* berbasis *Hybrid Genetic Algorithm* menghasilkan pola jaringan logistik yang optimal sebanyak 3 pola jaringan dari keseluruhan data set. Pada pola jaringan pertama terdiri dari 12 pulau, pola jaringan logistik kedua terdiri dari 17 pulau, dan pola jaringan ketiga terdiri dari 15 pulau.

5.2 Saran

Studi kasus yang digunakan pada penelitian ini adalah studi kasus transportasi laut yang dibatasi untuk permasalahan *minimum spanning tree*. Sehingga, proses penelitian hanya memperhatikan faktor jarak saja tanpa memperhatikan realita yang ada. Oleh karena itu, diperlukan penelitian lebih lanjut mengenai metode yang dapat menyelesaikan studi kasus sesuai dengan kendala-kendala yang ada dan dapat diterapkan di lapangan.

Hasil penelitian menunjukkan bahwa metode hybrid GA dapat menghasilkan solusi yang optimal, didukung dengan penelitian-penelitian

sebelumnya. Akan tetapi belum ada bukti analitis yang mendukung penelitian, sehingga diharapkan terdapat penelitian lebih lanjut mengenai studi kasus ini terutama dalam meminimalkan biaya transportasi pada fungsi objektif yang diberikan pada persamaan (2.1).

DAFTAR PUSTAKA

[1] Bezdek, James C., Ehrlich, R., Full, W. (1984). "FCM: The Fuzzy C-Means Clustering Algorithm". Computers & Geosciences. Vol.10, No. 2-3, Hal. 191-203.

[2] BPS Maluku (2013). Maluku Dalam Angka. Laporan Tahunan BPS Maluku, Ambon.

[3] Diabat, A. dan Deskoores, R. (2015). "A hybrid genetic algorithm based heuristic for an integrated supply chain problem". Journal of Manufacturing Systems.

[4] El-Mihoub, T. A., Hopgood, A. A., Nolle, L., dan Battersby, A. (2006). "Hybrid Genetic Algorithms: A Review". Engineering Letters. 13:2

[5] Gen, M., Altiparmak, F., dan Lin, L. (2006). 'A genetic algorithm for two-stage transportation problem using priority-based encoding". OR Spectrum. Vol. 28, Hal. 337–354.

[6] Gurning, S. (2006). *Analisa Konsep Trans-Maluku Sebagai Pola Jaringan Transportasi Laut di Propinsi Maluku*. Jurusan Teknik Sistem Perkapalan. Fakultas Teknologi Kelautan ITS Surabaya.

[7] Han, J., & Kamber, M. (2000). "Data Mining Concept and Techniques Second Edition". United Stated: Morgan Kaufman.

[8] Izakian, H. dan Abraham, A. (2011). "Fuzzy C-means and Fuzzy Swarm for Fuzzy Clustering Problem". Expert Systems with Applications, Vol 38, Hal. 1835–1838.

[9] Jo, J. B., Li, Y. & Gen, M. (2007). "Nonlinear Fixed Charge Transportation Problem by Spanning Tree-based Genetic Algorithm". Science Direct Computer & Industrial Engineering, Vol. 53, Hal. 290-298.

[10] Kusumadewi, S., & Purnomo, H. (2010). *Aplikasi Logika Fuzzy Untuk Pendukung Keputusan*. Jakarta: Graha Ilmu.

[11] Lampiran Peraturan Presiden Republlik Indonesia. Tahun 2012. Cetak Biru Pengembangan Sistem Logistik Nasional. 5 Maret 2012.

[12] Leski, J. M. (2016). "Fuzzy c-ordered-means clustering" Fuzzy Sets and Systems Vol. 286, Hal. 114–133.

[13] Munir, R. (2012). *Matematika Diskrit*, Bandung: Penerbit Informatika.

[14] Prahastono, I., King, D.J., Ozveren, C.S. dan Bradley, D. (2008). "Electricity load profile classification using Fuzzy C-Means method". In: 43rd International Universities Power Engineering Conference, Padova. IEEE.

[15] Prakash, A., Chan, T.S., Liao, H., Deshmukh, S.G. (2012). "Network optimization in supply chain: A KBGA approach". Decision Support Systems, Vol. 52, Hal. 528–538

[16] Prasetya, E. (2012). *Data Mining Konsep dan Aplikasi Menggunakan MATLAB*. Yogyakarta: Penerbit Andi.

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

[18] Sivanandam, S.N. Deepa, S.N. (2008). "Practical Genetic Algorithms". New york : Springer Science+Businnes Media.

[19] Wang, W. dan Zhang, Y., "On fuzzy cluster validity indices, Fuzzy Sets System", Vol. 158, No. 19, pp.2095-2117, 2007.

[20] Zaverdhi, S.A., Kesthehi, M.H., dan Moghaddam, R.T. (2011). "Solving Capacitated Fixed-charge Transportation Problem by Artificial Immune and Genetic Algorthm with a Prufer Number Representation". Expert System with Aplication, Vol. 38, Hal. 10462-10474.

[21] Zhou, J., Chen, L., dan Wang, K. (2015). "Path Optimality Conditions for Minimum Spanning Tree Problem with Uncertain Edge Weights. International Journal of Uncertainty", Fuzziness and Knowledge-Based Systems. Vol. 23, No. 1, Hal. 49–71.

[22] Shahab, M.L., Utomo, B.U., dan Irawan, M.I. (2016). "Decomposing and Solving Capacitated Vihicle Routing Problem (CVRP) using Two-Step Genetic Algorithm (TGSA)" Journal of Theoretical and Applied Information Technology. Vol. 87, No.3, Hal. 461-468.

BIODATA PENULIS

Penulis bernama Shinta Tri Kismati yang lahir di Blora, 28 Pebruari 1988 dan merupakan anak kedua dari lima bersaudara. Penulis menempuh pendidikan formal dimulai dari TK ABBA II Tarakan (1993-1994), SD Negeri 007 Tarakan (1994-2000), SMP Negeri 2 Tarakan (2000-2003), SMK Negeri 1 Tarakan Jurusan Adm. Perkantoran (2003-2006). Pada tahun 2006, penulis melanjutkan studi ke jenjang S1 dan diterima sebagai mahasiswa di Jurusan Pendidikan Matematika, Fakultas Keguruan dan Ilmu Pendidikan (FKIP), Universitas Borneo Tarakan. Selanjutnya, pada tahun 2013 penulis mendapat beasiswa Pra S2-S2 Saintek dan menjadi mahasiswa Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA), Institut Teknologi Sepuluh Nopember Surabaya dan mengambil bidang minat Matematika Terapan. Untuk informasi yang berkaitan dengan Tesis ini, dapat menghubungi penulis melalui email kismanti88@gmail.com.