Pengelompokan Data Menggunakan *Pattern Reduction Enhanced Ant Colony Optimization* dan *Kernel Clustering*

Dwi Taufik Hidayat[1], Chastine Fatichah[2], R.V. Hari Ginardi[3]

***Abstract*— One method of optimization that can be used for *clustering* is Ant Colony Optimization (ACO). This method is good in data *clustering*, but has disadvantage in terms of time and quality or solution convergence. In this study, ACO-based Pattern Reduction Enhanced Ant Colony Optimization (PREACO) method with a gaussian kernel function is proposed. First, it sets up initial solution. Second, the magnitude of pheromone is calculated to find the centroid randomly. With the initialized solution, the weight of the solution is calculated and the center of cluster is revised. The solution will be evaluated through a gaussian kernel functions. Function 'pattern enhanced reduction' is useful to ensure maximum value of pheromone update. Those steps will be conducted repeatedly until the best solution is chosen. Tests are performed on multiple datasets, with three test scenarios. The first test is carried out to get the right combination of parameters. Second, the error rate measurement and similarity data using Sum of Squared Errors is done. Third, level of accuracy of the methods ACO, ACO with the kernel, PREACO, and PREACO with the kernel is compared. The test results show that the proposed method has a higher accuracy rate of 99.8% for synthetic data, 93.8% for wine data than other methods. But it has a lower accuracy by 88.7% compared to the ACO.**

***Intisari*— Salah satu metode optimasi yang dapat digunakan untuk *clustering* adalah *Ant Colony Optimization* (ACO). Metode ini baik dalam melakukan *clustering* data, namun memiliki kelemahan pada sisi waktu dan kualitas atau konvergensi solusi yang dihasilkan. Pada makalah ini diajukan metode *Pattern Reduction Enhanced Ant Colony Optimization* (PREACO) dengan fungsi Kernel Gaussian yang berdasarkan ACO. Pada awal metode dilakukan inisialisasi solusi dan nilai *pheromone* untuk kemudian ditentukan *centroid* secara acak. Melalui solusi yang telah ada maka akan dihitung bobot solusi dan perbaikan pusat *cluster*. Solusi akan dievaluasi melalui fungsi Kernel Gaussian. Fungsi *pattern reduction enhanced* berguna untuk memastikan nilai update *pheromone* agar maksimal. Langkah ini akan dilakukan secara terus menerus sampai solusi terbaik terpilih. Uji coba dilakukan pada beberapa *data set*, dengan tiga skenario uji coba. Pengujian pertama dilakukan untuk mendapat kombinasi parameter yang tepat. Kedua, dilakukan pengukuran tingkat kesalahan dan kesamaan data dengan menggunakan pengukuran *Sum of Squared Error*. Ketiga dilakukan perbandingan tingkat akurasi metode ACO, ACO dengan kernel, PREACO, dan PREACO dengan kernel. Hasil pengujian menunjukkan bahwa metode usulan memiliki tingkat akurasi lebih tinggi sebesar 99,8% untuk data sintesis, 93,8% untuk data *wine* dibanding lainnya, tetapi memiliki akurasi lebih rendah dengan 88,7% dibanding ACO.**

***Kata Kunci*— Kernel *Clustering, Ant Colony Optimization, Pattern Reduction Enhanced Ant Colony Optimization*.**

I. PENDAHULUAN

Cluster merupakan pengelompokan dari banyak data dengan berdasarkan pada kesamaan atau perbedaannya. Kesamaan maupun perbedaan itu dilihat dari ciri yang dipunyai data tersebut. Misalnya saja terdapat sebuah data termasuk ke dalam *Cluster* tertentu dikarenakan memiliki ciri yang sama dengan *Cluster* itu. Pengklusteran sebuah data sesuai *cluster* masing-masing dilakukan dengan metode tertentu. Oleh karena itu melakukan *cluster* terdapat dua jenis yaitu *soft clustering* dan *hard clustering*. Maksudnya, sebuah data bisa termasuk ke dalam beberapa *cluster*. Hal ini disebut dengan *soft clustering*. Sebaliknya, sebuah data harus dimasukkan ke dalam satu *cluster* tertentu saja, disebut dengan *hard clustering*.

Telah banyak metode *clustering* dikembangkan. Metode *k-means*, *hierarchical algorithm*, SOM [1], [2], aglomerasi [3] adalah satu 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 ke dalam 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* di awal diperlukan. Penelitian tentang permasalahan tersebut masih terbuka untuk diteliti. Salah satu metode yang berusaha menyelesaikan permasalahan itu adalah kernel *clustering*, untuk mendefinisikan dengan baik jumlah *cluster*, sehingga pada akhirnya sebuah data akan dimasukkan ke dalam sebuah *cluster* dengan tepat.

Cluster analysis atau kernel *clustering* saat ini menjadi rujukan dalam melakukan pembelajaran tak terawasi pada proses pencarian *cluster*. Berangkat dari kelemahan ini, telah dilakukan penelitian dengan menggabungkan *fuzzy C-means* dan *fuzzy maximum likelihood* untuk mendapatkan *cluster* yang tak terawasi [4]. Telah diajukan juga sebuah metode *fuzzy clustering* untuk menentukan jumlah *cluster* secara dinamis [5]. Selain itu, juga banyak penelitian dalam penentuan jumlah *cluster*. Berangkat dari kelemahan yang dimiliki *K-means* terdapat banyak metode yang diajukan yaitu *X-means* [6] dan *G-means* [7]. Keduanya bertolak dari penentuan jumlah kelas.

[1]*Mahasiswa, Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabya 60111 INDONESIA, (tlp: 031-5939214; fax: 031-5939363; e-mail: dwitaufikhidayat@widyakartika.ac.id)*

[2,3]*Dosen, Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60111 INDONESIA (tlp: 031-5939214; fax: 031-5939363; e-mail: hari@its.ac.id, chastine@cs.its.ac.id)*

Penelitian di bidang kernel *clustering* banyak mengalami perkembangan. Salah satu metode hasil penelitian tersebut adalah menggabungkan *swarm intelligence* pada kernel *clustering*. Sebuah penelitian mengusulkan suatu metode gabungan tersebut yaitu *Automatic Kernel Clustering with Multi-Elitist Particle Swarm Optimization* (AKC-MEPSO) [8]. 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 PSO yang telah dimodifikasi (MEPSO).

Metode fungsi kernel yang banyak digunakan adalah *Polynomial* Kernel, Gaussian Kernel, *Radial* basis Kernel and *Hyper tangent* [9]. Penelitian yang menggunakan *kernel function* digabungkan dengan algoritme *bee colony optimization* (BCO), yang merupakan algoritme *evolutionary algorithm*, membuktikan bahwa *kernel function* dan *evolutionary algorithm* dapat dikolaborasikan untuk menentukan jumlah *cluster* secara tepat [10]. Penggabungan yang telah dilakukan 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 algoritme *evolutionary algorithm* membuka peluang akan jenis penelitian lain. Penelitian *Ant Colony Optimization* (ACO) [11] merupakan bagian dari algoritme itu, sehingga dapat digunakan pada penelitian ini untuk menambah kontribusi ilmu. Metode ini masih berkembang sampai saat ini.

Penelitian mengenai ACO telah disarikan, walaupun dalam mensarikannya dibandingkan dengan kemampuan yang dimiliki oleh BCO [12]. Pada dasarnya ACO lebih menjamin dari sisi konvergensi dan mampu beradaptasi dengan lingkungannya daripada BCO. Tetapi dari segi waktu dan komputasi BCO lebih unggul. Oleh karena itu terdapat pengembangan metode pada ACO juga banyak terjadi. Salah satunya, *Pattern Reduction Enhanced Ant Colony Optimization* (PREACO) [13], membuktikan bahwa ACO juga dapat lebih bagus dari sisi biaya komputasi dan waktu. Walaupun begitu, kualitas hasil mengalami penurunan akurasi sehingga memerlukan pengembangan selanjutnya. Penelitian tentang PREACO dikhususkan pada kasus *codebook generation* bukan untuk *clustering*, namun terbukti meningkatkan performa ACO dari sisi waktu komputasi.

Oleh karena itu, penelitian dalam makalah ini akan melakukan pengelompokan data menggunakan kombinasi metode PREACO dengan kernel *clustering*. Pada usulan ini 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 seperti *data set iris*, *wine*, *glass* dan beberapa yang lain.

Metode ini mempercepat proses pemilihan solusi dan menghindari redundansi pemilihan solusi untuk mendapatkan jumlah *cluster* dan proses *clustering* yang tepat. Adapun fungsi kernel yang digunakan adalah Kernel Gaussian. Pengujian akan dilakukan terhadap konvergensi metode dan juga akurasinya. Metode ini diujicobakan kepada *data set* standar klasifikasi yang terdapat pada UCI *data set*.

II. *ANT COLONY OPTIMIZATION* UNTUK *CLUSTERING*

Pada *hard partitioning* dicoba dilakukan pemisahan data menuju kelompok data yang sama secara pasti. Misalnya $X = \{x_1, x_2, x_3 ... x_n\}$ adalah sekumpulan data yang belum terkelompokkan dengan dimensi tertentu. Masing-masing data xi,j dalam sebuah vektor Xi memiliki atribut atau fitur (j=1,2,...,d) dengan (i=1,2,...,n). Untuk kemudian data-data ini mencoba memasukkan ke dalam sebuah partisi *cluster* C={$C1,C2,...,Ck$} 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 memetakan data dengan baik.

A. Ant Colony Optimization untuk Clustering

Algoritme ACO berawal dari optimasi. ACO juga digunakan untuk mengoptimasi algoritme *K-Means*. Selain digunakan untuk optimasi, algoritme ini digunakan untuk proses analisis *cluster*. Maka terciptalah algoritme *clustering* dari *ant colony*.

Clustering memetakan data sesuai dengan *cluster* menggunakan ACO. Algoritme dimulai dengan menyebarkan agen semut untuk menemukan sebuah solusi *cluster*. Solusi yang terbentuk kemudian memperbarui matriks *pheromone*. Matriks *pheromone* bertindak sebagai memori jejak solusi sementara, digunakan untuk memandu agen semut berikutnya memilih solusi yang sama.

Fungsi objektif diperlukan untuk menguji solusi yang terbentuk. Fungsi objektif berisikan persamaan jarak antar data dan pusat *cluster*. Jika nilai fungsi objektif semakin kecil maka solusi semakin baik.

Metode ini menggunakan kernel *clustering* sebagai fungsi objektif. Kernel *clustering* yang digunakan adalah Kernel Gaussian. Jika nilai hasil Kernel Gaussian semakin kecil maka hasil *cluster* semakin bagus. Fungsi objektif menggunakan kernel gaussian sebagai pembagi. Oleh karena itu, nilai menjadi terbalik, semakin besar hasil fungsi objektif, maka solusi semakin baik.

Setiap agen semut membentuk solusi. Solusi yang terbentuk terdiri atas kolom sebanyak jumlah data dan baris berisi angka *cluster*. Setiap agen semut menunjuk angka *cluster* untuk ditugaskan pada setiap elemen data dan diisikan pada kolom data yang sesuai.

Agen semut membentuk solusi dengan membuat bilangan acak dalam rentang antara 0 dan 1 ($0<q<1$). Kemudian bilangan acak tersebut digunakan untuk menunjuk angka *cluster* tertentu. Persamaan (1) menunjukkan mekanisme pemilihan angka *cluster* melalui angka acak tersebut.

$$s = \begin{cases} \arg\max_{j \in K} \{\tau_{ij} \cdot [\eta_{ij}]^{\beta}\}, \; jika\_q \le q0 \\ J, \; Jika\_lainnya \end{cases} \quad (1)$$

Misal sebagai contoh *N*=8 poin data, *K*=3, *R*=10. Agen semut pertama akan melakukan proses pencarian solusi. Pertama dibuat bilangan acak sebanyak *N* buah (0.9872, 0.8763, 0.7689, 0.65....). Nilai q0 didefinisikan secara manual, misal q0=0.98.

Langkah selanjutnya adalah membandingkan setiap bilangan acak dengan *q0*. Apabila dalam membandingkan bilangan acak kurang dari *q0*, maka langkah berikutnya adalah melakukan (1). *Cluster* terpilih berdasarkan perkalian nilai *pheromone* dengan informasi *heuristic* yang berupa jarak, sehingga otomatis *cluster* akan menunjuk pada nilai *pheromone* terbesar.

Jika bilangan acak lebih dari *q0* maka berdasarkan (1), dilakukan *J*. *J* adalah memilih variabel acak sesuai dengan distribusi probabilitas. Distribusi probabilitas dapat diselesaikan menggunakan (2).

$$pij = \frac{\tau_{ij}}{\sum_{k=1}^{K} \tau_{ik}}, \; j = 1,..., K \quad (2)$$

Proses pembobotan dilakukan setelah didapatkan solusi oleh agen semut. Bobot bernilai 1 untuk setiap *cluster* terpilih pada elemen data, sedangkan pada *cluster* lain bernilai 0. Misalnya, untuk elemen data pertama berisikan *cluster* 2, maka hasil pembobotan bernilai 0, 1, 0 dengan jumlah *cluster* berjumlah 3. Nilai 0 mewakili *cluster* 1 dan 3 serta 1 mewakili *cluster* 2 terpilih (3).

$$w_{ij} = \begin{cases} 1, \; jika\_elemen\_i\_termasuk\_clusterj \\ 0, \; jika\_sebaliknya \end{cases} \quad (3)$$

Perhitungan pusat *cluster* didasarkan pada matriks bobot yang telah tercipta. Hasil penjumlahan dari perkalian setiap data dan setiap bobot dibagi dengan jumlah bobot akan menghasilkan matriks pusat *cluster* (4).

$$m_{jv} = \frac{\sum_{i=1}^{N} w_{ij} x_{iv}}{\sum_{i=1}^{N} w_{ij}}, \; j = 1,..., K\_dan\_v = 1,..., n \quad (4)$$

Perhitungan dalam menentukan matriks solusi dilakukan secara terus menerus sepanjang *R* agen semut. Setiap solusi terbentuk selalu melalui proses pemilihan solusi, pencarian bobot dan memperbarui matriks pusat *cluster*.

Proses berikutnya setelah mendapatkan matriks solusi dengan ukuran *RxN* adalah pencarian solusi terbaik. Berdasarkan penelitian sebelumnya, jumlah solusi yang akan didapatkan akan diambil 20% dari total jumlah *R*. Apabila agen semut berjumlah sepuluh maka akan diambil dua solusi terbaik.

Proses mendapatkan solusi terbaik dilakukan dengan cara menghitung nilai fungsi objek setiap solusi, sehingga apabila terdapat sepuluh agen semut mewakili sepuluh solusi, maka terdapat sepuluh nilai fungsi objektif. Proses selanjutnya adalah mengurutkan semua solusi dari terkecil menuju terbesar. Tetapi pada kasus penelitian ini adalah dari terbesar menuju terkecil, dengan dua urutan teratas akan diambil untuk dilakukan proses peningkatan solusi pada *local search*.

Local search dilakukan dengan membuat bilangan acak antara 0 dan 1 sebanyak *N* data. Pada penelitian ACO, setiap bilangan acak dibandingkan dengan 0,01. Jika kurang dari batas tersebut, maka *cluster* pada solusi tersebut harus diganti dengan *cluster* lainnya tapi tidak boleh *cluster* yang sama. Pemilihan *cluster* dilakukan secara acak. Setelah itu, bobot, pusat *cluster*, dan nilai fungsi objektif dihitung ulang. Nilai fungsi objektif akan dibandingkan dengan nilai fungsi objektif sebelumnya. Misal, nilai fungsi objektif sebelum dilakukan *local search* adalah 1,876 dan nilai setelah proses tersebut 1,987. Melihat perbandingan tersebut, maka solusi setelah proses *local search* akan dipilih karena menghasilkan solusi terbaik. Langkah ini akan dilakukan berulang sampai dengan jumlah solusi terpilih.

Setelah terpilih solusi terbaik, maka langkah selanjutnya adalah memperbarui matriks *pheromone*. Agen semut berikutnya, sebelum memilih solusi *cluster* untuk masing-masing elemen data, terlebih dahulu harus melihat nilai matriks *pheromone* terbesar. Dengan melihat *pheromone* terbesar, maka agen semut akan memilih angka klaser sesuai dengan penunjukan posisi nilai *pheromone*.

B. PREACO

PR atau *pattern reduction* merupakan bagian dari metode PREACO. Fungsi PR menyesuaikan dengan algoritme 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. Terlebih dahulu didefinisikan nilai *R, q, lmd*. Setelah itu maka masuk pada perulangan untuk mencatat seberapa banyak penugasan agen semut dalam menunjuk *cluster* tertentu dan menyimpannya dalam suatu *array* tertentu. Setelah sampai pada titik *threshold* tertentu yaitu perkalian nilai *R, q, lmd*, maka nilai *pheromone* diberi nilai 1.

Proses algoritme dari metode yang diajukan dapat dilihat pada Gbr. 1.

C. Kernel Gaussian

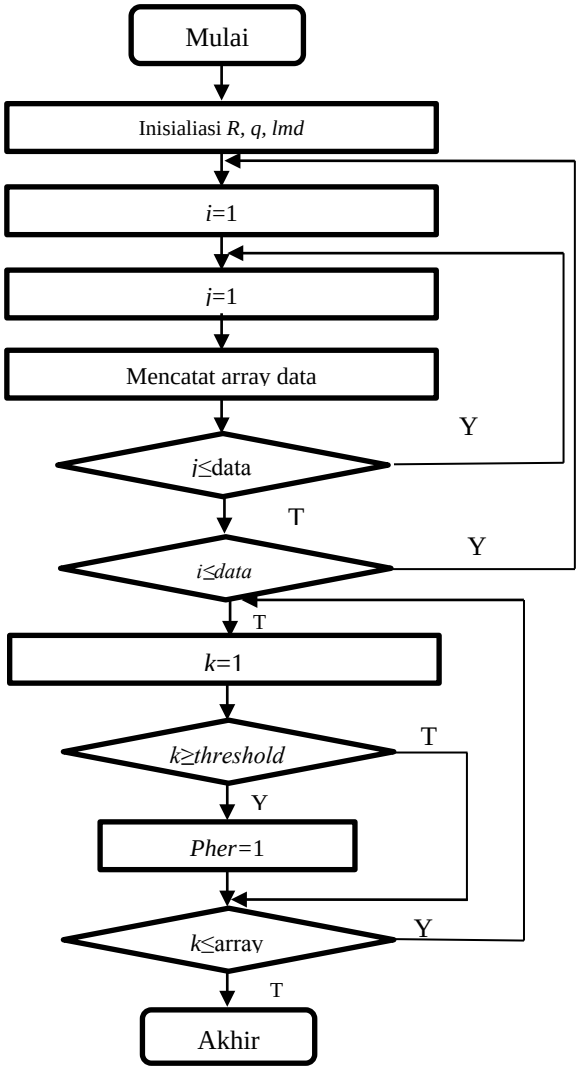
Di dalam melakukan proses pengelompokan data menggunakan metode ACO dibutuhkan fungsi objektif untuk memastikan validitasnya, karena penggunaan fungsi objektif yang tepat menentukan bagus tidaknya metode. Pada makalah ini digunakan fungsi objektif Gaussian. Fungsi objektif dapat kita sebut sebagai *fitness function* dengan (5) bergantung pada bobot (*w*), elemen data (*x*), dan pusat *cluster* (*m*), di mana nilai *K* adalah jumlah *cluster*, *N* data, dan *n* atribut data.

$$\text{Min } F(w,m) = \sum_{j=1}^{K} \sum_{i=1}^{N} \sum_{v=1}^{n} w_{ij} \left\| x_{iv} - m_{jv} \right\|^2 \quad (5)$$

Fungsi objektif Gaussian selanjutnya digunakan sebagai kernel metode pengelompokan data. Gaussian dipilih karena berdasarkan penelitian sebelumnya memiliki keunggulan lebih *robust* dan lebih responsif untuk data *multivariate*. Adapun fungsi objektif menggunakan Kernel Gaussian menggunakan

(6). Pada bagian atas dihitung jarak antar data dalam *cluster* sama, sedang pada bagian pembagi dihitung jarak antar pusat *cluster*.

$$CS(K) = \frac{\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\left\{d\left(\vec{X}_i, \vec{X}_q\right)\right\}\right]}{\sum_{i=1}^{K}\left[\min_{j\in K, j\neq i}\left\{d\left(\vec{m}_i, \vec{m}_q\right)\right\}\right]} \quad (6)$$



Gbr. 1 Algoritme *Pattern Reduction*.

TABEL I
DESKRIPSI DATA SET

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

Adanya Kernel Gaussian ini solusi yang terbentuk akan diberikan nilai berdasarkan tingkat kebenarannya. Semakin kecil nilainya, maka semakin bagus solusi yang terbentuk. Tetapi berdasarkan (6), semakin besar nilai Kernel Gaussian maka semakin bagus solusi yang terbentuk.

III. HASIL DAN PEMBAHASAN

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.

Skenario uji coba rumusan dilakukan menggunakan data dari UCI *data set*. Uji coba dilakukan untuk mendapatkan rasio efektivitas 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 yang dilakukan adalah sebagai berikut:

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 komposisi parameter yang tepat dengan mengujicobakan kemungkinan parameter pada beberapa *data set*. Pengambilan parameter didasarkan pada waktu nilai *Sum of Squared Error* (SSE) dan waktu komputasi. Pengujian kedua adalah pengujian menggunakan nilai SSE. Metode atau sistem diujicobakan pada banyak nilai *K*, sehingga pada hasil nilai *K* tersebut akan terlihat, pada K ke berapa solusi akan optimal dalam mempartisi data. Persamaan SSE yang digunakan dapat dilihat pada (7). Dengan nilai *K* adalah jumlah *cluster*, *x* mewakili elemen data, *m* adalah pusat *cluster*, di mana *C* adalah *cluster*.

$$SSE = \sum_{i=1}^{K}\sum_{x \in C_i}\left(x - m_i\right)^2 \quad (7)$$

$$\text{Akurasi} = \frac{Jumlah\_cluster\_yang\_benar}{Total\_jumlah\_cluster} \quad (8)$$

Pengujian dilakukan dengan melakukan koleksi data kemudian dilakukan normalisasi data, pengujian *algoritme*, evaluasi efisiensi, dan efektivitas pembentukan *cluster*. Untuk mendapatkan hasil performa yang bagus dari *algoritme*, dilakukan uji akurasi (8). Hal ini dilakukan dengan menghitung jumlah *cluster* yang benar dibanding total keseluruhan *cluster* yang terbentuk. Selain itu, untuk menguji metode diperlukan evaluasi antar metode.

A. Pemilihan Data Set

Data set yang digunakan pada penelitian ini adalah data *iris*, *wine*, dan T4, seperti pada Tabel I. Data *iris* dan *wine* sering digunakan pada metode yang sama berdasarkan penelitian sebelumnya. Sedangkan data T4 didapat untuk menguji kernel berdasarkan kedalaman data.

Data *Iris* pada tabel memiliki gambaran sebagai berikut. Data dengan jumlah sebanyak 150 memiliki jumlah *K* = 3 dan 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 *data set*.

Data *Wine* mengalami normalisasi data. Data ini memiliki keragaman yang sangat tinggi antar atribut sehingga diperlukan normalisasi data. Normalisasi data yang digunakan adalah menggunakan normalisasi *min max*, seperti pada (9). 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 dengan mengunduh melalui situs UCI *data set*.

$$NilaiBaru = \frac{NilaiAsal - NilaiMinAsal}{NilaiMaxAsal - NilaiMinAsal} \quad (9)$$

Sementara itu, 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.

B. 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 *data set* yang sama pada beberapa metode yang berbeda. Adanya parameter pada berbagai metode ini bertujuan untuk mendapatkan nilai parameter yang tepat pada masing-masing metode dengan hasil terbaik. Kriteria tepat adalah 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*.

C. Pengujian Menggunakan Sum of Squared Error (SSE)

Data yang telah terpetakan dan tersusun sesuai *cluster*-nya akan diujicobakan pada metode. Hasil keluaran dari aplikasi ini adalah berupa kumpulan *string* solusi sepanjang data. Matriks solusi yang terbentuk memiliki satu baris dengan jumlah kolom sepanjang jumlah data. Kemudian solusi ini dihitung menggunakan SSE.

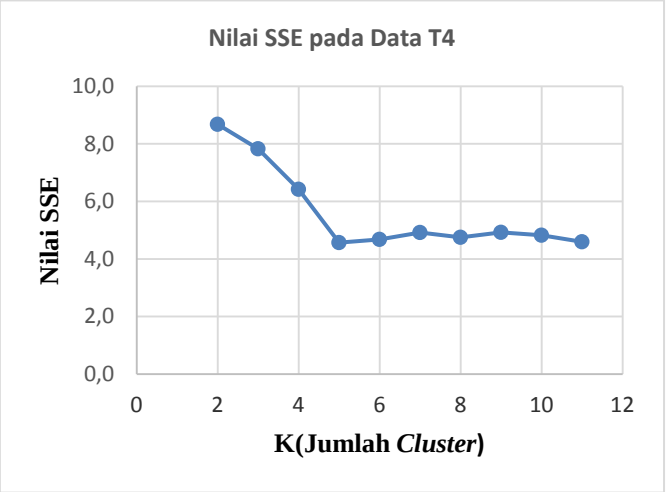
Skenario pengujian dilakukan pada semua jenis data dengan masing-masing diuji beragam nilai *K*. Nilai *K* yang diuji minimal berjumlah 2 sampai nilai *K* lebih dari 10. Untuk masing-masing nilai *K* akan didapatkan nilai OF dan SSE-nya, sehingga dengan begitu dapat terlihat dengan jelas kombinasi SSE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

Adapun kombinasi 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. Sebagai contoh disajikan gambar hasil uji SSE pada data sintesis T4 pada Gbr. 2.

D. Pengujian Menggunakan Akurasi

Pengujian dilakukan pada masing-masing *data set* 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.

Tabel II menggambarkan mengenai hasil uji coba metode untuk mengelompokkan data. Terdapat empat 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. Disajikan hasil *confusion clustering* untuk data *iris*.



Gbr. 2 Nilai SSE pada data T4.

Tabel III menceritakan bahwa metode ini mampu mengelompokkan data ke dalam *cluster* 1 sebanyak 49 buah data dengan salah mengelompokkan ke dalam *cluster* 3 sebanyak 1 buah data. *Cluster* 2 memiliki tingkat kesalahan *cluster* pada *cluster* 3 sebanyak 2 buah data. Dan *cluster* 3 mengalami kesalahan mengelompokkan pada *cluster*1 sebanyak 11 buah dan 2 buah data salah masuk kelompok *cluster* 2. Dari segi waktu komputasi, metode ini memiliki waktu komputasi sebesar 73 detik.

Seperti terlihat pada Tabel IV untuk metode ini menghasilkan kebenaran sempurna dengan berhasil memetakan 50 data dengan benar pada kelas 1. Sedangkan pada kelas 2 berhasil dipetakkan 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 ke dalam kelas 3. Waktu komputasi metode memerlukan 11 detik untuk mengelompokkan data.

Tabel V adalah matriks *confusion clustering* untuk data Iris menggunakan metode PREACO dengan kernel. 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.

TABEL II
AKURASI ALGORITME

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

TABEL III
HASIL METODE ACO UNTUK DATA IRIS

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

TABEL IV
HASIL METODE ACO DENGAN KERNEL UNTUK DATA IRIS

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

TABEL V
HASIL METODE PREACO DENGAN KERNEL UNTUK DATA IRIS

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

IV. KESIMPULAN

Makalah ini telah mengajukan sebuah gabungan metode PREACO dengan kernel untuk memperbaiki kinerja ACO *clustering*. 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 dibandingkan dengan metode ACO yaitu untuk data wine = 89,9%, dan data T4 = 70,8%. Namun ada 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.

Metode ACO untuk *clustering* (pengelompokan data) memiliki kecepatan dari sisi konvergensi, terlihat pada percobaan memiliki iterasi dan waktu lebih sedikit. Metode PREACO dengan Kernel lebih bagus dari ACO, baik dari waktu komputasi ataupun akurasinya. Metode ACO dengan Kernel lebih baik dari metode PREACO dengan kernel, baik dari waktu komputasi ataupun akurasinya.

REFERENSI

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

[2] P.B.S. Wiguna, B.S.Hartono, "Peningkatan Algoritme Porter Stemmer Bahasa Indonesia berdasarkan Metode Morfologi dengan Mengaplikasikan Tingkat Morfologi dan Aturan Kombinasi Awalan dan Akhiran," *JNTETI*, Vol. 2, No. 2, hal. 1-6 , Mei 2013.

[3] N.F. Azzahra, H. Ginardi, A. Saikhu, "Praproses Data Alir ADS-B dari *Multi-Receiver* dengan Pengelompokan Agglomerasi Berbasis Konsistensi Jarak" *JNTETI*, Vol. 4, No. 1, hal. 39-44, Februari 2015.

[4] I. Gath, A.B. Geva, "Unsupervised Optimal Fuzzy *Clustering*", *IEEE Trans. Pattern Anal, Machine Intell*, No. 11, hal.773-780, 1989.

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

[6] D. Pelleg, A. Moore, "X-means: extending k-means with efficient estimation of the number of clusters", *Proceedings of the Seventeenth International Conference on Machine Learning*, San Fransisco, hal. 727-734, 2000.

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

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

[9] M. Tushir, S. Srivastava, "A New Kernelized Hybrid c-mean *clustering* model with optimized parameters", *Appl. Soft Comput*, No. 10, hal. 381-389, 2010.

[10] 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*, hal. 107–122, 2014.

[11] Marco Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the tracelling salesman problem", *IEEE Transaction on Evolutionary Computation*,1997(1), hal. 53-66.

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

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