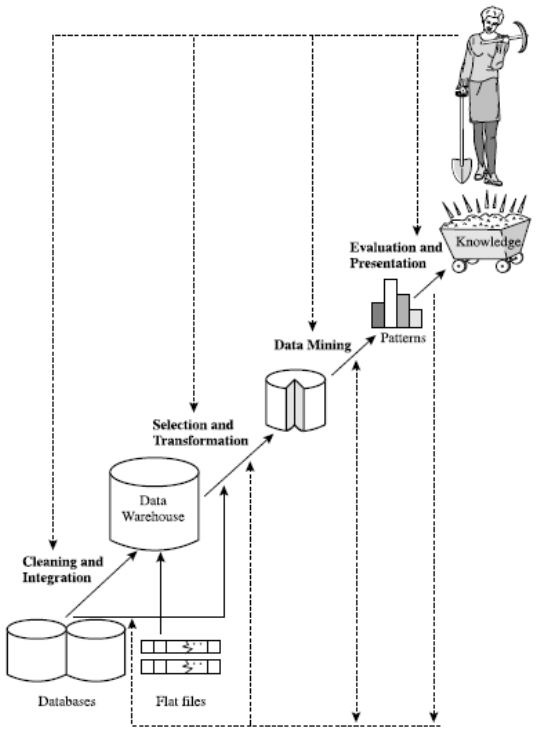
BAB 2
TINJAUAN PUSTAKA

2.1 Data Mining

Data mining merupakan kegiatan menambang data dalam jumlah yang besar dengan tujuan untuk menemukan pola dan pengetahuan yang bermakna sehingga dapat digunakan untuk mendukung pengambilan keputusan [**1**]. *Data mining* menggunakan teknik statistik, matematika, dan kecerdasan buatan untuk mengekstraksi dan mengidentifikasi informasi dan pengetahuan yang bermanfaat dari berbagai sumber *database*. Keluaran yang dihasilkan dari *data mining* dapat digunakan untuk memperbaiki dan membantu pengambilan keputusan di masa depan.

Sebagian orang melihat data *mining* hanya sebagai satu langkah penting pada proses KDD (*knowledge discovery from data*). Gambar 2.1 merupakan langkah-langkah dari proses KDD [**1**].



Gambar 2.1 Tahapan *Knowledge Discovery From Data* **[**1**]**

a. Data *Cleaning and Integration*

   Data *Cleaning* berguna untuk menghilangkan *noise* data yang tidak konsisten serta mengitegrasikan data, dimana sumber data yang terpecah disatukan.

b. Data *Selection and Transformation*

   Data *selection* memilih data yang sesuai dengan analisis yang akan dilakukan dan mengubahnya agar dapat digunakan didalam proses data *mining*.

c. Data *Mining*

   Proses penerapan teknik *data mining* seperti *clustering*, *classification*, atau *association rule* yang digunakan untuk mencari pola atau pengetahuan pada data.

d. *Evaluation and Knowledge Representasion*

   Proses untuk mengindentifikasi pola yang benar-benar menarik dan mewakili pengetahuan serta memvisualisasikannya kepada pengguna.

Dalam *data mining* terdapat beberapa teknik yang bisa digunakan untuk menambang pola atau pengetahuan dari data. Pemilihan teknik *data mining* biasanya dilihat bedasarkan sumber data yang akan diproses atau pola informasi seperti apa yang ingin diketahui. Beberapa teknik *data mining* yang umum digunakan saat ini adalah sebagai berikut :

1. *Association Rule*

   Association rule atau aturan asosiasi adalah salah satu teknik *data mining* dalam *Market Basket Analysis* untuk mencari hubungan antar-*item* dalam suatu data dan menampilkan bentuk *association rule* atau pola kombinasi dari suatu item [**1**]. *Association rule* (aturan asosiasi) akan menemukan pola tertentu untuk mengasosiasikan data yang satu dengan data yang lain.

2. Classification

   Classification atau Klasifikasi adalah salah satu teknik *data mining* suatu yang melakukan proses pengelompokan terhadap suatu obyek untuk masuk dalam suatu kelas tertentu dari sejumlah kelas yang tersedia [**1**].

Klasifikasi termasuk ke dalam *supervised learning* atau pembelajaran terbimbing dimana setiap objek data memiliki kelas yang menjadi patokan objek tersebut adalah anggota dari kelompok kelas tersebut.

3. Clustering

   Clustering adalah salah satu teknik *data mining* yang digunakan untuk mengelompokkan sejumlah data atau obyek ke dalam k buah *cluster* bedasarkan kemiripan antar datanya [**1**] .*Clustering* termasuk ke dalam *unsupervised learning* atau pembelajaran tidak terbimbing dimana tidak ada pendefinisian kelas objek sebelumnya. Sehingga *clustering* dapat digunakan untuk menentukan label kelas bagi data – data yang belum diketahui kelasnya.

2.2 Data Mining Berbasis Clustering

Data *mining* berbasis *clustering* adalah salah satu teknik *data mining* yang dapat digunakan untuk mencari pola atau pengetahuan dari data. Data yang akan diproses dalam data mining ada kalanya masih belum diketahui label kelasnya. Dalam kasus *data mining*, data semacam ini dapat dikelompokkan berdasarkan kemiripan dan ketidakmiripan antar atribut datanya atau sering disebut dengan pembelajaran tidak terbimbing (*unsupervised learning*). Karena tidak memerlukan label kelas didalam prosesnya, *clustering* sangat cocok digunakan untuk pengelompokkan data yang label kelasnya memang belum diketahui.

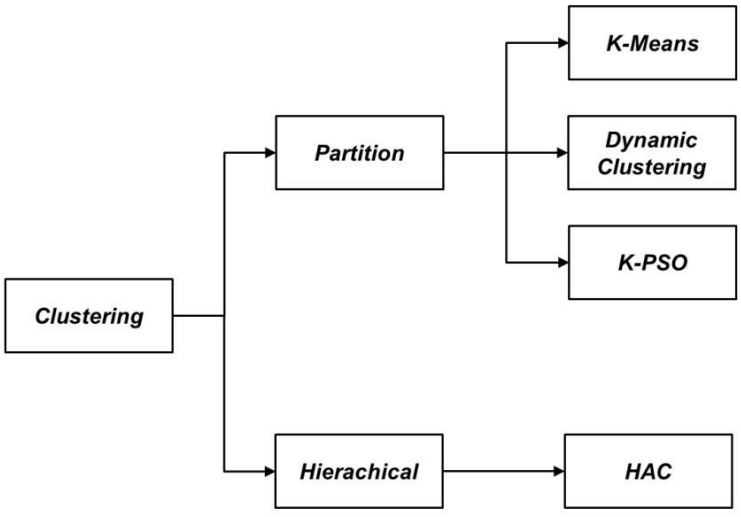
Clustering adalah salah satu teknik *data mining* yang digunakan untuk mengelompokkan sejumlah data atau obyek ke dalam k buah *cluster* bedasarkan kemiripan antar datanya [**1**].*Clustering* berusaha untuk memaksimalkan kemiripan anggota dalam satu *cluster* dan memaksimalkan perbedaan dengan *cluster* lainnya, sehingga kualitas *cluster* yang terbentuk dapat dikatakan baik. *Clustering* termasuk ke dalam *unsupervised learning* atau pembelajaran tidak terbimbing, artinya dalam proses clustering tidak memerlukan pendefinisian terhadap label kelas data.

Dalam *clustering* hal yang paling penting adalah mengukur kemiripan (*similarity*) atau ketidakmiripan (*dissimilarity*) dari data yang akan

dikelompokkan. Hasil *clustering* dapat dikatakan baik jika data memiliki kemiripan yang tinggi didalam satu *cluster* yang sama dan memiliki ketidakmiripan yang tinggi dengan data di *cluster* lainnya. Kemiripan antara dua data merupakan ukuran numerik dari seberapa dekat dua buah data tersebut. Kemiripan mempunyai nilai yang lebih tinggi jika karakteristik data tersebut sama. Ketidakmiripan antara dua data merupakan ukuran numerik dari seberapa jauh dua buah data tersebut. Ketidakmiripan sering disebut juga dengan *distance* (jarak) [**8**]. Pada proses *clustering*, untuk mengetahui kemiripan dan ketidakmiripan adalah dengan mengukur jarak antar data menggunakan metode pengukuran jarak [**8**]. Salah satu metode pengukuran jarak adalah metode jarak *Euclidian* (*Euclidian Distance*).

2.3 Teknik Dalam Clustering

Secara umum teknik *clustering* dibagi menjadi dua yaitu *clustering* berbasis hierarki dan *clustering* berbasis partisi. Pembagian teknik *clustering* dapat dilihat di gambar berikut [**8**].



Gambar 2.2 Teknik Dalam *Clustering*

1. *Clustering* Berbasis Partisi

   Dalam metode ini pengguna harus menentukan jumlah n partisi yang diinginkan lalu setiap data dimasukan pada salah satu partisi yang cocok dengan data tersebut sehingga tidak ada data yang mejadi *overlap*. Pada metode partisi satu data hanya boleh memiliki satu *cluster*.

   Contohnya: algoritma *K-Means* dan *Kmeans-PSO*.

2. *Clustering* Berbasis Hierarki

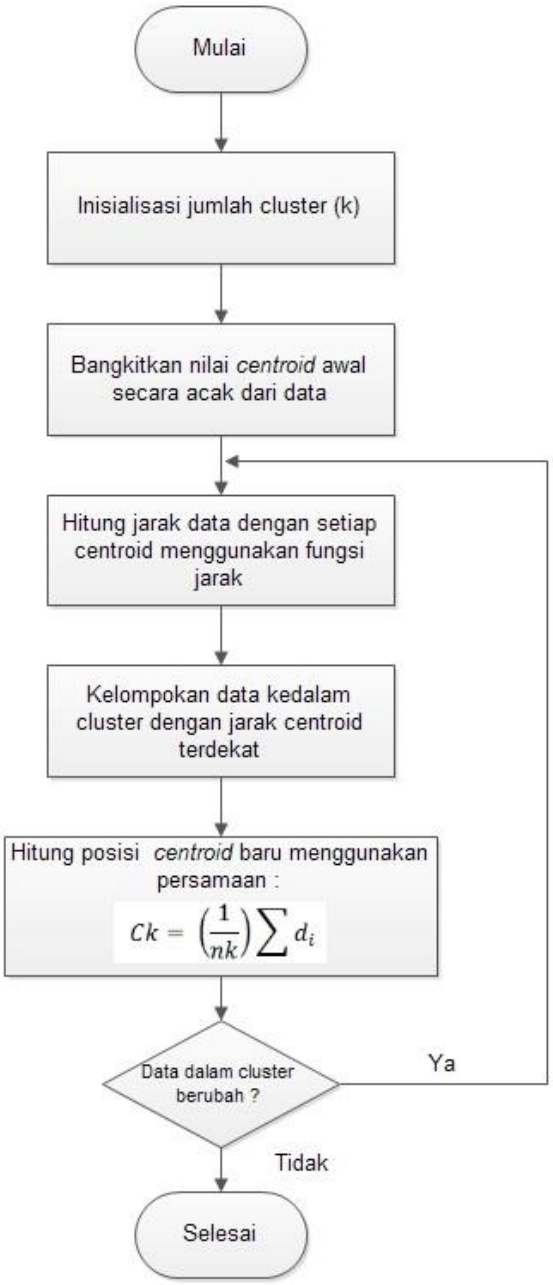
   Metode ini membuat hierarki berupa *dendogram* dimana data yang mirip akan disimpan pada hirarki yang berdekatan. Dalam metode hierarki satu data tunggal bisa dianggap sebuah *cluster*, dua atau lebih *cluster* kecil dapat bergabung menjadi sebuah *cluster* besar, begitu seterusnya hingga semua data dapat bergabung menjadi sebuah *cluster*. Metode ini terbagi menjadi dua kategori yaitu *bottom-up* yang menggabungkan *cluster* kecil menjadi *cluster* yang lebih besar dan *top-down* yang memecah *cluster* besar menjadi *cluster* lebih kecil.

   Contohnya: algoritma *Hierarchical Agglomerative Clustering* (HAC).

2.4 Algoritma K-Means

Algoritma *K-Means* merupakan salah satu algoritma dari teknik *clustering* berbasis partisi yang berusaha membagi data ke dalam beberapa *cluster*. Metode ini mengelompokan data ke dalam *cluster* bedasarkan kemiripan karakteristik yang dimiliki data, dimana setiap data yang memiliki karakteristik sama dikelompokkan ke dalam satu *cluster* yang sama dan data yang mempunyai karakteristik berbeda akan dikelompokkan ke dalam *cluster* lain [**8**].

Pada algoritma ini, yang menjadi pusat *cluster* dari *cluster* dinamakan *centroid*. *Centroid* merupakan titik pusat sebuah *cluster* yang dipilih secara acak dari seluruh kumpulan data. *K-Means* menyeleksi masing-masing atribut dari seluruh data dan memasukan data tersebut ke dalam salah satu *cluster* yang terbentuk berdasarkan jarak terdekat antara data dengan *centroid*. Proses ini akan dilakukan berulang kali sampai tidak ada lagi data yang berpindah antar *cluster*. Alur proses algoritma K-Means dapat dilihat pada flowchart berikut .



Gambar 2.3 Flowchart Algoritma K-Means

Langkah-langkah dari algoritma K-Means pada *flowchart* diatas adalah sebagai berikut:

1. Tentukan nilai k sebagai jumlah *cluster* yang ingin dibentuk
2. Bangkitkan nilai *centroid* awal yang ada di masing-masing *cluster*. Penentuan nilai *centroid* pada awal iterasi dilakukan secara acak.

3. Hitung jarak setiap data menggunakan metode pengukuran jarak hingga ditemukan jarak dari setiap data dengan setiap *centeroid*. Metode pengkuruan jarak yang digunakan pada penelitian ini adalah metode *Euclidian distance,* berikut rumusnya [**8**].

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ \dots\ + (x_n - y_n)^2}$$
$$= \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

(**2.1**)

Keterangan :

d(x,y) = jarak (*distance*) titik x dengan titik y

x = nilai x (x_1, x_2,…,x_n)

y = nilai y (y_1, y_2,…,y_n)

k = jumlah *Cluster*

i = jumlah data

4. Kelompokan setiap data bedasarkan jarak terdekat antara data dengan masing-masing *centroid*.
5. Tentukan posisi *centroid* baru dengan menghitung nilai rata-rata dari data yang ada didalam *centroid* yang sama. berikut rumusnya [**8**].

$$Ck = \left(\frac{1}{nk}\right)\sum d_i$$

(**2.2**)

Dimana nk adalah jumlah data dalam *cluster* k dan di adalah data yang ada didalam cluster k.

6. Kembali ke langkah 2, apabila masih ada data yang berpindah *cluster*. Proses *clustering* akan terus berulang sampai kondisi dimana data pada cluster saat ini sama dengan data pada cluster sebelumnya (sudah tidak ada lagi data yang berpindah kelompok).

2.5 Algoritma K-PSO

Algoritma K-PSO merupakan pengembangan dari algoritma K-Means. Dimana proses *clustering* K-Means di gabungkan dengan kemampuan optimasi algoritma dari *Particle swarm optimization*. Penggabungan antara PSO dengan K-Means bertujuan untuk menyelesaikan permasalahan K-means dalam pemilihan *centroid* awal yang tidak optimal. Penerapan PSO untuk *clustering* pertama kali dilakukan oleh Van der Merwe dan Engelbrecht [**4**]. Penelitian tersebut berhasil menyelesaikan permasalahan K-Means dengan menghasilkan *cluster* yang memiliki kualitas lebih baik dari algoritma K-Means.

2.5.1 *Particle Swarm Optimization*

Particle swarm optimization (PSO) merupakan algoritma yang dikembangkan oleh Kennedy dan R. Eberhart pada tahun 1995 yang terinspirasi dari perilaku pergerakan kawanan hewan seperti ikan, burung, dan kawanan serangga seperti semut rayap dan lebah [**9**]. PSO menerapkan sifat dari masing-masing individu dalam satu kelompok besar kemudian menggabungkan sifat sifat tersebut dalam menyelesaikan sebuah masalah. PSO adalah salah satu metode dari teknik kecerdasan buatan *swarm intelligence* yang berlandaskan pada perilaku kolektif (*collective beharviour*) dan dapat mengatur dirinya sendiri (*self-organizing*).

Dalam *swarm intelligence, particle swarm optimization* merupakan metode yang digunakan untuk penentuan posisi partikel terbaik. PSO menggunakan sekumpulan partikel yang bekerjasama, dimana setiap partikel merepresentasikan satu kandidat solusi untuk mengeksplorasi solusi-solusi yang memungkinkan bagi permasalahan optimasi. Masing-masing partikel diinisialisasi secara acak atau *heuristik*, kemudian partikel – partikel tersebut diperbolehkan untuk “terbang”. Pada setiap langkah optimasi. Masing-masing partikel diperbolehkan untuk mengevaluasi kemampuannya dan kemampuan dari partikel-partikel di sekitarnya. Masing-masing partikel dapat menyimpan solusi yang menghasilkan kemampuan terbaik sebagai salah satu kandidat solusi terbaik dari seluruh partikel yang dibangkitkan.

Terdapat 6 tahapan utama pada algoritma *particle swarm optimization, yaitu* [**9**] :

1. Inisialisasi populasi partikel awal

   Langkah ini merupakan tahap awal dimana jumlah dan koordinat partikel / posisi partikel ditentukan terlebih dahulu.
2. Melakukan perhitungan dan evaluasi nilai *fitness* dari setiap partikel

   Untuk memperoleh nilai fitness dibutuhkan nilai posisi setiap partikel tersebut untuk dievaluasi oleh fungsi *fitness*.
3. Menentukan *Pbest* (*personal best)*

   Pbest adalah posisi terbaik setiap partikel. Posisi ini merupakan posisi yang ditempati oleh partikel saat ini.
4. Menentukan *Gbest* (*global best*)

   Nilai *Gbest* merupakan nilai posisi partikel yang memiliki posisi paling baik dari partikel lainnya. Nilai *Gbest* ini didapat dengan membandingkan nilai *fitness* dari setiap partikel untuk kemudian koordinat posisi dari partikel yang memiliki *fitness* paling optimal tersebut akan dijadikan *Gbest*. Nilai fitness dikatakan optimal tergantung kasus yang dimiliki apakah maksimasi ataupun minimasi.
5. Menghitung nilai *velocity* (V)

   Untuk mendapatkan nilai V terbaru dapat digunakan persamaan umum kecepatan PSO untuk memperbaharui kecepatan. Rumus nilai kecepatan PSO adalah sebagai berikut [**9**].

$$V_i^{t+1} = \omega V_i^t + C_1 R_1 \left(P_i^t - X_i^t\right) + C_2 R_2 \left(P_g^t - X_i^t\right)$$

(**2.3**)

Keterangan :

V_i^t = Kecepatan partikel ke-i pada iterasi ke-t

ω = Beban partikel *(Innertia weight*)

C = Konstanta Akselarasi (*Acceralation Constans)*

R = Bilangan acak antara 0-1

X_i^t = Posisi partikel ke I pada iterasi ke-t

P_i = Solusi terbaik Lokal (*Local Best*)

P_g = Solusi terbaik Global (*Global Best*)

6. Menghitung nilai koordinat posisi terbaru.
   Memasukan hasil perhitungan nilai V dari setiap partikel ke dalam persamaan koordinat posisi PSO. Rumus Posisi PSO adalah sebagai berikut [**9**].

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (\mathbf{2.4})$$

   Keterangan :

   X_i^t = Posisi partikel ke I pada iterasi ke-t

   V_i^t = Kecepatan partikel ke-i pada iterasi ke-t

2.5.2 PSO Dalam kasus *Clustering*

Dalam kasus *clustering*, partikel pada PSO merepresetasikan nilai *centroid* dari tiap cluster yang dibentuk. Representasi partikel PSO menghasilkan sebuah matriks *centroid* yang dapat dilihat di Gambar 2.4 dengan k adalah jumlah cluster yang dibentuk dan d adalah dimensi data [**4**] [**10**].

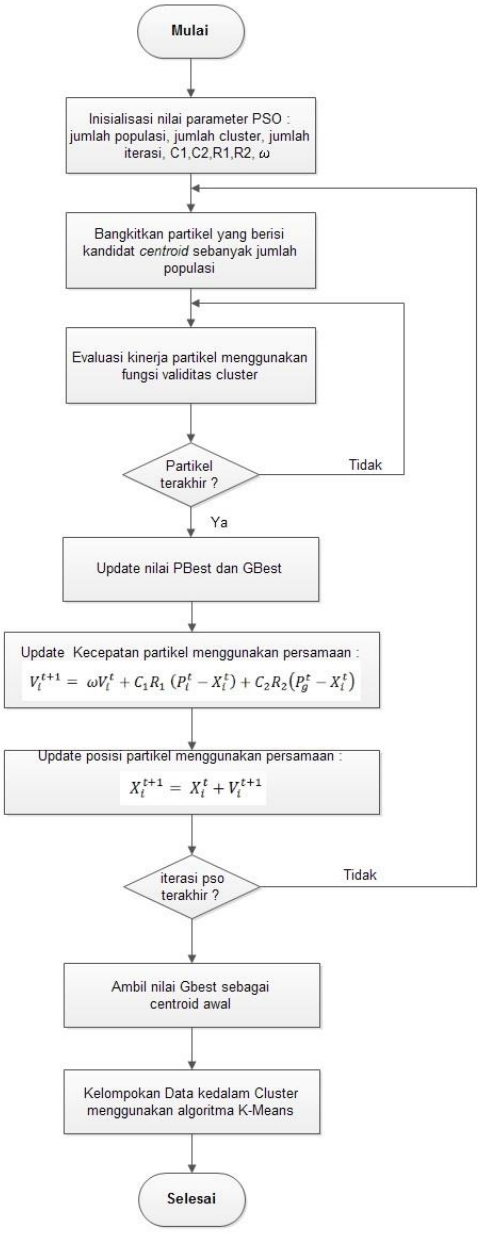
x_{11}	x_{12}	...	x_{1d}	...	x_{k1}	x_{k2}	...	x_{kd}

Gambar 2.4 Representasi Partikel Dalam Clustering

Swarm pada PSO merepresentasikan banyaknya kandidat *centroid* yang dibangkitkan. Dengan kata lain *swarm* berisi kumpulan partikel-partikel *centroid* yang dapat menjadi solusi pada sebuah permasalahan *clustering*. Untuk dapat memilih partikel solusi pada *swarm* diperlukan fungsi fitness (*fitness function*) sebagai alat untuk mengukur kebenaran dari masing-masing partikel yang dibangkitkan. Fungsi fitness Dalam kasus data clustering umumnya menggunakan rumus validasi *cluster* seperti *Davies-Bouldin Indeks, Silhoutte Index,*

Quantization error, dst. Partikel terbaik yang didapatkan dari hasil evaluasi akan menjadi *centeroid* yang digunakan pada proses *clustering*.

Algoritma K-PSO menggunakan PSO untuk menentukan *centroid* awal terbaik dari *dataset*. Selanjutnya menjalankan proses algoritma K-Means dengan menggunakan *centroid* yang dihasilkan oleh PSO sebagai nilai *centroid* awal. Alur dari algoritma K-PSO dapat dilihat pada flowchart di Gambar 2.5.



Gambar 2.5 Flowchart algoritma K-PSO

2.6 Algoritma HA-KMeans

Algoritma HA-KMeans merupakan pengembangan dari algoritma K-Means, Dimana proses *clustering* dari algoritma K-means di gabungkan dengan kemampuan dalam menduga jumlah cluster dari algoritma *Hierarchical Agglomerative Clustering*. Gabungan algoritma HAC dengan K-Means bertujuan untuk menyelesaikan permasalahan K-means yang tidak mampu menduga jumlah *cluster* terbaik secara otomatis dari suatu *dataset*. Penelitian yang memanfaatkan HAC untuk *clustering* pernah dilakukan oleh oleh Jung, dkk [**7**] dan Wirawan, dkk [**11**]. Pada penelitian mereka, Proses *clustering* dimulai dengan menjalankan proses dari HAC. Kemudian hasil jumlah *cluster* dari HAC menjadi masukan untuk nilai jumlah *cluster* pada proses K-Means. Algoritma yang menggunakan HAC untuk menduga jumlah *cluster* ternyata mampu menghasilkan kualitas hasil *clustering* yang lebih baik dari K-Means.

2.6.1 *Hierarchical Agglomerative Clustering*

Hierarchical Agglomerative Clustering (HAC) merupakan teknik *clustering* berbasis hierarki dengan pendekatan *bottom-up*, dimana proses *clustering*nya dimulai dari masing-masing data sebagai satu buah *cluster*, kemudian secara rekursif mencari *cluster* terdekat sebagai pasangan untuk bergabung menjadi sebuah *cluster* yang lebih besar. Proses tersebut terus diulang sampai tersisa satu cluster besar sehingga tampak bergerak ke atas membentuk hierarki [**12**]. *Clustering* berbasis hierarki sering ditampilkan dalam bentuk grafis menggunakan diagram yang mirip pohon yang disebut dengan *dendogram*. Dendogram menampilkan hubungan *cluster* dengan *subcluster*-nya bedasarkan urutan proses penggabungan *cluster*.

Pengelompokkan data menggunakan HAC diawali dengan merepresentasikan setiap data pada *dataset* sebagai satu kelompok, kemudian dilakukan perhitungan jarak antar kelompok [**13**]. Setelah itu dua kelompok yang memiliki jarak terdekat akan digabungkan menjadi sebuah kelompok baru. Jarak antara kelompok yang baru dengan kelompok lain selanjutnya dihitung menggunakan salah satu metode perhitungan jarak antar kelompok (*single linkage*, *complete linkage*, *average*

linkage, dll.). Proses perhitungan jarak dan penggabungan dua kelompok dilakukan secara iterative hingga hanya tersisa satu buah kelompok yang berisi seluruh objek.

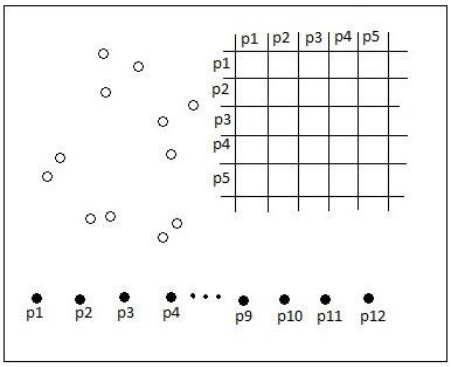
Perhitungan jarak antar kelompok menjadi faktor penting dalam metode HAC. Proses tersebut dilakukan untuk meminimalkan jarak antar objek dalam satu kelompok (*intra-cluster distance*) dan memaksimalkan jarak antara objek dalam satu kelompok dengan objek dalam kelompok lain (*inter-cluster distance*) [**12**]. Dengan kata lain, perhitungan jarak antara dua kelompok perlu dilakukan untuk mengetahui kemiripan antar kelompok tersebut.

2.6.2 HAC dalam kasus *Clustering*

Pada penyelesaian kasus data *clustering*, proses HAC digunakan untuk menentukan jumlah pembagian *cluster* dari suatu *dataset*. Tahapan algoritma HAC dalam *clustering* adalah sebagai berikut [**14**] :

1. Kondisi awal

Tahap awal dimulai dengan membentuk *cluster* dari setiap data dan membuat sebuah *proximity matrix* (matriks kedekatan). *Proximity matrix* yaitu matriks yang berisi jarak antar data.



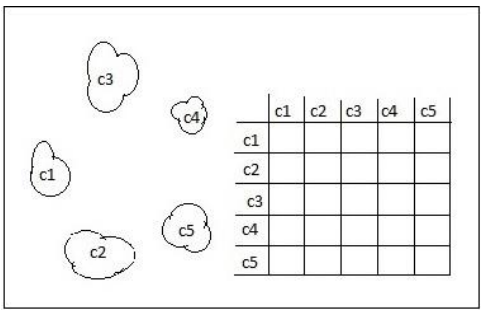
Gambar 2.6 Titik-titik individual dan *matriks* kedekatan

Seperti yang terlihat di Gambar 2.6. Metode ini dimulai dengan meletakan data sebagai sebuah *cluster* tersendiri. Misalkan terdapat 12 data mulai dari p1 hingga p12, berate jumlah *cluster* awal yang terbentuk adalah 12 *cluster*. *Matriks*

kedekatan diperlukan untuk menghitung jarak kedekatan antar data yaitu jarak antara p1 dengan p2, p1 dengan p3 , dan seterusnya hingga jarak ke n.

2. Kondisi *Intermediate*

Tahapan selanjutnya adalah mengabungkan dua *cluster* terdekat. Penggabungan ini dilakukan hingga hanya ada satu cluster yang tersisa.

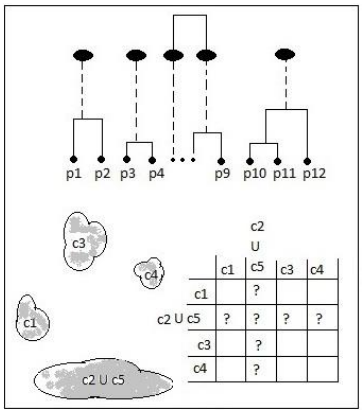


Gambar 2.7 Langkah penggabungan

Gambar 2.7 Memperlihatkan kondisi setelah beberapa langkah penggabungan. Pada gambar tersebut dimisalkan diperoleh beberapa *cluster* yaitu C1, C2, C3, C4 dan, C5.

3. Kondisi setelah penggabungan

Pada tahap ini dilakukan peng-*update*-an nilai yang terdapa didalam *proximity matrix.*



Gambar 2.8 Kondisi setelah penggabungan

Misalkan dari lima cluster yang terbentuk tersebut, terjadi lagi penggabungan dua *cluster* terdekat yaitu C2 dan C5 seperti yang terlihat pada gambar 2.8 Setiap terjadi proses penggabungan dua cluster maka dilakukan juga pembaharuan terhadap *matriks* kedekatan, dengan cara menghitung kembali jarak kedekatan antara *cluster* baru yang terbentuk dengan *cluster* lama yang tersisa.

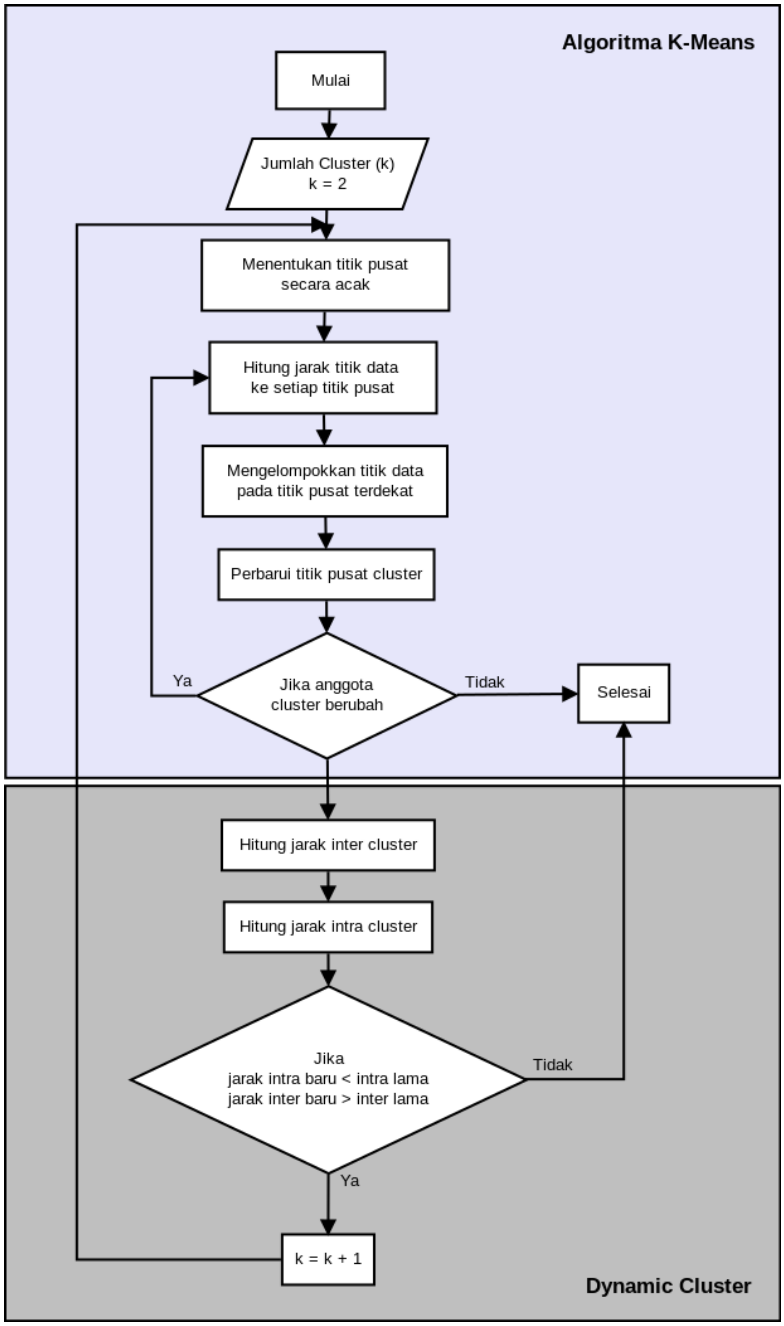
4. *Complete Linkage*

Complete linkage disebut juga maximum link dimana jarak antara dua buah *cluster* dihitung dari jarak terjauh antara anggota *cluster* satu dengan *cluster* yang kedua. Complete linkage merupakan salah satu metode yang dapat digunakan untuk menentukan jarak antar *cluster* pada *hierarchical agglomerative.* Pada metode *complete linkage*, proses pengkuran jarak antara dua *cluster* ditentukan oleh jarak terjauh antara dua objek dalam *cluster* yang berbeda [**13**] . *Complete linkage* membandingkan objek antar *cluster* yang paling berbeda di setiap iterasi. Setelah perhitungan jarak dilakukan menggunakan *complete linkage*, dua *cluster* yang memiliki jarak terkecil kemudian digabungkan. Persamaan berikut merupakan persamaan untuk menentukan jarak antara dua *cluster* dengan metode *complete linkage*, dimana $\delta(C_x, C_y)$ adalah jarak antara dua objek *x* dan *y* . Sedangkan C_x dan C_y merupakan dua *cluster* yang dibandingkan.

$$\delta(C_x, C_y) = \max\{\delta(x, y) | x \epsilon C_x, y \epsilon C_y\} \quad \textbf{(2.5)}$$

2.7 Algoritma *Dynamic Clustering*

Dynamic *Clustering* adalah algoritma yang dikembangkan untuk melakukan optimasi pada K-Means. Dynamic *Clustering* menambahkan algoritmanya dalam membantu K-Means untuk menduga jumlah *cluster* dari suatu *dataset*, sehingga algoritma ini dapat menemukan jumlah *cluster* yang optimal. Proses dari algoritma Dynamic *Clustering* diperlihatkan di Gambar 2.9 [**15**].



Gambar 2.9 Algoritma *Dynamic Clustering*

Dari Gambar 2.9 terlihat bahwa proses algoritma *Dynamic Clustering* bekerja setelah proses K-Means. Algoritma *Dynamic Clustering* akan menghitung jarak *inter* dan jarak *intra* dari proses sebelumnya, jika jarak *intra* baru lebih kecil dari jarak *intra* lama dan jarak *inter* baru lebih besar dari jarak *inter* lama maka algoritma *Dynamic Clustering* akan menambahkan nilai satu pada jumlah *cluster*

sebelumnya. Proses ini akan dilakukan secara berulang hingga algoritma *Dynamic Clustering* menemukan jarak *intra* yang terkecil dan jarak *inter* yang terbesar.

Kelebihan dari algoritma Dynamic Cluster adalah mampu menduga jumlah cluster terbaik dari suatu *dataset* dibandingkan dengan algoritma K-Means. Akan tetapi algoritma *Dynamic Cluster* masih mengadopsi kelemahan dari K-Means dalam pemilihan titik pusat awalyang dilakukan secara acak, sehingga tidak selalu menghasilkan kualitas hasil *clustering* yang baik.

2.8 Metode Validasi Cluster

Setelah kelompok-kelompok *cluster* terbentuk, timbul pertanyaan apakah *cluster* yang terbentuk tersebut valid atau tidak. Untuk menjawab pertanyaan tersebut dibutuhkan sebuah alat ukur yang dapat digunakan dalam menilai keakuratan *cluster* dari hasil proses *clustering*. Metode validasi *cluster* adalah sebuah teknik yang digunakan untuk mengetahui seberapa bagus kualitas hasil *cluster* yang telah diperoleh dari proses *clustering* [**12**]. Metode yang digunakan untuk menentukan ukuran validitas *cluster* pada penelitian ini adalah *Davies-bouldin Index.*

2.8.1 Davies-Bouldin *Index*

David L.Davies dan Donald W. Bouldin memperkenalkan sebuah metode untuk mengevaluasi *cluster* yang diberi nama sesuai dengan nama mereka, yaitu Davis-Bouldin *Index* (DBI) [**12**]. DBI merupakan salah satu teknik yang dapat digunakan untuk mengukur validitas *cluster* pada metode *clutestering* berbasis partisi yang didasarkan pada nilai kohesi dan separasi. Nilai kohesi didefinisikan sebagai jumlah dari kedekatan data terhadap *centroid* dari *cluster* yang diikuti. Sedangkan nilai separasi didasarkan pada jarak antar data dengan *centroid* diluar *cluster* yang diikuti.

Dalam DBI *Sum of square within cluster* (SSW) merupakan persamaan yang digunakan untuk mengetahui nilai dari *matriks* kohesi dalam sebuah *cluster* ke-i. Nilai SSW dapat dihitung dengan rumus sebagai berikut [**12**]:

$$SSW_i = \frac{1}{mi}\sum_{j=i}^{mi} d(x_j, c_i)$$

(**2.6**)

Sedangkan *Sum of square between cluster* (SSB) merupakan persamaan yang digunakan untuk mengetahui nilai separasi antar *cluster*. Nilai SSB dapat dihitung menggunakan rumus sebagai berikut [**12**] :

$$SSB_{i,j} = d(c_i, c_j)$$

(**2.7**)

Setelah nilai kohesi dan separasi diperoleh, kemudian dilakukan pengukuran rasio (*Rij*) untuk mengetahui nilai perbandingan antara *cluster* ke-i dan *cluster* ke-j. *Cluster* yang baik adalah *cluster* yang memiliki nilai kohesi sekecil mungkin dan separasi yang sebesar mungkin. Nilai rasio dihitung menggunakan rumus sebagai berikut [**12**]:

$$R_{i,j} = \frac{SSW_i + SSW_j}{SSB_{i,j}}$$

(**2.8**)

Nilai rasio yang diperoleh tersebut digunakan untuk mencari nilai *davies-bouldin index* (DBI) dari rumus berikut [**12**]:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} max_{i \neq j}(R_{i,j})$$

(**2.9**)

Hasil dari rumus tersebut merupakan nilai yang menjadi ukuran validitas dari *cluster* yang diuji. Semakin kecil nilai DBI yang diperoleh (non-negatif >= 0), maka semakin baik *cluster* yang diperoleh dari metode *clustering* yang digunakan [**12**].