OPTIMASI HASIL *CLUSTERING* FUZZY C-MEANS MENGGUNAKAN ALGORITMA GENETIKA
(Studi Kasus : *Clustering* Data Bunga Iris)

SKRIPSI

oleh :
ISROFI FEBRIAWAN
0610960032-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011

OPTIMASI HASIL *CLUSTERING* FUZZY C-MEANS MENGGUNAKAN ALGORITMA GENETIKA
(Studi Kasus : *Clustering* Data Bunga Iris)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

oleh :
ISROFI FEBRIAWAN
0610960032-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011

LEMBAR PENGESAHAN SKRIPSI

OPTIMASI HASIL *CLUSTERING* FUZZY C-MEANS MENGGUNAKAN ALGORITMA GENETIKA
(Studi Kasus : *Clustering* Data Bunga Iris)

oleh :
ISROFI FEBRIAWAN
0610960032-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 3 Agustus 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Dian Eka Ratnawati, S.Si., M.Kom	**Nurul Hidayat, S.Pd., M.Sc**
NIP.197306192002122001	**NIP.196804302002121001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	Isrofi Febriawan
NIM	**:**	0610960032-96
Jurusan	**:**	Matematika
Program Studi	**:**	Ilmu Komputer
Penulis skripsi berjudul	**:**	Optimasi Hasil *Clustering* Fuzzy C-Means Menggunakan Algoritma Genetika (Studi Kasus : *Clustering* Data Bunga Iris)

Dengan ini menyatakan bahwa :

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 3 Agustus 2011
Yang menyatakan,

Isrofi Febriawan
NIM. 0610960032-96

OPTIMASI HASIL *CLUSTERING* FUZZY C-MEANS MENGGUNAKAN ALGORITMA GENETIKA
(Studi Kasus : *Clustering* Data Bunga Iris)

ABSTRAK

Fuzzy C-Means (FCM) merupakan salah satu metode *clustering* yang memungkinkan satu bagian data menjadi milik dua atau lebih *cluster*. Penerapan FCM telah dilakukan terhadap berbagai kasus, diantaranya : *clustering* data *performance* mengajar dosen, analisa keluarga miskin, dan penentuan nilai akhir kuliah. Namun pada penelitian-penelitian tersebut hasil *clustering* yang diperoleh hanya menggunakan algoritma FCM. Karena sensitivitas inisiasi random pada FCM menyebabkan hasil *clustering* yang diperoleh terjebak dalam minimum lokal, maka pendekatan algoritma genetika biasanya dapat menghilangkan masalah tersebut.

Proses optimasi hasil *clustering* FCM menggunakan algoritma genetika pada data bunga Iris diawali dengan memasukkan data uji berupa dataset bunga Iris ke dalam proses FCM, kemudian matrik pusat *cluster* hasil proses FCM dievolusikan menggunakan algoritma genetika hingga didapatkan matrik pusat *cluster* yang lebih optimal. Tujuan yang diharapkan dalam proses optimasi hasil *clustering* ini yaitu meminimalkan fungsi objektif Jm. Semakin kecil nilai Jm, maka pusat *cluster* yang dihasilkan semakin optimal.

Berdasarkan hasil pengujian dan analisa yang telah dilakukan dapat diketahui bahwa algoritma genetika dapat diterapkan untuk optimasi hasil *clustering* Fuzzy C-Means dengan cara meminimalkan fungsi objektif Jm. Nilai Jm FCM-GA yang dihasilkan lebih kecil dari nilai Jm FCM untuk semua nilai c (jumlah *cluster*) yang diujicobakan. Selain itu rata-rata nilai *classification rate* (CR) FCM-GA (89.83%) lebih besar dari rata-rata nilai CR FCM (87.33%). Melalui pengujian didapatkan nilai c terbaik yaitu 3. Nilai c terbaik yang didapatkan sesuai dengan *dataset* yang menjadi data uji pada penelitian ini karena *dataset* telah terklasifikasi ke dalam 3 kelas.

Kata kunci : *Clustering*, Fuzzy C-Means (FCM), Algoritma Genetika, FCM-GA, *Classification Rate*, Jm, Bunga Iris.

OPTIMIZATION OF FUZZY C-MEANS CLUSTERING RESULT USING GENETIC ALGORITHM
(Case Study : Data Clustering of Iris Flower)

ABSTRACT

Fuzzy C-Means (FCM) is one of clustering method which allows a part of data belonging to two or more clusters. Application of FCM has been conducted in many cases, including : performance data clustering in lecture's teaching, analysis of marginal people, and the determination of final grades in college. But in those researches, the result obtained of clustering only using FCM algorithm. Because of random initiation sensitivity in FCM causing clustering result which is obtained trapped in local minimum, then genetic algorithm approach usually can solve that problem.

Optimization process of FCM clustering result using genetic algorithm in data of Iris flower begins by entering data testing in Iris dataset form into FCM process, then matrix of cluster center by FCM process evoluted using genetic algorithm to obtained more optimal cluster center. Expected goals in optimization process of clustering result is to minimize the objective function Jm. The smaller value of Jm, then the output of cluster center more optimal.

Based on the results of testing and analysis has been done can be seen that the genetic algorithm can be applied to optimize Fuzzy C-Means clustering result by minimizing the objective function Jm. Value of Jm FCM-GA is smaller than Jm FCM for all c (number of clusters) values which tested. The average value of classification rate (CR) FCM-GA (89.83%) is greater than the average value of CR FCM (87.33%). Through testing found that the best value of c is 3. The best value of c obtained in accordance with the datasets into the data testing in this research because the dataset have been classified into three classes.

Key words : Clustering, Fuzzy C-Means (FCM), Genetic Algorithms, FCM-GA, Classification Rate, Jm, Iris Flower.

KATA PENGANTAR

Alhamdulillahi Robbil 'alamin, puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan skripsi dengan judul "Optimasi Hasil *Clustering* Fuzzy C-Means Menggunakan Algoritma Genetika (Studi Kasus : *Clustering* Data Bunga Iris)" ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer.

Dalam penyelesaian skripsi ini, penulis mengalami keterbatasan ide tentang bagaimana menyusun sebuah laporan. Penulis bersyukur karena dapat menyelesaikan laporan skripsi dengan bantuan dari berbagai pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Dian Eka Ratnawati, S.Si., M.Kom., selaku dosen pembimbing utama yang telah memberikan bantuan berupa masukan, arahan, serta tidak jarang berupa ide yang menunjang penyelesaian skripsi ini.
2. Nurul Hidayat, S.Pd., M.Sc., selaku dosen pembimbing kedua yang telah membantu penulis dalam menyempurnakan penulisan skripsi ini.
3. Drs. Marji, M.T., selaku Ketua Program Studi Ilmu Komputer Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
4. Edy Santoso, S.Si., M.Kom., selaku Pembimbing Akademik.
5. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
6. Segenap dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
7. Segenap staf dan karyawan di Jurusan Matematika Fakultas MIPA Universitas Brawijaya yang telah membantu penulis dalam pelaksanaan penyusunan skripsi ini.
8. Keluarga tercinta dan *special person* yang telah memberikan do'a dan dukungannya selama ini.
9. Rekan-rekan di Program Studi Ilmu Komputer Universitas Brawijaya, khususnya angkatan 2006.

10. Semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan skripsi ini. Oleh karena itu, penulis sangat menghargai saran dan kritik yang sifatnya membangun untuk kelanjutan penelitian serupa di masa mendatang.

Malang, Agustus 2011

Penulis

DAFTAR ISI

BAB III METODOLOGI DAN PERANCANGAN SISTEM

BAB IV IMPLEMENTASI DAN PEMBAHASAN

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Metode *clustering* dengan pemodelan non-fuzzy menghasilkan pencarian yang lengkap pada ruang yang besar, sedangkan pemodelan fuzzy bisa menemukan hasil yang lebih spesifik dari pencarian. Kunci utamanya yaitu menemukan *cluster* yang tepat dari kumpulan data. Hamzah (2001) menunjukkan bahwa proses *cluster* secara fuzzy (*fuzzy clustering*) menunjukkan hasil yang lebih baik dan lebih alami dibandingkan dengan proses *cluster* dengan pendekatan tegas.

Fuzzy C-Means (FCM) adalah salah satu metode *clustering* yang memungkinkan satu bagian data menjadi milik dua atau lebih *cluster*. Metode ini dikembangkan oleh Dunn di tahun 1973 dan diperbaiki oleh Bezdek di tahun 1981.

Penerapan fuzzy seringkali digabungkan dengan algoritma genetika yang disebut sebagai aplikasi *genetic-fuzzy system* (GFS). Algoritma genetika merupakan salah satu metode optimasi yang cukup terkenal. Sejak dikembangkan pertama kali oleh Holland tahun 1975, algoritma genetika terus mengalami perkembangan dalam banyak aplikasi (Gen and Cheng, 2000). Pesatnya perkembangan aplikasi algoritma genetika pada berbagai masalah optimasi dipacu oleh perkembangan teknologi komputer dan mikroprosesor.

Salah satu minat dalam aplikasi *genetic-fuzzy system* (GFS) yaitu *genetic-fuzzy clustering*. Pada *fuzzy clustering* berbasis fungsi objektif, persoalan mencari *cluster* terbaik akan identik dengan persoalan optimasi fungsi objektif.

Penerapan FCM telah dilakukan terhadap berbagai kasus, diantaranya: *clustering* data performance mengajar dosen (Luthfi, 2007), analisa keluarga miskin (Wardani, 2010), dan penentuan nilai akhir kuliah (Khoiruddin, 2007). Namun pada penelitian-penelitian tersebut hasil *clustering* yang diperoleh hanya menggunakan algoritma FCM. Karena sensitivitas inisiasi random pada FCM menyebabkan hasil *clustering* yang diperoleh terjebak dalam minimum lokal, maka pendekatan algoritma genetika biasanya dapat menghilangkan masalah tersebut (Widyastuti dan Hamzah, 2007).

Penelitian optimasi FCM dengan pendekatan algoritma genetika pernah dilakukan sebelumnya dengan menggunakan *least square error* sebagai fungsi objektif (Alata, Molhim, and Ramini, 2008). Perbedaannya dengan penelitian ini yaitu terletak pada fungsi objektifnya. Penelitian ini menggunakan Jm sebagai fungsi objektif. Fungsi objektif Jm didefinisikan sebagai fungsi V (matrik sifat prototipe) dan U (fungsi keanggotaan fuzzy dari objek i=1,2,…,n menjadi anggota cluster j=1,2,…,c) yang diminimumkan melalui iterasi.

Data yang dipilih untuk digunakan dalam studi kasus yaitu dataset bunga iris karena data ini sudah tersedia di internet dan banyak digunakan sebagai bahan untuk menguji pengklasifikasian objek. Data tersebut berdasarkan penelitian Sir Ronald Aylmer Fisher pada tahun 1936. Selain itu, data ini telah terkelompokkan menjadi 3 kelas tertentu sehingga nantinya bisa dihitung *classification rate* hasil *clustering* dari penelitian ini terhadap hasil penelitian Fisher. Hal itulah yang menjadi latar belakang penulisan skripsi ini dengan judul "Optimasi Hasil *Clustering* Fuzzy C-Means Menggunakan Algoritma Genetika (Studi Kasus : *Clustering* Data Bunga Iris)".

1.2 Perumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan pada subbab 1.1, maka perumusan masalah dalam penelitian ini adalah :

1. Bagaimana penerapan algoritma genetika untuk optimasi hasil *clustering* Fuzzy C-Means (FCM),
2. Bagaimana hasil *clustering* FCM-GA (FCM dengan pendekatan algoritma genetika),
3. Berapa jumlah *cluster* terbaik dalam *clustering* data FCM-GA berdasarkan FU (koefisien partisi), HU (entropy partisi), dan SU (fungsi validitas kekompakan dan separasi).

1.3 Batasan Masalah

Untuk menyamakan persepsi tentang judul penelitian ini, maka penulis membatasi ruang lingkup penelitian sebagai berikut :

1. Hasil *clustering* FCM-GA yang dihitung meliputi nilai fungsi objektif Jm dan *classification rate*.
2. Data uji yang diteliti adalah dataset bunga Iris yang diambil dari alamat *http://archive.ics.uci.edu/ml/datasets/Iris*.

3. Penelitian ini hanya dilakukan terhadap hasil *clustering*, tidak termasuk beban dan waktu komputasi algoritma.

1.4 Tujuan Penelitian

Sesuai dengan perumusan masalah yang telah disebutkan maka tujuan penelitian adalah :

1. Menerapkan algoritma genetika untuk optimasi hasil *clustering* Fuzzy C-Means (FCM),
2. Menghitung hasil *clustering* FCM-GA (FCM dengan pendekatan algoritma genetika),
3. Menentukan jumlah *cluster* terbaik dalam *clustering* data FCM-GA berdasarkan FU (koefisien partisi), HU (entropy partisi), dan SU (fungsi validitas kekompakan dan separasi).

1.5 Manfaat Penelitian

Manfaat yang akan diperoleh dari penelitian pada skripsi ini adalah membantu pengelompokan (*clustering*) data bunga Iris ke dalam 3 jenis, yaitu *Iris Setosa, Iris Versicolor, dan Iris Virginica*, berdasarkan panjang kelopak, lebar kelopak, panjang mahkota, dan lebar mahkota.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang telah dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan algoritma Fuzzy C-Means *Clustering* dan optimasi menggunakan algoritma genetika dari berbagai referensi.
2. Pendefinisian dan Analisa Masalah
   Mendefinisikan dan menganalisa masalah untuk mencari solusi yang tepat.
3. Perancangan dan Implementasi Sistem
   Membuat perancangan perangkat lunak dengan analisa terstruktur dan mengimplementasikan hasil rancangan tersebut, yaitu membuat sistem optimasi hasil *clustering* Fuzzy C-Means menggunakan algoritma genetika.

4. Uji Coba dan Analisa Hasil Implementasi
   Menguji perangkat lunak dengan data yang sebenarnya dan menganalisa hasil dari implementasi tersebut.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut :

1. BAB I PENDAHULUAN
   Berisi tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan data bunga Iris, Fuzzy C-Means *Clustering*, dan algoritma genetika.
3. BAB III METODOLOGI DAN PERANCANGAN SISTEM
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam menyelesaikan masalah optimasi hasil *clustering* Fuzzy C-Means menggunakan algoritma genetika untuk studi kasus data bunga Iris.
4. BAB IV IMPLEMENTASI DAN PEMBAHASAN
   Bab ini berisi tentang penjelasan implementasi sistem dan hasil pengujian yang dilakukan.
5. BAB V PENUTUP
   Berisi tentang kesimpulan dari seluruh rangkaian penulisan skripsi ini serta kemungkinan saran pengembangannya.

BAB II
TINJAUAN PUSTAKA

Skripsi ini menggunakan beberapa tinjauan pustaka yang berhubungan dengan permasalahan yang sudah dijelaskan pada bab 1. Pada bab ini akan dijelaskan mengenai teori-teori pendukung yang akan digunakan dalam penyelesaian permasalahan. Tinjauan pustaka yang digunakan meliputi data bunga Iris, optimasi, Fuzzy C-Means (FCM) *clustering*, algoritma genetika, validitas *clustering*, dan *classification rate*.

2.1 Data Bunga Iris

Data bunga Iris merupakan data yang banyak digunakan sebagai bahan pengujian untuk pengklasifikasian objek. Data tersebut berdasarkan penelitian Sir Ronald Aylmer Fisher pada tahun 1936 yang berisi 150 sampel data dan terbagi menjadi 3 kelas, masing-masing terdiri dari 50 sampel data. Masing-masing kelas mewakili jenis bunga Iris yang meliputi : *Iris Setosa, Iris Versicolor, dan Iris Virginica* (Kadir, 2010).

Data ini memiliki 4 atribut untuk setiap sampel data yang mewakili fitur dalam pengklasifikasian, yaitu : panjang kelopak (*sepal length*), lebar kelopak (*sepal width*), panjang mahkota (*petal length*), dan lebar mahkota (*petal width*). Atribut tersebut diukur dalam satuan *centimeter* (cm) (Kadir, 2010).

Gambar 2.1 Tiga Jenis Bunga Iris (Sumber : Kadir, 2010)

2.2 Optimasi

Optimasi adalah suatu pendekatan normatif untuk mengidentifikasi penyelesaian terbaik dalam pengambilan keputusan dari suatu permasalahan. Penyelesaian permasalahan dalam optimasi diarahkan untuk mendapatkan titik maksimum atau titik minimum dari fungsi yang dioptimumkan. Tujuan dari optimasi adalah untuk meminimumkan usaha yang diperlukan dan memaksimalkan hasil yang diinginkan. Jika usaha yang diperlukan atau hasil yang diharapkan dapat dinyatakan sebagai fungsi dari peubah keputusan, maka optimasi dapat didefinisikan sebagai proses pencapaian kondisi maksimum dan minimum dari fungsi tersebut (Maarif, Machfud, dan Sukron, 1989).

Optimasi dapat digunakan untuk fungsi berkendala dan fungsi tidak berkendala. Penyelesaian permasalahan dapat berbentuk persamaan maupun pertidaksamaan. Unsur penting dalam masalah optimasi adalah fungsi tujuan, yang sangat bergantung pada sejumlah peubah masukan. Peubah-peubah ini dapat tidak saling bergantung atau saling bergantung melalui satu atau lebih kendala (Bronson, 1982).

Secara umum, fungsi tujuan merupakan langkah minimasi biaya atau penggunaan bahan baku, maksimasi hasil atau pemanfaatan bahan-bahan produksi atau proses, dan sebagainya. Penentuan fungsi tujuan dikaitkan dengan permasalahan yang dihadapi (Maarif, Machfud, dan Sukron, 1989).

2.3 Fuzzy C-Means (FCM) *Clustering*

Pada pendekatan tegas (*crisp*), untuk setiap objek ke-k secara tegas hanya dapat menjadi anggota cluster ke-i, dengan keputusan menjadi anggota cluster ke-i berdasarkan jarak minimal objek ke-k dengan pusat-pusat cluster ke-i. Algoritma pendekatan tegas dengan jumlah cluster k *clustering* dan pusat cluster ditentukan dengan cara rata-rata ini disebut sebagai K-Means (Widyastuti dan Hamzah, 2007).

Metode *fuzzy clustering* didasarkan pada kenyataan bahwa objek-objek tertentu mungkin tidak dapat dikelompokkan ke dalam cluster tertentu secara tegas. Pada pendekatan fuzzy, setiap objek ke-k dianggap menjadi anggota dari semua cluster ke-i dengan fungsi keanggotaan antara 0 sampai 1. Keputusan objek ke-i menjadi anggota cluster ke-j didasarkan pada fungsi keanggotaan yang

terbesar. Model ini dikenal sebagai Fuzzy C-Means (FCM) *Clustering* (Widyastuti dan Hamzah, 2007).

Konsep dasar FCM, pertama yaitu menentukan pusat cluster yang akan menandai lokasi rata-rata untuk tiap-tiap cluster. Pada kondisi awal, pusat cluster ini masih belum akurat. Tiap data memiliki derajat keanggotaan untuk tiap-tiap cluster. Dengan cara memperbaiki pusat cluster dan nilai keanggotaan tiap-tiap data secara berulang, maka dapat dilihat bahwa pusat cluster akan menuju lokasi yang tepat. Perulangan ini didasarkan pada minimasi fungsi objektif (Gelley, 2000).

Untuk menghasilkan formulasi yang presisi dalam menentukan kriteria clustering dapat ditempuh dengan metode fungsi objektif, yaitu dengan mengukur kemampuan untuk dilibatkan dalam cluster sebagai fungsi dari c (cacah cluster) dengan fungsi objektif tertentu (Zimmerman, 1991). FCM dengan fungsi objektif menggunakan jarak euclidean mengasumsikan bentuk fungsi objektif *spherical*.

Menurut Widyastuti dan Hamzah (2007), pemilihan fungsi objektif dan kriteria jarak sangat bergantung pada sebaran data objek. Kriteria cluster yang baik yaitu cluster yang meminimalkan fungsi objektif Jm, yang dirumuskan sebagai fungsi dari U dan V dalam persamaan 2.1. Semakin kecil nilai Jm, maka pusat cluster yang dihasilkan lebih baik dan hasil *clustering* menjadi lebih optimal.

$$\mathbf{Jm} = \sum_{i=1}^{c} \sum_{k=1}^{n} \mu_{ik}{}^{m} D_{ik}{}^{2} (v_i - \mathrm{x_k}) \qquad (2.1)$$

dimana :
c = jumlah cluster
n = jumlah data
m = derajat ke-*fuzzy*-an
untuk i = 1,2,…,c dan k = 1,2,…,n.

Matrik $U \in M_{cn}$ adalah matrik partisi berordo c x n dengan elemen matrik μ_{ik}. Untuk clustering secara tegas $\mu_{ik} \in \{0,1\}$, yaitu bernilai 1 jika objek ke-k adalah anggota cluster ke-i dan 0 jika tidak. Untuk *clustering* secara fuzzy $\mu_{ik} \in [0,1]$ yang bernilai real. Nilai parameter m merupakan derajat ke-*fuzzy*-an dari proses *clustering* dengan nilai $m \in [1, \infty]$. $V= [v_1, v_2, \ldots, v_c]$ adalah matrik parameter prototipe (pusat cluster) $v_i \in \Re^p$. $D_{ik}{}^2(v_i - x_k)$ adalah jarak dari vektor x_k ke pusat cluster v_i (Widyastuti dan Hamzah, 2007).

Struktur cluster terbaik yang dicari adalah penyelesaian minimum untuk persamaan 2.1 yang dapat diselesaikan dengan proses iterasi untuk penyelesaian optimasi fungsi. Awalnya dibangkitkan vektor pusat cluster secara acak lalu dilakukan iterasi untuk mengupdate matrik U. Iterasi dilakukan hingga matrik U relatif tidak berubah (Widyastuti dan Hamzah, 2007).

$$\mu_{ik} = \frac{\left[\sum_{j=1}^{m}\left(x_{ij} - v_{kj}\right)^2\right]^{\frac{-1}{w-1}}}{\sum_{k=1}^{c}\left[\sum_{j=1}^{m}\left(x_{ij} - v_{kj}\right)^2\right]^{\frac{-1}{w-1}}} \quad (2.2)$$

dimana :
m = jumlah atribut data
c = jumlah cluster
w = derajat ke-*fuzzy*-an
untuk j = 1,2,… , m dan k = 1,2,… , c.

Persamaan 2.2 digunakan untuk mencari nilai U yang telah diupdate. Kemudian vektor pusat cluster bisa dicari melalui persamaan 2.3.

$$V_{kj} = \frac{\sum_{i=1}^{n}\left((\mu_{ik})^w * X_{ij}\right)}{\sum_{i=1}^{n}(\mu_{ik})^w} \quad (2.3)$$

dimana :
n = jumlah data
w = derajat ke-*fuzzy*-an
untuk i = 1,2, … , n.

Penyelesaian secara klasik sering menghasilkan optimum lokal dari himpunan penyelesaian yang mungkin. Oleh karena itu, digunakan pendekatan algoritma genetika untuk menghindari optimum lokal pada proses penyelesaian optimasi.

Khoiruddin (2007) menjelaskan tentang algoritma Fuzzy C-Means Clustering (FCM) yaitu sebagai berikut :

1. Masukkan data awal, dan tentukan parameter yang terlibat. Data berupa matriks ukuran nxm (n = jumlah sampel data, m = atribut data). Parameter yang digunakan yaitu: jumlah cluster

(c), tingkat kekaburan/fuzzy (w), maxIterasi, akurasi (ϵ), fungsi objektif (Jm=0), iterasi awal (t=1),
2. Bangkitkan bilangan random sebagai derajat keanggotaan awal (μ),
3. Hitung pusat cluster tiap cluster menggunakan persamaan 2.3,
4. Hitung fungsi objektif pada iterasi ke-t menggunakan persamaan 2.1,
5. Update derajat keanggotaan μ menggunakan persamaan 2.2,
6. Cek kondisi berhenti :
   a. Jika ($| Jm_t - Jm_{t-1} | < \epsilon$) atau ($t > maxIterasi$) maka berhenti,
   b. Jika tidak, maka $t = t + 1$, ulangi langkah ke-3.

2.4 Algoritma Genetika

Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi alami dan genetika alami (Gen dan Cheng, 1997). Sejak dikembangkan hingga saat ini, algoritma genetika terus menjadi objek riset dalam berbagai aplikasi. Algoritma genetika sering digunakan karena banyak masalah di bidang sains dan teknik tidak dapat dipecahkan dengan algoritma deterministik biasa meskipun dengan waktu yang meningkat secara polinomial (Widyastuti dan Hamzah, 2007).

Konsep dasar yang mengilhami munculnya algoritma genetika yaitu teori evolusi alam yang dikemukakan oleh Charles Darwin. Dalam teori tersebut dijelaskan bahwa pada proses evolusi alami, setiap individu harus melakukan adaptasi terhadap lingkungan disekitarnya agar dapat bertahan hidup.

Individu yang lebih kuat akan memiliki tingkat ketahanan yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Sehingga populasi secara keseluruhannya akan lebih banyak memuat organisme yang fit (Kusumadewi dan Purnomo, 2005).

2.4.1 Komponen Dasar Algoritma Genetika

Algoritma genetika merupakan teknik pencarian yang didasarkan pada mekanisme seleksi dan genetika alami. Berbeda dengan teknik pencarian konvensional, algoritma genetika dimulai dengan membentuk kumpulan solusi (kromosom) yang dinamakan populasi. Solusi dari suatu populasi akan diambil untuk membentuk populasi baru. Solusi ini diambil berdasarkan nilai *fitness*. Proses

pembentukan populasi baru ini dilakukan dengan harapan bahwa populasi baru yang terbentuk merupakan populasi yang lebih baik dari sebelumnya (Dakka, 2009).

Secara umum, sebuah algoritma genetika mempunyai lima komponen dasar (Gen dan Cheng, 1997), yaitu :

1. Representasi kromosom berdasarkan permasalahan.
2. Bagaimana cara untuk membentuk initial populasi.
3. Mengevaluasi kromosom berdasarkan nilai *fitness*.
4. Operator genetika yang merubah komposisi kromosom anak selama proses reproduksi.
5. Nilai parameter dari algoritma genetika.

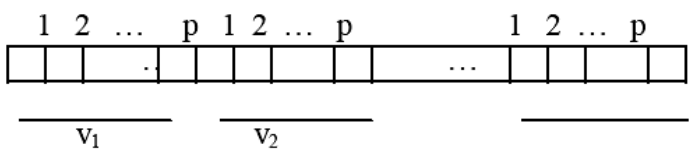
2.4.2 Pengkodean (*Encoding*)

Pengkodean solusi masalah ke dalam suatu kromosom di dalam algoritma genetika merupakan hal yang penting. Populasi awal yang berisi N kromosom dibangkitkan secara acak yang menjangkau keseluruhan ruang solusi. Proses evolusi dilakukan dengan melakukan operasi genetik (*crossover* dan mutasi) dan melakukan seleksi kromosom untuk generasi berikutnya hingga sejumlah generasi yang diinginkan dengan panduan fungsi *fitness* (Widyastuti dan Hamzah, 2007).

Algoritma genetika pada ruang solusi disebut phenotip dan algoritma pada ruang coding disebut genotip. Operasi genetika dilakukan pada ruang genotip, sedangkan operasi seleksi dilakukan pada ruang phenotip. Dalam *binary encoding* variabel keputusan diwakili oleh deretan bit 0 dan 1 yang panjangnya disesuaikan dengan ruang pencarian. Tiap bit 0,1 dapat dianggap sebagai sebuah gen (Widyastuti dan Hamzah, 2007).

Integer dan *literal permutation encoding* adalah kode terbaik untuk masalah *combinatorial optimization* karena inti dari masalah ini adalah mencari kombinasi atau permutasi terbaik dari item solusi terhadap kendala. *Encoding* dengan menggunakan data terstruktur lebih sesuai digunakan untuk masalah yang lebih komplek.

Encoding yang digunakan dalam penelitian ini yaitu *real encoding*, Struktur kromosom untuk matrik pusat cluster V dalam populasi yang dievolusikan adalah vektor real beranggotakan c x p elemen (c = cacah cluster dan p = cacah elemen dalam objek), seperti ditunjukkan gambar 2.2 (Widyastuti dan Hamzah, 2007).



Gambar 2.2 Struktur Kromosom untuk Encoding V

2.4.3 Fungsi *Fitness*

Fungsi *fitness* adalah fungsi yang digunakan untuk menentukan apakah suatu kromosom layak bertahan. Pada setiap generasi dipilih kromosom yang mendekati solusi dengan mengevaluasi fungsi kecocokan dari kromosom tersebut. Fungsi ini didefinisikan sedemikian rupa sehingga semakin besar nilai *fitness* maka semakin besar probabilitas untuk terseleksi pada generasi berikutnya. Untuk maksimasi, fungsi tujuan dapat dijadikan sebagai fungsi *fitness* sehingga kromosom yang mewakili nilai fungsi besar akan memiliki probabilitas terseleksi yang besar juga. Untuk minimasi dapat dirumuskan sedemikian rupa sehingga fungsi tujuan yang semakin kecil maka memiliki fungsi *fitness* yang besar (Widyastuti dan Hamzah, 2007).

Fungsi *fitness* bisa dimodifikasi dengan menambahkan suatu fungsi yang disebut fungsi penalti untuk penyelesaian yang berada di luar daerah *visible*, sehingga fungsi *fitness* menjadi seperti persamaan 2.4.

$$\text{fitness}(x) = f(x) + p(x) \qquad (2.4)$$

Fungsi *fitness* yang digunakan dalam penelitian ini adalah fungsi objektif Jm, yang dihitung menggunakan persamaan 2.1, dimana μ_{ik} dihitung dengan menggunakan persamaan 2.2.

2.4.4 Seleksi

Proses seleksi bertanggung jawab untuk melakukan pemilihan terhadap individu yang akan diikutkan dalam proses reproduksi. Langkah pertama yang dilakukan dalam proses seleksi adalah pencarian nilai *fitness*. Tujuan seleksi adalah untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang mempunyai nilai *fitness* terbaik. Metode seleksi yang digunakan dalam penelitian ini yaitu seleksi roda roulette.

Metode seleksi roda roulette merupakan suatu metode yang menirukan permainan *roulette-wheel* dimana masing-masing

kromosom menempati potongan lingkaran pada roda roulette secara proporsional sesuai dengan nilai *fitness*-nya. Kromosom yang memiliki nilai *fitness* lebih besar menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom bernilai *fitness* rendah (Suyanto, 2005).

Cinantya (2010) menjelaskan cara kerja seleksi *roulette wheel* yaitu sebagai berikut :

1. Menghitung nilai *fitness* setiap kromosom pada suatu populasi,
2. Menghitung total nilai *fitness* dari semua kromosom dalam populasi,
3. Menghitung peluang dari setiap kromosom yang ada,
4. Menghitung peluang kumulatif dari setiap kromosom,
5. Meng-*generate* nilai acak antara 1 – 100,
6. Melakukan pencarian pada kromosom yang ada dengan menjumlahkan nilai peluang kumulatif tiap kromosom. Jika diperoleh nilai lebih besar dari nilai acak, maka hentikan penelusuran dan pilih kromosom tersebut.

Metode seleksi roda roulette akan bermasalah saat terdapat perbedaan *fitness* yang besar. Misalnya, jika *fitness* kromosom yang terbaik adalah 90% dari semua roda *roulette* dapat menyebabkan kromosom lain memiliki kesempatan yang kecil untuk dapat terpilih.

2.4.5 Operator Genetika

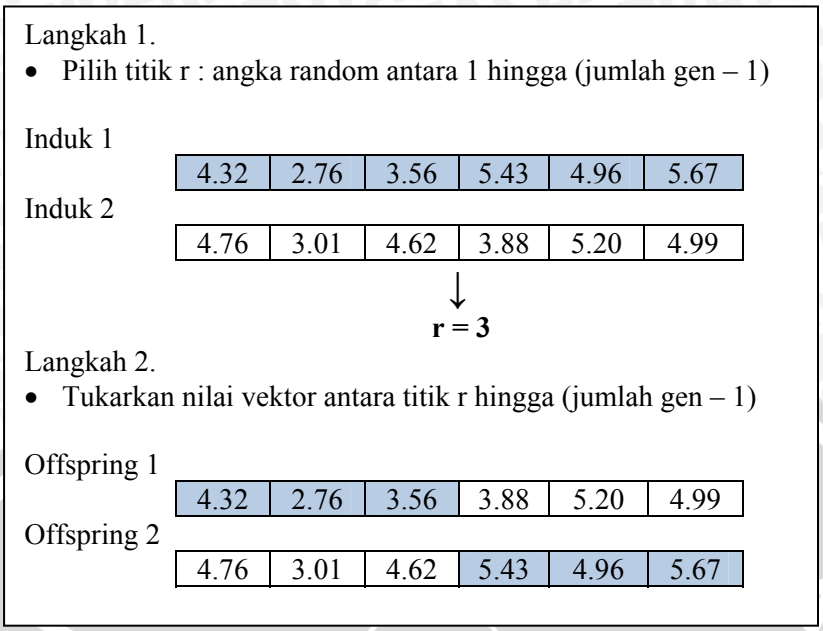
Operator genetika yang akan dibahas ada 2 macam, yaitu perkawinan silang dan mutasi.

2.4.5.1 Perkawinan Silang (*Crossover*)

Setelah menentukan metode pengkodean apa yang dipakai, maka langkah selanjutnya adalah perkawinan silang atau *crossover*. Proses perkawinan silang (*crossover*) berfungsi untuk menghasilkan keturunan dari dua buah kromosom induk yang terpilih. Kromosom anak yang dihasilkan merupakan kombinasi gen-gen yang dimiliki oleh kromosom induk.

Operasi *crossover* dipilih metode *one-cut point crossover*, yaitu untuk tiap kromosom induk ditetapkan 1 titik secara acak antara 1 hingga jumlah gen dalam kromosom, misal disebut titik r. Selanjutnya offspring didapat dengan menyilangkan (*crossover*) nilai mulai posisi r hingga akhir kromosom (Widyastuti dan Hamzah,

2007). Untuk lebih jelasnya urutan *crossover* disajikan dalam gambar 2.3.



Gambar 2.3 *One-Cut Point Crossover* untuk Kromosom

2.4.5.2 Mutasi

Setelah melalui proses perkawinan silang, proses mutasi dapat dilakukan pada *offspring*. Mutasi dilakukan dengan cara merubah sebuah gen atau lebih dari sebuah individu. Mutasi bertujuan untuk membentuk individu-individu yang memiliki kualitas di atas rata-rata. Selain itu mutasi digunakan untuk mengembalikan kerusakan materi genetika akibat proses *crossover*. Mutasi sangat berguna dalam mempertahankan keanekaragaman individu dalam populasi meskipun melalui mutasi tidak dapat diketahui apa yang terjadi pada individu baru.

Cara paling sederhana untuk melakukan mutasi adalah dengan mengubah satu atau lebih gen (bagian dari kromosom) dengan probabilitasnya tergantung pada *mutation rate*. Operasi ini dimulai dengan memilih individu dari populasi berdasarkan nilai *fitness*nya. Akan dipilih sebuah titik pada kromosom secara *random* dan gen

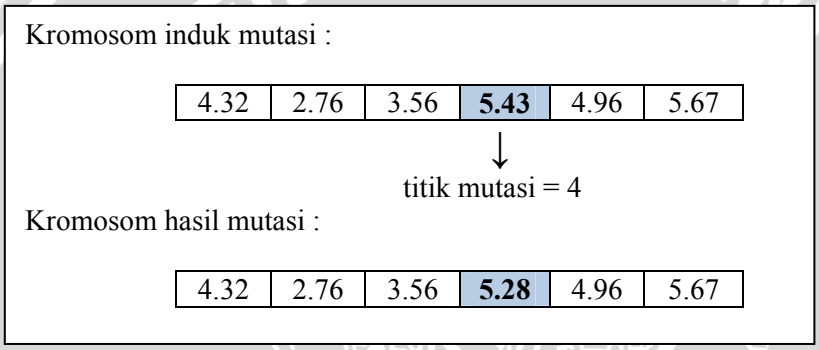
kromosom pada titik tersebut akan dirubah nilainya secara *random*. Perubahan individu yang dihasilkan tersebut kemudian akan dimasukkan ke dalam populasi.

Mutasi nilai gen dari suatu kromosom dapat dilakukan dengan dua cara (Novitasari, 2007), yaitu :

1. Cara Random, yaitu menentukan dua gen yang akan dimutasi. Setelah itu nilai kedua gen tersebut dirandom ulang untuk mendapatkan nilai yang baru.
2. Cara Swap atau penukaran yaitu dengan menukar langsung nilai dari gen. Pemilihan gen yang akan ditukar dilakukan secara random.

Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi.Tetapi bila peluang mutasi terlalu besar, maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya (Mawaddah, 2006).

Ilustrasi proses mutasi kromosom pada algoritma genetika digambarkan pada gambar 2.4.



Gambar 2.4 Mutasi Kromosom

Metode mutasi yang dipilih dalam penelitian ini yaitu *nonuniform mutation*. Algoritma mutasi yang digunakan dapat dituliskan sebagai berikut (Widyastuti dan Hamzah, 2007) :

1. Pilih kromosom yang akan dimutasi berdasarkan nilai Pm yang telah ditentukan
2. Tentukan digit (posisi) gen yang akan dimutasi, dengan menentukan k, yaitu bilangan random antara 1 hingga n
3. Pilih gen yang akan dimutasi, yaitu x_k
4. Tentukan batas bawah (x_k^L) dan batas atas (x_k^U)

5. Nilai offspring diperoleh (x_k') dipilih secara acak dari salah satu :
   $x_k' = x_k + \Delta(t, x_k^U - x_k)$ (2.5)
   atau
   $x_k' = x_k - \Delta(t, x_k - x_k^L)$ (2.6)
   dengan kisaran nilai real untuk x_k adalah $[x_k^L, x_k^U]$.
   Berdasarkan nilai $\Delta(t, x_k^U - x_k)$ atau $\Delta(t, x_k - x_k^L)$
   maka diperoleh :
   $\Delta(t,y) = yr(1 - t/T)^b$ (2.7)
   dengan r = bilangan random kisaran [0,1] dan b adalah bilangan bulat.

Untuk lebih jelasnya mengenai proses *nonuniform mutation*, bisa dilihat pada perhitungan manual di bab 3.

2.4.6 Parameter Genetika

Pengoperasian algoritma genetika dibutuhkan 4 parameter (Juniawati, 2003) yaitu :

1. Probabilitas Persilangan (*Probability Crossover*)
   Probabilitas persilangan menunjukkan kemungkinan *crossover* terjadi antara 2 kromosom. Jika tidak terjadi *crossover* maka keturunannya akan sama persis dengan kromosom orang tua, tetapi tidak berarti generasi yang baru akan sama persis dengan generasi yang lama. Jika probabilitas *crossover* 100% maka semua keturunannya dihasilkan dari *crossover*. *Crossover* dilakukan dengan harapan bahwa kromosom yang baru akan lebih baik.
2. Probabilitas Mutasi (*Probability Mutation*)
   Probabilitas mutasi menunjukkan kemungkinan mutasi terjadi pada gen-gen yang menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika probabilitasnya 100% semua kromosom dimutasi. Jika probabilitasnya 0% tidak ada yang mengalami mutasi.
3. Jumlah Individu
   Jumlah individu menunjukkan jumlah kromosom yang terdapat dalam populasi (dalam satu generasi). Jika hanya sedikit kromosom dalam populasi maka algoritma genetika akan mempunyai sedikit variasi kemungkinan untuk melakukuan *crossover* antara orang tua karena hanya sebagian kecil dari

search space yang dipakai. Sebaliknya jika terlalu banyak maka algoritma genetika akan berjalan lambat.

4. Jumlah Populasi

   Jumlah populasi menentukan banyaknya generasi yang dihasilkan, digunakan sebagai batas akhir proses seleksi, persilangan, dan mutasi.

2.5 Validitas *Clustering*

Hasil akhir FCM maupun FCM dengan pendekatan algoritma genetika (FCM-GA) adalah V,U dan Jm tertentu untuk suatu nilai c yang diinputkan. Pada beberapa kasus c yang tepat mungkin tidak diketahui. Untuk itu beberapa pendekatan telah diusulkan untuk menentukan nilai c sehingga hasil *clustering* dapat dianggap terbaik. Ukuran ini disebut sebagai validitas clustering. Chi et.al. (1996) menyebutkan ada 3 alat ukur yang digunakan yaitu :

1. **Koefisien partisi (*partition coefficient*)**

   Koefisien partisi didefinisikan sebagai fungsi U (derajat kenggotaan) dan c (jumlah cluster), dirumuskan melalui persamaan 2.8.

$$F(U,c) = \frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^2 \qquad (2.8)$$

   dimana :
   c = jumlah cluster
   n = jumlah data
   untuk i = 1,2,… , c dan k = 1,2,… , n.
   Koefisien partisi ini mengukur kedekatan dari semua sampel masukan terhadap prototipe yang terpilih. Untuk selanjutnya, koefisien partisi atau F(U,c) disingkat menjadi FU.

2. **Entropy partisi (*partition entropy*)**

   Entropy partisi didefinisikan sebagai fungsi U (derajat kenggotaan) dan c (jumlah cluster), dirumuskan melalui persamaan 2.9.

$$H(U,c) = -\frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}u_{ik}\log(u_{ik}) \qquad (2.9)$$

   dimana :
   c = jumlah cluster
   n = jumlah data

untuk i = 1,2,… , c dan k = 1,2,… , n.

Entropy partisi (0 < H(U,c) < log c) diterjemahkan sebagai derajat ketidakpastian (*degree of fuzziness*) dari objek. H(U,c) besar menunjukkan hasil clustering yang jelek. Untuk selanjutnya, entropy partisi atau H(U,c) disingkat menjadi HU.

3. **Fungsi validitas kekompakan dan separasi (*compactness and separation validity function*)**

Fungsi validitas kekompakan dan separasi didefinisikan sebagai fungsi U (derajat keanggotaan) dan c (jumlah cluster), dirumuskan melalui persamaan 2.10.

$$S(U,c) = \frac{\frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{2} \mid x_k - v_i \mid^2}{\min_{i,k} \mid v_k - v_i \mid^2} \qquad (2.10)$$

dimana :

c = jumlah cluster

n = jumlah data

untuk i = 1,2,… , c dan k = 1,2,… , n.

S(U,c) adalah rasio antara rata-rata jarak sampel dengan prototype yang terpilih dengan jarak minimum antar prototype. Untuk selanjutnya, fungsi validitas kekompakan dan separasi atau S(U,c) disingkat menjadi SU.

2.6 *Classification Rate*

Keberhasilan *clustering* terkadang diukur juga melalui kemampuan *clustering* dalam klasifikasi objek. Hal ini dapat ditentukan apabila objek-objek yang diklasifikasi telah diketahui berasal dari suatu kelas tertentu. Banyaknya objek yang dapat diklasifikasikan dengan tepat sesuai dengan kelas dari mana objek tersebut berasal disebut sebagai *classification rate*. Index ini dapat dituliskan seperti persamaan 2.11 (Widyastuti dan Hamzah, 2007).

$$CR = \frac{n}{N} x100\% \qquad (2.11)$$

dimana:

n = banyaknya objek yang dapat diklasifikasi dengan benar

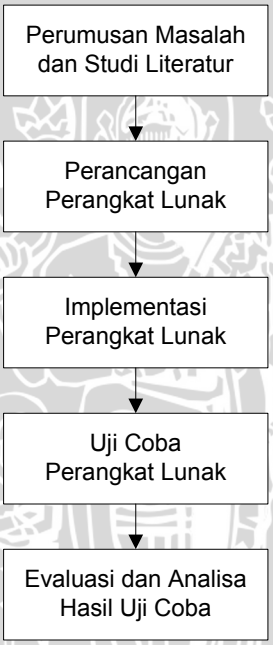
N = banyaknya objek yang diklasifikasi

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan sistem ini akan dibahas langkah-langkah yang digunakan dalam pembuatan perangkat lunak dan metode percobaan. Tahapannya adalah sebagai berikut.

1. Mempelajari metode yang digunakan melalui jurnal yang pernah ada maupun literatur lain dan merumuskan masalah yang dihadapi.
2. Merancang perangkat lunak.
3. Membuat perangkat lunak berdasarkan perancangan yang sudah dilakukan.
4. Uji coba perangkat lunak dengan memasukkan data uji.
5. Evaluasi dan analisa hasil uji coba yang sudah dilakukan oleh sistem.

Langkah- langkah pembuatan perangkat lunak digambarkan pada gambar 3.1.



Gambar 3.1 Diagram Alir Pembuatan Perangkat Lunak

3.1 Analisa Sistem

Pada subbab analisa sistem akan dijelaskan deskripsi sistem, batasan sistem, dan data yang digunakan pada penelitian skripsi ini.

3.1.1 Deskripsi Sistem

Sistem yang akan dibangun merupakan sistem yang dibuat untuk implementasi teori-teori yang didapat dari berbagai literatur mengenai permasalahan optimasi hasil *clustering* Fuzzy C-Means (FCM) menggunakan algoritma genetika. Sistem yang dibangun digunakan untuk menguji dan menganalisa hasil yang didapat dari permasalahan baik menggunakan algoritma FCM konvensional maupun menggunakan algoritma FCM dengan penambahan algoritma genetika (FCM-GA).

3.1.2 Batasan Sistem

Sistem ini dibangun untuk menyelesaikan permasalahan optimasi hasil *clustering* Fuzzy C-Means Clustering (FCM). Metode yang digunakan yaitu algoritma genetika untuk meningkatkan kinerja dari algoritma FCM. Beberapa parameter input yang digunakan dalam penelitian ini yaitu :

1. FCM :
   a. Tingkat kekaburan (*fuzziness*) = 2
   b. Batas iterasi = 100
   c. Batas akurasi ditentukan melalui uji coba 4 macam nilai, yaitu 0.1, 0.01, 0.001, dan 0.0001. Nilai yang digunakan adalah nilai yang menghasilkan Jm terbaik (paling kecil).
2. Algoritma Genetika :
   a. Jumlah individu populasi awal = 100
   b. Jumlah generasi = 500
   c. Probabilitas *crossover* (Pc) = 0.9
   d. Probabilitas mutasi (Pm) = 0.2

Parameter input ini akan diujikan pada jumlah *cluster* 2, 3, 4, dan 5.

3.1.3 Data yang Digunakan

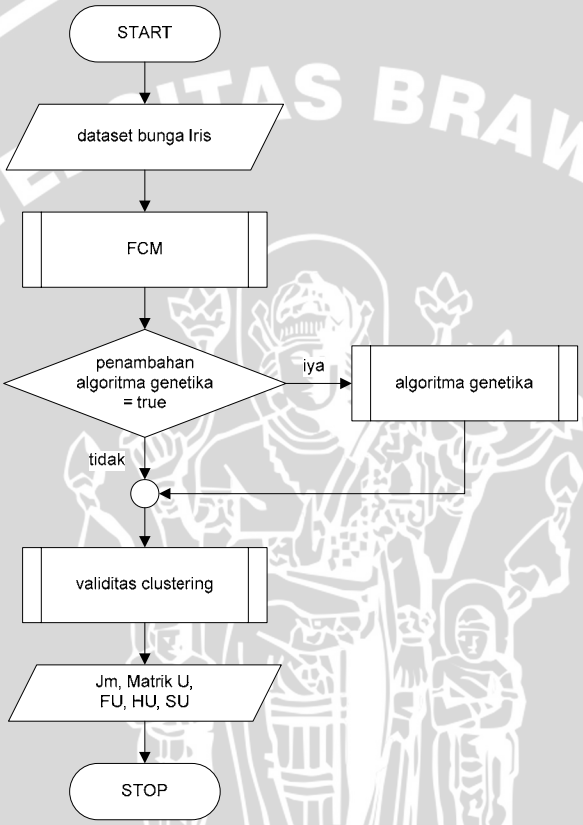
Pada penelitian ini, data yang digunakan dalam pengujian sistem adalah salah satu *dataset* yang disediakan oleh *UCI Machine Learning Repository* pada alamat website berikut.

- *http://archive.ics.uci.edu/ml/datasets/Iris*

data yang digunakan merupakan *dataset* bunga iris sebanyak 150 sampel data yang terbagi dalam 3 kelas, masing-masing sebanyak 50 sampel data. Tiap data memiliki 4 atribut.

3.2 Perancangan Sistem

Sistem yang akan dibangun menggunakan algoritma genetika untuk optimasi hasil *clustering* Fuzzy C-Means (FCM). Tahapan dari perancangan dimulai dari perhitungan menggunakan algoritma FCM, kemudian digunakan pendekatan algoritma genetika pada FCM serta menganalisa hasil perhitungan dari proses FCM sebelum dan sesudah ditambahkan algoritma genetika. Diagram alir perancangan sistem ini bisa dilihat pada gambar 3.2.



Gambar 3.2 Diagram Alir Perancangan Sistem

Data yang digunakan pada penelitian yaitu *dataset* bunga Iris. Data ini akan diproses menggunakan algoritma Fuzzy C-Means Clustering (FCM) sehingga diperoleh titik pusat cluster, nilai fungsi objektif (Jm), dan matrik derajat keanggotaan.

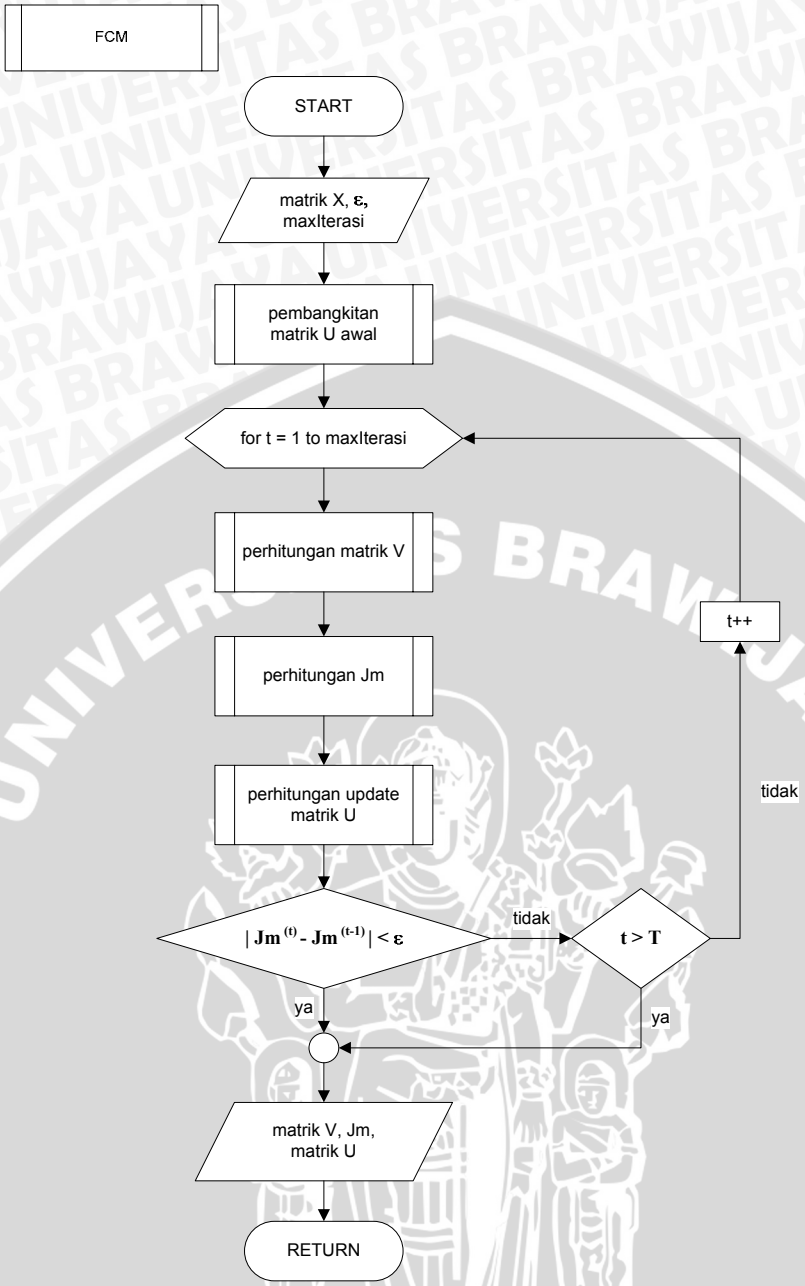
Titik pusat cluster dievolusikan menggunakan algoritma genetika sehingga diperoleh titik pusat cluster yang optimal. Titik pusat cluster hasil FCM dengan penambahan algoritma genetika (FCM-GA) digunakan untuk mencari nilai Jm yang baru dan dibandingkan dengan nilai Jm hasil algoritma FCM konvensional.

Selanjutnya dicari update matrik derajat keanggotaan untuk menentukan anggota cluster dari masing-masing data. Penentuan anggota cluster ini berguna untuk menghitung *classification rate*. Selain itu juga dicari nilai FU, HU, dan SU pada FCM maupun FCM-GA untuk menentukan jumlah *cluster* terbaik dari beberapa jumlah cluster yang diujicobakan.

3.2.1 Algoritma Fuzzy C-Means Clustering Konvensional

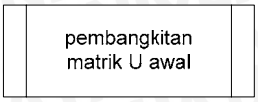
Algoritma Fuzzy C-Means (FCM) *Clustering* konvensional merupakan penyelesaian fuzzy clustering dengan cara iteratif, yaitu melakukan update pada matrik keanggotaan U dan matrik prototype cluster V. Dalam algoritma diperlukan sampel objek sebanyak n, tiap objek m parameter, dituliskan : $X=\{ x_1, x_2, \ldots , x_n\}$ $x_i \in R^P$, i=1,2,…,n. Ditentukan dalam proses penyelesaian melalui iterasi :
$U = [\mu_{ik}]$ matrik ukuran cxn; k=1,2,…,c ; i=1,2,…,n.
V = {v1,v2,…,vc}

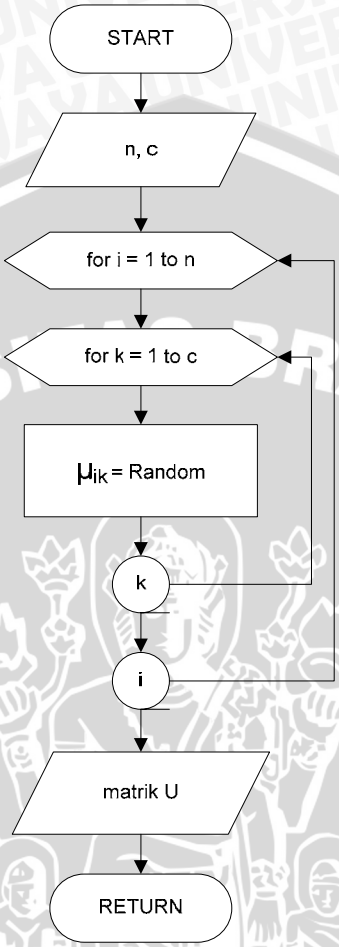
Diagram alir untuk algoritma Fuzzy C-Means Clustering (FCM) dijelaskan pada gambar 3.3.



Gambar 3.3 Diagram Alir Fuzzy C-Means (FCM)

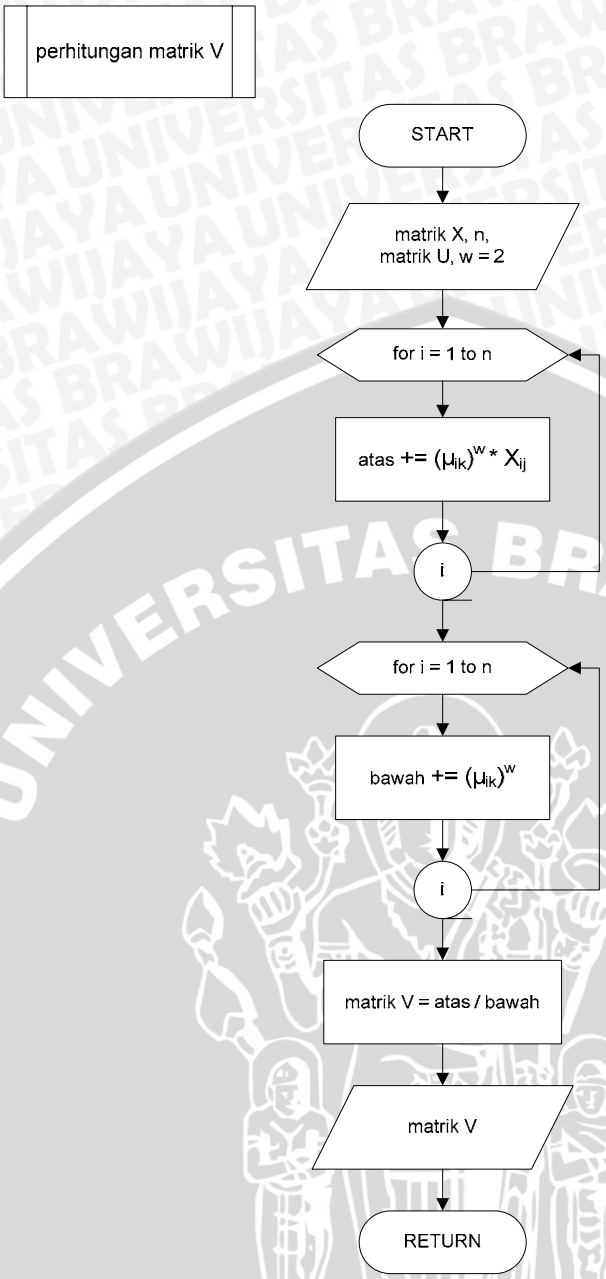
Pada pembangkitan matrik derajat keanggotaan (matrik U) awal, prosesnya dijelaskan pada diagram alir gambar 3.4.





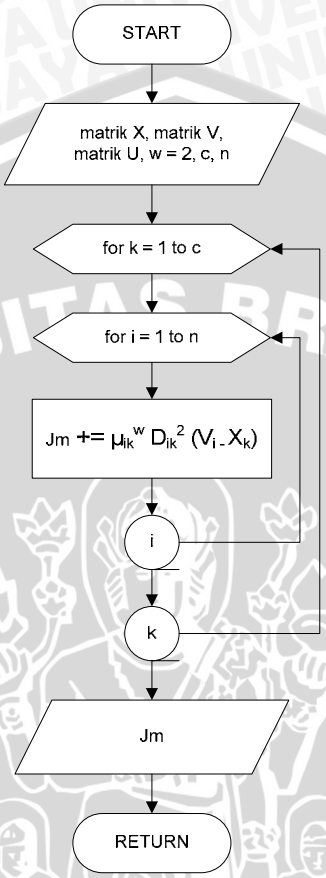
Gambar 3.4 Diagram Alir Pembangkitan Matrik U Awal

Pada perhitungan matrik pusat cluster (matrik V), prosesnya dijelaskan pada diagram alir gambar 3.5.



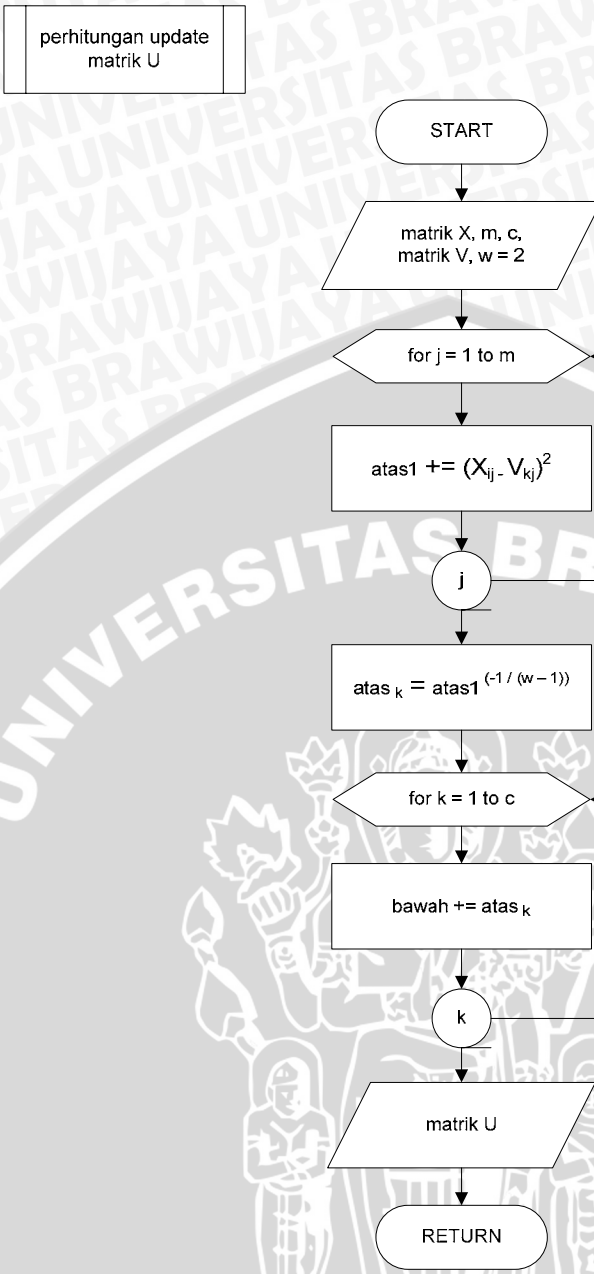
Gambar 3.5 Diagram Alir Perhitungan Matrik Pusat Cluster

Dari matrik pusat cluster yang telah didapatkan, maka fungsi objektif Jm bisa dihitung. Untuk perhitungan fungsi objektif Jm, prosesnya bisa dilihat pada diagram alir gambar 3.6.



Gambar 3.6 Diagram Alir Perhitungan Fungsi Objektif Jm

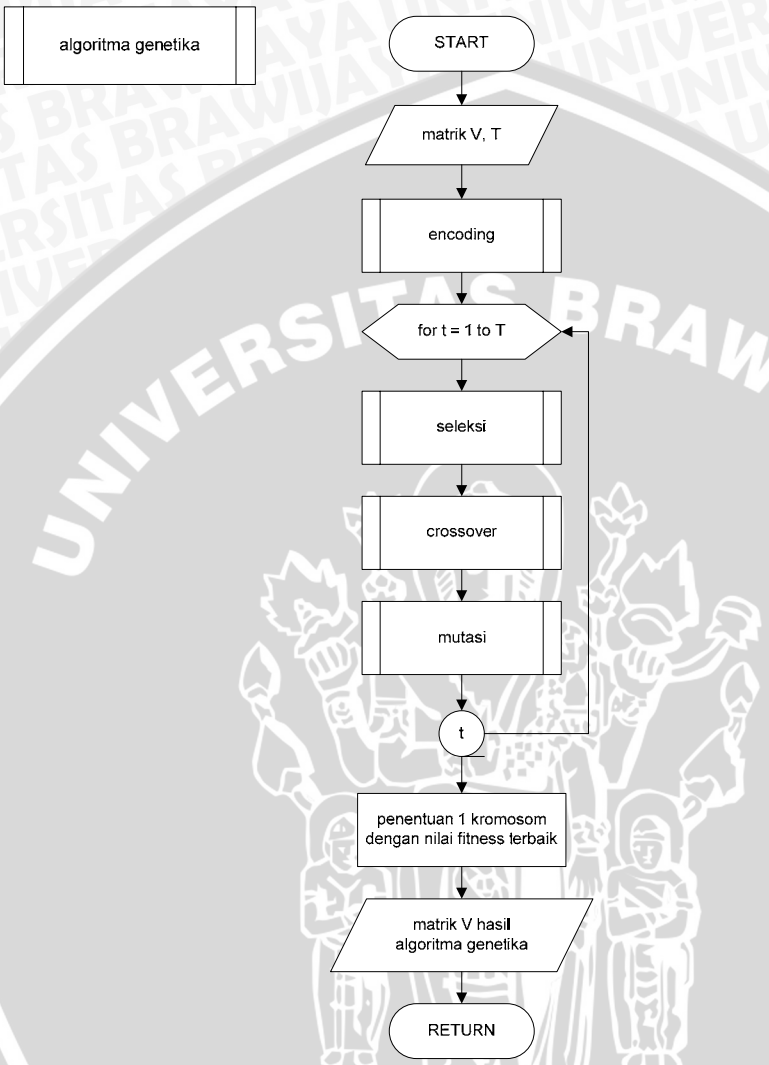
Matrik derajat keanggotaan (matrik U) yang semula didapatkan melalui pembangkitan nilai secara random, diupdate dengan menggunakan perhitungan yang prosesnya dijelaskan pada diagram alir gambar 3.7.



Gambar 3.7 Diagram Alir Perhitungan Update Matrik U

3.2.2 Algoritma Fuzzy C-Means Clustering dengan Pendekatan Algoritma Genetika (FCM-GA)

Pada pendekatan algoritma genetika untuk penyelesaian fuzzy c-means clustering digunakan pendekatan prototype-based algorithm, yaitu mengevolusikan matrik pusat cluster V. Diagram alir algoritma genetika yang digunakan dijelaskan pada gambar 3.8.

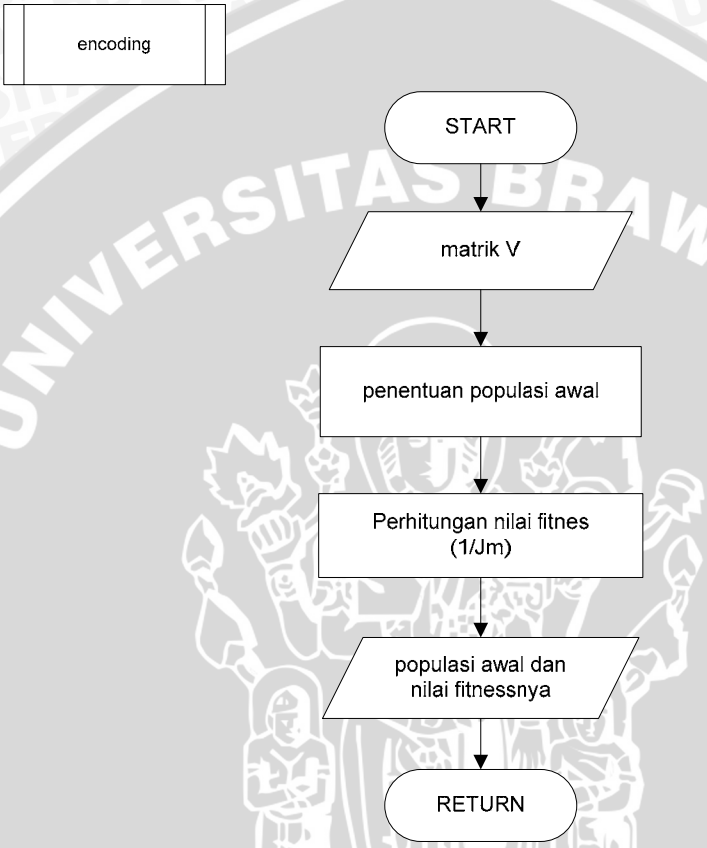


Gambar 3.8 Diagram Alir Algoritma Genetika

Adapun rangkaian tahap dalam algoritma genetika yang digunakan yaitu :

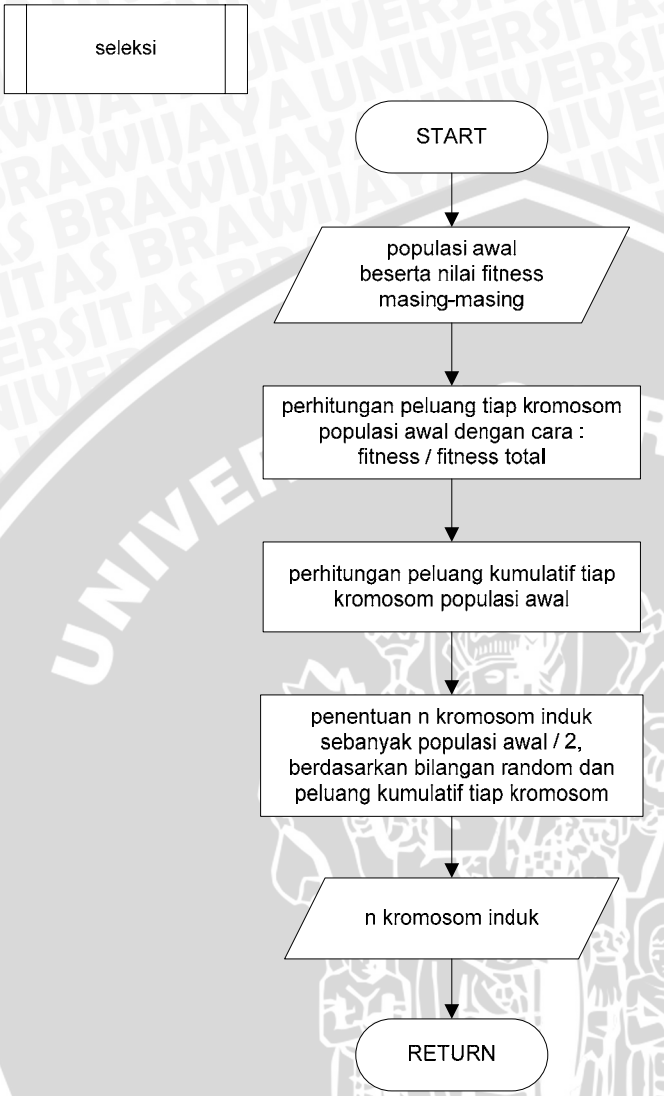
a. *Encoding*, menggunakan metode *real encoding*,
b. Fungsi fitness, menggunakan fungsi objektif Jm,
c. Seleksi, menggunakan metode *roulette wheel.*
d. *Crossover*, menggunakan metode *one cut-point crossover*,
e. Mutasi, menggunakan metode *nonuniform mutation*.

Pada diagram alir algoritma genetika gambar 3.8 terdapat 4 proses utama, yaitu encoding, seleksi, crossover, dan mutasi. Untuk proses encoding dijelaskan pada diagram alir gambar 3.9.



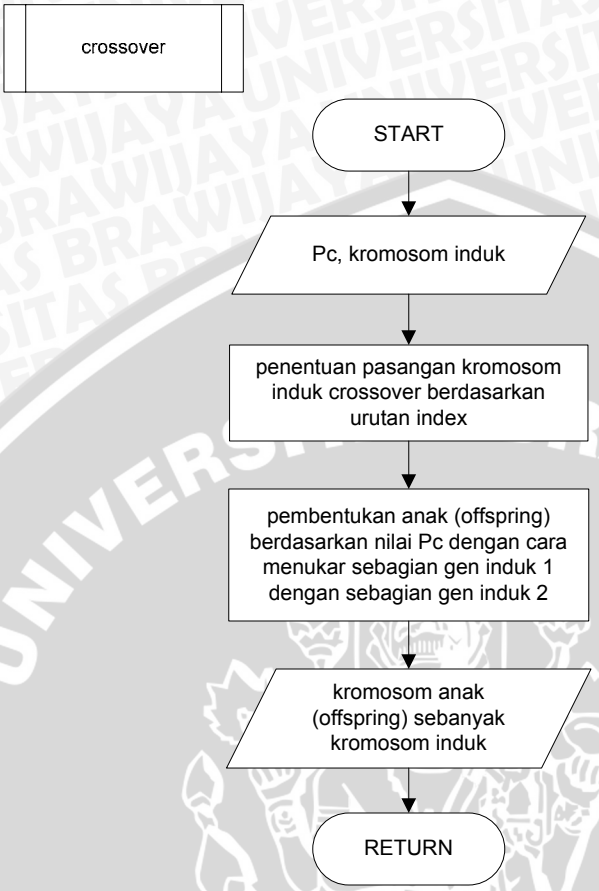
Gambar 3.9 Diagram Alir *Encoding*

Setelah proses encoding maka dilanjutkan pada proses seleksi, dimana kromosom hasil encoding diseleksi berdasarkan nilai fitnessnya. Proses seleksi dijelaskan pada diagram alir gambar 3.10.



Gambar 3.10 Diagram Alir Seleksi

Populasi hasil seleksi digunakan sebagai induk untuk proses berikutnya, yaitu crossover. Adapun proses crossover dijelaskan pada diagram alir gambar 3.11.



Gambar 3.11 Diagram Alir *Crossover*

Kromosom hasil crossover dijadikan sebagai kandidat kromosom yang akan dimutasi. Proses mutasi dijelaskan melalui diagram alir gambar 3.12.

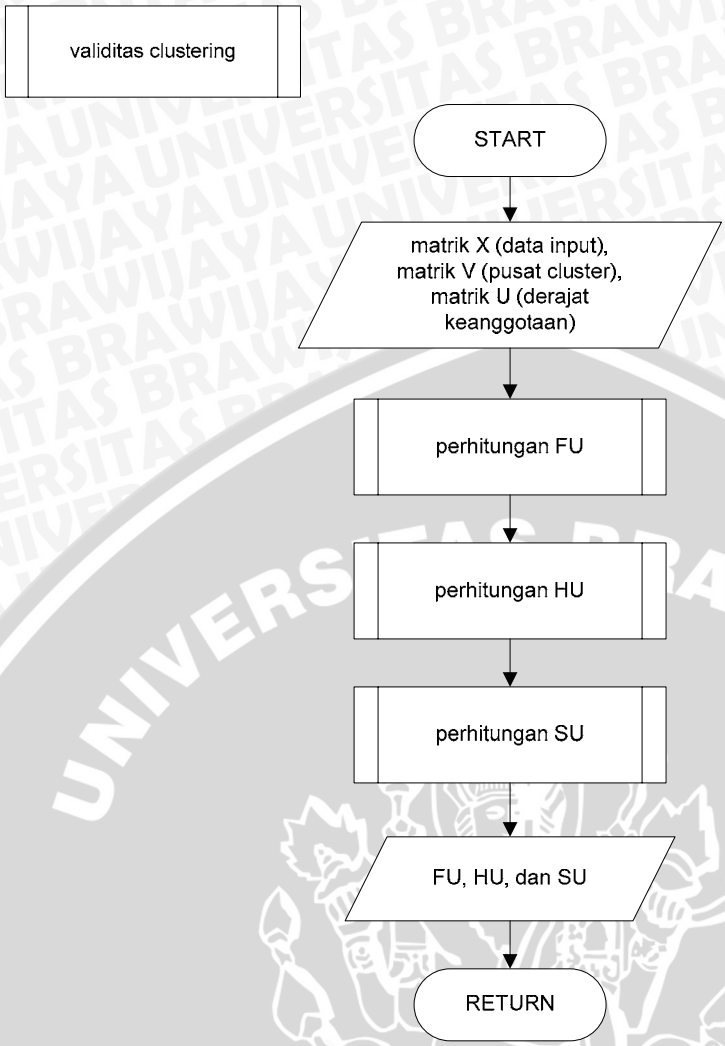
Gambar 3.12 Diagram Alir Mutasi

Setelah proses mutasi maka akan dicek apakah generasi ini sudah mencapai jumlah maksimum generasi, apabila belum maka proses algoritma genetika diulangi lagi. Populasi hasil proses mutasi digabungkan dengan populasi awal dan dipilih kromosom dengan nilai fitness terbaik sebanyak populasi awal. Kromosom-kromosom terpilih ini yang dijadikan sebagai populasi awal pada generasi berikutnya. Proses iterasi dilakukan hingga mencapai syarat berhenti, yaitu jika iterasi sudah mencapai jumlah maksimum generasi. Jika syarat berhenti terpenuhi, maka dipilih satu kromosom dengan nilai fitness terbaik.

Kromosom terpilih merupakan matrik pusat cluster hasil proses algoritma genetika. Matrik pusat cluster ini digunakan untuk mencari nilai fungsi objektif (Jm) dan matrik derajat keanggotaan.

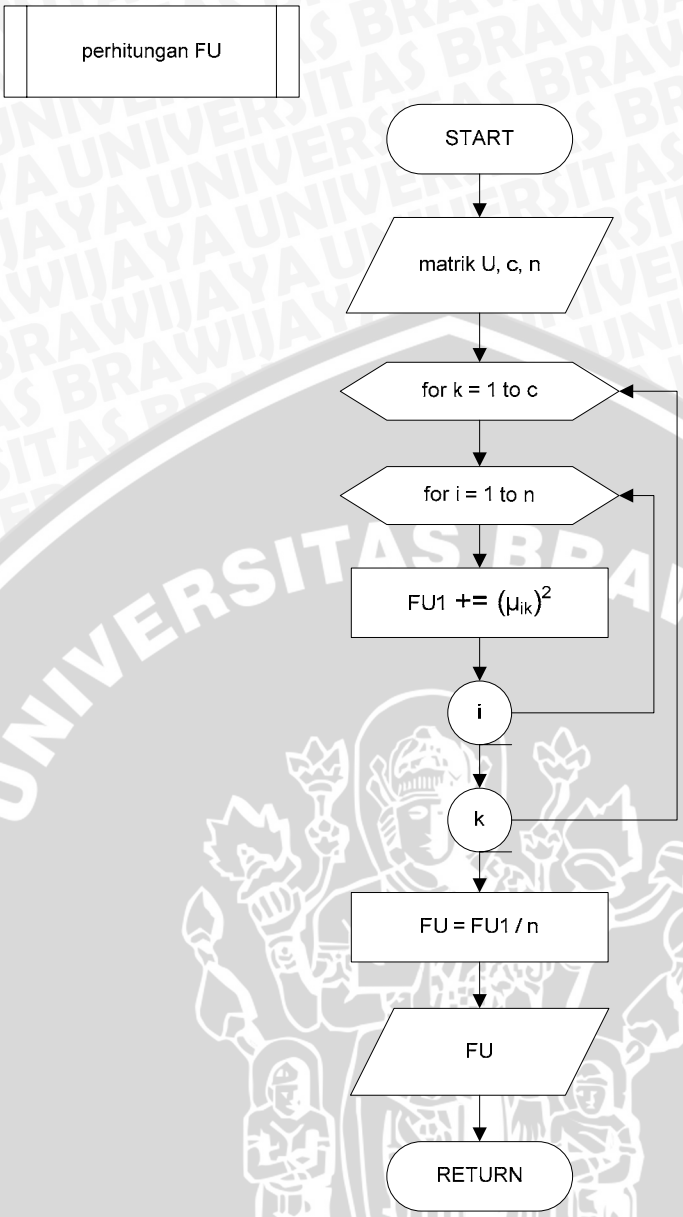
Pada bagian akhir proses FCM maupun FCM-GA, dicari nilai validitas clustering yang digunakan untuk pemilihan jumlah clustering yang terbaik dari beberapa nilai c yang diujicobakan.

Proses perhitungan validitas clustering terbagi menjadi 3 bagian, yaitu : perhitungan FU (koefisien partisi), perhitungan HU (entropy partisi), dan perhitungan SU (fungsi validitas kekompakan dan separasi). Proses perhitungan validitas clustering dijelaskan dalam diagram alir gambar 3.13.



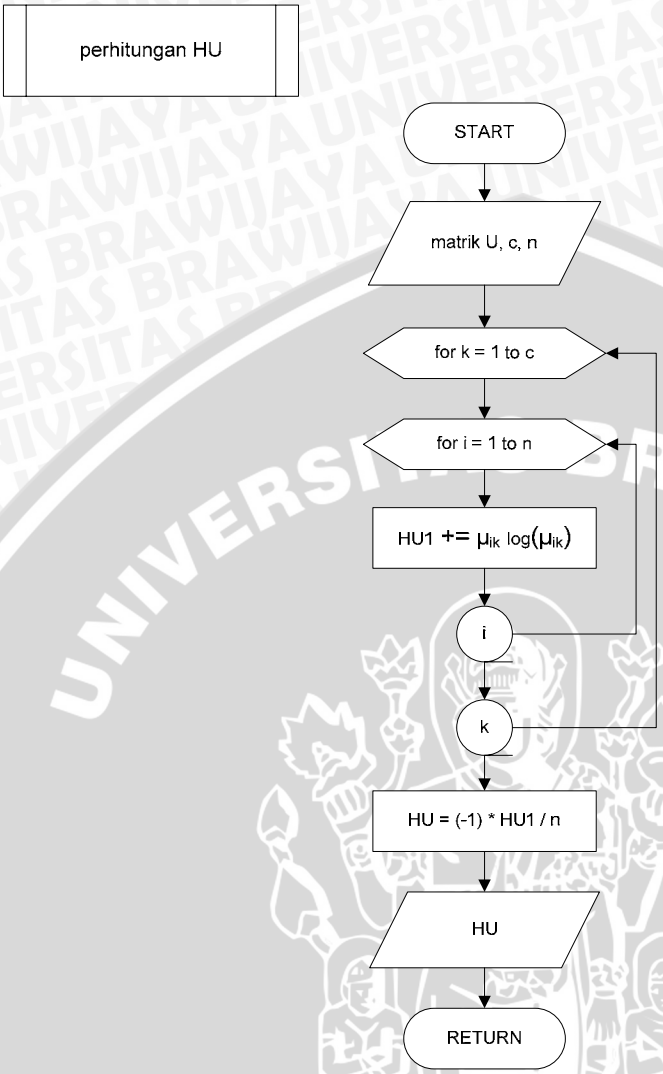
Gambar 3.13 Diagram Alir Perhitungan Validitas Clustering

Proses perhitungan FU (koefisien partisi) dijelaskan pada diagram alir gambar 3.14.



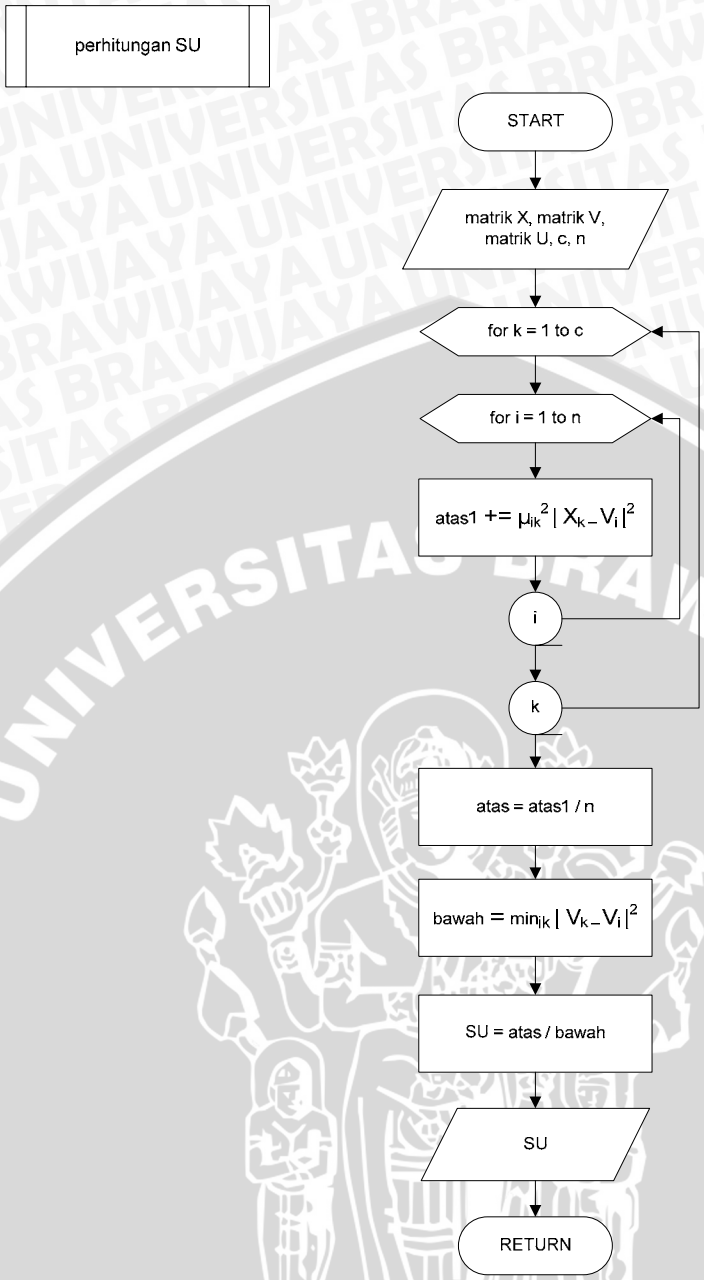
Gambar 3.14 Diagram Alir Perhitungan FU

Proses perhitungan HU (entropy partisi) dijelaskan pada diagram alir gambar 3.15.



Gambar 3.15 Diagram Alir Perhitungan HU

Proses perhitungan SU (fungsi validitas kekompakan dan separasi) dijelaskan pada diagram alir gambar 3.16.



Gambar 3.16 Diagram Alir Perhitungan SU

3.3 Perhitungan Manual

Perhitungan manual yang akan dijelaskan yaitu perhitungan manual FCM dan perhitungan manual algoritma genetika.

3.3.1 Perhitungan Manual FCM

Contoh permasalahan yang digunakan pada perhitungan proses FCM menggunakan data input sebanyak 5 buah data, dimana tiap data memiliki 2 atribut. Data input ini digambarkan pada tabel 3.1.

Tabel 3.1 Data Input untuk Proses FCM

Data ke-	Atribut 1	Atribut 2
1	34.0	11.0
2	20.9	8.8
3	27.7	10.2
4	19.0	10.2
5	21.6	8.7

3.3.1.1 Penentuan Data Masukan dan Parameter Awal

Berdasarkan data pada tabel 3.1, maka data masukan berupa matrik 5x2.

$$X = \begin{pmatrix} 34.0 & 11.0 \\ 20.9 & 8.8 \\ 27.7 & 10.2 \\ 19.0 & 10.2 \\ 21.6 & 8.7 \end{pmatrix}$$

Parameter-parameter awal yang digunakan yaitu:

- Jumlah cluster (c) = 2
- Tingkat kekaburan / ke-*fuzzy*-an (w) = 2
- Iterasi maksimal (maxIterasi) = 10
- Akurasi (ϵ) = 0.1
- Fungsi objektif awal (Jm_0) = 0
- Iterasi awal (t) = 1

3.3.1.2 Penentuan Derajat Keanggotaan Awal

Untuk menentukan derajat keanggotaan awal (μ) yaitu dibangkitkan bilangan secara random antara 0 hingga 1. Penentuan bilangan random ini berupa matrik U yang berukuran n x c, dimana n adalah jumlah data dan c adalah jumlah cluster. Misalkan saja data yang dibangkitkan yaitu sebagai berikut:

$$U = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \\ 0.3 & 0.7 \\ 0.8 & 0.2 \end{pmatrix}$$

Pada saat membangkitkan bilangan random matrik U, jumlah bilangan pada tiap barisnya harus sama dengan 1. Misalkan pada baris pertama, bilangan-bilangan random yang dibangkitkan yaitu 0.6 dan 0.4 dimana 0.6 + 0.4 = 1 dan seterusnya. Tabel matrik X dan matrik U bisa dilihat pada tabel 3.2 dan 3.3.

Tabel 3.2 Matrik Data Input (Matrik X)

Data (i)	X_{ij}	
	1	**2**
1	34	11
2	20.9	8.8
3	27.7	10.2
4	19	10.2
5	21.6	8.7

Tabel 3.3 Matrik Derajat Keanggotaan (Matrik U)

Data (i)	μ_{ik}	
	1	**2**
1	0.6	0.4
2	0.4	0.6
3	0.7	0.3
4	0.3	0.7
5	0.8	0.2

3.3.1.3 Perhitungan Pusat Cluster

Pusat cluster pada masing-masing cluster dihitung berdasarkan persamaan 2.3.

Hasil dari perhitungan berupa matrik V dengan ukuran c x m, dimana c adalah jumlah cluster dan m merupakan jumlah atribut data.

Berdasarkan persamaan tersebut, maka perhitungan pusat cluster pada masing-masing cluster yaitu:

$V_{11} = \sum ((\mu_{i1})^2 * X_{i1}) / \sum((\mu_{i1})^2)$
$= ((0.6)^2 * 34) + (0.4)^2 * 20.9 + (0.7)^2 * 27.7 + (0.3)^2 * 19 +$
$(0.8)^2 * 21.6) / ((0.6)^2 + (0.4)^2 + (0.7)^2 + (0.3)^2 + (0.8)^2)$
$= 25.684$

$V_{12} = \sum((\mu_{i1})^2 * X_{i2}) / \sum((\mu_{i1})^2)$
$= ((0.6)^2 * 11) + (0.4)^2 *8.8 + (0.7)^2 * 10.2 + (0.3)^2 * 10.2 +$
$(0.8)^2 * 8.7) / ((0.6)^2 + (0.4)^2 + (0.7)^2 + (0.3)^2 + (0.8)^2)$
$= 9.685$

$V_{21} = \sum((\mu_{i2})^2 * X_{i1}) / \sum((\mu_{i2})^2)$
$= ((0.4)^2 * 34) + (0.6)^2 * 20.9 + (0.3)^2 * 27.7 + (0.7)^2 * 19 +$
$(0.2)^2 * 21.6) / ((0.4)^2 + (0.6)^2 + (0.3)^2 + (0.7)^2 + (0.2)^2)$
$= 22.483$

$V_{22} = \sum((\mu_{i2})^2 * X_{i2}) / \sum((\mu_{i2})^2)$
$= ((0.4)^2 *11) + (0.6)^2 * 8.8 + (0.3)^2 * 10.2 + (0.7)^2 * 10.2 +$
$(0.2)^2 *8.7) / ((0.4)^2 + (0.6)^2 + (0.3)^2 + (0.7)^2 + (0.2)^2)$
$= 9.818$

Berdasarkan persamaan tersebut, maka matrik pusat cluster V yang didapat yaitu sebagai berikut:

$$V_{kj} = \begin{bmatrix} 25.684 & 9.685 \\ 22.483 & 9.818 \end{bmatrix}$$

Tabel matrik V bisa dilihat pada tabel 3.4.

Tabel 3.4 Matrik Pusat Cluster (Matrik V)

k	V_{kj}	
	1	2
1	25.684	9.685
2	22.483	9.818

3.3.1.4 Perhitungan Fungsi Objektif

Untuk menghitung fungsi objektif pada iterasi ke-t, digunakan persamaan 2.1. Perhitungan fungsi objektif dari contoh permasalahan ini yaitu sebagai berikut:

$C_{i1} = ((X_{i1} - V_{11})^2 + (x_{i2} - V_{12})^2) * (\mu_{i1})^2$

$C_{11} = ((34 - 25.684)^2 + (11 - 9.685)^2) * (0.6)^2 = 25.519$
$C_{21} = ((20.9 - 25.684)^2 + (8.8 - 9.685)^2) * (0.4)^2 = 3.787$
$C_{31} = ((27.7 - 25.684)^2 + (10.2 - 9.685)^2) * (0.7)^2 = 2.121$
$C_{41} = ((19 - 25.684)^2 + (10.2 - 9.685)^2) * (0.3)^2 = 4.045$
$C_{51} = ((21.6 - 25.684)^2 + (8.7 - 9.685)^2) * (0.8)^2 = 11.296$

$C_{i2} = ((X_{i1} - V_{21})^2 + (x_{i2} - V_{22})^2) * (\mu_{i2})^2$

$C_{12} = ((34 - 22.483)^2 + (11 - 9.818)^2) * (0.4)^2 = 21.446$
$C_{22} = ((20.9 - 22.483)^2 + (8.8 - 9.818)^2) * (0.6)^2 = 1.275$
$C_{32} = ((27.7 - 22.483)^2 + (10.2 - 9.818)^2) * (0.3)^2 = 2.463$
$C_{42} = ((19 - 22.483)^2 + (10.2 - 9.818)^2) * (0.7)^2 = 6.016$
$C_{52} = ((21.6 - 22.483)^2 + (8.7 - 9.818)^2) * (0.2)^2 = 0.081$

$Jm = \sum (C_{i1} + C_{i2}) = 78.049$

3.3.1.5 Update Derajat Keanggotaan μ

Dari nilai Jm yang telah diperoleh maka nilai derajat keanggotaan μ_{ik} yang berupa matrik U, diupdate menggunakan persamaan 2.2. Tabel perhitungan untuk meng-update derajat keanggotaan dari contoh permasalahan ini bisa dilihat pada tabel 3.5.

Tabel 3.5 Perhitungan Update Matrik Derajat Keanggotaan

I	$(X_{i1} - V_{11})^2$ L1	$(X_{i2} - V_{12})^2$ L2	$(X_{i1} - V_{21})^2$ L3	$(X_{i2} - V_{22})^2$ L4	LTotal
1	69.156	1.729	132.641	1.397	204.923
2	22.887	0.783	2.506	1.036	27.212
3	4.064	0.265	27.217	0.1460	31.692
4	44.676	0.265	12.131	0.1460	57.218
5	16.679	0.970	0.780	1.250	19.679

Kemudian hasil perhitungan untuk mendapatkan update derajat keanggotaan bisa dilihat pada tabel 3.6.

Tabel 3.6 Hasil Update Derajat Keanggotaan

μ_{i1}	μ_{i2}
(L1+L2)/LT	(L3+L4)/LT
0.346	0.654
0.870	0.130
0.137	0.863
0.785	0.215
0.897	0.103

Dari tabel 3.6, maka matrik U diupdate menjadi :

$$U = \begin{pmatrix} 0.346 & 0.654 \\ 0.870 & 0.130 \\ 0.137 & 0.863 \\ 0.785 & 0.215 \\ 0.897 & 0.103 \end{pmatrix}$$

3.3.1.6 Cek Kondisi Berhenti

Berdasarkan perhitungan fungsi objektif Jm, diketahui bahwa $Jm_t - Jm_{t-1} = 78.049 - 0 = 78.049$, maka $| Jm_t - Jm_{t-1} | > \epsilon$. Iterasi (t) ini adalah iterasi ke-1 sehingga t < maxIterasi. Hal ini berarti iterasi dilanjutkan ke iterasi ke-2 (t=2). Proses perhitungan diulangi kembali mulai perhitungan pusat cluster dengan menggunakan matrik update derajat keanggotaan yang baru.

Pada persoalan ini, iterasi dilakukan hingga iterasi ke-6 karena pada iterasi ke-6, nilai $| Jm_t - Jm_{t-1} | = | 22.519 - 22.567 | = 0.048$ sehingga $| Jm_t - Jm_{t-1} | < \epsilon$.

$$V_6 = \begin{bmatrix} 20.626 & 9.240 \\ 31.465 & 10.678 \end{bmatrix}$$

Jm = 22.519

$$U = \begin{bmatrix} 0.965 & 0.035 \\ 0.002 & 0.998 \\ 0.780 & 0.220 \\ 0.022 & 0.978 \\ 0.012 & 0.988 \end{bmatrix}$$

Dari matrik update derajat keanggotaan μ (matrik U) tersebut, maka *clustering* datanya bisa dilihat pada tabel 3.7.

Tabel 3.7 *Clustering* Data Hasil FCM

Data (i)	Cluster 1	Cluster 2	Masuk Cluster ke-
1	0.965	0.035	1
2	0.002	0.998	2
3	0.780	0.220	1
4	0.022	0.978	2
5	0.012	0.988	2

3.3.1.7 Validitas Clustering

Setelah diperoleh matrik U dari perhitungan sebelumnya, maka dilakukan pengukuran terhadap validitas clustering menggunakan tiga tolak ukur, yaitu FU, HU, dan SU. FU merupakan koefisien partisi untuk mengukur kedekatan dari semua sampel masukan terhadap prototype yang terpilih. HU merupakan entropy partisi yang diterjemahkan sebagai derajat ketidakpastian dari objek. Sedangkan SU merupakan fungsi validitas kekompakan dan separasi, yaitu rasio antara rata-rata jarak sampel dengan prototype yang terpilih dengan jarak minimum antar prototype.

Berdasarkan persamaan 2.8 maka perhitungan FU bisa dilihat pada tabel 3.8.

Tabel 3.8 Perhitungan Nilai FU

i	$(\mu_{i1})^2$	$(\mu_{i2})^2$	A+B
	A	B	
1	0.931225	0.001225	0.93245
2	0.000004	0.996004	0.996008
3	0.6084	0.0484	0.6568
4	0.000484	0.956484	0.956968
5	0.000144	0.976144	0.976288
∑			4.518514

FU = 4.519 / 5 = 0.904

Sedangkan berdasarkan persamaan 2.9 maka perhitungan HU seperti tabel 3.9.

Tabel 3.9 Perhitungan Nilai HU

i	$\mu_{i1} \log(\mu_{i1})$	$\mu_{i2} \log(\mu_{i2})$	A+B
	A	B	
1	-0.014931143	-0.050957618	-0.065888761
2	-0.00539794	-0.00086772	-0.00626566
3	-0.08416621	-0.14466701	-0.22883322
4	-0.036466701	-0.0094486	-0.045915301
5	-0.023049825	-0.005180139	-0.028229964
∑			-0.375132906

HU = -0.375 / -5 = 0.075

Berdasarkan persamaan 2.10 maka perhitungan SU yaitu sebagai berikut:

SU = Jm / n * jarak minimum antar pusat cluster

$= 22.519 / 5 * \sqrt{((20.626 - 31.465)^2 + (9.240 - 10.678)^2)}$

= 22.519 / 54.67

= 0.412

3.3.2 Perhitungan Manual Algoritma Genetika

Pada pendekatan algoritma genetika untuk optimasi hasil *clustering* Fuzzy C-Means digunakan pendekatan *prototype-based algorithm*, yaitu mengevolusikan matrik pusat cluster V.

3.3.2.1 *Encoding* Kromosom

Teknik encoding yang digunakan yaitu *real encoding*. Penetapan populasi awal ditentukan melalui pembentukan 4 buah kromosom yang nilai gen-nya ditentukan secara acak dengan interval antara nilai minimum dan nilai maksimum matrik V hasil proses FCM, yaitu V_6. Misalkan kromosom yang terbentuk seperti gambar 3.17.

Kromosom 1 :

20.32	9.4	31.12	10.54

Kromosom 2 :

9.41	20.43	10.11	31.76

Kromosom 3 :

10.54	20.98	9.99	31.09

Kromosom 4 :

20.43	10.65	31.12	9.24

Gambar 3.17 Kromosom Populasi Awal

3.3.2.2 Nilai *Fitness*

Perhitungan nilai *fitness* berdasarkan fungsi objektif Jm yang dihitung menggunakan persamaan 2.1. Dari hasil perhitungan Jm maka nilai *fitness* bisa didapatkan melalui persamaan : fitness = 1/Jm. Hal ini dikarenakan tujuan dari penelitian ini yaitu minimasi fungsi objektif Jm. Nilai Jm dan *fitness* dari masing-masing kromosom bisa dilihat pada tabel 3.10.

Tabel 3.10 Nilai Jm dan Fitness Kromosom Populasi Awal

Kromosom	Jm	Fitness
1	23.088	0.043313
2	2164.887	0.000462
3	2095.686	0.000477
4	31.937	0.031312

3.3.2.3 Seleksi

Kromosom-kromosom pada populasi awal yang telah dibangkitkan diseleksi dengan menggunakan metode *roulette wheel*. Dengan seleksi ini maka individu yang baik (nilai *fitness*-nya besar) akan mempunyai peluang yang lebih besar untuk terpilih.

Populasi awal diseleksi menjadi setengah dari populasi semula, maka populasi awal yang semula berjumlah 4 kromosom diseleksi menjadi 2 kromosom. Misalkan yang terpilih dalam proses seleksi ini yaitu kromosom 1 dan 4 maka kromosom-kromosom inilah yang akan diproses lebih lanjut.

3.3.2.4 *Crossover*

Metode yang digunakan dalam proses *crossover* ini adalah *one-cut point crossover*, yaitu untuk tiap kromosom induk ditetapkan 1 titik secara acak antara 1 hingga cxp, misal disebut titik r. Selanjutnya *offspring* didapat dengan menyilangkan (*crossover*) nilai mulai posisi r hingga akhir kromosom. Proses *crossover* tergantung pada suatu parameter yaitu probabilitas *crossover* (Pc). Misalkan probabilitas *crossover* adalah 0.5, artinya diharapkan 50% kromosom/individu yang akan mengalami *crossover*.

Langkah pertama yang dilakukan adalah membangkitkan nilai *random* antara 0 sampai 1. Jika nilai random < 0,5 maka kromosom tersebut yang akan dikenai proses *crossover*. Misalkan nilai *random* yang didapat adalah 0,76; 0, 81; 0,32; 0,21. Dari nilai *random* yang didapat, maka diperoleh 2 kromosom yang akan mengalami proses *crossover*, yaitu kromosom 3 dan kromosom 4. Langkah-langkah proses *one-cut point crossover* yaitu sebagai berikut.

Langkah 1.

Kromosom 1 (induk 1) :

20.32	9.4	31.12	10.54

Kromosom 4 (induk 2):

20.43	10.65	31.12	9.24

Gambar 3.18 Pembentukan Induk *Crossover*

Langkah 2.

Induk 1	20.32	9.4	31.12	10.54
Induk 2	20.43	10.65	31.12	9.24
Offspring 1	20.32	9.4	31.12	9.24
Offspring 2	20.43	10.65	31.12	10.54

Gambar 3.19 Pembentukan *Offspring Crossover*

3.3.2.5 Mutasi

Metode mutasi yang digunakan yaitu *nonuniform mutation.* Langkah-langkahnya adalah sebagai berikut :

Langkah 1.

Mutasi dilakukan berdasarkan pada probabilitas mutasi (Pm) yang telah ditentukan. Misalkan ditentukan nilai probabilitas mutasi sebesar 0.5, maka diharapkan akan ada 50% kromosom pada populasi yang akan mengalami proses mutasi. Lalu dibangkitkan nilai *random* antara 0 sampai 1. Jika nilai random < 0,25 maka kromosom tersebut akan mengalami mutasi. Misalkan nilai random yang didapat adalah 0,15 dan 0,83, maka kromosom yang akan mengalami mutasi adalah kromosom *Offspring* 1 hasil crossover.

1	2	3	4
20.32	9.4	31.12	9.24

Gambar 3.20 Kromosom yang akan Dimutasi

Langkah 2.

Ditentukan digit (posisi) gen yang akan dimutasi, dengan menentukan k, yaitu bilangan random antara 1 hingga n. Misal posisi yang terpilih adalah posisi ke-1.

Langkah 3.

Dipilih gen yang akan dimutasi, yaitu $x_k = 20.32$.

Langkah 4.

Tentukan batas bawah ($x_k^L = 9.24$) dan batas atas ($x_k^U = 31.12$)

Langkah 5.

Nilai *offspring* diperoleh melalui salah satu persamaan 2.5 atau 2.6 yang dipilih secara acak. Misalkan setelah pemilihan acak, terpilih persamaan 2.6. Dengan menggunakan nilai t = 1, T = 150, r = 0.3, dan b = 2, maka didapatkan nilai $x_k' = 20.74$.

Dari langkah-langkah tersebut, didapatkan kromosom hasil mutasi sebagai berikut :

Kromosom *offspring* mutasi :

20.74	9.4	31.12	9.24

Kromosom hasil mutasi ini digabungkan dengan populasi awal dan dipilih 4 kromosom dengan nilai fitness terbaik. Misalkan nilainya seperti tabel 3.11.

Tabel 3.11 Nilai Fitness Seluruh Kromosom

Kromosom	Fitness
Kromosom 1	0.043313
Kromosom 2	0.000462
Kromosom 3	0.000477
Kromosom 4	0.031312
Kromosom *offspring* 2	0.034772
Kromosom *offspring* mutasi	0.038423

Dari tabel 3.11, dipilih 4 kromosom dengan nilai fitness terbaik untuk dijadikan sebagai populasi akhir generasi ke-1. Populasi ini nantinya akan melanjutkan proses genetika dan menghasilkan generasi-generasi berikutnya hingga generasi terakhir (misalkan ditentukan jumlah generasi T = 150).

Jika tabel 3.11 merupakan generasi terakhir, maka dipilih 1 kromosom dengan nilai fitness terbaik, yaitu Kromosom 1. Kromosom inilah yang digunakan sebagai matrik pusat cluster baru untuk mencari nilai update matrik derajat keanggotaan.

Kromosom 1 :

20.32	9.4	31.12	10.54

Matrik pusat cluster menjadi :

$$V_{kj} = \begin{bmatrix} 20.32 & 9.4 \\ 31.12 & 10.54 \end{bmatrix}$$

Tabel 3.12 Update Matrik U dan Clustering Data Hasil FCM-GA

Data (i)	Cluster 1	Cluster 2	Masuk Cluster ke-
1	0.957	0.043	1
2	0.006	0.994	2
3	0.823	0.177	1
4	0.016	0.984	2
5	0.022	0.978	2

Selanjutnya bisa dicari nilai validitas clustering FU, HU, dan SU sebagaimana proses pada FCM.

FU = 4.54 / 5 = 0.908

HU = -0.377 / -5 = 0.0754

SU = Jm / n * jarak minimum antar pusat cluster
= 23.088 / 5 * $\sqrt{((20.32 - 31.12)^2 + (9.4 - 10.54)^2)}$
= 23.088 / 54.3
= 0.425

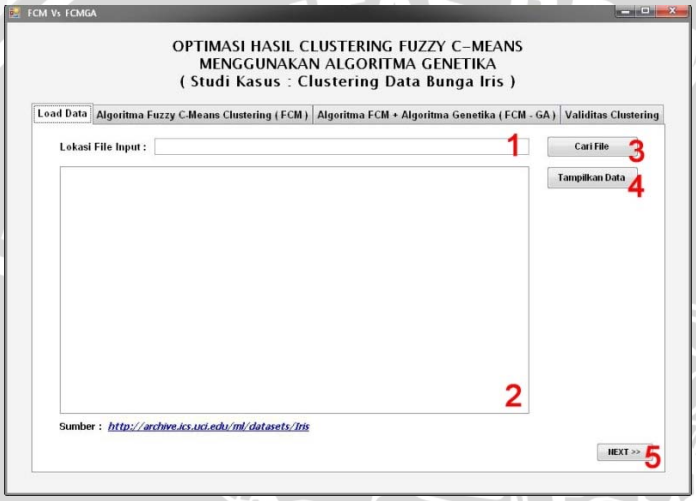
3.4 Rancangan *Interface*

Rancangan *interface* dapat dijelaskan sebagai berikut :

1. Terdapat 1 buah form yang terdiri dari 4 buah tab, yaitu tab *load* data, tab algoritma FCM, tab algoritma FCM-GA, dan tab validitas clustering.
2. Masing-masing tab saling berurutan dan berhubungan antara satu form dengan form lainnya. Hal ini dikarenakan data yang diproses di form tertentu merupakan hasil dari proses di form sebelumnya.

3.4.1 Tab *Load* Data

Tab *load* data digunakan untuk mencari file yang berisi data uji dan ditampilkan dalam *listview*. Antarmuka tab *load* data ditampilkan pada gambar 3.21.



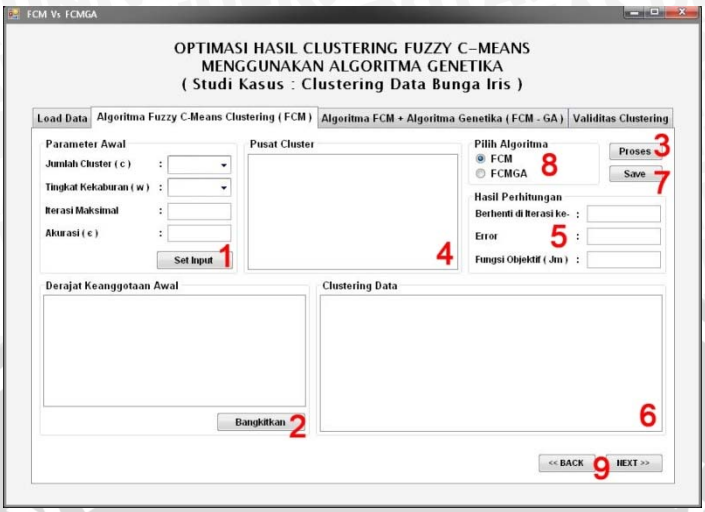
Gambar. 3.21 Tab *Load* Data

Penjelasan gambar 3.21 yaitu sebagai berikut :

- No. 1 berupa *textbox* untuk menampilkan direktori file data input.
- No. 2 berupa *listview* untuk menampilkan data input.
- No. 3 berupa *button* untuk mencari direktori file data input.
- No. 4 berupa *button* untuk menampilkan data input.
- No. 5 berupa *button* untuk berpindah halaman.

3.4.2 Tab Algoritma FCM

Tab algoritma FCM digunakan untuk proses FCM. Antarmuka tab ini ditampilkan pada gambar 3.22.



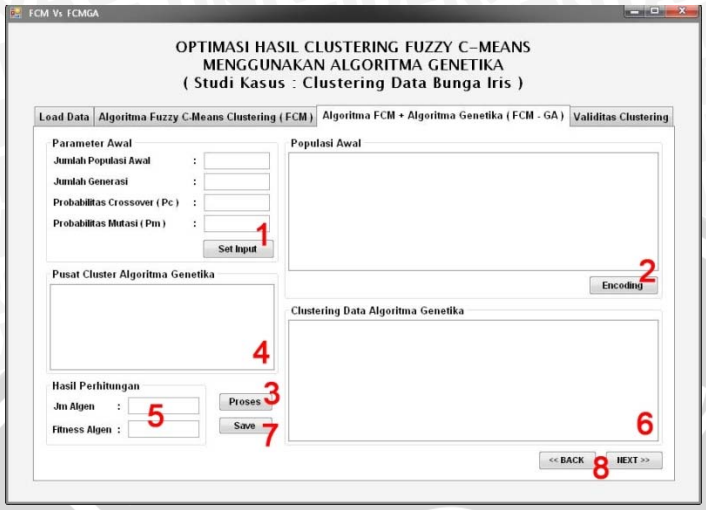
Gambar 3.22 Tab Algoritma FCM

Penjelasan gambar 3.22 yaitu sebagai berikut :

- No. 1 berupa *combobox*, *textbox*, dan *button* untuk mengatur input parameter awal.
- No. 2 berupa *listview* dan *button* untuk menampilkan matrik derajat keangggotaan awal.
- No. 3 berupa *button* untuk menjalankan proses FCM.
- No. 4 berupa *listview* untuk menampilkan pusat cluster.
- No. 5 berupa *textbox* untuk menampilkan nomor iterasi saat berhenti, nilai *error,* dan nilai fungsi objektif Jm.
- No. 6 berupa *listview* untuk menampilkan update matrik derajat keanggotaan.
- No. 7 berupa *button* untuk menyimpan data update matirk derajat keanggotaan ke dalam format excel.
- No. 8 berupa *radiobutton* untuk memilih FCM atau FCM-GA.
- No. 9 berupa *button* untuk berpindah halaman.

3.4.3 Tab Algoritma FCM-GA

Tab algoritma FCM-GA digunakan untuk proses FCM dengan penambahan algoritma genetika. Antarmuka tab ini ditampilkan pada gambar 3.23.



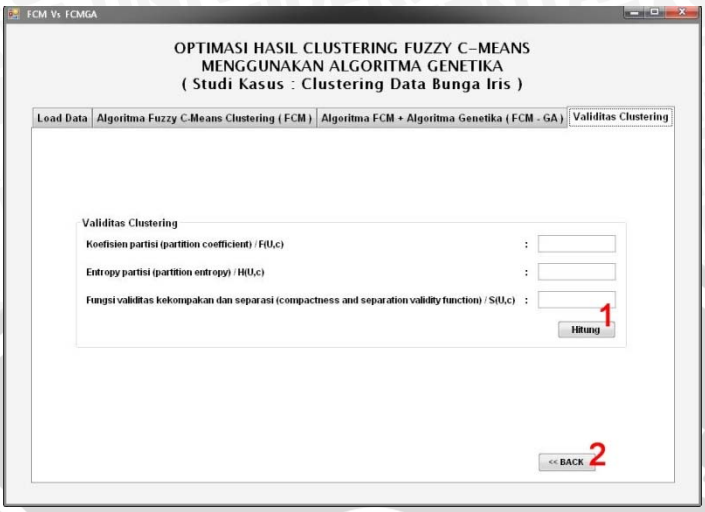
Gambar 3.23 Tab Algoritma FCM-GA

Penjelasan gambar 3.23 yaitu sebagai berikut :

- No. 1 berupa *textbox* dan *button* untuk mengatur input parameter awal.
- No. 2 berupa *listview* dan *button* untuk menampilkan data populasi awal.
- No. 3 berupa *button* untuk menjalankan proses algoritma genetika.
- No. 4 berupa *listview* untuk menampilkan pusat cluster.
- No. 5 berupa *textbox* untuk menampilkan nilai fungsi objektif Jm dan nilai *fitness*.
- No. 6 berupa *listview* untuk menmpilkan update matrik derajat keanggotaan.
- No. 7 berupa *button* untuk menyimpan data update matrik U ke dalam format excel.
- No. 8 berupa *button* untuk berpindah halaman.

3.4.4 Tab Validitas *Clustering*

Tab validitas *clustering* digunakan untuk menampilkan hasil perhitungan validitas *clustering*. Antarmuka tab ini ditampilkan pada gambar 3.24.



Gambar 3.24 Tab Validitas Clustering

Penjelasan gambar 3.20 yaitu sebagai berikut :

- No. 1 berupa *textbox* dan *button* untuk menampilkan nilai FU, HU, dan SU.
- No 2 berupa *button* untuk berpindah halaman.

3.5 Rancangan Uji Coba

Pada rancangan uji coba, akan dihitung nilai Jm, FU, HU, dan SU yang diperoleh dari algoritma Fuzzy C-Means (FCM) *Clustering* dan algoritma Fuzzy C-Means *Clustering* dengan pendekatan algoritma genetika (FCM-GA). Untuk nilai c tertentu, cluster yang baik adalah cluster yang memberikan nilai Jm terkecil dari percobaan yang dilakukan berulang-ulang. Apabila nilai c yang diujicobakan bermacam-macam, akan dicari nilai c terbaik, maka validitas cluster FU, HU, dan SU dapat dijadikan sebagai pedoman. Nilai c yang baik ditinjau dari derajat ke-*fuzzy*-an yaitu apabila nilai FU cukup besar dan HU serta SU cukup kecil.

Tabel 3.13 Rancangan Tabel Nilai Jm FCM dan FCM-GA

Jumlah Cluster (c)	Jm	
	FCM	FCM-GA
2		
3		
4		
5		

Tabel 3.14 Rancangan Tabel Validitas Clustering FCM-GA

c	FCM-GA		
	FU	HU	SU
2			
3			
4			
5			

Data input yang digunakan telah terklasifikasi ke dalam 3 kelas sehingga perhitungan *classification rate* hanya bisa dilakukan terhadap jumlah cluster 3 (c = 3). Pada penelitian ini dilakukan perhitungan *classification rate* sebanyak 4 kali untuk nilai c = 3. Kemudian dihitung nilai rata-ratanya.

Tabel 3.15 Rancangan Tabel *Classification Rate* (c = 3)

Percobaan ke-	FCM (%)	FCM-GA (%)
1		
2		
3		
4		
Rata-Rata		

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Implementasi perangkat lunak ini berupa aplikasi pemrograman yang menerapkan optimasi hasil *clustering* Fuzzy C-Means menggunakan algoritma genetika dengan cara meminimalkan fungsi objektif Jm. Adapun lingkungan implementasi akan dijelaskan ke dalam subbab lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam penerapan algoritma genetika untuk mengoptimasi FCM adalah sebagai berikut :

1. Prosesor Intel® Pentium® Dual CPU T2390 @1.86 GHz.
2. Memory 1 GB RAM
3. Harddisk dengan kapasitas 160 GB
4. Monitor
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat keras yang digunakan dalam penerapan algoritma genetika untuk mengoptimasi FCM adalah sebagai berikut :

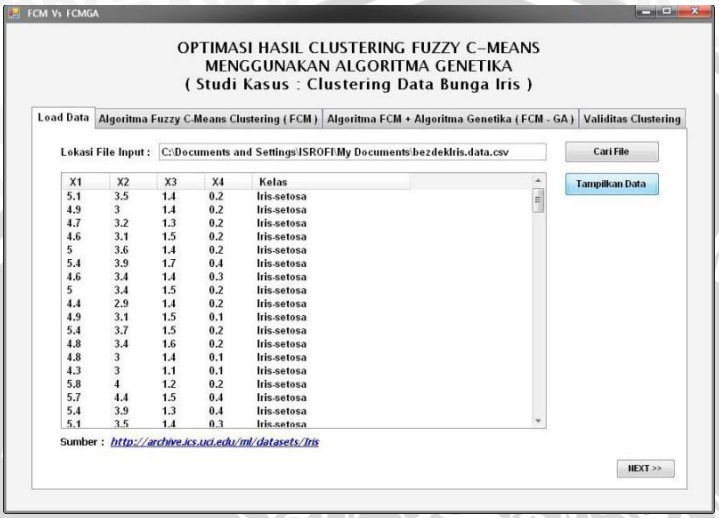
1. Sistem Operasi Microsoft Windows XP SP 2
2. Microsoft Visual C# 2008 Express Edition

4.2 Implementasi Program dan Antarmuka

Pada implementasi program optimasi hasil *clustering* Fuzzy C-Means menggunakan algoritma genetika ini terdapat 4 tahapan utama, yang meliputi *load* data, *clustering* menggunakan FCM, algoritma genetika, dan validitas clustering. Tahapan-tahapan tersebut akan dijelaskan dalam subbab berikut.

4.2.1 *Load* Data

Pada tahap ini dicari file berekstensi .csv yang berisi data input untuk diproses. Data input berupa data bunga Iris yang terdiri dari 150 data dan memiliki 4 atribut. Data ini telah terklasifikasi ke dalam 3 kelas, yaitu Iris Setosa, Iris Versicolor, dan Iris Virginica, masing-masing terdiri dari 50 data. Data bunga Iris bisa dilihat pada lampiran 1. Setelah file ditemukan, maka data bunga Iris akan ditampilkan ke dalam *listview*. Antarmuka dari proses *load* data bisa dilihat pada gambar 4.1.



Gambar 4.1 Antarmuka Proses *Load* Data

Data ini terletak di dalam file berekstensi .csv yang kemudian dimasukkan ke dalam *listview* dalam sistem, namun sebelumnya akan dirubah terlebih dahulu menjadi matrik X. Fungsi untuk merubah data input menjadi matrik X ini dijelaskan pada *source code* 4.1.

```
1.  public void konversiMatrik(string[] dataSplit){
2.  for (int i = 0; i < jumlahBaris; i++){
3.    data = dataSplit[i].Split(',');
4.    for (int j = 0; j < jumlahKolom - 1; j++){
5.      indeksPosisi[i, j] = (data[j]);
6.      dataInput[i, j] = Convert.ToDouble(data[j]);
7.    }
8.    for (int j = jumlahKolom - 1; j < jumlahKolom;
      j++){
9.      indeksPosisi[i, j] = (data[j]);
10.     dataInputString[i, j] = (data[j]);
11.   }
12. }}
```

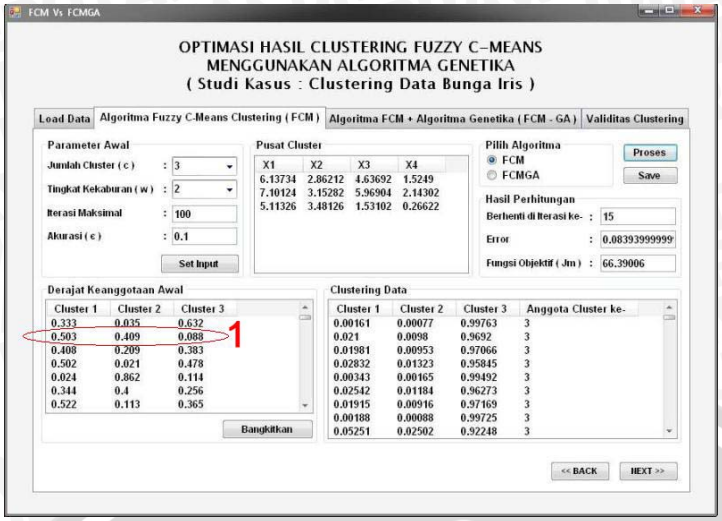
Source code 4.1 Fungsi Konversi Matrik X

Source code 4.1 baris ke-2 hingga baris ke-7 digunakan untuk mengkonversi data input yang berupa angka. Data ini yang nantinya digunakan sebagai data input dalam proses FCM maupun FCM-GA. *Source code* 4.1 baris ke-8 hingga baris ke-11 digunakan untuk menampilkan data input yang berupa huruf. Data ini hanya digunakan sebagai keterangan di *listview*.

4.2.2 *Clustering* Menggunakan FCM

Pada proses *clustering* menggunakan FCM terdapat 4 tahapan utama, yang meliputi : pembangkitan matrik derajat keanggotaan awal (matrik U awal), perhitungan matrik pusat cluster (matirk V), perhitungan fungsi objektif (Jm), dan update matrik derajat keanggotaan (update matrik U). Antarmuka program untuk proses FCM bisa dilihat pada gambar 4.2.

Proses pembangkitan matrik derajat keanggotaan awal dilakukan dengan cara membangkitkan nilai acak dimana jumlah nilai dalam satu baris matrik harus sama dengan 1, seperti terlihat dalam lingkaran berwarna merah bernomor 1 pada gambar 4.2. Proses pembangkitan matrik derajat keanggotaan awal bisa dilihat pada *source code* 4.2.



Gambar 4.2 Antarmuka Proses FCM

```
1.  public void matrikDerajatKeanggotaanAwal(int
    baris){
2.  for (int i = 0; i < jumlahBaris; i++){
3.    double jumlah = 0;
4.    for (int k = 0; k < jumlahCluster; k++){
5.      matrikU[i, k] = 0 + acak.Next();
6.      jumlah += matrikU[i, k];
7.    }
8.    for (int k = 0; k < jumlahCluster; k++){
9.      matrikU[i, k] /= jumlah;
10. }}}
```

Source code 4.2 Fungsi Pembangkitan Matrik U Awal

Setelah didapatkan matrik derajat keanggotaan awal, maka dilakukan perhitungan matrik pusat cluster. Fungsi untuk menghitung matrik pusat cluster dijelaskan dalam *source code* 4.3.

```
1.  public void matrikPusatCluster(double w){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int j = 0; j < jumlahKolom - 1; j++){
4.      for (int i = 0; i < jumlahBaris; i++){
5.        atas[k, j] += Math.Pow(matrikU[i, k], w) *
          dataInput[i, j];
6.  }}}
7.  for (int k = 0; k < jumlahCluster; k++){
8.    for (int i = 0; i < jumlahBaris; i++){
9.      bawah[i, k] = Math.Pow(matrikU[i, k], w);
10.     totalBawah[k] += bawah[i, k];
11. }}
12. for (int k = 0; k < jumlahCluster; k++){
13.   for (int j = 0; j < jumlahKolom - 1; j++){
14.     pusatCluster[k, j] = atas[k, j] /
        totalBawah[k];
15. }}}
```

Source code 4.3 Fungsi Perhitungan Matrik Pusat Cluster

Proses berikutnya yaitu perhitungan fungsi objektif Jm yang bisa dilihat pada *source code* 4.4.

```
1.  public void fungsiObjektif(double w){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      for (int j = 0; j < jumlahKolom - 1; j++){
5.        Jm1[i, k] += (Math.Pow(dataInput[i, j] -
          pusatCluster[k, j], 2));
6.  }}}
7.  for (int i = 0; i < jumlahBaris; i++){
8.    for (int k = 0; k < jumlahCluster; k++){
9.      Jm2[i, k] = Jm1[i,k] * (Math.Pow(matrikU[i,
        k], w));
10.     Jm += Jm2[i, k];
11. }}}
```

Source code 4.4 Fungsi Perhitungan Jm

Proses selanjutnya yaitu update matrik derajat keanggotaan yang dihitung untuk menggantikan matrik derajat keanggotaan awal. Hasil update matrik derajat keanggotaan ini digunakan untuk menentukan anggota cluster dari masing-masing data. Fungsi update matrik derajat keanggotaan dijelaskan pada *source code* 4.5.

Sedangkan penentuan anggota cluster dijelaskan pada *source code* 4.6.

```
1.  public void updateMatrikDerajatKeanggotaan(double
    w){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      for (int j = 0; j < jumlahKolom - 1; j++){
5.        atas[i, k] += (Math.Pow((dataInput[i, j] -
          pusatCluster[k, j]), 2));
6.  }}}
7.  for (int i = 0; i < jumlahBaris; i++){
8.    for (int k = 0; k < jumlahCluster; k++){
9.      double pangkat = (-1) / (w - 1);
10.     atas[i, k] = Math.Pow((atas[i, k]), pangkat);
11.     bawah[i] += atas[i, k];
12. }}
13. for (int i = 0; i < jumlahBaris; i++){
14.   for (int k = 0; k < jumlahCluster; k++){
15.     matrikU[i, k] = atas[i, k] / bawah[i];
16. }}}
```

Source code 4.5 Fungsi Perhitungan Update Matrik U

```
1.  public void anggotaCluster(){
2.  for (int i = 0; i < matrikU.GetLength(0); i++){
3.    letak[i] = 1;
4.    anggota[i] = matrikU[i, 0];
5.    for (int j = 0; j < matrikU.GetLength(1); j++){
6.      if (matrikU[i, j] > anggota[i]){
7.        anggota[i] = matrikU[i, j];
8.        letak[i] = j + 1;
9.  }}}}
```

Source code 4.6 Fungsi Penentuan Anggota Cluster

Proses FCM dilakukan berulang-ulang hingga memenuhi kondisi berhenti. Untuk memeriksa kondisi ini maka diperlukan fungsi cek kondisi berhenti yang dijelaskan pada *source code* 4.7.

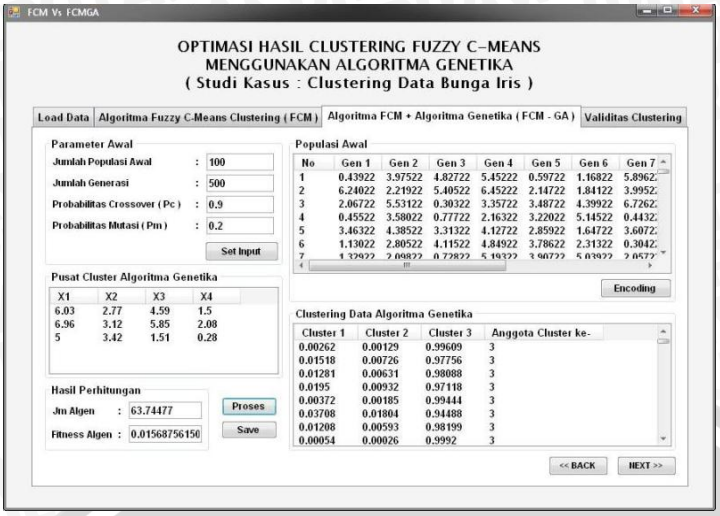
```
1.  public void cekKondisiBerhenti(){
2.  for (iterasi = 1; iterasi <= maxIterasi; iterasi++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      for (int k = 0; k < jumlahCluster; k++){
5.        matrikU_temp[i, k] = matrikU[i, k];
6.  }}
7.    matrikPusatCluster(tingkatKekaburan);
8.    fungsiObjektif(tingkatKekaburan);
9.    updateMatrikDerajatKeanggotaan(tingkatKekaburan);
10.   anggotaCluster();
11.   error = Math.Abs(Jm - JmSebelum);
12.   Jm_FCM = Jm;
13.   JmSebelum = Jm;
14.   Jm = 0;
15.   if (error < akurasi){
16.     break;
17. }}}
```

Source code 4.7 Fungsi Iterasi dan Cek Kondisi Berhenti FCM

Source code 4.7 Baris ke-2 digunakan untuk perulangan FCM mulai iterasi = 1 hingga mencapai maksimum iterasi. *Source code* 4.7 baris ke-3 hingga baris ke-5 digunakan untuk menyimpan nilai matrik U yang nantinya digunakan untuk perhitungan Jm pada proses algoritma genetika. *Source code* 4.7 baris ke-15 hingga baris ke-16 digunakan untuk memeriksa kondisi berhenti dari iterasi FCM.

4.2.3 Algoritma Genetika

Pada proses algoritma genetika, terdapat 4 tahap utama, yaitu encoding, seleksi, crossover, dan mutasi. Antarmuka program untuk proses algoritma genetika bisa dilihat pada gambar 4.3.



Gambar 4.3 Antarmuka Proses Algoritma Genetika

Pada proses algoritma genetika, tahap pertama yaitu proses *encoding*. Metode *encoding* yang digunakan yaitu *real encoding*, membangkitkan kromosom populasi awal melalui pembangkitan bilangan acak dengan interval tertentu. Di sini interval yang digunakan yaitu nilai minimum dan maksimum dari matrik pusat cluster hasil algoritma FCM. Proses *encoding* ini dijelaskan pada *source code* 4.8.

Setelah proses encoding selesai maka perlu dievaluasi, yaitu mencari nilai fitness untuk masiang-masing kromosom. Fungsi untuk evaluasi dijelaskan pada *source code* 4.9.

Kromosom hasil proses *encoding* diseleksi untuk mendapatkan kromosom-kromosom induk yang terbaik. Proses seleksi dijelaskan pada *source code* 4.10.

Kromosom hasil proses seleksi di-*crossover* untuk mendapatkan keragaman individu dan menghasilkan anak yang diharapkan lebih baik dari induknya. Metode yang digunakan adalah *one-point crossover*. Proses *crossover* dijelaskan pada *source code* 4.11.

Kromosom hasil crossover digunakan sebagai kandidat induk untuk proses mutasi. Metode mutasi yang digunakan adalah *nonuniform mutation.* Proses mutasi dijelaskan pada source code 4.12.

```
1.  public void encoding(){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int j = 0; j < jumlahKolom - 1; j++){
4.      pusatCluster_temp.Add(pusatCluster[k, j]);
5.  }}
6.  batasAtasInitPop = pusatCluster_temp.Max();
7.  batasBawahInitPop = pusatCluster_temp.Min();
8.  for (int i = 0; i < jumlahIndividu; i++){
9.    for (int krom = 0; krom < panjangKromosom;
      krom++){
10.     int rangeInitPop =
        Convert.ToInt32(batasAtasInitPop -
        batasBawahInitPop) * 1000;
11.     kromosom[i, krom] = rand.Next(0,
        rangeInitPop);
12.     kromosom[i, krom] = kromosom[i, krom] / 1000;
13.     kromosom[i, krom] = kromosom[i, krom] +
        batasBawahInitPop;
14. }}}
```

Source code 4.8 Fungsi *Encoding*

Source code 4.8 baris ke-6 digunakan untuk menghitung batas atas nilai random inisialisasi kromosom awal. *Source code* 4.8 baris ke-7 digunakan untuk menghitung batas bawah nilai random inisialisasi kromosom awal. Pembangkitan nilai untuk kromosom awal ditunjukkan pada *source code* 4.8 baris ke-8 hingga baris ke-13.

```
1.  public void evaluasi(){
2.  for (int i = 0; i < jumlahIndividu; i++){
3.    for (int krom = 0; krom < panjangKromosom;
      krom++){
4.      encoding_temp[krom] = kromosom[i, krom];}
5.    int count = 0;
6.    for (int k = 0; k < jumlahCluster; k++){
7.      for (int j = 0; j < jumlahKolom - 1; j++){
8.        pusatCluster[k, j] = encoding_temp[count];
9.        count++;
10.   }}
11.   hitungJm();
12.   popBaru.Add(new populasiBaru() { kromosomBaru =
      (double[])encoding_temp.Clone(),
      JmBaru=JmSebelum});
13.   fitness = (1 / JmSebelum);
14.   fitnessTotal = fitnessTotal + fitness;
15. }}
```

Source code 4.9 Fungsi Evaluasi

Source code 4.9 baris ke-11 merupakan pemanggilan fungsi untuk menghitung nilai Jm. *Source code* 4.9 baris ke-13 digunakan untuk menghitung nilai *fitness*. *Source code* 4.9 baris ke-14 digunakan untuk menghitung jumlah *fitness* total.

```
1.  public void seleksi(){
2.  for (int i = 0; i < jumlahIndividu; i++){
3.    peluang[i] = fitness / fitnessTotal;
4.    peluang[i] = Math.Round(peluang[i], 3);
5.    peluang[i] = peluang[i] * 1000;
6.    peluangKum = peluangKum + peluang[i];
7.    peluangKumulatif[i] = peluangKum;
8.  }
9.  int jumlahInduk_temp = jumlahIndividu / 2;
10. if (jumlahInduk_temp % 2 == 0){
11.   jumlahInduk = jumlahInduk_temp;
12. }
13. else{
14.   jumlahInduk = jumlahInduk_temp + 1;
15. }
16. for (int i = 0; i < jumlahInduk; i++){
17.   bilAcak[i] = rand.Next(1, 1000);
18.   for (int j = 0; j < jumlahIndividu; j++){
19.     if (bilAcak[i] <= peluangKumulatif[j]){
20.       indukCrossover[i] = j;
21.       break;
22. }}}}
```

Source code 4.10 Fungsi Seleksi

Proses penyeleksian kromosom ditunjukkan pada *source code* 4.10 baris ke-19 hingga baris ke-21. Jika bilangan acak yang dibangkitkan kurang dari atau sama dengan peluang kumulatif, maka kromosom terpilih sebagai induk *crossover*.

```
1.  public void crossover(){
2.  jumlahCrossover = jumlahInduk / 2;
3.  for (int i = 0; i < jumlahCrossover; i++){
4.    int titikCrossover=rand.Next(1,panjangKromosom-1);
5.    bilAcak2[i] = rand.Next(100);
6.    bilAcak2[i] = bilAcak2[i] / 100;
7.    int parent1 = indukCrossover[i * 2];
8.    int parent2 = indukCrossover[(i * 2) + 1];
9.    if (bilAcak2[i] < Pc){
10.     for (int krom=0;krom < panjangKromosom;krom++){
```

```
11.         if (krom < titikCrossover){
12.           child1[i, krom] = kromosom[parent1, krom];
13.           child2[i, krom] = kromosom[parent2, krom];
14.         }
15.         else{
16.           child1[i, krom] = kromosom[parent2, krom];
17.           child2[i, krom] = kromosom[parent1, krom];
18.         }
19.         indukMutasi[(i * 2), krom] = child1[i, krom];
20.         indukMutasi[((i*2)+1),krom] = child2[i, krom];
21.   }}
22.     else{
23.       for (int krom=0;krom < panjangKromosom;krom++){
24.         child1[i, krom] = kromosom[parent1, krom];
25.         child2[i, krom] = kromosom[parent2, krom];
26.         indukMutasi[(i * 2), krom] = child1[i, krom];
27.         indukMutasi[((i*2)+1),krom] = child2[i, krom];
28.   }}}}
```

Source code 4.11 Fungsi *Crossover*

Proses *one-cut point crossover* ditunjukkan pada *source code* 4.11 baris ke-10 hingga baris ke-20. Pada proses *one-cut point crossover* ini, nilai sebagian kromosom induk pertama akan ditukar dengan nilai sebagian kromosom induk kedua.

```
1.  public void mutasi(){
2.  int b = 2;
3.  for (int i = 0; i < jumlahInduk; i++){
4.    bilAcak3[i] = bilAcak3[i] / 100;
5.    if (bilAcak3[i] < Pm){
6.      for (int krom=0;krom<panjangKromosom;krom++){
7.       indukMutasi_temp[i].Add(indukMutasi[i,krom]);
8.       if (krom == titikMutasi){
9.         xk = indukMutasi[i, krom];
10. }}
11.     xkAtas = indukMutasi_temp[i].Max();
12.     xkBawah = indukMutasi_temp[i].Min();
13.     if (cekRumus == 0){
14.       xkMutasi = xk + ((xkAtas - xk) * r *
          Math.Pow((1 - (nomorGenerasi /
          jumlahGenerasi)), b));
15.       xkMutasi = Math.Round(xkMutasi, 2);
16.     }
17.     else{
18.       xkMutasi = xk - ((xk - xkBawah) * r *
```

```
        Math.Pow((1 - (nomorGenerasi /
        jumlahGenerasi)), b));
19.     xkMutasi = Math.Round(xkMutasi, 2);
20.   }
21.   for (int krom=0;krom<panjangKromosom;krom++){
22.     if (krom == titikMutasi){
23.       indukMutasi[i, krom] = xkMutasi;
24. }}}}
25. for (int i = 0; i < jumlahInduk; i++){
26.   for (int krom=0; krom < panjangKromosom;krom++){
27.     mutasi_temp[krom] = indukMutasi[i, krom];
28.   }
29.   int count = 0;
30.   for (int k = 0; k < jumlahCluster; k++){
31.     for (int j = 0; j < jumlahKolom - 1; j++){
32.       pusatCluster[k, j] = mutasi_temp[count];
33.       count++;
34. }}
35.   hitungJm();
36.   popBaru.Add(new populasiBaru() {
37.   kromosomBaru =
     (double[])mutasi_temp.Clone(),JmBaru=JmSebelum});
38. }
39. var popBr = from pB in popBaru orderby pB.JmBaru
    select pB
40. List<populasiBaru> list =
    popBr.ToList<populasiBaru>();
41. for (int i = 0; i < jumlahIndividu; i++){
42.   for (int krom = 0;krom<panjangKromosom;krom++){
43.     kromosom[i, krom] =
        list[i].kromosomBaru[krom];
44.     kromosom_Algen[krom] =
        list[0].kromosomBaru[krom];
45.   }
46.   Jm_Algen = list[0].JmBaru;
47.   fitness_Algen = 1 / Jm_Algen;
48. }}
```

Source code 4.12 Fungsi Mutasi

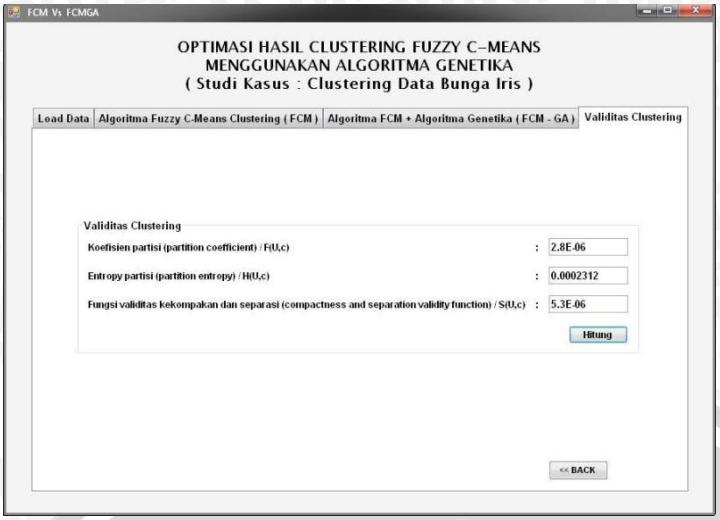
Proses *nonuniform mutation* ditunjukkan pada *source code* 4.12 baris ke-3 hingga baris ke-23. Setelah semua tahapan proses dilalui, maka iterasi dilakukan hingga mencapai jumlah generasi maksimum. Proses iterasi ini disebut regenerasi, dijelaskan pada *source code* 4.13.

```
1.  public void regenerasi(){
2.  nomorGenerasi = 1;
3.  for (int i = nomorGenerasi;i<=jumlahGenerasi;i++){
4.    evaluasi();
5.    seleksi();
6.    crossover();
7.    mutasi();
8.    peluangKum = 0;
9.    fitnessTotal = 0;
10.   popBaru.Clear();
11.   nomorGenerasi++;
12. }
13. int count = 0;
14. for (int k = 0; k < jumlahCluster; k++){
15.   for (int j = 0; j < jumlahKolom - 1; j++){
16.     pusatCluster[k, j] = kromosom_Algen[count];
17.     count++;
18. }}
19. updateMatrikDerajatKeanggotaan(tingkatKekaburan);
20. anggotaCluster();}
```

Source code 4.13 Fungsi Regenerasi

4.2.4 Validitas *Clustering*

Pada proses FCM maupun FCM-GA, dihasilkan matrik pusat cluster, fungsi objektif Jm, dan matrik derajat keanggotaan. Setelah masing-masing proses selesai, dicari nilai validitas clustering. Validitas clustering digunakan untuk menentukan nilai c (jumlah cluster) terbaik dari beberapa nilai c yang diujicobakan. Antarmuka program dari perhitungan nilai validitas clustering bisa dilihat pada gambar 4.4.



Gambar 4.4 Antarmuka Validitas Clustering

Adapun proses perhitungan validitas clustering yang terdiri dari koefisien partisi, entropy partisi, serta fungsi validitas kekompakan dan separasi, dijelaskan pada *source code* 4.14, *source code* 4.15, dan *source code* 4.16.

```
1.  public void koefisienPartisi(){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      FUc1[i, k] += (Math.Pow(matrikU[i, k], 2));
5.      FUc2 = FUc1[i, k] / jumlahBaris;
6.  }}}
```

Source code 4.14 Fungsi Koefisien Partisi

```
1.  public void entropyPartisi(){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      HUc1[i, k] += matrikU[i, k] *
        Math.Log10(matrikU[i, k]);
5.      HUc2 = (-1) * HUc1[i, k] / jumlahBaris;
6.  }}}
```

Source code 4.15 Fungsi Entropy Partisi

```
1.  public void fungsiValiditasKekompakanDanSeparasi(){
2.  for (int k = 0; k < jumlahCluster; k++){
3.    for (int i = 0; i < jumlahBaris; i++){
4.      for (int j = 0; j < jumlahKolom - 1; j++){
5.        jarak[i, k] += Math.Pow((dataInput[i, j] -
          pusatCluster[k, j]), 2);
6.        jarak2[i, k] += (Math.Sqrt(jarak[i, k])) *
          (Math.Pow(matrikU[i, k], 2));
7.        jarakSampelDenganPrototype = jarak2[i, k];
8.  }}}
9.  for (int k1 = 0; k1 < jumlahCluster; k1++){
10.   for (int k2 = k1 + 1; k2 < jumlahCluster; k2++){
11.     for (int j = 0; j < jumlahKolom - 1; j++){
12.       jarak3[k1, k2] +=
          Math.Pow((pusatCluster[k1,j] -
          pusatCluster[k2, j]), 2);
13.       jarak3Total[k1] = Math.Sqrt(jarak3[k1, k2]);
14.       jarakMinimumAntarPrototype =
          jarak3Total[k1];
15.       if (jarak3Total[k1] <
            jarakMinimumAntarPrototype &&
            jarakMinimumAntarPrototype != 0){
16.         jarakMinimumAntarPrototype =
            jarak3Total[k1];
17. }}}}
18. SUc = (jarakSampelDenganPrototype / jumlahBaris) /
    jarakMinimumAntarPrototype;}
```

Source code 4.16 Fungsi Validitas Kekompakan dan Separasi

4.3 Implementasi Uji Coba dan Analisa Hasil

Uji coba dilakukan pada data input berupa dataset bunga iris sebanyak 150 data dan memiliki 4 atribut. Data ini telah terklasifikasikan ke dalam 3 kelas, masing-masing sejumlah 50 data. Uji coba proses FCM digunakan parameter input yaitu sebagai berikut.

a. Tingkat kekaburan (*fuzziness*) = 2
b. Batas iterasi = 100
c. Batas akurasi ditentukan melalui uji coba 4 macam nilai, yaitu 0.1, 0.01, 0.001, dan 0.0001. Nilai yang digunakan adalah nilai yang menghasilkan Jm terbaik (paling kecil).

Sedangkan parameter input yang digunakan untuk algoritma genetika yaitu sebagai berikut.

a. Jumlah individu populasi awal = 100
b. Jumlah generasi = 500
c. Probabilitas *crossover* (Pc) = 0.9
d. Probabilitas mutasi (Pm) = 0.2

Uji coba dilakukan untuk 4 jenis jumlah cluster, yaitu 2, 3, 4, dan 5.

Untuk menentukan batas akurasi proses FCM yang akan digunakan dalam uji coba penelitian, maka dilakukan uji coba empat macam nilai batas akurasi, yaitu 0.1, 0.01, 0.001, dan 0.0001 pada jumlah cluster 2, 3, 4, dan 5.

Tabel 4.1 Uji Coba Nilai Batas Akurasi FCM

Jumlah Cluster (c)	Batas Akurasi	Jm FCM
2	0.1	133.07363
	0.01	133.06950
	0.001	133.06950
	0.0001	133.06950
3	0.1	66.37609
	0.01	66.53204
	0.001	66.53204
	0.0001	66.53204
4	0.1	49.77473
	0.01	49.87696
	0.001	49.89224
	0.0001	56.20381
5	0.1	36.31649
	0.01	39.10453
	0.001	39.12002
	0.0001	41.05311

Tabel 4.1 menunjukkan bahwa untuk nilai c = 2, Jm FCM terkecil diperoleh pada nilai 133.06950 melalui batas akurasi 0.01, 0.001 dan 0.0001. Untuk nilai c = 3, Jm FCM terkecil diperoleh pada nilai 66.37609 melalui batas akurasi 0.1.Untuk nilai c = 4, Jm FCM

terkecil diperoleh pada nilai 49.77473 melalui batas akurasi 0.1. untuk nilai c = 5, Jm FCM terkecil diperoleh pada nilai 36.31649 melalui batas akurasi 0.1.

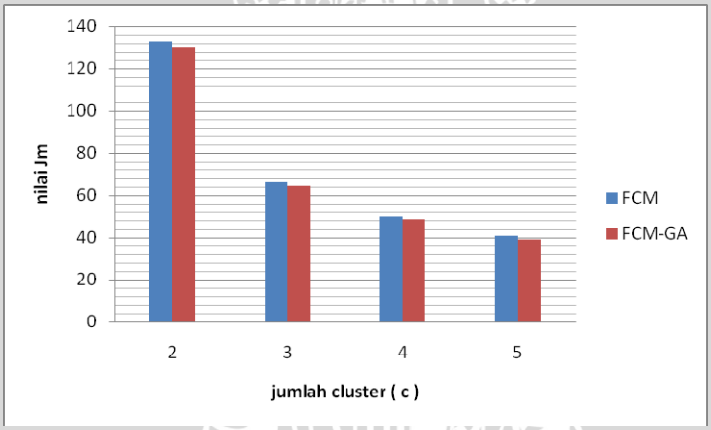
Nilai batas akurasi yang mendominasi untuk menghasilkan Jm FCM terkecil yaitu 0.1. Nilai ini ditunjukkan pada tabel 4.1 untuk jumlah cluster 3, 4, dan 5. Dengan demikian, maka nilai batas akurasi yang akan digunakan pada penelitian ini adalah 0.1.

Dari hasil uji coba program diperoleh nilai fungsi objektif (Jm) untuk jumlah cluster 2, 3, 4, dan 5 seperti terlihat pada tabel 4.1.

Tabel 4.2 Nilai Jm FCM dan FCM-GA

Jumlah Cluster (c)	Jm	
	FCM	FCM-GA
2	133.07387	130.41351
3	66.35933	64.31244
4	49.76890	48.62854
5	40.94172	39.31862

Dari tabel 4.1 bisa dibuat grafik nilai Jm FCM dan Jm FCM-GA untuk nilai c = 2, c = 3, c = 4, dan c = 5 seperti terlihat pada gambar 4.5.



Gambar 4.5 Grafik Nilai Jm FCM dan FCM-GA

Dari tabel 4.1 dan gambar 4.5 terlihat bahwa nilai Jm sebagai fungsi objektif yang diminimumkan pada FCM-GA lebih kecil atau

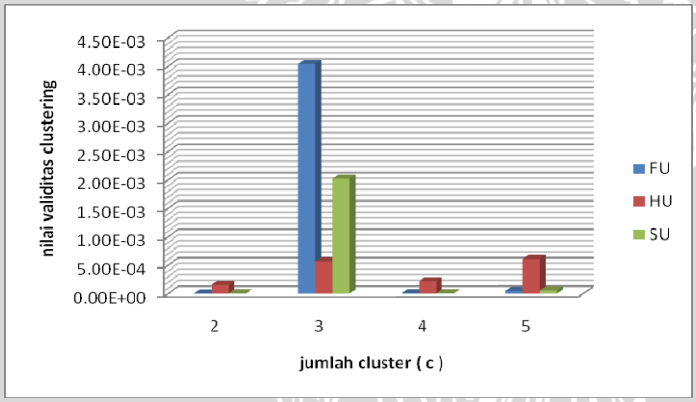
lebih minimum dari nilai Jm FCM untuk semua nilai c (jumlah cluster) yang diujicobakan. Hal ini menunjukkan bahwa pusat cluster yang dihasilkan oleh algoritma FCM-GA lebih baik dari algoritma FCM. Dengan demikian jarak antara data dan pusat cluster semakin kecil atau semakin dekat untuk masing-masing cluster.

Pengujian ini menggunakan 4 jenis c (jumlah cluster). Untuk menentukan jumlah cluster yang terbaik dari keempat jumlah cluster yang diujicobakan, maka nilai validitas clustering dijadikan acuan. Nilai c yang baik ditinjau dari derajat kekaburan (*fuzziness*) yaitu apabila FU besar, dan HU serta SU kecil. Untuk menentukan nilai c (jumlah cluster) terbaik dari 4 nilai c yang diujicobakan pada *clustering* data FCM-GA, maka bisa dilihat nilai validitas *clustering* FCM-GA pada tabel 4.3.

Tabel 4.3 Validitas Clustering FCM-GA

c	FCM-GA		
	FU	HU	SU
2	9E-07	0.0001505	2.2E-06
3	0.0040368	0.0005651	0.0020208
4	2.4E-06	0.0002173	4.9E-06
5	4.85E-05	0.0006078	5.63E-05

Dari tabel 4.3 dibuat grafik validitas clustering untuk proses FCM-GA yang bisa dilihat pada gambar 4.6.



Gambar 4.6 Grafik Validitas Clustering FCM-GA

Pada gambar 4.6 terlihat bahwa jumlah cluster terbaik untuk proses FCM-GA yaitu nilai c = 3 karena nilai FU lebih besar dari nilai HU dan SU. Hal ini sesuai dengan dataset yang menjadi input dalam pengujian karena dataset telah terklasifikasi menjadi 3 kelas, seperti terlihat pada data bunga Iris di lampiran 1.

Pada perhitungan *classification rate* digunakan c = 3 untuk menghitung berapa banyak data yang dikelompokkan oleh sistem secara benar. Nilai ini merupakan hasil dari rasio antara jumlah data yang anggota clusternya terklasifikasi secara benar terhadap jumlah data keseluruhan.

Karena data uji telah terklasifikasikan ke dalam 3 kelas, yaitu Iris Setosa, Iris Versicolor, dan Iris Virginica, maka perhitungan *classification rate* untuk FCM dan FCM-GA dilakukan pada nilai c (jumlah cluster) = 3. Percobaan dilakukan sebanyak 4 kali pada nilai c yang sama, yaitu c = 3, kemudian dihitung rata-ratanya. Hasil perhitungan *classification rate* bisa dilihat pada tabel 4.4.

Tabel 4.4 *Classification Rate* untuk c = 3

Percobaan ke-	FCM (%)	FCM-GA (%)
1	87.33	89.33
2	87.33	90.00
3	87.33	90.00
4	87.33	90.00
Rata-Rata	87.33	89.83

Berdasarkan tabel 4.4 terlihat bahwa nilai rata-rata *classification rate* FCM-GA yaitu 89.83%, lebih besar dari *classification rate* FCM yang besarnya 87.33%. Nilai ini menunjukkan bahwa *cluster* yang dihasilkan oleh FCM-GA lebih baik daripada *cluster* yang dihasilkan oleh FCM.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian dan analisa yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut :

1. Algoritma genetika telah berhasil diterapkan untuk optimasi hasil *clustering* Fuzzy C-Means dengan cara meminimalkan fungsi objektif Jm. Matrik pusat cluster hasil proses FCM dievolusikan dengan menggunakan algoritma genetika. Selanjutnya nilai Jm dapat dihitung dari matrik pusat cluster hasil optimasi yang telah diperoleh.
2. Hasil *clustering* yang didapatkan dari proses FCM-GA lebih baik daripada hasil *clustering* yang didapatkan dari proses FCM. Hal ini terlihat dari nilai Jm FCM-GA yang lebih kecil dari nilai Jm FCM untuk semua nilai c (jumlah cluster) yang diujicobakan. Selain itu juga ditunjukkan oleh rata-rata nilai *classification rate* (CR) FCM-GA (89.83%) yang lebih besar dari rata-rata nilai CR FCM (87.33%).
3. Penentuan jumlah cluster terbaik ditentukan dari nilai FU yang lebih besar daripada nilai HU dan SU. Nilai c (jumlah cluster) terbaik untuk *clustering* data bunga Iris pada proses FCM-GA berdasarkan FU, HU, dan SU pada validitas clustering yaitu c = 3. Nilai c terbaik yang didapatkan sesuai dengan *dataset* yang menjadi data uji pada penelitian ini karena *dataset* telah terklasifikasi ke dalam 3 kelas.

5.2 Saran

Adapun saran untuk penelitian di bidang yang sama yaitu sebagai berikut :

1. Pada penelitian ini, nilai *classification rate* yang didapatkan untuk proses FCM dan FCM-GA cenderung sama. Hal ini dikarenakan perhitungan dilakukan pada 1 nilai c (jumlah cluster), yaitu c = 3. Sebaiknya dilakukan perhitungan nilai *classification rate* untuk nilai c yang berbeda-beda agar nilai *classification rate* yang diperoleh bisa lebih beragam.
2. Nilai *classification rate* hasil optimasi *clustering* pada penelitian ini belum mencapai 100%. Hal ini bisa disebabkan oleh nilai

input parameter algoritma genetika yang digunakan, antara lain : jumlah individu populasi awal, jumlah generasi, probabilitas crossover (Pc), dan probabilitas mutasi (Pm). Untuk mendapatkan nilai *classification rate* yang lebih besar diperlukan nilai input parameter algoritma genetika yang lebih baik.

DAFTAR PUSTAKA

Alata, M., M. Molhim, and A. Ramini. 2008. *Optimizing of Fuzzy C-Means Clustering Algorithm Using GA*. World Academy of Science, Engineering and Technology.

Belanche, L. and Nebot, A. 2002. *Inteligent Data Analysis and Data Mining*. Wright State University.

Bronson, R. 1982. *Theory and Problem of Operation Research*. McGraw Hill Inc. USA.

Chi, Z., Yan, H., and Pham, T. 1996. *Fuzzy Algorithms: With Application to Image Processing and Pattern Recognition, Advance in Fuzzy System-Application and Theory, Vol. 10.* Word Scientific. Singapore.

Cinantya, Celia. 2010. *Implementasi Algoritma Genetika Untuk Penyelesaian Capacitated Vehicle Routing With Time Windows*. Universitas Brawijaya. Malang.

Dakka, F. 2009. *Optimasi Biaya Produksi Menggunakan Algoritma Genetika*. Universitas Brawijaya. Malang.

Gelley, N. and Jang, R. 2000. *Fuzzy Logic Toolbox*. Mathwork, Inc. USA.

Gen, M. and Cheng, R. 1997. *Genetic Algorithms and Engineering Design*. John Wiley and Sons. New York.

Gen, M. and Cheng, R. 2000. *Genetic Algorithms and Engineering Optimization*. John Wiley and Sons. New York.

Hamzah, A. 2001. *Pengenalan Pola dengan Fuzzy Clustering ACADEMIA ISTA, Vol.4.No.1*. Lembaga Penelitian, Institut Sains dan Teknologi AKPRIND. Yogyakarta.

Juniawati. 2003. *Optimasi Proses Pengolahan Mi Jagung Instan Berdasarkan Kajian Preferensi Konsumen. Skripsi* Departemen Teknologi Pangan dan Gizi, Fakultas Teknologi Pertanian, Institut Pertanian Bogor. Bogor.

Kadir, A. 2010. *Identifikasi Tiga Jenis Bunga Iris Menggunakan ANFIS.* Program Pascasarjana Teknik Elektro, Universitas Gadjah Mada. Yogyakarta.

Khoiruddin, A.A. 2007. *Menentukan Nilai Akhir Kuliah dengan Fuzzy C-Means*. Fakultas Teknologi Industri, Universitas Islam Indonesia.

Kusumadewi, S. dan Purnomo, H. 2005. *Penyelesaian Masalah Optimasi dengan Teknik-Teknik Heuristik*. Graha Ilmu. Yogyakarta.

Luthfi, E.T. 2007. *Fuzzy C-Means untuk Clustering Data (Studi Kasus : Data Performance Mengajar Dosen)*. STMIK AMICOM. Yogyakarta.

Maarif, M. S., Machfud, dan M. Sukron. 1989. *Teknik Optimasi Rekayasa Proses Pangan*. PAU-Pangan dan Gizi IPB. Bogor.

Mawaddah, N. K. 2006. *Optimasi Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika*. Universitas Brawijaya. Malang.

Novitasari, A. 2007. *Pemecahan Permasalahan Pencarian Rute Terpendek untuk Jasa Antar Jemput Penumpang Menggunakan Algoritma Genetika*. Universitas Brawijaya. Malang.

Simpson, J.J. *et al*. 2000. *An Improved Hybrid Clustering Algorithm for Natural Scenes*. IEEE Trans. on Geoscience and Remote Sensing, 38(2).

Suyanto. 2005. *Algoritma Genetika dalam Matlab*. Penerbit Andi. Yogyakarta.

Wardani, I.I.P. 2010. *Analisa Keluarga Miskin dengan Menggunakan MetodeFuzzy C-Means Clustering*. Politeknik Elektronika Negeri Surabaya. Surabaya.

Widyastuti, N. dan Hamzah, A. 2007. *Penggunaan Algoritma Genetika dalam Peningkatan Kinerja Fuzzy Clustering untuk Pengenalan Pola*. Institut Sains dan Teknologi AKPRIND.Yogyakarta.

Zimmerman, H.J. 1991. *Fuzzy Set Theory and Its Applications, 2nd Ed.* Norwel, Massachusetts.

LAMPIRAN

Lampiran 1. *Dataset* Bunga Iris

No.	X1	X2	X3	X4	Kelas
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa
25	4.8	3.4	1.9	0.2	Iris-setosa
26	5	3	1.6	0.2	Iris-setosa
27	5	3.4	1.6	0.4	Iris-setosa
28	5.2	3.5	1.5	0.2	Iris-setosa
29	5.2	3.4	1.4	0.2	Iris-setosa
30	4.7	3.2	1.6	0.2	Iris-setosa
31	4.8	3.1	1.6	0.2	Iris-setosa
32	5.4	3.4	1.5	0.4	Iris-setosa
33	5.2	4.1	1.5	0.1	Iris-setosa
34	5.5	4.2	1.4	0.2	Iris-setosa
35	4.9	3.1	1.5	0.2	Iris-setosa
36	5	3.2	1.2	0.2	Iris-setosa

37	5.5	3.5	1.3	0.2	Iris-setosa
38	4.9	3.6	1.4	0.1	Iris-setosa
39	4.4	3	1.3	0.2	Iris-setosa
40	5.1	3.4	1.5	0.2	Iris-setosa
41	5	3.5	1.3	0.3	Iris-setosa
42	4.5	2.3	1.3	0.3	Iris-setosa
43	4.4	3.2	1.3	0.2	Iris-setosa
44	5	3.5	1.6	0.6	Iris-setosa
45	5.1	3.8	1.9	0.4	Iris-setosa
46	4.8	3	1.4	0.3	Iris-setosa
47	5.1	3.8	1.6	0.2	Iris-setosa
48	4.6	3.2	1.4	0.2	Iris-setosa
49	5.3	3.7	1.5	0.2	Iris-setosa
50	5	3.3	1.4	0.2	Iris-setosa
51	7	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
56	5.7	2.8	4.5	1.3	Iris-versicolor
57	6.3	3.3	4.7	1.6	Iris-versicolor
58	4.9	2.4	3.3	1	Iris-versicolor
59	6.6	2.9	4.6	1.3	Iris-versicolor
60	5.2	2.7	3.9	1.4	Iris-versicolor
61	5	2	3.5	1	Iris-versicolor
62	5.9	3	4.2	1.5	Iris-versicolor
63	6	2.2	4	1	Iris-versicolor
64	6.1	2.9	4.7	1.4	Iris-versicolor
65	5.6	2.9	3.6	1.3	Iris-versicolor
66	6.7	3.1	4.4	1.4	Iris-versicolor
67	5.6	3	4.5	1.5	Iris-versicolor
68	5.8	2.7	4.1	1	Iris-versicolor
69	6.2	2.2	4.5	1.5	Iris-versicolor
70	5.6	2.5	3.9	1.1	Iris-versicolor
71	5.9	3.2	4.8	1.8	Iris-versicolor
72	6.1	2.8	4	1.3	Iris-versicolor
73	6.3	2.5	4.9	1.5	Iris-versicolor
74	6.1	2.8	4.7	1.2	Iris-versicolor
75	6.4	2.9	4.3	1.3	Iris-versicolor
76	6.6	3	4.4	1.4	Iris-versicolor
77	6.8	2.8	4.8	1.4	Iris-versicolor
78	6.7	3	5	1.7	Iris-versicolor

79	6	2.9	4.5	1.5	Iris-versicolor
80	5.7	2.6	3.5	1	Iris-versicolor
81	5.5	2.4	3.8	1.1	Iris-versicolor
82	5.5	2.4	3.7	1	Iris-versicolor
83	5.8	2.7	3.9	1.2	Iris-versicolor
84	6	2.7	5.1	1.6	Iris-versicolor
85	5.4	3	4.5	1.5	Iris-versicolor
86	6	3.4	4.5	1.6	Iris-versicolor
87	6.7	3.1	4.7	1.5	Iris-versicolor
88	6.3	2.3	4.4	1.3	Iris-versicolor
89	5.6	3	4.1	1.3	Iris-versicolor
90	5.5	2.5	4	1.3	Iris-versicolor
91	5.5	2.6	4.4	1.2	Iris-versicolor
92	6.1	3	4.6	1.4	Iris-versicolor
93	5.8	2.6	4	1.2	Iris-versicolor
94	5	2.3	3.3	1	Iris-versicolor
95	5.6	2.7	4.2	1.3	Iris-versicolor
96	5.7	3	4.2	1.2	Iris-versicolor
97	5.7	2.9	4.2	1.3	Iris-versicolor
98	6.2	2.9	4.3	1.3	Iris-versicolor
99	5.1	2.5	3	1.1	Iris-versicolor
100	5.7	2.8	4.1	1.3	Iris-versicolor
101	6.3	3.3	6	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3	5.8	2.2	Iris-virginica
106	7.6	3	6.6	2.1	Iris-virginica
107	4.9	2.5	4.5	1.7	Iris-virginica
108	7.3	2.9	6.3	1.8	Iris-virginica
109	6.7	2.5	5.8	1.8	Iris-virginica
110	7.2	3.6	6.1	2.5	Iris-virginica
111	6.5	3.2	5.1	2	Iris-virginica
112	6.4	2.7	5.3	1.9	Iris-virginica
113	6.8	3	5.5	2.1	Iris-virginica
114	5.7	2.5	5	2	Iris-virginica
115	5.8	2.8	5.1	2.4	Iris-virginica
116	6.4	3.2	5.3	2.3	Iris-virginica
117	6.5	3	5.5	1.8	Iris-virginica
118	7.7	3.8	6.7	2.2	Iris-virginica
119	7.7	2.6	6.9	2.3	Iris-virginica
120	6	2.2	5	1.5	Iris-virginica

121	6.9	3.2	5.7	2.3	Iris-virginica
122	5.6	2.8	4.9	2	Iris-virginica
123	7.7	2.8	6.7	2	Iris-virginica
124	6.3	2.7	4.9	1.8	Iris-virginica
125	6.7	3.3	5.7	2.1	Iris-virginica
126	7.2	3.2	6	1.8	Iris-virginica
127	6.2	2.8	4.8	1.8	Iris-virginica
128	6.1	3	4.9	1.8	Iris-virginica
129	6.4	2.8	5.6	2.1	Iris-virginica
130	7.2	3	5.8	1.6	Iris-virginica
131	7.4	2.8	6.1	1.9	Iris-virginica
132	7.9	3.8	6.4	2	Iris-virginica
133	6.4	2.8	5.6	2.2	Iris-virginica
134	6.3	2.8	5.1	1.5	Iris-virginica
135	6.1	2.6	5.6	1.4	Iris-virginica
136	7.7	3	6.1	2.3	Iris-virginica
137	6.3	3.4	5.6	2.4	Iris-virginica
138	6.4	3.1	5.5	1.8	Iris-virginica
139	6	3	4.8	1.8	Iris-virginica
140	6.9	3.1	5.4	2.1	Iris-virginica
141	6.7	3.1	5.6	2.4	Iris-virginica
142	6.9	3.1	5.1	2.3	Iris-virginica
143	5.8	2.7	5.1	1.9	Iris-virginica
144	6.8	3.2	5.9	2.3	Iris-virginica
145	6.7	3.3	5.7	2.5	Iris-virginica
146	6.7	3	5.2	2.3	Iris-virginica
147	6.3	2.5	5	1.9	Iris-virginica
148	6.5	3	5.2	2	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3	5.1	1.8	Iris-virginica

Lampiran 2. Percobaan ke-1 *Clustering* Data FCM-GA (c = 3)

No.	Cluster 1	Cluster 2	Cluster 3	Anggota Cluster ke-
1	0.00182	0.99433	0.00385	2
2	0.00825	0.97383	0.01792	2
3	0.00733	0.97725	0.01542	2
4	0.01063	0.96631	0.02306	2
5	0.00242	0.9925	0.00508	2
6	0.01634	0.9484	0.03526	2
7	0.00646	0.9799	0.01364	2
8	0.00112	0.99646	0.00242	2
9	0.02155	0.93263	0.04582	2
10	0.00695	0.97787	0.01518	2
11	0.00929	0.97107	0.01964	2
12	0.00325	0.98965	0.0071	2
13	0.01056	0.96671	0.02272	2
14	0.02533	0.92366	0.05101	2
15	0.03406	0.89995	0.066	2
16	0.04745	0.86061	0.09194	2
17	0.01434	0.95659	0.02907	2
18	0.00096	0.99698	0.00205	2
19	0.02582	0.91851	0.05567	2
20	0.00549	0.98296	0.01155	2
21	0.00876	0.9715	0.01974	2
22	0.00305	0.99044	0.00651	2
23	0.01389	0.95883	0.02728	2
24	0.00335	0.98892	0.00773	2
25	0.00947	0.96877	0.02175	2
26	0.0084	0.97264	0.01896	2
27	0.00048	0.99845	0.00107	2
28	0.00251	0.99209	0.0054	2
29	0.00265	0.99169	0.00566	2
30	0.00697	0.97765	0.01537	2
31	0.00716	0.97689	0.01595	2
32	0.0058	0.98146	0.01275	2
33	0.01906	0.94261	0.03834	2
34	0.02903	0.91396	0.05702	2
35	0.00549	0.98242	0.01208	2
36	0.00572	0.98231	0.01196	2
37	0.01087	0.96649	0.02263	2
38	0.00433	0.98665	0.00902	2
39	0.01915	0.94083	0.04002	2

40	0.00136	0.99569	0.00295	2
41	0.00176	0.99456	0.00368	2
42	0.04409	0.86218	0.09373	2
43	0.01539	0.95273	0.03189	2
44	0.00273	0.99117	0.00609	2
45	0.01278	0.95838	0.02884	2
46	0.00857	0.97278	0.01865	2
47	0.00685	0.9786	0.01455	2
48	0.00886	0.97232	0.01882	2
49	0.00677	0.97891	0.01432	2
50	0.00201	0.99367	0.00432	2
51	0.33855	0.04263	0.61881	3
52	0.09167	0.01978	0.88855	3
53	0.38729	0.03181	0.58091	3
54	0.10397	0.08523	0.8108	3
55	0.07291	0.01278	0.91431	3
56	0.03711	0.01488	0.94801	3
57	0.13377	0.02156	0.84467	3
58	0.13235	0.35814	0.50951	3
59	0.11181	0.02109	0.8671	3
60	0.11665	0.12295	0.7604	3
61	0.14786	0.27734	0.5748	3
62	0.0411	0.01897	0.93992	3
63	0.10946	0.07904	0.8115	3
64	0.01384	0.00295	0.98321	3
65	0.11019	0.14661	0.7432	3
66	0.14995	0.03385	0.8162	3
67	0.05788	0.02251	0.91961	3
68	0.07398	0.05325	0.87277	3
69	0.08978	0.02552	0.88469	3
70	0.09721	0.09393	0.80886	3
71	0.12555	0.02231	0.85214	3
72	0.05912	0.0347	0.90618	3
73	0.10714	0.01524	0.87763	3
74	0.02999	0.00742	0.96259	3
75	0.05482	0.01791	0.92728	3
76	0.10528	0.02493	0.86979	3
77	0.24128	0.02826	0.73046	3
78	0.38798	0.02391	0.58811	3
79	0.00551	0.00158	0.99291	3
80	0.11795	0.18775	0.6943	3
81	0.11176	0.12676	0.76148	3

82	0.11862	0.15822	0.72316	3
83	0.07816	0.06526	0.85657	3
84	0.14085	0.01762	0.84153	3
85	0.08948	0.04039	0.87013	3
86	0.10354	0.02905	0.86741	3
87	0.20936	0.02697	0.76367	3
88	0.07934	0.026	0.89466	3
89	0.07464	0.05274	0.87262	3
90	0.09166	0.07569	0.83265	3
91	0.07442	0.04066	0.88492	3
92	0.01611	0.00394	0.97995	3
93	0.07089	0.052	0.87711	3
94	0.13374	0.34022	0.52604	3
95	0.06181	0.03813	0.90006	3
96	0.06299	0.03877	0.89825	3
97	0.05282	0.03072	0.91645	3
98	0.02967	0.01094	0.95939	3
99	0.11978	0.435	0.44522	3
100	0.0595	0.03913	0.90137	3
101	0.82293	0.02052	0.15655	1
102	0.1988	0.02686	0.77434	3
103	0.98949	0.00116	0.00936	1
104	0.64136	0.02073	0.33792	1
105	0.89161	0.00948	0.09891	1
106	0.86634	0.02179	0.11187	1
107	0.165	0.09606	0.73893	3
108	0.91022	0.01195	0.07783	1
109	0.78232	0.01781	0.19987	1
110	0.90297	0.01456	0.08247	1
111	0.502	0.02461	0.47339	1
112	0.48017	0.02242	0.49741	3
113	0.89466	0.00814	0.09719	1
114	0.19466	0.03376	0.77158	3
115	0.34462	0.04206	0.61332	3
116	0.65778	0.02404	0.31818	1
117	0.70859	0.01664	0.27477	1
118	0.81259	0.0355	0.15191	1
119	0.79827	0.03618	0.16555	1
120	0.15706	0.02891	0.81403	3
121	0.96416	0.00369	0.03216	1
122	0.16708	0.03339	0.79953	3
123	0.83044	0.02859	0.14097	1

124	0.14458	0.01567	0.83976	3
125	0.93664	0.0058	0.05755	1
126	0.95446	0.0054	0.04013	1
127	0.08547	0.01155	0.90298	3
128	0.12598	0.01563	0.85839	3
129	0.74343	0.0174	0.23917	1
130	0.89102	0.01134	0.09764	1
131	0.91265	0.01122	0.07613	1
132	0.80512	0.03637	0.15852	1
133	0.75354	0.01776	0.2287	1
134	0.18467	0.01747	0.79786	3
135	0.40512	0.03272	0.56215	3
136	0.88601	0.01735	0.09665	1
137	0.75114	0.02374	0.22512	1
138	0.66718	0.01894	0.31388	1
139	0.08651	0.01382	0.89967	3
140	0.87242	0.0102	0.11738	1
141	0.89561	0.00974	0.09466	1
142	0.70257	0.02383	0.2736	1
143	0.1988	0.02686	0.77434	3
144	0.97295	0.00294	0.02411	1
145	0.8961	0.01123	0.09268	1
146	0.69844	0.02125	0.28031	1
147	0.23446	0.02303	0.74251	3
148	0.54668	0.02142	0.4319	1
149	0.62999	0.02992	0.34009	1
150	0.19844	0.02341	0.77815	3