PARTICLE SWARM OTIMIZATION PADA *FUZZY C-MEANS* UNTUK MENDETEKSI SERANGAN JARINGAN KOMPUTER

SKRIPSI

Oleh :

DIRGANTARA

0610960019-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011


UNIVERSITAS BRAWIJAYA

PARTICLE SWARM OTIMIZATION PADA *FUZZY C-MEANS* UNTUK MENDETEKSI SERANGAN JARINGAN KOMPUTER

SKRIPSI

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

Oleh :
DIRGANTARA
0610960019-96

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

2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

***PARTICLE SWARM OPTIMIZATION* PADA *FUZZY C-MEANS* UNTUK MENDETEKSI SERANGAN JARINGAN KOMPUTER**

Oleh:
DIRGANTARA
0610960019-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 27 Juni 2011
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Dewi Yanti L., S.Kom, M.Kom	Edy Santoso, S.Si., M.Kom
NIP. 198111162005012004	NIP. 197404142003121004

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 19670907 199203 1 001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	:	Dirgantara
NIM	:	0610960019-96
Jurusan	:	Matematika
Program Studi	:	Ilmu Komputer
Penulis Skripsi berjudul	:	*Particle Swarm Optimization* pada *Fuzzy c-means* Untuk Mendeteksi Serangan Jaringan Komputer

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 dalam isi dan tertulis pada daftar pustaka dalam Skripsi ini.
2. Apabila dikemudian 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, 27 Juni 2011
Yang menyatakan,

(Dirgantara)
NIM. 0610960019-96


UNIVERSITAS BRAWIJAYA

PARTICLE SWARM OPTIMIZATION PADA *FUZZY C-MEANS* UNTUK MENDETEKSI SERANGAN JARINGAN KOMPUTER

ABSTRAK

Intruction detection System (IDS) adalah perangkat lunak untuk mendeteksi aktivitas yang mencurigakan dalam jaringan komputer dengan menganalisa paket data yang masuk. Sebagian IDS bekerja dengan basis *anomaly*. Metode ini menggunakan teknik pengelompokan untuk mengklasifikasikan jenis serangan.

Fuzzy c-means merupakan salah satu algoritma *clustering*. *Fuzzy c-means* adalah sebuah teknik pengelompokan data dimana keberadaan tiap titik data pada setiap *cluster* ditentukan dengan menggunakan nilai derajat keanggotaan (Kusumadewi, 2004). Pada penelitian ini *Fuzzy c-means* akan diimplementasikan untuk mengelompokan aktivitas serangan jaringan komputer.

Record data aktivitas jaringan komputer memiliki *feature* yang besar yang menyebabkan hasil *clustering* tidak optimal. Oleh karena itu diperlukan algoritma optimasi. *Partikel Swarm Optimization(PSO)* adalah bagian dari *Swarm Intelligence* untuk menyelesaikan masalah optimasi yang memiliki kemampuan dalam mencapai titik optimum secara efektif (Kennedy, 2001).

Pada penelitian ini sebuah ruang permasalahan PSO memiliki beberapa *particle*. Setiap *particle* merupakan proses *clustering* yang memiliki *pbest* dan vector *velocity*. Pada setiap generasi, masing-masing *particle* akan berinteraksi untuk mencapai solusi hasil *clustering* terbaik. Pada akhir generasi akan dipilih *particle* tebaik dari semua *particle* untuk semua generasi.

Hasil dari penelitian ini, yaitu algoritma optimasi *Partikel Swarm Optimization(PSO)* dapat digunakan pada *Fuzzy c-means*. Penggunaan PSO pada *Fuzzy c-means* terbukti dapat meningkatkan nilai *objektif* hasil *clustering*. Dalam percobaan dengan beberapa parameter yang berbeda didapatkan nilai optimasi terbaik pada jumlah particle =0, w = 0,1 dan nilai optimasi paling buruk pada w = 0,9 dengan jumlah perulangan sebanyak 20 kali.

Keywords IDS, Fuzzy c-means, *Partikel Swarm Optimization*


UNIVERSITAS BRAWIJAYA

PARTICLE SWARM OPTIMIZATION IN FUZZY C-MEANS FOR DETECTION OF COMPUTER NETWORK ATTACKS

ABSTRACT

Intruction Detection System (IDS) is a software to detect suspicious activity in the system or computer network (Brenton, 2003). IDS identification of incoming data packets in the system or network and analyzing the possibility of network attacks. In the detection of computer network attacks, IDS works in several ways, including by using anomaly-based attack detection. This type of traffic patterns that may involve an attack on computer networks. This method uses clustering techniques to compare between normal activity with a variety of computer network attacks.

Fuzzy c-means is one of several clustering algorithms. Fuzzy c-means is a data clustering technique where the existence of each data point in each cluster is determined by using the highest degree of membership (Kusumadewi, 2004).

Record data in a computer network activity has a feature that is very large. This will cause the results of clustering is not optimal. Therefore we need algorithms optimization for clustering problems. Particle Swarm Optimization (PSO) is part of Swarm Intelligence to solve optimization problems. Particle Swarm Optimization (PSO) is used because it has the ability to achieve the optimum point and have a consistent and effective performance in handling the optimization problems (Kennedy, 2001).

The results of this study, namely Particle Swarm Optimization algorithm optimization (PSO) can be used on Fuzzy c-means. The use of PSO on Fuzzy c-means proven to increase the value of the objective results of clustering. In experiments with several different parameters obtained at the best optimization value w = 0.1 and the optimization of the worst at w = 0.9 with 20 times the number of iterations.

Keywords IDS, Fuzzy c-means, *Partikel Swarm Optimization*


UNIVERSITAS BRAWIJAYA

Kata Pengantar

Alhamdulillahi rabbil ‘alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, atas segala rahmat dan limpahan hidayah-Nya tugas akhir yang berjudul “Optimasi Particle Swarm Optimization pada Fuzzy c-means Untuk Mendeteksi Serangan Jaringan Komputer” ini dapat terselesaikan. Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian tugas akhir ini, penulis mendapat banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Dewi Yanti L., S.Si., M.Kom dan Edy Santoso, S.Si., M.Kom., selaku dosen pembimbing, terima kasih atas semua saran, bantuan, kritikan, waktu, dorongan semangat dan bimbingannya.
2. Marji, MT selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang. Terima kasih atas saran dan nasihat yang diberikan.
3. Agus Wahyu Widodo,ST selaku dosen Penasihat Akademik. Terima kasih atas semua support, masukan, dan bimbingan yang telah diberikan.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis serta segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya.
5. Bapak Sukoyuwono, Ibu Suhariah terima kasih atas cinta kasih saying, doa, dukunagan dan semangat yang tiada henti. Alhamdulillah memiliki kalian.
6. Teman-teman ilkomers 2006, terima kasih untuk semangat, cinta dan kebersamaan yang diberikan
7. Issoft team, terima kasih atas pengalaman kerja yang diberikan kepada penulis. InsyaAlloh sangat bermanfaat.
8. Pihak lain yang telah membantu terselesaikannya Tugas Akhir ini yang tidak bias penulis sebutkan satu-persatu

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Sebagai manusia, penulis menyadari tugas akhir ini masih jauh dari kesempurnaan dan banyak kekurangan, dengan kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 27 Januari 2011

Penulis

DAFTAR ISI


UNIVERSITAS BRAWIJAYA


UNIVERSITAS BRAWIJAYA

DAFTAR GAMBAR


UNIVERSITAS BRAWIJAYA

DAFTAR TABEL


UNIVERSITAS BRAWIJAYA

DAFTAR SOURCECODE


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1. Latar Belakang

Sistem keamanan komputer merupakan hal yang penting dalam membangun jaringan komputer. Ironisnya banyak serangan pada jaringan komputer yang tidak terdeteksi sehingga dapat mengganggu kinerja sistem komputer. Menurut data dari *CSI/FBI survey*, banyak perusahaan yang menganggarkan dana untuk terhindar dari masalah keamanan jaringan komputer dan telah mengkombinasikan beberapa teknologi sistem keamanan, dimana hampir 69%nya menggunakan solusi dari *Intrusion Detection System* (CSI, 2006).

Intruction detection System (IDS) adalah sebuah perangkat lunak untuk mendeteksi aktivitas yang mencurigakan dalam sistem atau jaringan komputer. IDS melakukan identifikasi terhadap paket data yang masuk dalam sistem atau jaringan dan menganalisa kemungkinan adanya serangan jaringan. Dalam mendeteksi serangan jaringan, IDS bekerja dengan beberapa cara. Cara yang paling populer dengan menggunakan deteksi berbasis *signal*, yang melibatkan lalulintas jaringan dan basis data yang berisi cara-cara serangan dan penyusupan yang sering terjadi. Cara jenis ini membutuhkan pembaharuan basis data IDS (Brenton, 2003).

Deteksi serangan dapat dilakukan dengan memanfaatkan *anomaly*. Jenis ini melibatkan pola lalulintas yang mungkin merupakan sebuah serangan terhadap sistem atau jaringan komputer. Metode ini menggunakan teknik pengelompokan untuk membandingkan lalulintas yang sedang dipantau dengan lalulintas normal yang biasa terjadi dalam jaringan komputer. Kelebihan metode ini adalah dapat mendeteksi bentuk aktifitas baru dalam jaringan yang belum terdapat di dalam basis data IDS (Singh, 2009).

Aktifitas dalam jaringan memiliki beberapa variabel yang membedakan antara aktivitas normal dengan beberapa jenis serangan pada jaringan komputer. Dengan menggunakan variabel aktifitas dalam jaringan komputer dapat dilakukan pengelompokan beberapa jenis serangan dalam jaringan komputer.

Terdapat beberapa metode pengelompokan data, diantaranya adalah *clustering* hirarki, *K-Means*, *fuzzy c-means* dan KNN. Skripsi ini mengimplementasikan salah satu teknik data mining yaitu *clustering*, metode *clustering* yang digunakan adalah algoritma *fuzzy c-means*. *Fuzzy c-means* merupakan salah satu dari sekian banyak algoritma clustering. *Fuzzy c-means* adalah suatu teknik pengklusteran data dimana keberadaan tiap titik data cluster ditentukan oleh derajat keanggotaan (Kusumadewi, 2004).

Fuzzy c-means merupakan metode yang efisien untuk memecahkan masalah klastering. Namun masalahnya adalah dimensi data aktivitas dalam jaringan yang besar akan meyebabkan metode ini sulit mencapai hasil yang maksimal.

Partikel Swarm Optimization(PSO) adalah bagian dari *Swarm Intelligence* untuk menyelesaikan masalah optimasi. *Partikel Swarm Optimization (PSO)* digunakan karena memiliki kemampuan dalam mencapai titik optimum serta memiliki performa yang konsisten dan efektif dalam menangani masalah optimasi (kennedy, 2001). PSO banyak digunakan dalam optimasi hasil klastering diantaranya zumstein (2009) dan Simha (2000). Dalam penelitian tersebut disebutkan bahwa optimasi dengan menggunakan PSO memberikan hasil yang lebih baik pada hasil *clustering*.

Berangkat dari latar belakang yang telah diuraikan, maka skripsi ini diberi judul **" Optimasi *Particle Swarm Optimalization* Pada *Fuzzy c-means* Untuk Mendeteksi Serangan Jaringan Komputer ".**

1.2. Rumusan Masalah

Permasalahan yang ditangani dalam skripsi ini adalah :

1. Bagaimana merancang *Fuzzy c-means* yang dioptimasi menggunakan *Particle Swarm Optimalization.*
2. Bagaimana mengimplementasikan deteksi serangan jaringan menggunakan *Fuzzy c-means* yang dioptimasi dengan *PSO* pada *dataset* aktivitas dalam jaringan**.**
3. Bagaimana menguji hasil klasifikasi yang terbaik antara *FCM* dan *FCM-PSO*.

1.3. Tujuan

Tujuan pembuatan skripsi ini adalah :

1. Mengimplementasikan *Particle Swarm Optimization* untuk mengoptimasi metode *Fuzzy c-means* untuk klastering *dataset* aktifitas jaringan computer.
2. Mengimplimentasikan deteksi serangan jaringan menggunakan *Fuzzy c-means* yang dioptimasi menggunakan *Particle Swarm Optimization*.
3. Menguji hasil klasifikasi terbaik antara *FCM* dan *FCM-PSO*

1.4. Batasan Masalah

Batasan Masalah pada skripsi adalah :

1. Tipe aktivitas yang digunakan pada proses *training* deteksi adalah terbatas pada 5 tipe serangan yaitu *neptun, smurf, ipsweep, pod* dan *normal*.
2. Data yang digunakan untuk proses training dan deteksi adalah data aktivitas di dalam jaringan yang terdiri dari 21 variabel yang memiliki nilai numerik.

1.5. Manfaat

Manfaat yang diperoleh dari penulisan tugas akhir ini adalah :

1. Melalui hasil program, administrator jaringan dapat menggunakan program tersebut sebagai rujukan dalam mencegah kemungkinan adanya serangan jaringan.
2. Dapat mempercepat proses pendeteksian serta dapat mengurangi tingkat kesalahan deteksi pada IDS.

1.6. Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep *fuzzy clustering* dan *PSO* dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dengan analisis

terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu *fuzzy clustering* untuk deteksi serangan jaringan.

4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian menarik kesimpulannya.

1.7. Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi 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 *fuzzy C-Mean dan Particle swarm Optimition.*
3. BAB III METODOLOGI DAN PERANCANGAN SISTEM
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam proses klastering dan *optimasi* menggunakan *Particle swarm Optimization.*
4. BAB IV PEMBAHASAN
   Pada bab ini akan dilakukan implementasi *fuzzy C-mean dan PSO* untuk deteksi serangan jaringan dan pengujian keakuratan sistem serta analisis sistem perangkat lunak yang dibangun yaitu apakah hasil dari *fuzzy C-Mean dan Particle swarm Optimization* menghasilkan informasi yang diinginkan.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

Untuk menyelesaikan rumusan permasalah diatas, diperlukan landasan teori . Adapun dasar teori yang digunakan dalam penelitian skripsi ini adalah menyangkut *Intrucion Detection System, Fuzzy c-means*, dan *Particle Swarm Optimization*

2.1 Intrusion Detection System(IDS)

Intrusi adalah aksi yang dapat membahayakan integritas, kerahasiaan dan ketersediaan sumber daya yang ada. Deteksi serangan jaringan adalah usaha untuk menemukan serangan dengan cara menguji dan mengobservasi berbagai data sebagai target serangan. Pada dasarnya serangan dapat dikategorikan sebagai *host based* dan *network based* (Beale, 2004).

Sistem pendeteksi serangan merupakan sebuah alarm yang dikonfigurasi untuk mengawasi jaringan, aktifitas-aktifitas berbahaya dan mungkin penyerang yang sudah diketahui. Sistem pendeteksi serangan dapat membangkitkan peringatan dan bahkan melakukan tindakan lebih lanjut. Seperti misalnya mematikan jaringan atau server, apabila sistem menemukan aktifitas yang mencurigakan. Deteksi serangan dapat dilakukan dengan cara melakukan pemantauan terhadap jaringan dan komputer dan kemudian melakukan proses analisa untuk menemukan kejanggalan-kejanggalan dalam jaringan komputer`(Beale, 2004).

Pada umumnya sistem pendeteksi *intrusi* umumnya terbagi menjadi tiga komponen utama yaitu sensor, *analyzer*, dan komponen manajemen. Sensor berfungsi untuk mengumpulkan paket-paket data. *Analyzer* berfungsi untuk menganalisa data-data yang diperoleh. Kemudian komponen menejemen berfungsi menyediakan pengelolaan antar muka sistem. Fungsi manajemen juga mencakup pengiriman peringatan kepada administrator jaringan (rehman, 2003).

Intrusion Detection System (*IDS*) adalah sebuah perangkat lunak yang dapat mendeteksi aktivitas yang mencurigakan dalam sebuah sistem atau jaringan. *IDS* dapat melakukan inspeksi terhadap lalu lintas *inbound* dan outbound dalam sebuah sistem atau jaringan, melakukan analisis dan mencari bukti dari percobaan *intrusi* (penyusupan).

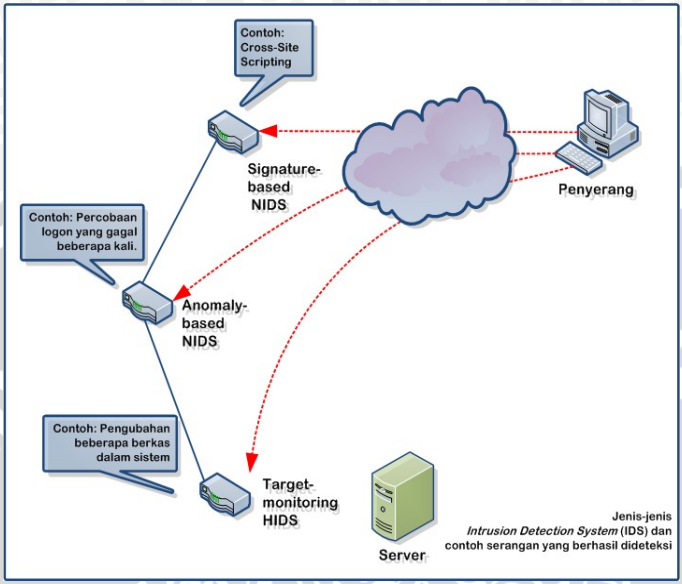
IDS dibedakan menjadi 2 macam yaitu :

Network-based Intrusion Detection System (NIDS): Semua lalu lintas yang mengalir ke sebuah jaringan akan dianalisis untuk mencari apakah ada percobaan serangan atau penyusupan ke dalam sistem jaringan. NIDS umumnya terletak di dalam segmen jaringan penting di mana server berada atau terdapat pada "pintu masuk" jaringan. Kelemahan NIDS adalah bahwa NIDS agak rumit diimplementasikan dalam sebuah jaringan yang menggunakan *switch* Ethernet, meskipun beberapa *vendor* switch Ethernet sekarang telah menerapkan fungsi IDS di dalam *switch* buatannya untuk memonitor port atau koneksi (bace, 2000).

Host-based Intrusion Detection System (HIDS): Aktivitas sebuah *host* jaringan individual akan dipantau apakah terjadi sebuah percobaan serangan atau penyusupan ke dalamnya atau tidak. HIDS seringnya diletakkan pada server-server kritis di jaringan, seperti halnya *firewall*, *web server*, atau server yang terkoneksi ke Internet (bace, 2000).Dalam membuat keputusan apakah sebuah paket data berbahaya atau tidak, IDS dapat mempergunakan beberapa metode.

Signature-based Intrusion Detection System. Pada metode ini, telah tersedia daftar signature yang dapat digunakan untuk menilai apakah paket yang dikirimkan berbahaya atau tidak. Sebuah paket data akan dibandingkan dengan daftar yang sudah ada. Metode ini akan melindungi sistem dari jenis-jenis serangan yang sudah diketahui sebelumnya. Oleh karena itu, untuk tetap menjaga keamanan sistem jaringan komputer, data signature yang ada harus tetap ter-*update* (rehman, 2003).

Anomaly-based Intrusion Detection System. Pada metode ini, pengelola jaringan harus melakukan konfigurasi terhadap IDS dan, sehingga IDS dapat mengatahui pola paket seperti apa saja yang akan ada pada sebuah sistem jaringan komputer. Sebuah paket *anomaly* adalah paket yang tidak sesuai dengan kebiasaan jaringan komputer tersebut. Apabila IDS menemukan ada *anomaly* pada paket yang diterima atau dikirimkan, maka IDS akan memberikan peringatan pada pengelola jaringan (IDS) atau akan menolak paket tersebut untuk diteruskan (IPS). Untuk metode ini, pengelola jaringan harus terus-menerus memberi tahu IDS bagaimana lalu lintas data yang normal pada sistem jaringan komputer tersebut, untuk menghindari adanya salah penilaian oleh IDS (rehman, 2003).



Gambar 2.1 *Intrucion Detection System*

Parameter pada aktivitas jaringan komputer ada banyak sekali. Adapun dalam penelitian ini akan difokuskan 7 parameter yang paling penting seperti pada tabel 2.1.

Table 2.1 tabel parameter aktivitas jaringan komputer

No	Nama variabel	Satuan
1	*duration:*	*crips*
2	*protocol_type:*	*continue*
3	*service:*	*continue*
4	*flag:*	*continue*
5	*src_bytes:*	*continue*
6	*dst_bytes:*	*continue*
7	*hot:*	*continue*
8	*num_failed_logins:*	*continue*
9	*logged_in:*	*continue*
10	*root_shell:*	*continue*
11	*num_shells:*	*continue*

12	*num_access_files:*	*continue*
13	*count:*	*continue*
14	*srv_count:*	*continue*
15	*serror_rate:*	*continue*
16	*srv_serror_rate:*	*continue*
17	*rerror_rate:*	*continue*
18	*srv_rerror_rate:*	*continue*
19	*same_srv_rate:*	*continue*
20	*diff_srv_rate:*	*continue*
21	*srv_diff_host_rate:*	*continue*

2.2. Data *Clustering*

Data *clustering* adalah suatu proses membagi sekumpulan data atau objek menjadi sejumlah kelas yang mempunyai nilai tertentu, yang disebut sebagai *cluster* (koleksi data yang memiliki kesamaan antara satu dengan yang lain sehingga dapat dikelompokan dalam satu group) (zainae, 1999).

Cara kerja data *clustering* adalah mengkoordinasi data-data yang ada ke dalam beberapa kelas, yang anggota dari kelas tersebut memiliki kesamaan dalam hal-hal tertentu. Teknik *clustering* dapat digunakan pada berbagai bidang antara lain, *data mining, machine learning*, dan pengenalan pola. *Clustering* juga merupakan salah satu komponen untuk melakukan eksplorasi terhadap data sehingga data tersebut dapat dianalisa dan menghasilkan informasi.

Jenis-jenis clustering berdasar kesamaan pada data :

1. *Distance Based Clustering*

   Jika dua atau lebih data yang memiliki jarak yang berdekatan berdasar sebuah jarak yang ditentukan antara dua atau lebih data, maka hal ini disebut *distance- based clustering*.

2. *Conceptual Clustering*

   Jika dua atau lebih data yang dimiilki oleh sebuah cluster memiliki kesamaan umum dalam konsep tertentu dan bukan berdasarkan kesamaan pengukuran, hal ini disebut sebagai *conseptual clustering*.

Secara umum data clustering yang digunakan untuk data matematis adalah *distance based clustering*. *distance based*

clustering memiliki jenis algoritma yang umum digunakan, yaitu

1. *Exclusive Clustering*
   Clustering jenis ini biasa disebut dengan hard clustering, karena algoritma clustering jenis ini menentukan apakah data tersebut masuk pada cluster tertentu dengan pernyataan ya atau tidak (0,1). Salah satu algoritma clustering jenis ini adalah K-Means.
2. *Overlaping Clustering*
   Algoritma clustering jenis ini menentukan seberapa besar suatu data menjadi milik cluster tertentu(0,1). Clustering fuzzy C- Means termasuk dalam algoritma jenis ini
3. *Hierarchichal Clustering*
   Algoritma clustering jenis ini menggunakan sistem cluster yang bertingkat dengan cara menggabungkan cluster-cluster terdekat.
4. *Probabilistic Clustering*
   Algoritma clustering ini menggunakan sebuah model seperti gaussian atau yang lainnya sebagai cluster.

2.3. Sistem *Fuzzy*

Menurut Kusuma Dewi (2003), terdapat beberapa hal yang perlu diketahui dalam memahami sistem fuzzy, yaitu:

1. Peubah fuzzy
   Merupakan variabel atau peubah yang akan dibahas dalam suatu sistem fuzzy. Contoh: umur, temperature, permintaan, dsb.
2. Himpunan fuzzy
   Adalah suatu kelompok yang mewakili suatu kondisi atau keadaan tertentu dalam suatu peubah fuzzy. contoh: peubah suhu terbagi menjadi 3 himpunan fuzzy, yaitu panas, hangat dan dingin. Himpunan fuzzy didasarkan pada gagasan untuk memperluas jangkauan fungsi karakteristik sedemikian hingga fungsi tersebut akan mencakup bilangan real pada interval [0,1]. Derajat keanggotaan menunjukkan bahwa suatu objek data dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga berada pada nilai yang terletak di anatara 0 sampai 1. Derajat keanggotaan suatu objek data tidak hanya bernilai

benar atau salah. Semakin besar nilai derajat keanggotaan maka semakin dominan data tersebut masuk pada sebuah klaster begitu juga sebaliknya.

3. Fungsi Keanggotaan

   Merupakan suatu kurva yang menunjukkan pemetaan objek data ke dalam derajat keanggotaan yang memiliki interval antara 0 sampai 1. Sedangkan tinggi himpunan fuzzy adalah derajat keanggotaan maksimum suatu objek data. Dalam algoritma *fuzzy c-means* nilai derajat keanggotaan didapatkan antara 0 sampai 1. Dalam kasus ini garis fungsi keanggotaannya membentuk garis yang lebih halus untuk mengindikasikan bahwa setiap objek titik data mungkin dapat menjadi kelompok lain dengan derajat keanggotaan berbeda

2.4. *Fuzzy clustering*

Pada analisis cluster klasik, kelompok- kelompok yang terbentuk setelah proses *clustering*, harus merupakan pembagian data set sedemikian rupa, sehingga untuk data yang berada pada kelompok lain. Walaupun demikian, syarat ini seringkali dirasakan terlalu kaku dan sulit dipenuhi pada aplikasi praktek, sehingga perlu diganti dengan syarat yang lebih ringan. Untuk itu digunakanlah kompromi dengan prinsip-prinsip fuzzy yang kemudian dikenal sebagai fuzzy clustering.

Dengan demikian dapat dikatakan *fuzzy clustering* adalah suatu teknik untuk mengelompokkan data, yaitu membagi data set yang diberikan ke dalam kelas-kelas atau kelompok-kelompok yang dikenal dengan sebutan cluster. Sedangkan tujuan dari pembagian ini adalah untuk memisahkan data set sedemikian rupa hingga untuk dua buah data pada cluster yang sama adalah semirip mungkin, dan untuk dua data pada cluster yang tidak sama adalah seberbeda mungkin.

Dalam matriks, biasanya kesamaan didefinisikan sebagai suatu jarak dari vector data ke sebuah objek atau pusat cluster. Pusat dari cluster biasanya belum diketahui sebelumnya dan dicari menggunakan algortima clustering pada saat membuat partisi (Kusumadewi, 2002).

2.5 Fuzzy C Mean

Fuzzy c-means pertama kali dikemukakan oleh Dunn (1973) dan dikembangkan oleh Bezdek (1981) yang banyak digunakan dalam pattern recognition. Metode ini merupakan pengembangan dari metode non hierarkhi *K-Means* Cluster, karena pada awalnya ditentukan dulu jumlah kelompok atau cluster yang akan dibentuk. Kemudian dilakukan iterasi sampai mendapatkan keanggotaan kelompok tersebut. Pemilihan metode ini didasarkan pada beberapa jurnal dan penelitian sebelumnya yang mengindikasikan bahwa metode fuzzy cluster merupakan metode yang paling robust, karena pusat cluster dan hasil pengelompokkan tidak berubah jika ada data baru yang ekstrim (Klawon, 2000). Metode ini juga memberikan hasil yang smooth (halus) karena pembobotan yang digunakan berdasarkan himpunan fuzzy (Shihab, 2000) kehalusan disini berarti objek pengamatan tidak mutlak untuk menjadi anggota satu kelompok saja, tapi juga mungkin menjadi anggota kelompok yang lain dengan ukuran

tingkat keanggotaan yang berbeda-beda. Objek akan cenderung menjadi anggota kelompok tertentu dimana tingkat keanggotaan objek dalam kelompok itu paling besar dibandingkan dengan kelompok lainnya. FCM baik digunakan untuk mengelompokkan objek terutama jika objek-objek tersebut tersebar berserakan dan terdapat nilai ekstrim didalamnya. Ketidakteraturan bukan berarti objek-objek tersebut tidak berpola, namun yang dimaksud ketidakteraturan ini berarti tidak ada kecenderungan yang pasti bahwa objek- objek tersebut akan mengelompok secara jelas.

2.6 Algoritma Fuzzy C Mean

Algoritma *Fuzzy C-Means (FCM)* adalah sebagai berikut:

1. Mauksan data yang akan dikelompokan dimasukan ke dalam sebuah matrix X, matrix X adalah matriks berukuran n x m (n = jumlah sampel data m adalah jumlah *feature* .
2. Tentukan :

   C = jumlah klaster.
   W = pangkat.
   MaxIter = maksimum jumlah iterasi.

ε = *Error* terkecil
Po=0 = Fungsi objektif awal
t=1 = iterasi awal

3. Bangkitkan nilai acak antara 0 dan 1 untuk mengisi matriks partisi awal yaitu matriks U, matriks U adalah metrik untuk menampung nilai derajat keanggotaan setiap input data di setiap kluste. Ukuran matriks U adalah n x c (n = jumlah sampel data c adalah jumlah klaster). Setiap elemen matriks U diisi dengan derajat keanggotaan μ_{ik} , dengan I = 1,2,…,n dan k = 1,2,..,c.
4. Tentukan pusat klaster *feature* ke-j pada klaster ke-k. pusat klaster digunakan untuk menghitung derajat keanggotaan setiap *input* data. Pusat klaster dapat dihitung dengan rumus 2.1 :

$$V_{kj} = \frac{\sum_{i=1}^{n}\left((\mu_{ik})^{w} * X_{ij}\right)}{\sum_{i=1}^{n}(\mu_{ik})^{w}} \qquad (2.1)$$

Keterangan :
V_{kj} = pusat klaster
j = 1,2, . . .,m dengan m adalah jumlah *feature*.
k = 1,2, . . .,c dengan c adalah jumlah klaster.
i = 1,2,…,n dengan n adalah jumlah input data.
w = adalah pangkat

5. Menghitung fungsi objektif pada iterasi ke-t dapat dihitung dengan menggunakan rumus 2.2.

$$Pt = \sum_{i=1}^{n}\sum_{k=1}^{c}\left(\left[\sum_{j=1}^{m}\left(X_{ij} - V_{kj}\right)^{2}\right](\mu_{ik})^{w}\right) \qquad (2.2)$$

Pt = fungsi objektif pada iterasi ke-t.
V_{kj} = pusat klaster pada klaster ke-k dan *feature* ke-j.
X_{ij} = input data ke-I dan *feature* ke-j.
w = pangkat.

6. Hitung perubahan matrik partisi awal U dengan rumus 2.3.

$$\mu_{ik} = \frac{\left[\sum_{j=1}^{m}\left(X_{ij}-V_{kj}\right)^2\right]^{\frac{-1}{w-1}}}{\sum_{k=1}^{c}\left[\sum_{j=1}^{m}\left(X_{ij}-V_{kj}\right)^2\right]^{\frac{-1}{w-1}}} \quad (2.3)$$

Keterangan :
μ_{ik} = nilai derajat keanggotaan.
V_{kj} = pusat klaster pada klaster ke-k dan *feature* ke-j.
X_{ij} = *input* data ke-I dan *feature* ke-j.

7. Cek kondisi berhenti
   a) Jika $(|Pt - Pt - 1| < \varepsilon)$ atau $(t > maxIter)$ maka berhenti,
   b) Jika tidak : t = t +1, ulangi langkah ke-4

2.7 Swarm Intelligence

Swarm Intelligence didasarkan pada fenomena biologis dari hewan yang hidup secara berkelompok. *Swarm Intelligence* memperhatikan tindakan dari setiap individu dalam kelompok besar yang melakukan suatu pekerjaan secara bersama sehingga menghasilkan kelakuan kelompok. System yang terdiri dari sekian banyak individu dengan interaksi antar individu yang sangat sederhana namun menghasilkan solusi yang sangat besar. Swarm intelligence merupakan kecerdasan yang muncul secara kolektif dari suatu grup yang berupa otonomi sederhana dari individu-individunya (said, 2008).

Pada umumnya tidak ada pengendali pusat untuk setiap partikel, namun interaksi antar partikel berpotensi untuk menghasilkan optimum global. Interaksi kolektif dari semua partikel dalam suatu sistem mengarah pada perilaku kolektif atau intelegen. Karakteristik dari *Swarm Intelegence* adalah kemampuan untuk bertindak dengan cara terkoordinasi tanpa kehadiran koordinator (Dorigo, 2004). Salah satu contoh dari *Swarm Intellegence* ini adalah *Particle Swarm Optimization Algorithm.*

2.8 Partikel Swarm Optimization

Pada tahun 1995, Kennedy dan Eberhart memperkenalkan teknik komputasi evolusioner yang disebut *Particle Swarm Optimization (PSO). PSO* adalah bagian dari *Swarm Intelligence* untuk menyelesaikan masalah optimasi. Metode ini terinspirasi oleh sekumpulan hewan yang saling berinteraksi untuk menemukan makanan. Setiap individu merupakan solusi yang potensial dari suatu populasi untulk menrmukan makanan. Setiap individu dalam *PSO* dapat disebut sebagai partikel. Setiap partikel memiliki nilai *fitnes* yang dievaluasi oleh fungsi *fitnes* agar optimal dan memiliki nilai vector *velocity* yang mengarahkan partikel kepada solusi permasalahan. (vitorino, 2008).

Optimasi dengan menggunakan *PSO* dimulai dengan populasi acak yang disebut partikel. PSO berhubungan dengan nilai *fitness* dan *velocity*. Partikel bergerak dari posisi awal menuju posisi terbaik dengan cara mengupdate *velocity* yang diperoleh dari posisi sebelumnya. Vector *velocity* diupdate untuk masing-masing partikel kemudian dijumlahkan dengan posisi partikel. Update nilai *velocity* dipengaruhi oleh dua solusi yaitu *global best* dan *personal best*. Solusi global best ditentukan dengan biaya paling rendah dari semua partikel, sedangkan personal best ditentukan dengan biaya paling rendah dari posisi awal setiap partikel.

Dalam *PSO*, setiap partikel bergerak menuju solusi dengan posisi yang lebih baik dan memiliki kemampuan untuk mengingat posisi terbaik sebelumnya sehingga dapat mempertahankan solusi optimum dari generasi ke generasi. Dalam teknik komputasi PSO juga terdapat parameter w (inertia weight) yang digunakan untuk mengontrol dampak dari adanya vector velocity yang diberikan oleh partikel (vitorino, 2008)

Dalam *PSO* semua partikel memiliki nilai vector *velocity dan* nilai *fitnes* yang selalu dievaluai oleh fungsi optimasi. *Fitness* dan *velocity* digunakan untuk mengatur arah dan kecepatan partikel

agar mencapai *global optimum* dengan cara mengikuti partikel optimum saat itu dan memperbaharui generasinya.

PSO bertujuan menghasilkan posisi partikel yang dihasilkan oleh perhitungan fungsi *fitness*. Setiap partikel dalam ruang populasi bergerak menuju posisi optimum dengan menyesuaikan terhadap posisi terbaik partikel sejauh ini dan posisi terbaik partikel di lingkungan itu. Persamaan matematika algoritma dari PSO dapat dilihat pada persamaan 2.13 dan 2.14.

$$v_{id} = w * v_{id} + a_1 * r_1(p_{id} - x_{id}) + a_2 * r_2(p_{gd} - x_{id}) \quad (2.4)$$

(persamaan 2.13 merupakan rumus untuk mencari vektor *velocity* partikel yang baru)

$$x_{id} = x_{id} + v_{id} \quad (2.5)$$

(persamaan 2.14 merupakan rumus untuk mencari posisi partikel yang baru)

2.9 Komponen *Particle Swarm Optimization*

Tujuan PSO adalah menghasilkan posisi partikel yang dihasilkan oleh perhitungan fungsi *fitness*. Setiap partikel dalam ruang populasi bergerak menuju posisi optimum dengan menyesuaikan terhadap posisi terbaik partikel sejauh ini dan posisi terbaik partikel di lingkungan itu.

Adapaun komponen-komponen yang digunakan PSO untuk menndapatkan global optimum adalah sebagai berikut :

a. Ukuran swarm (s)

Ukuran *swarm* merupakan komponen yang berfungsi untuk menentukan jumlah partikel yang berada pada populasi. Semakin banyak dan beragamnya partikel akan memungkinkan daerah pencarian semakin luas dan memberikan peluang yang lebih besar untuk mencapai nilai *fitness* yang bagus. Ukuran populasi

yang terlalu besar dapat menyebabkan menurunnya kecepatan konvergensi (Engelbrecht, 2007).

b. Personal best (ρ_{id})

Personal best adalah nilai *fitness* terbaik yang didapat oleh setiap partikel dengan membandingkannya dengan nilai *fitness* sebelumnya dari masing-masing partikel.

c. Global best (ρ_{gd})

Global best adalah posisi terbaik yang didapat dari perbandingan nilai *fitness* semua partikel yang ada dalam sebuah populasi.

d. Koefisien akselerasi (c_1 *dan* c_2)

Berguna untuk untuk mengontrol sejauh mana partikel akan bergerak menuju titik optimum. Kedua koefisien ini tidak mempengaruhi kekonvergenan algoritma PSO, tetapi pemilihan nilai yang tepat dapat meningkatkan kecepatan konvergensi dan dapat menyebabkan algoritma tidak mudah terjebak dalam optimasi lokal (Parsopoulus, 2002). Nilai dari koefisien akselerasi berkisar antara 0 dan 4.

e. Vector velocity (v_{id})

Vector velocity merupakan *vector* yang menggerakkan posisi partikel untuk mendapatkan posisi terbaik. *Vector velocity* digunakan untuk menentukan arah gerak partikel terhadap posisi semula. *Velocity* dibangkitkan secara *uniform random* (*uniform distribution*) dalam range sebaran sesuai dengan persamaan 2.15 (Clerc, 2006):

f. Inertia weight (w)

Parameter ini mengatur seberapa besar kecepatan iterasi sebelumnya mempengaruhi kecepatan iterasi berikutnya. Nilai *w* yang besar memfasilitasi pencarian minimum global sedangkan nilai *w* yang kecil memfasilitasi pencarian minimum lokal (Parsopoulos, 2002).

g. Jumlah iterasi ($Iter_{max}$)

Jumlah iterasi mempunyai andil besar dalam menemukan hasil yang lebih baik. Jumlah iterasi yang terlalu kecil akan menghentikan pencarian terlalu dini, tetapi jumlah iterasi yang terlalu besar akan berakibat waktu komputasi menjadi lebih lama (Engelbrecht, 2007).

Prosedur standar untuk menerapkan algoritma PSO adalah sebagai berikut:

1. Inisialisasi partikel-partikel dengan posisi dan *velocity* secara *random* dalam suatu ruang dimensi penelusuran.

2. Memperbaharui nilai *fitness* yang ada pada setiap partikel di setiap iterasi / generasi.

3. Membandingkan evaluasi *fitness* partikel dengan nilai *fitness pbest*-nya. Jika nilai *fitness* yang ada lebih baik dibandingkan dengan nilai *fitness pbest*-nya, maka *pbest* diset sama dengan nilai dan lokasi partikel tersebut.

4. Mengidentifikasi partikel dalam lingkungan dengan hasil terbaik sejauh ini.

5. Meng-*update velocity* dan posisi partikel. Untuk meng-*update* nilai *velocity* menggunakan persamaan 2.13 dan posisi partikel baru ditentukan dengan persamaan 2.14

6. Kembali ke langkah kedua sampai kriteria terpenuhi, biasanya berhenti pada nilai *fitness* yang cukup baik atau sampai pada jumlah iterasi maksimum.

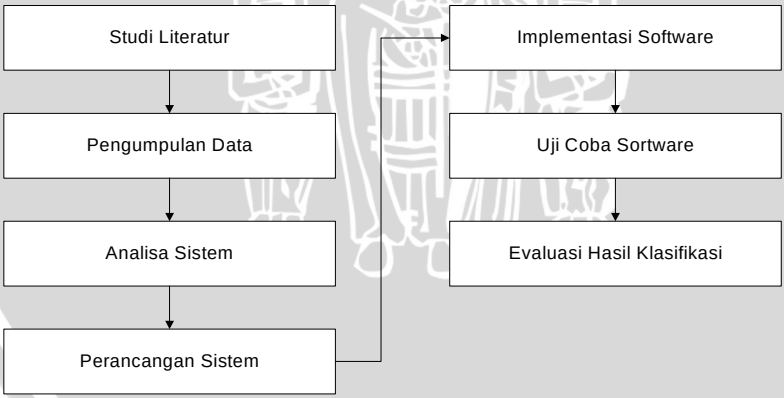
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian tentang *optimasi Fuzzy c-means* menggunakan *Particle Swarm Optimization* untuk deteksi anomaly serangan jaringan.

Penelitian dilakukan dengan tahapan-tahapan sebagai berikut :

1. Mempelajari literatur yang berhubungan *Intrusion detection system* dan metode yang akan digunakan yaitu *Fuzzy c-means* dan *Partikel Swarm Optimization*
2. Mengumpulkan data-data serangan jaringan dari KDDCUP 1999 yang telah dikelompokan berdasarkan jenis serangan.
3. Menganalisa dan merancang *Fuzzy c-means* dan *Partikel Swarm Optimization.*
4. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan.
5. Uji coba perangkat lunak *Fuzzy c-means dan Partikel Swarm Optimization* dengan memasukkan dataset aktivitas jaringan komputer.
6. Evaluasi hasil deteksi serangan jaringan yang dibuat oleh sistem.

Langkah-langkah penelitian dapat dilihat pada gambar 3.1



Gambar 3.1 Langkah-Langkah Penelitian

3.1 Data Penelitian

3.1.1 Studi Literatur

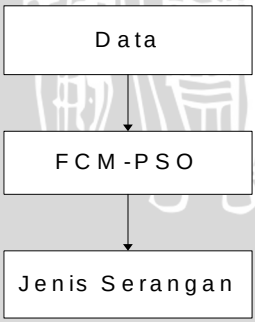
Dalam penelitian ini dibutuhkan studi literatur untuk merealisasikan tujuan dan penyelesaian masalah. Teori-teori mengenai serangan pada jaringan komputer, konsep *Fuzzy c-means* dan *Partikel Swarm Optimization* digunakan sebagai dasar penelitian yang diperoleh dari buku, jurnal dan browsing dari internet. Kemudian data yang diperoleh diubah sehingga dapat digunakan untuk analisis. Setelah dianalisis maka dapat di implementasikan ke dalam program.

3.1.2 Data Yang Digunakan

Data yang digunakan dalam penelitian ini adalah *dataset* record aktivitas dalam jaringan komputer yang diambil dari KDD CUP 1999. Terdapat 2 jenis *dataset*. Data yang pertama digunakan untuk training. Sedangkan data yang kedua adalah data untuk uji coba hasil deteksi.

3.2 Deskripsi Umum Sistem

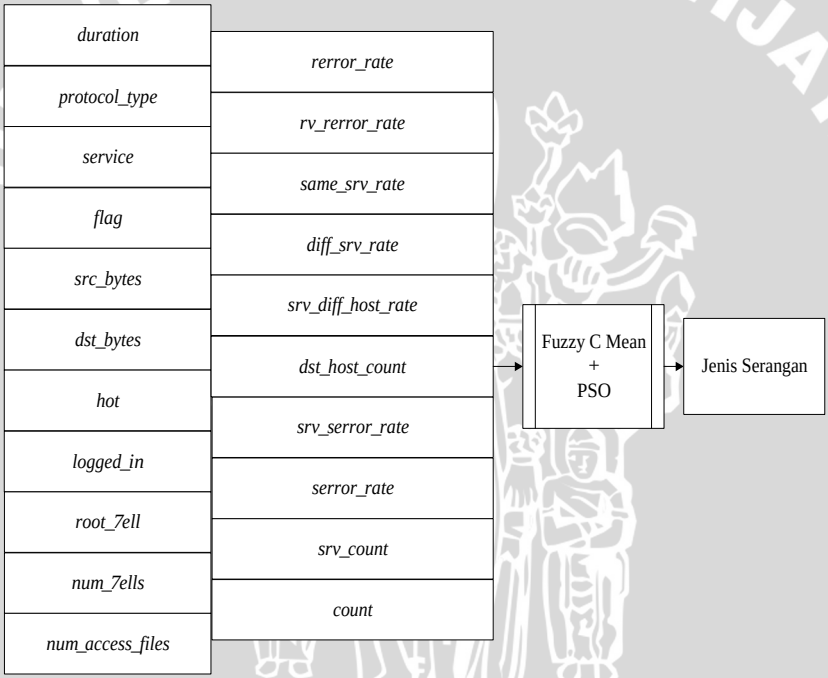
Secara umum sistem yang dibangun adalah perangkat lunak untuk mendeteksi serangan jaringan. Hasil deteksi berupa jenis serangan yang sedang dilakukan berdasarkan input yang diberikan. Jika aktivitas pada jaringan komputer sesuai dengan pola pada data hasil training maka dideteksi sebagai jenis aktivitas yang terdekat dengan pola tersebut. Secara umum tahapan klasifikasi seperti gambar 3.2.



Gambar 3.2 Tahapan training data

3.3 Desain Sistem

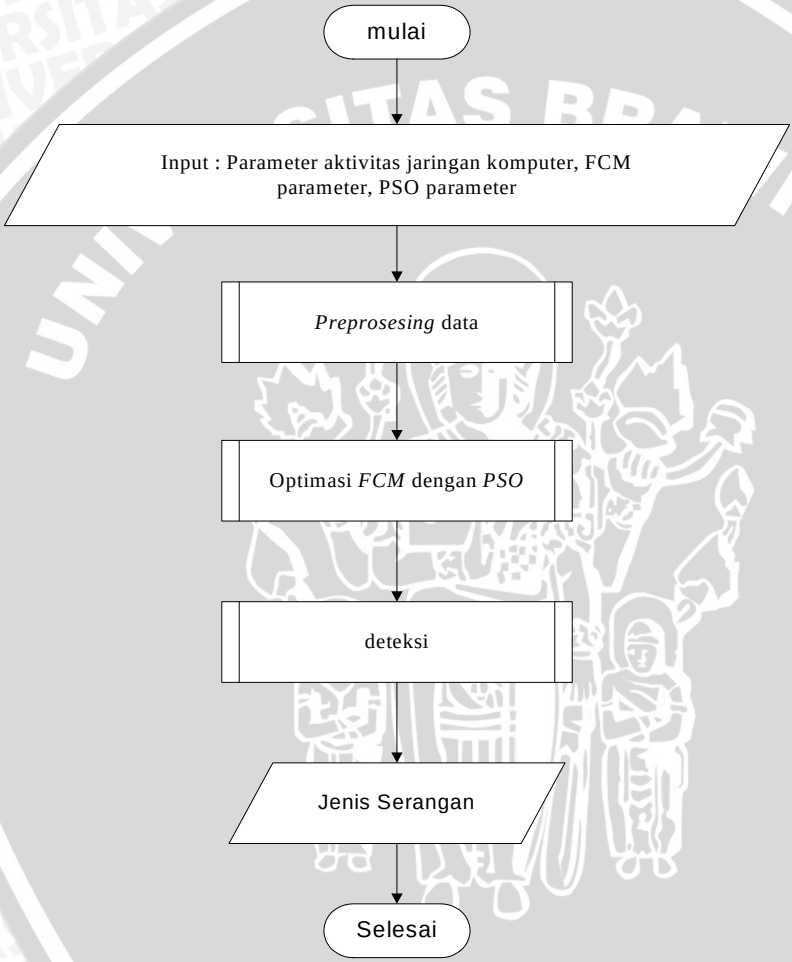
Penentuan jenis serangan merupakan proses aktualisasi dari *input* yang diberikan kepada *output* dengan menggunakan *Fuzzy C-Mean Clustering*. Elemen-elemen dalam penentuan jenis serangan meliputi beberapa parameter /feature. Dalam penelitian ini digunakan 29 feature yang memiliki nilai linguistic.Desain sistem untuk menentukan jenis serangan adalah seperti gambar 3.3.



Gambar 3.3. *Fuzzy c-means* untuk mendeteksi serangan jaringan

3.4 Perancangan Klasifikasi Data

Dalam perancangan fuzzy model ini akan dibuat *flowchart* seperti pada gambar 3.4.



Gambar 3.4. Perancangan *Fuzzy c-means*

Flowchart proses klasifikasi terhadap data-data serangan jaringan dapat dilihat pada gambar dan langkah-langkah sebagai berikut :

1. Mulai
2. Menginputkan *parameter* aktifitas serangan jaringan komputer, parameter *FCM* dan parameter *PSO*.
3. Preprosesing data untuk menormalisasikan data yang memiliki rentang variabel yang terlalu besar.
4. Mengklasifikasikan data menjadi beberapa jenis serangan jaringan menggunakan *fuzzy C-mean clustering* yang dioptimasi dengan *PSO*.
5. Proses deteksi merupakan pengujian hasil klasifikasi terbaik.
6. Selesai

3.5 Preprosesing Data

Pada proses awal data yang memiliki rentang variabel terlalu besar akan distandarkan untuk mengurangi kesalahan pada proses klasifikasi. Terdapat 3 feature yang akan dirubah kedalam nilai numerik yaitu *protocol type, service type,* dan *flag*

Data diperoleh dari beberapa penelitian yang dilakukan oleh beberapa peneliti. Data yang disediakan terdiri dari data untuk proses training dan data untuk uji coba hasil klasifikasi. Dataset untuk proses training telah diklasifikasin berdasarkan jenis serangan sedangkan data untuk proses uji coba merupakan data mentah

variabel *protocol type* merupakan variable *symbolic*. Pada penelitian ini batasan jenis protocol yang digunakan adalah *tcp, udp,* dan *icmp*. Variable jenis protocol diubah kedalam nilai numeric untuk proses klasifikasi dengan ketentuan seperti pada tabel 3.1.

Table 3.1. nilai *numeric* dari variable *protocol type*

No	*protocol type*	Nilai Numerik
1	*Tcp*	1
2	*Udp*	2
3	*Icmp*	3

Variable *service type* merupakan variable *symbolic*. Pada penelitian ini batasan *service type* yang digunakan adalah *smtp, ecr_i, imap4, telnet, ftp, ftp_data, private, time*. Variable *service type* diubah dalam nilai numeric untuk proses klasifikasi dengan ketentuan seperti pada tabel 3.2

Table 3.2 nilai numeric dari variable *service type*

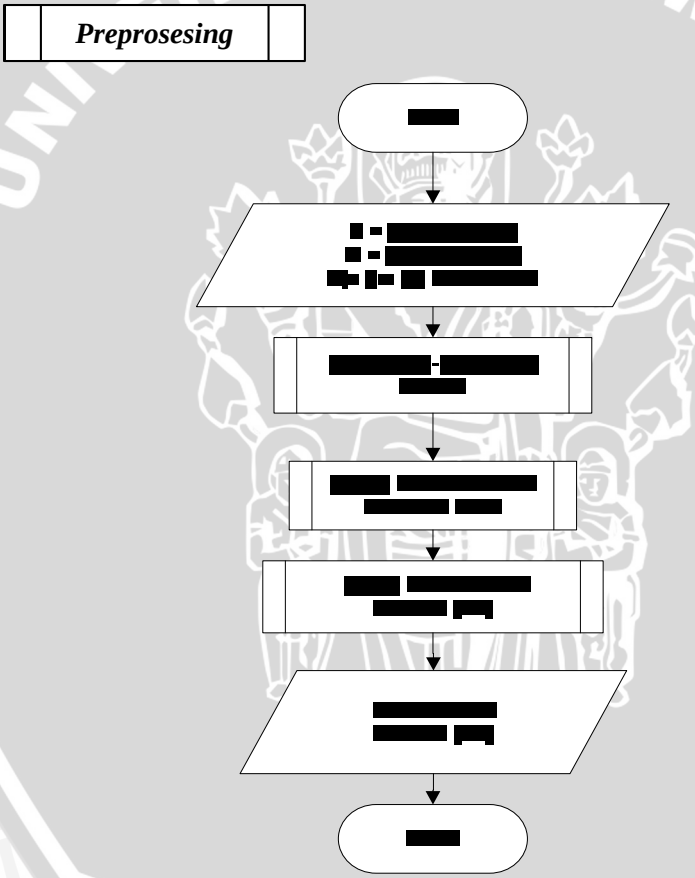
No	*Service type*	Nilai numeric
1	*Smtp*	1
2	*ecr_i*	2
3	*imap4*	3
4	*telnet*	4
5	*ftp*	5
6	*ftp_data*	6
7	*Private*	7
8	*Time*	8

variabel *flag* merupakan variable *symbolic*. Jenis *flag* yang digunakan dalam penelitian ini meliputi *SF, S1, REJ, S2, S0, S3, RSTO, RSTR, RSTO, S0, OTH, SH*. Variable *flag* diubah dalam nilai numeric untuk proses klasifikasi dengan ketentuan seperti pada tabel 3.3.

Table 3.3. nilai *numeric* dari variable *flag type*

No	*Flag type*	Numerik
1	*SF*	1
2	*S1*	2
3	*REJ*	3
4	*S2*	4
6	*S3*	6
7	*RSTR*	7
8	*RSTO*	8
10	*OTH*	10
11	*SH*	11

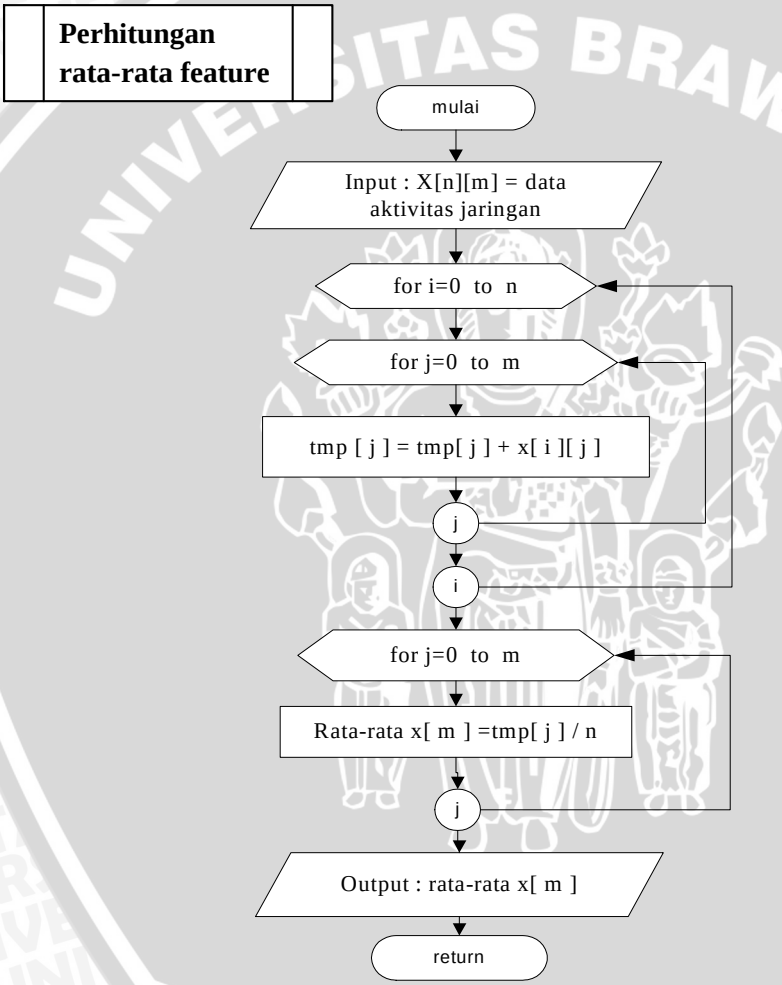
Variable dst_byte, serror_rate, dan dst_*host*_count mempunyai nilai continu dengan *range* 0 sampai 511 (*count*), 0 sampai 5203179 (*dst_byte*), 0 sampai 255 (*dst_host_count*). Perbedaan *range* nilai yang cukup besar antara variable *continue* dengan variable *numeric* akan menyebabkan hasil klasifikasi tidak maksimal. Maka perlu adanya *preprosing* data untuk mengubah data agar lebih mudah diklasifikasikan. Preprosesing terdiri dari 3 bagian yaitu menghitung rata-rata setiap feature, menghitung *mean absolute deviation(MAD)*dan menghitung *standardized feature(SF).*
Flowchart untuk *preprosesing* data adalah seperti pada gambar 3.5.



Gambar 3.5 *Flowchart* preprosesing data

3.5.1 Perhitungan rata-rata feature

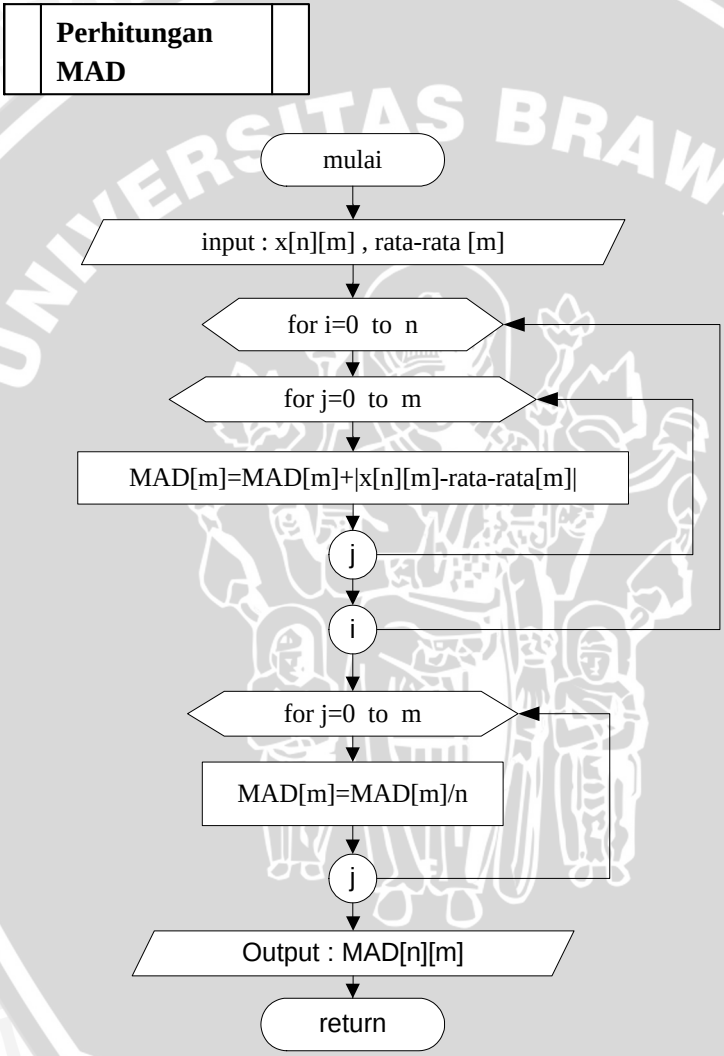
Proses ini digunakan untuk menghitung rata-rata setiap *feature* yang akan digunakan dalam proses selanjutnya. dalam proses ini digunakan beberapa variable yaitu *tmp dan* $\bar{x}$ *[j]*. *tmp* adalah tempat penyimpanan sementara jumlah semua feature. $\bar{x}$ [j] adalah rata-rata feature.*Flowchart* untuk *hitung rata-rata feature* data adalah seperti pada gambar 3.6.



Gambar 3.6 *Flowchart* perhitungan *mean* setiap *feature*

3.5.2 Perhitungan MAD

Proses ini digunakan untuk menghitung *mean absolute derivation* yang akan digunakan dalam proses selanjutnya. *mean absolute derivation* adalah jarak data asli dengan rata-rata setiap feature. *Flowchart* untuk *hitung MAD* adalah seperti gambar 3.7.

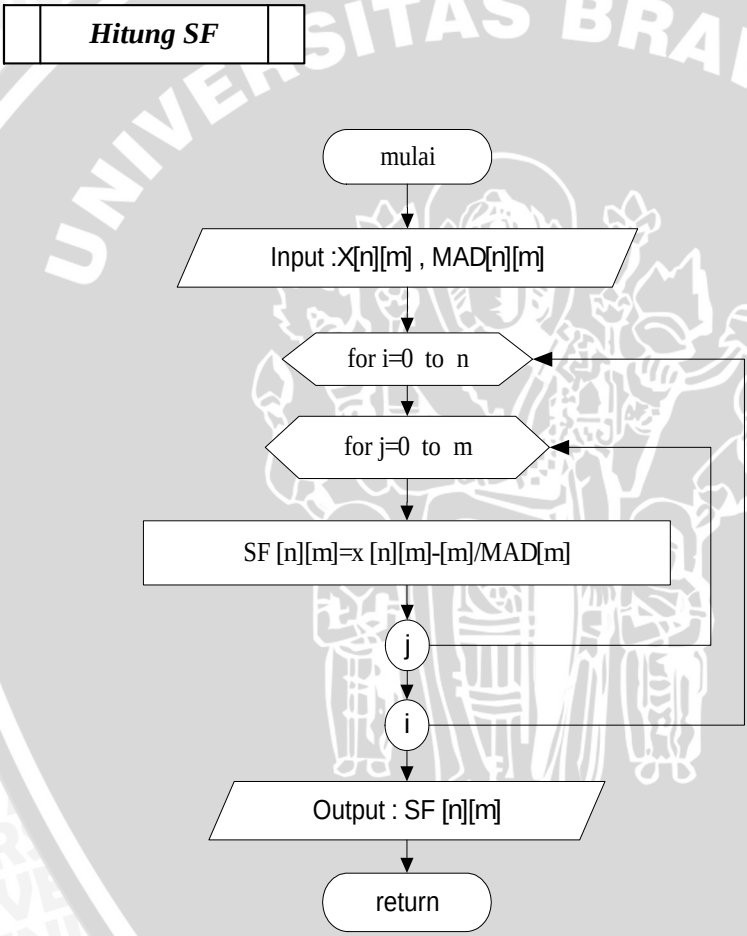


Gambar 3.7 *Flowchart* perhitungan *MAD* setiap *feature*

3.5.3 Perhitungan *standart feature(SF)*

proses ini digunakan untuk menghitung *standart feature* yang akan digunakan dalam proses klastering. Standart feature merupakan data record yang sudahdistandarkan antara feature satu dengan feature lainnya sehingga tingkat kesalahan pada proses klastering dapat diperkecil.

Flowchart untuk Perhitungan *standart feature(SF)* adalah seperti pada gambar 3.8.



Gambar 3.8 *Flowchart* perhitungan *Standarized feature*

3.6 Proses klastering data dengan PSO

Proses klastering data adalah proses untuk mengklasifikasikan data ke dalam beberapa kelas yang telah ditentukan. Dengan menggunakan *Fuzzy C Mean clustering*, setiap data memiliki derajat keanggotaan pada setiap kelas / klaster. Pada penilitian ini proses klasifikasi akan dioptimasi menggunakan *Partikel Swarm Optimation.*

Setiap partikel dalam algoritma PSO direpresentasikan sebagai proses klastering data. Setiap partikel pada setiap generasi memiliki nilai *fitness*. Jika nilai *fitness* lebih besar daripada nilai *gbest* maka posisi tersebut dinyatakan sebagai posisi terbaik.

Proses awal metode *FCM* dengan menggunakan *PSO* adalah menginisialisasi nilai derajat keanggotaan setiap data pada setiap klaster untuk setiap partikel. Proses inisialisasi derajat kenggotaan dilakukan secara random pada saat pertama kali. Selanjutnya nilainya akan diupdate pada setiap generasi. Nilai derajat keanggotaan untuk setiap data memiliki *range* antara 0 dan 1.

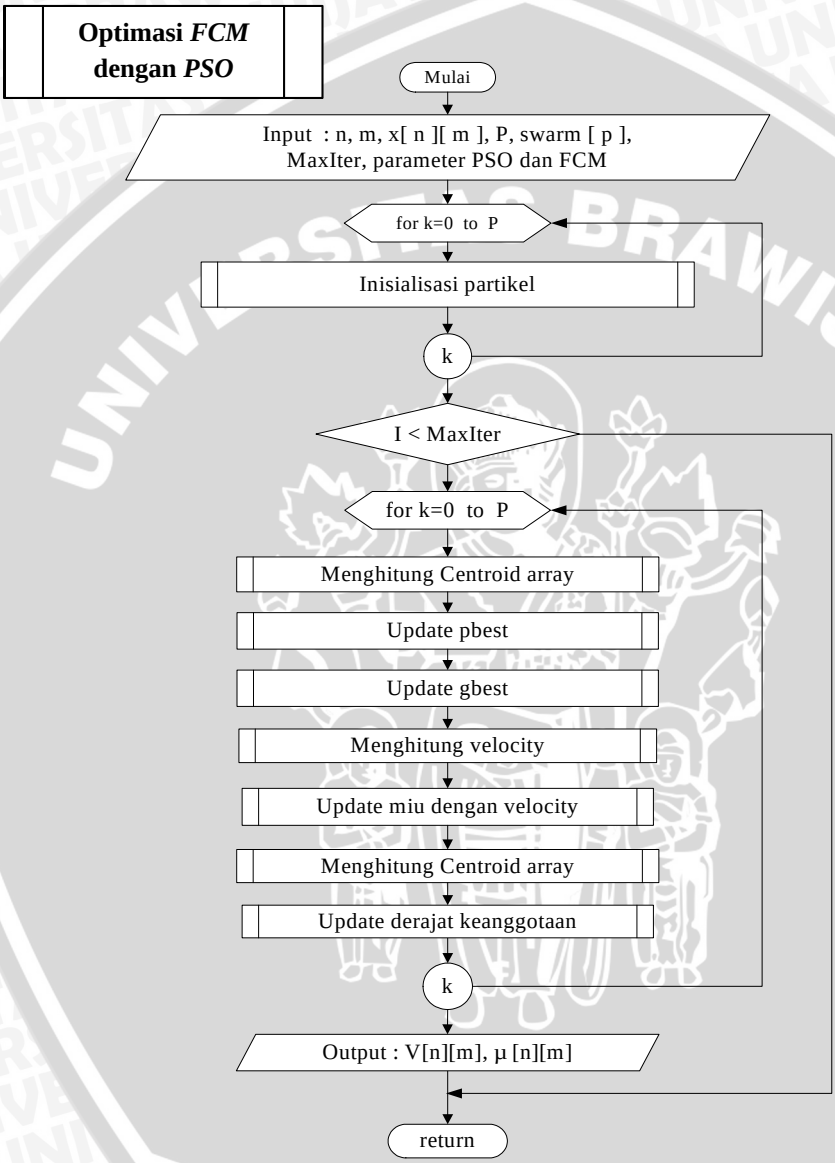
Proses kedua adalah menginisialisasi komponen awal dari setiap partikel *PSO* yaitu *c1, c2, q, w, pbest,* dan *gbest*. Komponen tersebut dimiliki oleh setiap partikel. Sedangkan *gbest* merupakan variable global. Semua komponen tersebut akan berubah pada setiap iterasi membentuk kombinasi terbaik yang akan mengarah pada *gbest.*

Proses ketiga adalah menentukan centroid array. Centroid array merepresentasikan pusat setiap feature di setiap klaster pada setiap partikel. Misalnya terdapat n input data, m jumlah cluster dan p jumlah partikel. Maka akan terbentuk centroidd array dengan dimensi *n x m x p.*

Proses keempat adalah menghitung nilai *fitness* setiap partikel. Nilai *fitness* mengindikasikan solusi terbaik dari semua populasi. Nilai *fitness* partikel didapatkan dari nilai konstanta dibagi dengan nilai objektif. Perhitungan untuk menentukan nilai fitness terdapat pada bab 2.

Proses kelima adalah meng-update nilai *velocity* untuk menentukan kombinasi terbaik. Perhitungan untuk meng-update nilai velocity terdapat pada bab 2. Nilai *velocity* akan mempengaruhi nilai parameter yang digunakan untuk menentukan *gbest.*

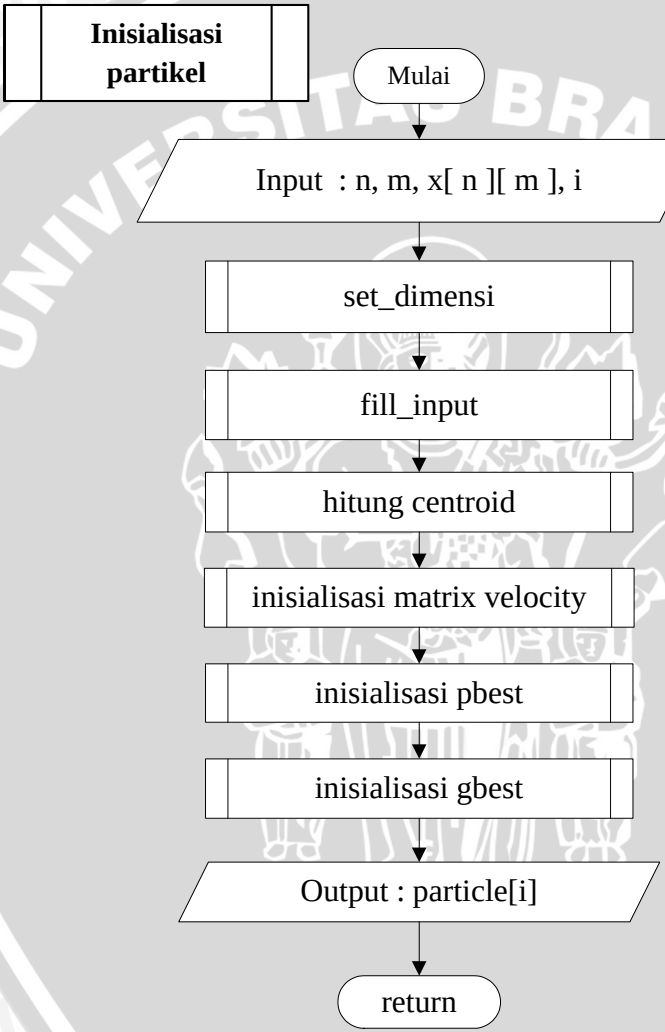
Flowchart proses klastering data dengan PSO dapat dilihat pada gambar 3.9.



Gambar 3.9 *flowchart* Optimasi *Fuzzy c-means* dengan *PSO*

3.6.1 Inisialisasi partikel

Proses inisialisasi merupakan proses awal saat partikel pertama kali dibentuk. Pada proses inisialisasi partikel ini meliputi *set_dimensi(), fill_input(), inisialisasi_u(), hitung_centroid(), inisialisasi_velocity(), inisialisasi_pbest(),* dan *inisialisasi_gbest().*

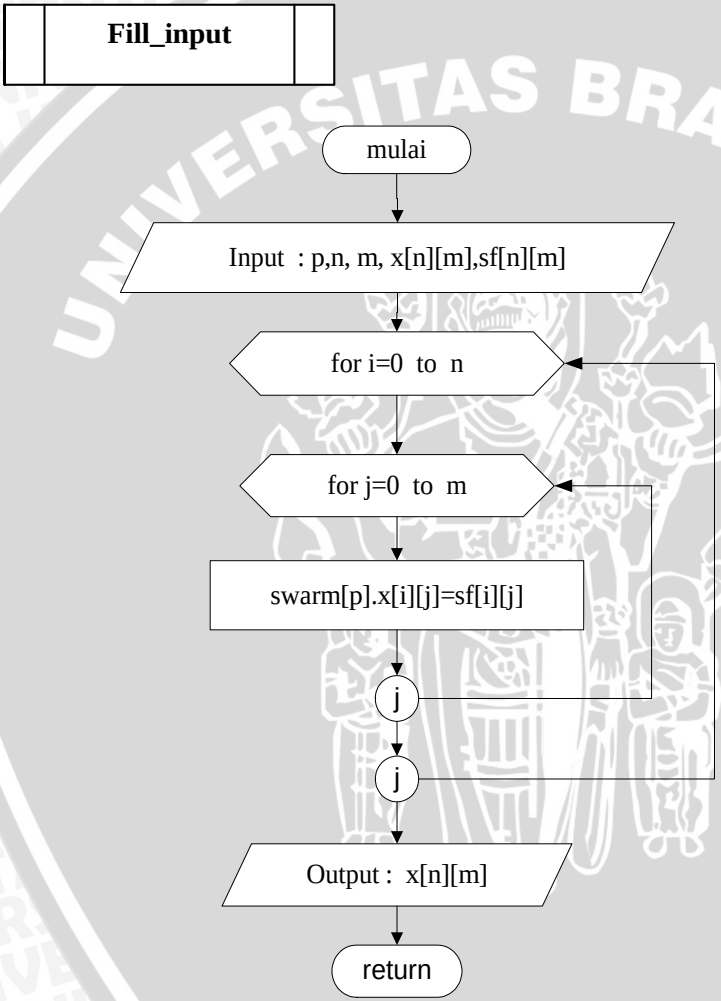


Gambar 3.10 *flowchart* inisialisasi partikel

3.6.1.1 Fill input

Proses fill input merupakan proses mengisi variabel array x[n][m] dengan nilai input dari file .txt yang sudah melewati proses standarisasi.

Flowchart untuk *fill_input* adalah seperti gambar 3.7.

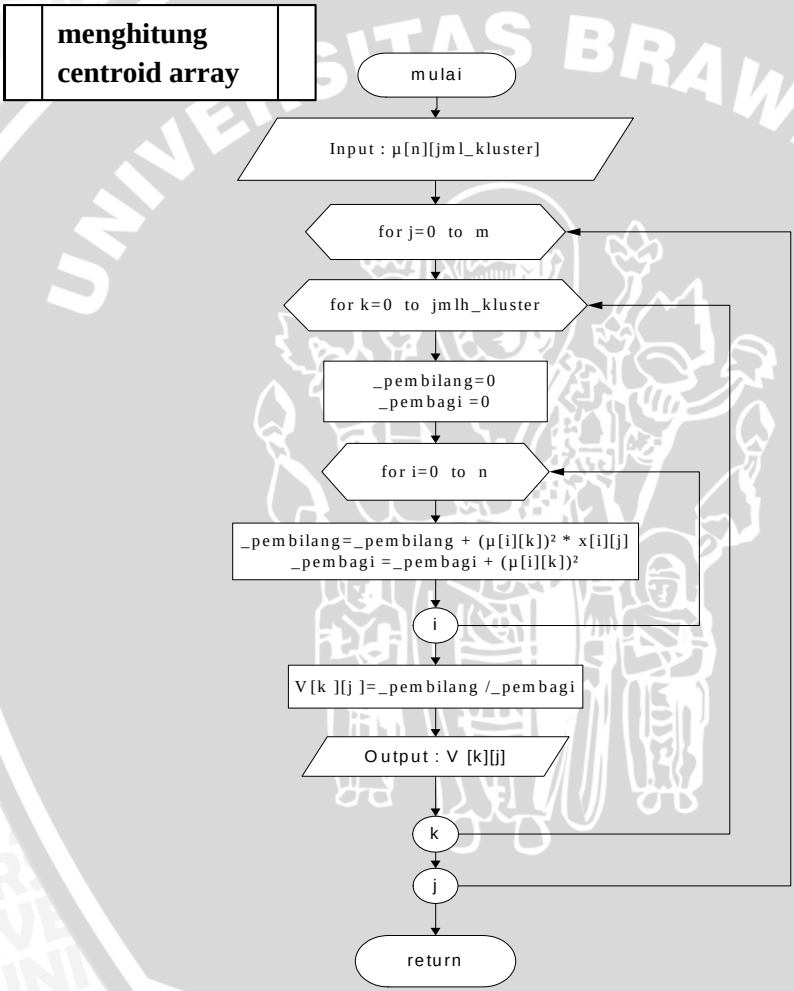


Gambar 3.11 *flowchart* fill input

3.6.1.2 Hitung centroid array

Proses ini digunakan untuk menghitung nilai titik *centroid* (pusat) setiap feature pada setiap kelas/klaster. Centroid digunakan untuk meng-update nilai derajat keanggotaan. V[k][j] merupakan nilai centrid klaster ke-k feature ke-j.

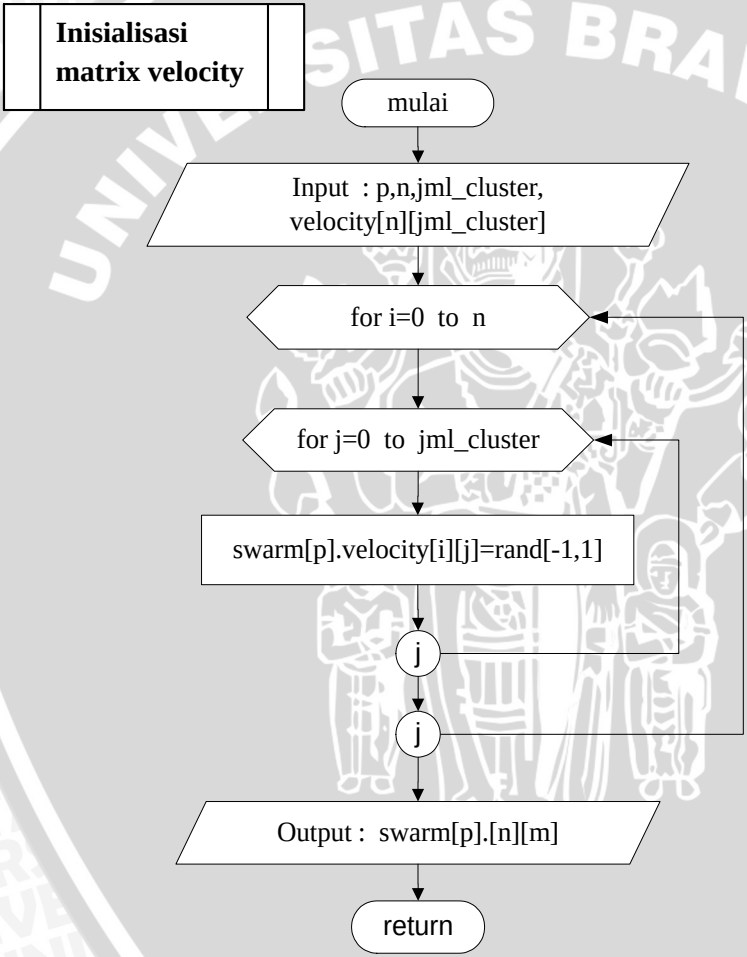
flowchart untuk perhitungan *centroid* array adalah seperti pada gambar 3.11



Gambar 3.12 *flowchart* perhitungan *centroid array*

3.6.1.3 Inisialisasi matrix velocity

Proses inisialisasi nilai matrix velocity digunakan untuk memberikan nilai awal pada matrix velocity secara acak dengan rentang nilai antara -1 dan 1. Nilai matrix velocity pada setiap iterasi akan berubah dan akan mengarahkan partikel pada global optimum.
flowchart untuk perhitungan *centroid* array adalah seperti pada gambar 3.11

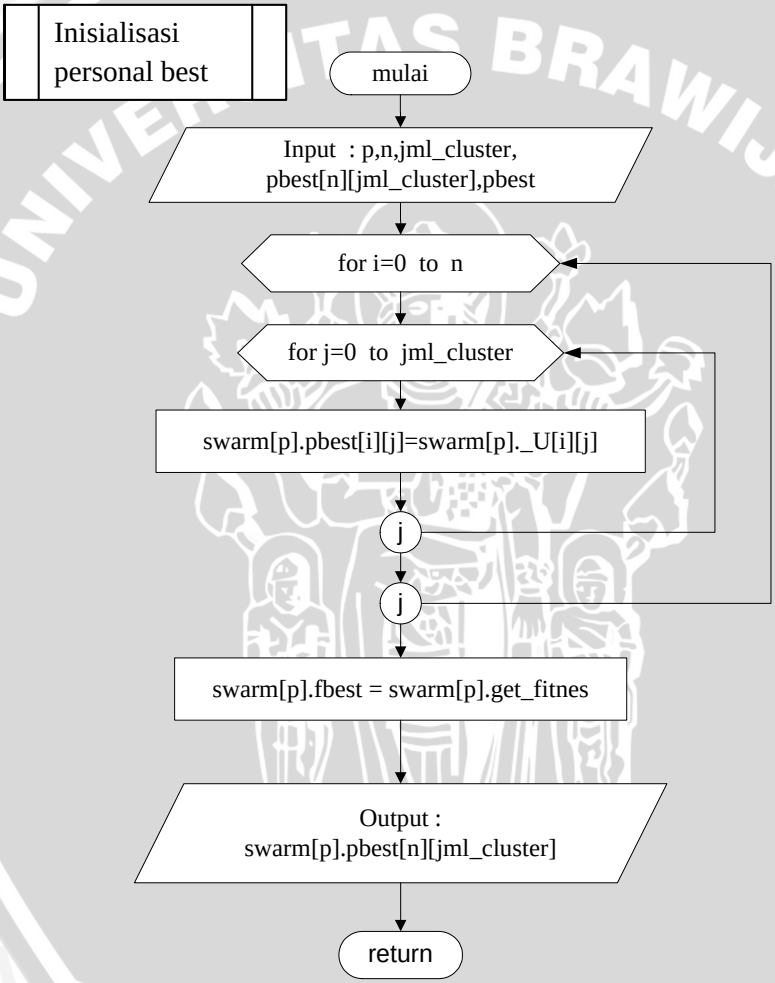


Gambar 3.13 *flowchart* inisialisasi matrix *velocity*

3.6.1.4 Inisialisasi pest

Proses ini digunakan untuk menghitung nilai personal best masing-masing partikel pada saat pertama kali partikel diinisialisasi. Pada generasi selanjutnya nilai pbest akan di-*update* untuk mencapai gbest

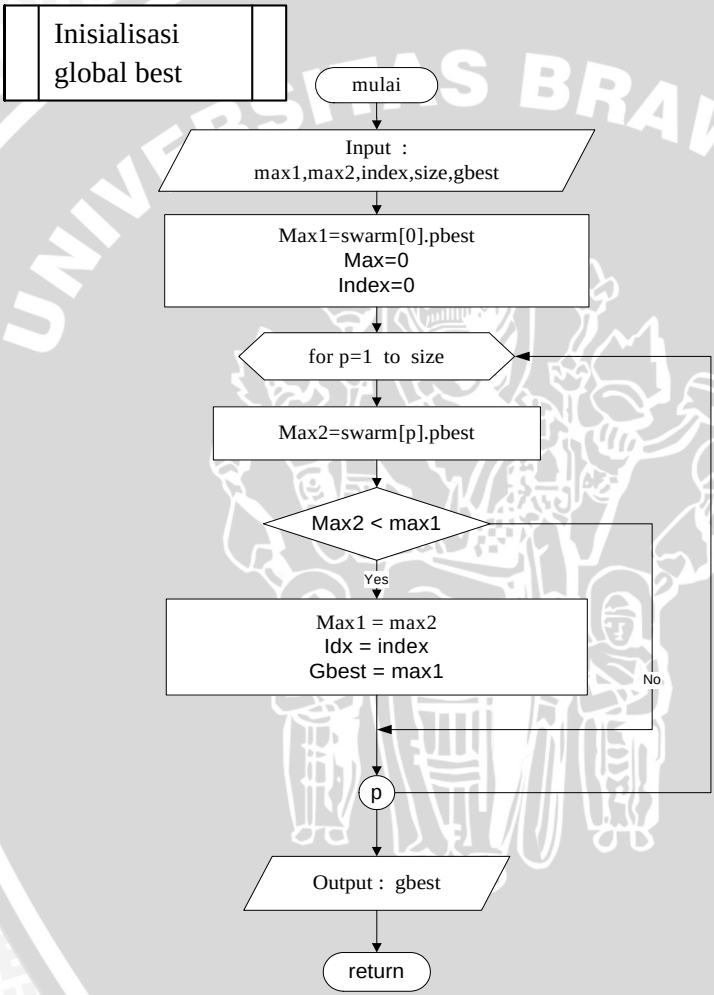
flowchart untuk perhitungan *centroid* array adalah seperti pada gambar 3.11



Gambar 3.14 *flowchart* inisialisasi *pbest*

3.6.1.5 Inisialisasi gbest

Proses ini digunakan untuk menghitung nilai global best dari semua partikel pada suatu swarm saat pertama kali partikel diinisialisasi. Pada generasi selanjutnya nilai gbest akan di-*update* dari nilai *pbest* *flowchart* untuk perhitungan *centroid* array adalah seperti pada gambar 3.11

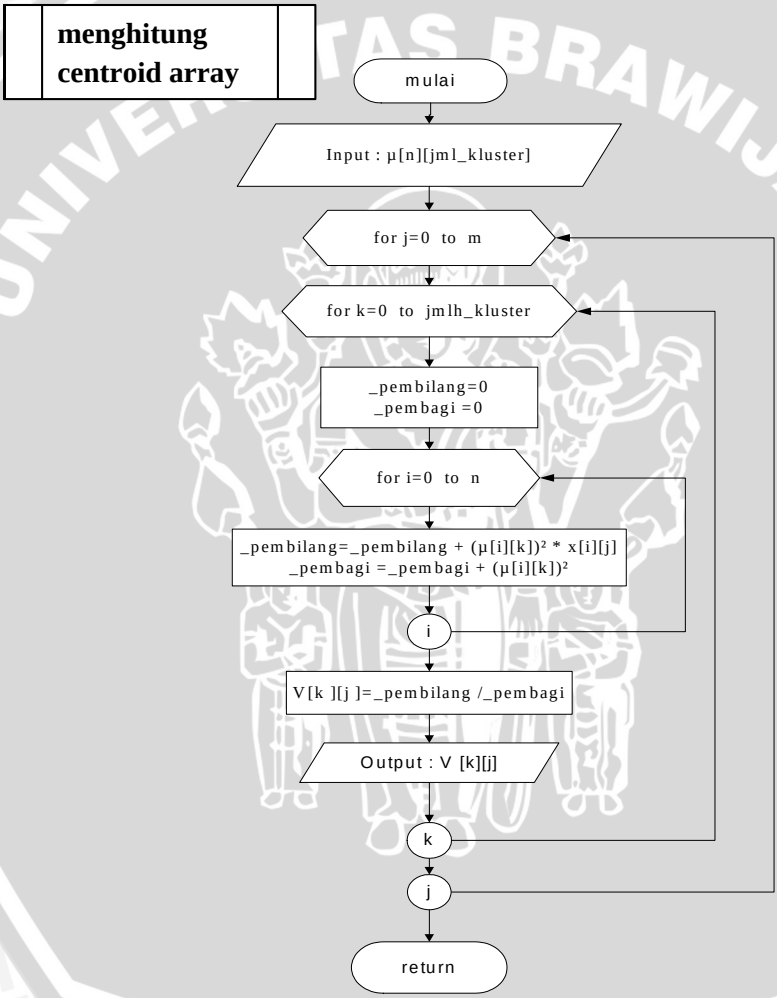


Gambar 3.15 *flowchart* inisialisasi *gbest*

3.6.2 Perhitungan centroid array

Proses ini digunakan untuk menghitung nilai titik *centroid* (pusat) setiap feature pada setiap kelas/klaster. Centroid digunakan untuk meng-update nilai derajat keanggotaan. V[k][j] merupakan nilai centrid klaster ke-k feature ke-j.

flowchart untuk perhitungan *centroid* array adalah seperti pada gambar 3.11

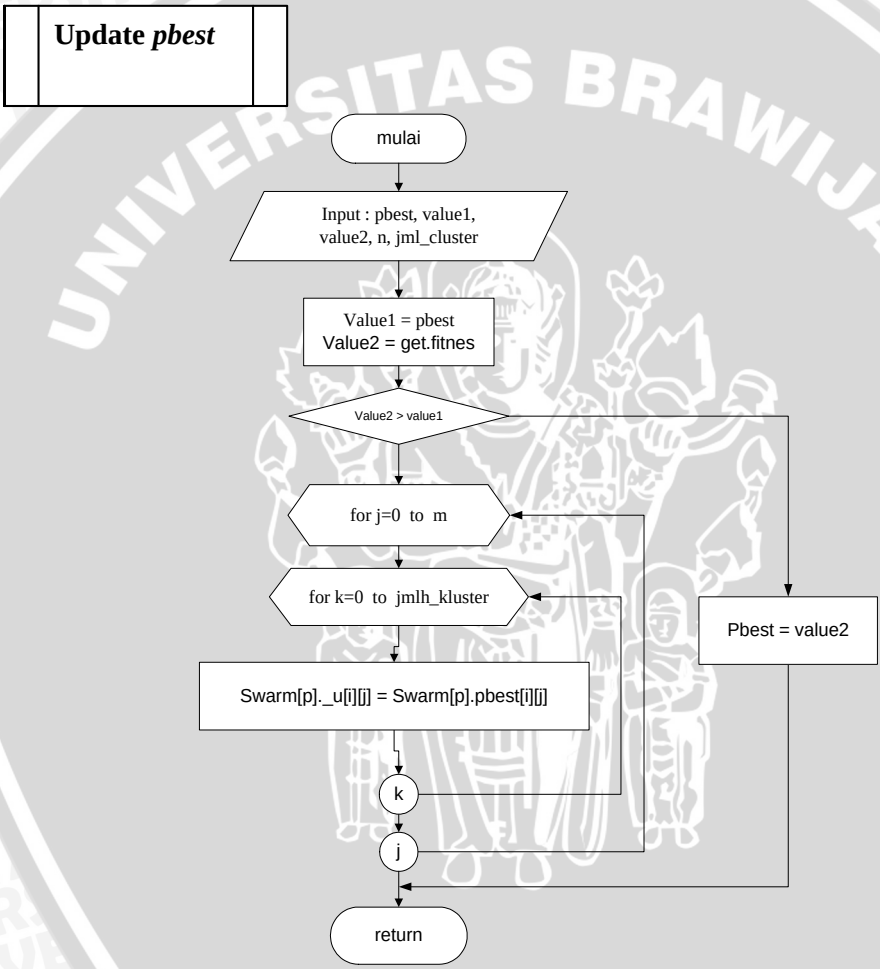


Gambar 3.16 *flowchart* perhitungan *centroid array*

3.6.3 Update pbest

Proses ini digunakan untuk menghitung kembali nilai personal best. Nilai pbest diperoleh dari partikel baru yang telah di-update menggunakan nilai velocity

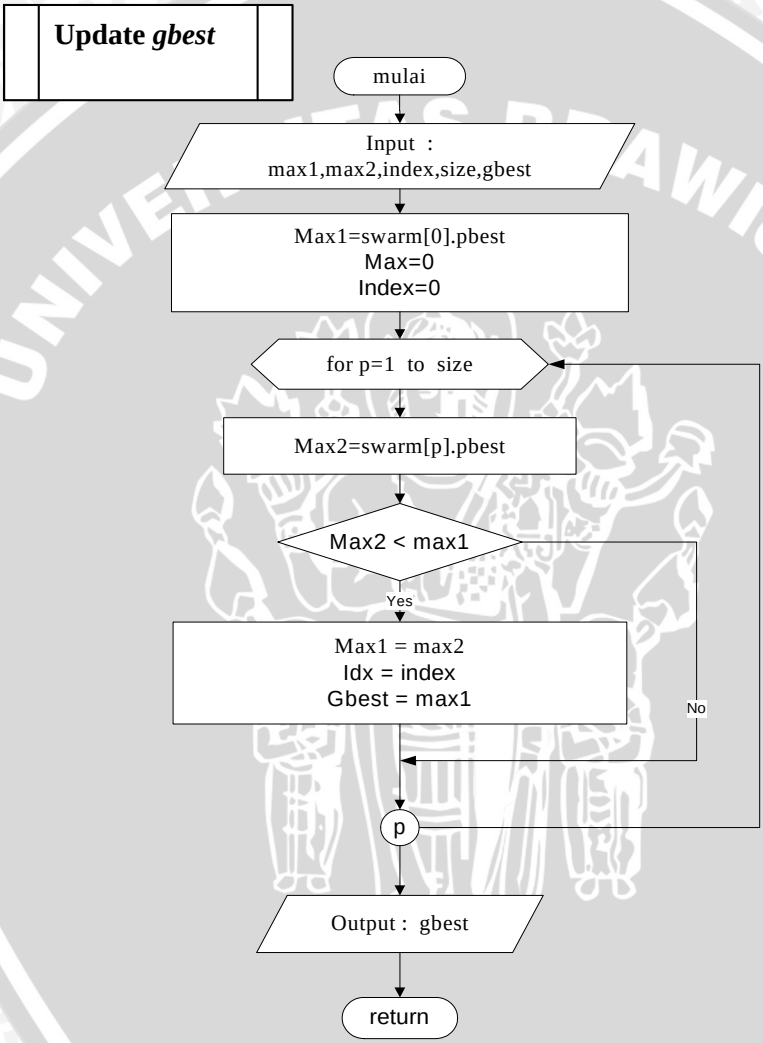
flowchart untuk perhitungan *centroid* array adalah seperti pada gambar 3.11



Gambar 3.17 *flowchart* update *pbest*

3.6.4 Update gbest

Proses ini digunakan untuk meng-update nilai gbest pada setiap generasi. Nilai gbest baru didapatkan dengan membandingkan semua nilai pbest baru.

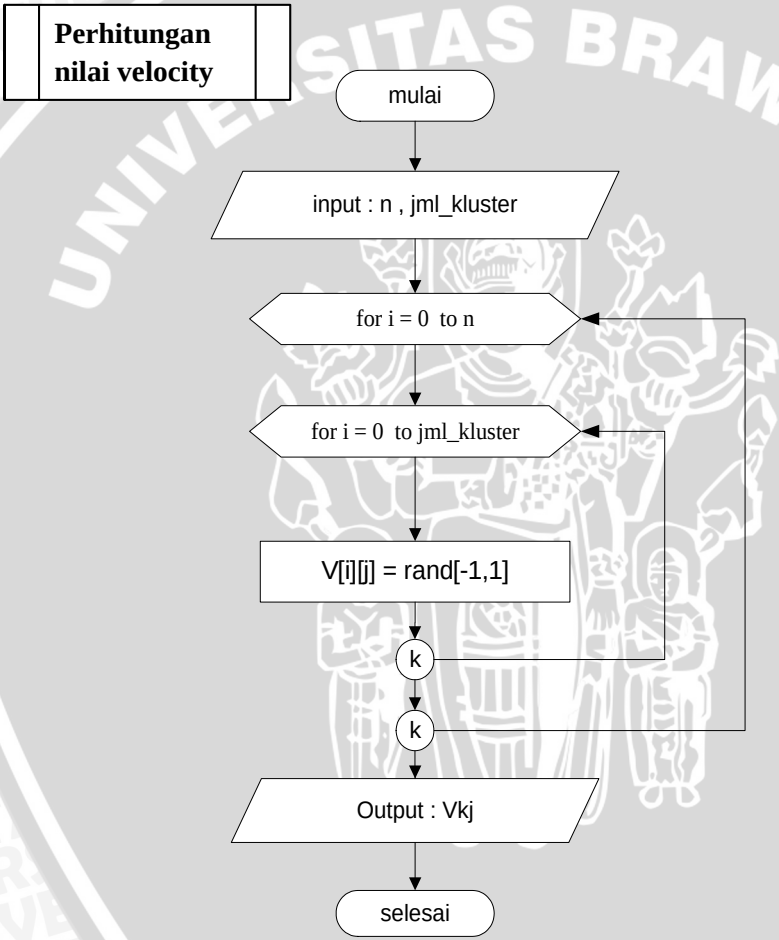


Gambar 3.18 *flowchart update gbest*

3.6.5 Perhitungan nilai velocity

Proses ini digunakan untuk mengarahkan partikel ke optimum global. *Inertia weight* digunakan untuk menyeimbengkan antara *gbest* dan *pbest.* Nilai velocity akan mengubah 4 parameter (q , c_1, c_2, dan σ) pada setiap generasi.

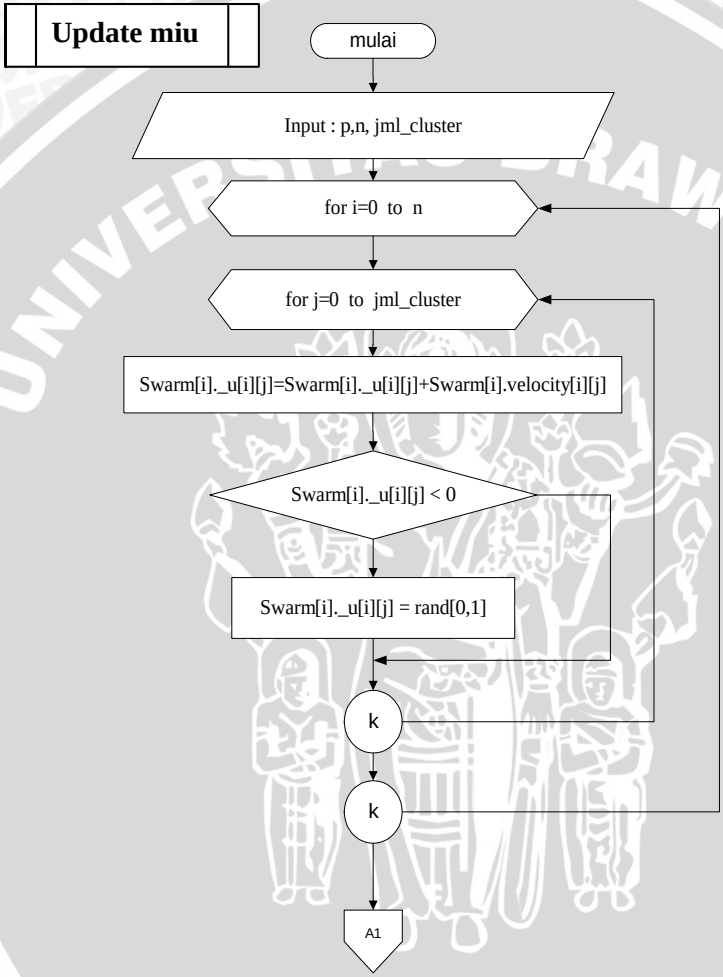
Flowchart untuk perhitungan nilai velocity adalah seperti pada gambar 3.13



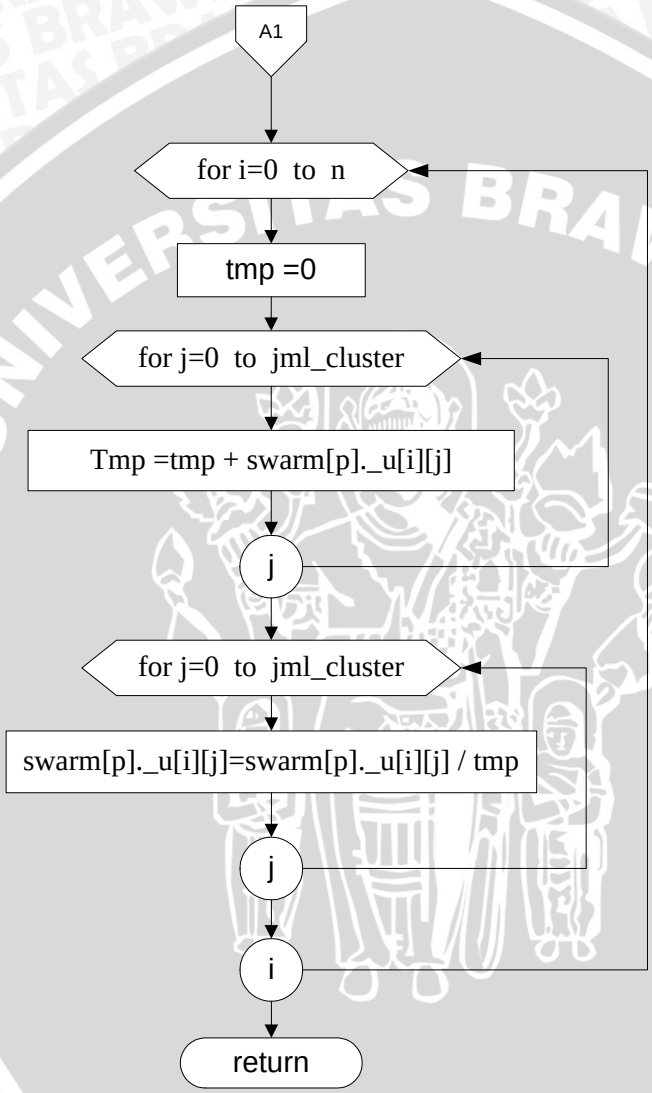
Gambar 3.19 *flowchart* perhitungan nilai *velocity*

3.6.6 Update miu menggunakan velocity

Proses ini digunakan untuk mengarahkan nilai derajat keanggotaan setiap pertikel pada nilai optimum.



Gambar 3.20 *flowchart* update derajat keanggotaan bagian I

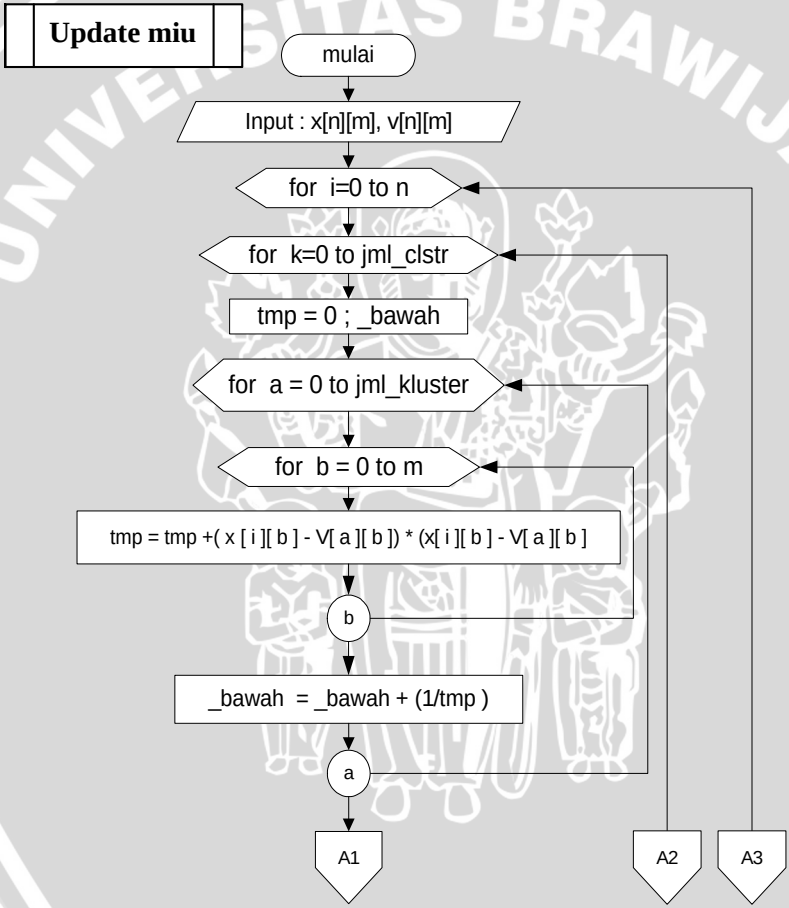


Gambar 3.21 *flowchart* update derajat keanggotaan bagian II

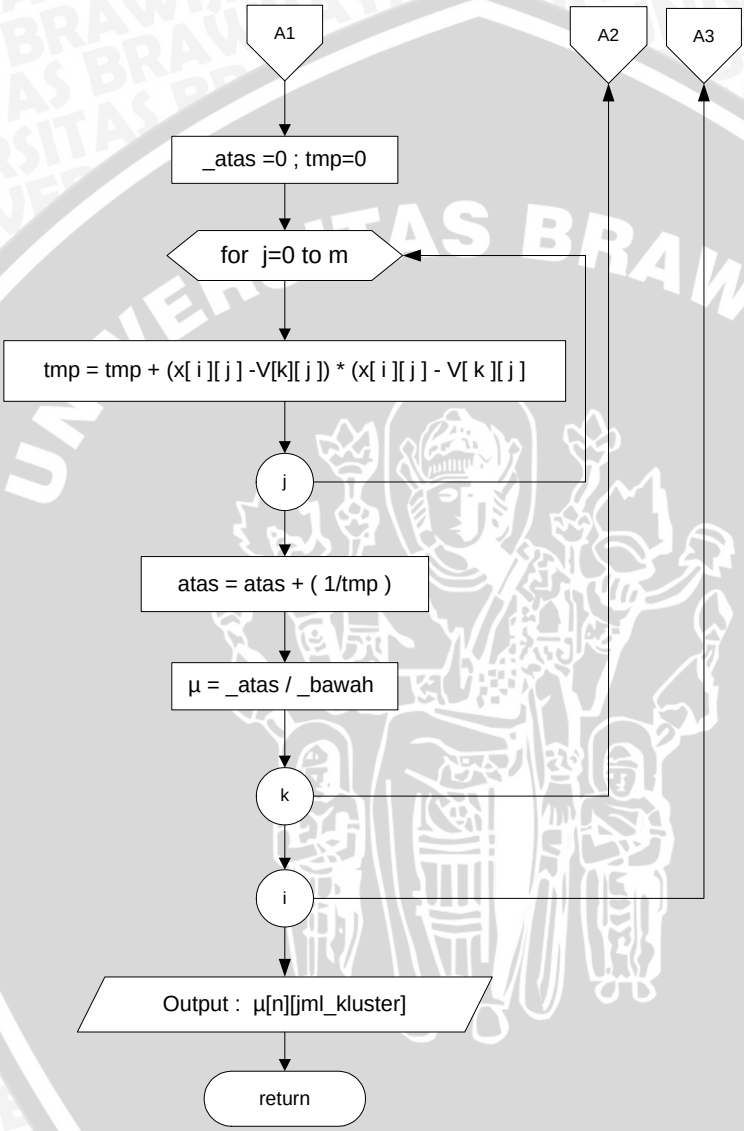
3.6.7 Update derajat keanggotaan

Proses ini digunakan untuk merubah nilai derajat keanggotaan setiap record data pada setiap klaster dengan rentang antara 0 sampai 1. Untuk perhitungan ini diperlukan beberapa variable yaitu *tmp,_atas,_bawah. tmp* digunakan sebagai penyimpanan sementara.

Flowchart untuk *update derajat keanggotaan* adalah seperti pada gambar 3.14.



Gambar 3.22 *flowchart* update derajat keanggotaan bagian I

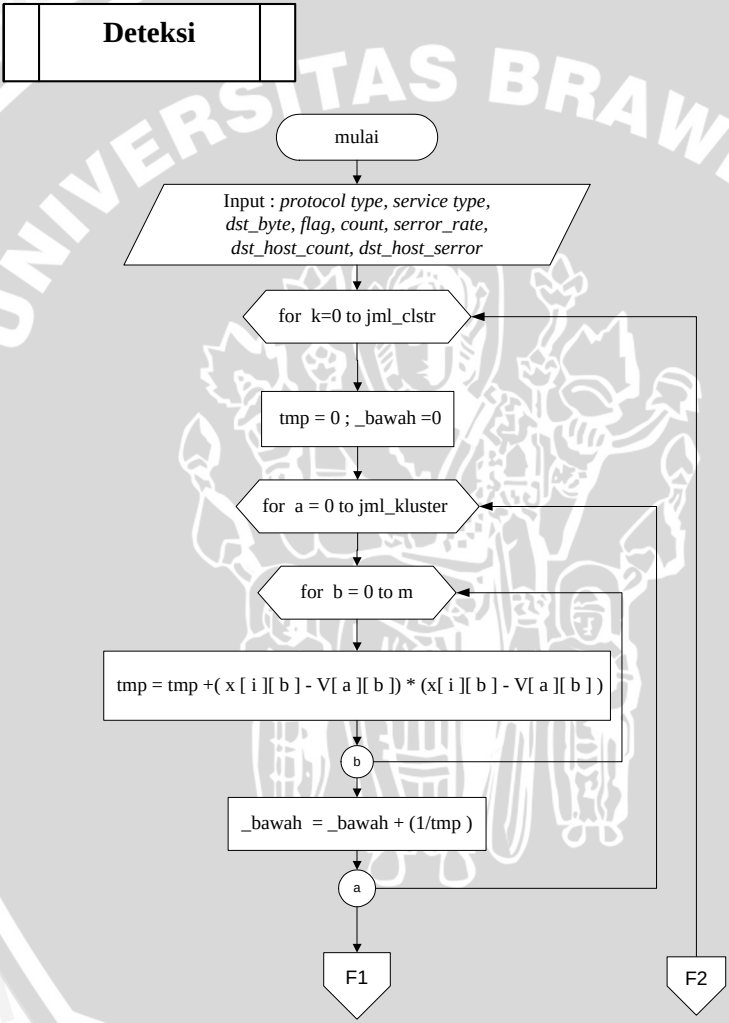


Gambar 3.23 *flowchart* update derajat keanggotaan bagian II

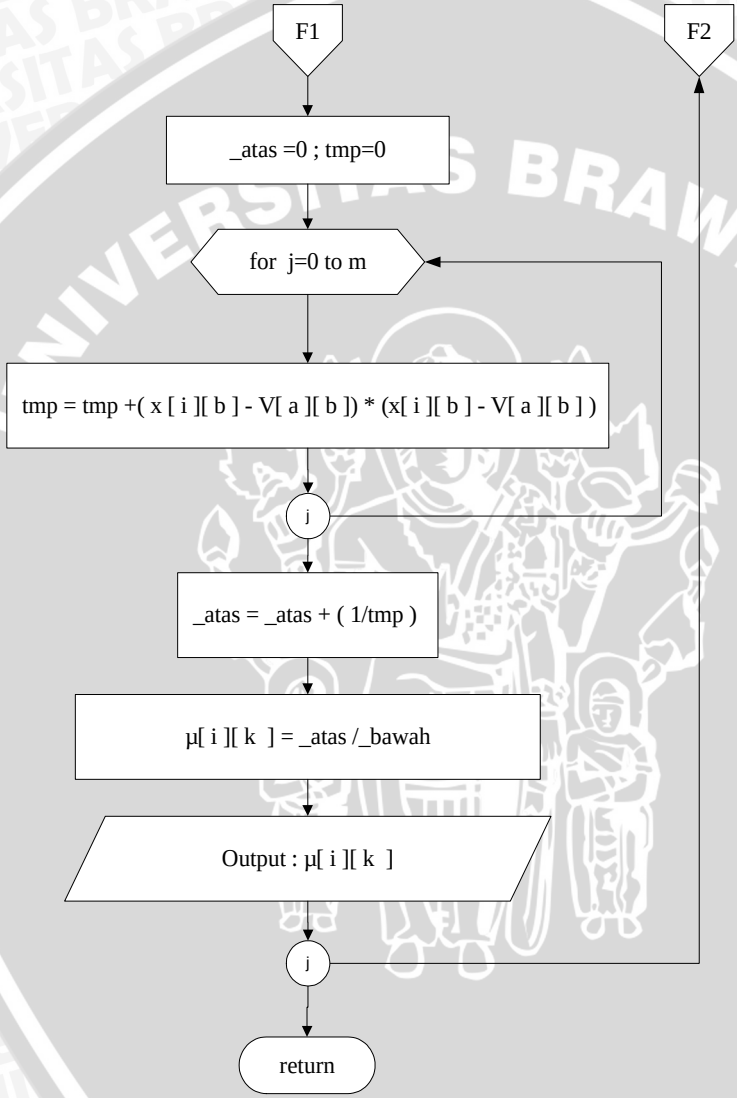
3.6.8 Proses deteksi

Proses deteksi merupakan tahap pengujian dari hasil klasifikasi menggunakan Fuzzy C Mean. Proses ini menggunakan data derajat keanggotaan yang sudah dioptimasi dengan menggunakan algoritma Partikel Swarm Optimization.

Flowchart untuk *proses deteksi* adalah seperti pada gambar 3.14.



Gambar 3.24 *flowchart* proses deteksi serangan bagian I



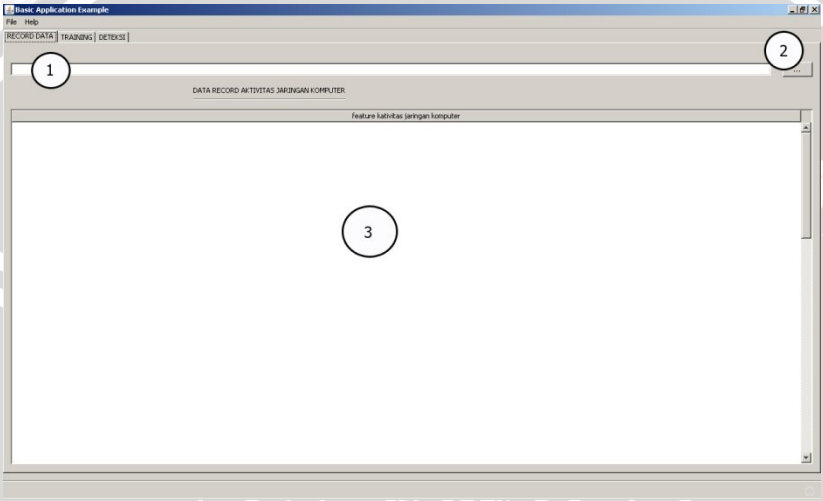
Gambar 3.25 *flowchart* proses deteksi serangan bagian II

3.7 Perancangan antar muka

Adapun beberapa form yang dapat dibuat yaitu :

3.7.1 Form load data

Form klastering digunakan untuk menginput record data, parameter FCm dan parameter. Kemudian dilakukan preprosesing data untuk menyeimbangkan nilai antar feature. Perancangan form untuk proses klastering adalah seperti pada gambar 3.13



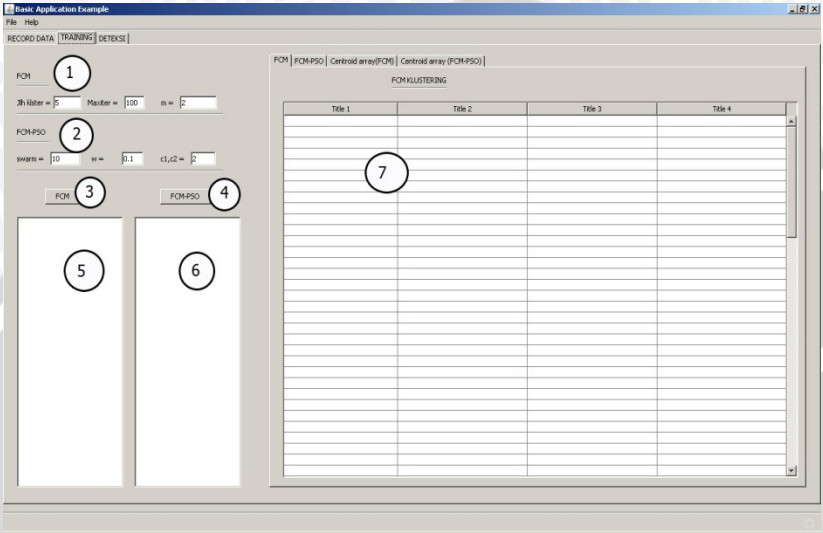
Gambar 3.13 *form load* data

Keterangan gambar :

1. Menampilkan direktori tempat file record aktivitas serangan aringan computer.
2. Me-*load* data akivitas serangan jaringan computer dan men-standarisasi data agar mudah ddilakukan proses klastering.
3. Menampilkan record data aktivitas jaringan computer yang telah distandarisasi

3.7.2 Form pengujian

Form pengujian digunakan untuk menampilkan record data yang sudah diklasifikasikan ke dalam beberapa klaster dan menguji beberapa record data baru. Perancangan form untuk proses pengujian adalah seperti pada gambar 3.14



Gambar 3.14 *form* klastering

Keterangan gambar :

1. Input komponen untuk klastering data menggunakan *Fuzzy c-means.*
2. Input komponen untuk klastering data menggunakan *Fuzzy c-means* dan dioptimasi menggunakan PSO.
3. Tombol FCM untuk memulai proses klastering menggunakan *Fuzzy c-means*
4. Tombol FCM untuk memulai proses klastering menggunakan FCM-PSO
5. Menampilkan nilai fitness hasil klasifikasi dengan FCM pada setiap iterasi / generasi.
6. Menampilkan nilai fitness hasil klasifikasi dengan FCM-PSO pada setiap iterasi / generasi.
7. Menampilkan hasil clustering dengan memberikan nilai derajat keanggotaan dan nilai centroid array

3.8 Contoh perhitungan

3.8.1 Proses Optimasi menggunakan PSO

Langkah pertama adalah menentukan parameter algoritma *Particle Swarm Optimizer* yang ditunjukan pada tabel 3.4 :

Table 3.4 parameter algoritma PSO

No	Symbol	Nilai	Keterangan
1	N	10	Jumlah partikel
4	C1	2.0	Koefisien akselarasi
5	C2	2.0	Koefisien akselarasi
6	R1	[0,1]	Nilai random pertama
7	R2	[0,1]	Nilai random kedua
8	ρ	2.0	Bobot
9	Q	2.0	Pangkat

Pada proses PSO setiap partikel merupakan proses klastering. Setiap partikel merupakan kemungkinan solusi dengan mengambil contoh dari proses 3.6.2

3.8.2 Proses klasifikasi menggunakan FCM

Misalnya terdapat rata-rata pendapatan 30 record data aktivitas jaringan komputer. Setiap record memiliki 21 variabel penanda setiap aktivitas. Dari 25 data tersebut akan diklasifikasikan ke dalam 5 klaster, satu untuk aktivitas normal dan 4 untuk jenis serangan. Parameter awal untuk algoritma FCM dapat dilihat seperti pada tabel 3.5.

Table 3.5 parameter algoritma FCM

No	Parameter	Nilai	Keterangan
1	C	5	
2	W	2	
3	MaxIter	100	
4	MaxError	$5 * 10^{-5}$	

Data aktivitas serangan jaringan yang terdiri dari 29 parameter dapat dilihat pada tabel 3.5.

Table 3.6 data parameter serangan jaringan komputer

No	duration	protocol_type	service	....	dst_host_srv_rerror_rate
1	0	1	1	...	1
2	0	1	1	...	1
3	0	1	1	...	1
4	0	1	1	...	1
5	0	1	1	....	1
6	0	1	1	...	1
7	0	1	1	...	1
8	0	1	1	...	1
9	0	1	1	...	1
10	0	1	1	....	1
11	0	1	1	...	1
12	0	1	1	...	1
13	0	1	1	...	1
14	0	1	1	...	1
15	0	1	1	....	1
16	0	1	1	...	1
17	0	1	1	...	1
18	0	1	1	...	1
19	0	1	1	...	1
20	0	1	1	....	1
21	0	1	1	...	1
22	0	1	1	...	1
23	0	1	1	...	1
24	0	1	1	...	1
25	0	1	1	....	1
26	0	1	1	....	1
27	0	1	1	...	1
28	0	1	1	...	1

29	0	1	1	...	1
30	0	1	1	...	1

Masing-masing kelas merupakan jenis aktivitas pada jaringan yang berbeda seperti terlihat pada tabel 3.6 :

Table 3.7 jenis aktivitas serangan jaringan komputer

No	Klaster	Aktivitas
1	C1	normal
2	C1	neptune
3	C3	smurf
4	C4	isweep
5	C5	pod

Derajat keanggotaan setiap *record* data pada setiap klaster ditentukan secara acak pada saat pertama kali. *Range* derajat keanggotaan setiap antara 0 dan 1. Jumlah derajat keanggotaan setiap record pada masing-masing klaster adalah 1. Nilai derajat keanggotaan awal proses klasting seperti pada tabel 3.7.

Table 3.8 derajat keanggotaan awal

No	Kluster1	Kluster2	Kluster3	Kluster4	Kluster5
1	0.30372	0.08522	0.01882	0.35125	0.24098
2	0.05149	0.221	0.21498	0.2222	0.29033
3	0.17338	0.248	0.13999	0.1987	0.23993
4	0.33971	0.2983	0.05443	0.29765	0.00991
5	0.41086	0.04243	0.29828	0.08642	0.16201
6	0.14951	0.06645	0.45676	0.09781	0.22948
7	0.1825	0.16054	0.13774	0.2077	0.31153
8	0.31884	0.3684	0.05957	0.03937	0.21382
9	0.04307	0.06258	0.20062	0.30359	0.39014
10	0.14102	0.21388	0.04264	0.21346	0.389
11	0.35168	0.22505	0.00453	0.0478	0.37094

12	0.07942	0.0337	0.2081	0.34808	0.33071
13	0.12396	0.27542	0.24741	0.20076	0.15244
14	0.2424	0.3511	0.11804	0.19362	0.09484
15	0.08846	0.28744	0.22167	0.21501	0.18742
16	0.17184	0.2356	0.14362	0.1193	0.32965
17	0.11657	0.21209	0.30566	0.11732	0.24835
18	0.38844	0.05384	0.40845	0.10465	0.04463
19	0.31867	0.01092	0.30276	0.22932	0.13833
20	0.10715	0.16582	0.19934	0.15421	0.37348
21	0.30136	0.19362	0.11062	0.05644	0.33796
22	0.25332	0.19114	0.2937	0.02869	0.23315
23	0.2559	0.23428	0.10908	0.15204	0.2487
24	0.20301	0.1746	0.04819	0.24059	0.33361
25	0.30447	0.29905	0.02385	0.13565	0.23698
26	0.09134	0.3655	0.18963	0.28545	0.06808
27	0.12948	0.09018	0.38965	0.16769	0.223
28	0.18241	0.21357	0.00419	0.29263	0.3072
29	0.2877	0.10075	0.2251	0.15857	0.22789
30	0.15262	0.08964	0.30264	0.05183	0.40327

Langkah pertama adalah menentukan titik centroid dengan menggunakan persamaan 2.1 sehingga didapatkan hasil seperti pada tabel 3.8.

Table 3.9 nilai titik centroid setiap klsater

	Duration	*protocol_type*	*Service*	*Flag*	...
Kluster1	0.03555	0.03585	0.03231	0.03651	...
Kluster2	0.02896	0.02593	0.03214	0.02763	...
Kluster3	0.03392	0.03475	0.03416	0.03615	...
Kluster4	0.02963	0.03144	0.03865	0.01877	...
Kluster5	0.03700	0.03718	0.03000	0.04050	...

Langkah kedua adalah menentukan nilai fitness setiap partikel pada iterasi pertama dengan menggunakan persamaan *2.2*

Nilai fitness setiap partikel dihitunga dengan menggunakan rumus yang sama dengan seperti pada tabel 3.9 .

Table 3.10 nilai fitness awal setiap partikel

no	partikel	Nilai objektif
1	Partikel1	0.496881362
2	Partikel2	0.498387982
3	Partikel3	0.507618188
4	Partikel4	0.483849193
5	Partikel5	0.484178344
6	Partikel6	0.479385601
7	Partikel7	0.505897145
8	Partikel8	0.498340195
9	Partikel9	0.511648837
10	Partikel10	0.498340195

Pada iterasi pertama didapat nilai pbest dengan membandingkan setiap generasi dari satu partikel.

Nilai gbest untuk generasi pertama = 0.507618188

Langkah selanjutnya adalah inisialisasi matrik velocity dengan nilai random antara -1 dan 1. Selanjutnya nilai derajat keanggotaan akan diupdate menggunakan persamaan 2.3 dan dengan menggunakan nilai matrik velocity.

Setelah interasi ke 20 didapatkan nilai derajat keanggotaan seperti pada tabel 3.10.

Table 3.11 nilai derajat keanggotaan pada iterasi ke-20

no	Kluster1	Kluster2	Kluster3	Kluster4	Kluster5
1	0.25966	0.20150	0.18825	0.18030	0.17028
2	0.97283	0.01149	0.00665	0.00513	0.00389
3	0.99933	0.00029	0.00016	0.00012	0.00009
4	0.97519	0.01091	0.00602	0.00456	0.00332
5	0.99692	0.00134	0.00075	0.00057	0.00041
6	0.92751	0.03213	0.01760	0.01324	0.00952
7	0.00333	0.00391	0.00309	0.00259	0.98708
8	0.00018	0.00022	0.00017	0.00014	0.99929
9	0.00018	0.00021	0.00017	0.00014	0.99931
10	0.00018	0.00021	0.00016	0.00014	0.99932
11	0.00018	0.00021	0.00017	0.00014	0.99931
12	0.00018	0.00022	0.00017	0.00014	0.99929
13	0.00078	0.00127	0.99696	0.00058	0.00041
14	0.00028	0.00046	0.99890	0.00021	0.00015
15	0.00028	0.00045	0.99891	0.00021	0.00015
16	0.00027	0.00045	0.99893	0.00021	0.00014
17	0.00025	0.00041	0.99903	0.00019	0.00013
18	0.00028	0.00046	0.99890	0.00021	0.00015
19	0.00151	0.00203	0.00148	0.99411	0.00087
20	0.00071	0.00097	0.00069	0.99723	0.00040
21	0.00102	0.00137	0.00099	0.99603	0.00059
22	0.00070	0.00096	0.00069	0.99725	0.00040
23	0.16119	0.26156	0.14853	0.35216	0.07657
24	0.00071	0.00096	0.00069	0.99724	0.00040
25	0.00081	0.99764	0.00075	0.00048	0.00031
26	0.00078	0.99773	0.00073	0.00046	0.00030
27	0.00067	0.99806	0.00062	0.00039	0.00026
28	0.00066	0.99808	0.00062	0.00039	0.00025

29	0.00062	0.99819	0.00059	0.00037	0.00024
30	0.00064	0.99814	0.00060	0.00038	0.00024

Setelah proses training maka akan didapatkan nilai derajat keanggotaan terbaik dan titik *centroid* yang akan di-*import* untuk digunakan dalam mendeteksi jenis serangan selanjutnya dan dihitung nilai titik klasternya. Hasil perhitungan adalah seperti pada tabel 3.11.

Untuk menghitung nilai derajat keanggotaan digunakan ecludien distance. Input untuk proses deteksi adalah seperti pada tabel 3.12.

Table 3.12 record aktivitas dalam jaringan komputer

No	Variabel	Nilai
1	*Duration*	1
2	*protocol_type*	1
3	*Service*	181
4	*Flag*	1
5	*src_bytes*	1
6	*dst_bytes*	0
7	*Hot*	157
8	*logged_in*	1
9	*root_7ell*	1
10	*num_7ells*	181
11	*num_access_files*	1
12	*count*	1
13	*srv_count*	0
14	*serror_rate*	57
15	*srv_serror_rate*	1
16	*rerror_rate*	1
17	*srv_rerror_rate*	8
18	*same_srv_rate*	1
19	*diff_srv_rate*	1
20	*srv_diff_host_rate*	0
21	*dst_host_count*	15

Dengan menggunakan *ecluidience distance* didapatkan derajat keanggotaan dari setiap klaster, input dari proses deteksi adalah record aktivitas jaringan komputer dan titik centroid yang telah disimpan dalam sebuah array. Output dari proses deteksi ini adalah derajat keanggotaan untuk setiap kelas (tipe serangan jaringan) seperti pada tabel 3.13.

Table 3.13 nilai kemungkinan jenis serangan

No	normal	neptune	smurf	isweep	pod
1	0.113	0.094	0.264	0.235	0.294

Untuk record aktivitas jaringan diatas diambil nilai derajat keanggotaan yang paling besar yaitu 0.294, sehingga masuk ke dalam jenis serangan *pod.*

.

3.9 Perancangan Pengujian dan Analisis

Untuk menganalisa baik atau tidak hasil sebuah clustering dapat dilakukan dengan beberapa cara antara lain.

3.9.1 Nilai Objektif

Uji coba dilakukan dengan membandingkan nilai objektif clustering data menggunakan *Fuzzy c-means* dan nilai objektif clustering data menggunakan *FCM-PSO*. Rata-rata nilai objektif pada dataset yang dicluster dengan menggunakan Fuzzy c-means dan *FCM-PSO* akan ditampilkan pada sebuah grafik.

3.9.2 Classification Rate

Uji coba dilakukan dengan membandingkan hasil deteksi menggunakan *Fuzzy c-means* dengan hasil klasifikasi yang telah dilakukan oleh admin jaringan komputer menggunakan perhitungan akurasi. Hasil uji coba disimpan seperti pada table 3.14.

Table 3.14 hasil uji coba

No.	Record Aktivitas	Deteksi menggunakan FCM	Deteksi menggunakan FCM & PSO	Hasil deteksi dari KDDCUP 1999

Untuk keakuratan datanya yakni mengenai keakuratan sistem akan dihitung dengan rumus seperti persamaan berikut.

Akurasi (%) = $\frac{jumlah\ deteksi\ yang\ akurat}{jumlah\ semua\ deteksi}$ x 100%


UNIVERSITAS BRAWIJAYA

BAB IV
HASIL DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Implementasi perangkat lunak ini berupa aplikasi pemrograman yang menerapkan metoda *Particle Swarm Optimalization* pada *Fuzzy C-Mean* untuk mendeteksi kemungkinan serangan pada jaringan computer. Adapun variable yang dipakai dalam penelitian ini adalah aktivitas dalam jaringan computer yang sudah disimpan dalam bentuk file *.txt*.Jumlah semua variabel yang dipakai adalah 38 variabel. 3 variable diataranya memiliki nilai pasti dan yang lainnya memiliki nilai *linguistic*. Adapun lingkungan implementasi akan dijelaskan ke dalam subbab lingkungan implementasi perangkat keras dan lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan optimasi PSO pada Fuzzy C-Mean adalah :

1. Prosesor Intel Pentium (R) Dual CPU T3400 @2.16Ghz.
2. Memori 2 Gb
3. Hardisk 80 Gb
4. Monitor 15’
5. Keyboard
6. Mouse

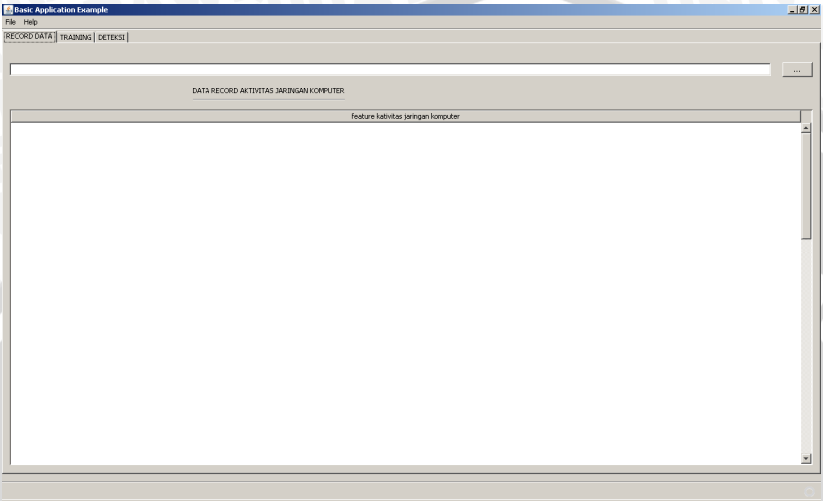
4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan optimasi PSO pada Fuzzy C-Mean adalah :

1. Sistem Operasi Window XP SP2
2. Netbean 6.9
3. Notepad

4.2 Implementasi Program

Tampilan *form* utama dari aplikasi deteksi serangan jaringan computer dapat dilihat pada gambar 4.1



Gambar 4.1 Tampilan Utama Aplikasi

Struktur Data

Struktur data untuk inisialisasi awal pada setiap partikel direpresentasikan sebagai berikut :

```
1.  partikel []swarm;
2.  partikel p;
3.  int maxiter,q;
4.  double w;
5.  public double [][]gbest;
6.  public double f_gbest;
7.  public double c1,c2,r1,r2;
```

Source code 4.1 Struktur Data dalam partikel

Sedangkan untuk menyimpan data aktivitas jaringan computer yang akan di-load melalui file txt akan diimplementasikan dalam struktur data sebagai berikut.

```
1.  //objek untuk data training
2.  FileChooserDemo d;
3.  int n,m,jml_kluster;
4.  double [][]x;
5.  int size;
6.  int indx;
8.  //objek untuk data testing
9.  FileChooserDemo tes;
10. double [][]data_test;
11. int ntest,mtest;
12. public double []tmp_test;
```

Source code 4.2 Struktur Data *record*

Sedangkan untuk menyimpan nilai standarisasi data untuk proses training dan proses testing akan diimplementasikan dalam struktur data sebagai berikut

```
1.  //data standarisation
2.  double []rata2x;
3.  double []mad;
4.  double [][]sf;
5.  //variable untuk standarisasi testing
6.  double []rata2x_test;
7.  double []mad_test;
8.  double [][]sf_test;
```

Source code 4.3 Struktur Data Standarisasi record

4.2.1 Fungsi Standarisasi Data

Fungsi standarisasi merupakan proses meminimalkan rentang nilai antar variabel aktivitas jaringan computer. Adapun fungsi standarisasi data dapat dilihat pada *source code 4.3.*

```
1.   public void rata2feature(){
2.   rata2x=new double[m];
3.   double []tmp=new double[m];
4.   for(int j=0;j<m;j++)
5.     tmp[j]=0;
6.    for(int i=0;i<n;i++)
7.      for(int j=0;j<m;j++){
8.        tmp[j]=tmp[j]+ x[i][j];
9.      }
10.     for(int j=0;j<m;j++)
11.       rata2x[j]=tmp[j]/n;
12.     }
13.  public void mad(){
14.  mad=new double[m];
15.  for(int j=0;j<m;j++)
16.   mad[j]=0;
17.   for(int i=0;i<n;i++)
18.   for(int j=0;j<m;j++)
19.   mad[j]=mad[j]+ Math.abs(x[i][j]-rata2x[j]);
20.   }
21.  public void sf_x(){
22.  sf=new double[n][m];
23.  for(int i=0;i<n;i++)
24.    for(int j=0;j<m;j++)
25. sf[i][j]=Math.abs((x[i][j]-rata2x[j])/mad[]);
26.       }
```

Source code 4.4 Standarisasi Data

4.2.2 Fungsi Inisialisasi Partikel

Fungsi inisialisasi partikel merupakan proses inisialisasi awal dimensi array untuk menyimpan data aktivitas serangan jaringan, nilai derajat keanggotaan dan nilai centroid array. Adapun fungsi inisialisasi partikel dapat dilihat pada *source code* 4.5.

```
1.  public void inisialisasi_partikel(int jml_ctr, int
    brs,int klm){
2.    this.jumlah_cluster=jml_ctr;
3.    this.n=brs;
4.    this.m=klm;
5.    this.x=new double[this.n][this.m];
6.    this._U=new double [n][jumlah_cluster];
7.    this._Vkj= new double  [jumlah_cluster][m];
8.    this.pbest=new double [n][jumlah_cluster];
9.    this.pbestV=new double [jumlah_cluster][m];
10.   velocity=new double [n][jumlah_cluster];
11. }
```

Source code 4.5 Inisialisasi partikel

4.2.3 Fungsi isi array record data

Fungsi isi record data merupakan proses mengisi data yang telah di import dari file txt ke dalam array masing-masing partikel. Adapun fungsi isi array record data diimplementasikan seperti *source code* 4.6

```
1.   public  void isi_x(double [][]p,int a,int b){
2.   for(int i=0;i<a;i++)
3.   for(int j=0;j<b;j++){
4.   this.x[i][j]=p[i][j];}
5.  }
```

Source code 4.6 isi *record* data

4.2.4 Fungsi inisialisasi derajat keanggotaan

Fungsi inisialisasi derajat keanggotaan merupakan fungsi untuk mengisi nilai derajat keanggotaan sebuah record data pada setiap kluster atau kelas dengan nilai acak antara 0 dan 1 dengan menggunakan aturan *Fuzzy C-Mean*. Sehingga jumlah semua derajat keanggotaan pada satu record data adalah 1. Adapun fungsi inisialisasi derajat keanggotaan diimplementasikan seperti *source code* 4.7.

```
1.  //fill miu with random value [0,1]
2.  public  void inisialisasi_U_acak()
3.  {
4.    BigDecimal bd;
5.    Random r=new Random();
6.    r=new rndom();
7.    for(int i=0;i<n;i++)
8.    {
9.      double sum=0.0;
10.     for(int j=0;j<jumlah_cluster;j++)
11.      {
12.       sum=sum+_U[i][j];
13.      }
14.     for(int j=0;j<jumlah_cluster;j++)
15.      {
16.       _U[i][j]=_U[i][j]/sum;
17.       bd=new BigDecimal(_U[i][j]);
18.       _U[i][j]=bd.doubleValue();
19.      }
20.   }
21. }
```

Source code 4.7 inisialisasi derajat keanggotaan

4.2.5 Fungsi inisialisasi velocity

Fungsi inisialisasi velocity merupakan proses inisialisasi awal matrix velocity dengan nilai acak antara [0,1]. Adapun fungsi inisialisasi velocity diimplementasikan seperti *source code* 4.8.

```
1.  //fill velocity with random value [-1,1]
2.  public void inisialisasi_velocity()
3.  {
4.    BigDecimal b;
5.    Random ran=new Random();
6.    for(int i=0;i<n;i++)
7.     for(int j=0;j<jumlah_cluster;j++)
8.     {
8.       double tmp=ran.nextDouble();
9.         if( tmp < 0.5){
10.           tmp=tmp-3*(tmp);
11.         }
12.        b=new BigDecimal(tmp);
13.    b=b.setScale(4,BigDecimal.ROUND_HALF_UP);
14.    velocity[i][j]=b.doubleValue();
15.       }
16.    }
```

Source code 4.8 inisialisasi *velocity*

4.2.6 Fungsi inisialisasi *pbest*

Fungsi inisialisasi pbest merupakan proses menentukan pbest awal setiap partikel. Pada iterasi pertama pbest merupakan partikel itu sendiri. Adapun fungsi inisialisasi pbest diimplementasikan seperti pada *source code* 4.9

```
1.  //create pbest for every particle
2.   public void inisialisasi_pbest(){
```

```
3.  //copy miu
4.  for(int i=0;i<this.n;i++)
5.    for(int j=0;j<this.jumlah_cluster;j++)
6.       this.pbest[i][j]=this._U[i][j];
7.       //fbest
8.       this.fbest=this.get_fitness();
9.  }
```

Source code 4.9 inisialisasi *pbest*

4.2.7 Inisialisasi *gbest*

Fungsi inisialisasi gbest merupakan proses pencarian fungsi objective optimal dari semua partikel dalam ruang permasalahan. adapun fungsi inisialisasi gbest diimplementasikan seperti pada *source code* 4.10

```
1.  //get gBest for every particle
2.  public void get_G_Best(partikel []particle){
3.   double max1=particle[0].fbest;
4.   double max2=0;
5.   int index=0;
6.   for(int p=1;p<this.swarm_size;p++){
7.     max2=particle[p].fbest;
8.     if(max2<=max1){
9.       max1=max2;
10.      this.indx=index;
11.      }
12.  }
13.  //set array best
14.  for(int i=0;i<this.n;i++)
15.  for(int j=0;j<this.jml_kluster;j++)
17.  gbest[i][j]=particle[this.indx].pbest[i][j];
18.  //mengeset nilai fitness_gbest
19.  this.f_gbest=max1;
```

```
20.   }
```

Source code 4.10 inisialisasi *gbest*

4.2.8 Fungsi get *fitness*

Fungsi *get_fitness()* merupakana proses untuk mendapatkan nilai fitness setiap partikel dengan menggunakan matrix derajat keanggotaan dan matrix centroid. Adapun fungsi get_fitness diimplementasikan seperti pada *source code* 4.11

```
1.  //calculate particle fitness
2.  public double get_fitness()
3.  {
4.    double p=0;
5.    for (int i=0;i<this.n;i++)
6.     for(int k=0;k<this.jumlah_cluster;k++){
7.       double w=0;
8.       double r=0;
9.       for(int j=0;j<this.m;j++)
10.        w=w+ (this.x[i][j]-this._Vkj[k][j])*(this.x[i][j]-
    this._Vkj[k][j]);
11.   r=w*((this._U[i][k])*(this._U[i][k]));
12.   p=p+r;
13.   }
14.   return p;
      }
```

Source code 4.11 *get fitness*

4.2.9 Fungsi hitung *centroid*

Fungsi hitung *centroid* merupakan proses untuk menghitung matrix centroid yang akan disimpan dalam matrix _Vkj[m][k] dengan m adalah jumlah feature dan k adalah jumlah kluster. Adapun fungsi hitung centroid diimplementasikan seperti *source code* 4.12.

```
1.  //menghitung nilai centroid array pada generasi ke
    generasi
2.   public  void hitung_Vkj(double pangkat)
3.   {
4.    for(int j=0;j<this.m;j++ )
5.     for(int k=0;k<this.jumlah_cluster;k++ )
6.     {
7.       double _atas=0;
8.       double _bawah=0;
9.       for (int i=0;i<this.n;i++){
10.    _atas=_atas +Math.pow(this._U[i][k],
       pangkat)*this.x[i][j];
11.    _bawah=_bawah +Math.pow(this._U[i][k], pangkat);
12.    }
13.   }
14.  }
```

Source code 4.12 hitung *centroid* array

4.2.10 Fungsi Inisialisasi matrix velocity

Fungsi inisialisasi matrix *velocity* adalah proses pemberian nilai awal pada matrix velocity. Nilai yang diberikan pada matrix velocity adalah nilai random dengan range antara -1 dan 1. Setelah beberapa iterasi nilai matrix velocity akan berubah untuk mengarahkan partikel pada global optimum. Adapun proses inisialisasi awal matrix velocity diimplementasikan seperti pada *source code* 4.13.

```
1.  //fill velocity with random value [-1,1]
2.          public void inisialisasi_velocity(){
3.              BigDecimal b;
4.              Random ran=new Random();
5.              for(int i=0;i<this.n;i++)
6.                  for(int j=0;j<this.jumlah_cluster;j++){
```

```
7.                          double tmp=ran.nextDouble();
8.                          if( tmp < 0.5){
9.                              tmp=tmp-2*(tmp);
10.                     }
11.                     b=new BigDecimal(tmp);
12.                     b=b.setScale(3,BigDecimal.ROUND_HALF_UP);
13.                     this.velocity[i][j]=b.doubleValue();
14.                     }
15.             }
```

Source code 4.13 *update* nilai *velocity*

4.2.11 Fungsi inisialisasi personal *best*

Fungsi inisialisasi personal best merupakan proses untuk mendapatkan nilai fitness terbaik masing-masing partikel pada awal proses iterasi. Adapun fungsi inisialisasi nilai personal best diimplementasikan seperti pada *source code* 4.14.

```
1.  //create pbest for every particle
2.          public void inisialisasi_pbest(){
3.              //copy miu
4.              for(int i=0;i<this.n;i++)
5.                  for(int j=0;j<this.jumlah_cluster;j++)
6.                  this.pbest[i][j]=this._U[i][j];
7.              //fbest
8.              this.fbest=this.get_fitness();
9.          }
```

Source code 4.14 *Update* nilai derajat keanggotaan

4.2.12 *Update* nilai velocity

Update nilai velocity merupakan proses memperbaharui nilai velocity. Perubahan nilai velocity didasarkan

pada parameter nilai *w, c1,c2, r1,*dan *r2.* Adapun fungsi update nilai velocity diimplementasikan seperti pada *source code* 4.15

```
1.  //update velocity values
2.    public void update_velocity(double [][]gbest,  double
    w, double c1, double c2,double r1,double r2){
3.  for(int i=0;i<this.n;i++)
4.    for(int j=0;j<this.jumlah_cluster;j++)
5.    this.velocity[i][j]=w*this.velocity[i][j] + c1*r1*(
    this.pbest[i][j] -_U[i][j])+c2*r2*(gbest[i][j]-
    _U[i][j]);
6.    }
```

Source code 4.15 *update* nilai *velocity*

4.2.13 Fungsi *update* derajat keanggotaan

Fungsi update derajat keanggotaan merupakan proses memperbaharui nilai derajat keanggotaan setiap record pada cluster tertentu. Adapun fungsi update derajat keanggotaan diimplementasikan seperti pada *source code* 4.16.

```
1.  //update miu values
2.  public  void update_U()
3.  {
4.  for(int i=0;i<this.n;i++)
5.  {
6.  int inc=-1;
7.  for(int k=0;k<this.jumlah_cluster;k++)
8.   {
9.    inc++ ;
10.   double tmp=0;
11.   double _bawah=0;
12. //menentukan pembagi
13. for( k=0;k<this.jumlah_cluster;k++)
14.  {
```

```
15.     tmp=0;
16.     for(int j=0;j<this.m;j++)
17.    tmp=tmp+ (this.x[i][j] - this._Vkj[k][j])*(this.x[i][j]
       - this._Vkj[k][j]);
18.     _bawah=_bawah+(1/tmp);
19.      }
20.     double _atas=0;
21.     tmp=0;
22.     for(int j=0;j<this.m;j++)
23.      tmp=tmp+(this.x[i][j]-
        this._Vkj[inc][j])*(this.x[i][j]-this._Vkj[inc][j]);
24.     //-------------------
25.     _atas=_atas+ 1/(tmp);
26.     //-------------------
27.     k=inc;
28.     this._U[i][k]=_atas/_bawah;
29.        }
30.      }
31.     }
```

Source code 4.16 *Update* nilai derajat keanggotaan

4.2.14 Update *miu* menggunakan matrix *velocity*

Fungsi update nilai derajat keanggotaan menggunakan matrix velocity adalah menambahkan nilai matrix derajat keanggotaan dengan nilai velocity yang sudah di-update. Adapun fungsi update derajat keanggotaan dengan menggunakan nilia velocity diimplementasikan seperti pada *source code* 4.17

```
1.  //update velocity values
2.  public void update_U_with_velocity()
3.   {
4.   for(int i=0;i<this.n;i++)
5.    for(int j=0;j<this.jumlah_cluster;j++)
6.    this._U[i][j]=_U[i][j]+this.velocity[i][j];
```

```
7.    //normalisasi jika ada nilai di bawah 0
8.    for(int i=0;i<this.n;i++)
9.    for(int j=0;j<this.jumlah_cluster;j++)
10.    {
11.      if(this._U[i][j]<0)
12.         this._U[i][j]=0;
13.    }
14.  //mengganti nilai 0 dengan random [0,1]
15.    Random r=new Random();
16.    for(int i=0;i<this.n;i++)
17.    for(int j=0;j<this.jumlah_cluster;j++)
18.     if(this._U[i][j]==0)
19.     {
20.      this._U[i][j]=r.nextDouble();
21.     }
22.  //normalisasi nilai derajat keanggotaan
23.    for(int i=0;i<this.n;i++)
24.     {
25.      double tmp=0;
26.      for(int j=0;j<this.jumlah_cluster;j++)
27.       {
28.        tmp=tmp+this._U[i][j];
29.       }
30.      for(int j=0;j<this.jumlah_cluster;j++)
31.      this._U[i][j]=this._U[i][j]/tmp;
32.               }
34.          }
```

Source code 4.17 update miu dengan menggunakan nilai velocity

4.2.15 Fungsi deteksi

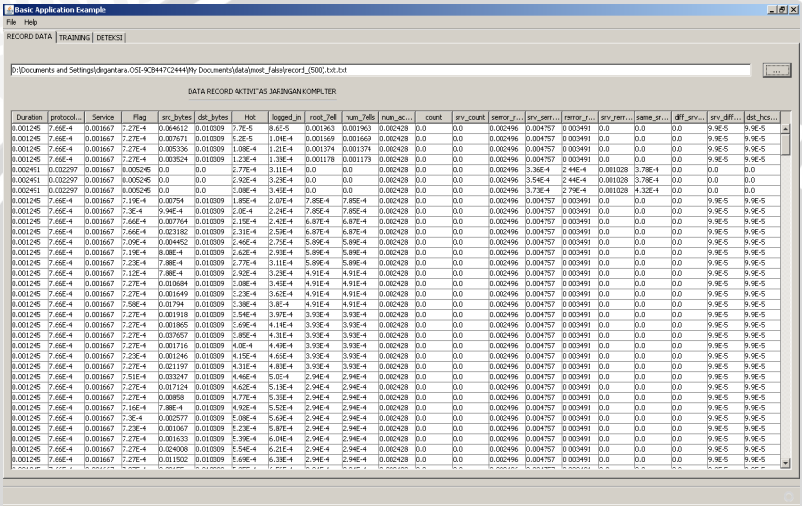
Fungsi deteksi merupakan tahap uji coba hasil clustering. Input proses deteksi berupa 1 (satu) record data yang memiliki jumlah feature sama dengan data training. Pencarian jarak terdekat dalam sebuah cluster menggunakan *ecluidien distance.* Adapun fungsi deteksi diimplementasikan seperti pada *source code* 4.18

```
1.  private void jButton8ActionPerformed(
    java.awt.event.ActionEvent evt) {
2.  // TODO add your handling code here:
3.  //memindah informasi info label klustering
4.   String []h=new String[this.jml_kluster];
5.   for(int y=0;y<this.jml_kluster;y++)
6.     h[y]=this.info[y];
7.     tmp_test=new double[this.jml_kluster];
8.     double min=0;
9.     int tnd=0;
10.    for(int i=0;i<this.jml_kluster;i++){
11.      tmp_test[i]=0;
12.      for(int j=0;j<this.m;j++)
13.          tmp_test[i]=tmp_test[i]+Math.pow( (p._Vkj[i][j]-
       this.data_test[0][j]), 2);
14.      if(i==0){
15.        min=tmp_test[i];
16.        tnd=i;
17.        }else if(tmp_test[i]<min ){
18.        min=tmp_test[i];
19.        tnd=i;
20.        } }
21.   JOptionPane.showMessageDialog(null, h[tnd]);
22.   }
```

Source code 4.18 proses deteksi serangan jaringan computer

4.3 Penerapan Aplikasi

Aplikasi klasifikasi jenis serangan menggunakan FPSO diterapkan dengan memasukkan dataset aktivitas jaringan computer yang diambil dari KKDCUP 1999, universitas Columbia. Adapun contoh data yang akan dimasukkan adalah seperti pada gambar 4.2.

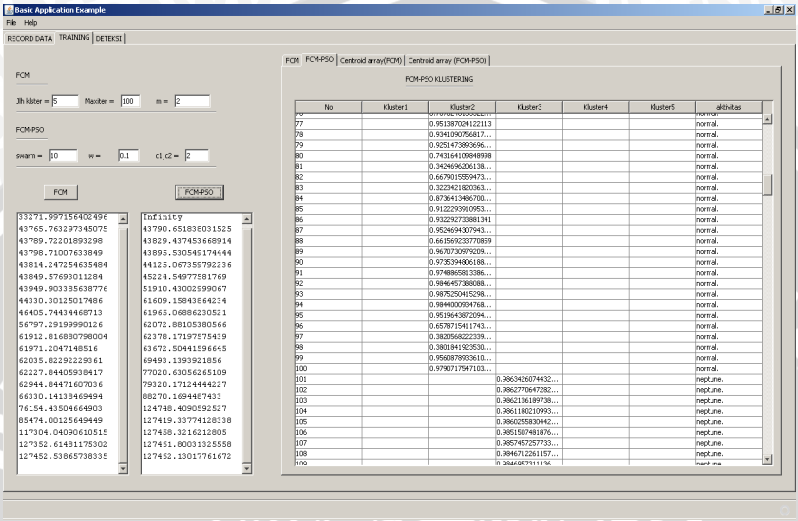


Gambar 4.2 data aktifitas serangan computer terstandarisasi

Pada gambar 4.2 adalah variabel aktivitas jaringan computer yang telah distandarisasi. Sebelum proses standarisasi terdapat rentang yang cukup besar antara variabel satu dengan yang lainnya. Proses standarisasi meliputi 3 tahap yaitu perhitungan rata-rata *feature*, perhitungan *mean absolute derivation(MAD)* dan perhitungan *standart feature(SF)*. Output dari proses standarisasi akan disimpan dalam matrik standart *feature* dan akan dipakai pada proses selanjutnya.

Langkah selanjutnya adalah melaukan training data aktivitas serangan jaringan computer dengan menggunakan klastering FCM dan *FCM-PSO*. Pada tahap ini akan ditampilkan nilai *fitness* masing-masing algoritma klastering pada setiap iterasi generasi. Output dari proses ini adalah nilai derajat keanggotaan terbesar pada masing-

masing cluster dan matrik centroid array. Centroid array dapat digunakan untuk proses selanjutnya yaitu proses deteksi.



Gambar 4.3 *form* proses *training*

Dari proses clustering akan didapatkan nilai *centroid* array yang akan digunakan pada proses selanjutnya yaitu deteksi serangan jaringan computer.

	Kluster1	Kluster2	Kluster3	Kluster4	Kluster5
Duration	0.002477771600083299	0.002483281220204727	0.001245274626852329	0.002480761759834208 3	0.001247062104717175
protocol_type	0.002281144126704828	0.003048313190702822 4	0.001528334593742922	0.002284597862298343	7.687817055798819E-4
Service	0.001667039906058 12	0.001667021926277451 7	0.003326395550177841 4	0.001667032744614153 3	0.001667044931342693 6
Flag	0.005193205203336579	3.5783248865246 19E-5	4.105014784671103E-6	0.003634756550526142	9.524164156118628E-4
src_bytes	2.477752741411103E-4	1.951943664187259E-4	1.417434693431749E-4	2.455293258986 35E-4	0.005531234947642565 4
dst_bytes	1.092005580613007 5E-4	6.372781658508 47E-5	3.859533740252402E-5	8.450525992549918E-5	0.010291660784643036
Hot	1.828173222641840 3E-4	2.125847776587549 2E-5	0.001657724916107421 2	0.007787077109579304	3.287304359943337 4E-4
logged_in	2.051012673830278 1E-4	4.591779334730277 3E-4	2.135428730475155E-4	0.008730756500400802	3.725638921306711 7E-4
root_7ell	2.493476743741072E-6	1.578104339081106E-6	0.009777152807974 31	2.114429296208722 5E-6	1.792866493731598 6E-4
num_7ells	2.493476743741072E-6	1.578104339081106E-6	0.009777152807974 31	2.114429296208722 5E-6	1.792866493731598 6E-4
num_access_files	0.002427994994533086	0.002427972477923438	2.943789225092583 6E-4	0.002427958096530396	0.002427943619035641 4
count	2.768636876133905 4E-7	1.533744148148369 6E-7	0.009946844233560187	2.283640215589414 4E-7	3.134717286522955E-7
srv_count	7.342585689401510 4E-6	0.009510668627325069	2.870455798425361 6E-6	6.318658841818388E-6	4.627838832260000 7E-4
serror_rate	0.002494766883593655	3.192524552136581E-5	0.002495903267380526 4	0.002495471397349279	0.002495262328721 4887
srv_serror_rate	2.574574010658655E-4	0.001160749818566904	2.153327145330620 2E-4	0.003831010362 9499 46	0.004754906982357194
rerror_rate	1.899020958240569 8E-4	0.00349074357438674	1.571082986765526 6E-4	0.002812305379578 8156	0.003490437417616955
srv_rerror_rate	0.001020721294259364 5	1.757616164082912 6E-6	0.003465146952189458	0.005613639648898684	3.889964345228 6E-6
same_srv_rate	2.364437287582080 5E-4	0.005362243334003407	1.069809842175477 7E-6	0.004303071060556278	8.199060206913803 6E-6
diff_srv_rate	5.189214536481871E-6	0.009927868069095262	4.070125570755836 7E-7	2.224261338944187 6E-6	3.103971099399899 6E-6
srv_diff_host_rate	1.285913926288072 6E-6	7.423415963735209 6E-7	0.009865108653614204	1.006185381208205 6E-6	9.910059387264625 5E-5
dst_host_count	1.285913926288072 6E-6	7.423415963735209 6E-7	0.009865108653614204	1.006185381208205 6E-6	9.910059387264625 5E-5

Gambar 4.4 *form centroid* array

Pada proses deteksi akan di-load 1(satu) aktifitas jaringan computer dan akan dihitung *ecluidien distance* (jarak terdekat) terhadap masing-masing kelas.

Table 4.1 record aktivitas jaringan computer

No	Variable	Nilai
1	*Duration*	0
2	*protocol type*	2
3	*Service*	4
4	*Flag*	0
5	*src bytes*	0
6	*dst bytes*	0
7	*Hot*	0
8	*logged in*	1
9	*root 7ell*	0
10	*num 7ells*	0.5
11	*num access files*	2
12	*Count*	26
13	*srv count*	1
14	*serror_rate*	0
15	*srv serror rate*	1
16	*rerror rate*	0.5
17	*rv rerror rate*	0
18	*same srv rate*	0
19	*diff srv rate*	0
20	*srv diff host rate*	0
21	*dst host count*	0

Langkah selanjutnya adalah menghitung nilai derajat keanggotaan sebuah record data pada setiap cluster yang sudah dibentuk pada proses training. Perhitungan jarak pada masing masing cluster menggunakan ecluidien distance.

Nilai derajat keanggotaan *record* data aktivitas jaringan komputer pada cluster portsweep adalah

$$\mu_{isweep} = \frac{3.0717279344552E}{4.2589637441258 53E} = 0.7212383384788987$$

Nilai derajat keanggotaan record data aktivitas jaringan komputer pada cluster pod adalah

$$\mu_{pod} = \frac{5.061475094081611E}{4.2589637441258 53E} = 0.001188428781781159$$

Nilai derajat keanggotaan record data aktivitas jaringan komputer pada cluster smurf adalah

$$\mu_{smurf} = \frac{1.6034035053228484E}{4.2589637441258 53E} = 0.03764773784548722$$

Nilai derajat keanggotaan record data aktivitas jaringan komputer pada cluster normal adalah

$$\mu_{normal} = \frac{8.905452284712614E}{4.2589637441258 53E} = 0.20909903957260492$$

Nilai derajat keanggotaan record data aktivitas jaringan komputer pada cluster neptune adalah

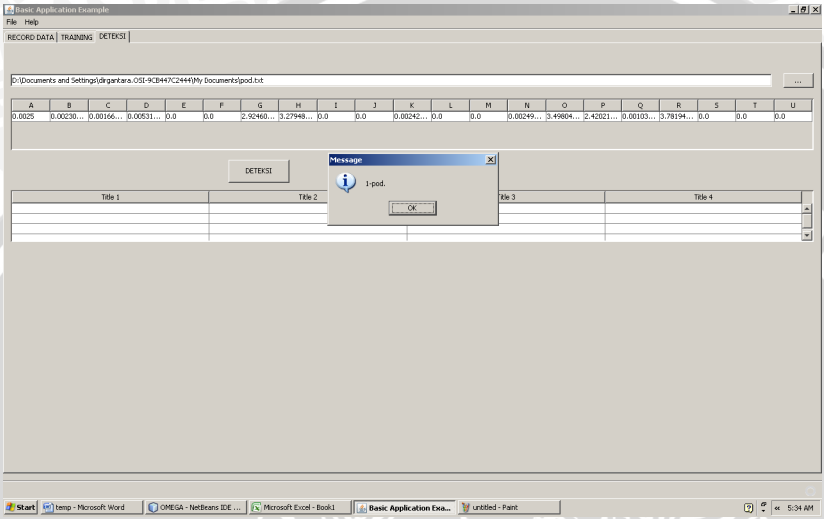
$$\mu_{neptune} = \frac{1.3128875557302597E}{4.2589637441258 53E} = 0.030826455321228102$$

Nilai derajat keanggotaan terlihat pada table 4.2

Tabel 4.2 derajat keanggotaan pada setiap klaster

No	Kluster	Nilai derajat keanggotaan
1	isweep	0.7212383384788987
2	**pod**	**0.001188428781781159**
3	smurf	0.03764773784548722
4	normal	0.20909903957260492
5	neptune	0.030826455321228102

Pada table 4.2 dapat dilihat nilai *ecluidien distance* (jarak terdekat) pada masing-masing cluster. Sehingga aktivitas jaringan computer pada dataset testing termasuk kedalam jenis serangan *pod.*



Gambar 4.5 *form* deteksi

4.4 Pengujian dan Analisis

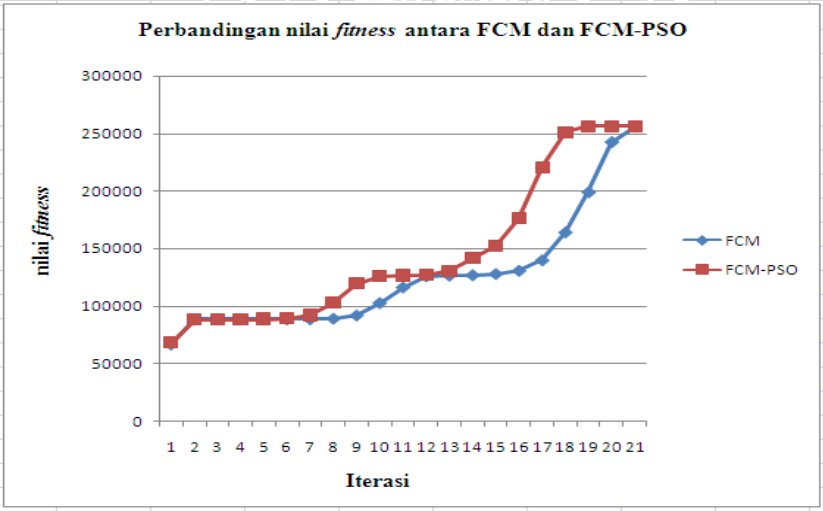
Dalam proses pengujiannya akan dilakukan dengan membandingkan nilai objektif dan *Classification Rate* antara hasil klastering menggunakan *Fuzzy c-means* dan *FCM-PSO*. Dalam proses pengujian ini juga dilakukan beberapa kombinasi komponen-komponen yang ada. Hal ini dimaksudkan untuk mendapatkan nilai komponen terbaik untuk proses klastering.

4.4.1 Nilai objektif

Pengujian ini dilakukan untuk mengetahui pengaruh optimasi *Particle Swarm Optimalization* terhadap nilai objektif dari proses klastering. Dalam proses pengujian akan digunakan komponen dalam PSO yaitu *inertia weight* (*w*) antara 0,1 dan 0,9. Tabel 4.4 merupakan hasil pengujian dari nilai *inertia weight*.

Tabel 4.3 Uji nilai objektif dengan menggunakan beberapa nilai *w*

dataset	Inertia weight	iterasi	Nilai objektif (Fuzzy c-mean)	Nilai objektif (FCM_PSO)
500	0,1	5	43853.1231	44255.438
		10	62492.624	62456.489
		15	85344.782	127341.573
		20	127461.264	127462.148
	0,5	5	33263.753	33784.220
		10	43944.023	44308.185
		15	62492.624	66478.352
		20	127461.264	127462.168
	0,9	5	33263.753	33844.032
		10	43944.023	43817.244
		15	62492.624	43853.357
		20	127461.264	43853.3571



Gambar 4.6 perbandingan nilai *fitness* pada 1000 *record* data

Pada nilai inertia weight 0,1 menunjukan bahwa nilai objektif hasil klastering menggunakan FCM-PSO lebih baik dibandingkan dengan menggunakan FCM. Pada iterasi ke 5,10,15, dan 20 sudah menunjukan hasil yang lebih baik walaupun waktu komputasi algoritma FCM-PSO lebih lama. Hal ini disebabkan karena FCM-PSO memiliki ruang permasalahan yang lebih besar.

Hasil akhir klasifikasi yang dilakukan pada dataset aktifitas serangan jaringan komputer dengan 500 *record* dengan jumlah *feature* 21 dan jumlah klaster 5 adalah seperti gambar 4.4. Pada gambar 4.3 dapat dilihat bahwa nilai *fitness* klastering dengan menggunakan optimasi *Particle Swarm Optimalization* menunjukan hasil yang lebih baik pada setiap generasi.

Dari hasil uji yang dilakukan, dapat diketahui bahwa komponen terbaik untuk melakukan optimasi menggunakan *Particle Swarm Optimalization* adalah pada P=10, w=0.1, $c1$=$c2$=2 dengan nilai *fitness* tertinggi yaitu 127462.148 sedangkan hasil optimasi paling rendah adalah dengan parameter w=0.9 dengan nilai fitness 43853.357.

Ada beberapa hal yang mempengaruhi hasil uji coba tersebut antara lain pada proses standarisasi banyak *feature* yang memiliki nilai 0 sehingga perbedaan antara klaster menjadi jelas. Sebaliknya jika keragaman nilai pada setiap *feature* lebih banyak maka hasil optimasi akan semakin terlihat.

Hal lain yang juga berpengaruh pada hasil uji coba adalah rentang nilai yang terlalu besar pada inisialisasi awal matrix *velocity* yaitu [-1,1] yang menyebabkan *Particle Swarm Optimalization* sulit mengarahkan pada hasil klasifikasi terbaik

4.4.2 Classification Rate

Uji coba dilakukan dengan membandingkan hasil deteksi menggunakan *Fuzzy c-means* dan FCM-PSO dengan hasil klasifikasi yang telah dilakukan oleh admin jaringan komputer menggunakan perhitungan akurasi. Hasil uji coba disimpan seperti pada table 3.14.

Tabel 4.4 perbandingan *classification rate*

No.	Record Aktivitas	Deteksi menggunakan FCM	Deteksi menggunakan FCM & PSO	Hasil deteksi dari KDDCUP 1999
1	500	97%	97%	100%
2	1000	98%	98%	100%
3	2000	98%	98%	100%

Dari tabel 4.5 dapat dilihat bahwa *classification rate* klastering dengan menggunakan optimasi *Particle Swarm Optimalization* sama dengan *classification rate* klastering dengan menggunakan *Fuzzy c-means*. Hal ini disebabkan karena sebagian variabel dalam dataset memiliki nilai 0 atau memiliki nilai yang sama sehingga perbedaan antar cluster sangat jelas.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian tentang optimasi *Particle Swarm Optimalization* pada *Fuzzy c-means* ini dapat diambil kesimpulan sebagai berikut:

1. Algoritma otimasi *Particle Swarm Optimalization* dapat digunakan pada *Fuzzy c-means* untuk clustering dataset aktivitas serangan jaringan komputer.

2. Dari uji coba yang telah dilakukan, dapat diketahui bahwa parameter terbaik yang digunakan pada proses optimasi adalah sebagai berikut : $c_1 = c_2 = 2.0$, $P = 10$ dan $w = 0.1$ dan telah diuji pada dataset aktivitas serangan jaringan komputer dengan jumlah *record* data tertentu yang dapat meningkatkan nilai *fitness* hasil *clustering*.

5.2 Saran

Pada penelitian ini clustering jenis serangan jaringan menggunakan dataset aktivitas jaringan computer yang didalamnya banyak mengandung variabel dengan nilai 0 sehingga perbedaan antara hasil clustering dengan menggunakan FCM dan FCM-PSO tidak begitu besar. Oleh karena itu selain menggunakan dataset pada penelitian ini akan lebih baik apabila menggunakan dataset dari beberapa sumber.

DAFTAR PUSTAKA

Bace, Rebbeca. 2000. Intrusion Detection. Macmillan Technical Publising. London

Carlisle, A. dan Dozier, G., 2001. *An Off-The-Shelf PSO, Proceedings of the 2001 Workshop on Particle Swarm Optimization.* pp. 1-6, Indianapolis, IN

Dorigo, Marco. 2004. *Ant Colony Optimization*. Sydney: Maccasuset Institute of Technology

Engelbrecht, A.P. 2007. *Computational Intelligence an Introduction*. New York: John Wiley & Sons

Klawon, F. 2001. What Is About fuzzy Clustering? Undersatnding And Improving The Concept Of The Fuzzier. Science Journal. http://public.rzfh-wolfebuettel.de/klawon

Kusuma, Dewi. 2002. Anilisis dan Desain Sistem fuzzy Menggunakan Toolbox Matlab. Graha Ilmu. Yogyakarta.

Parsopoulos, K. E. 2002. *Recent Approaches to Global Optimization Problems through Particle Swarm Optimization. NaturalComputing*, No. 1, pp.235–306.

Rehman, Rafeeq. 2003. The Linux Development Platform configuring using and maintenance. Prentice Hall PRT. London

Shihab, A. 2000. fuzzy Clsutering Algorithm And Their Application To Medical Image Analysis. Disertation,University of London. London.

Singh, 2003. *Network Security and Management*. Asoke K Ghosh Publisher. India.


UNIVERSITAS BRAWIJAYA

Lampiran Hasil kclustering menggunakan *FCM-PSO*

No	neptun	ipsweep	pod	smurf	normal	Aktivitas
1	0.1796	0.1885	0.1912	0.2437	0.1970	normal.
2	0.0128	0.0199	0.0227	0.9076	0.0370	normal.
3	0.0052	0.0085	0.0098	0.9595	0.0170	normal.
4	0.0103	0.0173	0.0202	0.9158	0.0363	normal.
5	0.0002	0.0005	0.0007	0.0006	0.9980	normal.
6	0.0003	0.0005	0.0007	0.0007	0.9978	normal.
7	0.0003	0.0006	0.0009	0.0009	0.9973	normal.
8	0.0184	0.0337	0.0393	0.8344	0.0742	normal.
9	0.0758	0.1108	0.1212	0.5326	0.1596	normal.
10	0.0208	0.0382	0.0448	0.8117	0.0845	normal.
11	0.0189	0.0346	0.0407	0.8293	0.0764	normal.
12	0.1494	0.1746	0.1809	0.2985	0.1967	normal.
13	0.0201	0.0368	0.0434	0.8183	0.0814	normal.
14	0.0240	0.0440	0.0520	0.7824	0.0977	normal.
15	0.0985	0.1352	0.1460	0.4409	0.1794	normal.
16	0.1401	0.1688	0.1764	0.3193	0.1954	normal.
17	0.0341	0.0550	0.0625	0.7559	0.0925	normal.
18	0.0115	0.0195	0.0226	0.9104	0.0361	normal.
19	0.0278	0.0510	0.0608	0.7470	0.1134	normal.
20	0.0132	0.0241	0.0287	0.8812	0.0527	normal.
21	0.0254	0.0466	0.0558	0.7686	0.1036	normal.
22	0.0006	0.0011	0.0013	0.9949	0.0022	normal.
23	0.1111	0.1467	0.1576	0.3982	0.1864	normal.
24	0.0069	0.0124	0.0149	0.9390	0.0268	normal.
25	0.0214	0.0391	0.0471	0.8060	0.0865	normal.
26	0.0071	0.0121	0.0143	0.9434	0.0231	normal.
27	0.0060	0.0103	0.0122	0.9516	0.0198	normal.
28	0.0174	0.0317	0.0384	0.8426	0.0699	normal.
29	0.1771	0.1889	0.1921	0.2443	0.1976	normal.

30	0.0034	0.0062	0.0075	0.9699	0.0131	normal.
31	0.0033	0.0059	0.0071	0.9714	0.0124	normal.
32	0.0012	0.0021	0.0025	0.9900	0.0043	normal.
33	0.0183	0.0335	0.0409	0.8334	0.0738	normal.
34	0.0806	0.1162	0.1295	0.5094	0.1644	normal.
35	0.0205	0.0376	0.0460	0.8128	0.0830	normal.
36	0.0273	0.0447	0.0524	0.7983	0.0773	normal.
37	0.0278	0.0509	0.0626	0.7456	0.1130	normal.
38	0.0278	0.0509	0.0628	0.7454	0.1131	normal.
39	0.0276	0.0506	0.0625	0.7469	0.1124	normal.
40	0.0026	0.0047	0.0058	0.9770	0.0099	normal.
41	0.1074	0.1433	0.1563	0.4086	0.1843	normal.
42	0.0066	0.0114	0.0138	0.9465	0.0217	normal.
43	0.0278	0.0509	0.0634	0.7448	0.1131	normal.
44	0.0261	0.0478	0.0596	0.7604	0.1061	normal.
45	0.0055	0.0096	0.0116	0.9550	0.0183	normal.
46	0.0079	0.0142	0.0177	0.9295	0.0306	normal.
47	0.0089	0.0161	0.0201	0.9203	0.0347	normal.
48	0.0287	0.0526	0.0660	0.7359	0.1167	normal.
49	0.1218	0.1848	0.1624	0.3437	0.1872	normal.
50	0.0425	0.0929	0.0776	0.6699	0.1171	normal.
51	0.1352	0.1765	0.1721	0.3235	0.1927	normal.
52	0.0148	0.0316	0.0298	0.8734	0.0505	normal.
53	0.0103	0.0214	0.0210	0.9111	0.0363	normal.
54	0.0079	0.0162	0.0163	0.9311	0.0284	normal.
55	0.0056	0.0111	0.0115	0.9520	0.0198	normal.
56	0.0481	0.0815	0.0845	0.6657	0.1201	normal.
57	0.0040	0.0078	0.0083	0.9654	0.0145	normal.
58	0.0032	0.0063	0.0067	0.9719	0.0119	normal.
59	0.0028	0.0054	0.0058	0.9761	0.0099	normal.
60	0.0017	0.0033	0.0036	0.9852	0.0062	normal.

61	0.0014	0.0027	0.0029	0.9880	0.0050	normal.
62	0.0017	0.0032	0.0035	0.9857	0.0059	normal.
63	0.0055	0.0100	0.0111	0.9552	0.0182	normal.
64	0.0492	0.0800	0.0868	0.6621	0.1219	normal.
65	0.1150	0.1530	0.1605	0.3837	0.1878	normal.
66	0.1156	0.1535	0.1611	0.3818	0.1880	normal.
67	0.0050	0.0090	0.0101	0.9593	0.0166	normal.
68	0.0022	0.0041	0.0046	0.9813	0.0077	normal.
69	0.9863	0.0030	0.0033	0.0033	0.0041	neptune.
70	0.9862	0.0030	0.0034	0.0033	0.0041	neptune.
71	0.9862	0.0030	0.0034	0.0033	0.0041	neptune.
72	0.9861	0.0030	0.0034	0.0034	0.0042	neptune.
73	0.9860	0.0030	0.0034	0.0034	0.0042	neptune.
74	0.9851	0.0032	0.0036	0.0036	0.0044	neptune.
75	0.9857	0.0031	0.0035	0.0034	0.0043	neptune.
76	0.9846	0.0033	0.0037	0.0037	0.0046	neptune.
77	0.9847	0.0033	0.0037	0.0037	0.0046	neptune.
78	0.9978	0.0005	0.0005	0.0005	0.0007	neptune.
79	0.9979	0.0005	0.0005	0.0005	0.0006	neptune.
80	0.9979	0.0004	0.0005	0.0005	0.0006	neptune.
81	0.9980	0.0004	0.0005	0.0005	0.0006	neptune.
82	0.9980	0.0004	0.0005	0.0005	0.0006	neptune.
83	0.9980	0.0004	0.0005	0.0005	0.0006	neptune.
84	0.9939	0.0013	0.0015	0.0015	0.0018	neptune.
85	0.9980	0.0004	0.0005	0.0005	0.0006	neptune.
86	0.9983	0.0004	0.0004	0.0004	0.0005	neptune.
87	0.9984	0.0003	0.0004	0.0004	0.0005	neptune.
88	0.9975	0.0005	0.0006	0.0006	0.0007	neptune.
89	0.9975	0.0005	0.0006	0.0006	0.0007	neptune.
90	0.9983	0.0004	0.0004	0.0004	0.0005	neptune.
91	0.9983	0.0004	0.0004	0.0004	0.0005	neptune.

92	0.9973	0.0006	0.0007	0.0006	0.0008	neptune.
93	0.9993	0.0002	0.0002	0.0002	0.0002	neptune.
94	0.9973	0.0006	0.0007	0.0007	0.0008	neptune.
95	0.9774	0.0048	0.0054	0.0054	0.0069	neptune.
96	0.9773	0.0048	0.0055	0.0054	0.0069	neptune.
97	0.9773	0.0048	0.0055	0.0055	0.0069	neptune.
98	0.9772	0.0049	0.0055	0.0055	0.0070	neptune.
99	0.9991	0.0002	0.0002	0.0002	0.0003	neptune.
100	0.9984	0.0004	0.0004	0.0004	0.0005	neptune.
101	0.9985	0.0003	0.0004	0.0004	0.0004	neptune.
102	0.9987	0.0003	0.0003	0.0003	0.0004	neptune.
103	0.9988	0.0003	0.0003	0.0003	0.0004	neptune.
104	0.9989	0.0002	0.0003	0.0003	0.0003	neptune.
105	0.9985	0.0003	0.0004	0.0004	0.0005	neptune.
106	0.9986	0.0003	0.0003	0.0003	0.0004	neptune.
107	0.9995	0.0001	0.0001	0.0001	0.0002	neptune.
108	0.9995	0.0001	0.0001	0.0001	0.0001	neptune.
109	0.9470	0.0116	0.0129	0.0128	0.0157	neptune.
110	0.9471	0.0116	0.0129	0.0128	0.0157	neptune.
111	0.9471	0.0116	0.0129	0.0128	0.0157	neptune.
112	0.9471	0.0115	0.0129	0.0128	0.0157	neptune.
113	0.9471	0.0115	0.0129	0.0128	0.0157	neptune.
114	0.9996	0.0001	0.0001	0.0001	0.0001	neptune.
115	0.9860	0.0030	0.0034	0.0034	0.0042	neptune.
116	0.9988	0.0003	0.0003	0.0003	0.0004	neptune.
117	0.9988	0.0003	0.0003	0.0003	0.0004	neptune.
118	0.9789	0.0045	0.0051	0.0051	0.0064	neptune.
119	0.9984	0.0003	0.0004	0.0004	0.0005	neptune.
120	0.9987	0.0003	0.0003	0.0003	0.0004	neptune.
121	0.9988	0.0002	0.0003	0.0003	0.0004	neptune.
122	0.9990	0.0002	0.0002	0.0002	0.0003	neptune.

123	0.9992	0.0002	0.0002	0.0002	0.0003	neptune.
124	0.9993	0.0002	0.0002	0.0002	0.0002	neptune.
125	0.9994	0.0001	0.0001	0.0001	0.0002	neptune.
126	0.9995	0.0001	0.0001	0.0001	0.0002	neptune.
127	0.9995	0.0001	0.0001	0.0001	0.0001	neptune.
128	0.9996	0.0001	0.0001	0.0001	0.0001	neptune.
129	0.9798	0.0043	0.0049	0.0049	0.0062	neptune.
130	0.9996	0.0001	0.0001	0.0001	0.0001	neptune.
131	0.9798	0.0043	0.0049	0.0048	0.0062	neptune.
132	0.9798	0.0043	0.0049	0.0048	0.0062	neptune.
133	0.9797	0.0043	0.0049	0.0049	0.0062	neptune.
134	0.9797	0.0043	0.0049	0.0049	0.0062	neptune.
135	0.9796	0.0043	0.0050	0.0049	0.0062	neptune.
136	0.9795	0.0044	0.0050	0.0049	0.0062	neptune.
137	0.9794	0.0044	0.0050	0.0049	0.0063	neptune.
138	0.9793	0.0044	0.0050	0.0050	0.0063	neptune.
139	0.9791	0.0044	0.0051	0.0050	0.0063	neptune.
140	0.9795	0.0044	0.0050	0.0049	0.0063	neptune.
141	0.9973	0.0006	0.0007	0.0007	0.0008	neptune.
142	0.9973	0.0006	0.0006	0.0006	0.0008	neptune.
143	0.9981	0.0004	0.0005	0.0005	0.0006	neptune.
144	0.9980	0.0004	0.0005	0.0005	0.0006	neptune.
145	0.9972	0.0006	0.0007	0.0007	0.0009	neptune.
146	0.9971	0.0006	0.0007	0.0007	0.0009	neptune.
147	0.9974	0.0006	0.0006	0.0006	0.0008	neptune.
148	0.9977	0.0005	0.0006	0.0006	0.0007	neptune.
149	0.9975	0.0005	0.0006	0.0006	0.0007	neptune.
150	0.9974	0.0005	0.0006	0.0006	0.0008	neptune.
151	0.9973	0.0006	0.0007	0.0007	0.0008	neptune.
152	0.0171	0.0269	0.8718	0.0310	0.0532	smurf.
153	0.0135	0.0215	0.8970	0.0249	0.0431	smurf.

154	0.0134	0.0214	0.8973	0.0248	0.0430	smurf.
155	0.0132	0.0211	0.8991	0.0244	0.0422	smurf.
156	0.0132	0.0211	0.8992	0.0244	0.0422	smurf.
157	0.0129	0.0207	0.9009	0.0240	0.0414	smurf.
158	0.0129	0.0207	0.9013	0.0239	0.0413	smurf.
159	0.0129	0.0206	0.9016	0.0238	0.0411	smurf.
160	0.0127	0.0203	0.9032	0.0235	0.0404	smurf.
161	0.0126	0.0203	0.9035	0.0234	0.0402	smurf.
162	0.0126	0.0202	0.9037	0.0234	0.0401	smurf.
163	0.0124	0.0199	0.9055	0.0230	0.0393	smurf.
164	0.0123	0.0198	0.9057	0.0229	0.0392	smurf.
165	0.0122	0.0196	0.9072	0.0226	0.0385	smurf.
166	0.0121	0.0195	0.9074	0.0225	0.0384	smurf.
167	0.0121	0.0195	0.9077	0.0225	0.0383	smurf.
168	0.0119	0.0192	0.9090	0.0222	0.0377	smurf.
169	0.0119	0.0192	0.9093	0.0221	0.0375	smurf.
170	0.0117	0.0190	0.9105	0.0218	0.0370	smurf.
171	0.0117	0.0189	0.9105	0.0218	0.0370	smurf.
172	0.0117	0.0189	0.9107	0.0218	0.0369	smurf.
173	0.0074	0.0122	0.9430	0.0140	0.0234	smurf.
174	0.0074	0.0122	0.9431	0.0140	0.0233	smurf.
175	0.0046	0.0077	0.9640	0.0089	0.0149	smurf.
176	0.0046	0.0077	0.9641	0.0088	0.0148	smurf.
177	0.0046	0.0077	0.9641	0.0088	0.0148	smurf.
178	0.0045	0.0076	0.9644	0.0088	0.0147	smurf.
179	0.0024	0.0041	0.9809	0.0047	0.0079	smurf.
180	0.0024	0.0040	0.9811	0.0047	0.0079	smurf.
181	0.0024	0.0040	0.9811	0.0047	0.0078	smurf.
182	0.0024	0.0040	0.9811	0.0047	0.0078	smurf.
183	0.0009	0.0016	0.9924	0.0019	0.0032	smurf.
184	0.0009	0.0016	0.9924	0.0019	0.0032	smurf.

185	0.0003	0.0005	0.9978	0.0005	0.0009	smurf.
186	0.0003	0.0005	0.9978	0.0005	0.0009	smurf.
187	0.0003	0.0005	0.9978	0.0006	0.0009	smurf.
188	0.0003	0.0005	0.9975	0.0006	0.0010	smurf.
189	0.0004	0.0007	0.9966	0.0008	0.0014	smurf.
190	0.0005	0.0008	0.9962	0.0009	0.0016	smurf.
191	0.0005	0.0008	0.9962	0.0009	0.0016	smurf.
192	0.0005	0.0008	0.9962	0.0009	0.0016	smurf.
193	0.0014	0.0025	0.9882	0.0029	0.0049	smurf.
194	0.0014	0.0025	0.9882	0.0029	0.0050	smurf.
195	0.0014	0.0026	0.9881	0.0030	0.0050	smurf.
196	0.0015	0.0027	0.9875	0.0031	0.0052	smurf.
197	0.0015	0.0027	0.9874	0.0031	0.0053	smurf.
198	0.0032	0.0058	0.9727	0.0068	0.0114	smurf.
199	0.0032	0.0059	0.9726	0.0068	0.0115	smurf.
200	0.0032	0.0059	0.9725	0.0069	0.0115	smurf.
201	0.0033	0.0061	0.9717	0.0071	0.0118	smurf.
202	0.0057	0.0104	0.9516	0.0121	0.0203	smurf.
203	0.0058	0.0106	0.9508	0.0123	0.0206	smurf.
204	0.0058	0.0106	0.9507	0.0123	0.0206	smurf.
205	0.0058	0.0107	0.9505	0.0124	0.0207	smurf.
206	0.0059	0.0109	0.9496	0.0126	0.0210	smurf.
207	0.0088	0.0163	0.9243	0.0190	0.0316	smurf.
208	0.0089	0.0166	0.9233	0.0192	0.0320	smurf.
209	0.0122	0.0230	0.8937	0.0267	0.0444	smurf.
210	0.0122	0.0230	0.8935	0.0268	0.0444	smurf.
211	0.0124	0.0233	0.8926	0.0270	0.0447	smurf.
212	0.0160	0.0305	0.8596	0.0354	0.0585	smurf.
213	0.0162	0.0307	0.8588	0.0356	0.0587	smurf.
214	0.0162	0.0307	0.8586	0.0357	0.0588	smurf.
215	0.0162	0.0308	0.8585	0.0357	0.0588	smurf.

216	0.0163	0.0311	0.8572	0.0361	0.0593	smurf.
217	0.0164	0.0311	0.8570	0.0361	0.0594	smurf.
218	0.0165	0.0314	0.8560	0.0364	0.0596	smurf.
219	0.0165	0.0315	0.8558	0.0365	0.0597	smurf.
220	0.0165	0.0315	0.8557	0.0365	0.0597	smurf.
221	0.0205	0.0395	0.8199	0.0457	0.0744	smurf.
222	0.0205	0.0395	0.8197	0.0458	0.0744	smurf.
223	0.0206	0.0396	0.8195	0.0459	0.0745	smurf.
224	0.0207	0.0399	0.8186	0.0462	0.0747	smurf.
225	0.0247	0.0478	0.7832	0.0553	0.0891	smurf.
226	0.0248	0.0481	0.7823	0.0556	0.0892	smurf.
227	0.0248	0.0481	0.7821	0.0557	0.0892	smurf.
228	0.0249	0.0482	0.7820	0.0557	0.0893	smurf.
229	0.0250	0.0485	0.7810	0.0560	0.0895	smurf.
230	0.0331	0.0651	0.7076	0.0753	0.1189	smurf.
231	0.0331	0.0651	0.7080	0.0752	0.1186	smurf.
232	0.0331	0.0651	0.7075	0.0753	0.1189	smurf.
233	0.0331	0.0650	0.7083	0.0751	0.1184	smurf.
234	0.0026	0.9803	0.0050	0.0050	0.0071	ipsweep.
235	0.0015	0.9885	0.0029	0.0029	0.0042	ipsweep.
236	0.0020	0.9848	0.0039	0.0038	0.0055	ipsweep.
237	0.0014	0.9891	0.0028	0.0027	0.0040	ipsweep.
238	0.0615	0.4473	0.1327	0.1323	0.2261	ipsweep.
239	0.0013	0.9898	0.0026	0.0026	0.0037	ipsweep.
240	0.0016	0.9877	0.0031	0.0031	0.0045	ipsweep.
241	0.0012	0.9906	0.0024	0.0024	0.0034	ipsweep.
242	0.0015	0.9885	0.0029	0.0029	0.0042	ipsweep.
243	0.0011	0.9913	0.0022	0.0022	0.0032	ipsweep.
244	0.0012	0.9908	0.0023	0.0023	0.0034	ipsweep.
245	0.0010	0.9920	0.0020	0.0020	0.0029	ipsweep.
246	0.0011	0.9915	0.0021	0.0022	0.0031	ipsweep.

247	0.0009	0.9928	0.0019	0.0018	0.0026	ipsweep.
248	0.0010	0.9920	0.0020	0.0020	0.0029	ipsweep.
249	0.0008	0.9939	0.0015	0.0015	0.0022	ipsweep.
250	0.0010	0.9926	0.0018	0.0019	0.0027	ipsweep.
251	0.0008	0.9939	0.0016	0.0015	0.0022	ipsweep.
252	0.0009	0.9932	0.0017	0.0017	0.0025	ipsweep.
253	0.0007	0.9945	0.0014	0.0014	0.0020	ipsweep.
254	0.0008	0.9938	0.0016	0.0016	0.0023	ipsweep.
255	0.0006	0.9951	0.0013	0.0012	0.0018	ipsweep.
256	0.0007	0.9943	0.0014	0.0014	0.0021	ipsweep.
257	0.0006	0.9956	0.0011	0.0011	0.0016	ipsweep.
258	0.0006	0.9957	0.0011	0.0011	0.0016	ipsweep.
259	0.0005	0.9961	0.0010	0.0010	0.0014	ipsweep.
260	0.0005	0.9962	0.0009	0.0010	0.0014	ipsweep.
261	0.0004	0.9966	0.0009	0.0009	0.0012	ipsweep.
262	0.0004	0.9967	0.0008	0.0009	0.0012	ipsweep.
263	0.0004	0.9970	0.0008	0.0008	0.0011	ipsweep.
264	0.0004	0.9971	0.0007	0.0007	0.0011	ipsweep.
265	0.0003	0.9974	0.0007	0.0007	0.0009	ipsweep.
266	0.0003	0.9974	0.0006	0.0007	0.0009	ipsweep.
267	0.0003	0.9978	0.0006	0.0006	0.0008	ipsweep.
268	0.0003	0.9978	0.0006	0.0006	0.0008	ipsweep.
269	0.0002	0.9981	0.0005	0.0005	0.0007	ipsweep.
270	0.0003	0.9978	0.0006	0.0006	0.0008	ipsweep.
271	0.0003	0.9980	0.0005	0.0005	0.0007	ipsweep.
272	0.0002	0.9983	0.0004	0.0004	0.0006	ipsweep.
273	0.0002	0.9986	0.0004	0.0004	0.0005	ipsweep.
274	0.0002	0.9985	0.0004	0.0004	0.0005	ipsweep.
275	0.0002	0.9988	0.0003	0.0003	0.0004	ipsweep.
276	0.0002	0.9987	0.0003	0.0003	0.0005	ipsweep.
277	0.0001	0.9990	0.0003	0.0003	0.0004	ipsweep.

278	0.0001	0.9989	0.0003	0.0003	0.0004	ipsweep.
279	0.0001	0.9991	0.0002	0.0002	0.0003	ipsweep.
280	0.0001	0.9990	0.0002	0.0003	0.0004	ipsweep.
281	0.0002	0.9985	0.0004	0.0004	0.0006	ipsweep.
282	0.0002	0.9985	0.0004	0.0004	0.0005	ipsweep.
283	0.0002	0.9982	0.0005	0.0005	0.0006	ipsweep.
284	0.0002	0.9983	0.0004	0.0004	0.0006	ipsweep.
285	0.0003	0.9979	0.0005	0.0005	0.0007	ipsweep.
286	0.0003	0.9980	0.0005	0.0005	0.0007	ipsweep.
287	0.0003	0.9976	0.0006	0.0006	0.0009	ipsweep.
288	0.0003	0.9976	0.0006	0.0006	0.0009	ipsweep.
289	0.0003	0.9973	0.0007	0.0007	0.0010	ipsweep.
290	0.0003	0.9973	0.0007	0.0007	0.0010	ipsweep.
291	0.0004	0.9969	0.0008	0.0008	0.0011	ipsweep.
292	0.0004	0.9969	0.0008	0.0008	0.0011	ipsweep.
293	0.0004	0.9965	0.0009	0.0009	0.0013	ipsweep.
294	0.0005	0.9964	0.0009	0.0009	0.0013	ipsweep.
295	0.0005	0.9957	0.0011	0.0011	0.0015	ipsweep.
296	0.0005	0.9957	0.0011	0.0011	0.0015	ipsweep.
297	0.0006	0.9956	0.0011	0.0011	0.0016	ipsweep.
298	0.0006	0.9955	0.0011	0.0012	0.0016	ipsweep.
299	0.0006	0.9951	0.0013	0.0013	0.0018	ipsweep.
300	0.0006	0.9950	0.0013	0.0013	0.0018	ipsweep.
301	0.0007	0.9945	0.0014	0.0014	0.0020	ipsweep.
302	0.0007	0.9944	0.0014	0.0014	0.0020	ipsweep.
303	0.0008	0.9939	0.0016	0.0016	0.0022	ipsweep.
304	0.0008	0.9938	0.0016	0.0016	0.0022	ipsweep.
305	0.0008	0.9934	0.0017	0.0017	0.0024	ipsweep.
306	0.0009	0.9931	0.0017	0.0018	0.0025	ipsweep.
307	0.0009	0.9928	0.0019	0.0019	0.0026	ipsweep.
308	0.0010	0.9925	0.0019	0.0019	0.0027	ipsweep.

309	0.0010	0.9921	0.0020	0.0020	0.0028	ipsweep.
310	0.0010	0.9918	0.0021	0.0021	0.0030	ipsweep.
311	0.0011	0.9914	0.0022	0.0022	0.0031	ipsweep.
312	0.0011	0.9910	0.0023	0.0023	0.0032	ipsweep.
313	0.0012	0.9906	0.0024	0.0024	0.0034	ipsweep.
314	0.0012	0.9903	0.0025	0.0025	0.0035	ipsweep.
315	0.0006	0.0012	0.0015	0.0016	0.9951	pod.
316	0.0005	0.0010	0.0013	0.0013	0.9960	pod.
317	0.0004	0.0009	0.0012	0.0012	0.9963	pod.
318	0.0004	0.0007	0.0010	0.0010	0.9970	pod.
319	0.0003	0.0007	0.0009	0.0009	0.9972	pod.
320	0.0003	0.0006	0.0008	0.0008	0.9974	pod.
321	0.0002	0.0005	0.0007	0.0007	0.9979	pod.
322	0.0002	0.0005	0.0006	0.0006	0.9981	pod.
323	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
324	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
325	0.0002	0.0003	0.0005	0.0005	0.9985	pod.
326	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
327	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
328	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
329	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
330	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
331	0.0002	0.0004	0.0006	0.0006	0.9981	pod.
332	0.0002	0.0005	0.0007	0.0006	0.9980	pod.
333	0.0003	0.0005	0.0007	0.0007	0.9978	pod.
334	0.0003	0.0006	0.0009	0.0009	0.9973	pod.
335	0.0006	0.0012	0.0015	0.0016	0.9951	pod.
336	0.0005	0.0010	0.0013	0.0013	0.9960	pod.
337	0.0004	0.0009	0.0012	0.0012	0.9963	pod.
338	0.0004	0.0007	0.0010	0.0010	0.9970	pod.
339	0.0003	0.0007	0.0009	0.0009	0.9972	pod.

340	0.0003	0.0006	0.0008	0.0008	0.9974	pod.
341	0.0002	0.0005	0.0007	0.0007	0.9979	pod.
342	0.0002	0.0005	0.0006	0.0006	0.9981	pod.
343	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
344	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
345	0.0002	0.0003	0.0005	0.0005	0.9985	pod.
346	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
347	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
348	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
349	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
350	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
351	0.0002	0.0004	0.0006	0.0006	0.9981	pod.
352	0.0002	0.0005	0.0007	0.0006	0.9980	pod.
353	0.0003	0.0005	0.0007	0.0007	0.9978	pod.
354	0.0003	0.0006	0.0009	0.0009	0.9973	pod.
355	0.0006	0.0012	0.0015	0.0016	0.9951	pod.
356	0.0005	0.0010	0.0013	0.0013	0.9960	pod.
357	0.0004	0.0009	0.0012	0.0012	0.9963	pod.
358	0.0004	0.0007	0.0010	0.0010	0.9970	pod.
359	0.0003	0.0007	0.0009	0.0009	0.9972	pod.
360	0.0003	0.0006	0.0008	0.0008	0.9974	pod.
361	0.0002	0.0005	0.0007	0.0007	0.9979	pod.
362	0.0003	0.0007	0.0009	0.0009	0.9972	pod.
363	0.0003	0.0006	0.0008	0.0008	0.9974	pod.
364	0.0002	0.0005	0.0007	0.0007	0.9979	pod.
365	0.0002	0.0005	0.0006	0.0006	0.9981	pod.
366	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
367	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
368	0.0002	0.0003	0.0005	0.0005	0.9985	pod.
369	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
370	0.0002	0.0003	0.0004	0.0004	0.9986	pod.

371	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
372	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
373	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
374	0.0002	0.0004	0.0006	0.0006	0.9981	pod.
375	0.0002	0.0005	0.0007	0.0006	0.9980	pod.
376	0.0003	0.0005	0.0007	0.0007	0.9978	pod.
377	0.0003	0.0006	0.0009	0.0009	0.9973	pod.
378	0.0006	0.0012	0.0015	0.0016	0.9951	pod.
379	0.0005	0.0010	0.0013	0.0013	0.9960	pod.
380	0.0004	0.0009	0.0012	0.0012	0.9963	pod.
381	0.0004	0.0007	0.0010	0.0010	0.9970	pod.
382	0.0003	0.0007	0.0009	0.0009	0.9972	pod.
383	0.0003	0.0006	0.0008	0.0008	0.9974	pod.
384	0.0002	0.0005	0.0007	0.0007	0.9979	pod.
385	0.0002	0.0005	0.0006	0.0006	0.9981	pod.
386	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
387	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
388	0.0002	0.0003	0.0005	0.0005	0.9985	pod.
389	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
390	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
391	0.0002	0.0003	0.0004	0.0004	0.9986	pod.
392	0.0002	0.0004	0.0005	0.0005	0.9985	pod.
393	0.0002	0.0004	0.0005	0.0005	0.9984	pod.
394	0.0002	0.0004	0.0006	0.0006	0.9981	pod.
395	0.0002	0.0005	0.0007	0.0006	0.9980	pod.
396	0.0003	0.0005	0.0007	0.0007	0.9978	pod.
397	0.0003	0.0006	0.0009	0.0009	0.9973	pod.