TESIS – KI092361

PENGELOMPOKAN DATA MENGGUNAKAN *PATTERN REDUCTION ENHANCED ANT COLONY OPTIMIZATION* DAN *KERNEL CLUSTERING*

Dwi Taufik Hidayat
5111201017

PEMBIMBING I
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

PEMBIMBING II
Dr. Ir. R.V. Hari Ginardi, M.Sc.

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI CERDAS & VISUALISASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS – KI092361

DATA CLUSTERING USING PATTERN REDUCTION ENHANCED ANT COLONY OPTIMIZATION AND KERNEL CLUSTERING

Dwi Taufik Hidayat
5111201017

SUPERVISOR
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

CO-SUPERVISOR
Dr. Ir. R.V. Hari Ginardi, M.Sc.

MASTER PROGRAM
INTELLIGENT COMPUTING AND VISUALIZATION
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
DWI TAUFIK HIDAYAT
NRP: 5111 201 017

Tanggal Ujian: 16 Mei 2016
Periode Wisuda: September 2016

Disetujui oleh:

Dr. Eng. Chastine Fatichah., S.Kom, M.Kom.
NIP. 197512202001122002 ..... (Pembimbing I)

Dr. Ir. R.V. Hari Ginardi., M.Sc.
NIP. 196505181992031003 ..... (Pembimbing II)

Dr. Eng. Nanik Suciati., S.Kom, M.Kom.
NIP. 197104281994122001 ..... (Penguji)

Arya Yudhi Wijaya, S.Kom., M.Kom.
NIP. 198409042010121002 ..... (Penguji)

Dini Adni Navastara, S.Kom., M.Sc.
NIP. 198510172015042001 ..... (Penguji)

Direktur Program Pascasarjana,

Prof. Ir. Djauhar Manfaat, M.Sc, Ph.D
NIP. 19601202 198701 1001

PENGELOMPOKAN DATA MENGGUNAKAN *PATTERN REDUCTION ENHANCED ANT COLONY OPTIMIZATION* DAN *KERNEL CLUSTERING*

Nama mahasiswa : Dwi Taufik Hidayat
NRP : 5111201017
Pembimbing I : Dr. Eng. Chastine Fatichah, S.Kom, M.Kom
Pembimbing II : Dr. Ir. R.V. Hari Ginardi, M.Sc

ABSTRAK

Salah satu metode optimasi yang dapat digunakan untuk *clustering* adalah *Ant Colony Optimization*(ACO). Namun memiliki kelemahan pada sisi waktu dan kualitas atau konvergensi solusi yang dihasilkan.

Pada penelitian ini mengaplikasikan metode *Pattern Reduction Enhanced Ant Colony Optimization*(PREACO) digabungkan dengan fungsi *Kernel Gaussian* berdasarkan ACO. Pertama, kami melakukan inisialisasi solusi. Kedua, nilai *pheromone* untuk menentukan centroid secara acak. Ketiga, kami menghitung bobot solusi dan terakhir, melakukan koreksi pusat kluster. Solusi yang telah terbentuk akan dievaluasi melalui fungsi *Kernel Gaussian*. Fungsi *pattern reduction enhanced* penting untuk memastikan nilai *update pheromone* agar maksimal. Langkah-langkah ini akan dilakukan berulang-ulang sampai solusi terbaik tercapai.

Uji coba dilakukan pada beberapa dataset, dengan menggunakan tiga skenario. Pertama, dilakukan untuk mendapat kombinasi parameter yang tepat. Kedua, pengukuran tingkat kesalahan dan kesamaan data dengan menggunakan pengukuran *Sum of Squared Error* dilakukan. Ketiga perbandingan tingkat akurasi metode ACO, ACO dengan Kernel, PREACO, dan PREACO dengan Kernel dilakukan. Hasil pengujian menunjukkan bahwa PREACO dengan Kernel memiliki tingkat akurasi lebih tinggi sebesar 99.8% dari ACO 70.8%, ACO dengan Kernel 99.3%, PREACO 69% untuk data sintesis, dan 93.8% lebih tinggi dari metode lainnya (ACO 89.9%, ACO dengan Kernel 93.3%, PREACO 85%) untuk data wine. Namun, memiliki akurasi lebih rendah dibanding ACO untuk data iris.

Kata kunci: *Kernel Clustering*, *Ant Colony Optimization*, *Pattern Reduction Enhanced Ant Colony Optimization*.

DATA *CLUSTERING* USING PATTERN REDUCTION ENHANCED ANT COLONY OPTIMIZATION AND KERNEL *CLUSTERING*

Student Name : Dwi Taufik Hidayat
Student Identity Number : 5111201017
Supervisor : Dr. Eng. Chastine Fatichah, S.Kom, M.Kom
Co-Supervisor : Dr. Ir. R.V. Hari Ginardi, M.Sc.

ABSTRACT

One of the optimization method used for clustering is Ant Colony Optimization (ACO). However, it has weaknesses in the aspect of time and the solution of convergence which is produced.

This research combines Pattern Reduction Enhanced Ant Colony Optimization (PREACO) method with Gaussian Kernel function based on ACO. First, it sets up initial solution. Second, the magnitude of pheromone is calculated to find the centroid randomly. Third, the solution weight is calculated; and the last, the correction of cluster center. The solution that has been already set up, will be evaluated through Gaussian Kernel function. The function of 'pattern enhanced reduction' is essential to ensure the update of pheromone value in order to be maximum. Those steps will be conducted repeatedly until the best solution is reached.

The evaluation will be conducted with several data sheets by using three scenarios. First, it is conducted to get the precised combination of parameters. Second, The measurement of error level is counted by using Sum of Squared Error. Third, the comparison of the accuracy of ACO, ACO with Kernel, PREACO, and PREACO with Kernel is conducted. The result shows that PREACO with Kernel method has the accuracy level of 99.8%, higher than ACO (70.8%), ACO with Kernel (99.3%), PREACO (69%) for Synthetic data; and for wine data, it has the accuracy level of 93.8%, higher than the other metode (ACO 89.9%, ACO with Kernel 93.3%, PREACO 85%). However, it has lower accuracy than ACO interm of Iris data.

Keywords: *Kernel Clustering, Ant Colony Optimization, Pattern Reduction Enhanced Ant Colony Optimization*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Clustering merupakan pengelompokan data berdasarkan pada kesamaan atau perbedaannya. Kesamaan maupun perbedaan itu dilihat dari ciri yang dipunyai data tersebut. Misalnya terdapat sebuah data termasuk kedalam *cluster* tertentu dikarenakan memiliki ciri yang sama dengan rata-rata data dalam *cluster* itu.

Metode *clustering* berdasarkan sifat pengelompokannya terbagi menjadi dua jenis yaitu *soft clustering* dan *hard clustering*. Maksudnya, satu data bisa termasuk kedalam beberapa *cluster* hal ini disebut dengan *soft clustering*. Sebaliknya, sebuah data harus dimasukkan kedalam satu *cluster* tertentu saja disebut dengan *hard clustering*.

Metode k-means, hierarchical algorithm, SOM(Tan, 2006) adalah beberapa dari beragam metode *clustering*. Metode tersebut dapat dengan baik melakukan *clustering*, tetapi masih memiliki permasalahan yaitu pada penentuan jumlah *cluster* di awal dan juga masih memasukkan sebuah data kedalam beberapa *cluster* yang berbeda. Oleh karena itu akan menjadi masalah ketika sebuah data masuk dalam beberapa *cluster* yang berbeda, itu artinya data tersebut tidak terdefinisi dengan baik pengklusterannya. Oleh sebab itu penentuan jumlah *cluster* diawal perlu dilakukan. Penelitian tentang permasalahan tersebut masih terbuka untuk diteliti. Salah satu metode yang digunakan untuk memecahkan permasalahan itu adalah *kernel clustering* yang mendefinisikan dengan baik jumlah *cluster*, sehingga pada akhirnya sebuah data akan dimasukkan kedalam sebuah *cluster* dengan tepat.

Cluster analisis atau *kernel clustering* saat ini menjadi rujukan dalam melakukan pembelajaran tak terawasi pada proses pencarian *cluster*. Gath dan Geva (Gath, 1989) melakukan penelitian berdasarkan kelemahan ini yaitu dengan menggabungkan fuzzy C-means dan fuzzy maximum likelihood untuk mendapatkan *cluster* yang tak terawasi. Lorette dkk (Lorette, 2000) mengajukan sebuah metode fuzzy *clustering* untuk menentukan jumlah *cluster* secara dinamis. Selain itu, banyak penelitian dalam penentuan jumlah *cluster*. Berdasarkan dari kelemahan yang dimiliki K-means terdapat banyak metode yang diajukan yaitu X-

means (Pelleg, 2000) dan G-means (Hamerly, 2003), keduanya bertolak dari penentuan jumlah kelas.

Penelitian dibidang *kernel clustering* banyak mengalami perkembangan. Salah satu metode hasil penelitian tersebut adalah menggabungkan *swarm intelligence* pada *kernel clustering*. Penelitian yang dilakukan oleh (Das,2008), mengusulkan suatu metode gabungan tersebut yaitu *automatic kernel clustering with multi-elitist PSO* (AKC-MEPSO). Penelitian ini menyajikan suatu pengelompokan data dari data set yang kompleks tanpa terlebih dahulu harus memiliki pengetahuan awal mengenai *cluster*-nya. Metode ini menggunakan fungsi kernel yang dapat menggantikan fungsi jarak dan dapat melakukan pengelompokan data dengan baik walau data itu secara linear tidak terpisahkan. Dan juga digabungkan dengan metode *Particle Swarm Optimization* (PSO) yang telah dimodifikasi (MEPSO).

Metode fungsi kernel yang banyak digunakan adalah Polynomial Kernel, *Kernel gaussian*, Radial basis Kernel and Hyper tangent (M.Tushir, 2010). Dengan dilakukannya penelitian oleh (Kuo, 2014) yang menggunakan kernel function digabungkan dengan algoritma *bee colony optimization* (BCO) notabene merupakan algoritma *evolutionary algorithm*. Hal ini membuktikan bahwa kernel function dan evolutionary algorithm dapat dikolaborasikan untuk menentukan jumlah *cluster* secara tepat. Penggabungan yang telah dilakukannya disebut dengan AKC-BCO. Dari metode AKC-BCO dan AKC-MEPSO membuka peluang untuk pengembangannya. Penggabungan fungsi kernel dengan metode *swarm intelligence* atau *evolutionary algorithm* lain dapat dilakukan.

Penggunaan algoritma *evolutionary algorithm* membuka peluang akan jenis penelitian lain. Penelitian *Ant Colony Optimization* (ACO) merupakan bagian dari algoritma itu. Sehingga dapat digunakan pada penelitian ini untuk menambah kontribusi ilmu. Metode ini diperkenalkan dan ditemukan oleh (Dorigo, 1996) dan masih berkembang sampai saat ini.

Penelitian mengenai *Ant Colony Optimization* (ACO) telah disarikan oleh (Aditi, 2008). Walaupun dalam mensarikannya dibandingkan dengan kemampuan yang dimiliki oleh BCO. Pada dasarnya ACO lebih menjamin dari sisi konvergensi dan mampu beradaptasi dengan lingkungannya daripada BCO. Tetapi dari segi

waktu dan komputasi masih unggul BCO. Oleh karena itu terdapat pengembangan metode pada ACO juga banyak terjadi. Salah satunya *Pattern Reduction Enhanced Ant Colony Optimization* (PREACO) yang diusulkan oleh (Chun, 2013) membuktikan bahwa ACO juga dapat lebih bagus dari sisi biaya komputasi dan waktu. Walaupun begitu, hasil yang diperoleh mengalami penurunan akurasi sehingga memerlukan pengembangan selanjutnya.

Oleh karena itu penelitian ini akan melakukan pengelompokan data menggunakan PREACO dan *kernel clustering*. Diharapkan proses *clustering* memiliki waktu komputasi cepat dan kepastian dari sisi konvergensinya. Pengujian dari rumusan ini menggunakan data set standar yang juga dilakukan peneliti lain (Kuo, 2014) yaitu seperti data set iris, wine dan data sintesis.

Metode PREACO mempercepat proses pemilihan solusi dan menghindari redundansi pemilihan solusi untuk mendapatkan hasil *cluster* dan proses *clustering* yang tepat. Adapun fungsi kernel yang digunakan adalah *kernel gaussian* berdasarkan penelitian (Kuo, 2014). Pengujian akan dilakukan terhadap konvergensi metode dan juga akurasinya. Metode ini diujicobakan kepada data set standar klasifikasi yang terdapat pada UCI data set.

1.2 Perumusan Masalah

1. Bagaimana metode PREACO dan K*ernel Clustering* dapat memperbaiki ACO dalam melakukan pengelompokan data?
2. Bagaimana menguji kinerja dari metode yang diusulkan pada beberapa data set?

1.3 Batasan Masalah

1. Data yang diuji coba adalah data standar yang biasa diujikan antara lain data set sintesis, iris, dan wine.
2. Dalam mengaplikasikan metode menggunakan pemrograman java.

1.4 Tujuan Penelitian

Tujuan dari Penelitian ini adalah memperbarui metode Ant Colony Optimization untuk *clustering* melalui penerapan metode PREACO dan *Gaussian*

Kernel, selain itu bertujuan untuk menguji kinerja dari metode yang telah diperbarui.

1.5 Manfaat Penelitian

Hasil penelitian ini dapat dimanfaatkan untuk mendapatkan *cluster* data secara tepat dengan tingkat konvergensi akurat dan dapat digunakan pada aplikasi industri, marketing atau bisnis, dan lainnya untuk pengenalan pola data.

1.6 Kontribusi Penelitian

Kontribusi penelitian ini adalah mengajukan metode PREACO dan *Kernel Clustering* untuk pengelompokan data.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini dijelaskan teori mengenai dasar dalam pengerjaan tesis. Dasar teori yang digunakan meliputi *clustering*, *kernel clustering*, *ant colony optimization*, *pattern reduction enhanced ant colony optimization*.

2.1 *Clustering*

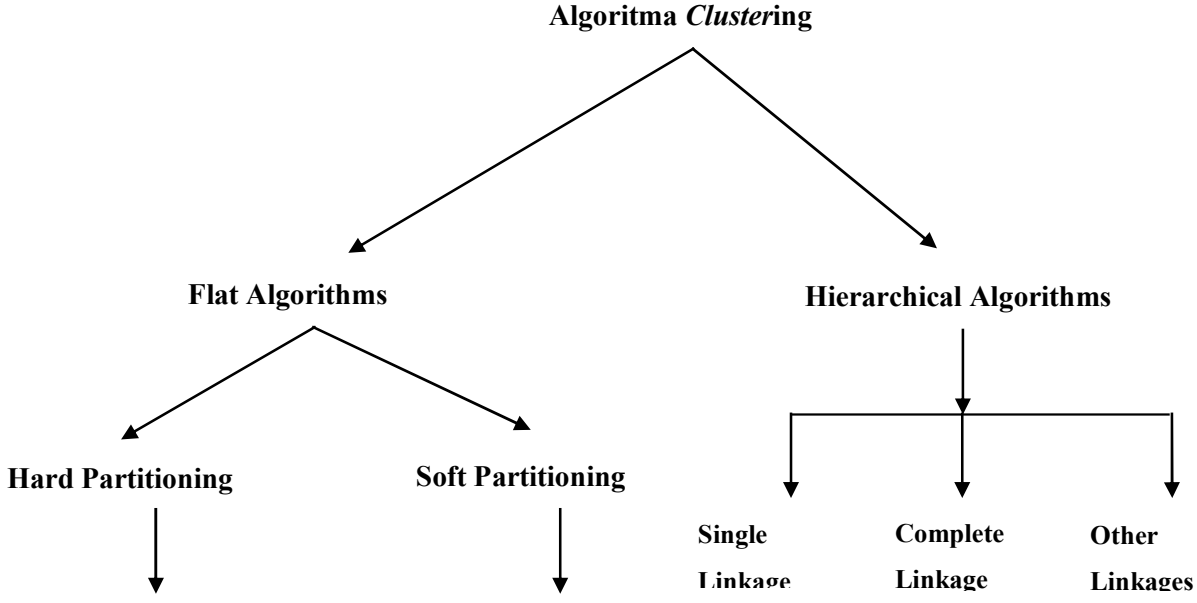
Analisa *cluster* adalah memilih dan mengelompokkan data kedalam kelompok atau *cluster* tertentu. Pada dasarnya untuk dapat melakukan *clustering* atau pengelompokan data dilakukan dengan memaksimalkan sifat homogen yang ada pada data dan faktor external data. Dalam arti *cluster* itu memiliki homogenitas yang tinggi didalamnya dan memiliki faktor pemisah antar *cluster*. Dapat dikatakan bahwa dalam satu *cluster* terdapat kesamaan antar data dan memiliki perbedaan data dengan *cluster* lain. Oleh karena itu semakin sama data maka masuk dalam *cluster* yang sama sedangkan semakin berbeda data maka termasuk kedalam *cluster* lain.

Memisahkan data kedalam kelompok yang sama ciri dan sifatnya dengan baik dapat dikatakan termasuk bagian dari pembelajaran. Pembelajaran jenis ini dikategorikan sebagai pembelajaran tak terawasi. Hal ini berbeda dengan klasifikasi, yang didalamnya sudah ada pembeda dalam bentuk kelas alias sudah ada petunjuk. Pembelajaran berguna untuk mengenali sesuatu, sebagai contoh seorang anak perlu dapat membedakan antara gajah dengan kuda. Pada keduanya terdapat ciri yang berbeda, oleh karena itu diperlukan proses dalam pengenalan ciri masing-masing seperti halnya ciri gajah berbeda dengan ciri yang dimiliki kuda. Untuk membuat dan merumuskan bagaimana cara untuk dapat membedakan data dengan baik perlu dilakukan kajian atau penelitian. Hasil dari penelitian itu digunakan pada sebuah sistem untuk mempelajari sesuatu, maka dapat dengan mudah mempelajarinya.

Penelitian dalam bidang analisa *cluster* telah banyak dilakukan. Hal ini dilakukan untuk mendapatkan metode atau algoritma dalam melakukan pengelompokan data alami secara tepat. Telah banyak metode dalam analisa *cluster* yang telah diajukan memiliki keunggulan dan kelemahan. Sehingga dalam bidang ini selalu membuka peluang dalam pengembangannya. Salah satu peluang

penelitian yang masih dapat dikembangkan adalah dalam penentuan jumlah *cluster*. Karena penentuan jumlah *cluster* yang tepat, penting dilakukan untuk mengelompokkan data dan memisahkan data dengan tepat pula.

Hasil dari sebuah penelitian analisa *cluster* adalah dalam bentuk metode atau algoritma. K-means, adalah salah satunya. Metode ini menentukan *cluster* dengan menentukan jumlah *cluster* sesuai yang diinginkan yaitu menentukan nilai K. Tetapi hal ini menjadi kelemahannya, karena tidak bisa menentukan dengan baik jumlah *cluster* secara otomatis, selain itu juga terdapat banyak kelemahan didalamnya. Gath dan Geva (Gath, 1989) melakukan penelitian berdasarkan dari kelemahan ini yaitu dengan menggabungkan fuzzy C-means dan fuzzy maximum likelihood untuk mendapatkan *cluster* yang tak terawasi. Lorette dkk (Lorette, 2000) mengajukan sebuah metode fuzzy *clustering* untuk menentukan jumlah *cluster* secara dinamis. Selain itu juga banyak penelitian dalam penentuan jumlah *cluster*. Berdasarkan kelemahan yang dimiliki K-means terdapat banyak metode yang diajukan yaitu X-means (Pelleg, 2000) dan G-means (Hamerly, 2003), keduanya bertolak dari penentuan jumlah kelas. Pada gambar 2.1 merupakan gambar jenis metode turunan analisa *clustering*.



Gambar 2.1 Jenis Algoritma *Clustering*

Pada gambar 2.1 metode K-means termasuk dalam kategori *hard partitioning*. Didalamnya juga terdapat Self Organizing Maps dan DBSCAN. Pada algoritma K-means nilai K ditentukan pada awal algoritma. Dan hasil dari metode hard partitioning adalah berupa penentuan kelompok data secara pasti atau *crisp*. Sebagai contoh akan dijelaskan bagian dari metode *clustering hard partitioning*.

2.1.1 K-Means

Algoritma *clustering* K-Means berasal dari pengelompokan data secara statistik. Metode ini berusaha memisahkan dan mengelompokkan data kedalam *cluster* yang memiliki kesamaan. Penentuan tingkat kesamaan dapat berasal dari berbagai ciri. Kesamaan yang paling mudah digunakan adalah berdasarkan jarak antar data *Euclidean Distance*.

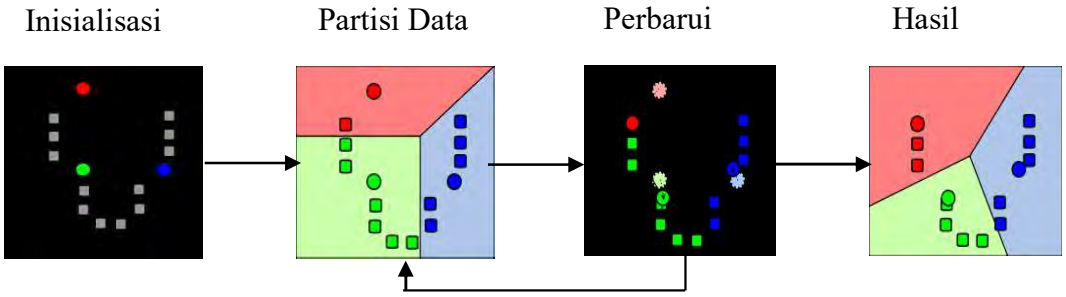
Berikut langkah algoritma K-Means :

1. Terlebih dahulu menentukan nilai K sebagai perwakilan jumlah *cluster* yang akan dicari.
2. Memilih data secara acak sebagai pusat *cluster* awal.
3. Menggunakan jarak *Euclidean Distance* untuk menghitung jarak terdekat antara data dengan pusat *cluster*. Data akan terkelompokkan sesuai jarak minimalnya terhadap pusat *cluster*.
4. Dengan menggunakan langkah 3 yang telah terbentuk, kemudian menghitung ulang pusat *cluster* dengan menggunakan *mean* dari masing-masing *cluster*.
5. Jika pusat yang baru terbentuk sama dengan langkah 3, maka proses pencarian selesai. Jika tidak, maka proses mengulang dari langkah 3-5.

Langkah pertama membutuhkan penentuan nilai K sebagai bentuk awal jumlah *cluster* yang diinginkan pada set data. Berikutnya, algoritma akan secara acak menentukan berapa jumlah K sebagai pusat *cluster*. Untuk kemudian data akan dikumpulkan pada *cluster* yang memiliki kesamana dengan pusat *cluster*. Pengukuran kesamaan data dapat dilakukan menggunakan berbagai cara, salah satu cara yang paling mudah adalah menggunakan rumus jarak *Euclidean Distance*.

Setelah mengumpulkan data pada *cluster* masing-masing, langkah berikutnya adalah memperbarui pusat *cluster*. Cara untuk memperbarui pusat

cluster adalah salah satunya menggunakan rata-rata (mean) dari semua data dalam satu *cluster*. Perbaruan pusat *cluster* terus dilakukan sepanjang iterasi atau sampai pada kesamaan antara pusat *cluster* yang baru dengan pusat *cluster* yang sama. Sedangkan visualisasi dari metode K-Means dapat dilihat pada gambar 2.2.



Gambar 2.2 Visualisasi Proses K-Means

Ketika K-Means melakukan penghitungan kesamaan antara pusat data dengan data dihitung menggunakan kernel. Kernel yang digunakan pada K-Means menghitung jarak antara data dengan pusat *cluster* kemudian dipilih jarak paling kecil atau minimal. Persamaan yang digunakan adalah persamaan (2.1). Dengan *Xi* data dalam *cluster* yang sama dengan pusat *cluster Mi*.

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{p=1}^{d} (x_{i,p} - x_{j,p})^2} = \left\| \vec{x}_i - \vec{x}_j \right\|. \qquad (2.1)$$

2.1.2 Kernel *Clustering*

Dalam beberapa tahun terakhir, metode Kernel telah mendapat perhatian utama, terutama karena meningkatnya popularitas Vector Machines Support. Fungsi kernel dapat digunakan dalam berbagai aplikasi karena mereka menyediakan jembatan sederhana dari linearitas non-linearitas untuk algoritma yang dapat dinyatakan dalam dot produk.

Metode Kernel adalah kelas algoritma untuk analisis pola, yang elemen paling dikenal adalah *support vector machine* (SVM). Tugas umum analisis pola adalah untuk menemukan dan mempelajari jenis umum hubungan (seperti *cluster*, peringkat, komponen utama, korelasi, klasifikasi) pada jenis umum data (seperti urutan, dokumen teks, set titik, vektor, gambar, grafik, dll).

Analisa *Clustering* kategori *hard partitioning* mencoba melakukan pemisahan data menuju kelompok data yang sama secara pasti. Misalnya X = {

$\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ adalah sekumpulan data yang belum terkelompokkan dengan dimensi tertentu. Masing-masing data $x_{i,j}$ dalam sebuah vektor X_i memiliki atribut atau fitur (j=1,2,...,d) dengan (i=1,2,...,n). Untuk kemudian data-data ini mencoba memasukkan kedalam sebuah partisi *cluster* $C=\{C_1,C_2,...,C_k\}$ dari partisi k. Kondisi terpisahkan dengan baik ketika tingkat kesamaan antar data dalam satu *cluster* maksimum dan jarak antar pusat *cluster* sejauh mungkin. Oleh karena itu dibutuhkan sebuah fungsi kernel yang harus bisa memetakkan data dengan baik.

Fungsi Kernel harus kontinyu, simetris, dan yang lebih tepat harus memiliki positif (semi- definit) *Gram matrix*. Kernel yang dikatakan memenuhi teorema Mercer adalah semi-definit positif, yang berarti Matrik kernel mereka tidak memiliki nilai Eigen non-negatif. Penggunaan pasti kernel positif menjamin bahwa masalah optimasi akan cembung dan solusi akan menjadi unik.

Macam dasar kernel *clustering* yang sering digunakan adalah Mercer Kernel. Mercer kernel terdapat dua jenis yaitu Polynomial dan Gaussian. Masing-masing memiliki karakteristik tersendiri. Untuk rumusan kedua kernel dapat dilihat pada persamaan (2.2) dan (2.3).

$$K^{(p)}(x_i, x_j) = (1 + x_i \cdot x_j)^p, p \in \mathrm{N}. \tag{2.2}$$

$$K^{(g)}(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right), \sigma \in R. \tag{2.3}$$

Berdasarkan berbagai kernel yang ada dapat digunakan untuk berbagai keperluan *clustering*. Salah satunya adalah *Kernel gaussian* (Das, 2008 dan Kuo, 2014) atau lebih dikenal dengan istilah Radial Basis Function yang digunakan pada proses *clustering.* Penelitian tersebut kernel yang digunakan lebih akurat dalam melakukan pengelompokan data pada berbagai permasalahan daripada Kernel Linear dan Polynomial. *Kernel gaussian* juga digunakan dalam penelitian (Das, 2008) melalui metode algoritma MEPSO (Multi-Elitis Particle Swarm Optimization). Penelitian tersebut diperkenalkan istilah CS Kernel sebagai fungsi obyektif dalam perhitungan kernelnya.

Dalam algoritma MEPSO *Kernel gaussian* digunakan pada perhitungan *kernel clustering* dalam melakukan proses *clustering*. Metode yang diadaptasi dari metode lama *Particle Swarm Optimization* ini lebih menekankan pada kernel.

Kernel gaussian digunakan untuk menggantikan algoritma kesamaan dasar yaitu rumus jarak *Euclidean Distance* (2.1). *Kernel gaussian* menggunakan sigma (σ)>0 dan mengkombinasikannya dengan persamaan jarak kemudian dimasukkan pada persamaan CS Kernel. Sedangkan CS kernel merupakan *Cluster validity index* yang sering diterapkan untuk mengukur tingkat validitas dari hasil *clustering*.

Dengan menggunakan CS Kernel dipercaya mampu menangani pengelompokan data (*clustering*) pada data yang memiliki perbedaan densitas dan/atau ukuran daripada rumusan validitas lainnya. Sebagai akibatnya terdapat memerlukan komputasi lebih sesuai dengan peningkatan nilai *k* dan *n* (Chou,2004).

2.1.3 Nilai Index Validasi *Clustering*

Nilai index validasi *clustering* pada dasarnya merupakan metode atau fungsi untuk mengevaluasi hasil *clustering* secara matematika statistik. Pada umumnya, index validasi *clustering* memiliki tujuan yaitu digunakan untuk menentukan jumlah *cluster* dan kedua digunakan menentukan partisi antar *cluster* yang tepat. Pendekatan tradisional adalah dengan mencoba beberapa jumlah *cluster* yang berbeda pada suatu algoritma untuk kemudian menilai dan memilih tingkat validasi yang baik.

Fungsi index validasi untuk membuat semakin baik dalam melakukan partisi data. Dengan melihat dua karakter utama yaitu : Kohesi dan pemisahan (Das, 2008). Kohesi artinya memiliki kesamaan setinggi mungkin dalam satu *cluster*. Nilai variasi fitness dari *cluster* yang terbentuk mengindikasikan kohesinya. Sedangkan pemisahan berarti memiliki keterpisahan antar *cluster* yang baik. Sehingga denga dua hal ini index validasi dapat digunakan untuk menguji optimalisasi dari suatu metode *clustering*.

Dengan adanya berbagai index validasi menunjukkan performasi dari suatu algoritma *clustering*. Seperti Dunn's Index, Calinski-Harabasz index, DB Index dan CS (*Classification and Separation*) Index. Baik itu maksimum atau minimum nilai yang dikeluarkan oleh metode index tersebut menunjukkan bahwa partisi data hasil *clustering* optimal. Metode index validasi tersebut cocok digunakan digabungkan dengan berbagai algoritma optimasi seperti GA, PSO, dan lainnya.

2.1.3.1 Rumus CS Index

Menurut penelitian (Chou dkk,2004) telah mengusulkan suatu metode index validasi CS untuk melakukan evaluasi terhadap algoritma *clustering*. Metode ini pada awalnya adalah gabungan antara Index validasi CE dan S. Baik CE (*Classification Entropy*) dan S (*Separation Measure*) tidak dapat menghasilkan partisi yang baik. Oleh karena itu muncul perbaikan dengan istilah CS atau gabunga dari CE dan S.

Untuk dapat menerapkan CS *Measure* index selanjutnya kita sebut CS index atau CS saja ini kedalam sistem atau metode *clustering* terdapat kondisi yang harus dipenuhi. kondisi ini adalah terlebih dahulu harus sudah didapatkan nilai pusat *cluster*. Karena didalamnya mengandung unsur perbandingan jarak antar pusat *cluster*.

Suatu rumus jarak yang mengukur dari satu data ke data lain ($\vec{X}_i, \vec{X}_j$) dilambangkan dengan $d(\vec{X}_i, \vec{X}_j)$. Kemudian rumusan CS index dapat digambarkan sebagai berikut :

$$CS(K) = \frac{\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\left\{d\left(\vec{X}_i, \vec{X}_q\right)\right\}\right]}{\sum_{i=1}^{K}\left[\min_{j \in K, j \neq i}\left\{d\left(\vec{m}_i, \vec{m}_q\right)\right\}\right]}. \qquad (2.4)$$

Seperti yang terlihat pada persamaan (2.4) perhitungan tersebut menghitung jarak data dalam *cluster* dibanding dengan jarak antar *cluster*, hal ini memiliki prinsip dasar yang sama dengan validasi index DI dan DB.

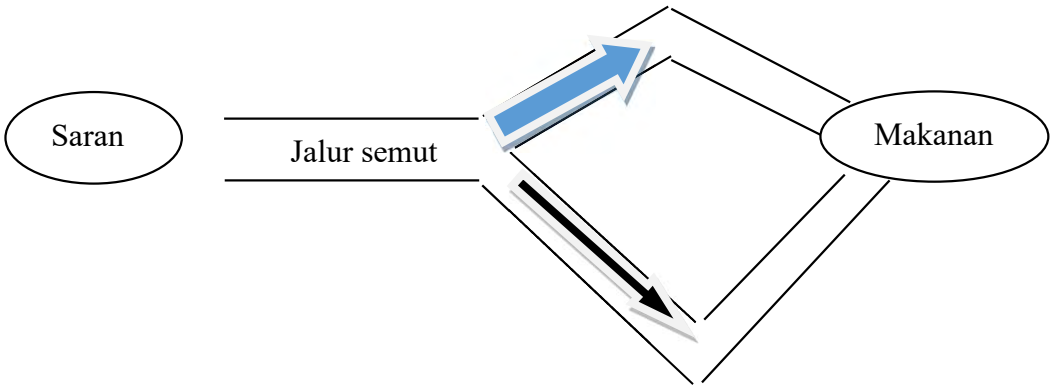
Berdasarkan index validasi oleh (Chou dkk, 2004) memiliki keunggulan yaitu lebih efisien dalam menangani *clustering* data dengan perbedaan densitas data dan/atau ukuran daripada index validasi lain. Senyampang peningkatan komputasi yang tinggi bergantung pada peningkatan nilai K dan n.

2.2. Ant Colony Optimization

Ant Colony Optimization (ACO) merupakan metode optimasi yang diperkenalkan dan ditemukan oleh Dorigo (1996) bersama teman-temannya. Teknik ini berdasarkan *swarm intelligence*, yang mengambil inspirasi dari alam,

dalam hal ini adalah perilaku semut. Perilaku semut dalam rangka mencari makanan dipelajari dan kemudian dijadikan dalam bentuk algoritma. Semut dalam mendapatkan makanannya diberitahukan kepada semut lain untuk dibawa ke sarang. Dalam menentukan rute terbaik menuju sarang, antar semut berkomunikasi melalui zat yang ditinggalkan pada jalan yang dilalui. Zat itu adalah pheromone sebagai zat kimia yang digunakan semut untuk menunjukkan jalan pada sumber makanan. Perilaku semut ini diadopsi dalam metode optimasi oleh (Dorigo, 1996).

Metode optimasi ACO mengadopsi perilaku semut dalam menemukan rute terbaik dari sarang menuju sumber makanan. Pada dasarnya setiap semut yang bertugas membawa makanan mengambil jalan masing-masing dan meninggalkan pheromone untuk diketahui kawannya. Semakin banyak pheromone yang dilewati pada jalan tertentu, maka semakin besar semut lain akan mengambil jalan itu. Seperti terlihat pada gambar 2.3 visualisasi perilaku semut. Panah biru menunjukkan semut lebih memilih jalur terpendek dibandingkan jalur dengan warna anak panah hitam.



Gambar 2.3. Visualisasi perilaku semut mendapatkan sumber makanan

Berdasarkan perilaku semut ini dapat digunakan untuk memecahkan pencarian solusi dari sebuah masalah. Setiap semut melakukan perjalanan untuk mencari sumber makanan atau dapat dianalogikan mencari suatu solusi. Jalur yang tepat oleh semut akan menjadi semut lain dalam menemukan sumber makanan. Sehingga hal ini dapat dianalogikan dengan mendapatkan solusi yang tepat dari berbagai macam solusi yang ada.

Perilaku semut dalam mendapatkan solusi memiliki metode atau cara tersendiri. Solusi yang telah terpilih atau terlewati disimpan dengan memberikan

petunjutk. Petunjuk pada jalan disebut sebagai pheromone. Pheromone ini diberikan oleh setiap semut ketika melewati suatu jalur solusi.

Rangkaian pheromone ini akan diperbarui berdasarkan probabilitas tertentu. Sehingga semakin besar pheromone yang ada, akan menentukan semut berikutnya dalam mengambil arah jalur. Selain itu, apabila salah satu pheromone menjadi pilihan semut lain, maka pheromone lain akan dengan sendirinya hilang dari kemungkinan. Sehingga lambat laun jalur akan tertinggal satu.

Tabel 2.1 Analogi perilaku di alam dengan ilmu komputasinya

Alam	**Ilmu Komputer**
Habitat	Graph (titik dan jalur)
Sarang dan sumber makanan	Titik dalam graph: awal dan akhir
Semut	Agen buatan
Visibility	Jarak contoh *euclidean distance*
Pheromon	Matrik pheromon buatan
Perilaku pengambilan jalur	Titik acak yang dipandu oleh matrik pheromon

Tabel 2.1 merupakan gambaran analogi perilaku semut di alam terhadap sistem ACO. Dari perilaku tersebut memiliki skema yaitu, pertama membangun solusi semut, kedua menentukan jumlah pheromone berdasarkan solusi sebelumnya dan ketiga menentukan jarak. Seperti yang terlihat pada algoritma dibawah.

Algoritma :

1. Mengatur parameter inisialisasi, menentukan jalur pheromone
2. While(kondisi penghentian tak terpenuhi) maka lakukan
3. Temukan solusi
4. Perbarui jalur atau jejak pheromone
5. Fungsi Penentuan rute
6. akhir

Ketika semut berjalan mendapatkan jalur menuju sumber makanan, maka istilah ‘jalur’ pada metode *clustering* sama dengan istilah *node*, *edge*, atau *state*. Setelah menentukan *state*, kewajiban semut berikutnya adalah menentukan langkah mengambil *state* berikutnya. Proses penentuan *state* berikutnya diatur

menggunakan prinsip probabilitas. Persamaan (2.5) merupakan bentuk persamaan itu. Hasil dari persamaan tersebut menentukan pilihan semut atau agen dalam mengambil jalur atau solusi berikutnya.

$$\Pr ob = \frac{\tau(e) \bullet \eta(e)}{\sum_{jalur} \tau(e') \bullet \eta(e')}. \tag{2.5}$$

$$\tau(e) = \begin{Bmatrix} (1-\rho), \tau(e), tidak\_dilalui \\ (1-\rho) \cdot \tau(e) + new\_pheromone, jika\_dilalui \end{Bmatrix}. \tag{2.6}$$

Setiap kali semut memilih suatu jalur atau solusi, maka semut akan memperbarui jumlah pheromone yang telah ada. Hal ini akan terus dilakukan sampai benar-benar solusi ditemukan. Persamaan (2.6) merupakan rumusan dari memperbarui pheromone. Adapun parameter ρ memiliki range nilai paling kecil 0 dan paling tinggi 1 ($0< \rho<1$). Semakin besar nilai *evaporate rate* (1- ρ) maka solusi sebelumnya semakin cepat dilupakan.

Metode Ant Colony Optimization cukup banyak digunakan dalam bidang optimasi. Walaupun GA (Genetic Algorithm) lebih banyak digunakan. Metode berbasis perilaku alam juga dapat dilihat pada optimasi menggunakan perilaku lebah. Bee Colony Optimization (BCO) juga bagus digunakan untuk optimasi. Menurut (Aditi,2000) Terdapat kelebihan dan kekurangan jika dibandingkan dengan optimasi BCO. Pada dasarnya dari sisi *robustness* ACO lebih *powerfull*.

2.2.1. *Ant Colony Optimization* Untuk *Clustering*

ACO untuk *clustering* merupakan metode optimasi yang digunakan untuk melakukan pengelompokan data. Menurut (Shelokar,2004) bahwa melakukan partisi data kedalam partisi yang tepat dapat dilakukan menggunakan perilaku koloni semut. Ketika koloni semut mencari jalan untuk mendapatkan sumber makanan, maka mereka membuat suatu jejak dengan meninggalkan zat pheromon. Zat inilah yang akan memberitahu lokasi dari sumber makanan terdekat. Oleh karena itu penentuan langkah mengikuti pheromon menjadi kunci pemilihan solusi bagi algoritma ini.

Tujuan dari metode ini adalah untuk memisahkan dan memasukkan data sesuai kelompoknya dengan baik dan benar. Untuk melakukan itu digunakan sum square dari jarak euclidean distance antar data dan juga jarak pusat *cluster* dengan

data minimal. Sejumlah *R* agen semut dikirim untuk mencari solusi yang tepat. Pada awal sistem akan diberikan sebuah isian parameter yang harus dipenuhi. Salah satunya adalah jumlah agen semut, evaporation rate, dan jumlah *cluster* (K). Matrik solusi sejumlah agen dibentuk pada awal sistem dengan tidak ada awal solusi.

Setiap *R* mewakili solusi yang terbentuk. Solusi pada awalnya dikosongkan. Sehingga setiap semut akan membawa solusi kosong, dan akan diisi berdasarkan proses ACO. Setiap komponen solusi diisi oleh masing-masing data. Semut akan memetakkan setiap data kedalam *cluster* yang sesuai untuk kemudian disimpan pada matrik solusi. Satu solusi berisikan kumpulan angka *cluster*. Adapun panjang solusi adalah sebanyak N data. Dan matrik kumpulan solusi berisikan solusi dengan ukuran matrik *R* x *N* solusi. Itu berarti matrik solusi terdapat baris sebanyak agen semut dengan jumlah kolom solusi sebanyak data ditambah satu kolom untuk *objective function*-nya. Misalnya, terdapat *N*=8 data, dengan *K*=3 *cluster*. Pada solusi pertama tersebutlah solusi seperti dibawah.

Tabel 2.2 Interpretasi Solusi

	Data Ke							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Cluster Ke	2	1	3	2	2	3	2	1

Berdasarkan deret solusi tersebut, setiap data telah terisi partisi *cluster*. Misalnya pada elemen data pertama terisikan angka 2, ini artinya elemen data pertama dimasukkan kedalam *cluster* 2, elemen data kedua kategori *cluster* 1 begitu seterusnya. Untuk membangun sebuah solusi maka sistem memilih solusi berdasarkan matrik *pheromone* yang ada. Sebelum agen memberikan elemen data *cluster* yang sesuai maka terlebih dahulu melihat matrik *pheromone*, untuk kemudian diputuskan jenis *cluster*nya.

Matrik pheromon menyimpan informasi pheromon yang akan diperbarui setiap kali melakukan iterasi. Pada saat awal sistem berjalan matrik pheromon diberikan angka kecil yang sama yaitu τ_0. Matrik τ_{ij} menggambarkan nilai pheromone data ke i termasuk *cluster* ke j. Oleh karena itu ukuran matrik pheromon adalah *N* x *K*, dimana *N* adalah data dan *K* adala jumlah *cluster*. Ini berarti setiap

data memiliki jumlah pheromon pada setiap *cluster*nya. Dan jumlah pheromon ini akan terus diperbarui sepanjang iterasi sistem ACO.

Setiap iterasi sistem agen semut akan membangun solusi berdasarkan jumlah pheromon pada pheromon matrik. Setelah terbentuk kumpulan solusi, maka dihitung bobot dari solusi yang terbentuk begitu juga pusat *cluster*nya. Agen akan melakukan ini sepanjang jumlah semut. Setelah melalui iterasi sejumlah *R* semut, maka sistem akan melakukan local search untuk mencari solusi yang optimal. Untuk kemudian akan dicari nilai objective functionnya. Langkah berikutnya adalah melakukan pembaruan pheromon. Maka, pada iterasi berikutnya agen semut akan membangun solusi dengan mengikuti matrik pheromon yang terbaru. Sehingga pemilihan solusi berikutnya semakin superior, karena perbaruan pheromon berdasarkan solusi terpilih. Gambaran secara umum sistem ini mewakili bentuk sistem secara keseluruhan seperti yang terlihat pada gambar 2.4.

Gambar 2.4 tentang alur sistem untuk ACO *clustering* menjelaskan bahwa sistem dimulai dari solusi kosong kemudian dilanjutkan dengan pencarian solusi berulang sejumlah *R* agen. Masing-masing agen akan membangun solusi dengan dengan ukuran matrik *R* x *N* ditambah dengan kolom objective function. Untuk kemudian dipilih sejumlah *L* terbaik. Dengan begitu terdapat sejumlah *L* solusi yang akan digunakan untuk mencari solusi optimal melalui local search. Setelah melalui local search, maka sistem akan mendapatkan solusi yang terbaik dan menggunakan solusi ini untuk memperbaiki matrik pheromon. Dari matrik ini akan dilakukan iterasi untuk mendapatkan solusi terbaik. Pembangunan solusi berikutnya berdasarkan matrik pheromon dari solusi yang terbaik. Maka, harusnya generasi solusi berikutnya semakin baik dari sebelumnya, minimal sama. Sebagai yang terakhir, langkah solusi akan disimpan dan dicetak.

Terdapat beberapa langkah yang memerlukan penjelasan lebih detil. langkah itu adalah fungsi membangun solusi, melakukan pembobotan, menghitung pusat *cluster*, menghitung *objective function*, fungsi *local search*, dan mekanisme memperbarui pheromone. Langkah-langkah ini memerlukan penjelasan detil karena prinsip pengerjaannya menggunakan rumusan matematika dan memerlukan kejelasan.

Awal dari sistem adalah proses membangun solusi oleh sejumlah *R* agen semut. Terdapat beberapa cara dalam membangun solusi untuk agen semut. Setiap agen akan menunjuk suatu angka *cluster* kemudian menyimpannya sesuai urutan elemen data. Mulai dengan membuat bilangan acak dari distribusi uniform (0,1) sejumlah data. Dari masing-masing data akan dibandingkan dengan *q0*. *q0* ditentukan diawal, dengan rentang bilangan dari 0 sampai 1 ($0<q0<1$). Adapun cara dalam menentukan solusi adalah sebagai berikut (Shelokar,2004):

1. Terdapat bilangan acak sejumlah data kemudian dibandingkan dengan *q0*. Jika bilangan acak < *q0* maka pilih *cluster* dengan melihat pheromon tertinggi pada matrik pheromone yang bersangkutan. Dan / atau aturan kedua dilakukan.
2. Kedua adalah jika bilangan acak ≥ *q0* maka lakukan generalisasi data pada matrik pheromon untuk kemudian dilakukan simulasi aturan. Menggunakan persamaan generalisasi *pheromone* (2.7).

$$pij = \frac{\tau_{ij}}{\sum_{k=1}^{K} \tau_{ik}}, j = 1,...,K. \qquad (2.7)$$

Sebagai contoh terdapat 8 elemen data dengan jumlah *K*=3 dan *q0*=0.98. Maka, terdapat 8 buah bilangan acak (0.693241, 0.791452, 0.986542, 0.998734, 0.234567, 0.967231, 0.098763, 0.567432). Setelah itu diperiksa masing-masing bilangan acak dengan *q0*. Untuk elemen data (1, 2, 5, 6, 7, 8) mengikuti aturan satu karena bilangan acak < *q0*. Sedangkan untuk elemen data ke 3 dan 4 mengikuti aturan 2.

Untuk elemen 3 dan 4 terlebih dahulu melakukan generalisasi matrik pheromon. Generalisasi pheromon menggunakan rumus persamaan (2.7). Dengan *pij* adalah nilai pheromone pada baris ke i, dan i mewakili urutan elemen data serta kolom ke j mewakili *cluster*. Sebagai contoh terdapat matrik pada awalnya 0.02; 0.02, 0.02 menjadi generalisasi 0.3333, 0.3333, 0.3333. Pheromon pertama sebagai *cluster* 1, pheromon kedua *cluster* 2, dan ketiga *cluster*3. Langkah berikutnya adalah mendapatkan bilangan acak sekali lagi tetapi hanya satu bilangan misalnya 0.853213. Maka berdasarkan prinsip simulai probabilitas dari 0 dan 1. Maka elemen data ke 3 termasuk *cluster* 3. Hal ini berdasarkan prinsip aturan jika bilangan acak termasuk kedalam rentang 0 sampai 0.3333 *cluster* 1. Untuk rentang 0.3334 sampai

0.6666 termasuk *cluster* 2. Dan jika bilangan acak lebih dari 0.6666 maka termasuk *cluster* 3. Hal yang sama diberlakukan untuk elemen data ke 4. Maka solusi yang didapat dapat berupa (1, 1, 2, 3, 1, 1, 1, 1).

Dengan memberlakukan aturan tersebut maka akan didapatkan sebuah solusi yang tidak benar-benar acak. Atau dengan kata lain, acak yang terarah kepada nilai pheromon tertinggi. Dengan adanya kemungkinan bilangan acak diatas 0.98, namun tetap terarah kepada pheromon tertinggi.

Langkah berikutnya setelah mendapatkan solusi dari fungsi membangun solusi adalah menghitung bobot setiap solusi terbentuk. Penghitungan bobot ini berdasarkan aturan, jika elemen data termasuk bobot yang bersangkutan maka bernilai 1 dan sebaliknya 0. Untuk matrik bobot sendiri berukuran *N* x *K*. Dimana *N* adalah sejumlah data dengan *K* adalah sejumlah *cluster*.

$$w_{ij} = \begin{cases} 1, jika\_elemen\_i\_termasuk\_clusterj \\ 0, jika\_sebaliknya \end{cases}$$

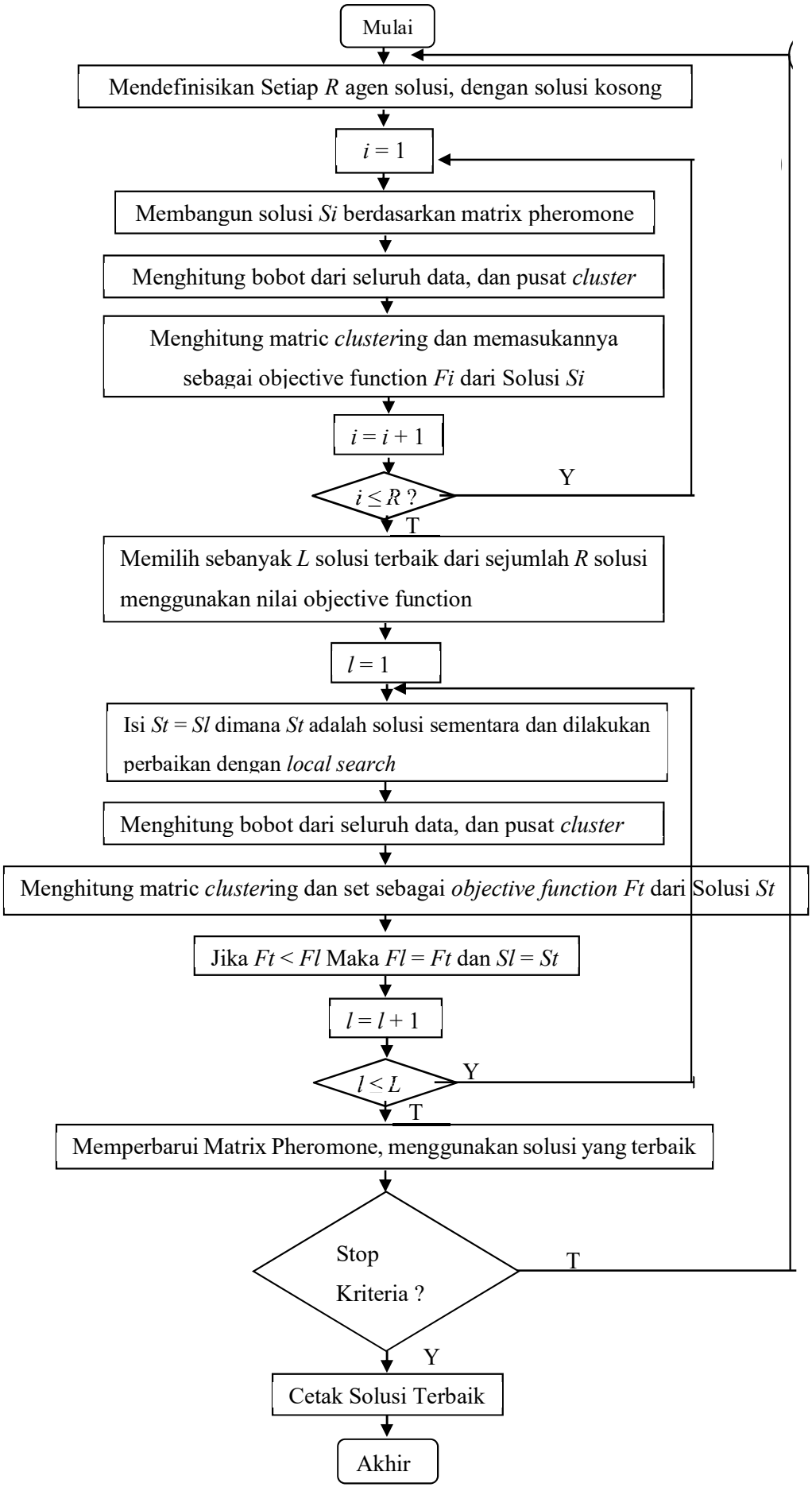
Dimana $i = 1, 2,...,N$ dan $j = 1, 2,...,K$.

Oleh karena itu, berdasarkan contoh solusi yang telah ada, maka dapat dilihat pada tabel 2.3.

Setelah mendapatkan nilai bobot maka langkah selanjutnya adalah menghitung pusat *cluster* berdasarkan tabel 2.2 atau matrik bobot yang telah terbentuk. Tabel 2.2 menunjukkan bahwa setiap elemen data ke i termasuk kedalam *cluster* j dilambangkan dengan 1 dan sebaliknya 0. Perhitungan pusat *cluster* sesuai dengan persamaan (2.8).

$$m_{jv} = \frac{\sum_{i=1}^{N} w_{ij} x_{iv}}{\sum_{i=1}^{N} w_{ij}}, j = 1,...,K\_dan\_v = 1,...,n \qquad (2.8)$$

Untuk setiap elemen data terdapat atribut. Masing-masing atribut akan dikalikan dengan nilai bobot pada masing-masing kolom matrik bobot(w_{11}=1, w_{12}=0, w_{13}=0). Hasil dari perkalian ini dijumlah sepanjang atribut untuk kemudian dibagi dengan jumlah dari bobot pada *cluster* yang sama. Sehingga terbentuklah matrik pusat *cluster* dengan ukuran *K* x *n*. Dimana K adalah jumlah *cluster*, dan n adalah jumlah atribut. Maka nilai dari m_{jv} adalah matrik baris ke j mewakili *cluster* sedangkan v mewakili posisi atribut. Seperti contoh pusat *cluster* pada tabel 2.4.



Gambar 2.4 Algoritma ACO untuk *Clustering*

Tabel 2.3 Contoh Matrik Bobot yang Terbentuk

N\K	1	2	3
1	1	0	0
2	1	0	0
3	0	1	0
4	0	0	1
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0

Tabel 2.4 Contoh Matrik Pusat *Cluster*

K \ n	1	2	3	4
1	4.95	3.20	1.45	0.20
2	4.83	3.40	1.43	0.23
3	5.05	3.55	1.50	0.30

Langkah selanjutnya setelah menemukan nilai matrik pusat *cluster* adalah menghitung fungsi obyektif. Perhitungan nilai ini untuk mengukur optimal tidaknya solusi *cluster* yang telah terbentuk. Dengan menerapkan prinsip jarak yang sudah dikenal yaitu *euclidean distance*, untuk menghitung jarak data dengan pusat *cluster*nya. Sehingga hal ini menunjukkan semakin tinggi nilai fungsi obyektif (*OF*) solusi yang terbentuk semakin terpartisi dengan baik. Namun pada untuk mendapatkan nilai *OF* pada penelitian ini adalah dengan menghitung 1 / *OF*. Sehingga hal ini menjadi berkebalikan yaitu semakin kecil nilai *OF* yang terbentuk, maka solusi semakin optimal. Adapun untuk mendapatkan nilai *OF*, maka dapat dilihat pada persamaan (9).

$$\text{Min } F(w,m) = \sum_{j=1}^{K}\sum_{i=1}^{N}\sum_{v=1}^{n} w_{ij}\left\|x_{iv} - m_{jv}\right\|^2 \qquad (2.9)$$

Persamaan (2.9) merupakan perwujudan kontrol metode *clustering* agar dapat terpisah dengan baik. Bobot dikalikan dengan jarak antara data dengan pusat *cluster*. Dari perhitungan tersebut setiap nilai *OF* yang terbentuk akan dimasukkan

pada solusi yang terbentuk. Jadi setiap solusi yang terbentuk akan memiliki masing-masing nilai *OF*. Sehingga dengan adanya nilai ini, maka dapat dipilih solusi yang terbaik berdasarkan nilai *OF* terendah. Dan mengurutkan matrik solusi berdasarkan nilai *OF* dari terkecil hingga *OF* terbesar.

Setelah mendapatkan nilai *OF* dan mendapatkan solusi dengan nilai *OF* terkecil, maka langkah berikutnya adalah melakukan local search. Tetapi sebelum melakukan local search pada solusi yang terbentuk dilakukan sebuah pemilihan sejumlah solusi. Pemilihan sejumlah solusi ini sebanyak *L*. *L* yang terbentuk berdasarkan penelitian (Shelokar,2004) ditentukan dengan mengalikan sebanyak 20% dari total solusi atau jumlah agen (*R*). Misalnya *R*=10 maka, *L* yang terbentuk adalah sebanyak *L* = 20% x 10 yaitu 2 buah. Dengan begitu, diambil sebanyak 2 solusi pada matrik solusi diperingkat 1 dan 2.

Adapun mengenai *local search* memiliki tujuan untuk mendapatkan solusi yang terbaik dari yang terbaik dengan memperbaiki *OF* dan solusi. Dengan adanya tujuan ini dapat bermanfaat untuk membuat optimal solusi terpilih.

Berikut langkah dalam melakukan local search. Terlebih dahulu menentukan nilai p_{ls}, nilai ini ditentukan pada awal sistem. Nilai yang diambil berkisar antara 0 dan 1. Misalnya, diambil nilai p_{ls}=0.01.

1. Pertama setelah menentukan nilai p_{ls}, maka membuat bilangan acak sejumlah data. Misalnya, (0.693241, 0.791452, 0.986542, 0.998734, 0.234567, 0.967231, 0.001076, 0.567432).
2. Melakukan perbandingan dengan p_{ls}, jika setiap bilangan acak yang terbentuk ≤ p_{ls} maka, dilakukan pemilihan *cluster* secara acak kecuali *cluster* yang telah terpilih. Kemudian menyimpannya atau menggantinya. Misalnya, pada langkah 1 terbaca bahwa pada solusi elemen data ke 7 yang memenuhi syarat. Maka *cluster* pada bagian ke 7 diganti selain 1, misalnya 2.
3. Setelah itu lakukan penghitungan bobot dan juga pusat *cluster* pada solusi tersebut.
4. Melakukan penghitungan *OF* setelah mendapatkan bobot dan pusat *cluster*.
5. Untuk kemudian dibandingkan dengan nilai *OF* pada solusi sebelumnya, jika *OF* yang terbentuk misalnya *Ofl* lebih kecil dengan *Oft*, maka solusi yang baru akan menggantinya dengan solusi hasil dari local search.

6. Ulangi mulai dari langkah 1 sampai selesai.

Langkah berikutnya adalah memilih dari *L* solusi terpilih mengambil *OF* terendah untuk menjadi solusi terbaik. Dengan adanya solusi terbaik ini akan digunakan pada penghitungan pheromon yang baru (2.10).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{l=1}^{L} \Delta\tau_{ij}^{l} \qquad (2.10)$$
$$i = 1,...,N; j = 1,...,K$$

Pheromon yang baru dihitung dengan menggunakan *evaporation rate* dan nilai *OF*. Untuk nilai $\Delta\tau_{ij}^{l}$ sama dengan 1/*OF*. Untuk data yang termasuk kedalam *cluster* tersebut. Sedangkan evaporation rate dapat dihitung dengan 1-ρ. Dengan mendapatkan nilai *OF* dan 1/*OF* untuk memperbaiki pheromon yang terbaru, maka akan mendapatkan sebuah petunjuk untuk semut berikutnya dalam membangun solusi. Karena solusi berikutnya terbentuk dengan mengacu pada solusi sebelumnya ditandai dengan perbaikan jumlah pheromon. Jika iterasi sudah dilaksanakan semua, maka solusi yang terbaik sudah didapatkan dan siap untuk dicetak. Namus hasil dari *clustering* yang terbentuk masih sulit untuk konvergen.

2.2.2. *Ant Colony Optimization* Untuk *Clustering* Dengan *Heuristic Information*

Ant Colony Optimization untuk *Clustering* yang dikembangkan oleh (Shelokar,2004) sulit untuk konvergen dengan komputasi lama menuntut untuk diperbarui dan dikembangkan. Terdapat pengembangan metode yang telah dilakukan antara lain oleh (Bao,2007;Liu,2010; Tiwari,2010). Masing-masing peneliti memiliki pendekatan yang berbeda.

Peneliti yang berusaha menambahkan ACO dengan informasi *heuristic* menjadi lebih *robust* adalah (Bao,2007). Informasi *heuristic* adalah menghitung nilai jarak euclidean antara pusat data dengan data, dilakukan sebelum menemukan solusi. Sehingga untuk memilih solusi berikutnya, menjadikan informasi ini sebagai parameter dalam pemilihannya.

Penelitian berikutnya berusaha menggantikan fungsi jarak euclidean dengan fungsi dunn's index atau jaccard index (Liu,2010). Penggantian ini digunakan untuk mempercepat tingkat konvergensinya. Jaccard index digunakan pada awal

sistem agar sistem memiliki prediksi berapa jumlah *K*. Sehingga tidak memerlukan inisialisasi nilai *K* terlebih dahulu.

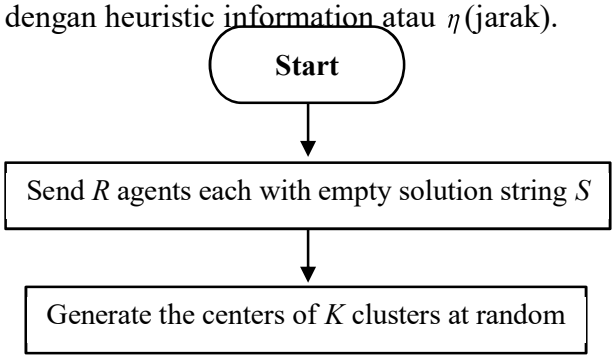
Penelitian (Tiwari,2010) lebih mengedepankan dari kontrol nilai *K*. Berbeda dengan (Liu,2010) yang memberikan sebuah kemungkinan nilai *K* diawal. Penelitian ini mengontrol nilai *K* ketika *K* melebihi 10. Jika nilai jumlah *K* melebihi 10 maka diarahkan untuk mengulang sistem. Selebihnya algoritma masih menggunakan (Shelokar,2004).

Dengan lebih memperhatikan beberapa pembaruan yang ada, maka pada bagian ini lebih memilih penelitian yang menambah informasi heuristic. Karena hal ini kembali kepada sistem ACO pada dasarnya dan algoritma lebih *robust*.

Peneliti mengganti *local search* dengan prinsip genetic algorithm dan pusat *cluster* awal. Lebih tepatnya menggunakan prinsip crossover pada local search untuk mendapatkan kumpulan solusi yang efektif. Penambahan terjadi adalah menambahkan matrik pusat *cluster* secara random.

Menciptakan matrik pusat *cluster* pada awal sistem dilakukan. Untuk melakukannya peneliti membuat suatu bilangan acak sepanjang data. Hal ini dilakukan untuk mendapatkan data secara acak sebagai centroid atau pusat *cluster*. Pembuatan pusat *cluster* yang didapatkan dari penunjukan data secara acak ini mirip dengan langkah-langkah awal K-Means (gambar 2.5).

Perbedaan berikutnya adalah ketika proses membangun solusi. Pada dasarnya aturan membangun solusi adalah sama. Yaitu dalam membangun solusi berdasarkan 2 aturan sama sebelumnya. Yang membedakan adalah proses perkalian antara pheromone dengan heuristic information atau η (jarak).



Gambar 2.5 Potongan algoritma ACO dan Heuristic Bagian Pusat *Cluster*

$$s = \begin{cases} \arg\max_{j \in K} \{\tau_{ij} \cdot [\eta_{ij}]^{\beta}\}, jika\_q \le q0 \\ J, Jika\_lainnya \end{cases} \quad (2.11)$$

Pembangunan solusi ketika *q* yang ada atau bilangan acak yang tercipta ≤ *q0* maka memilih perkalian matrik dengan informasi jarak terbesar. Jika tidak, maka melakukan proses kedua atau *J*.

Sebelum memilih solusi terlebih dahulu membuat matrik yang menyimpan perkalian pheromon dengan informasi jarak. Dimana $\eta=1/d_{ij}$, jarak d_{ij} yang dimaksud adalah jarak Euclidean antara data dengan pusat *cluster*. Oleh karena itu pada iterasi awal, diperlukan pembuatan pusat *cluster* secara acak terlebih dahulu. Sedangkan nilai β didapat dari penelitian sebelumnya yaitu β=2, merupakan parameter yang mempengaruhi jarak.

Sedangkan jika bilangan acak melebihi *q0* maka lakukan aturan 2 yaitu membuat aturan simulasi probabilitas. Perbedaan yang nyata adalah generalisasi pheromon yang digunakan dikalikan dengan η(2.12).

$$p_{ij} = \frac{\tau_{ij}[\eta_{ij}]^{\beta}}{\sum_{k=1}^{K} \tau_{ij}[\eta_{ij}]^{\beta}}, j = 1,...,K \quad (2.12)$$

P_{ij} merupakan nilai hasil probabilitas untuk elemen data ke *i* dengan *cluster j*. Kemudian hasil ini disimpan dalam matrik baru untuk memilih aturan sesuai aturan no.2 pada fungsi membangun solusi.

Selain itu perbedaan juga terjadi pada persamaan memperbarui pheromon. Persamaan (2.10) perbedaan terletak pada penambahan ρ sebelum $\sum_{l=1}^{L} \Delta\tau_{ij}^{l}$. Sehingga persamaan yang digunakan pada penelitian ACO dengan informasi heuristik adalah persamaan (2.13) dan (2.14).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{bs} \quad (2.13)$$
$$i = 1,...,N; j = 1,...,K$$

Dimana,

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/F_{bs}, Jika\_elemen\_i\_masuk\_cluster\_j\_dari\_solusi\_terbaik \\ 0, Jika\_lainnya \end{cases}$$

(2.14)

2.4. PREACO (*Pattern Reduction Enhanced Ant Colony Optimization*)

PREACO adalah metode berdasarkan pada *ant colony optimization* (ACO). Metode ini diperkenalkan oleh (Chun, 2013). Pada dasarnya metode ini bermula menggabungkan *pattern reduction* (PR) kedalam ACO agar dapat berjalan dengan baik dan cepat. Oleh karena itu pada metode ACO diberikan suatu tambahan fungsi. Seperti yang terlihat pada algoritma berikut:

Algoritma PREACO:

1. *Initialization()*
2. *While the termination criterion is not met*
3. *s = SolutionConstruction()*
4. τ *= PheromoneUpdate()*
5. *PatternReduction(s)*
6. *LocalSearch(s)*
7. *End*
8. *Resultl.*

Langkah pada algoritma PREACO pada langkah 1 (satu) sampai dengan langkah 8 (delapan) merupakan algoritma ACO dengan tambahan fungsi PR. Pada langkah 5 merupakan penambahan dari sisi PREACO untuk mengurangi *redundant* (pengulangan proses) dan mencari pheromone yang tepat. Karena redundant dalam pencarian sumber makanan pada oleh semut sering kali diulang, sehingga akan memakan banyak waktu. Dengan adanya tambahan ini maka fungsi iterasi pencarian solusi dapat dimaksimalkan.

Langkah 5 (lima) menjelaskan mengenai fungsi yang didalamnya terdapat berbagai manfaat. *Pattern Reduction* memiliki 3 (tiga) proses yaitu *detection*, *compression*, and *removal*. Ketiga proses ini dilakukan untuk mendeteksi pilihan jalur yang maksimal sedangkan *compression* dan *removal* mencari dan menghilangkan langkah pemilihan jalur yang kurang maksimal agar algoritma dapat lebih optimal dan hemat waktu serta tepat sasaran.

Penjelasan detil mengenai *Pattern Reduction* dapat dilihat pada *pseudocode* berikut yang menjelaskan mengenai *detection operator*. Operator deteksi ini

berfungsi untuk memaksimalkan jalur semut dalam mencari sumber makanan. Ini juga bagian dari *Pattern Reduction*.

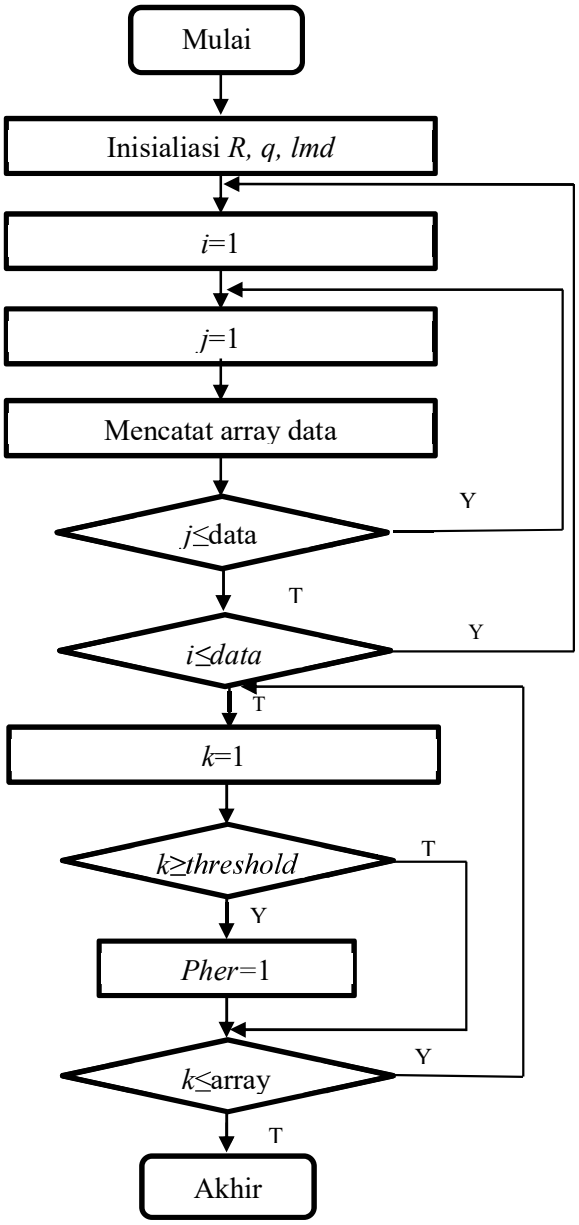
Algoritma *detection operation*:

1. /* *Simpan jumlah berapa kali semut melewati jalur**/
2. For k=1 to Θ /* Θ *adalah jumlah dari individu semut**/
3. For i=1 to n /* *n adalah jumlah dari jalur**/
4. $v_{i,s_i^k} \leftarrow v_{i,s_i^k} + 1$ /*v *adalah sebuah array untuk menyimpan berapa kali jalur terlewati**/
5. Akhir
6. Akhir
7. /***Menentukan tingkat kualitas konvergensi* Ψ *atau waktu jelajah**/
8. *For each edge* e_{ij}
9. *If* $v_{ij} \geq \psi$
10. $\xi_{ij} \leftarrow 1$ /***array menyimpan jalur yang telah dilalui semut dalam waktu dan iterasi tertentu**/
11. Akhir

Bagian kedua dari *Pattern Reduction* ini akan menjamin semut dalam pemilihan semut atau jalur agar tetap pada jalur yang benar. Dalam artian *edge*(jalur) yang telah dipilih oleh semut lain dan telah sampai pada jalur maksimal maka akan diberi prioritas dan agar semut berikutnya dapat mengikuti jalur ini. Dengan begitu jalur yang sudah dilewati tidak perlu dilakukan pengecekan lagi, cukup jalur dengan prioritas tinggi yang dilewati individu semut. Untuk lebih detail flowchart dapat dilihat pada gambar 2.6.

Pada gambar 2.6 menceritakan fungsi *Pattern Reduction* dari algoritma PREACO. Metode ini ditambahkan pada pertengahan proses metode ACO. Pada fungsi ini menghasilkan nilai *pheromone* sama dengan 1 (satu). Nilai hasil dari proses ini akan dijadikan patokan pada pemilihan solusi oleh semut selanjutnya. Hasil dari proses pada gambar 2.6 mempunyai maksud memotong proses redundansi pencarian solusi dan memastikan nilai pheromone agar tetap tinggi. Misal untuk elemen data 1 (satu) diacu pada klaster 2 (dua), dengan nilai pheromon

0.02 maka setelah melalui proses iterasi akan menghasilkan nilai pheromone tertinggi yaitu 1 (satu). Dengan begitu, untuk agen semut berikutnya pada elemen data 1 (satu) otomatis akan menunjuk pada klaster 2 (dua) sesuai dengan solusi agen sebelumnya.



Gambar 2.6 Flowchart Fungsi *Pattern Reduction*

BAB 3
METODE PENELITIAN

Pada penelitian ini, terdapat beberapa tahapan penelitian yang harus dilalui. Pada penelitian ini menggunakan data standar dan metode pembanding untuk melakukan evaluasi. Adapun tahapan penelitian ini adalah sebagai berikut :

3.1. Studi literatur

Pada tahapan ini dilakukan telaah terhadap referensi yang sesuai dengan penelitian. Referensi ini dapat berupa buku, jurnal, maupun narasumber dari artikel atau wawancara. Pencarian metode terbaru dari metode *clustering*. Dari beberapa metode *hard clustering* terdapat metode berdasarkan ACO. Metode ACO yang pada dasarnya untuk melakukan optimasi digunakan sebagai *clustering*. Selain itu penelitian dilakukan pada bagian kernel *clustering*, pada penelitian terbaru terdapat metode MEPSO yang menggunakan *Kernel gaussian* sebagai kernel *clustering*. Pada metode-metode ini masih dimungkinkan dikembangkan untuk mendapatkan performansi maksimal. Oleh karena itu, pada penelitian ini akan mencoba melakukan pengubahan pada bagian kernel dari metode ACO *clustering*.

3.2. Pendefinisian data set

Untuk menguji metode baru dilakukan pengujian dengan data set standar. Data set ini biasa digunakan untuk menguji metode *clustering*. Salah satu data set yang populer didapatkan dari UCI dataset. Data tersebut antara lain iris, wine, glass, dan data sintesis.

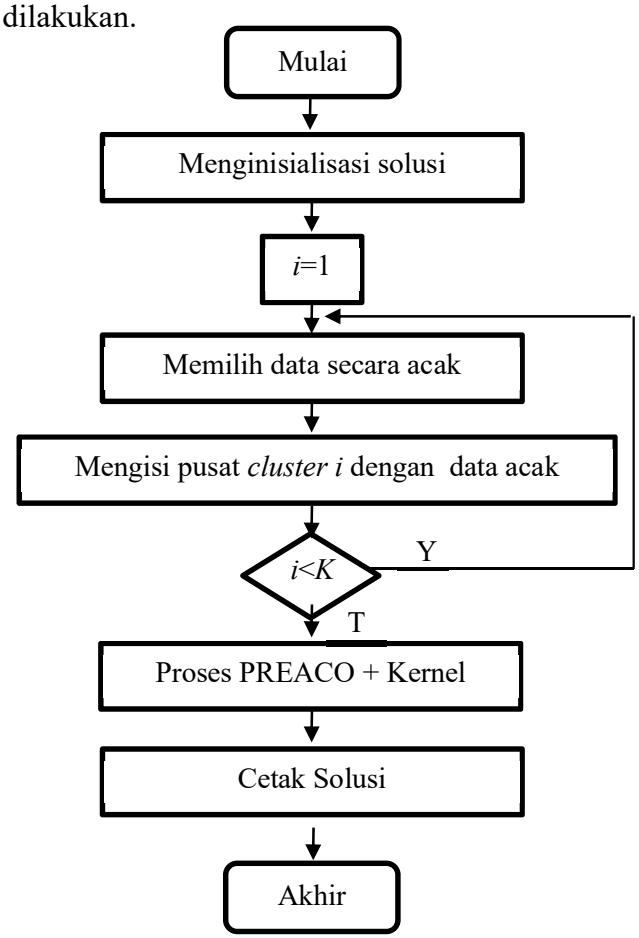
3.3. Desain Sistem

Setelah mendapatkan data set yang jelas, maka dilakukan perhitungan dan percobaan serta modifikasi metode yang sudah ada. Modifikasi dilakukan dengan proses matematis melalui rumusan.

Dari beberapa perbaruan pada metode ACO untuk *clustering*, maka ketertarikan peneliti adalah menggabungkan beberapa pembaruan tersebut. Sehingga terbentuk suatu metode terbaru. Adapun perincian metode dapat dilihat pada gambar 3.1.

Pada penelitian ini menggunakan *kernel Gaussian* sebagai fungsi obyektif. Fungsi ini digabungkan dengan persamaan *CS index* (2.4). Persamaan ini akan menghasilkan suatu nilai yang mengindikasikan bahwa data telah terpartisi dengan baik. Yaitu jarak kedekatan antar data dalam suatu *cluster* yang sama semakin dekat, dengan jarak pemisah antar pusat *cluster* semakin jauh. Sehingga pada fungsi obyektif ini semakin rendah nilai *CS* maka semakin bagus partisi data yang dihasilkan.

Namun fungsi obyektif yang kami gunakan adalah berdasarkan penelitian sebelumnya yaitu 1 / *OF*. Sehingga jika *OF* = *CS*, semakin tinggi nilai *CS*, maka semakin tinggi pula nilai *OF*. Oleh karena, nilai fungsi obyektif dijadikan pembagi, maka aturannya berubah. Yaitu, semkin besar fungsi obyektif, maka semakin baik partisi data yang dilakukan.



Gambar 3.1 Flowchart Desain Sistem Baru

3.3.1 Generate Initial Population

Solusi yang diusung oleh semut adalah terdiri dari kumpulan string. Kumpulan string sepanjang jumlah data tersebut akan berisikan bilangan *cluster* atau kelompoknya. Hal ini berarti, satu string mewakili satu data satu kelompok *cluster*. Contoh dari string solusi dengan jumlah data 8 dengan kelas 3 sebagai berikut : 1, 1, 1, 1, 2, 2, 3, 3. Bentuk ini akan menjadi hasil akhir dari metode.

Agen semut yang digunakan pada ACO untuk *Clustering* tidak hanya satu tetapi puluhan. Dengan adanya puluhan agen semut, maka hal ini menggambarkan sejumlah alternatif kumpulan solusi. Agen semut dimasukkan sebagai parameter awal. Sebagai contoh terdapat R=10, ini berarti jumlah perulangan solusi adalah sebanyak sepuluh kali. Keluaran dengan adanya agen ini adalah sebanyak 10 matrik solusi.

Awal system terdapat sebanyak R solusi, tetapi pada akhirnya hasil atau keluaran dari sistem adalah satu solusi terbaik. Kriteria terbaik terdapat beberapa kontrol. Adapun kontrol solusi adalah, yang memiliki nilai fungsi obyektif tertinggi dan nilai SSE paling kecil. Sehingga dengan adanya kontrol ini, maka solusi akan mengarah semakin kohesi atau sama dan juga terpartisi dengan baik. Kohesi terbukti dengan nilai SSE terkecil dan terpartisi dengan baik ditunjukkan oleh nilai fungsi obyektif tinggi.

3.3.2 Membangun Solusi

Salah satu fungsi dalam desain tersebut adalah *Soluction Construction* atau membangun solusi. Pada fungsi ini melakukan rekonstruksi atau membangun jalur menuju sumber makanan atau dianalogikan mendefinisikan solusi dari suatu permasalahan. Hal ini dilakukan berdasarkan jejak *pheromone* dan informasi yang ada. Masing-masing individu (semut) mencari sumber dari jalur ke jalur sampai semua jalur terlewati. Gambaran ini menunjukkan kemungkinan atau probabilitas solusi. Seperti yang terlihat pada persamaan (2.1). Dimana τ adalah mewakili *pheromone* dan η merupakan informasi heuristik, dan *e* adalah jalur yang dilalui semut. Pada dasarnya informasi heuristik dapat diasumsikan invers dari jarak atau

dalam artian η = 1/d, dimana *d* adalah sebuah rumus jarak *euclidean* dari data terhadap centroid dalam *cluster* yang sama. Langkah yang dilakukan adalah mengirim individu semut dalam melakukan pencarian sumber makanan secara acak, sesuai dengan persamaan (2.11).

3.3.3 UpdatePheromone

Sedangkan fungsi *updatePheromone* diperlukan untuk melakukan pembaruan dari *pheromone* yang ditinggalkan individu/semut. Hal ini diasosiasikan sebagai pengalaman pencarian dari semut. Setiap *pheromone* yang ditinggalkan sebagai jejak akan dipilih oleh individu, semakin banyak pilihan individu maka probabilitas solusi pada jalur itu semakin besar. Hal ini dapat dilihat pada persamaan (2.13) dan (2.14).

$$\Delta \tau_{ij}^{k} = \frac{1}{CS_{\text{kernel}}(k)} \tag{3.1}$$

Sedangkan pada Fungsi kernel digunakan untuk melakukan penyeleksian sub solusi dengan menghitung *fitnes function* dari solusi yang muncul. GaussianKernel digunakan pada rumusan ini berdasarkan penelitian (Das, 2008 dan Kuo, 2014) seperti pada persamaan (2.4). Untuk nilai GaussianKernel didapatkan dari persamaan $K(\vec{x}_i - \vec{x}_j) = \exp\left(-\frac{\vec{x}_i - \vec{x}_j^{\,2}}{2\sigma^2}\right)$.

Perhitungan fungsi obyektif adalah berdasarkan persamaan (3.1). Dimana hasil dari penghitungan CS index dimasukkan pada bagian pembagi. Sehingga semakin kecil nilai CS, maka akan semakin besar nilai fungsi obyektif.

3.3.4 Proses PREACO

PR atau *pattern reduction* merupakan bagian dari metode PREACO. Fungsi PR menyesuaikan dengan algoritma ACO, fungsi ini dilakukan setelah proses *local search* dilakukan. Hal ini terus dilakukan sampai pada kondisi berhenti terjadi apabila waktu dan konvergensinya terpenuhi. Jika jalur dan waktu belum tercapai maka akan diulang prosesnya, selain itu maka akan tercapai kondisi optimal. Untuk

menunjukkan proses algoritma dari metode yang diajukan dapat dilihat pada gambar 3.1.

3.3.5. Kernel Gaussian

Didalam melakukan proses pengelompokan data menggunakan metode ACO membutuhkan fungsi obyektif untuk memastikan validitasnya. Oleh karena penggunaan fungsi obyektif yang tepat menentukan bagus tidaknya metode. Pada penelitian ini menggunakan fungsi obyektif Gaussian. Fungsi obyektif dapat kita sebut sebagai fitness function dengan persamaan (3.1).

Fungsi obyektif Gaussian selanjutnya digunakan sebagai kernel metode pengelompokan data. Gaussian dipilih berdasarkan penelitian sebelumnya memiliki keunggulan lebih *robust* dan lebih respon untuk data *multivariate*. Adapun fungsi obyektif menggunakan *kernel gaussian* menggunakan persamaan (2.3).

Adanya *kernel gaussian* ini solusi yang terbentuk akan diberikan nilai berdasarkan tingkat kebenarannya. Semakin kecil nilainya maka semakin bagus solusi yang terbentuk. Tetapi berdasarkan persamaan (3.1), maka semakin besar nilai *kernel gaussian* maka semakin bagus solusi yang terbentuk. Berikut adalah gambaran langkah dari keseluruhan proses desain sistem.

1. Set parameter yang ada, termasuk jumlah *cluster* dan parameter kernel, tentukan juga nilai *K* dan jumlah semut (*R*), misal *R*=10.
2. Generate individu. Individu tersusun atas nilai threshold dan centroid *cluster*. Secara acak tentukan *centroid cluster*.
3. Hitung nilai *fitness* (f_i) untuk menggambarkan kualitasnya. Berikut penjabarannya:

a) Hitung jarak data dengan semua centroid yang aktif pada solusi menggunakan persamaan jarak Euclidean.

b) *Assign* data pada *cluster* yang sesuai.

c) Gunakan fungsi kernel untuk menghitung CS_{kernel}. Untuk masing-masing semut, hitung fitness-nya.

$$f_i = \frac{1}{CS_{\ker nel_i}(k) + eps'}$$, dimana *eps* adalah nilai konstanta kecil.

d) Tentukan probabilitas untuk *subsolution* berdasarkan persamaan (2.7).

e) Update Centroid dengan persamaan pada persamaan (2.8).

4. Lakukan *Updatepheremone* seperti pada persamaan (2.12) dan (2.13).

5. Lakukan *pseudocode* dari *Patterreduction* di atas dapat menggambarkan prosesnya.

6. Tercapai kondisi konvergen dengan waktu yang singkat, maka terpenuhi jika tidak, maka ulang langkah 2 sampai 5.

7. Kondisi optimal tercapai jika sudah konvergen atau mengalami stagnasi dalam pembagian *cluster* selain itu kondisi optimal dapat dilihat dari jumlah iterasinya yang mencapai maksimal.

3.4. Skenario Uji Coba

Pengujian rumusan yang baru diujicobakan menggunakan tools (alat bantu) Eclipse dan bahasa pemrograman java. Setelah dilakukan pengkodean, maka kode ini akan diujicobakan pada data set yang ada. Selain itu metode sebelumnya juga dilakukan proses yang sama untuk melihat hasilnya. Setelah muncul hasil pada masing-masing metode akan dilakukan proses pembandingan performa.

Untuk skenario uji coba rumusan dilakukan menggunakan data dari UCI dataset. Uji coba dilakukan untuk mendapatkan rasio efektifitas dari rumusan terhadap jumlah *cluster*. Pengujian dilakukan pada masing-masing data set (iris, wine, dan data sintesis). Pada pengujian PREACO (Chun,2013) terdapat parameter yang digunakan terhadap data set. Parameter ini akan digunakan pada uji rumusan ini.

Skenario Uji Coba :

1. Melakukan pencarian dan pengujian kumpulan parameter pada setiap data untuk mendapatkan parameter yang tepat.
2. Melakukan pengujian dengan beberapa nilai *K* untuk mendapatkan nilai SSE terbaik.
3. Melakukan uji akurasi dengan nilai *K* disesuaikan dengan threshold.

Pengujian yang pertama adalah mendapatkan kompoisisi parameter yang tepat dengan mengujicobakan kemungkinan paramater pada beberapa data set. Pengambilan parameter didasarkan pada waktu nilai SSE dan waktu komputasi. Kedua adalah pengujian menggunakan nilai SSE (*Sum of Squared Error*). Metode

atau sistem diuji cobakan pada banyak nilai *K* (jumlah *cluster*). Sehingga pada hasil nilai *K* tersebut akan terlihat, pada *K* keberapa solusi akan optimal dalam mempartisi data. Persamaan SSE yang digunakan dapat dilihat pada persamaan (3.3).

Pengujian dilakukan dengan melakukan koleksi data kemudian dilakukan normalisasi data, pengujian algoritma, evaluasi efisiensi, dan efektifitas pembentukan *cluster*. Untuk mendapatkan hasil performansi yang bagus dari algoritma dilakukan uji akurasi (3.2). Hal ini dilakukan dengan menghitung jumlah *cluster* yang benar dibanding total keseluruhan *cluster* yang terbentuk. Selain itu untuk menguji metode diperlukan evaluasi antar metode.

$$\text{Akurasi} = \frac{Jumlah\_cluster\_yang\_benar}{Total\_jumlah\_cluster} \tag{3.2}$$

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} (x - m_i)^2 \tag{3.3}$$

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan secara detail metode yang telah didesain. Untuk selanjutnya melakukan pengujian terhadap metode yang telah terbentuk. Pengujian dilakukan pada beberapa dataset dan menghitung validasi serta akurasi metode.

4.1 Bahasa Pemrograman

Pada bab ini kami mulai dengan menggambarkan *tools* yang kami gunakan dalam mengimplementasikan desain metode. Penggunaan pemrograman bahasa JAVA menjadi pilihan untuk menerapkan metode ACO untuk *clustering*. Pemilihan ini berdasarkan kemampuan peneliti dalam menguasai bahasa pemrograman JAVA.

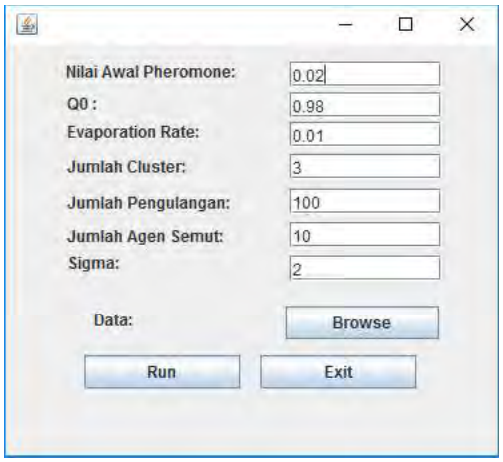
Sedangkan *tools* untuk menerapkan bahasa pemrograman JAVA menggunakan Eclipse. Mengenai penggunaan seri Eclipse bebas, hal ini dikarenakan metode yang akan diterapkan masih sederhana. Pemilihan Eclipse disebabkan peneliti merasa telah berpengalaman menggunakannya. Untuk menerapkan metode ACO *clustering* ini dapat menggunakan *tools* dan bahasa pemrograman apa saja. Hal ini disebabkan metode termasuk dalam kategori struktural, yaitu dapat diterapkan dengan beberapa fungsi dan *flowchart* sederhana.

4.2 Parameter Metode Ant Colony Optimization *Clustering*

Setelah melakukan pengkodean metode, langkah selanjutnya adalah menguji parameter ACO. Hal ini perlu dilakukan mengingat pentingnya peran parameter dalam menentukan keberhasilan ACO dalam melakukan *clustering*. Parameter yang digunakan dapat dilihat pada gambar 4.1.

Seperti terlihat pada gambar 4.1 menunjukkan tampilan awal aplikasi yang berisikan beberapa parameter. Nilai awal pheromone adalah untuk menentukan nilai pheromone sebelum melakukan iterasi pertama. Nilai ini yang akan menjadi rujukan pada proses membangun solusi. Sebagai awal iterasi, nilai ini akan disimpan pada matrik berukuran N x K. Dengan N adalah banyak data dan K jumlah *cluster*. Hal ini diperlukan, karena apabila pada saat membuat matrik pheromone dalam kondisi random atau acak maka, dalam membangun solusi akan lebih variatif

dan sulit konvergen. Jika ditentukan sama terlebih dahulu, maka dalam membangun solusi akan lebih ringan karena metode sebagian sudah konvergen dan sistem hanya mencari sisa *cluster* yang salah.



Gambar 4.1. Parameter ACO

Parameter berikutnya adalah *q0* yang menentukan pada saat membangun solusi. Tingkat nilai adalah dimulai dari 0 sampai dengan 1 atau (0<*q0*<1). Menurut pengalaman peneliti, dan juga peneliti sebelumnya terdapat beragam nilai *q0* yang dapat digunakan. Hal ini disesuaikan dengan beberapa kasus percobaan. Nilai *q0* yang optimal sehingga dapat menghasilkan solusi yang cepat konvergen adalah antara 0.8, 0.85, sampai 0.98. Pada saat peneliti melakukan penelitian untuk nilai 0.1 sampai dengan 0.5, solusi yang dibangun sulit konvergen. Hal ini terjadi karena terlalu banyak penggantian sub solusi. Misalnya, terdapat solusi dengan 4 data berikut (1,1,2,3) pada iterasi berikutnya berubah menjadi (1,2,2,3) kemudian berubah sepanjang iterasi. Hal ini akan menyulitkan sistem untuk cepat konvergen. Maka, peneliti mengambil keputusan untuk menggunakan nilai *q0* dengan nilai 0.98.

Parameter yang berpengaruh berikutnya *evaporation rate*. Nilai ini bukan langsung digunakan, karena untuk mendapatkan nilai *evaporation rate* maka terlebih dahulu harus dikurangkan dengan 1. Pada tampilan program gambar 4.1 bernama *evaporation rate* mewakili (1-ρ). Nilai yang dapat digunakan adalah 0, 0.1, 0.01. Berdasarkan pengalaman uji coba, nilai yang paling sering menghasilkan SSE (*Sum Squared of Error*) kecil dengan fungsi obyektif besar adalah 0.01.

Tidak kalah pentingnya adalah parameter jumlah *cluster*. Nilai ini sebagai wakil dari nilai *K*, seperti pada desain. Nilai *K* menentukan data dibagi menjadi sebanyak *K* kelompok. Nilai ini beragam berdasarkan data uji. Untuk melakukan pengujian nantinya akan mencari nilai SSE dan Akurasi pada beberapa nilai *K*.

Parameter iterasi atau jumlah pengulangan menjadi sangat penting karena akan menentukan tingkat komputasi. Lama atau tidaknya metode ditentukan oleh paramater ini. Semakin banyak iterasi yang diberikan maka metode akan semakin lama. Selain itu, semakin banyak jumlah iterasi cukup menentukan untuk membenarkan apabila terdapat *cluster* yang salah. Walau pada beberapa kasus percobaan, semakin banyak iterasi menghasilkan nilai SSE yang lebih besar.

Penentu beragamnya solusi adalah agen semut yang dikirim. Agen semut disini berperan untuk melakukan iterasi membangun solusi. Pada setiap iterasi, akan memberikan ragam solusi yang dapat sama dengan solusi sebelumnya atau bahkan berbeda dari solusi sebelumnya. Semakin banyak agen yang dikirim akan solusi yang terbentuk akan beragam.

Dampak beragamnya jumlah solusi baik disebabkan karena jumlah iterasi atau jumlah agen semut menyebabkan lama waktu komputasi. Hal ini seperti yang terjadi pada beberapa percobaan ketika dengan agen semut berjumlah 40 dan berjumlah 30, lebih lama agen berjumlah 40. Sedangkan hasil yang dibangun bergantung pada jumlah iterasi dan parameter lain. Bahkan, ketika dicoba menggunakan jumlah iterasi sama dengan jumlah agen semut berbeda, maka hasil akhir solusi yang terbentuk cukup berbeda. Terdapat pula agen jumlah lebih sedikit dengan jumlah iterasi sama, memiliki nilai Fungsi Obyektif lebih tinggi atau lebih baik dan nilai SSE lebih kecil.

Untuk menentukan besar kecilnya kernel diperlukan sigma. Sigma adalah mewakili ragam atau standar deviasi kernel. Pada *kernel gaussian* nilai sigma ini sebagai pembagi dengan nilai sigma dikuadratkan dan dikalikan dengan 2 ($2 \times \sigma^2$). Sigma yang dapat digunakan adalah bernilai 0.9, 1.1, dan 2. Berdasarkan pengalaman uji coba, nilai yang sering menghasilkan solusi baik adalah bernilai 2.

4.3 Pemilihan Data Set

Data set yang digunakan pada penelitian ini adalah data iris, wine, dan T4 (tabel 4.1). Data iris dan wine sering digunakan pada metode yang sama

berdasarkan penelitian sebelumnya. Sedangkan data T4 didapat untuk menguji kernel berdasarkan kedalaman data.

Tabel 4.1 Deskripsi Data Set

Data Set	**Jumlah Data**	**Jumlah *Cluster* (K)**	**Jumlah Atribut**
Iris	150	3	4
Wine	178	3	13
T4	400	5	2

Data Iris pada tabel memiliki gambaran sebagai berikut. Data dengan jumlah sebanyak 150. Memiliki jumlah *K* (*Cluster*)=3, Jumlah atribut=4. Data yang digunakan telah diurutkan dengan data 1 sampai 50 termasuk *cluster* 1, data ke-51 sampai data ke-100 *cluster* 2, dan selebihnya sampai 150 termasuk *cluster* 3. Data diambil dan diunduh dari situs UCI dataset.

Data Wine mengalami normalisasi data. Pada data ini memiliki keragaman yang sangat tinggi antar atribut sehingga diperlukan normalisasi data. Normalisasi data yang digunakan adalah menggunakan normalisasi min max (4.1). Untuk melakukan normalisasi ini dilakukan pada perangkat lunak aplikasi Microsoft Excel. Sedangkan karakter data wine terdapat jumlah *cluster* 1 = 59 buah data, 71 buah data untuk *cluster* 2, dan 48 buah data untuk *cluster* 3. Data telah terurutkan dari *cluster* 1 sampai *cluster* 3. Data wine didapatkan dari mengunduh melalui situs UCI dataset.

$$NilaiBaru = \frac{NilaiAsal - NilaiMinAsal}{NilaiMaxAsal - NilaiMinAsal} \quad (4.1)$$

Sedangkan data T4 adalah dari penelitian sebelumnya. Data ini berupa data sintesis, berjumlah 400. Data telah terurutkan dari *cluster* 1 sampai dengan *cluster* 5. Masing-masing *cluster* memiliki jumlah data sebanyak 80.

4.4 Pencarian Parameter ACO

Percobaan ini mencari nilai yang tepat dari berbagai macam parameter yang ada untuk dijadikan rujukan pengujian. Parameter ACO untuk *clustering* meliputi banyaknya jumlah iterasi, agen, *q0*, *evaporation rate*, *K*, dan sigma. Parameter-

parameter tersebut akan menentukan nilai keluaran sse dan of serta diketahui waktu komputasinya.

Adapun percobaan dilakukan pada satu dataset yang sama pada beberapa metode yang berbeda. Dengan adanya parameter pada berbagai metode ini bertujuan untuk mendapatkan nilai parameter yang tepat pada masing-masing metode dengan hasil terbaik. Kriteria tepat jika nilai SSE kecil dengan waktu yang sedikit. Kumpulan parameter akan diurutkan berdasarkan jumlah SSE terkecil. Hal ini diperlukan karena, nilai SSE adalah suatu metode yang menilai seberapa dekat relasi data dalam suatu *cluster*.

4.4.1 Percoban untuk Pencarian Parameter Data IRIS

Percobaan untuk pencarian parameter terbaik data Iris dilakukan. Percobaan tersebut menggunakan metode ACO, ACO dengan Kernel, dan PREACO dengan Kernel.

Tabel 4.2 Uji Parameter Terbaik Untuk Metode ACO *Clustering*

			Parameter					
Waktu(s)	**sse**	**Of**	**Iterasi**	**Agen**	**q0**	**eva**	**K**	**sig**
57	14.58	92.94	2000	40	0.90	0.10	3	2
24	**15.02**	**96.05**	**800**	**40**	**0.90**	**0.10**	**3**	**2**
56	15.05	95.46	2000	40	0.90	0.10	3	2
28	15.06	97.01	900	40	0.85	0.10	3	2
21	15.11	96.78	700	40	0.75	0.10	3	2
14	25.42	207.05	700	30	0.98	0.01	3	2
21	25.44	198.79	700	40	0.98	0.01	3	2
21	25.70	180.88	700	40	0.8	0.01	3	2
21	25.81	223.33	700	40	0.98	0.01	3	2
9	25.83	212.45	700	20	0.98	0.01	3	2

Tabel 4.2 menggambarkan hasil percobaan pencarian parameter untuk metode ACO pada data IRIS. Pada tabel tersebut telah terurutkan berdasarkan nilai SSE terkecil.

Tabel 4.3 menggambarkan hasil percobaan pencarian parameter untuk metode ACO dan GAUSSIAN pada data IRIS. Metode ini menggantikan fungsi obyektif pada ACO dengan fungsi Gaussian. Pada tabel tersebut telah terurutkan berdasarkan nilai SSE terkecil.

Tabel 4.4 menggambarkan hasil percobaan pencarian parameter untuk metode PREACO dan Gaussian sebagai fungsi obyektifnya pada data IRIS. Pada tabel tersebut telah terurutkan berdasarkan nilai SSE terkecil.

Tabel 4.3 Uji Parameter Terbaik Untuk Metode ACO-Gaussian

Parameter								
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**K**	**sig**
1	**15.10**	**1.19**	**2**	**15**	**0.98**	**0.01**	**3**	**2.0**
1	15.10	1.19	2	15	0.98	0.00	3	2.0
7	15.10	1.19	10	10	0.98	0.10	3	2.0
1	15.10	1.19	2	15	0.98	0.00	3	2.0
1	15.10	1.12	2	15	0.98	0.00	3	1.1
2	15.10	1.19	3	15	0.98	0.01	3	2.0
8	15.12	1.59	10	10	0.98	0.10	3	0.9
0	15.13	1.19	1	10	0.98	0.10	3	2.0
0	15.13	1.19	1	15	0.98	0.10	3	2.0
1	15.13	1.19	2	15	0.98	0.01	3	2.0

Tabel 4.4 Uji Parameter Terbaik Untuk Metode Preaco

Parameter								
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**K**	**Sig**
8	15.06	1.21	8	17	0.98	0.01	3	2
8	15.06	1.20	9	16	0.98	0.01	3	2
8	15.06	1.20	8	17	0.98	0.01	3	2
8	15.06	1.20	8	18	0.98	0.01	3	2
7	**15.06**	**1.15**	**10**	**12**	**0.98**	**0.01**	**3**	**2**
6	15.10	1.19	10	10	0.98	0.00	3	2
7	15.10	1.19	10	12	0.98	0.01	3	2
6	15.57	1.03	10	10	0.98	0.10	3	2
0	16.37	0.94	1	10	0.98	0.01	3	2
0	17.13	0.80	1	10	0.98	0.01	3	2

4.4.2 Percoban untuk Pencarian Parameter Data Wine

Tabel 4.5 menggambarkan hasil percobaan pencarian parameter untuk metode ACO pada data WINE. Pada tabel tersebut telah terurutkan berdasarkan nilai SSE dan waktu terkecil. Berbeda dengan percobaan pada data Iris, pada data Wine memiliki waktu relatif lebih lama. Hal ini terjadi disebabkan jumlah data untuk data Wine lebih banyak dibanding dengan data Iris.

Percobaan untuk pencarian parameter terbaik data Wine dilakukan. Percobaan tersebut menggunakan metode ACO, ACO dengan Kernel, dan PREACO dengan Kernel.

Tabel 4.5 Uji Parameter Terbaik Untuk Metode ACO

Parameter							
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**sig**
131	11.84	87.49	2000	20	0.75	0.10	0.9
196	11.95	87.93	2000	30	0.80	0.10	1.1
131	12.03	88.23	2000	20	0.75	0.10	1.1
131	12.13	88.45	2000	20	0.80	0.10	1.1
133	12.14	88.46	2000	20	0.75	0.10	2.0
116	12.14	90.21	900	40	0.80	0.10	1.1
78	12.33	93.62	600	40	0.80	0.10	1.1
89	12.46	103.41	700	40	0.90	0.10	1.1
96	12.47	94.24	1000	30	0.85	0.10	1.1
64	12.55	92.77	1000	20	0.90	0.10	1.1
67	12.69	96.23	700	30	0.90	0.10	1.1
89	12.85	103.37	700	40	0.90	0.10	2.0
77	12.91	99.99	600	40	0.90	0.10	1.1
44	**13.03**	**102.18**	**700**	**20**	**0.90**	**0.10**	**1.1**
116	13.04	101.15	900	40	0.80	0.10	2.0
116	13.16	105.34	900	40	0.80	0.10	0.9
96	13.16	99.86	1000	30	0.80	0.10	1.1
78	13.22	104.31	600	40	0.85	0.10	1.1

Tabel 4.6 Uji Parameter Terbaik Untuk Metode ACO-GAUSSIAN

Parameter							
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**sig**
7	**12.05**	**0.73**	**2**	**10**	**0.98**	**0.10**	**0.9**
6	12.06	0.66	2	10	0.98	0.00	2.0
52	12.06	0.66	10	15	0.98	0.10	2.0
51	12.12	0.72	10	15	0.85	0.01	0.9
3	12.13	0.66	1	10	0.98	0.10	2.0
11	12.14	0.65	2	15	0.98	0.10	2.0
10	12.14	0.72	2	15	0.85	0.01	0.9
5	12.14	0.65	1	15	0.98	0.10	2.0
12	12.15	0.65	3	10	0.98	0.10	2.0
15	12.17	0.64	3	15	0.98	0.10	2.0
34	12.18	0.71	10	10	0.85	0.01	0.9
7	12.23	0.62	2	10	0.98	0.10	2.0
7	12.41	0.66	2	10	0.98	0.10	1.1
3	12.42	0.64	1	10	0.90	0.00	1.1

Tabel 4.6 menggambarkan hasil percobaan pencarian parameter untuk metode ACO dan GAUSSIAN pada data WINE. Metode ini menggantikan fungsi obyektif pada ACO dengan fungsi Gaussian. Tabel tersebut telah terurutkan berdasarkan nilai SSE dan waktu terkecil.

Tabel 4.7 Uji Parameter Terbaik Untuk Metode Preaco

Parameter							
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**Sig**
38	11.54	0.87	8	10	0.98	0.01	2.0
8	**12.02**	**0.67**	**2**	**12**	**0.98**	**0.10**	**2.0**
56	12.05	0.66	8	18	0.90	0.01	2.0
10	12.06	0.66	2	16	0.98	0.10	2.0
52	12.06	0.66	8	18	0.98	0.10	2.0
58	12.06	0.66	8	18	0.98	0.01	2.0
75	12.06	0.66	10	18	0.98	0.01	2.0
41	12.06	0.66	10	10	0.98	0.01	2.0
38	12.07	0.66	9	10	0.98	0.01	2.0
68	12.07	0.66	9	18	0.98	0.01	2.0
61	12.08	0.66	9	16	0.98	0.01	2.0
56	12.09	0.66	8	16	0.90	0.01	2.0
59	12.11	0.65	8	17	0.90	0.01	2.0
64	12.11	0.65	9	17	0.98	0.01	2.0
72	12.12	0.65	10	17	0.98	0.01	2.0
67	12.12	0.63	10	16	0.98	0.01	2.0
3	12.13	0.66	1	10	0.98	0.10	2.0
5	12.14	0.69	1	17	0.90	0.00	1.1
6	12.15	0.65	1	18	0.98	0.10	2.0
46	12.15	0.65	9	12	0.98	0.01	2.0
35	12.17	0.63	8	10	0.90	0.01	2.0
49	12.17	0.65	10	12	0.98	0.01	2.0

Tabel 4.7 menggambarkan hasil percobaan pencarian parameter untuk metode PREACO dan Gaussian sebagai fungsi obyektifnya pada data WINE. Tabel tersebut telah terurutkan berdasarkan nilai SSE dan waktu terkecil. Sehingga dengan adanya parameter ini, akan digunakan pada proses pengelompokan data. Untuk parameter terpilih pada metode PREACO dengan Fungsi *Kernel gaussian* terdiri dari iterasi=2, agen 12, *q0*=0.87, dan nilai *evaporation rate*=0.10. Parameter yang terbentuk digunakan sebagai patokan uji coba.

4.4.3 Percoban untuk Pencarian Parameter Data T4

Percobaan untuk pencarian parameter terbaik data T4 dilakukan. Percobaan tersebut menggunakan metode ACO, ACO dengan Kernel, dan PREACO dengan Kernel.

Tabel 4.8 Uji Parameter Terbaik Untuk Metode ACO

			Parameter				
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**sig**
90	10.28	91.29	900	40	0.90	0.10	1.1
90	10.45	146.47	900	30	0.98	0.01	1.1
95	1055	94.22	1000	40	0.90	0.10	2.0
74	10.70	142.44	1000	30	0.98	0.01	1.1
89	10.73	143.48	900	40	0.98	0.01	2.0
96	10.73	89.45	1000	40	0.85	0.10	2.0
78	10.76	93.69	900	30	0.85	0.10	2.0
97	11.04	89.97	1000	40	0.80	0.10	2.0
94	11.14	143.49	900	30	0.98	0.01	2.0
119	11.24	98.30	900	40	0.85	0.10	2.0
85	11.34	98.32	1000	30	0.85	0.10	2.0
76	11.36	105.57	1000	30	0.90	0.10	2.0
87	11.37	99.75	900	40	0.90	0.10	2.0
75	11.44	142.4	1000	30	0.98	0.01	2.0
27	**11.52**	**151.8**	**600**	**20**	**0.98**	**0.00**	**2.0**
72	11.52	102.22	900	30	0.90	0.10	1.1
88	11.61	101.25	900	40	0.98	0.10	1.1
97	11.64	92.13	1000	40	0.80	0.10	2.0
45	11.65	100.61	900	20	0.75	0.10	2.0
78	11.66	97.01	1000	30	0.85	0.10	1.1

Tabel 4.8 menggambarkan hasil percobaan pencarian parameter untuk metode ACO pada data T4. Tabel tersebut telah terurutkan berdasarkan nilai SSE. Dan pemilihan parameter berdasarkan pada nilai SSE kecil dan Waktu juga kecil.

Tabel 4.9 menggambarkan hasil percobaan pencarian parameter untuk metode ACO dan GAUSSIAN pada data T4. Metode ini menggantikan fungsi obyektif pada ACO adalah fungsi Gaussian. Tabel tersebut telah terurutkan berdasarkan nilai SSE. Dan pemilihan parameter berdasarkan pada nilai SSE kecil dan Waktu juga kecil.

Tabel 4.9 Uji Parameter Terbaik Untuk Metode ACO-GAUSSIAN

			Parameter				
Waktu(s)	**sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**Sig**
19	4.64	1.66	3	15	0.98	0.00	1.1
12	4.72	1.52	3	10	0.98	0.10	1.1
45	4.74	1.47	10	10	0.98	0.10	2.0
44	4.75	1.47	10	10	0.98	0.10	1.1
21	4.80	1.37	3	15	0.98	0.00	2.0
13	4.83	1.41	2	15	0.98	0.10	1.1
14	4.83	1.39	3	10	0.98	0.01	2.0
6	4.85	1.51	1	15	0.98	0.10	2.0
20	4.85	1.28	3	15	0.98	0.01	2.0
9	4.88	1.25	2	10	0.98	0.01	2.0
9	4.93	1.23	2	10	0.98	0.00	2.0
20	4.93	1.37	3	15	0.98	0.10	1.1
20	4.93	1.26	3	15	0.90	0.01	1.1
20	4.96	1.16	3	15	0.90	0.10	1.1
10	4.97	1.21	2	10	0.98	0.10	2.0
13	4.97	1.18	3	10	0.98	0.00	1.1
20	4.97	1.23	3	15	0.90	0.01	2.0
20	4.98	1.18	3	15	0.90	0.00	1.1
13	5.01	1.09	2	15	0.98	0.10	2.0
20	5.03	1.08	3	15	0.98	0.01	1.1
48	5.03	1.17	10	10	0.98	0.00	2.0
20	5.05	1.33	3	15	0.90	0.10	2.0
49	5.05	1.34	10	10	0.98	0.00	1.1
6	5.07	1.29	1	15	0.98	0.01	2.0
6	5.08	1.34	1	15	0.98	0.10	2.0
8	5.12	1.11	2	10	0.85	0.10	1.1
12	5.12	1.29	3	10	0.98	0.10	2.0
13	5.15	1.17	3	10	0.90	0.10	1.1
4	**5.16**	**1.07**	**1**	**10**	**0.98**	**0.10**	**2.0**
14	5.17	1.08	3	10	0.90	0.10	2.0
4	5.21	0.94	1	10	0.98	0.01	2.0
9	5.22	0.92	2	10	0.98	0.10	1.1

Tabel 4.10 menggambarkan hasil percobaan pencarian parameter untuk metode PREACO dan Gaussian sebagai fungsi obyektifnya pada data T4. Tabel tersebut telah terurutkan berdasarkan waktu terkecil. Dan pemilihan parameter berdasarkan pada nilai SSE kecil dan Waktu juga kecil.

Tabel 4.10 Uji Parameter Terbaik Untuk Metode Preaco

Parameter							
Waktu(s)	**Sse**	**of**	**Iterasi**	**Agen**	**q0**	**Eva**	**Sig**
5	**4.78**	**1.56**	**1**	**10**	**0.98**	**0.10**	**2.0**
5	5.59	0.95	1	10	0.85	0.01	2.0
5	5.64	0.89	1	10	0.85	0.10	2.0
5	5.72	0.88	1	10	0.85	0.10	1.1
5	5.96	0.81	1	10	0.90	0.10	2.0
5	6.14	0.81	1	10	0.85	0.00	1.1
5	6.62	0.81	1	10	0.85	0.00	0.9
5	9.25	0.49	1	10	0.85	0.10	0.9
5	9.29	0.41	1	10	0.85	0.00	2.0
16	4.84	1.34	3	12	0.98	0.10	2.0
16	4.97	1.21	3	12	0.98	0.01	2.0
17	7.08	0.69	3	12	0.98	0.10	1.1
21	4.57	1.93	3	16	0.98	0.01	2.0
22	4.98	1.61	3	17	0.98	0.10	2.0
22	5.27	1.07	3	16	0.98	0.01	1.1
23	5.09	1.50	3	16	0.98	0.10	2.0
23	5.39	1.01	3	17	0.98	0.01	2.0
24	4.87	1.38	3	18	0.98	0.10	2.0
24	4.99	1.13	3	18	0.98	0.01	2.0
24	5.14	1.25	3	17	0.98	1.00	1.1

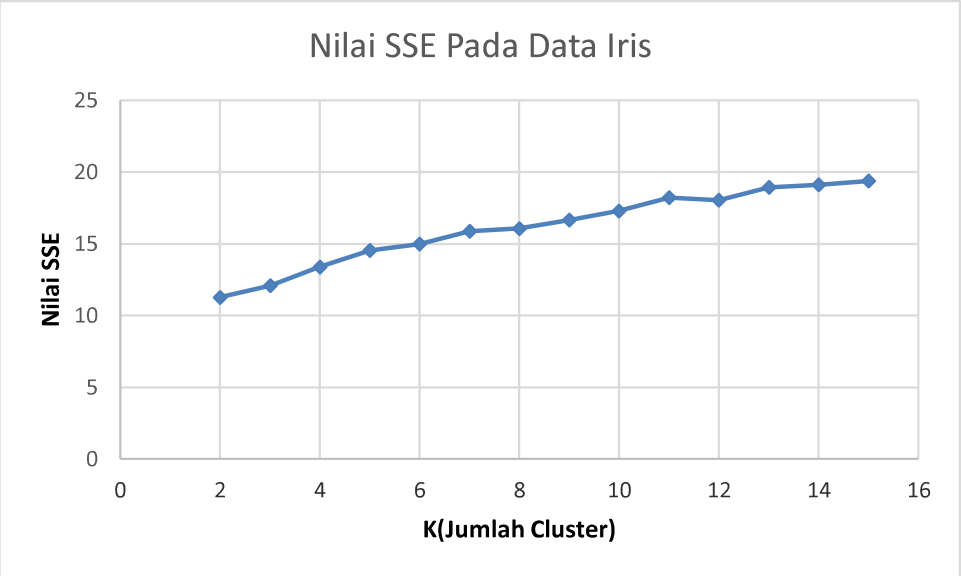
4.5 Pengujian Menggunakan Sum of Squared Error

Data yang telah terpetakan dan tersusun sesuai *cluster*nya akan diuji cobakan pada metode. Hasil keluaran dari aplikasi ini adalah berupa kumpulan string solusi sepanjang data. Matrik solusi yang terbentuk memiliki satu baris dengan jumlah kolom sepanjang jumlah data. Untuk kemudian solusi ini dihitung menggunakan SSE (*Sum of Squared Error*).

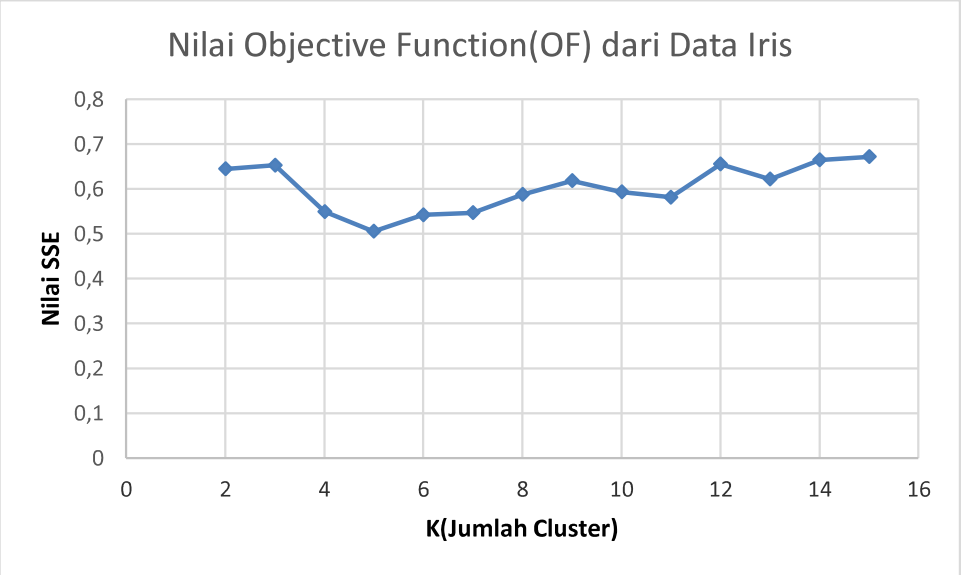
Skenario pengujian dilakukan pada semua jenis data dengan masing-masing diuji beragam nilai *K*. Nilai *K* yang diuji minimal berjumlah 2 sampai nilai *K* lebih dari 10. Untuk masing-masing nilai *K* akan didapatkan nilai fungsi obyektif (*OF*) dan SSE-nya. Sehingga dengan begitu dapat terlihat dengan jelas, kombinasi SSE dan *OF* yang terbaik yang akan diambil sebagai solusi.

Adapun kombinasi yang solusi terbaik dilihat dari nilai SSE dan *OF*. Berdasarkan penelitian sebelumnya pada *Kernel gaussian*, semakin besar nilai *OF*

maka solusi akan semakin optimal. Sedangkan semakin kecil nilai SSE maka solusi terbentuk semakin benar. Berikut gambaran solusi yang telah didapat.



Gambar 4.2 Nilai SSE data Iris

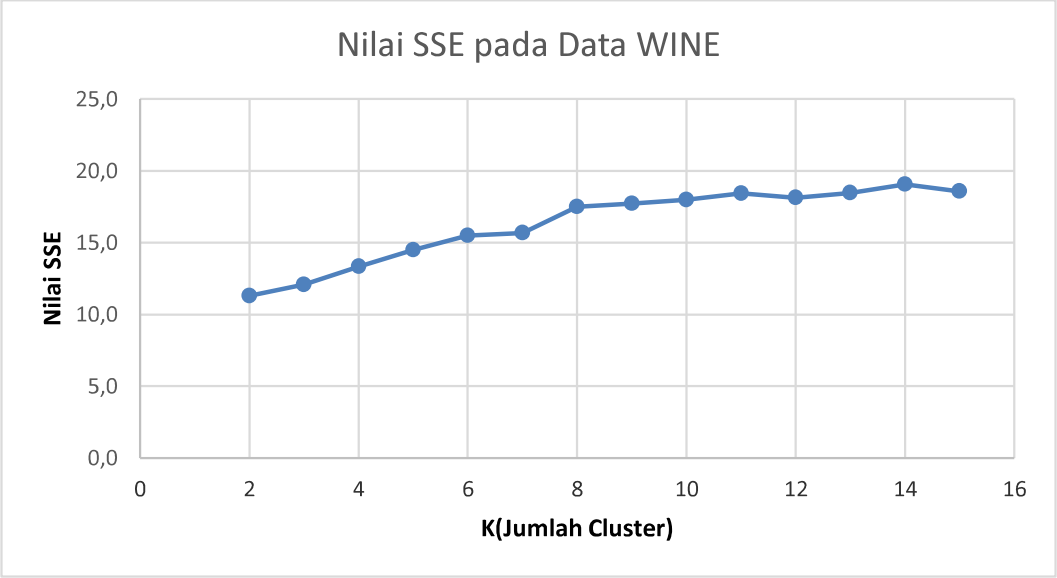


Gambar 4.3 Nilai Fungsi Obyektif Data Iris

Pada gambar 4.2 dan gambar 4.3 menggambarkan pengujian metode pada data iris. Pada sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan sumbu *Y* atau vertikal untuk nilai SSE. Pada nilai SSE data iris paling rendah untuk nilai *K*=5, dan kedua yaitu *K*=3. Sedangkan nilai error atau SSE tertinggi adalah pada saat pengujian dengan nilai *K*=2.

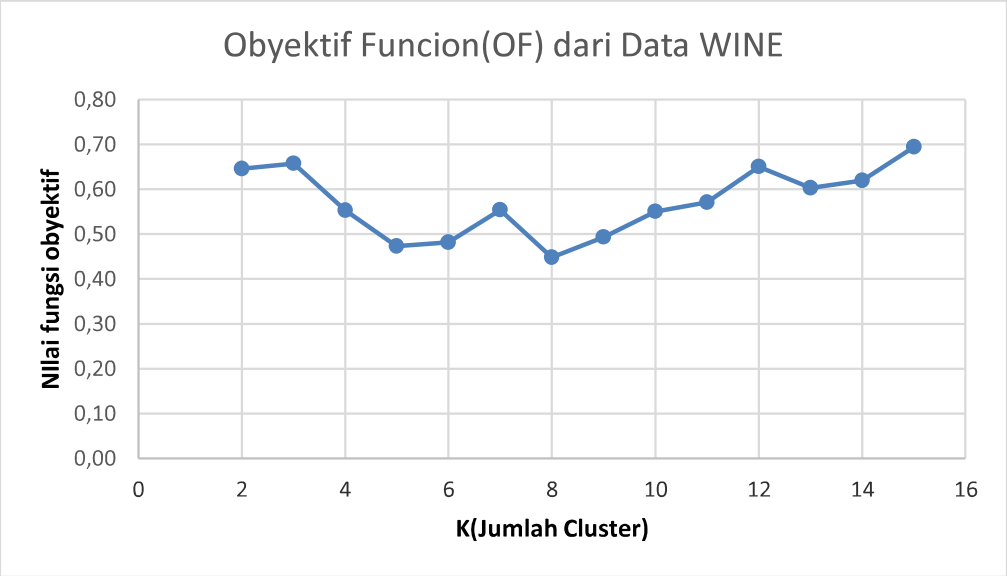
Sedangkan pada gambar 4.3 penghitungan fungsi obyektif (OF) pada data iris. Dengan sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan sumbu *Y* atau vertikal untuk nilai *OF*. Terlihat pengujian pada *K*=2 adalah memilki nilai *OF* tertinggi diikuti *K*=3 terus menurun sampai terendah pada *K*=11.

Pada metode ini menggunakan gabungan nilai SSE dan *OF* untuk mendapatkan hasil optimal. Nilai yang digunakan adalah nilai SSE dengan jumlah kesalahan paling sedikit. Dan nilai *OF* paling besar akan diambil untuk solusi optimal. Oleh karena itu nilai SSE paling sedikit adalah pada saat *K*=5 sedangkan nilai *OF* pada saat *K*=5 rendah. Kemudian untuk nilai *OF* paling tinggi adalah pada saat nilai *K*=2 sedangkan nilai SSE pada saat *K*=2 tertinggi. Hal ini juga tidak dapat diterima oleh, karena itu solusi yang optimal terjadi pada saat ***K*=3** dengan nilai SSE terendah kedua setelah *K*=5 dengan nilai *OF* tertinggi kedua setelah *K*=2.



Gambar 4.4 Nilai SSE data Wine

Pada gambar 4.4 dan gambar 4.5 menggambarkan pengujian metode pada data wine. Pada sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan sumbu *Y* atau vertikal untuk nilai SSE. Pada nilai SSE data wine paling rendah untuk nilai *K*=2, dan kedua yaitu *K*=3. Sedangkan nilai error atau SSE tertinggi adalah pada saat pengujian dengan nilai *K*=14.



Gambar 4.5 Nilai OF Data Wine

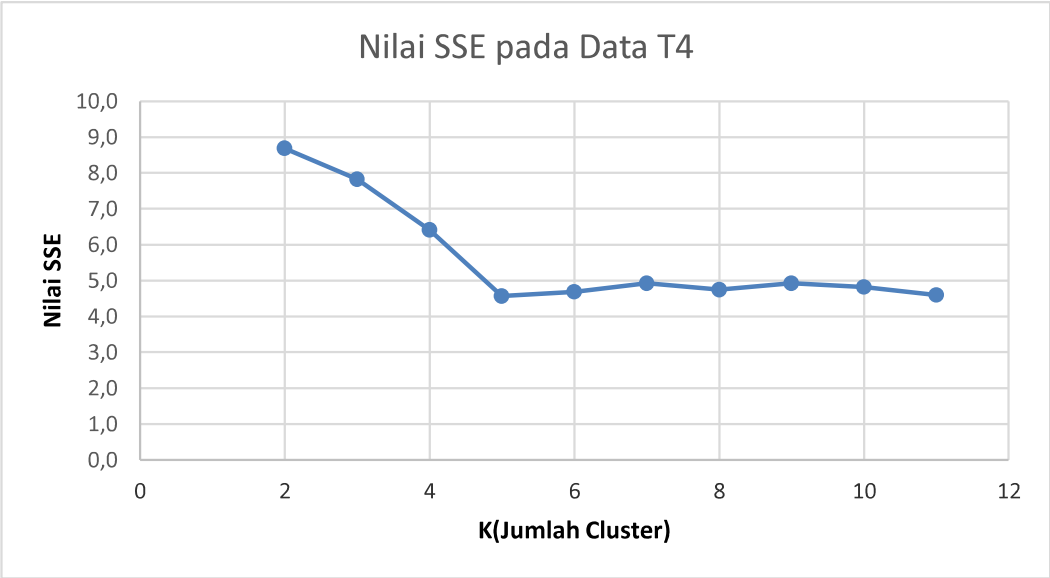
Sedangkan pada gambar 4.5 penghitungan Fungsi Obyektif (OF) pada data wine. Dengan sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan sumbu *Y* atau vertikal untuk nilai *OF*. Terlihat pengujian pada *K*=15 adalah memiliki nilai *OF* tertinggi diikuti *K*=3 terus menurun sampai terendah pada *K*=8.

Pada metode ini menggunakan gabungan nilai SSE dan OF untuk mendapatkan hasil optimal. Nilai yang digunakan adalah nilai SSE dengan jumlah kesalahan paling sedikit. Dan nilai OF paling besar akan diambil untuk solusi optimal. Oleh karena itu nilai SSE paling sedikit adalah pada saat *K*=2 sedangkan nilai OF pada saat *K*=2 rendah. Kemudian untuk nilai OF paling tinggi adalah pada saat nilai *K*=15 sedangkan nilai SSE pada saat *K*=15 tinggi. Hal ini juga tidak dapat diterima, oleh karena itu solusi yang optimal terjadi pada saat ***K*=3** dengan nilai SSE terendah kedua setelah *K*=2 dengan nilai OF tertinggi kedua setelah *K*=15.

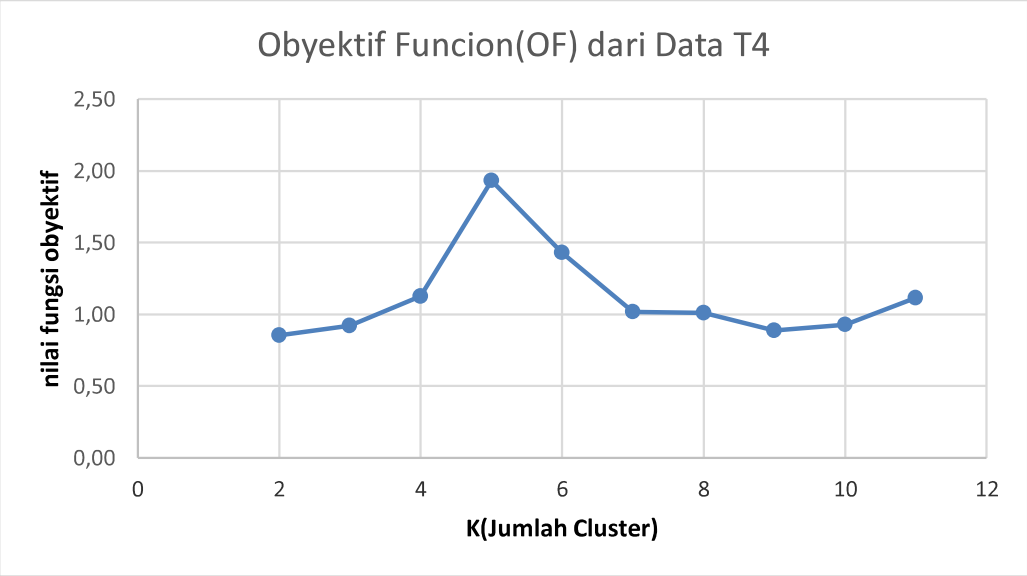
Pada gambar 4.6 dan gambar 4.7 menggambarkan pengujian metode pada data T4. Pada sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan sumbu *Y* atau vertikal untuk nilai SSE. Pada nilai SSE data T4 paling rendah untuk nilai *K*=5. Sedangkan nilai error atau SSE tertinggi adalah pada saat pengujian dengan nilai *K*=2.

Sedangkan pada gambar 4.7 penghitungan OF (*Objective Function*) pada data T4. Dengan sumbu *X* atau horisontal untuk nilai *K* (jumlah *cluster*), sedangkan

sumbu *Y* atau vertikal untuk nilai OF. Terlihat pengujian pada *K*=5 adalah memiliki nilai OF tertinggi terus menurun sampai terendah pada *K*=2.



Gambar 4.6 Nilai SSE Data T4



Gambar 4.7 Nilai OF Data T4

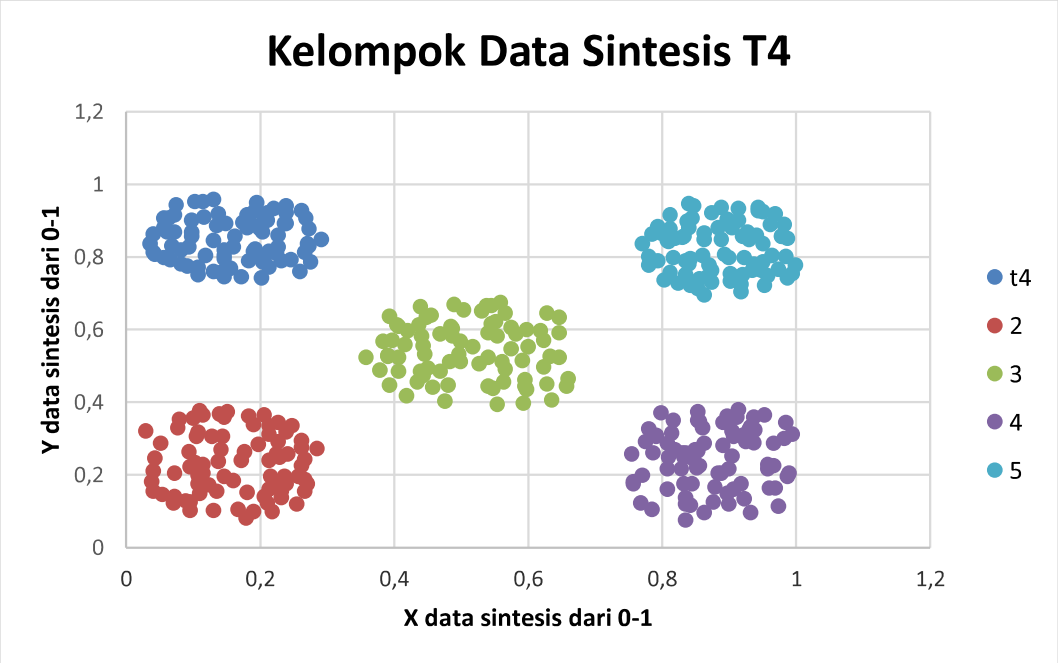
Pada metode ini menggunakan gabungan nilai SSE dan OF untuk mendapatkan hasil optimal. Nilai yang digunakan adalah nilai SSE dengan jumlah kesalahan paling sedikit. Dan nilai OF paling besar akan diambil untuk solusi optimal. Oleh karena itu nilai SSE paling sedikit adalah pada saat *K*=5. Kemudian

untuk nilai OF paling tinggi adalah pada saat nilai *K*=5. Oleh karena itu solusi yang optimal terjadi pada saat ***K*=5** dengan nilai SSE terendah dengan nilai OF tertinggi.

4.6 Pengujian Menggunakan Akurasi

Seperti pada penjelasan bab 3 tentang pengujian, salah satunya adalah akurasi. Untuk menguji akurasi ini dihitung setelah mendapatkan nilai SSE dan nilai OF pada pengujian SSE. Oleh karena itu prosesnya berkesinambungan. Untuk mengetahui kondisi benar tidaknya solusi yang terbentuk pada SSE terendah dengan OF tertinggi.

Pengujian dilakukan pada masing-masing dataset setelah menghitung nilai SSE. Berikut deskripsi pada masing-masing data. Pada penghitungan akurasi terdapat dua parameter hitung yaitu jumlah data yang tepat terklasifikasi dan jumlah keseluruhan data.



Gambar 4.8 Visualisasi Pengelompokan Data Sintesis T4 Asal

Gambar 4.8 adalah visualisasi data sistesis T4 yang menjadi sumber acuan. Data terdiri dari dua (2) variabel atau field yaitu sumbu x dan sumbu y. Masing-masing nilai paling rendah 0 dengan nilai maksimal 1 dengan persebaran yang merata dan terbentuk menjadi 5 kelompok data. Untuk selanjutnya data ini akan digunakan sebagai uji coba metode. Dengan adanya visualisasi ini, diharapkan akan

dapat melihat perbedaan hasil yang diciptakan oleh metode dibandingkan dengan data asal.

Pada tabel 4.11 menggambarkan mengenai hasil uji coba metode untuk mengelompokkan data. Terdapat 4 metode yang ditunjukkan pada tabel tersebut yaitu metode ACO, ACO+*Kernel gaussian*, PREACO, PREACO+*Kernel gaussian*. Masing-masing hasil akurasi diujicobakan pada data set iris, wine dan T4.

Tabel 4.11 Akurasi Algoritma

DATA SET	Algoritma			
	ACO	ACO+GaussianKernel	PREACO	PREACO+Gaussian
Iris	89.30%	90.70%	90%	88.70%
Wine	89.90%	93.30%	85%	93.80%
T4	70.80%	99.30%	69%	99.80%

4.6.1 Pengujian Akurasi Metode ACO

Metode ACO melakukan pengelompokan data terhadap data iris. Hasil dapat dilihat pada tabel 4.12. Tabel tersebut berisikan data yang terkelompokkan pada klaster yang sesuai maupun salah diikuti waktu yang dibutuhkan untuk mengelompokkan data. Kolom 'Kelas Sebenarnya' berisikan label klaster, 'Hasil *Cluster*' kolom yang berisikan label klaster yang sesuai maupun yang salam, dan kolom 'Waktu' menggambarkan total waktu dalam detik (s) yang dibutuhkan untuk mengelompokkan data tersebut.

Pada tabel 4.12 menceritakan bahwa metode ini mampu mengelompokkan kedalam klaster 1 sebanyak 49 buah data dengan salah mengelompokkan kedalam klaster 3 sebanyak 1 buah data. Pada klaster 2 memiliki tingkat kesalah klaster pada klaster 3 sebanyak 2 buah data. Dan klaster 3 mengalami kesalahan mengelompokkan pada klaster1 sebanyak 11 buah dan 2 buah data salah masuk kelompok klaster 2. Dari segi waktu komputasi, metode ini memiliki waktu komputasi sebesar 73 detik (s).

Sedangkan pada tabel 4.13 menggambarkan hasil pengelompokan oleh metode ACO pada data Wine. Dengan kesalahan klaster 1 sebanyak 12 buah data. Tingkat kesalahan klaster pada klaster 2 sebanyak 4 buah data. Sedangkan pada klaster 3 hanya memiliki 1 kesalahan.

Senada dengan tabel 4.12 dan 4.13 menggambarkan tentang Matrik *Confusion clustering* untuk data iris dan wine menggunakan metode ACO. Pada tabel 4.14 menunjukkan hasil pengelompokan data untuk data sintesis. Metode ACO mengelompokkan data dengan beberapa kesalahan. Kesalahan tersebut yaitu, pada kelompok 1 masih terdapat kesalahan pengelompokan kepada kelompok 2 sebanyak 10 buah. Sedangkan kesalahan terbanyak terjadi pada kelompok data 3 untuk kelas 3 terjadi kesalahan sebanyak 57 dengan data benar hanya 13 buah. Selain itu waktu yang dibutuhkan untuk proses selama 204 detik. Sedangkan pada gambar 4.9 menggambarkan visualisasi hasil pengelompokan dengan metode ACO.

Tabel 4.12 Matrik *Confusion Clustering* Untuk Data Iris Metode ACO

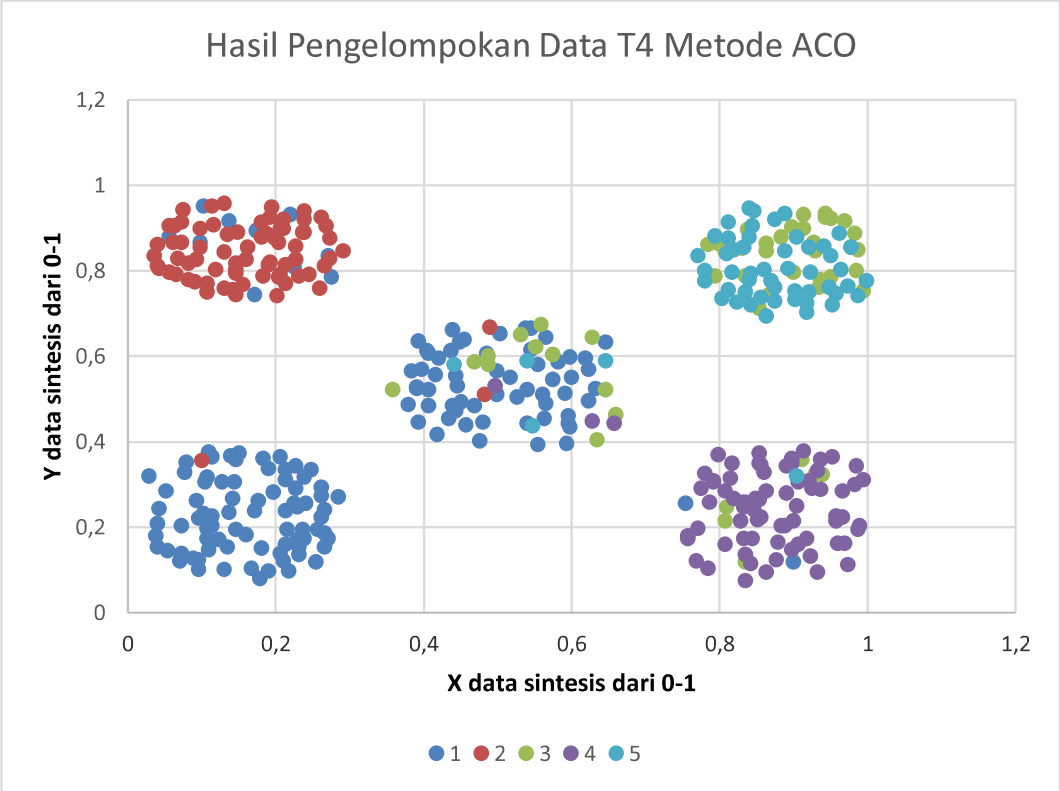
Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**49**	0	1	73
2	0	**48**	2	
3	11	2	**37**	

Tabel 4.13 Matrik *Confusion Clustering* Untuk Data Wine Metode ACO

Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**46**	12	0	272
2	2	**67**	2	
3	1	0	**48**	

Tabel 4.14 Matrik *Confusion Clustering* Untuk Data T4 Metode ACO

Kelas	**Hasil *Cluster***					**Waktu**
Sebenarnya	**1**	**2**	**3**	**4**	**5**	**(s)**
1	**70**	10	0	0	0	204
2	1	**79**	0	0	0	
3	2	58	**13**	3	4	
4	0	2	6	**71**	1	
5	0	0	30	0	**50**	



Gambar 4.9 Visualisasi Pengelompokan Data T4 Metode ACO

4.6.2 Pengujian Akurasi Metode ACO Dengan Fungsi Kernel Gaussian

Uji coba pengelompokan data menggunakan metode ACO dengan Fungsi *Kernel gaussian* menghasilkan Matrik *confusion clustering*. Seperti terlihat pada tabel 4.15 untuk metode ini menghasilkan kebenaran sempurna dengan berhasil memetakkan 50 data dengan benar pada kelas 1. Sedangkan pada kelas 2 berhasil memetakkan 48 data dengan kesalahan 2 buah data termasuk kelas 3. Sedangkan pada kelas 3 menghasilkan kesalahan lebih banyak, yaitu 14 data salah memetakkannya pada kelas 2 yang seharusnya termasuk kedalam kelas 3. Waktu komputasi metode memerlukan 11 detik untuk mengelompokkan data.

Tabel 4.15 Matrik *Confusion Clustering* Untuk Data Iris Metode ACO+Kernel

Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**50**	0	0	11
2	0	**48**	2	
3	0	14	**36**	

Hasil dari percobaan pengelompokan data metode ACO dengan Kernel pada data Wine tersaji pada tabel 4.16. Pada keadaan tersebut menyatakan bahwa metode untuk data kelas 1 dapat terkelompok dengan baik. Kondisi terparah adalah pada pengelompokan data pada kelas 2 dengan kesalahan sebanyak 11 data salah kelompok. Sedangkan pada kelas 3 hanya terjadi 1 buah data salah kelompok. Metode ini membutuhkan waktu 63 detik untuk melakukan pengelompokan data.

Tabel 4.16 Matrik *Confusion Clustering* Untuk Data Wine Metode ACO+Kernel

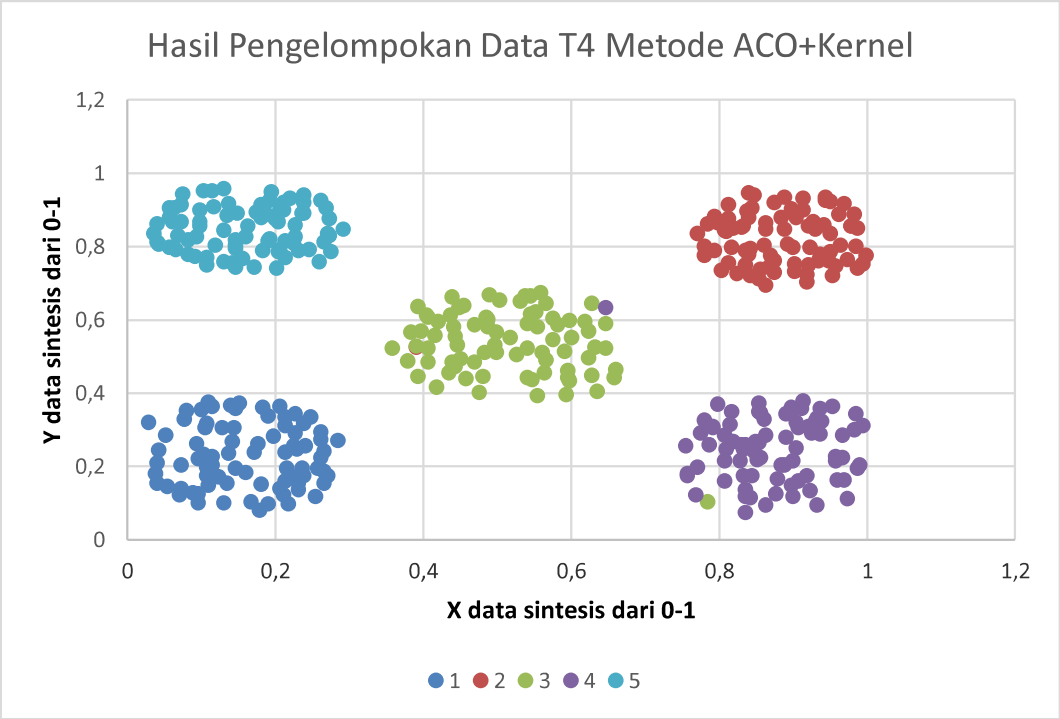
Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**58**	0	0	63
2	7	**60**	4	
3	0	1	**48**	

Pada tabel 4.17 menggambarkan hasil pengelompokan data untuk metode ACO dengan Kernel pada data sintesis T4. Pada data T4 memiliki 5 kelas kelompok data. Data pada kelas 1,2 dan 5 masing data dapat terkelompokkan dengan baik. Sedangkan data untuk kelas 3 dan 4 masih terdapat data salah terkelompok. Dengan waktu proses membutuhkan 64 detik.

Adapun gambaran hasil dari pengelompokan data T4 untuk metode ACO dengan Kernel dapat dilihat pada gambar 4.10. Gambar tersebut menampilkan 5 warna berbeda untuk membedakan jenis kelas kelompok data. Sehingga dengan adanya visualisasi ini dapat mengetahui kesalahan kelompok data.

Tabel 4.17 Matrik *Confusion Clustering* Untuk Data T4 Metode ACO+Kernel

Kelas	**Hasil *Cluster***					**Waktu**
Sebenarnya	**1**	**2**	**3**	**4**	**5**	**(s)**
1	**80**	0	0	0	0	64
2	0	**80**	0	0	0	
3	0	0	**78**	1	1	
4	0	0	1	**79**	0	
5	0	0	0	0	**80**	



Gambar 4.10 Visualisasi Pengelompokan Data T4 Metode ACO+Kernel

4.6.3 Pengujian Akurasi Metode PREACO dengan Fungsi Kernel Gaussian

Tabel 4.18 adalah matrik *confusion clustering* untuk data Iris menggunakan metode PREACO. Hasil dari metode ini berhasil memetakkan data dengan tepat pada kelas 1. Tetapi mengalami kesalahan pengelompokan data pada kelas 2 sebanyak 3 buah dan kelas 3 sebanyak 14 buah kesalahan. Waktu komputasi yang dibutuhkan adalah 9 detik.

Lain pada tabel 4.19 menggambarkan hasil pengelompokan data untuk data Wine menggunakan metode PREACO. Hasil pengelompokan data dengan 10 kesalahan pada kelas 2 dan 1 kesalahan pengelompokan data pada kelas 3. Sedangkan pada tabel 4.20 hasil pengelompokan data untuk data T4 dengan hanya menghasilkan 1 kesalahan untuk kelas 1 dengan waktu 74 detik dengan visualisasi hasil pengelompokan data pada gambar 4.11.

Tabel 4.18 Matrik *Confusion Clustering* Untuk Data Iris Metode PREACO

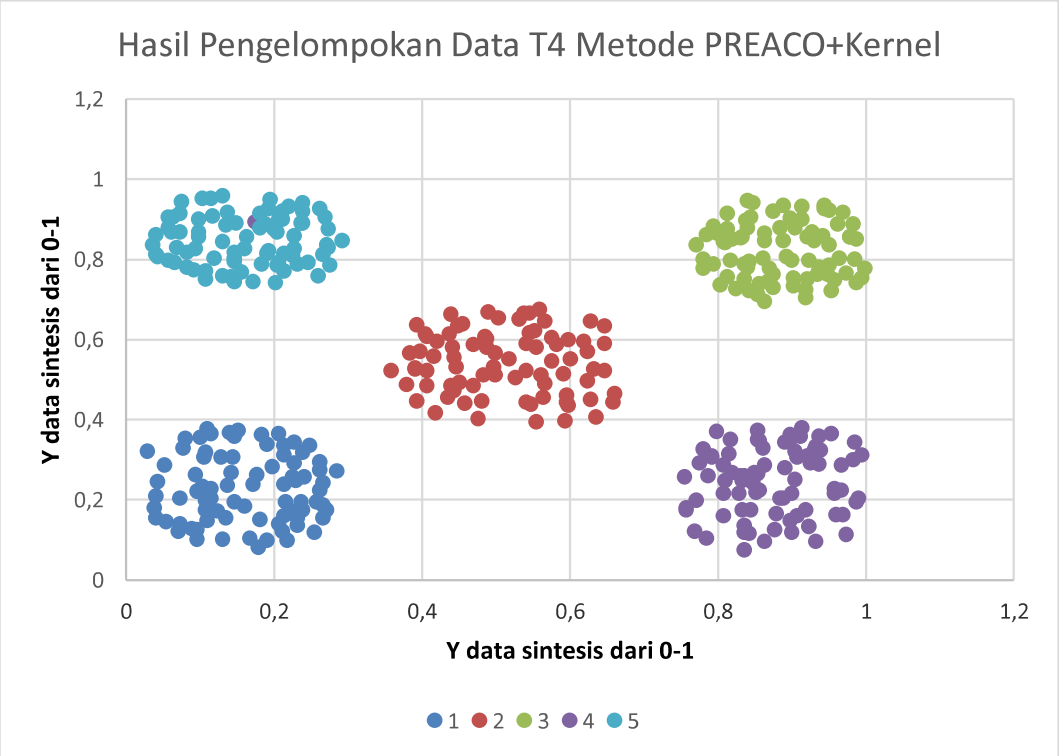
Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**50**	0	0	9
2	0	**47**	3	
3	0	14	**36**	

Tabel 4.19 Matrik *Confusion Clustering* Untuk Data Wine Metode PREACO

Kelas	**Hasil *Cluster***			**Waktu**
Sebenarnya	**1**	**2**	**3**	**(s)**
1	**58**	0	0	**44**
2	3	**61**	7	
3	0	1	**48**	

Tabel 4.20 Matrik *Confusion Clustering* Untuk Data T4 Metode PREACO

Kelas	**Hasil *Cluster***					**Waktu**
Sebenarnya	**1**	**2**	**3**	**4**	**5**	**(s)**
1	**79**	0	0	1	0	**74**
2	0	**80**	0	0	0	
3	0	0	**80**	0	0	
4	0	0	0	**80**	0	
5	0	0	0	0	**80**	



Gambar 4.11 Visualisasi Pengelompokan Data T4 Metode PREACO+Kernel

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut :

1. Berdasarkan hasil uji coba metode PREACO dengan *Kernel gaussian* dapat mengelompokkan dengan baik pada tingkat akurasi pada data wine= 93.8%, dan data T4=99.8% dan memiliki nilai SSE rendah. Hasil akurasi ini lebih baik dibanding dengan metode ACO yaitu untuk data wine=89.9%, dan data T4=70.8%. Namun memiliki kelemahan pada akurasi untuk data iris= 88.7% metode PREACO sedangkan metode ACO memiliki akurasi pada data iris=89.3%. Hal ini terjadi dapat disebabkan karena pada data cluster 2 dan 3 memiliki kedekatan yang beragam.
2. Metode ACO untuk *clustering* (Pengelompokan data) memiliki kecepatan dari sisi konvergen, terlihat pada percobaan memiliki iterasi dan waktu lebih sedikit.
3. Metode PREACO dengan Kernel lebih bagus dari ACO baik dari waktu komputasi ataupun akurasinya.
4. Metode ACO dengan Kernel, lebih baik dari metode PREACO dengan kernel baik dari waktu komputasi ataupun akurasinya.

5.2 Saran

Penelitian berikutnya dapat dilakukan pada optimasi parameter ACO Kernel. Karena untuk mengaplikasikan metode ini membutuhkan kombinasi beberapa parameter.

LAMPIRAN

HASIL UJI COBA PENGELOMPOKAN DATA IRIS

CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	1	3	1	2
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		1	3	2	3	3		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		2	2	3	1	1		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	2	3	2	1		2	1	1	1	2		3	2	2	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	2	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	3	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		2	1	1	1	2		3	1	1	1	2
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	1	3	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	1	1	1	2
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	1	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	1	1	1	2		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	1	3	1	2
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	2	3	2	1
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		3	1	1	1	2
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		NB :				
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		CA	CLUSTER ASAL			
1	3	2	3	3		2	1	1	1	2		3	1	1	1	2		P	PREACO			
1	3	2	3	3		2	1	1	1	2		3	2	3	1	2		A	ACO			
1	3	2	3	3		2	1	1	1	2		3	2	3	2	1		AG	ACO – GAUSSIAN			
1	3	2	3	3		2	1	1	2	1		3	2	3	2	1		PG	PREACO – GAUSS			

HASIL UJI COBA PENGELOMPOKAN DATA WINE

CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG
1	2	2	1	1		1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	1	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	2	3	3		3	3	3	3	3
1	2	2	1	1		1	1	2	1	1		2	1	1	2	2		2	1	1	2	2		3	2	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	1	1	1	1		1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	1	2	1	1		2	3	3	3	3		2	3	1	2	2		3	1	3	3	3
1	2	2	1	1		1	1	2	1	1		2	1	1	2	2		2	1	1	2	3		3	3	3	3	3
1	2	2	1	1		1	1	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		3	1	1	2	2		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	3	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		3	1	3	3	3		3	3	3	3	3
1	1	2	1	1		1	2	2	1	1		2	1	1	1	1		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		1	2	2	1	1		2	1	1	2	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		2	2	2	1	1		2	1	1	2	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		2	1	1	2	2		2	2	1	2	2		3	3	3	3	3		3	3	3	3	3
1	1	1	1	1		2	1	1	3	3		2	1	1	2	2		3	3	3	3	3		3	3	3	3	3
1	2	2	1	1		2	1	3	3	3		2	1	1	2	2		3	3	3	3	3						
1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3		NB :				
1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	1	3	3	3		CA	CLUSTER ASAL			
1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3		P	PREACO			
1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3		A	ACO			
1	2	2	1	1		2	2	1	2	2		2	1	1	2	2		3	3	3	3	3		AG	ACO – GAUSSIAN			
1	2	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3		PG	PREACO – GAUSS			
1	2	2	1	1		2	1	1	3	3		2	1	1	2	2		3	3	3	3	3						
1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						
1	2	2	1	1		2	1	1	3	3		2	1	1	2	2		3	3	3	3	3						
1	2	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						
1	2	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						
1	1	1	1	1		2	1	2	1	1		2	1	1	2	2		3	3	3	3	3						
1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	1	3	3	3						
1	2	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						
1	1	2	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						
1	1	1	1	1		2	1	1	2	2		2	1	1	2	2		3	3	3	3	3						

HASIL UJI COBA PENGELOMPOKAN DATA T4

CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG
1	4	2	3	5		1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	5	4	2
1	4	2	3	5		1	1	2	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	4	1	2	1		2	1	1	2	1		2	1	1	2	1		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	1	1	2	1		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	1	1	2	1		2	3	1	2	1		3	3	3	4	2		3	1	1	4	2
1	1	2	3	5		1	1	2	3	5		2	3	2	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	4	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	1	3	5		1	1	2	3	5		2	1	1	2	1		2	3	1	2	1		3	1	2	4	2		3	1	1	4	2
1	1	1	3	5		1	4	2	3	5		2	1	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	1	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	1	1	2	1		2	3	1	2	1		3	1	1	4	2		3	5	3	4	2
1	4	1	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	3	4	2
1	1	2	3	5		1	1	2	3	5		2	4	1	2	1		2	1	1	2	1		3	1	1	4	2		3	1	4	4	2
1	1	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	3	4	2
1	4	1	3	5		1	1	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	3	1	4	2
1	4	2	3	5		1	4	2	3	5		2	4	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	1	4	2		3	1	5	4	2
1	1	2	3	5		1	1	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	1	4	2		3	1	1	4	2
1	1	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	1	1	3	4		1	4	1	3	5		2	1	1	2	1		2	1	1	2	1		3	1	4	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	5	1	4	2		3	4	1	4	2
1	1	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	5	4	4	2
1	4	1	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	4	1	4	2		3	3	1	4	2
1	4	2	3	5		1	1	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	1	4	2		3	1	1	4	2
1	1	2	3	5		1	1	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	1	2	3	5		1	4	1	3	5		2	3	1	2	1		2	3	1	2	1		3	1	2	4	2		3	1	1	3	2
1	2	2	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	4	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	3	4	2		3	5	1	4	2
1	4	2	3	5		1	4	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	3	4	2		3	5	1	4	2
1	4	2	3	5		1	1	2	3	5		2	1	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	3	1	2	1		2	1	1	2	1		3	1	1	4	2		3	1	5	4	2
1	4	2	3	5		1	1	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	1	2	3	5		1	1	1	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	3	4	2
1	1	2	3	5		1	4	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	4	1	3	5		2	3	1	2	1		2	1	1	2	1		3	1	3	4	2		3	5	3	4	2
1	4	2	3	5		1	1	2	3	5		2	3	1	2	1		2	5	1	2	1		3	1	1	4	2		3	1	1	4	2
1	4	2	3	5		1	1	2	3	5		2	3	1	2	1		2	3	1	2	1		3	1	1	4	2		3	1	5	4	2

CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG		CA	P	A	AG	PG	
3	1	1	4	2		4	1	3	5	4		4	5	4	5	4		5	2	5	1	3		5	2	3	1	3	
3	1	3	4	2		4	1	4	5	4		4	5	3	5	4		5	2	3	1	3		5	2	5	1	3	
3	1	1	4	2		4	5	4	5	4		4	1	3	5	4		5	2	5	1	3		5	2	3	1	3	
3	4	1	4	2		4	1	4	5	4		4	5	4	5	4		5	2	5	1	3		5	1	3	1	3	
3	1	3	4	2		4	1	3	5	4		4	1	4	5	4		5	1	5	1	3		5	2	3	1	3	
3	1	3	4	2		4	5	4	5	4		4	5	4	5	4		5	2	5	1	3		5	2	3	1	3	
4	3	1	5	4		4	5	4	5	4		4	5	4	5	4		5	1	5	1	3		5	2	3	1	3	
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	2	5	1	3		5	2	5	1	3	
4	1	4	5	4		4	5	4	5	4		5	2	5	1	3		5	1	3	1	3		5	2	5	1	3	
4	5	4	5	4		4	5	4	5	4		5	2	5	1	3		5	2	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	1	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	3	1	3		5	1	3	1	3							
4	5	4	5	4		4	5	4	5	4		5	1	3	1	3		5	2	5	1	3		NB :					
4	3	4	5	4		4	5	4	5	4		5	1	5	1	3		5	2	3	1	3		CA	CLUSTER ASAL				
4	1	4	5	4		4	5	1	5	4		5	1	5	1	3		5	2	5	1	3		P	PREACO				
4	5	4	5	4		4	1	4	5	4		5	1	5	1	3		5	2	5	1	3		A	ACO				
4	1	4	5	4		4	5	4	5	4		5	2	5	1	3		5	2	5	1	3		AG	ACO – GAUSSIAN				
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	2	5	1	3		PG	PREACO – GAUSS				
4	5	4	5	4		4	5	4	5	4		5	2	3	1	3		5	2	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	1	3	1	3							
4	1	4	5	4		4	1	4	5	4		5	2	3	1	3		5	5	3	1	3							
4	1	4	5	4		4	1	4	5	4		5	2	5	1	3		5	1	5	1	3							
4	1	4	5	4		4	5	4	5	4		5	2	5	1	3		5	1	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	1	5	1	3		5	1	5	1	3							
4	5	4	5	4		4	5	4	5	4		5	1	5	1	3		5	1	3	1	3							
4	5	4	5	4		4	5	4	5	4		5	2	5	1	3		5	1	5	1	3							
4	1	4	5	4		4	5	4	5	4		5	2	5	1	3		5	2	5	1	3							
4	1	4	5	4		4	1	4	5	4		5	2	5	1	3		5	2	5	1	3							
4	1	4	5	4		4	5	5	5	4		5	2	3	1	3		5	1	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	2	3	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	5	1	3		5	2	3	1	3							
4	5	4	5	4		4	5	3	5	4		5	2	5	1	3		5	2	3	1	3							
4	5	4	5	4		4	5	4	5	4		5	1	5	1	3		5	2	3	1	3							
4	1	4	5	4		4	5	4	5	4		5	2	3	1	3		5	2	3	1	3							
4	5	4	5	4		4	5	4	5	4		5	2	3	1	3		5	2	5	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	3	1	3		5	2	5	1	3							
4	5	3	5	4		4	3	4	5	4		5	2	5	1	3		5	2	3	1	3							
4	5	4	5	4		4	1	4	5	4		5	2	3	1	3		5	1	3	1	3							
4	5	4	5	4		4	5	4	5	4		5	2	3	1	3		5	1	5	1	3							

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut :

1. Berdasarkan hasil uji coba metode PREACO dengan *Kernel gaussian* dapat mengelompokkan dengan baik pada tingkat akurasi pada data wine= 93.8%, dan data T4=99.8% dan memiliki nilai SSE rendah. Hasil akurasi ini lebih baik dibanding dengan metode ACO yaitu untuk data wine=89.9%, dan data T4=70.8%. Namun memiliki kelemahan pada akurasi untuk data iris= 88.7% metode PREACO sedangkan metode ACO memiliki akurasi pada data iris=89.3%. Hal ini terjadi dapat disebabkan karena pada data cluster 2 dan 3 memiliki kedekatan yang beragam.
2. Metode ACO untuk *clustering* (Pengelompokan data) memiliki kecepatan dari sisi konvergen, terlihat pada percobaan memiliki iterasi dan waktu lebih sedikit.
3. Metode PREACO dengan Kernel lebih bagus dari ACO baik dari waktu komputasi ataupun akurasinya.
4. Metode ACO dengan Kernel, lebih baik dari metode PREACO dengan kernel baik dari waktu komputasi ataupun akurasinya.

5.2 Saran

Penelitian berikutnya dapat dilakukan pada optimasi parameter ACO Kernel. Karena untuk mengaplikasikan metode ini membutuhkan kombinasi beberapa parameter.

DAFTAR PUSTAKA

A. Lorette, X. Descombes, J. Zerubia. 2000. Fully Unsupervised Fuzzy *Clustering* With Entropy Criterion. 15Th International Conference on Pattern Recognition, IEEE, pp. 986-989.

C. Aditi, A. Darade. 1998. Swarm Intelligence Techniques: Comparative Study of ACO and BCO. Mumbay Unversity.

Chou, C.H., Su, M.C., Lai, E., 2004. A New *Cluster* Validity Measure and its Application to Image Commpression. Pattern Anal. Appl. 7 (2), 205-220.

Chun-Wei Tsai, S.P. Tseng, C.S. Yang, M.C. Chiang. 2013. PREACO: A fast ant colony optimization for codebook generation. Applied Soft Computing 13 (3008–3020).

D. Pelleg, A. Moore. 2000. X-means: extending k-means with efficient estimation of the number of *cluster*s. Proceedings of the Seventeenth International Conference on Machine Learning, San Fransisco, pp. 727-734.

G. Hamerly, C. Elkan. 2003. Learning the K in K-means, in:7th Annual Conference on Neural Information Processing Systems (NIPS), Vancouver and Whistler, British Columbia, Canada, pp. 281-288.

I. Gath, A.B. Geva. 1989.Unsupervised Optimal Fuzzy *Clustering*, IEEE Trans. Pattern Anal. Machine Intell. 11(773-780).

M. Tushir, S. Srivastava. 2010. A New Kernelized Hybrid c-mean *clustering* model with optimized parameters. Appl. Soft Comput. 10(381-389).

M. Dorigo, V. Maniezzo, A. Colorni. 1996. Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on SMC. Vol. 26, No. 1, pp. 29-41.

P.N. Tan, M. Steinbach, V. Kumar. 2006. Introduction to Data Mining. Pearson Education, Inc., Boston. (567-575).

R.J. Kuo, Y.D. Huang,C.C. Lin, Y.H. Wu, F.E. Zulvia. 2014. Automatic kernel *clustering* with bee colony optimization algorithm.Information Sciences 283 (107–122).

S. Das, A. Abraham, A. Konar. 2008. Automatic kernel *clustering* with a multi-elitist particle swarm optimization algorithm. Pattern Recogn. Lett. 29 (688–699).

BIOGRAFI PENULIS

Dwi Taufik Hidayat merupakan anak pertama dari orang tua bernama Mulyadi dan Chusnul Chotima, lahir di Madiun pada tanggal 24 April 1985. Penulis menempuh pendidikan di SD Kejapanan 2 Gempol Pasuruan, SMP Negeri 1 Gempol Pasuruan, SMA Negeri 1 Pandaan. Kemudian penulis melanjutkan pendidikan S1 di Jurusan Teknik Informatika, Universitas 17 Agustus 1945 Surabaya dari tahun 2004 hingga tahun 2009. Pada tahun 2011 penulis melanjutkan pendidikan S2 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan mengambil bidang minat Komputasi Cerdas dan Visualisasi. Bidang penelitian yang diminati oleh penulis adalah Sistem Cerdas. Penulis dapat dihubungi melalui email di taufikdwi17@gmail.com.